Opacity from Observers with a Bounded Memory

Andrew Wintenberg, *Student Member, IEEE*, Stéphane Lafortune, *Fellow, IEEE*, and Necmiye Ozay, *Senior Member, IEEE*

***Abstract*— Opacity is an information-flow property capturing privacy from observers that are aware of a system's dynamics. The potential for an observer with perfect recall to reason about long histories of the system poses a challenge for opacity verification. In this paper, we address this challenge by proposing a new notion of opacity over automata, called bounded memory opacity, with respect to an observer with a bounded memory. We show that verifying this weaker notion of opacity has reduced computational complexity compared to general opacity (co-NP vs. PSPACE). Furthermore, we present a corresponding verification algorithm using an encoding to the Boolean satisfiability problem (SAT). We demonstrate this approach on randomly generated automata as well as a web server load-hiding example.**

***Index Terms*— Opacity, Privacy, Discrete Event Systems, Computational Complexity**

I. Introduction

RECENTLY, we have become increasingly reliant upon cyber-physical systems (CPS) which integrate physical processes across cyber-networks. Many of these systems, from the smart grid to medical devices, communicate sensitive information which is vulnerable to eavesdropping. As leaking this information can lead to serious harm to both the system and its users, such systems are subject to strict privacy and security requirements. Particularly in the areas of CPS and discrete event systems (DES), such privacy requirements have been modeled using the information-flow property of *opacity*. Opacity captures the inability of a passive eavesdropper or observer to deduce some *secret behavior* using their knowledge of the system's dynamics.

To model the diverse privacy requirements encountered in practice, many notions of opacity have been proposed. These notions are characterized by the type of secret behaviors considered and the capabilities of the observer. For example, the secrets considered in current-state opacity [1] and initial-state opacity [2] are the current and initial state, respectively. Likewise language-based opacity (LBO) [3] considers behavior within a given language to be secret. These notions along with many others assume that the observer has partial observation of the system but impose no other constraints. In particular, it is implicitly assumed that observers have *perfect recall*, always deducing a secret correctly if possible. While this approach provides strong theoretical guarantees of privacy, it presents a number of challenges in practice. First, an observer may have limited computational resources to perform deduction, especially in an embedded setting. Second, the computational resources required to verify these privacy guarantees may be prohibitive as we must consider every possible way information may leak. Indeed, the aforementioned notions of opacity over automata are all readily transformed into one another [4]–[6], and the common verification problem is known to be PSPACE-complete [7]. This high complexity is observed in the poor exponential scalability of verification algorithms in practice.

In this work, we address these challenges by proposing a new notion of opacity reflecting an additional constraint on the observer: the amount of memory available to them. We characterize opacity from the observer's point of view, modeling their deductions with a nondeterministic automaton that marks observations deemed secret. In this form, we can impose a bound $k \in \mathbb{N}$ on the size of this automaton, representing the memory available to the observer. Our proposed notion of k-bounded memory opacity (k-BMO) requires that no such automaton exists. We establish basic properties of this notion, including that the verification problem is co-NP-complete, reduced from the PSPACE-completeness for LBO. In addition to these results, we develop a verification approach using an encoding into the Boolean satisfiability problem (SAT) and demonstrate it on a number of examples.

II. Preliminaries

A. Automata

In this work, we model systems with nondeterministic finite automata (NFA). For a detailed introduction to automata in the context of DES, see [8]. The set of strings over a finite set of events Σ is denoted by Σ^*, including the *empty string* ϵ. For a string t, $|t|$ denotes its length while for a set Q, $|Q|$ denotes its cardinality. A *nondeterministic finite automaton* (NFA) over Σ is a tuple $G = (Q, \Sigma, \delta, q_0, Q_m)$ with a finite set of states Q, transition relation $\delta \subseteq Q \times \Sigma \times Q$, initial state $q_0 \in Q$, and a set of marked states $Q_m \subseteq Q$. The transition relation can be extended inductively to the domain $Q \times \Sigma^* \times Q$. The *language* of G reaching states Q' is the set $\mathcal{L}_{Q'}(G) = \{t \in \Sigma^* \mid \exists q \in$

This work was supported by US National Science Foundation awards CNS-1837680 and ECCS-2144416, as well as a sponsored research award from Cisco Research.

A. Wintenberg, S. Lafortune, and N. Ozay are at the University of Michigan, Ann Arbor, USA. <awintenb,stephane,necmiye>@umich.edu

$Q'.\ (q_0, t, q) \in \delta\}$. The *language generated* by G is the set $\mathcal{L}(G) = \mathcal{L}_Q(G)$ while the *language marked* by G is the set $\mathcal{L}_m(G) = \mathcal{L}_{Q_m}(G)$. We say an NFA G is a deterministic finite automaton (DFA) if $|\{q' \mid (q, \sigma, q') \in \delta\}| \leq 1$ for each state q and event σ.

Given a subset of *observable events* $\Sigma_o \subseteq \Sigma$, the *natural projection* $\mathrm{P} : \Sigma^* \to \Sigma_o$ is defined recursively for strings t by $\mathrm{P}(t) = \epsilon$ if $t = \epsilon$, $\mathrm{P}(t\sigma) = \mathrm{P}(t)\sigma$ for $\sigma \in \Sigma_o$, and $\mathrm{P}(t\sigma) = \mathrm{P}(t)$ for $\sigma \in \Sigma \setminus \Sigma_o$. We denote the preimage of this map, called the *inverse projection*, by P^{-1}. The *observer* of G is a DFA G_{obs} over Σ_o with states in 2^Q such that $\mathcal{L}_m(G_{obs}) = \mathrm{P}(\mathcal{L}_m(G))$. This is also referred to as the powerset construction or determinization. Given NFAs $G = (Q, \Sigma, \delta, q_0, Q_m)$ and $A = (Q_A, \Sigma_o, \delta_A, q_{A,0}, Q_{A,m})$, their *parallel composition* is an automaton $A||G$ with states $Q_A \times Q_G$ such that $\mathcal{L}_m(A||G) = \mathrm{P}^{-1}(\mathcal{L}_m(A)) \cap \mathcal{L}_m(G)$. Given NFAs G and A, we can also construct automata marking the complement language $\Sigma^* \setminus \mathcal{L}_m(G)$ and the union language $\mathcal{L}_m(G) \cup \mathcal{L}_m(A)$.

B. Complexity Theory

We now briefly review concepts from the study of complexity theory needed to characterize our results. For more information, consult a standard reference such as [9]. A *decision problem* relates the inputs of an algorithm to a corresponding yes or no answer as output. Complexity theory classifies these problems according to the time or space needed to solve them in a given model of computation, such as Turing machines. For example, the classes P and PSPACE denote problems that can be solved by a deterministic machine in polynomial time and space, respectively, as a function of the input size. Likewise NP denotes problems solved by nondeterministic machines in polynomial time, i.e., problems for which there exist *certificates* proving the correctness of *yes* answers in polynomial time. Similarly, co-NP denotes the complement of NP, i.e., problems for which there exist certificates proving the correctness of *no* answers in polynomial time. While it is known that $\mathsf{NP}, \mathsf{co-NP} \subseteq \mathsf{PSPACE}$, it is unknown if this inclusion is strict. A problem in NP is said to be NP-*complete* if it is as hard as any other problem in NP, i.e., there is a polynomial time algorithm reducing any NP problem to it. A similar definition is made for PSPACE-completeness.

For example, the Boolean satisfiability problem (SAT), which asks if a given Boolean formula can be satisfied, is known to be NP-complete. Indeed a satisfying assignment is a certificate for satisfiability. Similarly, the complement problem asking if a formula is unsatisfiable is co-NP-complete. In addition the MAX-SAT problem of maximizing the sum of weights assigned to satisfied clauses of the formula is NP-complete as well. While these problems cannot be solved in polynomial time (unless P = NP), advanced heuristics have resulted in solvers for SAT and MAX-SAT that are sufficient to solve many problems in practice.

III. Opacity Formulation

In this section, we present an alternative characterization of opacity from the viewpoint of the observer which we use to propose a new notion of opacity against observers with bounded memory. We consider a system modeled by an NFA $G = (Q, \Sigma, \delta, q_0, Q_m)$ generating the system's behavior $\mathcal{L}(G)$, which is divided into two classes, *secret* and *nonsecret*. We assume the automaton G marks the nonsecret behavior $\mathcal{L}_m(G)$. In addition we assume the system's behavior is observed through the projection P of observable events $\Sigma_o \subseteq \Sigma$. Then, *opacity* requires that an observer of this system cannot deduce when a secret behavior has occurred. Importantly, we assume the observer knows the model of the system G and thus deduces a secret if their observations are not consistent with nonsecret behavior. We call such observations $t \in \mathrm{P}(\mathcal{L}(G))$ *violating* i.e., if $\mathrm{P}(t) \notin \mathrm{P}(\mathcal{L}_m(G))$. This motivates the following definition.

Definition 3.1 (Language-Based Opacity (LBO)): An NFA G is said to be *opaque* or simply LBO for Σ_o if

$$\mathrm{P}(\mathcal{L}(G)) \subseteq \mathrm{P}(\mathcal{L}_m(G))\,. \tag{1}$$

Many existing notions of opacity may be formulated in this way as shown in [4], [5]. For example, current-state opacity requires visits to so-called *secret states* to be hidden. Given an automaton $G = (Q, \Sigma, \delta, q_0, Q_m)$ with secret states $Q_S \subseteq Q$, if $Q_m = Q \setminus Q_S$, the current-state opacity of G is equivalent to the opacity of G as in Definition 3.1.

An observer of the system trying to deduce if an observation t was violating considers a version of the regular language acceptance problem, i.e., is t an element of $\mathrm{P}(\mathcal{L}_m(G))$? We will formulate a new notion of opacity capturing a restriction on the algorithms the observer uses to solve this problem, namely the amount of available *memory states*. As noted in [10], these algorithms can be viewed as passive *attacks* on the system which do not alter the system's behavior. We adopt this terminology, and model these attacks with an NFA $A = (Q_A, \Sigma_o, \delta_A, q_{A,0}, Q_{A,m})$ which tracks observations of the system G and marks some which are violating. We can use this notion of attacks to provide an alternative characterization of opacity.

Proposition 3.1: The system NFA G is *not* LBO if and only if there exists an attack NFA A with the following properties

1) *Correctness*: The attack marks only violating observations
$$\mathcal{L}_m(A) \cap \mathrm{P}(\mathcal{L}_m(G)) = \emptyset\,. \tag{2}$$
2) *Nontriviality*: The attack marks some observation
$$\mathcal{L}_m(A) \cap \mathrm{P}(\mathcal{L}(G)) \neq \emptyset\,. \tag{3}$$

Proof: If there exists a correct and nontrivial attack A, then there exists a string $t \in \mathcal{L}_m(A)$ contained by $\mathrm{P}(\mathcal{L}(G))$ but not by $\mathrm{P}(\mathcal{L}_m(G))$. Hence t is violating and thus G is not LBO. Conversely if G is not LBO, then there exists such a string that is violating. Thus the attack A marking this single string is necessarily correct and nontrivial. ■

Note that we do not require attacks to deduce *all* violating observations, just one. Indeed, due to nondeterminism an attack can be correct and nontrivial even if it rejects a violating observation on one run but accepts it on another. Because of this, the proposed memory bound is not directly related to the standard notion of space complexity for the regular language acceptance problem. Alternatively, the smallest attack which deduces every violating observation can be computed by

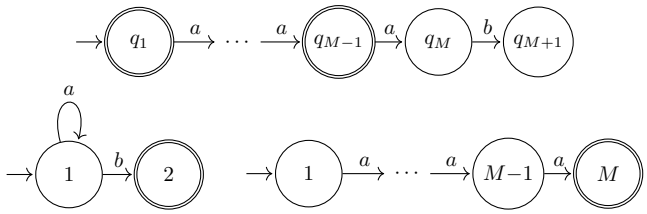

Fig. 1. A system NFA $\boldsymbol{G}$ (top) and corresponding attack NFAs $\boldsymbol{A}$ (bottom left) and $\boldsymbol{A'}$ (bottom right). States $\boldsymbol{q_M}$ and $\boldsymbol{q_{M+1}}$ act as secret states in $\boldsymbol{G}$ as in Example 3.1.

applying state minimization to the complement of the observer of G. The number of states in an attack automaton A represents the number of memory states utilized by a corresponding nondeterministic deduction algorithm. We can then define a notion of opacity capturing a restriction on the memory available to an observer as a bound on the number of such states.

Definition 3.2: Given $k \in \mathbb{N}$ and $\Sigma_o \subseteq \Sigma$, we say the system NFA G is *k-bounded memory opaque* (or k-BMO) if there is *no* correct and nontrivial attack with k states.

A. Properties of Bounded Memory Opacity

We now observe a number of simple properties about this notion of opacity.

Proposition 3.2: Consider an NFA G and bound $k \in \mathbb{N}$.

1) If G is $(k+1)$-BMO, then G is k-BMO.
2) If G is LBO, then G is k-BMO.
3) If G is $2^{|Q|}$-BMO, then G is LBO.

Proof:

1) We can add unreachable states to an attack without altering its correctness or nontriviality.
2) If G is LBO, i.e., $\mathrm{P}(\mathcal{L}(G)) = \mathrm{P}(\mathcal{L}_m(G))$, then any correct attack cannot be nontrivial.
3) If G is not LBO, then its observer with marked and unmarked states swapped is a correct and nontrivial attack with size at most $2^{|Q|}$.

■

In general, we are interested in the smallest bound k for which a system is k-BMO or equivalently, the size of *minimal* attacks. While we can always construct attacks recognizing the smallest violating observation or attacks that are deterministic, the following examples show that such attacks may not be minimal.

Example 3.1: Consider the DFA G depicted in Figure 1 with $M+1$ states and all events observable. The only violating strings are $t_1 = a^{M-1}$ and $t_2 = a^{M-1}b$. The attack A depicted in Figure 1 is minimal, marking the string t_2 in composition with G. Furthermore, it is clear that any correct attack marking t_1, such as the one depicted in Figure 1, must have at least M states, counting the occurrences of a. So the shortest violation of opacity may not always correspond to a minimal attack.

Example 3.2: Consider the nondeterministic attack A with 4 states depicted in Figure 2. We will construct a system G for which this attack is minimal. Let G' denote the complement of the observer of A with all unmarked states removed. As

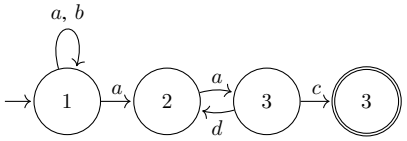

Fig. 2. A nondeterministic attack $\boldsymbol{A}$ from which we construct the system $\boldsymbol{G}$ in Example 3.2.

a result $\mathcal{L}_m(G') = \mathcal{L}(G')$ contains strings whose prefixes are not in $\mathcal{L}_m(A)$, i.e., $\mathcal{L}(G') \cap \mathcal{L}_m(A) = \emptyset$. Let G denote the union NFA construction for G' and an automaton generating the secret string $abaadac \in \mathcal{L}_m(A)$ and marking its strict prefixes. We do not depict G here due to space constraints. By construction, we see A is correct and nontrivial, yet applying the verification method developed later in Section IV, we can determine that there is no deterministic attack with size 4. So in general there may be no minimal attack that is deterministic.

IV. Verification and Falsification

In this section, we discuss the problem of verifying k-BMO. We show this problem is co-NP-complete, or equivalently, that falsifying k-BMO by synthesizing an attack is NP-complete. In addition we present a verification approach based upon an encoding into SAT. Formally, we state the problem of verifying k-BMO as follows.

Problem 1: Determine if a system modeled by an NFA G is k-BMO for a given $k \in \mathbb{N}$ and observable events Σ_o.

A. Verifying k-BMO is co-NP

It is well-known that the complexity of verifying opacity in general is PSPACE-complete [7]. By relaxing the requirement that an observer never deduces a secret, i.e., LBO, to the proposed notion of k-BMO, we reduce the complexity of the verification problem.

Theorem 4.1: Verifying k-BMO is co-NP.

Proof: Let G be the system NFA, Σ_o a set of observable events, and $k \in \mathbb{N}$ a bound represented in unary which all serve as input to the problem. Let n and m denote the number of states and events of G, respectively. To show the problem is co-NP, it suffices to show that we can check if an attack A with size k (serving as a certificate) is correct and nontrivial in polynomial time. Using properties of the parallel composition, A is correct if the following equivalent conditions hold

$$\mathcal{L}_m(A) \cap \mathrm{P}(\mathcal{L}_m(G)) = \emptyset \;\Leftrightarrow\; \mathcal{L}_{Q_{A,m} \times Q_m}(A||G) = \emptyset\,. \quad (4)$$

Likewise, A is nontrivial if the following equivalent conditions hold

$$\mathcal{L}_m(A) \cap \mathrm{P}(\mathcal{L}(G)) \neq \emptyset \;\Leftrightarrow\; \mathcal{L}_{Q_{A,m} \times Q}(A||G) \neq \emptyset\,. \quad (5)$$

The language of an NFA is nonempty if and only if its marked states are reachable from initial ones, which can be checked using breadth-first search in linear time in the number of edges. So both of these conditions can be checked over $A||G$ with time $O(|Q_A|^2 \cdot |Q|^2 \cdot |\Sigma|) = O(k^2 n^2 m)$. ■

While we show next that this problem is co-NP-complete, the result of Theorem 4.1 is significant as such problems can

often be solved in practice with SAT solvers. To demonstrate this, we develop a SAT encoding for verification in Section IV-C whose performance is evaluated in Section V.

Remark 1: While we have proposed a bound on the memory of a potential attacker, one may instead consider verifying opacity over strings with a bounded length, similar to the concept of *bounded-model checking* [11]. While such methods can be very efficient using symbolic techniques [12], it is not immediately clear how long the strings considered must be in order to achieve some privacy requirement. As demonstrated in Figure 1, there may a simple attack to deduce a secret occurred while the shortest violating string is arbitrarily long (bounded by the number of states in the system). However, we can note for any attack A on the system G, a minimal string marked by the attack will only visit the states of the composition $A||G$ at most once. Hence with $k = |A|$ and $n = |G|$, if there are no strings violating opacity with length $kn - 1$, the system is k-BMO.

B. Verifying k-BMO is co-NP-Complete

To show that verifying k-BMO is co-NP-complete, we adapt the proof of PSPACE-completeness for verifying LBO [13]. This proof constructs a reduction from the universality problem which asks if an NFA G marks every string, i.e., $\Sigma^* \subseteq \mathcal{L}_m(G)$? Without loss of generality, we may assume that $\mathcal{L}(G) = \Sigma^*$ and all events are observable in which case, G is LBO if and only if it is not universal. This completes the reduction to the universality problem which is known to be PSPACE-complete [14]. To show the NP-completeness of our problem, we consider a variant of the universality problem over bounded strings. The *bounded nonuniversality* problem asks for an NFA G and bound $n \in \mathbb{N}$, does G not mark all strings of length at most n, i.e. $\Sigma^{\leq n} \not\subseteq \mathcal{L}_m(G)$? We now present a reduction from falsifying k-BMO to the bounded nonuniversality problem which is known to be NP-complete [15].

Theorem 4.2: Verifying k-BMO is co-NP-complete.

Proof: Consider an NFA G' and bound $n \in \mathbb{N}$ represented in unary which serve as inputs to the problem. Let $G_{\leq n}$ denote an automaton with $n+1$ states generating all strings with length at most n and marking none of them. Let G be the union automaton of G' and $G_{\leq n}$ restricted to strings of length n so $\mathcal{L}(G) = \Sigma^{\leq n}$ and $\mathcal{L}_m(G) = \mathcal{L}(G') \cap \Sigma^{\leq n}$. By construction, G is nonuniversal with bound n if and only if there exists a string $t \in \mathcal{L}(G)$ but $t \notin \mathcal{L}_m(G)$, i.e., t is violating. As we can construct an attack with $n+1$ states that only marks t, we see G' is nonuniversal with bound n if and only if G is k-BMO for $k = n+1$. As G may be constructed in polynomial time, this procedure describes a reduction from the bounded nonuniversality problem to falsifying bounded memory opacity. ∎

C. Verification using SAT

In order to verify k-BMO effectively, we can express it as a Boolean satisfiability problem. That is we can encode an attack A with propositional variables and develop constraints modeling correctness and nontriviality. Formally given the fixed system NFA G, we consider an attack A with states $Q_A = \{0, \cdots, k-1\}$. Without loss of generality, we assume the initial state is 0 and that the attack has a single marked state given by $k-1$. We introduce the variables $\tau_A(q_A, \sigma, q'_A)$ meaning the corresponding transition is present in A, i.e., $(q_A, \sigma, q'_A) \in \delta_A$. From equations (4)-(5), correctness and nontriviality of A correspond to language emptiness/nonemptiness in the composition $A||G$. In order to encode this composition, we encode the observability of an event $\sigma \in \Sigma$ with a formula $O(\sigma)$. Then the presence of a transition in the composition from (q_A, q) to (q'_A, q') over event σ where $(q, \sigma, q') \in \delta$ is given by the formula $\tau(q_A, q, \sigma, q'_A, q')$ defined by

$$(\neg O(\sigma) \wedge (q_A = q'_A)) \vee (O(\sigma) \wedge \tau_A(q_A, \sigma, q'_A)) \,. \quad (6)$$

To encode language emptiness, we recall that the language marked by an automaton is empty if and only if its marked states are not reachable from the initial state in the underlying graph. We can represent reachability in the composition with the variables $R(q_A, q)$, whose truth indicates that $(q_A, q) \in Q_A \times Q$ is reachable from the initial state $(q_{A,0}, q_0)$. To encode reachability in SAT, we use constraints similar to [16] based upon acyclicity over auxiliary variables T. These constraints require the initial state to be reachable, i.e., $R(q_{A,0}, q_0)$, and for all other states (q_A, q) that

$$R(q'_A, q') \leftarrow \bigvee_{\substack{\sigma \in \Sigma, q_A \in Q_A \\ (q, \sigma, q') \in \delta}} R(q_A) \wedge \tau(q_A, q, \sigma, q'_A, q') \quad (7)$$

$$R(q'_A, q') \rightarrow \bigvee_{\substack{\sigma \in \Sigma, q_A \in Q_A \\ (q, \sigma, q') \in \delta}} R(q_A) \wedge \tau(q_A, q, \sigma, q'_A, q') \wedge T(q_A, q, q'_A, q') \quad (8)$$

$$\text{Acyclic}(T) \,. \quad (9)$$

Constraint (7) ensures R is true for reachable states while constraints (8)-(9) ensure R is true only for reachable states. Here, the constraint $\text{Acyclic}(T)$ denotes a formula that is satisfied when the graph over nodes $Q_A \times Q$ with edges encoded by T is acyclic. We can think of this graph as a spanning tree of the reachable set rooted at the initial state. Critically, this formulation for reachability results in a total number of constraints that is linear in the number of transitions in the composed automaton (viewing acyclicity as a single constraint which is natively supported by solvers like [16]). From equations (4)-(5), we see the encoded attack is correct and nontrivial if the following constraint is satisfied

$$\varphi_{Q_m} = \bigwedge_{q \in Q_m} \neg R(k-1, q) \wedge \bigvee_{q \in Q \setminus Q_m} R(k-1, q), \quad (10)$$

where $k-1$ is the marked state of A. Then there exists an attack A encoded by these variables that is correct and nontrivial, i.e., G is not k-BMO, if and only the constraints (6)-(10) are satisfiable. Furthermore, the total number of constraints is $O(k^2 n^2 m)$ where $n = |Q|$ and $m = |\Sigma|$. This formulation of verification as a constraint satisfaction problem has many advantages. In particular, it is easy to incorporate extensions or additional constraints on the attacks as demonstrated in the following section.

V. Experiments and Examples

In this section, we investigate the performance of the proposed SAT encoding for verifying k-BMO. We first demonstrate its superior scalability in comparison to a standard approach for verifying LBO on randomly generated automata. We then present an example showing how the SAT encoding can be easily extended to solve more general problems. This example system models server load-balancing with quantitative constraints on observations available to an attacker.

A. Comparing Opacity Verification Methods

We compare an implementation[1] of the proposed SAT encoding for verifying k-BMO with a standard method for verifying LBO based upon constructing the observer (the NFA G is LBO if all states in its observer are marked). In particular we encode the constraints for verifying k-BMO developed in Section IV-C into the solver GraphSAT [16] which natively supports the acyclicity constraint (9). We evaluate the runtime of both implementations on randomly generated automata. For a given number of states n and a fixed number of events $m = 10$, transitions are included in the automata independently with a fixed probability. The probability is selected such that the expected number of transitions is $2nm$ where the exponential blowup of the observer construction is encountered [17]. We do this to demonstrate for a fixed bound k that the proposed method performs well in the worst-case scenario for verifying LBO.

The resulting runtimes for verification were averaged over 30 instances[2] for each size n ranging from 5 to 17. As the absolute runtimes are sensitive to details of each implementation, we depict the verification runtimes as a percentage of the time to verify LBO in Figure 3. We observe that for a fixed k, this ratio for verifying k-BMO decreases steadily indicating that the proposed method, while exponential itself, scales exponentially slower with automata size than the observer construction for LBO. While there are more efficient approaches to verify LBO such as modular [18] or antichain-based methods [19], it is likely similar trends will exist in comparison to verifying k-BMO due to the different complexity classes.

B. Server Load Hiding

Many cyber-physical systems operate, in part, over the public Internet, using remote servers to offer critical services. Security for these critical services is often based on *location hiding*, e.g., hiding the true address of a server behind a network of proxies [20]. While this can provide protection against distributed denial of service (DDoS) attacks, resourceful attackers can learn the structure of simple, static networks to bypass these measures [21], [22]. When the server location cannot be hidden, it may be desirable to instead hide which servers are under a heavy load as such servers are attractive targets for DDoS attacks.

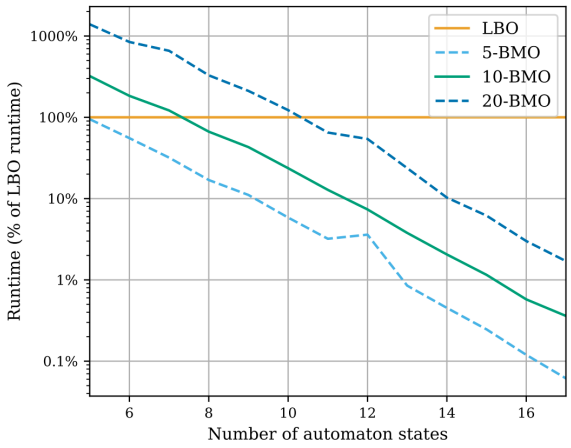

Fig. 3. The runtimes to verify different notions of opacity as a percentage of the runtime to verify LBO.

We consider the problem of verifying that these loads are hidden in the simplified load-balancing system depicted in Figure 4 in which users send requests to the balancer which then assigns these requests to servers . We model the load balancer with a DFA G_L that arbitrarily assigns requests to available servers. The overall system G is given by the parallel composition of the load balancer G_L with n_U users and n_S servers with a capacity $C \in \mathbb{N}$ modeled by the DFAs $G_{U,i}$ and $G_{S,j}$ depicted in Figure 5

$$G = G_{U,1} || \cdots || G_{U,n_U} || G_{S,1} || \cdots || G_{S,n_S} || G_L \,. \tag{11}$$

We will extend the SAT constraints developed in Section IV-C to encode opacity against attackers that can compromise user devices with a k-bounded memory. Formally, if the attacker has compromised user i, the events $\mathbf{req}_i$ and $\mathbf{res}_i$ become observable. We can incorporate this choice of observability for event σ by viewing $O(\sigma)$ from constraint (6) as a decision variable. The attacker then aims to solve a kind of *optimal sensor placement* problem, choosing users i to observe with uniform cost $c_i = 1$. By incorporating the constraints for k-BMO in the MAX-SAT framework, we can model these costs with soft clauses $\neg(O(\mathbf{req}_i) \vee O(\mathbf{res}_i))$ with weight c_i. We require that the attacker cannot deduce that a specific server is heavily loaded, i.e., at secret state C. To model this, we let $Q_{m,j}$ denote states of G where server j does not pass through state C. Then, we can encode opacity with respect to all of the server secrets by replacing the constraint φ_{Q_m} from (10) in the SAT encoding with the constraint $\varphi = \bigwedge_{j=1}^{n_S} \varphi_{Q_{m,j}}$.

By solving the resulting instance of MAX-SAT, we can determine the minimum cost of an attack with size k (if one exists) as the total weight of the solution clauses. This corresponds to the number of users that must be compromised to deduce when a server is heavily loaded. We report the results for solving this problem over a variety of parameters in Table I. As we would expect, there must be sufficiently many users for an attack to exist, i.e. $n_U \geq C n_S$. Interestingly, the size of an attack may be smaller than the number of users that it monitors. Similar to Example 3.1, the minimum cost attack

[1]Implementation available at `https://gitlab.eecs.umich.edu/M-DES-tools/bounded-opacity`

[2]The SAT solver failed to terminate on 11 out of the 1170 of instances within a five minute timeout which are not included in our analysis. We note that the unpredictability of the runtime of SAT solvers presents a limitation to our approach in practice.

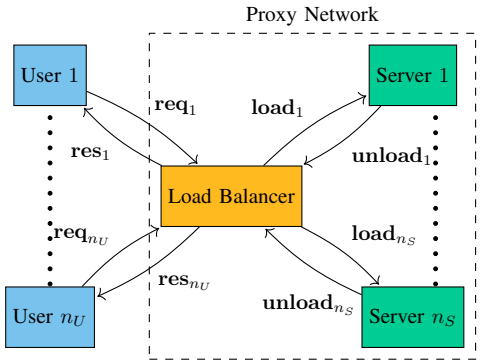

Fig. 4. The architecture of the load-balancing system.

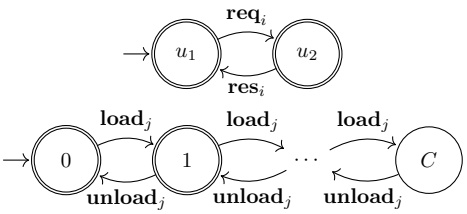

Fig. 5. The automata $G_{U,i}$ (top) modeling user i and $G_{S,j}$ (bottom) modeling server j.

in the first system utilizes the events of all 5 users but itself has only 4 states.

VI. Conclusion

In this paper, we have presented a new notion of opacity expressing privacy from an observer with a bounded memory. We derived a number of its basic properties, including the co-NP-completeness of its verification problem. We demonstrated the applicability of this notion on a number of experiments utilizing a SAT encoding for verification. There are several directions for future work utilizing this notion of opacity. Beyond verification, it may desirable to design supervisors to enforce k-BMO. In particular, it may be possible to use the SAT constraints we have developed with a quantified Boolean formula solver to perform bounded synthesis of supervisors as in [10]. Furthermore, while we have considered non-stochastic system models in this work, stochastic models may be more appropriate in many settings. For example, we may wish to relax our notion of opacity to prevent an observer from deducing a secret with high probability as in [23].

VII. Acknowledgments

The authors gratefully acknowledge helpful discussions with Jiří Balun on complexity theory. They would also like to acknowledge Ashish Kundu and Jayanth Srinivasa of Cisco Research in motivating the server load-hiding problem.

n_U	n_S	C	$\|Q\|$	k	Time(s)	#Clauses	Opaque	Cost
5	1	5	248	4	20.3	3.5×10^6	No	5
5	1	6	248	5	29.4	6.9×10^6	Yes	n/a
4	2	2	419	3	22.3	3.9×10^6	Yes	n/a
4	2	2	419	4	55.5	9.7×10^6	No	3

TABLE I
VERIFICATION RESULTS FOR THE SERVER LOAD-HIDING SYSTEM.

References

[1] A. Saboori and C. N. Hadjicostis, "Notions of security and opacity in discrete event systems," in *2007 46th IEEE Conference on Decision and Control*, Dec. 2007, pp. 5056–5061.
[2] ——, "Verification of initial-state opacity in security applications of discrete event systems," *Information Sciences*, vol. 246, pp. 115–132, Oct. 2013.
[3] F. Lin, "Opacity of discrete event systems and its applications," *Automatica*, vol. 47, no. 3, pp. 496–503, Mar. 2011.
[4] Y.-C. Wu and S. Lafortune, "Comparative analysis of related notions of opacity in centralized and coordinated architectures," *Discrete Event Dynamic Systems*, vol. 23, no. 3, pp. 307–339, Sep. 2013.
[5] A. Wintenberg, M. Blischke, S. Lafortune, and N. Ozay, "A general language-based framework for specifying and verifying notions of opacity," *Discrete Event Dynamic Systems*, vol. 32, no. 2, pp. 253–289, Jun. 2022.
[6] J. Balun and T. Masopust, "Comparing the notions of opacity for discrete-event systems," *Discrete Event Dynamic Systems*, vol. 31, no. 4, pp. 553–582, Dec. 2021.
[7] F. Cassez, J. Dubreil, and H. Marchand, "Synthesis of opaque systems with static and dynamic masks," *Formal Methods in System Design*, vol. 40, no. 1, pp. 88–115, Feb. 2012.
[8] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*, 3rd ed. Springer Nature, 2021.
[9] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. USA: W. H. Freeman & Co., 1979.
[10] L. Lin, Y. Zhu, and R. Su, "Towards bounded synthesis of resilient supervisors," in *2019 IEEE 58th CDC*, 2019, pp. 7659–7664.
[11] E. Clarke, A. Biere, R. Raimi, and Y. Zhu, "Bounded Model Checking Using Satisfiability Solving," *Formal Methods in System Design*, vol. 19, no. 1, pp. 7–34, Jul. 2001.
[12] A. Męski, W. Penczek, M. Szreter, B. Woźna-Szcześniak, and A. Zbrzezny, "BDD-versus SAT-based bounded model checking for the existential fragment of linear temporal logic with knowledge: Algorithms and their performance," *Autonomous Agents and Multi-Agent Systems*, vol. 28, no. 4, pp. 558–604, Jul. 2014.
[13] F. Cassez, J. Dubreil, and H. Marchand, "Synthesis of opaque systems with static and dynamic masks," *Formal Methods in System Design*, vol. 40, no. 1, pp. 88–115, Feb. 2012.
[14] L. J. Stockmeyer and A. R. Meyer, "Word problems requiring exponential time(Preliminary Report)," in *Proc. of the Fifth Annual ACM Symposium on Theory of Computing*, ser. STOC '73. New York, NY, USA: Association for Computing Machinery, Apr. 1973, pp. 1–9.
[15] S. Cho and D. T. Huynh, "The parallel complexity of finite-state automata problems," *Information and Computation*, vol. 97, no. 1, pp. 1–22, Mar. 1992.
[16] B. Pandey and J. Rintanen, "Planning for Partial Observability by SAT and Graph Constraints," *Proc. of the Intl. Conference on Automated Planning and Scheduling*, vol. 28, pp. 190–198, Jun. 2018.
[17] G. van Noord, "Treatment of Epsilon Moves in Subset Construction," *Computational Linguistics*, vol. 26, no. 1, pp. 61–76, Mar. 2000.
[18] B. Lennartson, M. Noori-Hosseini, and C. N. Hadjicostis, "State-labeled safety analysis of modular observers for opacity verification," *IEEE Control Systems Letters*, vol. 6, pp. 2936–2941, 2022.
[19] L. Doyen and J.-F. Raskin, "Antichain algorithms for finite automata," in *Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2010, pp. 2–22.
[20] A. D. Keromytis, V. Misra, and D. Rubenstein, "SOS: Secure overlay services," *ACM SIGCOMM Computer Communication Review*, vol. 32, no. 4, pp. 61–72, Aug. 2002.
[21] V. Kambhampati, C. Papadopolous, and D. Massey, "Epiphany: A location hiding architecture for protecting critical services from DDoS attacks," in *IEEE/IFIP Intl. Conference on Dependable Systems and Networks (DSN 2012)*, Jun. 2012, pp. 1–12.
[22] J. Wang and A. A. Chien, "Understanding when location-hiding using overlay networks is feasible," *Computer Networks*, vol. 50, no. 6, pp. 763–780, Apr. 2006.
[23] A. Saboori and C. N. Hadjicostis, "Current-State Opacity Formulations in Probabilistic Finite Automata," *IEEE Transactions on Automatic Control*, vol. 59, no. 1, pp. 120–133, Jan. 2014.