A Formal Verification Approach To Revealing Stealth Attacks on Networked Control Systems

Nikola Trčka, Mark Moulin, Shaunak Bopardikar, Alberto Speranzon
United Technologies Research Center
411 Silver Ln
East Hartford, CT

ABSTRACT

We develop methods to determine if networked control systems can be compromised by stealth attacks, and derive design strategies to secure these systems. A stealth attack is a form of a cyber-physical attack where the adversary compromises the information between the plant and the controller, with the intention to drive the system into a bad state and at the same time stay undetected. We define the discovery problem as a formal verification problem, where generated counterexamples (if any) correspond to actual attack vectors. The analysis is entirely performed in Simulink, using Simulink Design Verifier as the verification engine. A small case study is presented to illustrate the results, and a branch-and-bound algorithm is proposed to perform optimal system securing.

1. INTRODUCTION

A Cyber-Physical System (CPS) is a system consisting of computational (i.e. cyber) components that interact with physical entities. In this paper we focus on networked control systems, a special class of CPSs, where the main components are sensors, actuators, and controllers, all interconnected through, for example, ethernet or WiFi, and operating in a closed loop to maintain desired behavior of a physical plant.

Today, networked control systems support many critical infrastructures, like e.g. water and gas distribution and transportation systems, to name a few. To ensure safe and reliable behavior of these systems, their security must be of primary importance. Security analysis, or sometimes called risk assessment, is a process that identifies and evaluates system vulnerabilities, typically based on a system model and a predefined set of attack models. In this paper we propose a novel model-based security analysis technique for networked control systems, focusing on stealth attacks, a special class of cyber-physical attacks, where a possible adversary tries to compromise the system while staying undetected at the same time.

HiCoNS'14, April 15–17, 2014, Berlin, Germany.
ACM 978-1-4503-2652-0/14/04.
http://dx.doi.org/10.1145/2566468.2566484.

The mechanism of a stealth attack on a networked control system is depicted in Figure 1. We assume that the control and measurement signals are communicated over a network that is either physically or only logically composed of several secure and non-secure links. The attacker has the ability to intercept packets that flow over the non-secure links and replace them with arbitrary packets (or achieve the same effect by directly compromising sensors/actuators), and is not able to compromise any of the block components of the system. The intention of the attacker is to guide the system into a bad state, while at the same time making sure that the controller does not see any irregularities in the measurements (cf. Figure 1). For this paper we do not consider any specific diagnostic procedure, but rather a general (albeit strict) method where the measure of detectability is simply based on the difference between the attacked and the nominal measurements. Our techniques, however, can be easily applied to other monitoring and diagnostics procedures as well. In addition to the main detection mechanism operating in the control block, we also assume that a range check on the actuator signal is performed at the plant.

Stealth attacks came into prominence owing to the several recent orchestrations of attacks on CPSs such as the Stuxnet [7], a power grid by the infamous "Dr. Chaos" [6], and a company's internal heating ventilation and cooling system [5]. In control systems literature, Smith, in [10], first introduced and demonstrated a *stealth/covert attack* on a closed-loop control system. In this scenario, the attacker can compromise, with a judicious choice, both the input (actuators) and the output (measurements) of a closed loop control system so that the it is not possible to detect the attack from the output. Teixeira et al. [12, 11] study the problem in the linear, yet similar, setting and provide methods to change the system model so that a class of stealthy attacks on the actuators gets revealed. Dan and Sandberg in [4] consider stealth attacks on static linear systems and provide algorithms to secure measurements that require a maximum amount of attacker resources. Pasqualetti et al. [9] provide a more general framework to analyze several types of attacks on power systems and networks. In particular, general conditions for attack detection and identifiability for descriptor linear time-invariant systems are considered. More recently, in [3], we extended the analysis for linear systems to account for attacks on several points of a closed loop linear control system, along with techniques to prevent stealth attacks.

Not restrictive to having a *physical* element, stealth attacks can also be envisioned to occur on software systems in which threads between two interacting pieces of code can be compromised. Therefore, our goal is to design tools that can analyze not only systems modeled as differential algebraic equations but can in principle also handle complex, possibly non-linear or discrete-event systems.

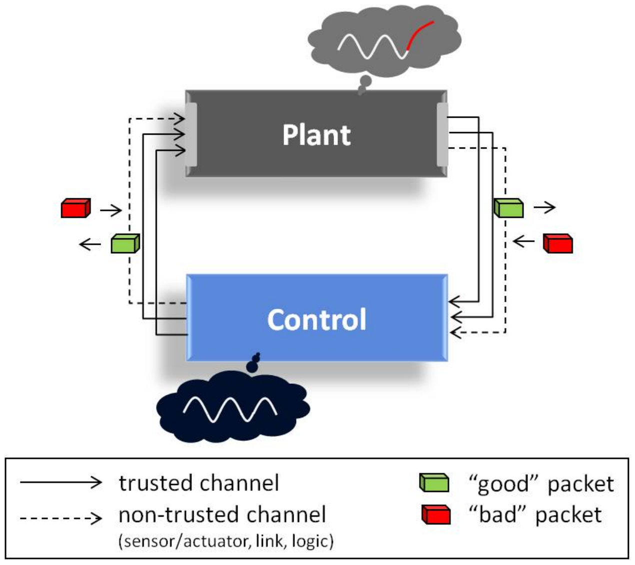


Figure 1: Depiction of a stealth attack on a networked control system

Formal verification [1, 2, 8] (in the context of checking safety) is an automated technique that given an open (and thus non-deterministic) model of a system and a desired safety property for this system, checks whether the property is always satisfied on the system or not. If the property is proven false, a counterexample is usually provided, in the form of an (external) input signal that drives the system from its initial state to the first violation point. For scalability reasons, the search space covering all possible behaviors is rarely constructed in full, and typically exists in an implicit form only. In addition, the analysis is most often bounded in depth, i.e. performed only from time 0 to some predefined model time T. Formal verification is very flexible, and can in principle handle a large class of systems, requiring only a model for which an executable semantics is provided.

This paper proposes to use formal verification to prove that a system is secure against stealth attacks of the form from Figure 1. The source of non-determinism in this context comes from the degrees of freedom the attacker has, namely the network links he has access to and the packet content he can inject, while the property to verify characterizes the attack itself, namely its stealth dynamics and its intended outcome (or goal). If the system is proven safe by formal verification, we are sure that no stealth attacks can exist on this system, meaning that every attack will be detectable from the measurements. If it is proven insecure, the generated counterexample directly maps to a stealth attack, i.e. to a sequence of inputs that guarantees stealth behavior and guides the system into a "bad" state that corresponds to the intended attack outcome. This attack can then be analyzed and, based on the gained insight, additional components of the system are secured and the verification process is repeated. Conceptually, the approach can be visualized as in Figure 2.

For broader applicability, our approach is entirely implemented in the Matlab/Simulink environment, the commercial framework commonly used for modeling control systems. We provide a generic Simulink template to cover a large class of networked control systems, only requiring from users to instantiate the system dynamics and the control logic, and to set desired verification parameters. The actual formal verification analysis is performed using Simulink Design Verifier [8], the verification engine of Matlab/Simulink, using its property proving capability in particular.

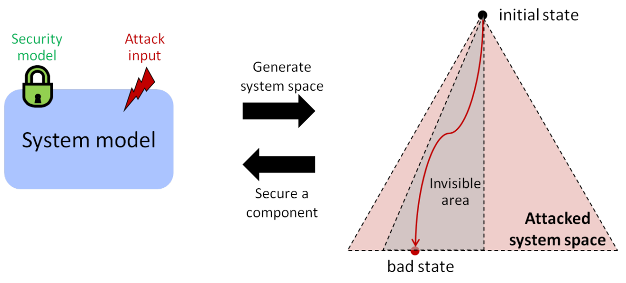


Figure 2: Formal verification approach to revealing stealth attacks

The remainder of the paper is organized as follows. In the next section we give some preliminaries. Section 3 introduces the notions of stealth attack and system security. In Section 4, we study three special cases of stealth attacks in more detail. The formal verification approach to system security is explained in Section 5. Section 6 shows a small case study to illustrate the formal verification approach. In Section 7 we cover the system design problem by presenting an algorithm for optimal system securing. Finally, in Section 8 we give conclusions and directions for future work.

2. PRELIMINARIES

$\mathbb{N}$ denotes the set of natural numbers, including zero, and $\mathbb{N}^+ = \mathbb{N} \setminus \{0\}$. $\mathbb{R}$ is the set of real numbers, and $\mathbb{R}^{\geqslant 0}$ is the set of non-negative reals. $\mathbb{R}^{m\times n}$, for $m, n \in \mathbb{N}^+$, is the set of all real matrices of dimension $m \times n$. We define $\mathbb{R}^n = \mathbb{R}^{n\times 1}$. $\wp(A)$ denotes the power set of a set A. Given a function f, we write $dom(f)$ and $range(f)$ for the domain and the range of f respectively.

3. SYSTEM UNDER STEALTH ATTACK

We consider a closed-loop discrete time dynamical system $(P, Q, \mathcal{C})$ described by the equations in Figure 3, where for every *time instant* $k \in \mathbb{N}$, the vector $x_k \in \mathbb{R}^n$ denotes the *state* of the system, $y_k \in \mathbb{R}^m$ is the *measurement* vector, $u_k \in \mathbb{R}^p$ is the *control input* vector, $P : \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}^n$ is the *plant dynamics* function, $Q : \mathbb{R}^n \to \mathbb{R}^m$ is the *measurement function*, and $\mathcal{C} : \mathbb{R}^m \to \mathbb{R}^p$ is the *control (law)* function. We call (P, Q) the *plant*, and $(P, Q, \mathcal{C}, x_0)$ is the *initialized system* from some *initial state* $x_0 \in \mathbb{R}^n$.

Given an initialized system $(P, Q, \mathcal{C}, x_0)$, we define the set of all *reachable states* of $(P, Q, \mathcal{C}, x_0)$ by $reach(P, Q, \mathcal{C}, x_0) = \{x_0, x_1, x_2, \ldots\}$. This notion is generalized to an arbitrary initial set $X_0 \subseteq \mathbb{R}^n$ by defining $reach(P, Q, \mathcal{C}, X_0) = \bigcup_{x_0 \in X_0} reach(P, Q, \mathcal{C}, x_0)$.

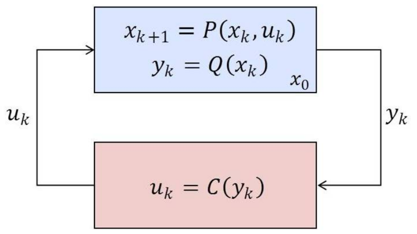
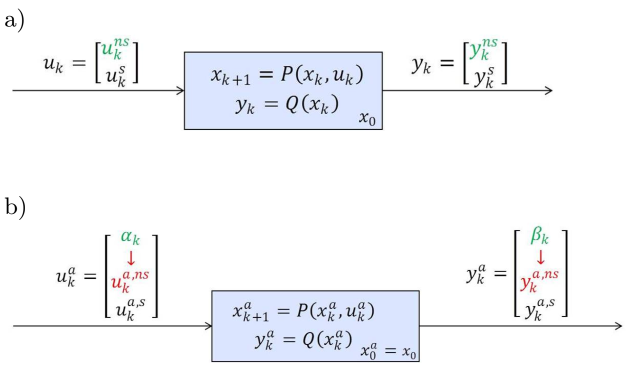


Figure 3: Discrete-time dynamical system



a)
$u_k = \begin{bmatrix} u_k^{ns} \\ u_k^s \end{bmatrix}$
$x_{k+1} = P(x_k, u_k)$
$y_k = Q(x_k)$
x_0
$y_k = \begin{bmatrix} y_k^{ns} \\ y_k^s \end{bmatrix}$
b)
$u_k^a = \begin{bmatrix} \alpha_k \\ \downarrow \\ u_k^{a,ns} \\ u_k^{a,s} \end{bmatrix}$
$x_{k+1}^a = P(x_k^a, u_k^a)$
$y_k^a = Q(x_k^a)$
$x_0^a = x_0$
$y_k^a = \begin{bmatrix} \beta_k \\ \downarrow \\ y_k^{a,ns} \\ y_k^{a,s} \end{bmatrix}$

Figure 4: a) Nominal system with signal decomposition, and b) the same system under a stealth attack

3.1 Signal partitioning and attacked system

To formalize the notion of stealth attack, we first divide the measurement and the control vector into their non-secure and secure parts. In other words, we assume that u_k has a block vector form $(u_k^{\mathrm{ns}}\ u_k^{\mathrm{s}})^{\mathrm{T}}$, where $u_k^{\mathrm{ns}} \in \mathbb{R}^{p_{\mathrm{ns}}}$, $u_k^{\mathrm{s}} \in \mathbb{R}^{p_{\mathrm{s}}}$ and $p_{\mathrm{ns}} + p_{\mathrm{s}} = p$ for some $p_{\mathrm{ns}}, p_{\mathrm{s}} \in \mathbb{N}^+$, and similarly that $y_k = (y_k^{\mathrm{ns}}\ y_k^{\mathrm{s}})^{\mathrm{T}}$ with $y_k^{\mathrm{ns}} \in \mathbb{R}^{m_{\mathrm{ns}}}$, $y_k^{\mathrm{s}} \in \mathbb{R}^{m_{\mathrm{s}}}$ and $m_{\mathrm{ns}} + m_{\mathrm{s}} = m$ for some $m_{\mathrm{ns}}, m_{\mathrm{s}} \in \mathbb{N}^+$. Given a set of measurement and control vectors V, we write V^{ns} for the set of vectors containing only the non-secure parts of the vectors in V, and similarly for the secure case. Given a matrix $M \in \mathbb{R}^{m \times p}$, the matrix $M^{\mathrm{ns,s}} \in \mathbb{R}^{m_{\mathrm{ns}} \times p_{\mathrm{s}}}$ denotes the block of M that corresponds to the non-secure part of y_k and the secure part of u_k. Similarly, we define $M^{\mathrm{ns,ns}} \in \mathbb{R}^{m_{\mathrm{ns}} \times p_{\mathrm{ns}}}$, $M^{\mathrm{s,ns}} \in \mathbb{R}^{m_{\mathrm{s}} \times p_{\mathrm{ns}}}$ and $M^{\mathrm{s,s}} \in \mathbb{R}^{m_{\mathrm{s}} \times p_{\mathrm{s}}}$.

Figure 4a shows the nominal system from Figure 3 with the measurement and the control vectors decomposed into their secure and non-secure parts. Figure 4b shows the dynamics of this system when it is under attack. At every time instant k, the non-secure part of the input is replaced by some vector $\alpha_k \in \mathbb{R}^{p_{\mathrm{ns}}}$, called the attack-intention vector, that is used to directly control the system towards the intended bad state, while at the same time the non-secure part of the measurements is replaced by a masking vector $\beta_k \in \mathbb{R}^{m_{\mathrm{ns}}}$ to hide the effect of intention vectors from previous steps and possibly manipulate the controller into producing an incorrect command. Given this, we define the notion of stealth attack as follows.

Definition 1 (Stealth attack). *Let $\varepsilon \in \mathbb{R}^{\geqslant 0}$ and $\Pi \subseteq \mathbb{R}^n \times \mathbb{N}$. A pair of sequences $\alpha_0, \ldots, \alpha_K \in \mathbb{R}^{p_{ns}}$ and $\beta_0, \ldots, \beta_K \in \mathbb{R}^{m_{ns}}$, for $K \in \mathbb{N}$, is called a* (Π, ε)-(stealth) attack *on an initialized system $(P, Q, \mathcal{C}, x_0)$ if the following holds:*

1. *$\alpha_k \in range(\mathcal{C})^{ns}$ and $\beta_k \in range(Q)^{ns}$ for every $k \in [0, K]$ - every injected signal is in the acceptable (i.e. nominal) range,*
2. *$(x_K^a, K) \in \Pi$ - the last state of the attack is according to the given relation Π,*
3. *$\| y_k - y_k^a \| \leq \varepsilon$ for every $k \in [0, K]$ - attack is undetectable (up to the given ε and the norm function) on the measurements, and*
4. *$\alpha_k \neq u_k^{ns}$ for at least one $k \in [0, K]$ - attack is not trivial (this condition is of importance only if Π does not involve system state; see next section).*

We call K the length *of the attack, Π the* attack intention property *(representing the "bad state" condition in terms of state value and time stamp) and ε the* detectability threshold *(representing the sensitivity of the potential monitoring facility in the controller).*

We now define what it means for a dynamical system and its plant to be considered secure, modulo some detectability threshold and an attack intention property. Intuitively, a system is secure from some initial state if it cannot be attacked from that state. This definition further extends to a class of initial states by requiring security property for each element of the class. Finally, a plant is considered secure if no matter what control algorithm is attached to it, the corresponding system is secure.

Definition 2 (Secure system). *Given $\varepsilon \in \mathbb{R}^{\geqslant 0}$ and $\Pi \subseteq \mathbb{R}^n \times \mathbb{N}$, an initialized system $(P, Q, \mathcal{C}, x_0)$ is* (Π, ε)-secure *if there exists no (Π, ε) attack on $(P, Q, \mathcal{C}, x_0)$. Given a non-empty set $X_0 \subseteq \mathbb{R}^n$, the system $(P, Q, \mathcal{C})$ is* (Π, ε, X_0)-secure *if $(P, Q, \mathcal{C}, x_0)$ is (Π, ε)-secure for every $x_0 \in X_0$. The plant structure (P, Q) is (Π, ε, X_0)-secure if the corresponding system structure $(P, Q, \mathcal{C})$ is (Π, ε, X_0)-secure for every control law function $\mathcal{C}$.*

4. SPECIAL CASES

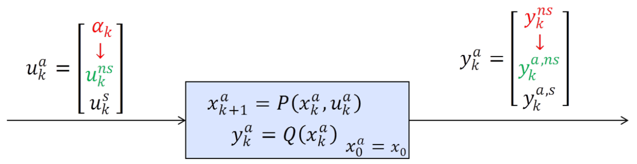
In this section we identify certain special classes of stealth attacks. These special cases are interesting as they often arise in practice, and can accelerate the formal verification approach by reducing complexity or even enable efficient analytic/closed-form solutions.

4.1 Case 1: Equality of measurement signals

The first case is the singular case where $\varepsilon = 0$ and the norm function is the entry-wise absolute value. In this case the attacker can only form the masking signal by taking $\beta_k = y_k^{\mathrm{ns}}$ for every $k \in [0, K]$, which simplifies the undetectability requirement in Definition 1 to:

- $y_k^{\mathrm{s}} = y_k^{a,\mathrm{s}}$ for every $k \in [0, K]$ - attack is undetectable on the secure part of the measurements (and thus completely).

Given this and the fact that the attacked control signal would now always be equal to the nominal signal, the attack can be depicted as in Figure 5. The benefit of this simplification is that it eliminates one degree of freedom the



$u_k^a = \begin{bmatrix} \alpha_k \\ \downarrow \\ u_k^{ns} \\ u_k^s \end{bmatrix}$
$x_{k+1}^a = P(x_k^a, u_k^a)$
$y_k^a = Q(x_k^a)$
$x_0^a = x_0$
$y_k^a = \begin{bmatrix} y_k^{ns} \\ \downarrow \\ y_k^{a,ns} \\ y_k^{a,s} \end{bmatrix}$

Figure 5: System under a stealth attack for the special case of $\varepsilon = 0$

attacker has, drastically reducing the search space that formal verification needs to explore. It is important to note, however, that in this case the corresponding security analysis would not capture attacks that e.g. take advantage of system noise to achieve the stealth property.

4.2 Case 2: State-independent attack intention property

Another special case to consider is when $\Pi \overset{\text{def}}{\equiv} \Pi_K = \mathbb{R}^n \times \{K\}$ for some $K \in \mathbb{N}$, i.e. when the attack intention property is trivial on the attack state condition, but holds only at time K. This case is relevant when we want to secure our system for all possible stealth attacks of length K, irrespective of their intentions that regard system state. In this case the following result naturally follows:

THEOREM 1. *Let $X_0 \subseteq \mathbb{R}^n$ be such that reach$(P, Q, \mathcal{C}, X_0) \subseteq X_0$, i.e. a set of system states closed under reachability. If there exists a one-step attack (Π_1, ε) from every $x_0 \in X_0$, then there is also a K-step attack (Π_K, ε) (for any $K \in \mathbb{N}$) from every such x_0. More importantly, if the system $(P, Q, \mathcal{C})$ is proven $(\Pi_1, \varepsilon, X_0)$-secure, then there is no state from which it can be attacked (for any attack length).*

4.3 Case 3: Linear system case

We now consider the important case when the system under consideration has linear dynamics. The purpose of this is to clarify the definitions of the previous section; for a more thorough treatment of the stealth attack prevention problem in the linear setting, please see [3].

Central to this section is the following theorem that gives a convenient matrix rank condition for system security when the system is linear and in addition satisfies the requirements of Case 1 and Case 2.

THEOREM 2. *Let $(P, Q, \mathcal{C})$ be a system where $P(x_k, u_k) \equiv Ax_k + Bu_k$ and $Q(x_k) \equiv Cx_k$ for some matrices $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times p}$ and $C \in \mathbb{R}^{m\times n}$. Then, for every $K \in \mathbb{N}^+$ and every $x_0 \in \mathbb{R}^n$, the initialized system $(P, Q, \mathcal{C}, x_0)$ is $(\Pi_K, 0)$-secure, for $\Pi_K = \mathbb{R}^n \times \{K\}$, iff rank$([CA^iB]^{s,ns}) = p_{ns}$, where $i \in \mathbb{N}$ is the smallest index for which $CA^iB \neq 0$.*

PROOF. The proof is based on the unfolding of the third condition in Definition 1. First, for $k = 0$, we have $[Cx_0]^{\mathrm{s}} = [Cx_0^a]^{\mathrm{s}}$ which always holds because $x_0^a = x_0$. For $k = 1$, we have

$$\begin{aligned}
&[Cx_1]^{\mathrm{s}} = [Cx_1^a]^{\mathrm{s}} \\
&\text{iff } \left[C\left(Ax_0 + B\begin{pmatrix} u_0^{\mathrm{ns}} \\ u_0^{\mathrm{s}} \end{pmatrix}\right)\right]^{\mathrm{s}} = \left[C\left(Ax_0 + B\begin{pmatrix} \alpha_0 \\ u_0^{\mathrm{s}} \end{pmatrix}\right)\right]^{\mathrm{s}} \\
&\text{iff } [CAx_0]^{\mathrm{s}} + [CB]^{\mathrm{s,ns}}u_0^{\mathrm{ns}} + [CB]^{\mathrm{s,s}}u_0^{\mathrm{s}} \\
&\qquad\qquad = [CAx_0]^{\mathrm{s}} + [CB]^{\mathrm{s,ns}}\alpha_0 + [CB]^{\mathrm{s,s}}u_0^{\mathrm{s}} \\
&\text{iff } [CB]^{\mathrm{s,ns}}(u_0^{\mathrm{ns}} - \alpha_0) = 0.
\end{aligned}$$

Further, for $k = 2$, we derive

$$\begin{aligned}
&[Cx_2]^{\mathrm{s}} = [Cx_2^a]^{\mathrm{s}} \\
&\text{iff } \left[C\left(Ax_1 + B\begin{pmatrix} u_1^{\mathrm{ns}} \\ u_1^{\mathrm{s}} \end{pmatrix}\right)\right]^{\mathrm{s}} = \left[C\left(Ax_1 + B\begin{pmatrix} \alpha_1 \\ u_1^{\mathrm{s}} \end{pmatrix}\right)\right]^{\mathrm{s}} \\
&\text{iff } [CAx_1]^{\mathrm{s}} + [CB]^{\mathrm{s,ns}}u_1^{\mathrm{ns}} = [CAx_1]^{\mathrm{s}} + [CB]^{\mathrm{s,ns}}\alpha_1 \\
&\text{iff } \left[CA\left(Ax_0 + B\begin{pmatrix} u_0^{\mathrm{ns}} \\ u_0^{\mathrm{s}} \end{pmatrix}\right)\right]^{\mathrm{s}} + [CB]^{\mathrm{s,ns}}u_1^{\mathrm{ns}} \\
&\qquad\qquad = \left[CA\left(Ax_0 + B\begin{pmatrix} \alpha_0 \\ u_0^{\mathrm{s}} \end{pmatrix}\right)\right]^{\mathrm{s}} + [CB]^{\mathrm{s,ns}}\alpha_1 \\
&\text{iff } [CAB]^{\mathrm{s,ns}}u_0^{\mathrm{ns}} + [CB]^{\mathrm{s,ns}}u_1^{\mathrm{ns}} = [CAB]^{\mathrm{s,ns}}\alpha_0[CB]^{\mathrm{s,ns}}\alpha_1 \\
&\text{iff } [CAB]^{\mathrm{s,ns}}(u_0^{\mathrm{ns}} - \alpha_0) + [CB]^{\mathrm{s,ns}}(u_1^{\mathrm{ns}} - \alpha_1) = 0.
\end{aligned}$$

Taking index i into account and extending the derivation to $k = K + i + 1$, the result can be written in matrix form as follows:

$$M = \begin{pmatrix} [CA^iB]^{\mathrm{s,ns}} & 0 & \dots & 0 \\ [CA^{i+1}B]^{\mathrm{s,ns}} & [CA^iB]^{\mathrm{s,ns}} & \dots & 0 \\ \vdots & \vdots & \ddots & 0 \\ [CA^{i+K}B]^{\mathrm{s,ns}} & [CA^{i+K-1}B]^{\mathrm{s,ns}} & \dots & [CA^iB]^{\mathrm{s,ns}} \end{pmatrix}$$

$$M \cdot \begin{pmatrix} u_0^{\mathrm{ns}} - \alpha_0 \\ u_1^{\mathrm{ns}} - \alpha_1 \\ \vdots \\ u_K^{\mathrm{ns}} - \alpha_K \end{pmatrix} = 0$$

From this it follows that a non-trivial attack of length K does not exist iff the above matrix M has full column rank. This is, however, only possible when $[CA^iB]^{\mathrm{s,ns}}$ is full column rank, i.e. when its rank is p_{ns}. Note that the first condition of Definition 1 can always be taken care of by scaling. $\square$

Note that the condition in Theorem 2 does not depend on the control law function. Therefore, in the linear case, with Π only looking at the attack length and $\varepsilon = 0$, all the notions of Definition 2 coincide.

Although this very special case allows for an analytical solution and does not require the formal verification approach, it is unfortunately not often seen in practice. Its solution efficiency, however, may potentially be exploited to guide the verification procedure (not studied in this paper).

5. SECURITY AS A FORMAL VERIFICATION PROBLEM

In this section we show how to apply formal verification to analyze system security. As explained in the introduction, the basic idea is to encode the negation of the attack conditions from Definition 1 as a verification property, so that when a counterexample is generated it directly maps to a stealth attack of the shortest length. More importantly, if no counterexample is generated, we can conclude that no stealth attack is possible, i.e. that the system is secure. We use Simulink to model the control system and Simulink Design Verifier [8] as the formal verification engine. The overall approach is illustrated in 6.

Figure 7 shows the generic Simulink template we use to prove system security. Blocks Plant and Controller represent nominal behavior, while their counterparts on the bottom represent the attacked system with exactly the same internal

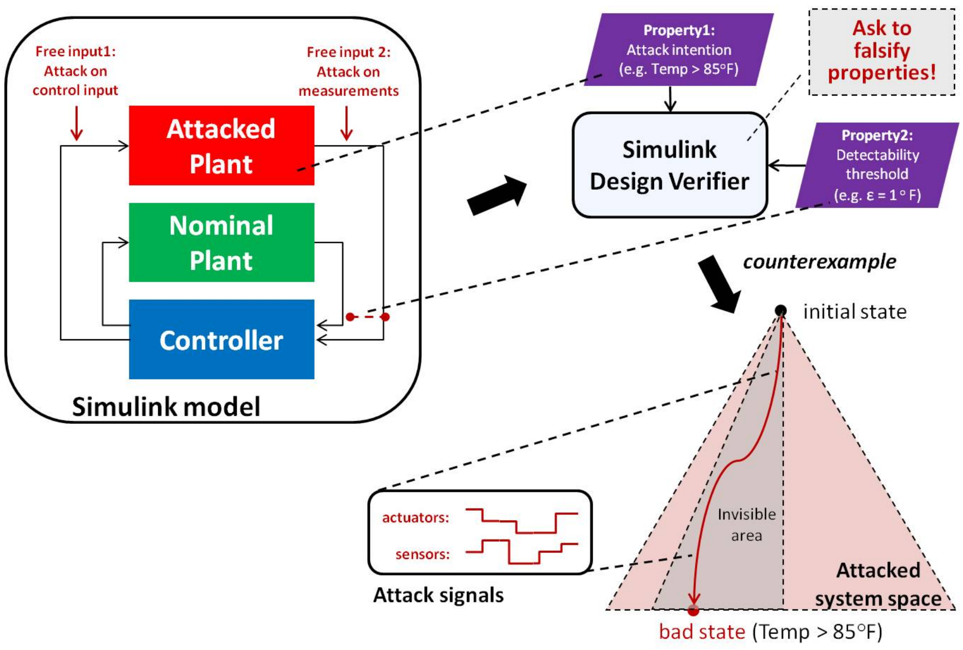

Free input1: Attack on control input
Free input 2: Attack on measurements
Attacked Plant
Nominal Plant
Controller
Simulink model
Property1: Attack intention (e.g. Temp > 85°F)
Ask to falsify properties!
Simulink Design Verifier
Property2: Detectability threshold (e.g. ε = 1° F)
counterexample
initial state
Invisible area
Attacked system space
actuators:
sensors:
Attack signals
bad state (Temp > 85°F)

Figure 6: Using Simulink Design Verifier to discover stealth attacks (or prove their absence)

dynamics but different input signals. Initial state is modeled as a free input block, with an assumption block that captures the desired range of initial conditions. Signals α and β form the attack we are searching for, so they are also modeled as free inputs and have associated assumption blocks to restrict their ranges in accordance with the first condition of Definition 1. We use Demux blocks to split control and output signals into their secure and non-secure parts, and Mux blocks to merge these parts back together.

The remaining three conditions of Definition 1 are captured inside the Attack_Condition block. This block takes the following signals as input: i) the difference between the nominal and the attacked output (for the attack undetectability property), ii) the attacked state (for the attack intention property), and iii) the difference between the injected α signal and the original non-secure part of the control signal (to ensure that the attack is not trivial). For every $k \in [0, K]$ the block outputs the joint validity of the three corresponding conditions from Definition 1 and sends it to the property block of Simulink Design Verifier for falsification. It is implemented as a conjunction of the Π signal, which includes a time counter and only operates on the last value of the state signal, and two more signals that correspond to the other two attack conditions that must be evaluated over the complete trajectory. The first signal ensures that we have the undetectability property holding along the whole trajectory, while the second one ensures that at least one α_k along this trajectory is different from the part of the nominal control output it replaces. The values of the trajectory properties for these signals are stored in their corresponding delay blocks, with initial values 0 and 1 respectively.

Figure 8 shows how the setup simplifies drastically when we assume the special case of a single fixed initial state and with $\varepsilon = 0$. The free input blocks for the initial state and the β signal are removed, and there is no need to replicate the control block anymore because the controller now receives

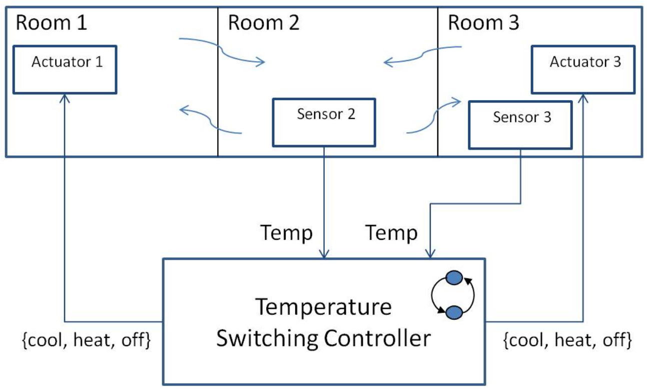


Figure 9: Three-room temperature control system

the same input both in the nominal and the attacked case. The only degree of freedom that now needs to be considered in security analysis is the injection signal α on the controller output.

We note that the models in Figures 7 and 8 are both for the case where a specific controller is taken into consideration; proving security for all possible controllers simply means replacing the controller block in these figures by a free input block (restricted to the same range).

6. CASE STUDY

We illustrate our approach on a small case study, a simple temperature control system depicted in Figure 9. The plant in the system comprises of three adjoint rooms, where the first and the third room have a cooling/heating device in them (which can be actuated by the controller), and the second and the third room have temperature sensors (which can be read by the controller). The controller implements

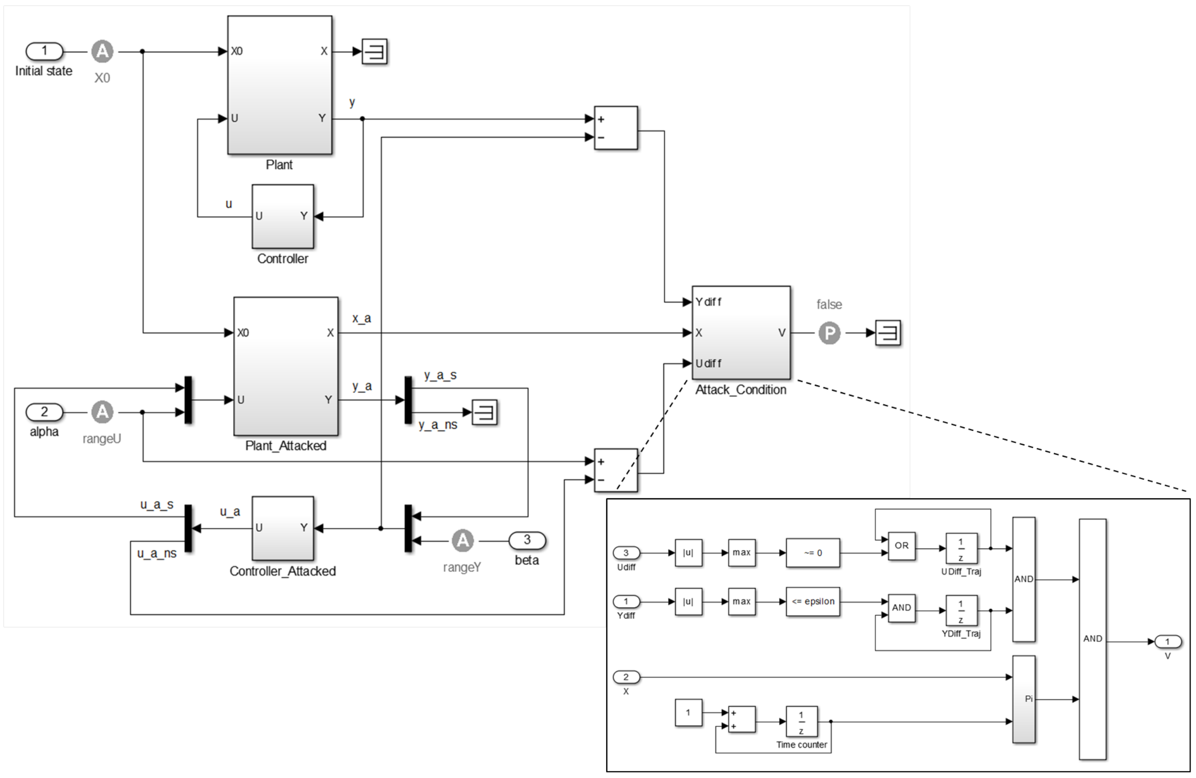


Figure 7: Generic Simulink model template for generation of stealth attacks

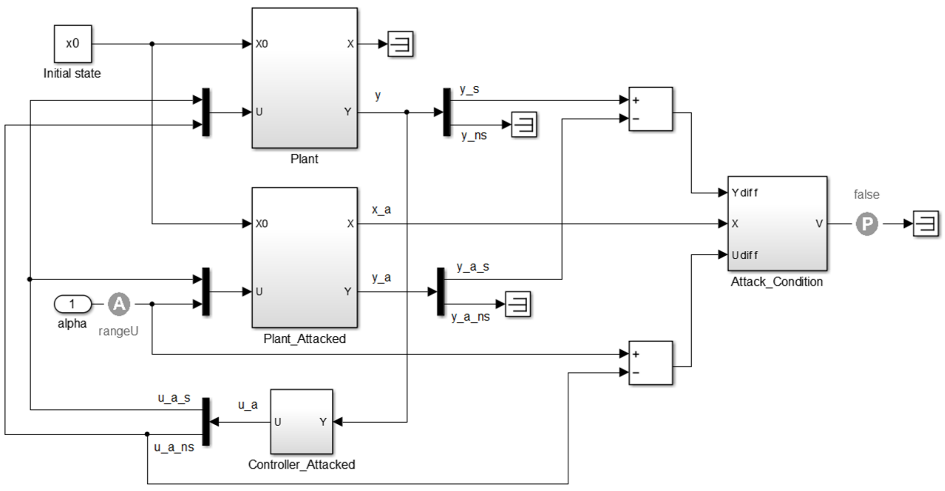


Figure 8: Simulink model template for the special case 1 ($\varepsilon = 0$) and unique initial state

$$\begin{pmatrix} T_{1,k+1} \\ T_{2,k+1} \\ T_{3,k+1} \end{pmatrix} = \begin{pmatrix} 1-c_{12} & c_{12} & 0 \\ c_{21} & 1-c_{21}-c_{23} & c_{23} \\ 0 & c_{32} & 1-c_{32} \end{pmatrix} \cdot \begin{pmatrix} T_{1,k} \\ T_{2,k} \\ T_{3,k} \end{pmatrix} + \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} u_{1,k} \\ u_{2,k} \end{pmatrix}$$

$$y_k = \begin{pmatrix} y_{1,k} \\ y_{2,k} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} T_{1,k} \\ T_{2,k} \\ T_{3,k} \end{pmatrix}$$

$$u_{i,k} = \begin{cases} -0.25, & y_{i,k} \geq T_{\max} \\ 0, & T_{\min} < y_{i,k} < T_{\max} \\ 0.25, & y_{i,k} \leq T_{\min}, \end{cases} \text{for } i \in \{1,2\}.$$

Table 1: Dynamics of the three-room temperature control system from Figure 9

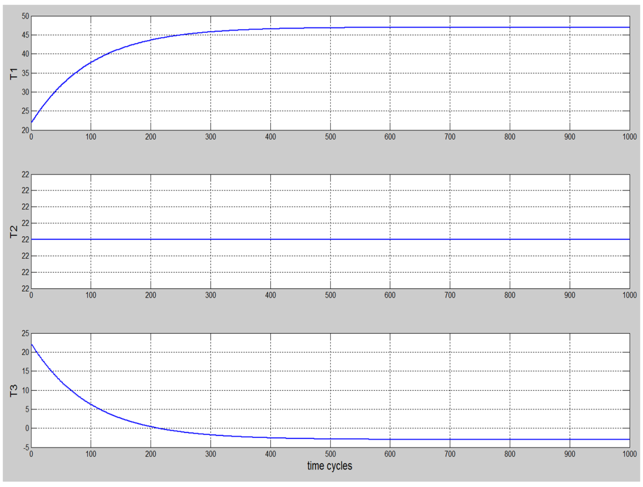


Figure 10: Nominal open-loop behavior of the three-room temperature control system

an on/off control, i.e. it chooses to cool or heat the rooms, or do nothing, based on two predefined set points $T_{\min}$ and $T_{\max}$. The choice to heat/cool room 1 is based on the measured temperature of room 2. The change of temperature between the rooms is assumed to follow the discretized basic first-order heat balance model, given in Table 1, where positive constants c_{12}, c_{21}, c_{23}, c_{32} represent the heat transfer coefficients between the rooms. For this paper, we assume a symmetric scenario and set $c_{12} = c_{21} = c_{23} = c_{32} = 0.01$.

Figure 10 shows the nominal system behavior from state $T_1 = T_2 = T_3 = 22$ when it operates in open-loop. Constant symmetric heating of room 1 and cooling of room 2 does not change the temperature in room 2. Moreover the heat transfer between rooms 1 and 3 through room 2 stabilizes the temperatures in these rooms at the unacceptable levels of $T_1 = 47$ and $T_3 = -3$. The goal of attacker is to drive this stable system into some unacceptable state. In order to analyze feasible scenarios of such stealth attacks, we made several experiments changing system secure and non-secure links, and changing the level of detectability ε.

In our first experiment, the system is initialized with values $T_1 = 22$, $T_2 = 22$ and $T_3 = 22$, and we set $T_{\min} = 21$ and $T_{\max} = 23$. We assume that the links from the controllers to the actuators in room 1 and room 3, and the link from

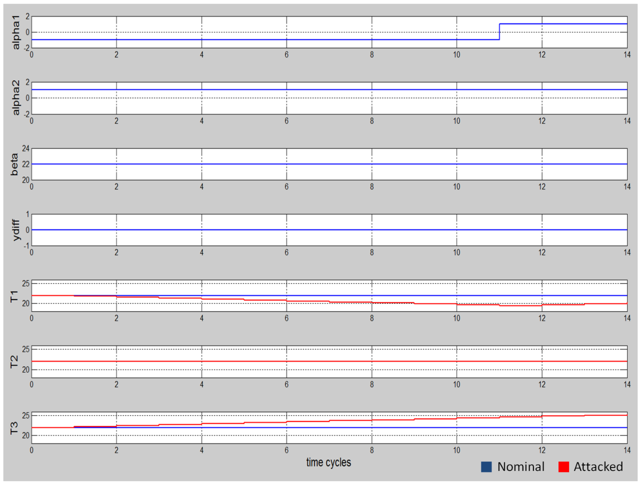


Figure 11: Analysis of the three-room temperature control system. Experiment 1: Only the link from the sensor in room 2 is secure and $\varepsilon = 0$.

the sensor in room 3 to the controller, are not-secure, and that the link from the sensor in room 2 to the controller is secure. Further, we take $\varepsilon = 0$ and define $\Pi = \{(x,k) \mid x \in \mathbb{R}^n,\ \max_{i\in[1,n]}\{x(i)\} > 25,\ k \in [0,20] \subseteq \mathbb{N}\}$. The intention of the attacker in this case is to be completely stealth for the maximum of 20 time units and, if physically possible, make sure that in this interval at least one room has temperature above 25 degrees. Applying our method to this setup we receive the results presented in Figure 11. The plot of signal T_2 is showing stealth behavior of the attack, while the signals for T_1 and T_3 are showing that there is a clear difference between the nominal $T_1^{\text{nom}} = T_3^{\text{nom}} = 22$ and the attacked system state. The length of the generated attack is 14, meaning that the system entered a bad state (where at least one state element exceeded the given bound of 25 - in this case T_3) before the bound 20 was reached. The α and β signals indicate that the attack was executed by simultaneous cooling of room 1 (signal α_1), heating of room 3 up to a point after which this room does not influence the system (signal α_2), and sending a fake measurement signal (equal to nominal) to the controller.

Our second experiment is similar to the first one except that we set $\varepsilon = 1$, and we assume that the link from the controller to the actuator in room 3 is now secure. Figure 12 shows that an attack of length 14 still exists, where the system hits a bad state in room 1 (where T_1 is above 25). In this case the attack is still based on simultaneous heating of rooms 1 and 3. However, the cooling is not anymore enforced directly through an actuator (which is now secure) but through the β signal sending the fake measurement value of $T_{\max} = 23$. Note that this attack is only possible for detectability thresholds that are bigger or equal $23 - 22 = 1$.

For the third experiment, we secure all components but the actuator in room 1, and we reduce the detectability threshold ε to 0.3. Figure 13 shows that in this case room 1 can be directly attacked through an actuator to reach level 25, while the temperature in room 2 is only deviated by 0.2 from the nominal case. According to this experiment, the initial phase of attack can target only the room 1, while the change of temperature in room 2 is small due to low heat

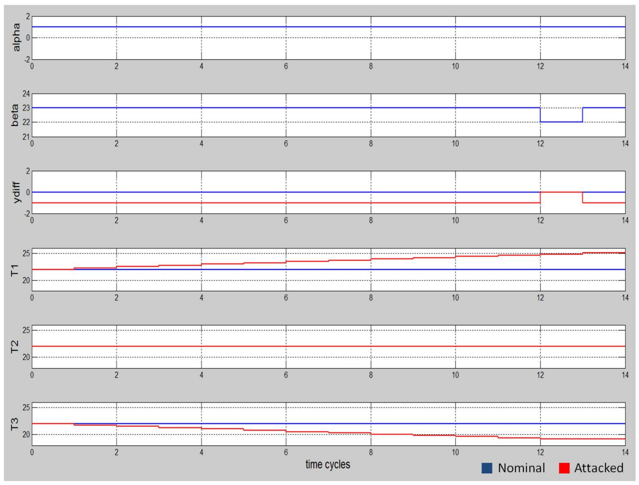


Figure 12: Analysis of the three-room temperature control system. Experiment 2: The link from the sensor in room 2 and the link to the actuator in room 3 are secure, and $\varepsilon = 1$.

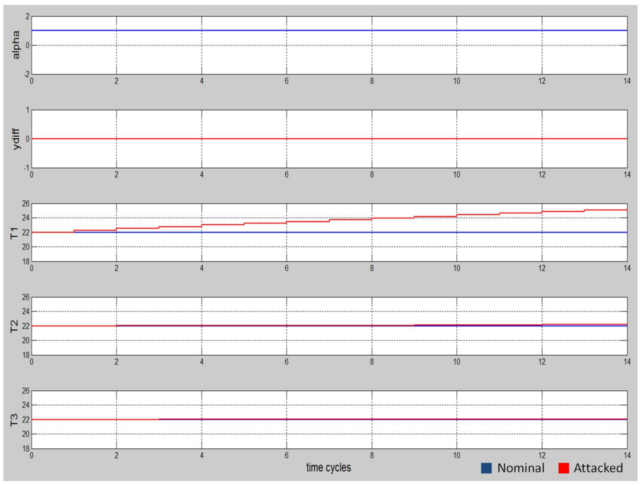


Figure 13: Analysis of the three-room temperature control system. Experiment 3: Only the actuator in room 1 is non-secure, and $\varepsilon = 0.3$.

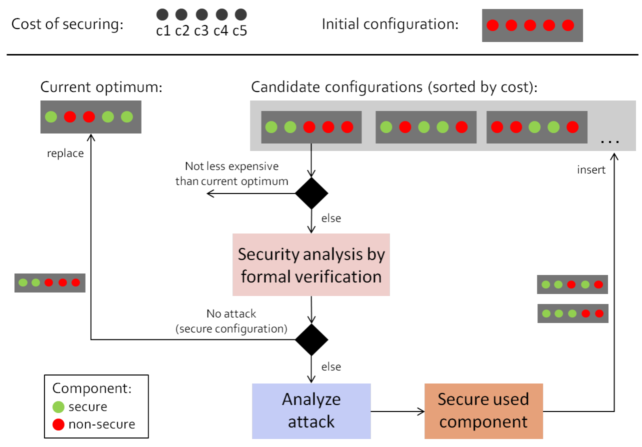

Cost of securing:
c1 c2 c3 c4 c5
Initial configuration:
Current optimum:
Candidate configurations (sorted by cost):
replace
Not less expensive than current optimum
insert
else
Security analysis by formal verification
No attack (secure configuration)
else
Component:
secure
non-secure
Analyze attack
Secure used component

Figure 14: Algorithm for optimal system securing - Snapshot of operation

transfer constants and undetected due to the low sensitivity of the detector (high ε).

For our final experiment, we set the detectablity threshold back to zero and weaken Π to Π_K for some arbitrary $K \in \mathbb{N}$. As our plant is linear, we can now apply the matrix rank condition from Theorem 2. We consider two cases: i) when only the sensor in room 2 is secure (as in Experiment 1 above), and ii) when both the sensor in room 2 and the actuator in room 3 are (as in Experiment 2). The calculation results in i) $[CB]^{\mathrm{s,ns}} = (0\ 0)$, $[CAB]^{\mathrm{s,ns}} = (0.01\ 0.01)$ and ii) $[CB]^{\mathrm{s,ns}} = (0)$, $[CAB]^{\mathrm{s,ns}} = (0.01)$. In the first case $[CAB]^{\mathrm{s,ns}}$ is not full column rank, meaning that the attacker can inject non-trivial stealth behavior into the system (with or without a real intention) for an arbitrary period of time. In the second case $[CAB]^{\mathrm{s,ns}}$ is full column rank and so this behavior is not possible (under the given strict detectability threshold).

7. OPTIMAL SECURE SYSTEM DESIGN

In this section we present a branch-and-bound algorithm that traverses the space of possible secure/non-secure system decompositions and returns a secure configuration of the minimal cost. For the search we assume that all security elements have a certain predefined cost. Algorithm 1 shows the pseudo code of the algorithm, while Figure 14 depicts the algorithm in operation.

The algorithm takes the user defined cost function as input, as well as all the parameters needed for formal verification analysis. It maintains a priority queue (lowest cost first) of candidate *configurations*, where a configuration is defined as a set of secure components (components 1 to p correspond to actuators, while components $p+1$ to $p+m$ correspond to sensors). Initially, the queue contains the empty configuration only, i.e., we first assume that all components are not secure. When the configuration from the top of the queue is taken, it is immediately discarded if its cost is not lower than the cost of the current optimum final configuration $conf_{\min}$ (line 7); otherwise, it is passed to the formal verification engine. If the verification analysis results in no attack (line 15), the configuration is considered final and the current optimum is updated. If the analysis generated an attack (line 9), for each component used at some place in the attack a new configuration is created containing this component and all the components of the original configuration. The new configurations are then added to the queue based on their costs (line 13), unless they are already in the queue. The algorithm stops when the configuration queue becomes empty, in which case the current optimal configuration represents the global optimum and the solution to the problem.

8. CONCLUSIONS

We presented a formal verification based methodology for discovering stealth attacks on networked control systems. The formal analysis is entirely performed in Simulink, using Simulink Design Verifier, and generates explicit attacks vectors in case the system is found to be non secure. The technique was demonstrated on a simple temperature control system, for different input parameters. To facilitate

Algorithm 1: Branch-and-Bound Algorithm For Optimal System Securing

```
Input: $\mathcal{M}$ - Simulink model template from Figure 7
       $K \in \mathbb{N}$ - time bound for verification
       $\mathbb{R}^{\geqslant 0} \ni \varepsilon$ - observability threshold
       $\Pi \subseteq \mathbb{R}^n \times \mathbb{N}$ - attack intention property
       cost : $[0, p+m] \to \mathbb{N}$ - costs for actuator/sensor
              (link) securing
1  $Conf = \wp([0, p+m])$ - configuration space
2  $Conf \ni conf_{\min} = \perp$ - current minimum-cost
   configuration (undefined initially)
3  **Queue**$\langle Conf \rangle \ni queue = [\emptyset]$ - queue of candidate
   configurations (containing empty configuration initially)
   // explore queue until empty
4  while $queue \neq []$ do
       // take first configuration and remove it from
       queue
5      $conf = \mathbf{head}(queue)$
6      $queue = \mathbf{tail}(queue)$
       // proceed with configuration only if its cost is
       smaller than current minimum
7      if $conf_{min} = \perp \vee cost(conf) < cost(conf_{min})$ then
           // apply formal verification
8          $(\alpha, \beta) = \mathbf{FormalVerification}(\mathcal{M}, K, \varepsilon, \Pi, conf)$
           // check if an attack was discovered or not
9          if $(\alpha, \beta) \neq \perp$ then
               // system not secure; analyze attack
               signals to see what components are used
10             for $i \in [0, p+m] \setminus conf$ do
                   // if actuator/sensor $i$ is used at some
                   place in attack, secure it and make a
                   new candidate configuration
11                 for $k \in [0, K]$ do
12                     if $i \leq p \wedge \alpha_k(i) \neq u_k^{ns}(i) \vee i >$
                       $p \wedge \beta_k(i-p) \neq y_k^{ns}(i-p)$ then
                           // insert based on cost (lowest
                           cost first)
13                         $\mathbf{insert}(queue, conf \cup \{i\})$
14                         break
15         else
               // new secure configuration found; update
               current minimum
16             $conf_{\min} = conf$
```

system securing, an algorithm that generates optimal secure configurations based on a user-defined cost function was proposed.

For our future work we plan to address the following challenges: i) scalability - find the right level of overapproximation for our model that would simplify the analysis and make it more scalable, while at the same time not introduce many spurious attacks; ii) metrics - come up with a set of metrics to quantify the negative impact of the attack, and subsequently use them in cost/security tradeoff analysis; and iii) extend analysis to other types of attacks.

9. REFERENCES

[1] A. Armin, A. Cimatti, E. Clarke, M. Fujita, and Y. Zhu. Symbolic model checking using SAT procedures instead of BDDs. In *Proceedings of the 36th annual ACM/IEEE Design Automation Conference*, pages 317–320. ACM, 1999.

[2] C. Baier and J.-P. Katoen. *Principles of model checking*, volume 26202649. MIT press Cambridge, 2008.

[3] S. Bopardikar and A. Speranzon. On analysis and design of stealth-resilient control systems. In 6*th IEEE International Symposium on Resilient Control Systems*, 2013.

[4] G. Dan and H. Sandberg. Stealth attacks and protection schemes for state estimators in power systems. In *IEEE Int. Conf. on Smart Grid Communications*, 2010.

[5] D. Goodin. Intruders hack industrial heating system using backdoor posted online, 2012. Available online at: `http://arstechnica.com/security/2012/12/intruders-hack-industrial-control-system-using-backdoor-exploit/`.

[6] F. W. Jr. 'Dr. Chaos' gets seven more years in jail, 2005. Available online at: `http://www.scmagazine.com/dr-chaos-gets-seven-more-years-in-jail/article/32757/`.

[7] J. Leyden. Stuxnet 'a game changer for malware defence', 2010. Available online at: `http://www.theregister.co.uk/2010/10/09/stuxnet\_enisa\_response/`.

[8] MathWorks. Simulink Design Verifier. `http://www.mathworks.com/products/sldesignverifier/`.

[9] F. Pasqualetti, F. Dorfler, and F. Bullo. Attack detection and identification in cyber-physical systems. *IEEE Transaction on Automatic and Control*, 2012. Conditionally accepted.

[10] R. Smith. A decoupled feedback structure for covertly appropriating networked control systems. In *18th International Federation of Automatic Control World Congress*, 2011.

[11] A. Teixeira, D. Pérez, H. Sandberg, and K. Johansson. Attack Models and Scenarios for Networked Control Systems. In *Proceedings of the 1st International Conference on High Confidence Networked Systems*, HiCoNS '12, pages 55–64. ACM, 2012.

[12] A. Teixeira, I. Shames, H. Sandberg, and K. H. Johansson. Revealing Stealthy Attacks in Control Systems. In *50th Annual Allerton Conf. on Communication, Control and Comp.*, 2012.