Efficient Computation of the Mean Time to Security Failure in Cyber Physical Systems *

Jose M. Martinez
ECE Department
Duke University
Durham, NC, USA
j.m.martinez@duke.edu

Kishor S. Trivedi
ECE Department
Duke University
Durham, NC, USA
ktrivedi@duke.edu

Benny N. Cheng
Naval Surface Warfare Center
Corona Division
Norco, CA, USA
benny.cheng@navy.mil

ABSTRACT

In this paper, we present a computationally efficient technique for calculating the mean time to security failure (MTTSF) of a mobile cyber physical system (CPS). The CPS analyzed here has been comprehensively studied by other authors using stochastic reward nets (SRN). In simple terms, the CPS is composed of a collection of communicating nodes, which are subject to security attacks. An intrusion detection mechanism is used to detect such attacks based on a voting scheme of some selected nodes. Three sources of failure are considered: successful inside attacks, Byzantine failure condition and energy exhaustion. The numerical solution technique proposed here takes advantage of the acyclic structure of the underlying Markov chain (MC) that captures the CPS dynamics. The proposed approach avoids the generation of the actual state-space of the MC, by performing a direct recursive computation with a space complexity proportional to a fraction of the number of nodes considered, which is orders of magnitude smaller than in previous works. This enables the calculation of the MTTSF for systems composed of several thousands of nodes without using parallelism.

CCS Concepts

•Security and privacy → Mobile and wireless security; Intrusion detection systems; •Computer systems organization → Reliability; •Mathematics of computing → Markov processes;

Keywords

Cyber Physical System, Security, MTTF, Markov Chain, Stochastic Reward Net.

*DISTRIBUTION A. Approved for public release; distribution is unlimited.

VALUETOOLS 2016, Taormina, Italy

DOI: 10.475/123_4

1. INTRODUCTION

We revisit a security model developed in [7] which considered a mobile cyber physical system (CPS) consisting of a number of communicating (mobile) nodes, subject to security attacks. The aim of that work was to analyze the reliability (security) of the system by calculating the mean time to security failure (MTTSF) under two main conditions: intrusion attacks and energy exhaustion.

A comprehensive Markov model of such a system was also developed in [7]. Since the state space will increase exponentially as the number of nodes being modeled, the original authors used a high level modeling formalism based on Stochastic Petri Nets known as Stochastic Reward Nets (SRN) [1]. Such a formalism allows a concise specification and automated generation/solution of underlying continuous time Markov chain (CTMC) model. Nevertheless, such a largeness tolerance approach is limited in the size of the system it can handle.

In this paper, we develop an efficient numerical technique for calculating the MTTSF or, equivalently, the MTTF (mean time to failure) from the reliability context [4], by exploiting the special structure of the underlying continuous time Markov chain (CTMC). We propose a scalable solution to the original model, in order to gain a tremendous advantage in time and space required for the computation. We note, to the best of our knowledge, that algorithms for the transient state probability computation for acyclic CTMC have been published [6] [5] [8], but algorithms for the computation of MTTF have not.

The rest of the paper is organized as follows. Section 2 gives a brief description of the cyber phyisical system under study. Section 3 presents a model of the system using stochastic reward nets (SRN). Section 4 describes the underlying continuous time Markov chain generated from the SRN model. Section 5 develops the fundamentals of the calculations of the MTTSF. Section 6 presents the recursion scheme for calculating the MTTSF, the main result of this paper. Section 7 describes the computational and numerical aspects of the new approach. Finally, Section 8 presents the conclusions and outlines some future research.

2. CYBER PHYSICAL SYSTEM

The CPS reference model, in general terms, is based on a real-world architecture system [11], which is composed of a certain number of mobile nodes (128 in the original paper). Each node uses sensors for localization, ranging its neighbors (periodically), and for measuring and communicating any detectable phenomena nearby.

VALUETOOLS 2016, October 25-28, Taormina, Italy

DOI 10.4108/eai.25-10-2016.2266825

The intrusion detection system functionality is distributed to all nodes in the system for intrusion and fault tolerance. A control node manages the mobile sensors (nodes) and is able to respond to the sometimes changing conditions (e.g., different types and strength of attacks). This control unit is considered fault and intrusion free.

The intrusion detection mechanism involves two main steps/parts: 1) the selection of a group of detectors and 2) a time interval for the periodic invocation of the mechanism. In broad terms, each node exchanges information about its location and identification periodically. A coordinator is chosen among the neighbor nodes, then this node selects m detectors randomly that will participate in the voting procedure (including itself). Finally, a node is considered as good or bad based on the majority vote.

The attack model described here considers a node capture attack which involve taking control of a good node by deceiving authentication and turning it into a bad node that will be able to generate inside attacks. The attackers primary objective is to cause impairment failure by performing persistent, random, or insidious attacks. To prevent these types of attacks, some intrusion detection mechanism is necessary. Furthermore, any detection mechanism is subject to false alarms; therefore, some level of tolerance is necessary within this mechanism.

The Byzantine fault model [3] is used to define a security failure; that is, if 1/3 or more of the nodes are compromised, then the system fails. The reason is that once the system contains more than 1/3 captured nodes, it is impossible to reach a consensus, therefore causing a security failure.

A system (security) failure can occur due to a security condition or due to an energy-exhaustion condition. In terms of security failure, two conditions are considered: 1) a Byzantine failure condition [3]; and 2) an impairment failure condition. A Byzantine failure occurs when one-third or more of the nodes have been captured by the attacker, causing the system to fail. On the other hand, an impairment failure occurs when the system is prevented from working properly due to undetected compromised nodes performing successful attacks. In terms of energy-exhaustion condition, the periodic sensing procedure (part of the intrusion detection mechanism) plays an important role, besides the normal power consumption of the nodes, in causing a system failure due to nodes running out of energy.

Our focus in this paper is to calculate the expected time until the whole system fails, the MTTSF, assuming that all nodes are working properly at the beginning of the analysis.

3. STOCHASTIC REWARD NET MODEL

Following the development in [7], Figure 1 shows the SRN model that describes the behavior of the system under analysis, where the different variables are defined as follows: N_G is the number of good nodes (nodes that have not been captured by an attacker); N_B is the number of nodes that have been turned bad; N_E is the number of evicted nodes; i.e., nodes that have been detected as bad ones by the intrusion detection mechanism; N_F is a binary variable, indicating with 1 a security failure and with 0 that there is no failure of that type (yet); and N_D is a binary variable that indicates with 1 a system energy exhaustion failure and with 0 that there is still energy available. The definition of the transition firing rates in term of the parameters of the system are given in Table 1.

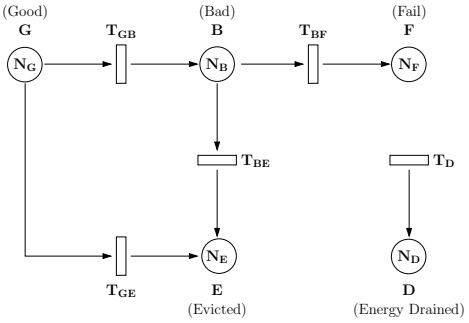

Figure 1: SRN model

Transition Name	Rate
T_{GB}	$N_G\,\lambda_c$
T_{GE}	$N_G\,\frac{P_{\text{fp}}}{T_{\text{IDS}}}$
T_{BF}	$N_B\,p_a\,\lambda_f$
T_{BE}	$N_B\,\frac{(1-P_{\text{fn}})}{T_{\text{IDS}}}$
T_D	$\frac{1}{N_{\text{IDS}}\,T_{\text{IDS}}}$

Table 1: SRN transition firing rates

The dynamics of the system is captured by the following SRN input parameters:

- The initial state of the system is given by the condition that all nodes are good; i.e., $N_G = N$ (and $N_B = N_D = N_E = N_F = 0$).
- Transition T_{GB} models the capture of a good node by an attacker. Transition T_{BE} models a bad node being detected correctly and evicted. Transition T_{GE} models a good node being detected as a bad one and being evicted (false positive detection).
- A capture rate λ_c characterizes the process that a good node is being compromised; therefore the firing rate of T_{GB} is given by $N_G\lambda_c$. A token in place B (*bad*) means that a node is compromised but is still undetected.
- A transition from place B (*bad*) to E (*evicted*) is characterized with a rate $N_B\,\frac{1-P_{fn}}{T_{IDS}}$ which considers a false negative probability (P_{fn}) of the voting-based intrusion detection mechanism. This transition models the eviction of a detected bad node. The intrusion detection procedure is carried out periodically, with period T_{IDS}.
- A transition from place G (*good*) to E (*evicted*) has a firing rate $N_G\,\frac{P_{fp}}{T_{IDS}}$ which accounts for the probability (P_{fp}) of having a false positive detection by the voting-based mechanism. That node will be taken out (evicted) from the system.
- When a node is captured, it will perform attacks with a probability p_a. The success of such attacks (from

N_B compromised nodes) is modeled through the rate λ_f. Consequently, the transition T_{BF} has a firing rate $N_B\, p_a\, \lambda_f$ leading the system to be irreversibly compromised (placing a token in F).

- A transition from place D (*energy drained*) accounts for the exhaustion of the system energy. The transition rate is given by $\frac{1}{N_{IDS}\, T_{IDS}}$, where N_{IDS} is the number of intrusion detection intervals completed before exhausting its energy, due to energy consuming tasks performed by the voting-based intrusion detection mechanism.

To complete the SRN model, we need to consider the conditions that induce a system failure (an undetected bad node performing an attack, a Byzantine failure or a energy drained condition); i.e.,

- When the number of bad nodes (N_B) is at least 1/3 of the total number of nodes ($N_B + N_G$), or equivalently $N_G \leq 2N_B$, the system fails because of a Byzantine failure. This condition is modeled by disabling all transitions in the SRN, under these conditions. SRN has a feature known as halting condition to be able to indicate this easily.
- The voting-based intrusion detection mechanism works properly by selecting m nodes; therefore a system (security) failure will occur if the number of good nodes and undetected bad nodes is smaller than m; i.e., $N_G + N_B < m$. This condition can be also modeled by disabling all transitions in the SRN under this condition.
- A token in place F means that a system failure occurred due to a captured node performing a successful attack.
- A token in place D means that enough energy was drained from the system to cause a system failure.

As was described in Section 2, a false alarm can be triggered by the security detection mechanism, which is characterized by probabilities P_{fp} and P_{fn}. The system false-positive, P_{fp}, and system false-negative, P_{fn}, detection probabilities are two characteristics of the intrusion detection mechanism that account for the possibility of detecting a good node as a bad one or a bad node as a good one, respectively. They both depend on the proportion of bad nodes in the system at the moment of performing the majority voting. Equations (1), (2), (3), and (4) below describe these probabilities. We show here the expressions allowing the calculation of these probabilities, in order to illustrate the complexity involved in their computation, but we have left out the detailed explanation of the terms since it is out of the scope of this paper. We refer to the reader to [7] for a deeper description of the expressions:

$$P_{\mathrm{fp}}(N_G, N_B) = \sum_{i=0}^{\lfloor (m-1)/2 \rfloor} A(N_G, N_B, p_a, m, i) + \sum_{j=0}^{\lfloor (m-1)/2 \rfloor} B(N_G, N_B, p_a, m, j, p_{\mathrm{fp}}), \tag{1}$$

$$P_{\mathrm{fn}}(N_G, N_B) = \sum_{i=0}^{\lfloor (m-1)/2 \rfloor} A(N_G, N_B, p_a, m, i) + \sum_{j=0}^{\lfloor (m-1)/2 \rfloor} B(N_G, N_B, p_a, m, j, p_{\mathrm{fn}}), \tag{2}$$

where $A(\cdot)$ and $B(\cdot)$ are given by:

$$A(N_G, N_B, p_a, m, i) = \frac{\binom{N_B p_a}{\lceil (m+1)/2 \rceil + i} \binom{N_G + N_B(1-p_a)}{m - \lceil (m+1)/2 \rceil - i}}{\binom{N_G + N_B}{m}}, \tag{3}$$

$$B(N_G, N_B, p_a, m, j, p_{\mathrm{x}}) = \binom{N_B p_a}{j} \cdot \sum_{k=\lceil (m+1)/2 \rceil - j}^{m-j} \left[\frac{\binom{N_G + N_B(1-p_a)}{k} \binom{N_G + N_B(1-p_a) - k}{m-j-k}}{\binom{N_G + N_B}{m}} \cdot p_{\mathrm{x}}^{k} (1 - p_{\mathrm{x}})^{m-j-k} \right], \tag{4}$$

where p_x is replaced with p_{fp} or p_{fn}, accordingly; and p_{fp} and p_{fn} are the per-host false-positive and per-host false-negative probabilities of the intrusion detection system.

Note that the calculation of P_{fn} and P_{fp} gets very complex as the number of nodes increases. Infinite precision software (e.g., Mathematica [12]) is currently used for the implementation of these expressions. In Section 7, we will revisit this issue in terms of the overall computation of the MTTSF.

4. UNDERLYING STOCHASTIC PROCESS

The underlying stochastic process of the SRN given in Figure 1 is a continuous time Markov chain (CTMC). Considering a state definition given by the 5-tuple (N_G, N_B, N_E, N_F, N_D), a general description of the state transition diagram corresponding to the CTMC is given in Figure 2, where the failure states (absorbing) due to a captured node performing a successful attack and due to energy exhaustion are shown with thick rectangles, and where $\gamma_e = \frac{P_{fp}}{T_{IDS}}$, $\gamma_{be} = \frac{(1-P_{fn})}{T_{IDS}}$, $\gamma_f = p_a \lambda_f$, and $\gamma_d = \frac{1}{N_{IDS}\, T_{IDS}}$. Note that γ_e and γ_{be} are state dependent through the probabilities P_{fp} and P_{fn}, respectively.

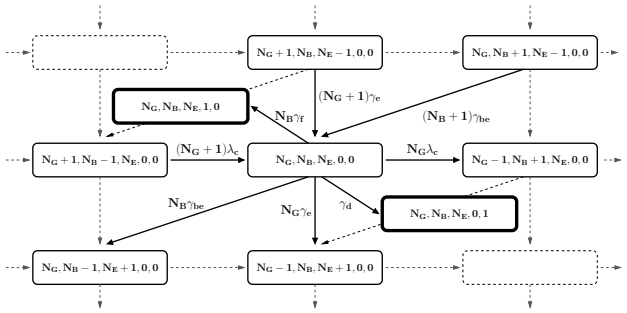

Figure 2: State transition diagram for state $(\mathbf{N_G}, \mathbf{N_B}, \mathbf{N_E}, \mathbf{N_F}, \mathbf{N_D})$. Absorbing states are shown with a thick border

In order to graphically describe the Byzantine condition and the minimum number of nodes required for the voting-based mechanism, we will consider a small running example given by a total number of nodes, $N = 8$, and the number of nodes necessary for the voting-based intrusion detection mechanism, $m = 5$. The resulting Markov chain state diagram is shown in Figure 3.

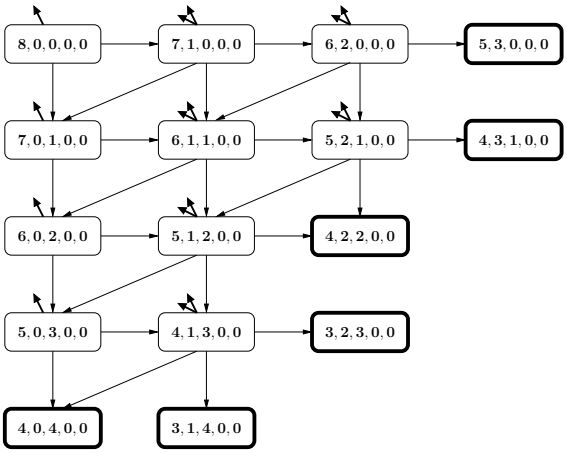

Figure 3: Example of underlying Markov chain for $\mathbf{N = 8}$ and $\mathbf{m = 5}$

For purposes of clarity, in Figure 3 we left out those absorbing states corresponding to the conditions given when $N_F = 1$ or $N_D = 1$ from the SRN model, although the transitions to those states are shown with small thick arrows coming out from the non-absorbing states. Note that in the case of states corresponding to $N_B = 0$ there is only one of such transition since only the possibility of $N_D = 1$ is present. The remaining absorbing states are shown with a thick border and correspond to the Byzantine failure condition ($N_G \leq 2\,N_B$), which are shown on the right hand side of Figure 3, and to the condition of having a number of detectors smaller than necessary for performing the voting-based intrusion detection ($N_G + N_B < m$), which are shown on the bottom of Figure 3.

Next, we will consider a simplification of the Markov chain by collecting all the absorbing states into just one, denoting the system failure condition. This will allow us to facilitate our analysis in order to compute the MTTSF. Figure 4 shows how this state space reduction is carried out in the running example.

Examining Figure 4 we can conclude, without loss of generality, that the underlying stochastic process is an acyclic Markov chain. In what follows we will use the acyclic characteristic of this CTMC to devise a recursive scheme for evaluating the MTTSF.

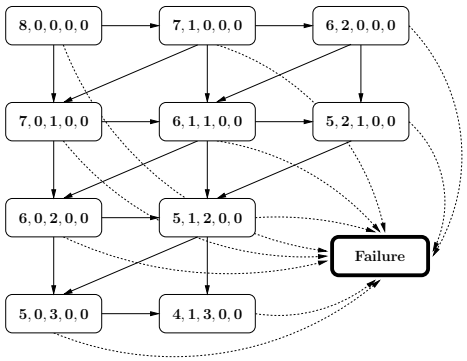

Figure 4: Reduced Markov chain for $\mathbf{N = 8}$ and $\mathbf{m = 5}$ (one absorbing state)

5. MEAN TIME TO SECURITY FAILURE

In Section 4, we saw that the underlying stochastic process is an acyclic Markov chain. Let us consider now a topological sort [2] of the states given by traversing the chain (see Figure 4) from top to bottom and from left to right and leaving the "Failure" state (absorbing) for last; i.e., in our running example, the ordered set $\{(8,0,0,0,0), (7,1,0,0,0), (6,2,0,0,0), (7,0,1,0,0), (6,1,1,0,0), (5,2,1,0,0), (6,0,2,0,0), (5,1,2,0,0), (5,0,3,0,0), (4,1,3,0,0,0), \text{(Failure)}\}$ corresponds to the state numbering from 1 to 11. It is easy to see that the underlying infinitesimal generator matrix is upper triangular, as is shown in Figure 5, where the nonzero entries are shaded. This implies that the calculation of the MTTSF can be greatly simplified, as we will explain below.

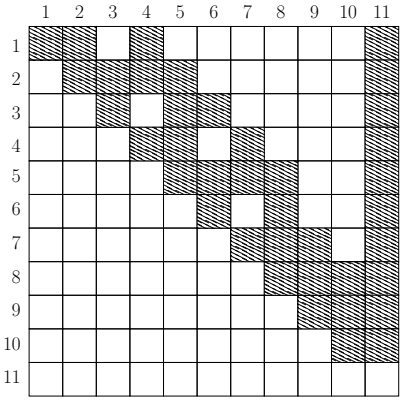

Figure 5: Infinitesimal generator matrix for $\mathbf{N = 8}$ and $\mathbf{m = 5}$ (with one absorbing state)

The calculation of the MTTSF is equivalent to the one of MTTF (concept from reliability analysis) applied to the security context. In [10] a MTTF computation method of a CTMC with failure states considered as absorbing states is shown. We use that result here to derive our efficient

recursion scheme for the evaluation of MTTSF.

Let $\mathcal{X} = \{\mathcal{X}(t),\, t \geq 0\}$ be a CTMC with finite state space $\mathcal{S} = \{1, 2, \ldots, |\mathcal{S}|\}$. Let $\mathbf{Q} = [q_{i,j}]_{i,j \in \mathcal{S}}$ be the infinitesimal generator matrix that characterizes $\mathcal{X}$, where $q_{i,j}$, with $i \neq j$, denotes the transition rate from state i to state j, and where $q_{i,i} = -\sum_{j \neq i} q_{i,j}$. Let $L_i(t)$ be the expected cumulative time that the CTMC spends in state i during the interval $(0, t]$. Let $\vec{L}(t)$ be the row vector such that $\vec{L}(t) = [L_i(t)]_{i \in \mathcal{S}}$. Then, the following system of differential equations holds:

$$\frac{d}{dt}\vec{L}(t) = \vec{L}(t)\,\mathbf{Q} + \vec{\pi}(0)\,, \tag{5}$$

where $\vec{\pi}(0) = [\pi_i(0)]_{i \in \mathcal{S}}$ is the initial probability vector of $\mathcal{X}$.

Next, consider that the state space $\mathcal{S}$ is partitioned in two sets: the set $\mathcal{T}$ of transient (non-absorbing) states and the set $\mathcal{A}$ of absorbing states. Let $\hat{\mathbf{Q}}$ be a submatrix of $\mathbf{Q}$ and let $\hat{\vec{\pi}}(0)$ be a subvector of $\vec{\pi}(0)$, where their elements correspond to states in $\mathcal{T}$ (in our running example, $\mathcal{T} = \{1, \ldots, 10\}$, see Figure 5). Then, by defining $\tau_i = \lim_{t \to \infty} L_i(t)$ and assuming that this limit exists for $i \in \mathcal{T}$, the following result can be deduced from Eq. (5):

$$\hat{\vec{\tau}}\,\hat{\mathbf{Q}} = -\hat{\vec{\pi}}(0)\,, \tag{6}$$

where $\hat{\vec{\tau}} = [\hat{\tau}_i]$, with $i = \{1, \ldots, |\mathcal{T}|\}$, is a subvector of $\vec{\tau}$ where its elements correspond to states in $\mathcal{T}$. Consequently, the MTTF can be obtained as follows:

$$MTTF = \sum_{i \in \mathcal{T}} \tau_i \tag{7}$$

Computation of Eq. (6) is carried out (usually) by using iterative methods (e.g., Jacobi, Gauss-Seidel, SOR) [9]. One of the key results of this paper is the simplification of the computation of $\hat{\vec{\tau}}$ when $\hat{\mathbf{Q}}$ is upper triangular, as is the case of the CTMC we consider here.

Consider $\hat{\mathbf{Q}} = \hat{\mathbf{D}} + \hat{\mathbf{U}}$, with $\hat{\mathbf{D}}$ being a diagonal matrix and $\hat{\mathbf{U}}$ being an strictly upper triangular matrix. Then, Eq.(6) can be written as:

$$\hat{\vec{\tau}}\,(\hat{\mathbf{D}} + \hat{\mathbf{U}}) = -\hat{\vec{\pi}}(0) \tag{8}$$

Reordering terms, we obtain:

$$\hat{\vec{\tau}}\,\hat{\mathbf{D}} = -\hat{\vec{\tau}}\,\hat{\mathbf{U}} - \hat{\vec{\pi}}(0) \tag{9}$$

Reuniting to get $\hat{\vec{\tau}}$ on the left hand side of Eq. (9):

$$\hat{\vec{\tau}} = -\hat{\vec{\tau}}\,\hat{\mathbf{U}}\,\hat{\mathbf{D}}^{-1} - \hat{\vec{\pi}}(0)\,\hat{\mathbf{D}}^{-1} \tag{10}$$

Assuming that $\hat{\pi}_1(0) = 1$ (i.e., all N nodes are working properly initially), we can derive a recursive formulation for the computation of $\hat{\tau}_i$ by writing Eq. (10) in scalar form, i.e.,

$$\hat{\tau}_i = \begin{cases} \dfrac{1}{-\hat{q}_{1,1}}\,, & i = 1\,, \\[2ex] \dfrac{1}{-\hat{q}_{i,i}} \displaystyle\sum_{j=1}^{i-1} \hat{\tau}_j\, \hat{q}_{j,i}\,, & 1 < i \leq |\mathcal{T}|\,. \end{cases} \tag{11}$$

Notice that the terms involved in the calculation of $\hat{\tau}_i$ depend only on: 1) the input and output rates of the state i, and 2) the $\hat{\tau}_j$s associated to states j with a direct arc to state i.

Next, it will be shown that this recursive scheme for computing $\hat{\tau}_i$ will allow us to devise an efficient calculation approach for the MTTSF without requiring the generation of the whole state-space of the underlying SRN, allowing savings of orders of magnitude in space and time, thus making the computation of MTTSF for CPS with thousands of nodes feasible.

6. MTTSF RECURSIVE COMPUTATION

Following the same topological structure of the running example (Figure 4), the graph-based description of Eq. (11) is depicted in Figure 6, where each cell corresponds to a value $\hat{\tau}_i$, expressed now in terms of N_E and N_B, as $\hat{\tau}_{N_E,N_B}$. The computation proceeds by levels (given by N_E) and in each level the calculation is performed from left to right (from $N_B = 0$ to $N_B = \lceil (N - N_E)/3 - 1 \rceil$. Each $\hat{\tau}_{N_E,N_B}$ depends on some previous values, indicated with directed arcs. The output transitions, from each cell to the (unique) absorbing state, are shown with an arc pointing to a small shaded circle.

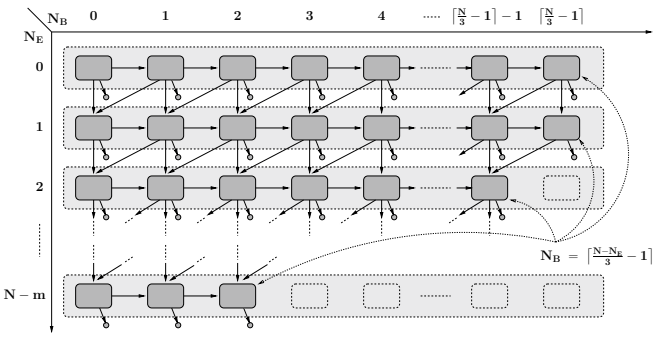

Figure 6: Recurrence graph for the calculation of $\hat{\tau}_i$ in terms of coordinates N_E and N_B.

Based on the above recurrence description, the MTTSF formula in Eq. (7) and the corresponding $\hat{\tau}_i$ recursion formulation given in Eq. (11) can be both rewritten in terms of $\hat{\tau}_{N_E,N_B}$:

$$MTTSF = \sum_{N_E=0}^{N-m} \sum_{N_B=0}^{\left\lceil \frac{N-N_E}{3} - 1 \right\rceil} \hat{\tau}_{N_E,N_B}\,, \tag{12}$$

where the index of the innermost summation moves within a level (fixed value of N_E) and the index of the outermost one moves along the N_E values (from 0 to $N - m$).

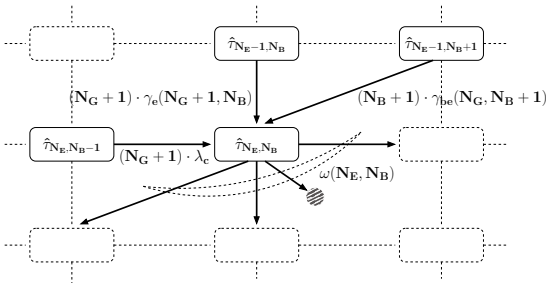

Figure 7: Detailed $\hat{\tau}_{\mathbf{N_E,N_B}}$ evaluation

Recalling that $N_G = N - N_B - N_E$ and considering the detailed description of the computation of $\hat{\tau}_{N_E,N_B}$ in Figure 7 (where γ_e and γ_{be} depend on the current state through the current values of N_G and N_B), the following recursive expression for $\hat{\tau}_{N_E,N_B}$ can be easily obtained:

$$\begin{aligned}\hat{\tau}_{N_E,N_B} = & \frac{1}{\omega(N_E,N_B)} \\ & \cdot\Big[(N_G+1)\cdot\gamma_e(N_G+1,N_B)\cdot\hat{\tau}_{N_E-1,N_B} \\ & +(N_B+1)\cdot\gamma_{be}(N_G,N_B+1)\cdot\hat{\tau}_{N_E-1,N_B+1} \\ & +(N_G+1)\cdot\lambda_c\cdot\hat{\tau}_{N_E,N_B-1}\Big]\end{aligned} \tag{13}$$

where the accompanying factor of $\hat{\tau}_{N_E-1,N_B}$, $\hat{\tau}_{N_E-1,N_B+1}$ and $\hat{\tau}_{N_E,N_B-1}$ are the transition rates from states $(N_G+1, N_B, N_E-1, 0, 0)$, $(N_G, N_B+1, N_E-1, 0, 0)$ and $(N_G+1, N_B-1, N_E, 0, 0)$, respectively; and $\omega(N_E,N_B)$ corresponds to the transition rate out from state $(N_G, N_E, N_B, 0, 0)$, see Figure 2, i.e.,

$$\begin{aligned}\omega(N_E,N_B) = & N_G\cdot\gamma_e(N_G,N_B)+N_G\cdot\lambda_c \\ & +N_B\cdot\gamma_{be}(N_G,N_B)+N_B\cdot\gamma_f+\gamma_d\end{aligned} \tag{14}$$

Finally, the boundary conditions necessary to complete this recursion are:

$$\hat{\tau}_{N_E,-1}=0\,,\quad \hat{\tau}_{-1,N_B}=0\,,\quad \hat{\tau}_{N_E,\left\lceil\frac{N-N_E}{3}-1\right\rceil+1}=0\,. \tag{15}$$

Equations (12), (13), (14) and (15) define an efficient computation approach for calculating the MTTSF of the CPS model given in Figure 1, without generating the underlying CTMC.

7. COMPUTATIONAL COMPLEXITY

Computing the MTTSF involves calculating all the $\hat{\tau}_{N_E,N_B}$ within the different levels (N_E) as depicted in Figure 6. It can be easily seen that this can be accomplished by only using two arrays of size $\lceil N/3\rceil$. The total number of levels that has to be computed is $N-m$, and in each level the actual number of $\hat{\tau}_{N_E,N_B}$ that must be calculated is $\lceil(N-N_E)/3\rceil$. Consequently, the approximate total number of $\hat{\tau}_i$ terms needed to be computed is $(\lceil N/3\rceil+\lceil m/3\rceil)\,(N-m)/2$.

The calculations involved within each $\hat{\tau}_{N_E,N_B}$ include the factors P_{fn} and P_{fp}, which get very complex as the number of nodes N and detectors m increase. In our algorithm implementation we have used infinite precision software (e.g., Mathematica). A recursion scheme for calculating P_{fn} and P_{fp} can be envisioned using the state transition diagram structure. This will be done in the future. Note that the complexity in calculating these factors stays the same as with the original paper.

Our technique yields exactly the same results as with the original SRN and the solution technique used in [7]. That is, there is no sacrifice in accuracy in achieving the scalability.

Furthermore, we must emphasize that our technique completely avoids the generation and storage of the infinitesimal generator matrix **Q**. Consequently, we also avoid the application of an iterative solution technique such as Jacobi, Gauss-Seidel or SOR, for solving the linear system given in Eq. (6), as it was done in the original paper, where the number of equations could be in the order of millions.

The enormous savings in space can be appreciated in Table 2 where the state-space cardinality in the original paper is compared with the space requirements of our recursion. The first column shows the number of nodes considered, the second and third columns show the number of states and transitions generated by the SRN in Figure 1, respectively, and the fourth column shows the total size of the vectors (arrays) that are needed during the whole computation of the recursion. It can be easily appreciated that using the original approach (solving the SRN) imposes a great demand on memory resources when just a few thousands of nodes are analyzed. On the other hand, with our approach, the memory requirement are upper-bounded by the number of nodes, making it possible to analyze systems with thousands of nodes in a single machine.

N	No. States (approx.)	No. Transitions (approx.)	Recursion (No. values)
100	5 000	8 350	67
200	20 000	33 400	134
500	125 000	208 700	334
1 000	500 000	835 000	667
2 000	2 000 000	3 340 000	1 334
5 000	12 500 000	20 875 000	3 334
10 000	50 000 000	83 500 000	6 667
20 000	200 000 000	334 000 000	13 334
50 000	1 250 000 000	2 087 500 000	33 334

Table 2: No. States/No. Transitions as a function of N in the original SRN, compared with the storage needed for the proposed recursion

Table 3 shows the computation time for several combinations of number of nodes (N) and number of detectors (m), considering that the parameters used (for the rest of the variables) are the same as in the original paper. We must note that the computation time greatly depends on the calculation of P_{fn} and P_{fp}, a cost that is inherited from the original paper, keeping open the possibility for future improvement since it was not the focus of our contribution. The implementation of the algorithm was done completely using Mathematica [12].

8. CONCLUSIONS

In this paper, we developed a numerical approach to efficiently compute the mean time to system failure of a cy-

N	Time (sec)	
	m = 3	m = 5
1 000	3	4
5 000	83	90
10 000	349	370
20 000	1 388	1 981
50 000	8 796	9 247

Table 3: Computation time of the implemented recursion in Mathematica (2.8GHz Intel Core i5)

ber physical system composed of a network of communicating/sensing mobile nodes, where a system (security) failure is caused by either a security intrusion attack or due to energy exhaustion.

The numerical technique presented here takes advantage of the underlying structure of the continuous-time Markov chain that captures the CPS behavior. It computes the MTTSF precisely and recursively, without generating the actual Markov chain, allowing enormous savings in space and time compared to the original procedure based on modeling and solving an SRN, where a linear system of equations (with millions of equations, eventually) has to be solved in order to compute such a measure. The approach presented here allow us to calculate the MTTSF for CPS models consisting on thousands of nodes using a single machine and with just a few lines of code.

Our efficient numerical technique takes the computation of the MTTF (from the reliability context) and adapts it for the case of an acyclic Markov chain with an upper triangular infinitesimal generator matrix.

In terms of future work, one task considers extending the MTTF calculation technique in order to include general acyclic Markov chains; i.e., not necessarily with an upper triangular infinitesimal generator matrix. Another task considers the efficient computation of the false alarm probabilities P_{fn} and P_{fp}, using a recursion scheme that also takes advantage of the state transition diagram structure of the underlying Markov chain. And another direction of improvement would be to consider other type of security failure condition different from the Byzantine condition.

9. ACKNOWLEDGMENTS

We thank Robert Mitchell, Ph.D., from Sandia National Laboratories, for early discussions related to his original work.

This research was partially supported by US NSF grant number CNS-1523994, by US Navy NEEC grant number N00174-16-C-0036, by IBM under a faculty grant, and by NATO under Science for Peace project number 984425.

10. REFERENCES

[1] G. Ciardo, A. Blakemore, P. F. Chimento, J. K. Muppala, and K. S. Trivedi. Automated generation and analysis of markov reward models using stochastic reward nets. In C. D. Meyer and R. J. Plemmons, editors, *Linear Algebra, Markov Chains, and Queueing Models*, pages 145–191. Springer New York, New York, NY, 1993.

[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 3rd edition, 2009.

[3] L. Lamport, R. Shostak, and M. Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3):382–401, July 1982.

[4] E. E. Lewis. *Introduction to Reliability Engineering*. Wiley, 2nd edition, 1996.

[5] C. Lindemann, M. Malhotra, and K. S. Trivedi. Numerical methods for reliability evaluation of Markov closed fault-tolerant systems. *IEEE Transactions on Reliability*, 44(4):694–704, December 1995.

[6] R. A. Marie, A. L. Reibman, and K. S. Trivedi. Transient solution of acyclic Markov chains. *Performance Evaluation*, 7(3):175–194, August 1987.

[7] R. Mitchell and I.-R. Chen. Effect of intrusion detection and response on reliability of cyber physical systems. *IEEE Transactions on Reliability*, 62(1):199–210, March 2013.

[8] H. Nabli. Performability measure for acyclic Markovian models. *Computers & Mathematics with Applications*, 35(8):41–51, 1998.

[9] W. J. Stewart. *Introduction to the Numerical Solution of Markov Chains*. Princeton University Press, 1994.

[10] K. S. Trivedi. *Probability and Statistics with Reliability, Queuing, and Computer Science Applications*. Wiley, 2nd edition, 2001.

[11] US Department of Homeland Security. BAA-09-02 geospatial location accountability and navigation system for emergency responders (GLANSER), 2009.

[12] Wolfram Research, Inc. Mathematica, Version 10.4. Champaign, Illinois, 2016. http://www.wolfram.com.