2016

Metrics for Secrecy and Resilience in Cyber-Physical-Systems

Mariam Wajdi Ibrahim
Iowa State University

Metrics for secrecy and resilience in cyber-physical-systems

by

Mariam Wajdi Ibrahim

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Electrical Engineering

Program of Study Committee:

Ratnesh Kumar, Major Professor

Samik Basu

Akhilesh Tyagi

Yong Guan

Zhenqiang Gong

Iowa State University

Ames, Iowa

2016

DEDICATION

I would like to dedicate this dissertation to my family whose love and support enlighten my life. My parents Margo Alamat and Wajdi Ibrahim, my sisters Rania, Rawan and Noor, my brothers Anwar, Ihsan and Marwan, and my dearest nieces Maria and Sophia.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my thanks to those who helped me with various aspects of conducting research and the writing of this dissertation. First and foremost, Dr. Ratnesh Kumar for his guidance and patience throughout this research. I would also like to thank my committee members and all the professors who gave me courses at ISU for their efforts and contributions to this work. Furthermore, I would like to express my gratitude to the German Jordanian University for the financial support during my stay at ISU.

ABSTRACT

In this dissertation, we study the problem of *Secrecy and Resiliency* quantification for cyber physical systems. Secrecy (also known as confidentiality) refers to the ability to withstand attempts to uncover information/behaviors, whereas resilience (also known as integrity) refers to the ability to withstand attempts to modify information/behaviors. Thus, former is an observability related attribute while the latter is an attribute related to controllability. In this dissertation we are primarily concerned with protecting systems behaviors from being revealed or altered.

Unlike information, behaviors cannot be encrypted and may instead be protected by providing covers that generate indistinguishable observations from behaviors needed to be kept secret. Such a scheme may still leak information about secrets due to statistical difference between the occurrence probabilities of the secrets and their covers. Jensen-Shannon Divergence (JSD) is a possible means of quantifying statistical difference between two distributions and can be used to measure such information leak as presented in this dissertation. Using JSD, we quantify loss of secrecy in stochastic partially-observed discrete event systems in two settings: (i) the centralized setting, corresponding to a single attacker/observer, and (ii) the distributed collusive setting, corresponding to multiple attackers/observers, exchanging their observed information. In the centralized case, an observer structure is formed and used to aide the computation of JSD, in the limit, as the length of observations approach infinity to quantify the worst case loss of secrecy. In the distributed collusive case, channel models are introduced to extend the system model to capture the effect of exchange of observations, that allows the JSD computation of the centralized case to be applied over the extended model to measure the distributed secrecy loss.

We also formulate a measure for *resiliency* for dynamical hybrid systems with focus on power systems. The resiliency measure, called *Level-of-Resilience* (LoR), determined by examining:

(i) the *Region-of-Stability-Reduction* (RoS_R), as the *RoS* evolves under attack and recovery actions as captured by a "*modal-RoS*", (ii) the eventual *Level-of-Performance-Reduction* (LoP_R), as measured by percentage of reduction of load served, and (iii) *Recovery-Time (RT)*, which is the time system takes to detect and recover from an attack or a fault. We illustrate our measure by comparing resiliency level of two power systems under two different attack scenarios.

The level of resilience of a given system is assessed under various attack scenarios. We present a model-based approach for generating such attack scenarios. This requires a comprehensive description of the system model (describing architecture and connectivity, components and behaviors, assets, defenses, vulnerabilities, atomic attacks), as well as of security/resiliency properties being investigated. A state exploration based approach has been proposed to find all behaviors/paths of the model leading to those reachable states where the specified security/resiliency properties are violated. An *attack graph* is a collection of all paths from initial states to such reachable violating states. We present a model-based attack graph generation approach and its implementation.

CHAPTER 1. INTRODUCTION

The dissertation explores the topics of metrics for *Secrecy* and *Resilience* in *Cyber-Physical-Systems*. Quantifying the ability to hide secrets (sensitive information) from a single observer/collusive observers is a challenge. This dissertation provides a means to quantify this in terms of a type of distance measure between the distributions of observations arising from a secret and its cover. Dually, we also identify measures to describe how resilient the system is against adverse events. Measuring the secrecy and resiliency level may aid system engineers to revisit the system design for achieving satisfactory metric.

Cryptography is used to protect the content of information (e.g, a message) by making it undecipherable, however, behaviors (as opposed to information) may not be encrypted, and may only be protected by partially or fully hiding through creation of ambiguity by providing covers that generate indistinguishable observations from secrets; for instance, in [Shoukry et al. (2013)], the attacker can initiate an attack against the *Anti-lock Braking System (ABS)* speed sensor used in automobiles. In such attack, the attacker alters the physical environment surrounding the *ABS* sensor (i.e., by shielding the sensor from its surrounding) using certain spoofing electromagnetic device. Doing so, this will allow the attacker to inject his/her own speed signal, and thus corrupting the actual *ABS* speed measurement, and gaining an overall control over the automobile maneuvers. In this scenario, data encryption may not help protecting the system against this spoofing attack since the measured signal itself is corrupted, also the behavioral maneuvers of the automobile cannot be encrypted too.

In general, secrecy may arise from partial observability, through the creation of ambiguity about system secret state by producing other state (i.e., cover) under indistinguishable observation; for instance, an attacker who passively tries to infer the navigation/maneuvers of an automobile, using a set of sensors deployed in certain places in the road, an ambiguity

about his/her observation of the automobile track can be generated by having other possible maneuvers with same sensor observations (act as a cover) from the secret ones. Researchers in the field of security and privacy have explored many techniques for hiding secrets based on ambiguation schemes such as, *Steganography and Watermarking* [Kundur and Ahsan (2003); Christian S. Collberg (2002)], *Network level Anonymization* [Ren and Wu (2010)], and *Software Obfuscation* [Garg et al. (2013)].

A first part of this dissertation focuses on the confidentiality (secrecy) property for systems modeled as partially-observed stochastic discrete event systems (stochastic PODES), which are Markovian generators of *arbitrary* long sequences with labeled transitions being *partially-observed* [Ibrahim et al. (2016a,b)]. Systems receive inputs and produce outputs, and in the process may leak some undesired information, and this has been formalized and quantified.

A second security aspect of this dissertation is about measuring the resiliency of dynamical systems. Resiliency can be defined as "*ability to withstand adverse events*". Their quantification allows system designers to assess system security under attacks. We introduce the notion of Level-of-Resilience (LoR) as a way to compare the resiliency of different systems, subject to various attack scenarios [Ibrahim et al. (2016c)]. An adverse event can affect both system's stability and performance, and whose recovery time is also another important metric. Accordingly, in our present work, we consider size of Region-of-Stability (RoS), the Level-of-Performance (e.g., load served in case of a power system), and Recovery-Time as part of the proposed resilience metric. We show these can be computed through power system examples subjected to a sequence of atomic attacks, and show how one power system topology may be better over another with respect to the proposed resiliency measure.

The level of resilience of a given system is assessed under various attack scenarios. We present a model-based approach for generating such attack scenarios (sequence of attack/recovery actions resulting in system compromise). This requires a comprehensive description of the system model, as well as the security/resiliency properties being investigated. A state exploration based approach has been proposed to find all behaviors/paths of the model leading to those reachable states where security/resiliency property is violated. An *attack graph* is a collection of all paths from initial states to such reachable violating states. Once attack graphs are gen-

erated for systems, further analysis can be done to compare the resiliency levels against the attack scenarios within the graph. The framework was already proposed in [Jha et al. (2002); Sheyner et al. (2002)], we build explicit model for state-space exploration, identifying the appropriate state-variables and transition rules. We also provided an initial implementation of state exploration based attack graph generation using architectural description tool together with a back-end model-checker.

1.1 Related Works

Various notions of information secrecy have been explored in literature. For example, [Smith (2009)] examines non-interference, requiring that secrets (private variables) do not interfere with or influence the observables (public variables). Non-interference is a logical notion that is either satisfied or violated, and as such it does not allow the quantification of the degree to which a system may violate the property. In contrast, for stochastic systems, the mutual information between the private and public variables can be used to quantify the level of interference, and hence loss of secrecy [Smith (2009)]. Mutual information is only an average case measure, and a worst case measure can also be defined, using for example *min-entropy* [Espinoza and Smith (2013)]. Extension of the notion of non-interference over behaviors (sequences) was explored in [Takai and Kumar (2009)], requiring that every secret behavior must be masked by a cover behavior.

For information leakage over *sequences of observations* from a stochastic systems, mutual information can again be used to quantify the level of secrecy loss, and as shown in [Chen et al. (2015)], it can be related to a certain Jensen-Shannon Divergence (JSD) computation, and can be used to measure the disparity between the distributions of a secret versus its cover as a way to quantify the secrecy [Bryans et al. (2011)]. The definition of S_τ-secrecy proposed by [Ibrahim et al. (2014)] bounds the probability of revealing the secret over the set of *all* behaviors. It is shown that S_τ-Secrecy can be viewed as a generalization of the logical secrecy defined in [Takai and Kumar (2009)], and that it is a variant of the divergence used in [Bryans et al. (2011)]. The above mentioned secrecy notions (also referred to *opacity* in literatures), along with related articles have been reviewed in a recent survey [Jacob et al. (2015)].

Dually, we also identify measures to describe how resilient the system is against adverse events, which we illustrate for power systems. Various measures of power system resilience have been investigated throughout literature. For instance, the average efficiency of the network [Latora and Marchiori (2001)], adapted to the case of the North American power grid, is used by [Kinney et al. (2005)] to quantify the performance of grid operations before and after the occurrence of breakdowns. This measure is based on the most efficient path between the generation substation i and the distribution substation j, where path efficiency between two nodes i and j is the harmonic composition of the efficiencies of the component edges. Then, the damage caused by a failure is defined as the normalized efficiency loss.

The duration of unscheduled outages due to failure of distribution system is also proposed in [Maliszewski and Perrings (2012)] as a resilience measure. A recent survey [Willis and Loa (2015)] summarized resilience measures of energy distribution systems. The building blocks of resilience are: *inputs* available to support resilience, *capacities*, which are the ways in which inputs are organized to support resilience, *capabilities* of what tasks can be performed, the *performance* and *outcomes* that describe what is produced by an engineered system. These building blocks address the goal of reducing the damage from disasters.

1.2 Organization of Dissertation

The rest of this dissertation is organized as follows. Secrecy quantification in stochastic PODESs in the presence of a single attacker/observer, having partial observability of system behaviors for revealing sensitive/secret states (reachable by secret I/O behaviors), is introduced in chapter 2. We employ the JSD based measure of secrecy loss, and propose a method to compute it in the centerlized setting. The computation of "limiting" JSD measure, quantifying the worst case statistical difference that is defined over arbitrary long observation sequences, is presented. The proposed JSD based quantification for secrecy loss is shown to be equivalent to the mutual information between the distribution over the observations and that over the possible status of system execution (whether secret or cover).

In chapter 3 we study the secrecy quantification in stochastic PODESs in the distributed collusive setting, where there exist multiple observers/attackers that have their own personal

observations, and also collude by exchanging their observations over channels, that introduce delays that are bounded. To compute JSD measure in this setting, we introduce channel models and use those to extend the system model as in [Qiu and Kumar (2008)], capturing own observations as well as the delayed communicated observations. The JSD computation approach of the centralized setting of chapter 2 is then employed to the extended model to yield the JSD measure of the distributed collusive setting.

In chapter 4, we present the notion of Level-of-Resilience (LoR) as a way to compare the resiliency of different systems, subject to various attack scenarios. An adverse event can affect both system's stability and performance, and whose recovery time is also another important metric. Accordingly, we consider size of Region-of-Stability (RoS), the Level-of-Performance (e.g., load served in case of a power system), and Recovery-Time as part of the proposed resilience metric.

In chapter 5, we present a model-based approach for generating attack scenarios that can be used in the resiliency analyses. This requires a comprehensive description of the system model (describing architecture and connectivity, components and behaviors, assets, defenses, vulnerabilities, atomic attacks), as well as of security/resiliency properties being investigated. A state exploration based approach has been proposed to find all behaviors/paths of the model leading to those reachable states where the specified security/resiliency properties are violated. An *attack graph* is a collection of all those paths. We present a model-based attack graph generation approach and its implementation. In Chapter 6, we summarize the work and conclude with the discussions of future work.

CHAPTER 2. QUANTIFICATION OF CENTRALIZED SECRECY IN STOCHASTIC DISCRETE EVENT SYSTEMS

In this chapter, we study secrecy quantification in stochastic PODESs in the presence of a single attacker/observer, having partial observability of system behaviors for revealing sensitive system behaviors. We propose a JSD based quantification to measure the secrecy loss in this centralized setting as introduced below. The proposed JSD based quantification for secrecy loss is shown to be equivalent to the mutual information between the distribution over possible observations and that over possible status of system execution (secret versus cover).

2.1 Notations and Preliminaries

For an event set Σ, define $\overline{\Sigma} := \Sigma \cup \{\epsilon\}$, where ϵ denotes "no-event". The set of all finite length event sequences over Σ, including ϵ is denoted as Σ^*, $\Sigma^+ := \Sigma^* - \{\epsilon\}$, and Σ^n is the set of event sequences of length $n \in \mathbb{N}$. A *trace* is a member of Σ^* and a *language* is a subset of Σ^*. We use $s \leq t$ to denote if $s \in \Sigma^*$ is a prefix of $t \in \Sigma^*$, and $|s|$ to denote the length of s or the number of events in s. For $L \subseteq \Sigma^*$, its prefix-closure is defined as $pr(L) := \{s \in \Sigma^* | \exists t \in \Sigma^* : st \in L\}$ and L is said to be prefix-closed (or simply closed) if $pr(L) = L$, i.e., whenever L contains a trace, it also contains all the prefixes of that trace. For $s \in \Sigma^*$ and $L \subseteq \Sigma^*$, $L\backslash s := \{t \in \Sigma^* | st \in L\}$ denotes the set of traces in L *after* s.

2.1.1 Stochastic PODES.

We can model a stochastic PODES by a *stochastic automaton* $G = (X, \Sigma, \alpha, x_0)$, where X is the set of states, Σ is the finite set of events, $x_0 \in X$ is the initial state, and $\alpha : X \times \Sigma \times X \to [0, 1]$ is the probability transition function [Garg et al. (1999)], and $\forall x \in$

$X, \sum_{\sigma\in\Sigma}\sum_{x'\in X}\alpha(x,\sigma,x') = 1$. A non-stochastic PODES can be modeled as the same 4-tuple, but by replacing the transition function with $\alpha : X \times \Sigma \times X \to \{0,1\}$, and a non-stochastic DES is deterministic if $\forall x \in X, \sigma \in \Sigma, \sum_{x'\in X}\alpha(x,\sigma,x') \in \{0,1\}$. The transition probability function α can be generalized to $\alpha : X \times \Sigma^* \times X$ in a natural way: $\forall x_i, x_j \in X, s \in \Sigma^*, \sigma \in \Sigma, \alpha(x_i, s\sigma, x_j) = \sum_{x_k\in X}\alpha(x_i,s,x_k)\alpha(x_k,\sigma,x_j)$, and $\alpha(x_i,\epsilon,x_j) = 1$ if $x_i = x_j$ and 0 otherwise.

Define the language generated by G as $L(G) := \{s \in \Sigma^* \mid \exists x \in X, \alpha(x_0,s,x) > 0\}$. For a given G, a *component* $C = (X_C, \alpha_C)$ of G is a "subgraph" of G, i.e., $X_C \subseteq X$ and $\forall x, x' \in X_C$ and $\sigma \in \Sigma$, $\alpha_C(x,\sigma,x') = \alpha(x,\sigma,x')$ whenever the latter is positive, and $\alpha_C(x,\sigma,x') = 0$ otherwise. C is said to be a *strongly connected component* (SCC) or *irreducible* if $\forall x, x' \in X_C$, $\exists s \in \Sigma^*$ such that $\alpha_C(x,s,x') > 0$. A SCC C is said to be *closed* if for each $x \in X_C$, $\sum_{\sigma\in\Sigma}\sum_{x'\in X_C}\alpha_C(x,\sigma,x') = 1$. The states which belong to a closed SCC are *recurrent states* and the remaining states (that do not belong to any closed SCC) are *transient states*. Another way to identify recurrent versus transient states is to consider the steady-state state distribution π^* as the fixed-point of $\pi^* = \pi^*\Omega$, where π^* is a row-vector with the same size as X, and Ω is the transition matrix with ijth entry being the transition probability $\sum_{\sigma\in\Sigma}\alpha(i,\sigma,j)$. (In case Ω is periodic with period $d \neq 1$, we consider the set of fixed-points of $\pi^* = \pi^*\Omega^d$). Then any state i is recurrent if and only if there exists a reachable fixed point π^* such that the ith entry of π^* is nonzero.

2.1.2 Information Theoretic Notations.

For a probability distribution p over discrete set A, its entropy is defined as $H(p) = -\sum_{a\in A}p(a)\log p(a)$. For two probability distributions p and q over A, their Kullback-Leibler (KL) divergences denoted as $D_{KL}(p,q)$, is defined as $D_{KL}(p,q) = \sum_{a\in A}p(a)\log\frac{p(a)}{q(a)}$. Given $\lambda_1 > 0$ and $\lambda_2 > 0$ satisfying $\lambda_1 + \lambda_2 = 1$, the Jensen-Shannon Divergence (JSD) between p and q under the weights (λ_1, λ_2), is defined as $D(p,q) = \lambda_1 D_{KL}(p, \lambda_1 p + \lambda_2 q) + \lambda_2 D_{KL}(q, \lambda_1 p + \lambda_2 q)$, which is equivalent to $D(p,q) = H(\lambda_1 p + \lambda_2 q) - \lambda_1 H(p) - \lambda_2 H(q)$ (for more details, refer to [Cover and Thomas (2012); Lin (1991)]).

For two probability distributions p over A and q over B, their mutual information is defined as $I(p,q) = \sum_{a\in A, b\in B}Pr(a,b)\log\frac{Pr(a,b)}{p(a)q(b)}$, which can also be equivalently defined as $I(p,q) =$

$H(p) - H(p|q)$, where the conditional entropy $H(p|q)$ is given as $H(p|q) = -\sum_{b\in B} q(b) \sum_{a\in A} Pr(a|b) \log Pr(a|b)$.

2.1.3 Secret/non-Secret states

Certain system states may be considered sensitive and hence secret, whereas the remaining states act as covers for the secrets. Define $X_s \subseteq X$ as the set of secret states (reachable by secret I/O behaviors) whereas, $X_c \subseteq X$ is the set of cover states (reachable by cover I/O behaviors).

The events in Σ executed by the system are observed by an observer (an attacker or an adversary) through an observation mask $M : \overline{\Sigma} \to \overline{\Delta}$, where Δ is the set of observed symbols, and $M(\epsilon) = \epsilon$. (M can be extended to Σ^* as follows: $M(\epsilon) = \epsilon$ and $\forall s \in \Sigma^*, \sigma \in \overline{\Sigma}, M(s\sigma) = M(s)M(\sigma)$.) The following section describes the computation of an observer transition structure for G that can be used to track its evolution over its observed symbols Δ, and also the associated transition matrices $\{\Theta(\delta) \mid \delta \in \Delta\}$.

Remark 1. *If the desired secrecy property is specified for behaviors (as opposed to states) then a specification-based model-refinement step can be used to convert it to a state-based specification. Letting $L = L(G)$ denote the set of all behaviors (traces) of a stochastic PODES G as introduced in the notation section, suppose $K \subset L$ models the secret behaviors (also called a specification), while the remaining traces in $L - K$ act as its cover. K may be modeled by a deterministic acceptor $R = (Y, \Sigma, \beta, y_0)$ such that $L(R) = K$. By introducing a dump state D in R, and completing its transition function, we can obtain $\overline{R} = (\overline{Y}, \Sigma, \overline{\beta}, y_0)$, where $\overline{Y} = Y \cup D$, and $\forall \overline{y}, \overline{y}' \in \overline{Y}, \sigma \in \Sigma$,*

$$\overline{\beta}(\overline{y}, \sigma, \overline{y}') := \begin{cases} \beta(\overline{y}, \sigma, \overline{y}') \text{ if } (\overline{y}, \overline{y}' \in Y) \wedge (\beta(\overline{y}, \sigma, \overline{y}') > 0), \\ 1 \text{ if } [(\overline{y} = \overline{y}' = D) \vee (\overline{y}' = D \wedge \sum_{y\in Y} \beta(\overline{y}, \sigma, y) = 0)]. \end{cases}$$

Then, the system model can be refined with respect to the specification to identify the secret and cover behaviors as states *in the refined system $G^R = G||\overline{R}$, and is given by:*

$G^R = (X \times \overline{Y}, \Sigma, \gamma, (x_0, y_0))$, *where* $\forall (x, \overline{y}), (x', \overline{y}') \in X \times \overline{Y}, \sigma \in \Sigma$, γ *is defined as:*

$$\gamma((x,\overline{y}),\sigma,(x',\overline{y}')) := \begin{cases} \alpha(x,\sigma,x') \text{ if } [(\overline{y},\overline{y}' \in Y \wedge \beta(\overline{y},\sigma,\overline{y}') > 0) \vee (\overline{y} = \overline{y}' = D) \\ \qquad\qquad \vee \left(\overline{y}' = D \wedge \sum_{y \in Y} \beta(\overline{y},\sigma,y) = 0\right)], \\ 0 \text{ otherwise.} \end{cases}$$

2.2 Observer Transition Structure

We describe the computation of an observer transition structure that can be used to track the evolution of G over its observed symbols Δ, and the associated transition matrices $\{\Theta(\delta) \mid \delta \in \Delta\}$. Given the system model G (or equivalently the the refined system model G^R as in remark 1), and its observation mask $M : \overline{\Sigma} \to \overline{\Delta}$, define the set of traces originating at x, terminating at x' and executing a sequence of unobservable events followed by a single observable event with observation δ as $L_G(x, \delta, x') := \{s \in \Sigma^* | s = u\sigma, M(u) = \epsilon, M(\sigma) = \delta, \gamma(x, s, x') > 0\}$. Define its probability, $\alpha(L_G(x, \delta, x')) := \sum_{s \in L_G(x,\delta,x')} \gamma(x, s, x')$, and denote it as $\theta_{x,\delta,x'}$. Also, define $\lambda_{ij} = \sum_{\sigma \in \Sigma_{uo}} \gamma(i, \sigma, j)$ as the probability of transitioning from x to x' while executing a single unobservable event. Then, letting $i = x$ and $j = x'$, $\theta_{i,\delta,j} = \sum_k \lambda_{ik}\theta_{k,\delta,j} + \sum_{\sigma \in \Sigma: M(\sigma) = \delta} \gamma(i, \sigma, j)$, where the first term on the right hand side (RHS) corresponds to transitioning in at least two steps (i to intermediate k unobservably, and k to j with a single observation δ at the end), whereas the second term on RHS corresponds to transitioning in exactly one step [Chen et al. (2015); Ibrahim et al. (2015)]. Thus, for each $\delta \in \Delta$, all the probabilities $\{\theta_{i,\delta,j} | i, j \in X\}$ can be found by solving the following matrix equation [Wang and Ray (2004)]:

$$\Theta(\delta) = \Lambda\Theta(\delta) + \Gamma(\delta), \tag{2.1}$$

where $\Theta(\delta), \Lambda$ and $\Gamma(\delta)$ are all $|X| \times |X|$ square matrices whose ij^{th} elements are given by $\theta_{i,\delta,j}, \lambda_{ij}$ and $\sum_{\sigma \in \Sigma: M(\sigma) = \delta} \gamma(x, \sigma, x')$, respectively.

Example 1. *Fig. 2.1(a) is an example of a stochastic automaton G. The set of states is $X = \{0, 1, 2\}$ with initial state $x_0 = 0$, event set $\Sigma = \{a, b, c\}$. A state is depicted as a node, whereas a transition is depicted as an edge between its origin and termination states, with*

its event name and probability value labeled on the edge. The observation mask M is such that $M(c) = \epsilon$ and for all other events $\sigma \in \{a, b\}$, $M(\sigma) = \sigma$. Suppose R is given in Fig. 2.1(b), i.e., $K = L(R) = ab^*, L - K = ca^* \cup (ca^*b)^+ \cup (ca^*b)^+ab^*$. Then the refinement G^R automaton is shown in Fig. 2.1(c). Let the state space of G^R be indexed as the following order: $\{(0,0),(2,1),(1,D),(0,D),(2,D)\}$. Then, by solving (2.1) we get:

$$\Theta(a) = \begin{bmatrix} 0 & 0.5 & 0.375 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.75 & 0 & 0 \\ 0 & 0 & 0.375 & 0 & 0.5 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\Theta(b) = \begin{bmatrix} 0 & 0 & 0 & 0.125 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.25 & 0 \\ 0 & 0 & 0 & 0.125 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

2.3 Illustrative Example: AES Side-Channel Attack

We consider a version of cache side-channel attack that can be used to compromise AES (Advanced Encryption Standard), adopted from [Zhang and ruby B. Lee (2014)]. The difference in access times of cache hit versus miss may be used to learn the AES key as described below.

AES is a symmetric crypto-system, which processes data blocks of 16, 24, or 32 bytes, using encryption keys of the same size as data, corresponding to "AES-16", "AES-24", or "AES-32". In what follows below, we consider AES-16 for illustration purposes. For encryption, the plain-text block is converted into the cipher-text block, both viewed as 4×4 array of bytes, in several rounds. The intermediate results of rounds are also of same sizes, and are termed "states". (For AES-16, the number of rounds Nr equals 10 [Kak (2015); Daemen and Rijmen (1999)].) For setting up the keys for the various rounds, a key expansion algorithm is applied to an initial

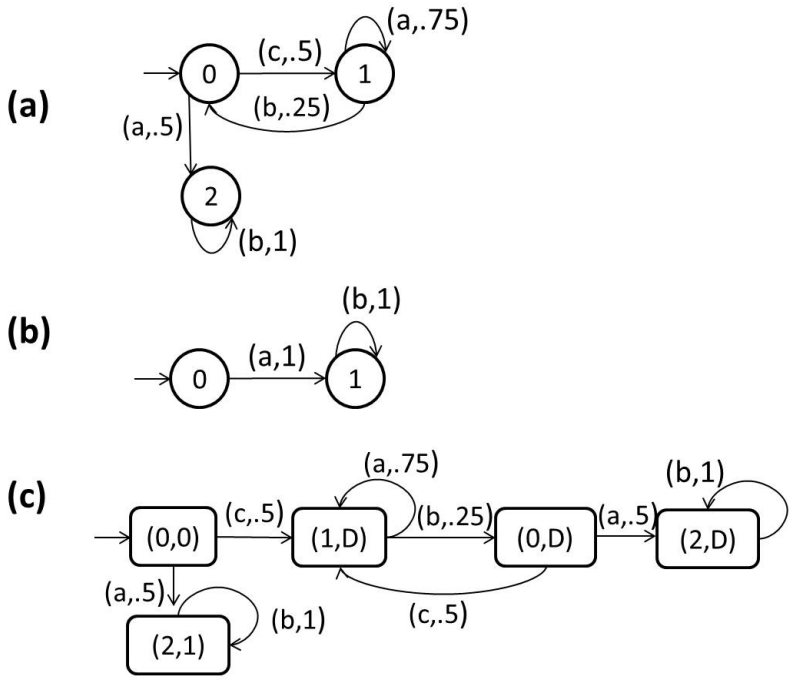

Figure 2.1 *(a) Stochastic automaton G, (b) deterministic secret specification R, (c) refinement G^R*

key $K^{(0)}$, outputting a linear array of 4-byte words, of length $4Nr$, corresponding to the keys $\{K^{(r)}, r = 1, ..., N_r\}$ for the future rounds.

Starting from a 16-byte plain-text $P = (p_0, ..., p_{15})$, encryption proceeds by computing a 16-byte intermediate state $x^{(r)} = (x_0^{(r)}, ..., x_{15}^{(r)})$ at each round r. The initial state $x^{(0)}$ is computed by $x_i^{(0)} = p_i \oplus k_i,\ i = 0, ..., 15$, and the next $Nr - 1$ rounds for $r = 0, ..., Nr - 2$ are computed as follows:

$$
\begin{aligned}
(x_0^{(r+1)},\ x_1^{(r+1)},\ x_2^{(r+1)},\ x_3^{(r+1)}) \leftarrow\ & T_0[x_0^{(r)}] \oplus T_1[x_5^{(r)}] \oplus T_2[x_{10}^{(r)}] \oplus T_3[x_{15}^{(r)}] \oplus K_0^{(r+1)} \\
(x_4^{(r+1)},\ x_5^{(r+1)},\ x_6^{(r+1)},\ x_7^{(r+1)}) \leftarrow\ & T_0[x_4^{(r)}] \oplus T_1[x_9^{(r)}] \oplus T_2[x_{14}^{(r)}] \oplus T_3[x_3^{(r)}] \oplus K_1^{(r+1)} \\
(x_8^{(r+1)},\ x_9^{(r+1)},\ x_{10}^{(r+1)},\ x_{11}^{(r+1)}) \leftarrow\ & T_0[x_8^{(r)}] \oplus T_1[x_{13}^{(r)}] \oplus T_2[x_2^{(r)}] \oplus T_3[x_7^{(r)}] \oplus K_2^{(r+1)} \\
(x_{12}^{(r+1)},\ x_{13}^{(r+1)},\ x_{14}^{(r+1)},\ x_{15}^{(r+1)}) \leftarrow\ & T_0[x_{12}^{(r)}] \oplus T_1[x_1^{(r)}] \oplus T_2[x_6^{(r)}] \oplus T_3[x_{11}^{(r)}] \oplus K_3^{(r+1)}, \qquad (2.2)
\end{aligned}
$$

where the notation $T_n[m]$ denotes the index-m entry of table T_n that is used to store pre-computed transformations of states, involving the operations of substitute-bytes, shift-rows,

and mix-columns. The last round is also computed using (2.2), except that tables $T_0, ..., T_3$ are replaced by tables $T_0^{(10)}, ..., T_3^{(10)}$, respectively. (The last round does not need the mix-columns operation and so uses different tables.)

An attacker may populate a cache line with an initial state $x_i = p_i \oplus k_i$, generated using a known plain-text p_i and a known key k_i, $i = 0, ..., 15$. When the host populates the same cache line with another initial state $x'_i = p'_i \oplus k'_i$, using another plain-text p'_i, also known to the attacker, and a key k'_i that is unknown to the attacker, a cache hit, as indicated by a shorter access time, can indicate $x_i = x'_i$, implying $p_i \oplus k_i = p'_i \oplus k'_i$, from which the attacker can infer the unknown key, $k'_i = p'_i \oplus p_i \oplus k_i$. Thus each cache hit, which may be thought of host's cache line interfering with the attacker's cache line, provides an opportunity for an attacker to infer one byte of the key used by a host.

To provide additional protection against this vulnerability, the system may introduce random evictions of the cache. Figures 2.2(a), and 2.2(b) show the abstracted versions of the two cache architectures, with no protection and with added protection, respectively, where the models track the status of an individual cache line. (Similar models track other cache lines.) The 4 states in Fig. 2.2(a) are: "A" (occupied by the attacker and of low confidentiality), "H", "A_H" (occupied, respectively, by the host, and the attacker while occupied by the host in the previous step, both of high confidentiality), or "I' (invalid—that has no valid contents from attacker or host, and also of low confidentiality). If the host holds its own data in cache, its cache access results in a hit (H_{hit}), but if the attacker evicts the host's data in the cache lines by requesting cache access, it results in a miss (H_{miss}). The attackers cache hit and miss, A_{hit} and A_{miss} are dually defined. Note that the occurrence of A_{miss} or A_{hit} can be used to infer "H" or "A_H" states, using which one byte of the encryption key can be compromised. However, an attacker can only observe its own cache hits and misses (i.e., A_{hit} and A_{miss} are the only observable events). In Fig. 2.2(b), random cache eviction is introduced by the system to invalidate the data, denoted by "Inv" event. This introduces ambiguity in the attacker's knowledge about the occupancy of the cache, i.e., when it observes a cache miss, it does not know whether it is due to the processor's eviction or due to the host's cache access. Then, in Fig. 2.2(b), we can view $\{H, A_H\}$ to be the high confidential or "secret" states whereas

$\{I, A, I'\}$ to be the low confidential or "cover" states, which present ambiguity against the "secret" states.

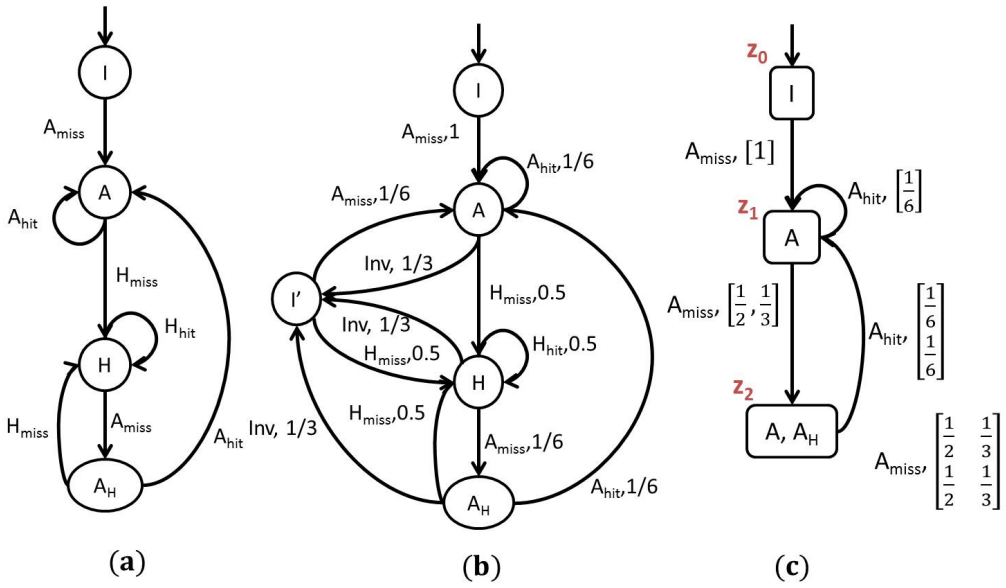

Figure 2.2 (a) Cache side-channel attack model with no evictions, (b) cache side-channel attack model with random evictions, (c) observer for the cache side-channel attack with random evictions

A goal of secrecy or confidentiality is to ensure that with high probability, the secret states are masked by the cover states under system observations. We present an approach to quantify the ability to keep the secrets confidential for probabilistic systems. This employs an information theoretic approach to measure the amount of information leaked about the secrets into the observables. The details are provided in the next sections.

2.4 Jensen-Shannon Divergence Based Secrecy Quantification

The statistical difference between the conditional distributions of secrets versus covers over the system observations of a common length, provides a measure of the amount of secrecy leaked by a system. A possible way of measuring difference between two distributions is the JSD (Jensen Shannon Divergence) measure. Here we present a way to compute the JSD measure for stochastic PODESs.

Given a length-n observation $o \in \Delta^n$, let $p_n(o)$ denote its probability. Then, since the occurrences of observations of length n are mutually disjoint, $\sum_{o\in\Delta^n} p_n(o) = 1$, i.e., p_n is a probability distribution over Δ^n. Then its entropy is given as:

$$H(p_n) = -\sum_{o\in\Delta^n} p_n(o)\log p_n(o).$$

Lemma 1. *The entropy p_n as defined above for length-n observation can be recursively computed as follows:*

$$H(p_n) = H(p_{n-1}) - \sum_{o\in\Delta^{n-1}} p_{n-1}(o)\sum_{\delta\in\Delta} Pr(\delta|o)\log Pr(\delta|o).$$

Proof.

$$\begin{aligned}
H(p_n) &= -\sum_{o\in\Delta^n} p_n(o)\log p_n(o)\\
&= -\sum_{o\in\Delta^{n-1}}\sum_{\delta\in\Delta} p_n(o\delta)\log p_n(o\delta)\\
&= -\sum_{o\in\Delta^{n-1}}\sum_{\delta\in\Delta} p_{n-1}(o)Pr(\delta|o)\log(p_{n-1}(o)Pr(\delta|o))\\
&= -\sum_{o\in\Delta^{n-1}} p_{n-1}(o)\sum_{\delta\in\Delta} Pr(\delta|o)(\log p_{n-1}(o) + \log Pr(\delta|o))\\
&= -\sum_{o\in\Delta^{n-1}} p_{n-1}(o)\sum_{\delta\in\Delta} Pr(\delta|o)\log Pr(\delta|o)\\
&\quad -\sum_{o\in\Delta^{n-1}} p_{n-1}(o)\sum_{\delta\in\Delta} Pr(\delta|o)\log p_{n-1}(o)\\
&= -\sum_{o\in\Delta^{n-1}} p_{n-1}(o)\sum_{\delta\in\Delta} Pr(\delta|o)\log Pr(\delta|o)\\
&\quad -\sum_{o\in\Delta^{n-1}} p_{n-1}(o)\log p_{n-1}(o)\sum_{\delta\in\Delta} Pr(\delta|o)\\
&= -\sum_{o\in\Delta^{n-1}} p_{n-1}(o)\sum_{\delta\in\Delta} Pr(\delta|o)\log Pr(\delta|o)\\
&\quad -\sum_{o\in\Delta^{n-1}} p_{n-1}(o)\log p_{n-1}(o)\\
&= -\sum_{o\in\Delta^{n-1}} p_{n-1}(o)\sum_{\delta\in\Delta} Pr(\delta|o)\log Pr(\delta|o) + H(p_{n-1}).
\end{aligned}$$

Thus Lemma 1 is established. □

Define two more probability distributions over Δ^n: probability that an observation $o \in \Delta^n$ is generated by some secret in X_s, denoted $p^s_n(o)$, versus that is generated by some cover in X_c,

denoted $p_n^c(o)$. The entropy of p_n^s and p_n^c are given, respectively, by:

$$H(p_n^s) = -\sum_{o\in\Delta^n} p_n^s(o)\log p_n^s(o) \tag{2.3}$$

$$H(p_n^c) = -\sum_{o\in\Delta^n} p_n^c(o)\log p_n^c(o). \tag{2.4}$$

The ability of an intruder to identify secret versus cover behaviors based on observations of length n, depends on the disparity between the two distributions p_n^s versus p_n^c: If p_n^s and p_n^c are identical, i.e., with "zero disparity", there is no way to statistically tell apart secrets from covers, and in that case there is perfect secrecy. However, when p_n^s and p_n^c are different, then one could characterize the ability of an intruder to discriminate secrets from covers based on length-n observations, using the JSD between p_n^s and p_n^c, under the weights $(\lambda_n^s, \lambda_n^c)$, denoted

$$\begin{aligned} D(p_n^s, p_n^c) &= \lambda_n^s D_{KL}(p_n^s, \lambda_n^s p_n^s + \lambda_n^c p_n^c) + \lambda_n^c D_{KL}(p_n^c, \lambda_n^s p_n^s + \lambda_n^c p_n^c) \\ &= H(\lambda_n^s p_n^s + \lambda_n^c p_n^c) - \lambda_n^s H(p_n^s) - \lambda_n^c H(p_n^c), \end{aligned} \tag{2.5}$$

where λ_n^s is the probability of secrets and λ_n^c is the probability of covers, respectively, generating length-n observations. Note that JSD is symmetric in its arguments and bounded by 0 and 1. The following theorem shows that the JSD measure is indeed a useful measure of information revealed, as it equals the mutual information between the observations p_n and the status (whether secret or cover) of system executions. This status can be captured by a bi-valued random variable Λ_n, defined for each $n \in \mathbb{N}$, such that $Pr((\Lambda_n = 1) \equiv [x \in X_s]) = \lambda_n^s$ and $Pr((\Lambda_n = 0 \equiv [x \in X_c])) = \lambda_n^c$.

Theorem 1. *The JSD as defined in (2.5) is equivalent to the mutual information of Λ_n and p_n, i.e.,*

$$D(p_n^s, p_n^c) = I(\Lambda_n, p_n).$$

Proof. According to the definition of mutual information, we have

$$I(\Lambda_n, p_n) = H(p_n) - H(p_n|\Lambda_n).$$

The conditional entropy $H(p_n|\Lambda_n)$ can be expressed as follows:

$$\begin{aligned}
H(p_n|\Lambda_n) &= Pr(\Lambda_n = 1)H(p_n|\Lambda_n = 1) + Pr(\Lambda_n = 0)H(p_n|\Lambda_n = 0) \\
&= -\lambda_n^s \sum_{o\in\Delta^n} Pr(o|\Lambda_n = 1)\log Pr(o|\Lambda_n = 1) - \lambda_n^c \sum_{o\in\Delta^n} Pr(o|\Lambda_n = 0)\log Pr(o|\Lambda_n = 0) \\
&= -\lambda_n^s \sum_{o\in\Delta^n} p_n^s(o)\log p_n^s(o) - \lambda_n^c \sum_{o\in\Delta^n} p_n^c(o)\log p_n^s(o) \\
&= \lambda_n^s H(p_n^s) + \lambda_n^c H(p_n^c),
\end{aligned}$$

where we utilize the fact that $Pr(o|\Lambda_n = 1) = p_n^s(o)$ and $Pr(o|\Lambda_n = 0) = p_n^c(o)$. Substituting $H(p_n|\Lambda_n)$ into the definition of mutual information $I(\Lambda_n, p_n)$, and considering relationship in (2.5), we have

$$I(\Lambda_n, p_n) = H(p_n) - \lambda_n^s H(p_n^s) - \lambda_n^c H(p_n^c) = D(p_n^s, p_n^c).$$

Thus, the proof is completed. □

When $D(p_n^s, p_n^c) = I(\Lambda_n, p_n) = 0$, length-$n$ observations are independent of system execution status, and thus no secret information can be leaked through length-n observations. On the other hand, when $D(p_n^s, p_n^c) = I(\Lambda_n, p_n) > 0$, the dependence of length-n observations and system status can be measured by the JSD, $D(p_n^s, p_n^c)$, which in turn quantifies the extent to which system secrecy can be leaked by length-n observations.

2.5 Recursive Characterization

An intruder is likely to discriminate more if he/she observes for a longer period, and accordingly, our goal is to evaluate the worst-case loss of secrecy, as obtained in the limit: $\lim_{n\to\infty} D(p_n^s, p_n^c)$. This worst-case JSD provides an upper bound to quantify the amount of information leaked about secrets. To compute the worst-case loss of secrecy, we first develop a recursive computation for $D(p_n^s, p_n^c)$, relating it to distributions of length-$(n-1)$ observations and divergence of length-1 distributions. For $o \in \Delta^*$ and $\delta \in \Delta$, define the distributions of secret versus cover upon a single observation δ following a history of observation o: $p^{s|o}(\delta), p^{c|o}(\delta)$. Further, define $\lambda^{s|o}$ as the probability of secrets and $\lambda^{c|o}$ as the probability of covers, respectively, generating length-1 observations following a history of observation o. Following the

definition of JSD, we have

$$D(p^{s|o}, p^{c|o}) = H(\lambda^{s|o}p^{s|o} + \lambda^{c|o}p^{c|o}) - \lambda^{s|o}H(p^{s|o}) - \lambda^{c|o}H(p^{c|o}). \tag{2.6}$$

The following lemma characterizes the computation of the length-1 JSD given observation o

Lemma 2. *Given observation o, the length-1 JSD between $p^{s|o}$ and $p^{c|o}$ can be computed as:*

$$D(p^{s|o}, p^{c|o}) = H(\lambda^{s|o}p^{s|o} + \lambda^{c|o}p^{c|o}) + H(\{\lambda^{s|o}, \lambda^{c|o}\}) - H(\lambda^{s|o}p^{s|o}) - H(\lambda^{c|o}p^{c|o}), \tag{2.7}$$

Proof. By expanding (2.6), we have

$$\begin{aligned}
D(p^{s|o}, p^{c|o}) &= H(\lambda^{s|o}p^{s|o} + \lambda^{c|o}p^{c|o}) + \sum_{\delta\in\Delta} \lambda^{s|o}p^{s|o}(\delta) \log \frac{\lambda^{s|o}p^{s|o}(\delta)}{\lambda^{s|o}} \\
&\quad + \sum_{\delta\in\Delta} \lambda^{c|o}p^{c|o}(\delta) \log \frac{\lambda^{c|o}p^{c|o}(\delta)}{\lambda^{c|o}} \\
&= H(\lambda^{s|o}p^{s|o} + \lambda^{c|o}p^{c|o}) + \sum_{\delta\in\Delta} \lambda^{s|o}p^{s|o}(\delta) \log \lambda^{s|o}p^{s|o}(\delta) \\
&\quad + \sum_{\delta\in\Delta} \lambda^{c|o}p^{c|o}(\delta) \log \lambda^{c|o}p^{c|o}(\delta) - \lambda^{s|o} \log \lambda^{s|o} \left(\sum_{\delta\in\Delta} p^{s|o}(\delta) \right) \\
&\quad - \lambda^{c|o} \log \lambda^{c|o} \left(\sum_{\delta\in\Delta} p^{c|o}(\delta) \right).
\end{aligned}$$

Since $\left(\sum_{\delta\in\Delta} p^{s|o}(\delta)\right) = \left(\sum_{\delta\in\Delta} p^{c|o}(\delta)\right) = 1$, we have

$$\begin{aligned}
D(p^{s|o}, p^{c|o}) &= H(\lambda^{s|o}p^{s|o} + \lambda^{c|o}p^{c|o}) + \sum_{\delta\in\Delta} \lambda^{s|o}p^{s|o}(\delta) \log \lambda^{s|o}p^{s|o}(\delta) \\
&\quad + \sum_{\delta\in\Delta} \lambda^{c|o}p^{c|o}(\delta) \log \lambda^{c|o}p^{c|o}(\delta) - \lambda^{s|o} \log \lambda^{s|o} - \lambda^{c|o} \log \lambda^{c|o} \\
&= H(\lambda^{s|o}p^{s|o} + \lambda^{c|o}p^{c|o}) + H(\{\lambda^{s|o}, \lambda^{c|o}\}) \\
&\quad - H(\lambda^{s|o}p^{s|o}) - H(\lambda^{c|o}p^{c|o}).
\end{aligned}$$

Thus, Lemma 2 is established. □

Using the above lemma, we can next provide the following recursive computation for JSD.

Lemma 3. *The JSD over the distribution of secret and that of cover under length-n observations can be recursively computed by:*

$$D(p_n^s, p_n^c) = H(\{\lambda_n^s, \lambda_n^c\}) + \sum_{o\in\Delta^{n-1}} p_{n-1}(o) \Big[-H(\{\lambda^{s|o}, \lambda^{c|o}\}) + D(p^{s|o}, p^{c|o}) \Big]. \tag{2.8}$$

Proof. We first define some notations, for simplicity of presentation, as follows:

$$\begin{aligned}
\widetilde{p}^s_n(o) &= \lambda^s_n p^s_n(o) \\
\widetilde{p}^c_n(o) &= \lambda^c_n p^c_n(o) \\
\widetilde{p}^s(\delta|o) &= \lambda^{s|o} p^{s|o}(\delta) \\
\widetilde{p}^c(\delta|o) &= \lambda^{c|o} p^{c|o}(\delta) \\
p(\delta|o) &:= \lambda^{s|o} p^{s|o}(\delta) + \lambda^{c|o} p^{c|o}(\delta) = Pr(\delta|o).
\end{aligned}$$

We start by deriving a recursive computation for $H(p^s_n)$ as defined in (2.3), as follows:

$$\begin{aligned}
H(p^s_n) &= -\sum_{o\in\Delta^n} p^s_n(o)\log p^s_n(o) \\
&= -\frac{1}{\lambda^s_n}\sum_{o\in\Delta^n} \widetilde{p}^s_n(o)\log\frac{\widetilde{p}^s_n(o)}{\lambda^s_n} \\
&= -\frac{1}{\lambda^s_n}\sum_{o\in\Delta^{n-1}}\sum_{\delta\in\Delta} \widetilde{p}^s_n(o\delta)\log\frac{\widetilde{p}^s_n(o\delta)}{\lambda^s_n} \\
&= -\frac{1}{\lambda^s_n}\sum_{o\in\Delta^{n-1}}\sum_{\delta\in\Delta} p_{n-1}(o)\widetilde{p}^s(\delta|o)\log\frac{p_{n-1}(o)\widetilde{p}^s(\delta|o)}{\lambda^s_n} \\
&= -\frac{1}{\lambda^s_n}\sum_{o\in\Delta^{n-1}} p_{n-1}(o)\sum_{\delta\in\Delta}\widetilde{p}^s(\delta|o) \\
&\qquad\qquad \Big[\log p_{n-1}(o) + \log(p^{s|o}(\delta)\lambda^{s|o}) - \log\lambda^s_n\Big] \\
&= -\frac{1}{\lambda^s_n}\sum_{o\in\Delta^{n-1}} p_{n-1}(o)\log p_{n-1}(o)\sum_{\delta\in\Delta}\widetilde{p}^s(\delta|o) \\
&\quad -\frac{1}{\lambda^s_n}\sum_{o\in\Delta^{n-1}} p_{n-1}(o)\sum_{\delta\in\Delta}\widetilde{p}^s(\delta|o)\log(p^{s|o}(\delta)\lambda^{s|o}) \\
&\quad +\frac{1}{\lambda^s_n}\sum_{o\in\Delta^{n-1}} p_{n-1}(o)\sum_{\delta\in\Delta}\widetilde{p}^s(\delta|o)\log\lambda^s_n \\
&= -\frac{1}{\lambda^s_n}\sum_{o\in\Delta^{n-1}} p_{n-1}(o)\log p_{n-1}(o)\lambda^{s|o} + \log\lambda^s_n \\
&\quad +\frac{1}{\lambda^s_n}\sum_{o\in\Delta^{n-1}} p_{n-1}(o)H(\lambda^{s|o}p^{s|o}).
\end{aligned}$$

Similarly, $H(p^c_n)$ as defined in (2.4) can be recursively characterized as:

$$\begin{aligned}
H(p^c_n) = &-\frac{1}{\lambda^c_n}\sum_{o\in\Delta^{n-1}} p_{n-1}(o)\log p_{n-1}(o)\lambda^{c|o} + \log\lambda^c_n \\
&+\frac{1}{\lambda^c_n}\sum_{o\in\Delta^{n-1}} p_{n-1}(o)H(\lambda^{c|o}p^{c|o}).
\end{aligned}$$

Expand (2.5) using the above recursion and Lemma 1, yielding the follows:

$$
\begin{aligned}
D(p_n^s, p_n^c) &= H(p_n) - \lambda_n^s H(p_n^s) - \lambda_n^c H(p_n^c) \\
&= H(p_n) - \lambda_n^s \log \lambda_n^s - \lambda_n^c \log \lambda_n^c + \sum_{o \in \Delta^{n-1}} p_{n-1}(o) \log p_{n-1}(o) \lambda^{s|o} \\
&\quad - \sum_{o \in \Delta^{n-1}} p_{n-1}(o) H(\lambda^{s|o} p^{s|o}) + \sum_{o \in \Delta^{n-1}} p_{n-1}(o) \log p_{n-1}(o) \lambda^{c|o} \\
&\quad - \sum_{o \in \Delta^{n-1}} p_{n-1}(o) H(\lambda^{c|o} p^{c|o}) \\
&= H(p_n) + H(\{\lambda_n^s, \lambda_n^c\}) + \sum_{o \in \Delta^{n-1}} p_{n-1}(o) \log p_{n-1}(o) \\
&\quad + \sum_{o \in \Delta^{n-1}} p_{n-1}(o) \Big[-H(\lambda^{s|o} p^{s|o}) - H(\lambda^{c|o} p^{c|o}) \Big] \\
&= H(\{\lambda_n^s, \lambda_n^c\}) + \sum_{o \in \Delta^{n-1}} p_{n-1}(o) \Bigg[- \sum_{\delta \in \Delta} p(\delta|o) \log p(\delta|o) \\
&\quad - H(\lambda^{s|o} p^{s|o}) - H(\lambda^{c|o} p^{c|o}) \Bigg] \\
&= H(\{\lambda_n^s, \lambda_n^c\}) + \sum_{o \in \Delta^{n-1}} p_{n-1}(o) \Big[H(\lambda^{s|s} p^{s|s} + \lambda^{c|o} p^{c|o}) \\
&\quad - H(\lambda^{s|o} p^{s|o}) - H(\lambda^{c|o} p^{c|o}) \Big]
\end{aligned}
$$

Finally, by substituting (2.7) in Lemma 2, we have

$$
D(p_n^s, p_n^c) = H(\{\lambda_n^s, \lambda_n^c\}) + \sum_{o \in \Delta^{n-1}} p_{n-1}(o) \Big[-H(\{\lambda^{s|o}, \lambda^{c|o}\}) + D(p^{s|o}, p^{c|o}) \Big].
$$

Thus, Lemma 3 is established. □

The computation of JSD for a PODES is challenging since a finite-state DES under partial observations is potentially infinite-state (with the state-space being conditional state distributions following observations). However, a finite-state observer representation is possible, which we construct and employ for divergence computation. First we map the JSD computation to a computation based on the state-distribution following an observation. Each observation $o \in \Delta^*$ results in a conditional state distribution $\pi(o)$, which can be computed recursively as follows: for any $o \in \Delta^*, \delta \in \Delta$: $\pi(\epsilon) = \pi_0$ and $\pi(o\delta) = \frac{\pi(o) \times \Theta(\delta)}{||\pi(o) \times \Theta(\delta)||}$ [Chen and Kumar (2015)], where π_0 is the initial state distribution, whereas the computation of transition matrix $\Theta(\delta)$ is given in Section 2.2. Let Π denote the set of all such conditional state distributions, and for each $\pi \in \Pi$ and $n \in \mathbb{N}$, denote $P_n(\pi) = Pr(o \in \Delta^n : \pi(o) = \pi)$, which is the probability that the set

of all observations of length n, upon which the conditional state distribution is π. For a state distribution π, define the following notations:

$$\lambda^{s|\pi} := \sum_{\delta \in \Delta} \pi\Theta(\delta)\mathcal{I}^s, \ \lambda^{c|\pi} := \sum_{\delta \in \Delta} \pi\Theta(\delta)\mathcal{I}^c$$
$$p^{s|\pi}(\delta) := \frac{\pi\Theta(\delta)\mathcal{I}^s}{\lambda^{s|\pi}}, \ p^{c|\pi}(\delta) := \frac{\pi\Theta(\delta)\mathcal{I}^c}{\lambda^{c|\pi}},$$

where $\mathcal{I}^s$ and $\mathcal{I}^c$ denote indicator column vectors of same size as number of states, with binary entries to identify the secret versus cover states (states reached by traces in K versus $L - K$). The length-1 JSD, conditioned upon a current state distribution π (similar to (2.6)), is given by:

$$D(p^{s|\pi}, p^{c|\pi}) = H(\lambda^{s|\pi}p^{s|\pi} + \lambda^{c|\pi}p^{c|\pi}) - \lambda^{s|\pi}H(p^{s|\pi}) - \lambda^{c|\pi}H(p^{c|\pi}).$$

Following the definitions introduced above, and the recursion result in Lemma 3, the next lemma can be obtained, which characterizes the recursive computation of JSD based on distributions over system state space.

Lemma 4. *For a stochastic DES G, the JSD over the distribution of secret and that of cover under length-n observations, as given in (2.5), can be rewritten as:*

$$D(p_n^s, p_n^c) \quad = \quad H(\{\lambda_n^s, \lambda_n^c\}) \ + \ \sum_{\pi \in \Pi} P_{n-1}(\pi)\Big[-H(\{\lambda^{s|\pi}, \lambda^{c|\pi}\}) \ + \ D(p^{s|\pi}, p^{c|\pi})\Big]. \quad (2.9)$$

In the limit when $n \to \infty$, if the distribution $P_n(\cdot)$ over Π converges to $P^*(\cdot)$, then $\lim_{n\to\infty} D(p_n^s, p_n^c)$ exists. See for example [Kaijser (1975)] for a condition under which such a convergence is guaranteed.

2.6 Observer-based Computation of Worst-case Secrecy Loss

For an observer *Obs*, the computation of $\lim_{n\to\infty} D(p_n^s, p_n^c)$ using (2.9), requires the computation of $\lim_{n\to\infty} P_{n-1}(\pi)$ which can be accomplished with the help of an observer that we introduce next. An observer tracks the possible system states following each observation, and also allows the computation of the corresponding state distribution. We let *Obs* denote an observer automaton with state set $Z \subseteq 2^X$, so that each node $z \in Z$ of the observer is a subset of system states, i.e., $z \subseteq X$, and we use $|z|$ to denote the number of system states in z. *Obs*

is initialized at node $z_0 = \{x_0\}$, and there is a transition labeled with $\delta \in \Delta$ from node z to z' if and only if every element of z' is reachable from some elements of z along a trace that ends in the only observation δ. Associated with this transition is the transition probability matrix $\Theta_{z,\delta,z'}$ of size $|z|$ by $|z'|$ (a sub-matrix of $\Theta(\delta)$ matrix given in Section 2.2), whose ijth element $\theta_{i,\delta,j}$ is given by the transition probability from ith element x of z to jth element x' of z' while producing the observation δ, and equals $\alpha(L_G(x, \delta, x'))$.

Example 2. *Consider the system, specification and refinement models of Fig. 2.3(a), (b) and (c), respectively, where $M(u) = \epsilon$, $M(a) = a$ and $M(b) = b$. Then, the corresponding observer Obs is given in Fig. 2.3(d), where each state in observer is a subset of states of the refined-system G^R, and transitions are on observed events that are labeled by their occurrence transition probability matrices.*

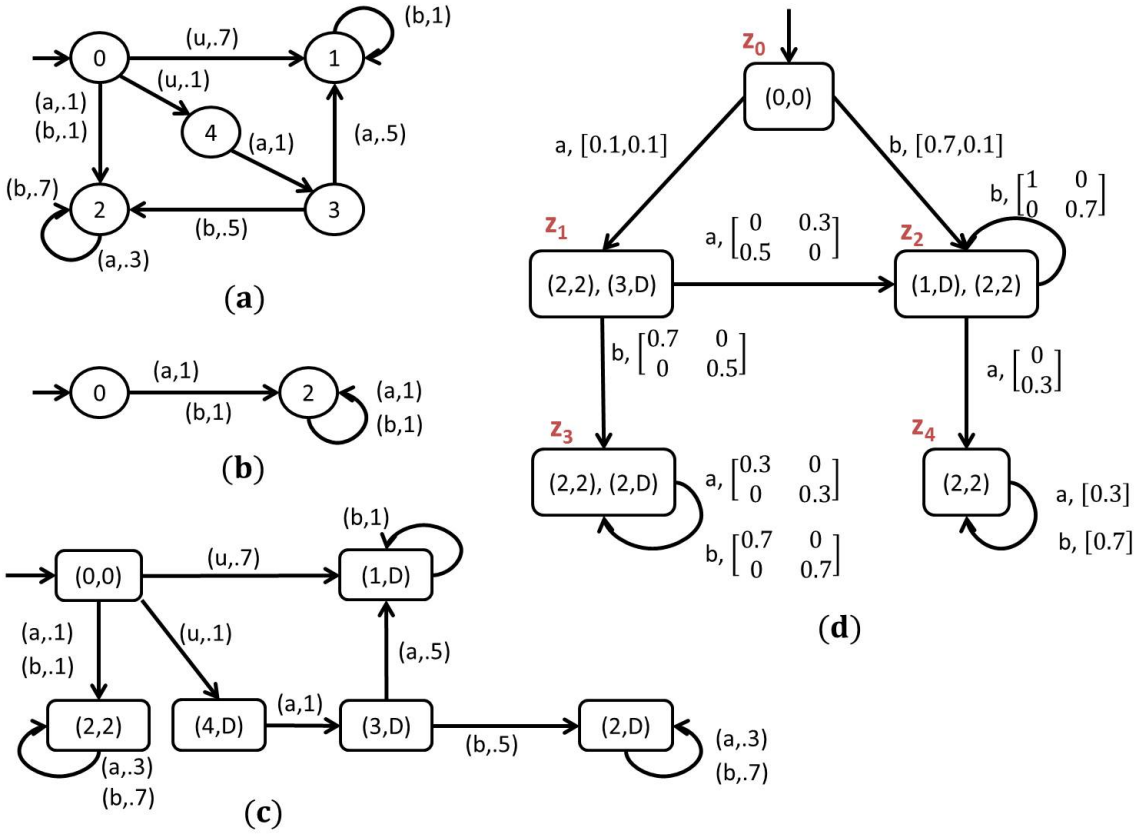

Figure 2.3 (a) System model G, (b) specification for secrets, R, (c) refined system model G^R, (d) observer model

Associated with each observation $o \in \Delta^*$, there is a reachable state distribution $\pi(o)$ as discussed earlier. Let the state z be reached in *Obs* following observation o. Then obviously the number of positive elements of $\pi(o)$ is the same as the number of elements in z. Then with a slight abuse of notation, we also use $\pi(o)$ to denote the row-vector containing only positive elements, and of same size as the number of elements in the node reached by o in *Obs*. Then $\pi(o)$ can also be recursively computed as follows: for any $o \in \Delta^*, \delta \in \Delta$: $\pi(\epsilon) = 1$ and $\pi(o\delta) = \frac{\pi(o) \times \Theta_{z_o,\delta,z_{o\delta}}}{||\pi(o) \times \Theta_{z_o,\delta,z_{o\delta}}||}$, where z_o and $z_{o\delta}$ are the nodes reached in *Obs* following o and $o\delta$ respectively. Then it can be seen that along any cycle in *Obs*, the distribution upon completing the cycle is a function of the distribution upon entering the cycle, through a sequence of transition matrix-multiplications and their normalization. In case of steady-state, those two distributions will be the same, namely, a fixed point of that function.

Given the *Obs* with state space Z for system G with secret X_s, and cover X_c, let $\widetilde{\Theta}$ be a $(\sum_z |z|) \times (\sum_z |z|)$ square matrix, whose ijth block is the $|z_i| \times |z_j|$ matrix $\sum_\delta \Theta_{z_i,\delta,z_j}$. The fix point distribution associated with $\widetilde{\Theta}$ can be obtained by solving $\pi^* = \pi^* \widetilde{\Theta}$, where π^* is a row vector of size $\sum_z |z|$. For each $z_i \in Z$, let $p(z_i)$ be the summation of the ith block of π^*, then z_i is said to be *recurrent* if $p(z_i) > 0$. Also note that for each $z \in Z$, exists a sufficiently large N such that $p(z) = \sum_{o \in \Delta^N:\ o \text{ reaches } z} p_N(o)$. In other words, $p(z)$ is the probability of all sufficiently long observations that reach the observer state z. With a slight abuse of notations, define λ^s as the summation of the elements of π^* corresponding to secret states, i.e., $\lambda^s := \pi^* \mathcal{I}^s$, and $\lambda^c = 1 - \lambda^s$.

For a set of recurrent nodes $\{z_1, z_2, \ldots, z_n\}$ that forms a SCC in *Obs*, define a set of distributions $\{\pi^*_{z_1}, \pi^*_{z_2}, \ldots, \pi^*_{z_n}\}$ to be a set of steady-state distributions if $\forall i, j, \delta$ such that Θ_{z_i,δ,z_j} is defined, the following holds:

$$\pi^*_{z_j} = \frac{\pi^*_{z_i} \Theta_{z_i,\delta,z_j}}{||\pi^*_{z_i} \Theta_{z_i,\delta,z_j}||},$$

i.e., $\pi^*_{z_i}$ represents a steady-state conditional distribution following a single sufficiently long observation o, that reaches z_i. Note that in this case, any other extension of o that also reaches z_i will induce the same conditional distribution $\pi^*_{z_i}$. There may exist multiple set of steady-state distributions for a given set of recurrent nodes, denoted as $\{\{\pi^*_{z_1,k}, \ldots, \pi^*_{z_n,k}\}, k \in \mathbb{N}\}$. Then if

steady-state always exists, for any sufficiently long observation that reaches a recurrent node z, there exists $k \in \mathbb{N}$ such that $\pi(o) = \pi^*_{z,k}$. Denote $p(z,k) := Pr[\{o \mid o \text{ reaches } z \text{ and } \pi(o) = \pi^*_{z,k}\}]$. Note that when the set of steady state distributions is a singleton, and hence unique, $p(z,k) = p(z)$.

Let $\mathcal{I}^s_{z'}$ and $\mathcal{I}^c_{z'}$ be indicator column vectors with binary entries of size $|z'|$ for identifying, within z', the secret and cover states, respectively. For each steady-state distribution $\pi^*_{z,k}$ of each recurrent node z, define:

$$\lambda^{s|\pi^*_{z,k}} := \sum_{\delta\in\Delta} \pi^*_{z,k}\Theta_{z,\delta,z'}\mathcal{I}^s_{z'},\ \lambda^{c|\pi^*_{z,k}} := \sum_{\delta\in\Delta} \pi^*_{z,k}\Theta_{z,\delta,z'}\mathcal{I}^c_{z'}$$
$$p^{s|\pi^*_{z,k}}(\delta) := \frac{\pi^*_{z,k}\Theta_{z,\delta,z'}\mathcal{I}^s_{z'}}{\lambda^{s|\pi^*_{z,k}}},\ p^{c|\pi^*_{z,k}}(\delta) := \frac{\pi^*_{z,k}\Theta_{z,\delta,z'}\mathcal{I}^c_{z'}}{\lambda^{c|\pi^*_{z,k}}}.$$

In the following, we assume the existence of steady-state:

Assumption 1. *Assume that for any sufficiently long observations $o_1 \leq o_2$, if Obs reaches the same node following o_1 and o_2, then $\pi(o_1) = \pi(o_2)$.*

Then following the above definitions and Lemma 4, next theorem provides computation of $\lim_{n\to\infty} D(p^s_n, p^c_n)$, under Assumption 1.

Theorem 2. *Consider a system G with secret X_s, and cover X_c. Then under Assumption 1, the worst case secrecy loss, i.e., JSD between p^s_n and p^c_n when $n \to \infty$, is given by:*

$$\lim_{n\to\infty} D(p^s_n, p^c_n) = H(\{\lambda^s, \lambda^c\}) + \sum_{z:z \text{ is recurrent}} \sum_{k\in\mathbb{N}} p(z,k)\left[-H(\{\lambda^{s|\pi^*_{z,k}}, \lambda^{c|\pi^*_{z,k}}\}) + D(p^{s|\pi^*_{z,k}}, p^{c|\pi^*_{z,k}})\right].$$

The next assumption assumes that for each set of recurrent nodes in *Obs*, there only exists one set of steady-state distributions.

Assumption 2. *For each set of recurrent nodes in Obs, $k = 1$, i.e., the set of steady-state distributions is unique, so that $p(z,k) = p(z)$.*

Theorem 3. *Consider a system G with secret X_s, and cover X_c. Then under Assumptions 1 and 2, the worst case secrecy loss, i.e., JSD between p^s_n and p^c_n when $n \to \infty$, is given by:*

$$\lim_{n\to\infty} D(p^s_n, p^c_n) = H(\{\lambda^s, \lambda^c\}) + \sum_{z:z \text{ is recurrent}} p(z)\left[-H(\{\lambda^{s|\pi^*_z}, \lambda^{c|\pi^*_z}\}) + D(p^{s|\pi^*_z}, p^{c|\pi^*_z})\right].$$

Example 3. *We revisit Example 2. Then based on Obs of Fig. 2.3(d), the following computation illustrates the steps of JSD computation.*

1. $\sum_z |z| = 8$ *and so* $\widetilde{\Theta}$ *is a* 8×8 *matrix given as:*

$$\widetilde{\Theta} = \begin{bmatrix} 0 & 0.1 & 0.1 & 0.7 & 0.1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.3 & 0.7 & 0 & 0 \\ 0 & 0 & 0 & 0.5 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.7 & 0 & 0 & 0.3 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Then, $\pi^* = \begin{bmatrix} 0 & 0 & 0 & 0.75 & 0 & 0.07 & 0.05 & 0.13 \end{bmatrix}$. *Therefore,* $p(z_0) = p(z_1) = 0$, $p(z_2) = 0.75$, $p(z_3) = 0.12$ *and* $p(z_4) = 0.13$.

2. *Here* $\mathcal{I}^s = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}^T$, $\mathcal{I}^c = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}^T$. *And so,* $\lambda^s = 0.2$ *and* $\lambda^c = 0.8$.

3. *Here* z_2, z_3 *and* z_4 *are recurrent nodes, and each of them forms a SCC. We have* $\pi^*_{z_2} = [1\ 0]$, $\pi^*_{z_4} = [1]$, *and while there are multiple solutions to the equation set* $\pi^*_{z_3} = \frac{\pi^*_{z_3}\Theta_{z_3,a,z_3}}{\pi^*_{z_3}\Theta_{z_3,a,z_3}}$ *and* $\pi^*_{z_3} = \frac{\pi^*_{z_3}\Theta_{z_3,b,z_3}}{\pi^*_{z_3}\Theta_{z_3,b,z_3}}$, *only* $\pi^*_{z_3} = [0.5833\ 0.4167]$ *is reachable. Thus, each set of recurrent nodes is a singleton set, and each with a unique fixed-point distribution. Therefore, for each recurrent node* z, $p(z,k) = p(z)$.

4. *Here* $\mathcal{I}^s_{z_2} = [0\ 1]^T$, $\mathcal{I}^c_{z_2} = [1\ 0]^T$, $\mathcal{I}^s_{z_3} = [1\ 0]^T$, $\mathcal{I}^c_{z_3} = [0\ 1]^T$, $\mathcal{I}^s_{z_4} = [1]^T$ *and* $\mathcal{I}^c_{z_4} = [0]^T$. *For* z_2 *and* $\pi^*_{z_2}$, $\lambda^{s|\pi^*_{z_2}} = 0$, $\lambda^{c|\pi^*_{z_2}} = 1$, $p^{c|\pi^*_{z_2}}(b) = \frac{\pi^*_{z_2}\Theta_{z_2,b,z_2}\mathcal{I}^c_{z_2}}{\lambda^{c|\pi^*_{z_2}}} = 1$, $p^{s|\pi^*_{z_2}}(a) = p^{c|\pi^*_{z_2}}(a) = p^{s|\pi^*_{z_2}}(b) = 0$. *For* z_3 *and* $\pi^*_{z_3}$, $\lambda^{s|\pi^*_{z_3}} = 0.5833$, $\lambda^{c|\pi^*_{z_3}} = 0.4167$, $p^{s|\pi^*_{z_3}}(a) = \frac{\pi^*_{z_3}\Theta_{z_3,a,z_3}\mathcal{I}^s_{z_3}}{\lambda^{s|\pi^*_{z_3}}} = 0.3$, $p^{s|\pi^*_{z_3}}(b) = \frac{\pi^*_{z_3}\Theta_{z_3,b,z_3}\mathcal{I}^s_{z_3}}{\lambda^{s|\pi^*_{z_3}}} = 0.7$, $p^{c|\pi^*_{z_3}}(a) = \frac{\pi^*_{z_3}\Theta_{z_3,a,z_3}\mathcal{I}^c_{z_3}}{\lambda^{c|\pi^*_{z_3}}} = 0.3$, $p^{c|\pi^*_{z_3}}(b) = \frac{\pi^*_{z_3}\Theta_{z_3,b,z_3}\mathcal{I}^c_{z_3}}{\lambda^{c|\pi^*_{z_3}}} = 0.7$. *For* z_4 *and* $\pi^*_{z_4}$, $\lambda^{s|\pi^*_{z_4}} = 1$, $\lambda^{c|\pi^*_{z_4}} = 0$, $p^{s|\pi^*_{z_4}}(a) = \frac{\pi^*_{z_4}\Theta_{z_4,a,z_4}\mathcal{I}^s_{z_4}}{\lambda^{s|\pi^*_{z_4}}} = 0.3$, $p^{s|\pi^*_{z_4}}(b) = \frac{\pi^*_{z_4}\Theta_{z_4,b,z_4}\mathcal{I}^s_{z_4}}{\lambda^{s|\pi^*_{z_4}}} = 0.7$, $p^{c|\pi^*_{z_4}}(a) = p^{c|\pi^*_{z_4}}(b) = 0$.

5. *Then, we have*

$$\lim_{n\to\infty} D(p_n^s, p_n^c) = H(\{\lambda^s, \lambda^c\}) + \sum_{z:z \text{ is recurrent}} p(z)\left[-H(\{\lambda^{s|\pi_z^*}, \lambda^{c|\pi_z^*}\}) + D(p^{s|\pi_z^*}, p^{c|\pi_z^*})\right] = \mathbf{0.6043}.$$

Thus, for the system in Fig. 2.3, the worst case secrecy loss, as measured by the limiting JSD, is **0.6043**.

Application to Cache Side-Channel Attack.

For the cache side-channel attack model of Fig. 2.2(b), the observer model is given in Fig. 2.2(c). It can be computed that $p(z_1) = 1/6$, $p(z_2) = 5/6$, $\pi_{z_1}^* = [1]$, $\pi_{z_2}^* = [0.6\ 0.4]$, $\lambda_s = 1/3$ and $\lambda_c = 2/3$. From which, the limiting divergence $\lim_{n\to\infty} D(p_n^s, p_n^c) = \mathbf{0}$, meaning that no amount of secrecy could be leaked through the side-channel if the cache line is periodically evicted by the processor.

CHAPTER 3. QUANTIFICATION OF DISTRIBUTED SECRECY IN STOCHASTIC DISCRETE EVENT SYSTEMS UNDER BOUNDED-DELAY COMMUNICATIONS

In this chapter, we study the secrecy quantification in stochastic PODESs in the presence of distributed collusive attackers/observers, each with its own local partial observability, and where the local observers collude and exchange their observations over communication channels with bounded delays, to be able to infer more about the system secrets [Ibrahim et al. (2016a)].

3.1 d-Delaying&Masking Communication Channel

Fig. 3.1(a) shows the architecture of a system with distributed observers/attackers, where it is assumed for simplicity and without loss of any generality that there are two local observers at two local sites $I = \{1, 2\}$. Each site has three modules [Qiu and Kumar (2008)]: (i) observation mask $M_i : \overline{\Sigma} \rightarrow \overline{\Delta}_i$, where Δ_i is the set of locally observed symbols and $M_i(\epsilon) = \epsilon$ (M_i can be extended to Σ^* as follows: $M_i(\epsilon) = \epsilon$, and $\forall s \in \Sigma^*, \sigma \in \overline{\Sigma}, M_i(s\sigma) = M_i(s)M_i(\sigma)$), (ii) communication channels $C_{ij}^{(d)}, j \neq i,\ i, j \in I$, which are lossless and order-preserving, but introduce delays bounded by d, and (iii) observer Obs_i, that tracks the system "information-state" following the arrival of its local observations and the communicated observations received from other sites $j \in I, j \neq i$.

The communication channel is a "*delay-block*" with d-bounded communication delay that holds the transmitted information in First-In-First-Out ($FIFO$) manner for at most d delay steps. Accordingly, since there can be at most d events executed by system G between the transmission and the reception of a message on a channel, the channel has a maximum queue length $d+1$. Also, the channel queue evolves whenever a system event occurs, or a transmitted

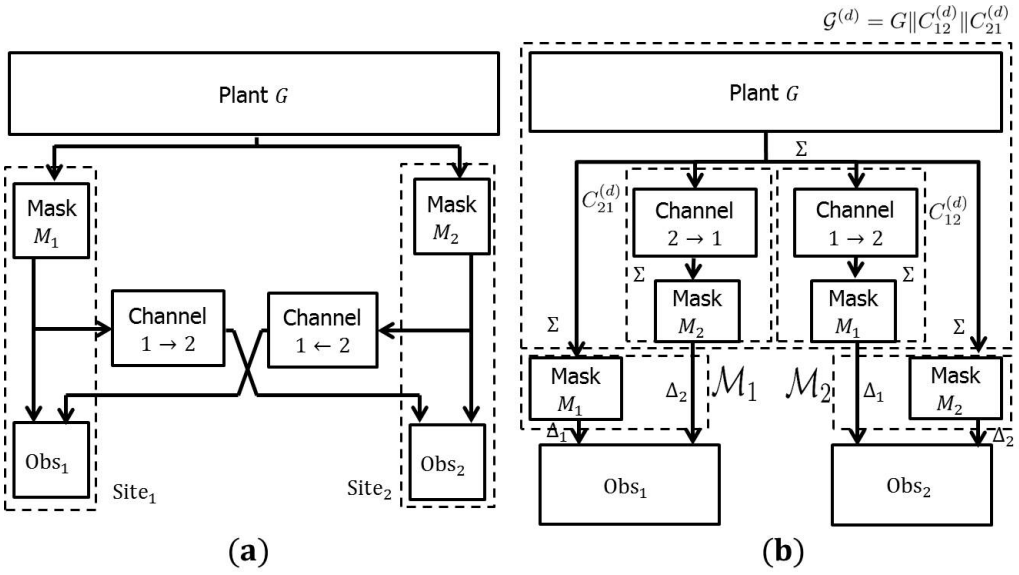

Figure 3.1 (a) Distributed secrecy system architecture to (b) equivalent system architecture

observation is delivered to a destination observer, where such arrival and departure events occur asynchronously. Accordingly, the d-delaying&masking non-stochastic channel model from site-i to site-j $(i \neq j,\ i,j \in I)$ is of the form, $C_{ij}^{(d)} = (Q_{ij}^{(d)}, \Sigma \cup \overline{\Delta}_i, \beta_{ij}^{(d)}, q_0)$, with the elements as follows. $Q_{ij}^{(d)} \subseteq \Sigma^*$ denotes the set of states, which are the event traces executed in the system but their observed values pending to be delivered at the destination. For $q \in Q_{ij}^{(d)}$, it holds that $|q| \leq d+1$. $\Sigma \cup \overline{\Delta}_i$ is the event set of $C_{ij}^{(d)}$, where Σ is its set of input events and $\overline{\Delta}_i$ is its set of output events. Without loss of generality, we assume that $\Sigma \cap \overline{\Delta}_i = \emptyset$, and $\Delta_i \cap \Delta_j = \emptyset, (j \neq i)$ (otherwise, we can simply rename some of the symbols). $q_0 = \epsilon$ is the initial state, whereas the transition function $\beta_{ij}^{(d)}$ is defined as follows:

1. "Arrival" due to an event execution in the system: $\forall q \in Q_{ij}^{(d)}, \forall \sigma \in \Sigma$, if $|q| \leq d$, then $\beta_{ij}^{(d)}(q, \sigma) = q\sigma$,

2. "Departure" due to a reception at the destination observer: $\forall q \in Q_{ij}^{(d)}, \forall \delta_i \in \overline{\Delta}_i$, if $M_i(head(q)) = \delta_i$, then $\beta_{ij}^{(d)}(q, \delta_i) = q \backslash head(q)$,

3. Undefined, otherwise,

where $head(q)$ is the first event in trace q, and the after operator "\" in $q \backslash head(q)$ returns the trace after removing the initial event $head(q)$ from the trace q.

Example 4. *A system model G is shown in Fig. 3.2(a). Suppose the observation masks of two local sites are defined as follows:*

- $M_1(a) = a'$, $M_1(b) = M_1(u) = \epsilon$, *and*
- $M_2(b) = b'$, $M_2(a) = M_2(u) = \epsilon$.

For delay $d = 0$, Fig. 3.2(b) shows the model $C_{12}^{(0)}$, and for delay $d = 1$, Fig. 3.2(c) and Fig. 3.2(d) show the models $C_{12}^{(1)}$ and $C_{21}^{(1)}$, respectively. If we follow the trace bab' in $C_{21}^{(1)}$, the states ϵ, b, ba and a are traversed sequentially. This corresponds to the situation in which site-2 sends out its observation b' to site-1 following the execution of ba in the system, whereas the observation of event a is pending to be received at site-1.

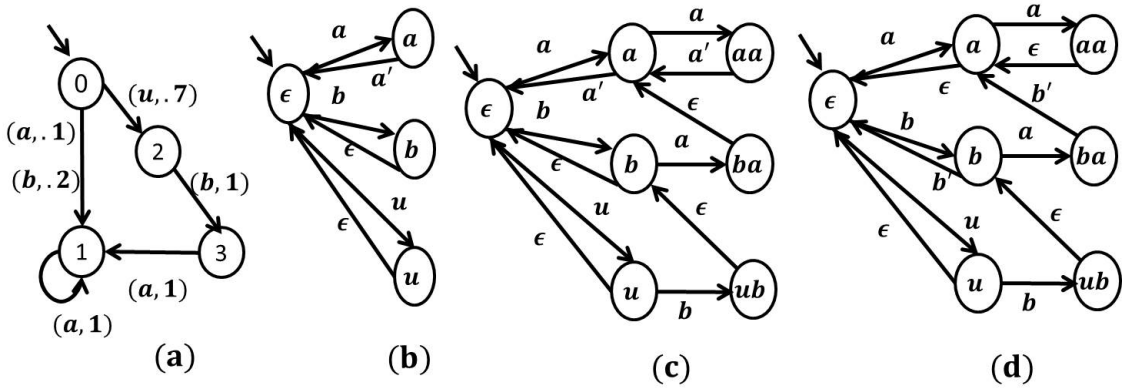

Figure 3.2 (a) Stochastic PODES G, (b) $C_{12}^{(0)}$, (c) $C_{12}^{(1)}$, (d) $C_{21}^{(1)}$

Next, since the operations of masking and delaying can be interchanged, the behaviors under the schematic of Fig. 3.1(a) are equivalent to those of Fig. 3.1(b). Then, it is clear that the distributed setting of Fig. 3.1(a) can be converted to a decentralized setting of Fig. 3.1(b), having an extended system $\mathcal{G}^{(d)}$ and local observers having the extended observation masks $\{\mathcal{M}_i\}$, defined below. The extended system is given by $\mathcal{G}^{(d)} = G\|_{i,j\in I, i\neq j} C_{ij}^{(d)}$, whereas the extended system model $\mathcal{G}_i$ at site-i ($i \in I$) includes the system model and only the incoming channel models: $\mathcal{G}_i = G\|_{j\in I-\{i\}} C_{ji}^{(d)}$. The *extended system* $\mathcal{G}_i$ "generates" events in $\Sigma \cup_{j\neq i} \Delta_j$, which are observed by site-i observer Obs_i through an extended observation mask $\mathcal{M}_i$: $\overline{\Sigma} \cup_{j\in I-\{i\}} \overline{\Delta}_j \to \overline{\Delta} = \cup_{i\in I} \overline{\Delta}_i$. $\mathcal{M}_i$ acts the same as M_i for events in Σ, whereas it is an identity

mask for events in Δ_j $(j \neq i)$. Formally, it is defined as follows:

$$\mathcal{M}_i(\sigma) := \begin{cases} M_i(\sigma), & \sigma \in \Sigma, \\ \sigma, & \sigma \in \Delta_j \ (j \neq i). \end{cases} \tag{3.1}$$

Next, we assign probabilities to transitions in $\mathcal{G}_i$ as follows. For each state in $\mathcal{G}_i$, the transition is either one of the system events, or at most one of channel j $(j \neq i)$ events (either arrival or departure of that channel). Suppose at a system $\mathcal{G}_i$ state, with vector of all incoming channel lengths $\vec{k}$, the system event is picked with probability $p^0_{\vec{k}}$, and suppose the channel j $(j \neq i)$ event can occur with probability $p^j_{\vec{k}}$ such that, $p^0_{\vec{k}} + \sum_{j \neq i} p^j_{\vec{k}} = 1$. We also require that when all channels are empty $(\vec{k} = \vec{0})$, $p^0_{\vec{k}} = 1$ (so no channel output can occur when channels are empty), when any channel is full $(\vec{k} = \overrightarrow{d+1})$, $p^0_{\vec{k}} = 0$ (so no channel input can occur when any channel is full), and if channel j has higher queue length than channel j' $(\vec{k_j} \geq \vec{k'_j})$, then it can be expected that $p^j_{\vec{k}} \geq p^{j'}_{\vec{k}}$ (channel j event is more likely than channel j' event when channel j has more number of pending observations). With this choice of selection probability of events, extended system model is given by $\mathcal{G}_i = (X \times (\Pi_{j \neq i} Q^{(d)}_{ji}), \Sigma \cup_{j \neq i} \overline{\Delta}_j, \gamma, (x_0, \vec{q_0}))$, where $\forall (x, \vec{q}), (x', \vec{q'}) \in X \times (\Pi_{j \neq i} Q^{(d)}_{ji}), \sigma \in \Sigma \cup_{j \neq i} \overline{\Delta}_j$,

$$\gamma((x, \vec{q}), \sigma, (x', \vec{q'})) = \begin{cases} \alpha(x, \sigma, x') \times p^0_{\vec{k}} & \text{if } \sigma \in \Sigma, \\ p^j_{\vec{k}} & \text{if } \sigma \in \cup_{j \neq i} \overline{\Delta}_j, \end{cases}$$

and otherwise, $\gamma((x, \vec{q}), \sigma, (x', \vec{q'})) = 0$.

The computation of an observer transition structure for $\mathcal{G}_i$ and the associated transition matrices $\{\Theta(\delta) \mid \delta \in \Delta\}$, is exactly the same as in the centralized setting, as is described in Section 2.2.

Example 5. *Continuing Example 4, suppose the delay bound $d = 1$, so there are three possibilities for the length of the only channel, $\vec{k} = \{0, 1, 2\}$. Let $p^0_0 = 1$, $p^0_1 = 0.5$, $p^0_2 = 0$ (implying $p^2_0 = 1 - p^0_0 = 0$, $p^2_1 = 1 - p^0_1 = 0.5$, $p^2_2 = 1 - p^0_2 = 1$). Fig. 3.3(a) shows the extended system model $\mathcal{G}_1$ at site-1. Following remark 1, suppose R is given in Fig. 3.3(b), i.e., $K = L(R) = a^+ \cup ba^*$. The extended plant model at site-1 can be refined with respect to the specification to identify the secret and cover behaviors as* states *in the refined plant, and is given by $\mathcal{G}^R_1 = G || C^{(d)}_{21} || \overline{R}$. Then, the refinement $\mathcal{G}^R_1$ is shown in Fig. 3.3(c). So for*

example, at the initial state $(0, \epsilon, 0)$*, the channel is empty, and no channel events occur at this state* ($p_0^2 = 0$ *while* $p_0^0 = 1$*). Then, for any system event* $\sigma \in \Sigma$, $\gamma((0, \epsilon, 0), u, (2, u, D)) = \alpha(0, u, 2) \times p_0^0 = 0.7 \times 1 = 0.7$, $\gamma((0, \epsilon, 0), b, (1, b, 1)) = \alpha(0, b, 1) \times p_0^0 = 0.2 \times 1 = 0.2$, *and* $\gamma((0, \epsilon, 0), a, (1, a, 1)) = \alpha(0, a, 1) \times p_0^0 = 0.1 \times 1 = 0.1$. *Whereas, at state* $(2, u, D)$*, there is observation* u *queued up in the channel. Thus, either the system can execute a new event* $b \in \Sigma$*, with probability* $\gamma((2, u, D), b, (3, ub, D)) = \alpha(2, b, 3) \times p_1^0 = 1 \times p_1^0 = 0.5$*, or a channel event can occur, with probability* $\gamma((2, u, D), \epsilon, (2, \epsilon, D)) = p_1^2 = 0.5$*. The remaining state transitions can be computed similarly. The models* $\mathcal{G}_2$ *and* $\mathcal{G}_2^R$ *at site-2 can be generated in a manner similar to* $\mathcal{G}_1$ *and* $\mathcal{G}_1^R$*, respectively.*

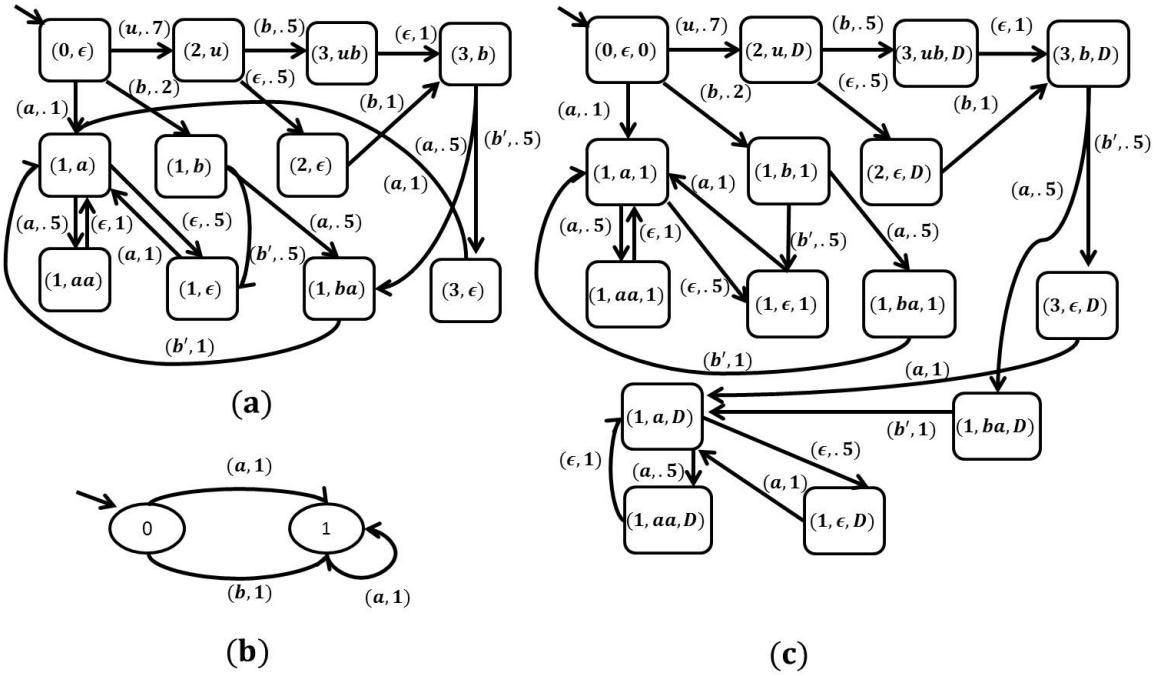

Figure 3.3 (a) Extended system model $\mathcal{G}_1$ at site-1, (b) specification for secrets, R, (c) refined system model $\mathcal{G}_1^R$

3.2 Jensen-Shannon Divergence-based Distributed Secrecy Quantification

In chapter 2, we presented a way to compute JSD-based measure of secrecy loss for stochastic PODES when there is a single observer. To compute the secrecy loss in the distributed setting, resulting from the aggregated observations at any site-i ($i \in I$), which include it's own im-

mediate observations and the delayed communicated observations from other distributed sites, the JSD computation can be carried out over the extended system model $\mathcal{G}_i$, following the method introduced in Sect. 2.6. The example below illustrates the extended observer structure and the corresponding JSD based secrecy loss computation in a distributed collusive setting, respectively.

Example 6. *Consider the refined extended system model of Fig. 3.3(c) at site-1, where* $\mathcal{M}_1(a) = a'$, $\mathcal{M}_1(b) = \mathcal{M}_1(u) = \epsilon$, *while the extended mask function is the identity function over the received observations,* $\Delta_2 = \{b'\}$. *Then, Fig. 3.4(a) shows the extended observer* Obs_1.

Then, based on Obs_1, *the following computation illustrates the steps of JSD computation at site-1.*

1. $\sum_z |z| = 14$ *and so* $\widetilde{\Theta}$ *is a* 14×14 *matrix with entries:*
   $\widetilde{\Theta}(1,2) = \widetilde{\Theta}(1,3) = \widetilde{\Theta}(1,5) = 0.1$, $\widetilde{\Theta}(1,4) = \widetilde{\Theta}(1,6) = 0.35$, $\widetilde{\Theta}(2,7) = \widetilde{\Theta}(2,8) = \widetilde{\Theta}(7,7) = \widetilde{\Theta}(7,8) = \widetilde{\Theta}(8,7) = \widetilde{\Theta}(8,8) = \widetilde{\Theta}(9,11) = \widetilde{\Theta}(9,12) = \widetilde{\Theta}(10,13) = \widetilde{\Theta}(10,14) = \widetilde{\Theta}(11,11) = \widetilde{\Theta}(11,12) = \widetilde{\Theta}(12,11) = \widetilde{\Theta}(12,12) = \widetilde{\Theta}(13,13) = \widetilde{\Theta}(13,14) = \widetilde{\Theta}(14,13) = \widetilde{\Theta}(14,14) = 0.5$, $\widetilde{\Theta}(3,9) = \widetilde{\Theta}(4,10) = \widetilde{\Theta}(5,9) = \widetilde{\Theta}(6,10) = 1$ *and zeros elsewhere.*
   Then, $\pi^* = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0.05 & 0.05 & 0 & 0 & 0.1 & 0.1 & 0.35 & 0.35 \end{bmatrix}$. *Therefore,* $p(z_0) = p(z_1) = p(z_2) = 0$, $p(z_3) = 0.1$, $p(z_4) = 0$, *and* $p(z_5) = 0.9$.
2. *Here* $\mathcal{I}^s = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}^T$,
   $\mathcal{I}^c = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}^T$. *And so* $\lambda^s = 0.3$ *and* $\lambda^c = 0.7$.
3. *Here* z_3, *and* z_5 *are recurrent nodes, and each of them forms a SCC. We have* $\pi^*_{z_3} = [0.5\ 0.5]$, *and while there are multiple solutions to the equation set* $\pi^*_{z_5} = \frac{\pi^*_{z_5}\Theta_{z_5,a',z_5}}{||\pi^*_{z_5}\Theta_{z_5,a',z_5}||}$, *only* $\pi^*_{z_5} = [0.11\ 0.11\ 0.39\ 0.39]$ *is reachable. Thus, each set of recurrent nodes is a singleton set, and each with a unique fixed-point distribution. Therefore, for each recurrent node* z, $p(z,k) = p(z)$.
4. *Here* $\mathcal{I}^s_{z_3} = [1\ 1]^T$, $\mathcal{I}^c_{z_3} = [0\ 0]^T$, $\mathcal{I}^s_{z_5} = [1\ 1\ 0\ 0]^T$, $\mathcal{I}^c_{z_5} = [0\ 0\ 1\ 1]^T$. *For* z_3 *and* $\pi^*_{z_3}$, $\lambda^{s|\pi^*_{z_3}} = 1$, $\lambda^{c|\pi^*_{z_3}} = 0$, $p^{s|\pi^*_{z_3}}(a') = \frac{\pi^*_{z_3}\Theta_{z_3,a',z_3}\mathcal{I}^s_{z_3}}{\lambda^{s|\pi^*_{z_3}}} = 1$, $p^{s|\pi^*_{z_3}}(b') = p^{c|\pi^*_{z_3}}(b') =$

$p^{c|\pi^*_{z_3}}(a') = 0$. *For* z_5 *and* $\pi^*_{z_5}$, $\lambda^{s|\pi^*_{z_5}} = 0.22$, $\lambda^{c|\pi^*_{z_5}} = 0.78$, $p^{s|\pi^*_{z_5}}(a') = \frac{\pi^*_{z_5}\Theta_{z_5,a',z_5}\mathcal{I}^s_{z_5}}{\lambda^{s|\pi^*_{z_5}}} = 1$, $p^{c|\pi^*_{z_5}}(a') = \frac{\pi^*_{z_5}\Theta_{z_5,a',z_5}\mathcal{I}^c_{z_5}}{\lambda^{c|\pi^*_{z_5}}} = 1$, $p^{s|\pi^*_{z_5}}(b') = p^{c|\pi^*_{z_5}}(b') = 0$.

5. *Then, we have*

$$\lim_{n\to\infty} D_1(p^s_n, p^c_n) = H(\{\lambda^s, \lambda^c\}) + \sum_{z:z \text{ is recurrent}} p(z)\left[-H(\{\lambda^{s|\pi^*_z}, \lambda^{c|\pi^*_z}\}) + D_1(p^{s|\pi^*_z}, p^{c|\pi^*_z})\right] = \mathbf{0.197}.$$

Note this happens to be the same as JSD measure of secrecy loss at site-2.

In contrast, when there is no collusion among observers (so there is no communication among the two sites), Fig. 3.4(b) and Fig. 3.4(c) show, respectively, the refined system G^R *(no incoming channels and so identical refined model at all sites) and the corresponding site-1 observer structure. The JSD value, computed in same manner as above but with respect to the observer structure of Fig. 3.4(c), is simply* **Zero**, *i.e., no amount of secrets is revealed under no collusion. This is because for every observation, the probability of it coming from secrets in* K *vs from covers in* $L - K$ *is exactly the same.*

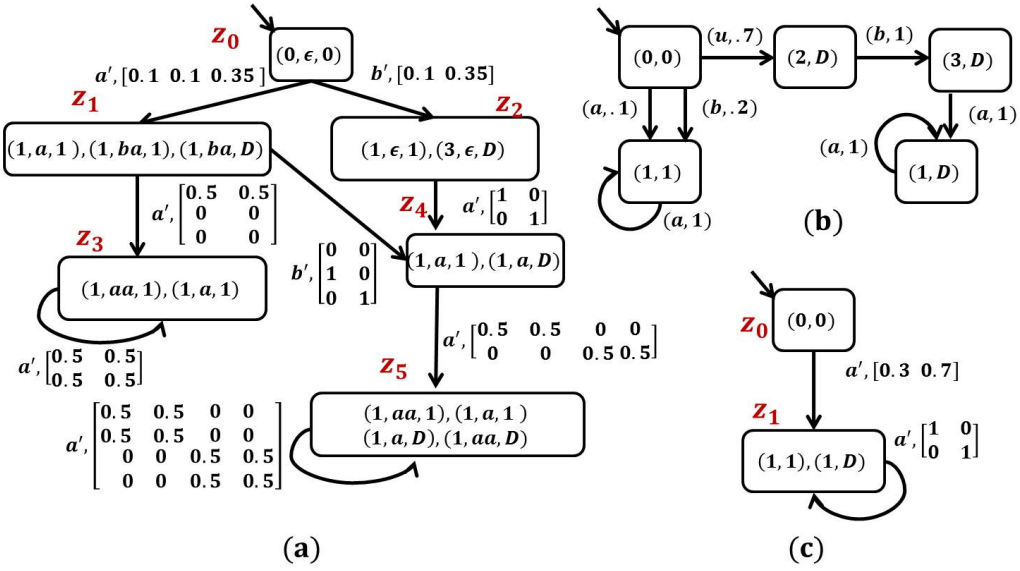

Figure 3.4 (a) Observer Obs_1 for the system of Fig. 3.3(c), (b) Model G^R for system of Fig. 3.2(a) under no collusion, (c) Observer under no collusion

CHAPTER 4. A RESILIENCY MEASURE AND ITS EXAMINATION VIA POWER SYSTEMS

The discussion in this chapter is directed towards Power Systems to make those more concrete, but the concepts examined and the measure proposed apply generally to any cyber-physical-system.

4.1 Power System Dynamics and Transient Stability to Large Disturbances

A power system consists of generator (PV) buses, for which generator real power and voltage magnitude are specified, load (PQ) buses, for which real and reactive load powers are specified, and occasionally a slack bus, for which the voltage magnitude and phase are specified (typically zero phase is used, making this bus as a reference). The dynamics at a generator can be modeled by a pair of differential equations that are referred to as the swing equations [Pai (1989)]. The swing equations for generator i in an interconnected power system are expressed as:

$$\dot{\delta}_i = \omega_i \tag{4.1}$$

$$M_i\dot{\omega}_i = -D_i\omega_i + P_{m,i} - P_{e,i} \qquad i = 1, ...n, \tag{4.2}$$

where the electrical power of generator i satisfies:

$$P_{e,i} = \sum_{j=1}^{n} |E_i| \times |E_j| \times [G_{ij}\cos(\delta_i - \delta_j) + B_{ij}\sin(\delta_i - \delta_j)],$$

In (4.1), (4.2), M_i is inertia constant; D_i is damping constant, $P_{m,i}$ is mechanical power input; $P_{e,i}$ is the electrical power output; δ_i is angle of internal complex voltage of i^{th} machine; and ω_i is rotor angle velocity of the i^{th} machine with respect to the reference frequency of the power system ω_r. In (4.3), E_i is i^{th} machine's internal complex voltage; $G_{ij} = G_{ji} \geq 0$ is the Kron-reduced equivalent conductance between generator i and generator j; $B_{ij} = B_{ji} > 0$ is the Kron-

reduced equivalent susceptance between generator i and generator j, and $Y_{ij} = G_{ij} + \sqrt{-1}B_{ij}$ is the Kron-reduced equivalent admittance between generator i and generator j (the ij^{th} element of Kron-reduced equivalent admittance matrix Y_I of size $|n| \times |n|$). The solution of (4.1)-(4.3) in steady state yields the so called *power flow solutions* that yield the magnitude and phase angle of the voltage at each bus, and the power flowing in each line.

Protective relays can be placed throughout the system to detect large disturbances and to trigger the opening of circuit breakers to isolate the power system [Varaiya et al. (1985)]. Therefore, the power system can be considered as going through changes in configuration in three stages, from pre-fault, faulted, to post-fault systems. Accordingly, the underlying dynamical system can be described by a set of three differential equations:

$$\dot{\vec{x}}(t) = f_I(\vec{x}(t)), \quad -\infty < t < t_f \tag{4.3}$$

$$\dot{\vec{x}}(t) = f_F(\vec{x}(t)), \quad t_f \leq t < t_c \tag{4.4}$$

$$\dot{\vec{x}}(t) = f_P(\vec{x}(t)), \quad t_c \leq t < \infty, \tag{4.5}$$

where $\vec{x}(t)$ is a vector of state variables of the system at time t, f_I is initial pre-fault flow, f_F is flow under fault, and f_P is post-fault flow. The pre-fault system is at a known initial stable steady state equilibrium. A large disturbance/fault occurs at time t_f, and then, the system is in the faulted condition before it is cleared at time t_c by the protective system operation. The *critical clearing time*, denoted as t_c^*, is the largest value of t_c, so, the trajectory for (4.5) with initial condition $\vec{x}(t_c)$ will converge to a stable equilibrium point of (4.5).

For the sake of illustration, we consider a pair of power systems with identical buses, generators and loads but with different topologies, as shown in Fig. 4.1. For the 1^{st} power system, PS_1, bus 1 is the *slack* bus, buses 2 and 3 are the generator buses, and buses 4, 5, and 6 are the load buses. The machine, load and line data, generation schedule, and reactive power limits for the regulated buses, along with power flow solution data for PS_1 are shown in Tables 4.1-4.5. The 2^{nd} power system, PS_2, has same generation and loads as PS_1, but with different topology, where line L_{16} appears as line L_{45}. PS_2 data are also given in Tables 4.1-4.6. We show that while the two systems are served by the same generators, and serve the same set of loads, they have different resiliency to the same attacks owing to their topological

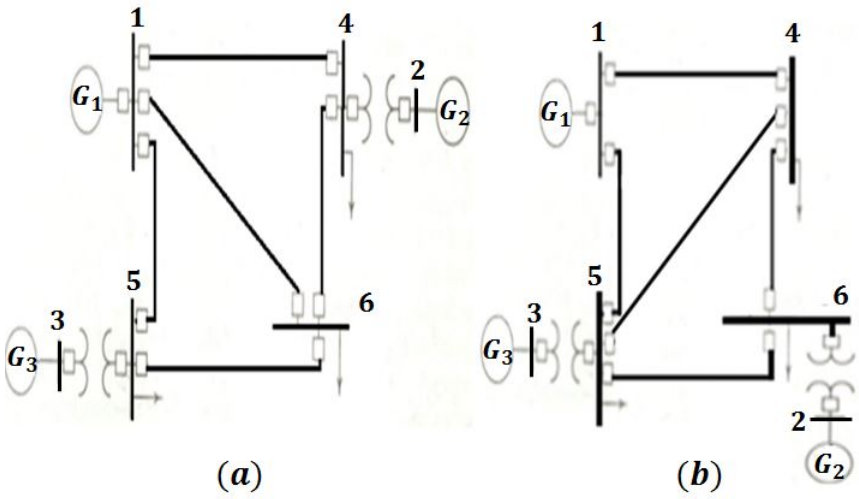

Figure 4.1 (a) System PS_1, and (b) System PS_2

difference. Here, we provide the modeling data for the power systems PS_1 and PS_2 that we use as running examples. The corresponding Matlab code is provided in Appendix A.

Table 4.1 Machine data for both PS_1 and PS_2

Gen.	$R_a(PU)$	$X_d(PU)$	**M**(sec^2/rad)	**D(sec/rad)**
1	0	0.2	0.106	0.12
2	0	0.15	0.021	0.12
3	0	0.25	0.027	0.12

Table 4.2 Generation schedule for both PS_1 and PS_2

Bus No.	**Voltage(Mag.)**	**Generation(MW)**	**Qmin.(Mvar)**	**Qmax.(Mvar)**
1	1.06	0		
2	1.04	150	0	140
3	1.03	100	0	90

4.2 Modal-Region-of-Stability and Level-of-Resilience

For a nonlinear autonomous system, the stability region is defined as the set of all initial points from which the autonomous system eventually converges to a stable-equilibrium-point (SEP) [Khalil (2002)]. For the simplicity of discussion, we assume that the system is lossless, so the transfer admittance is purely imaginary [Liu and Thorp (1997)]. Equation (4.3) then

Table 4.3 Load data for both PS_1 and PS_2

Bus No.	**Load(MW)**	**Load(Mvar)**
1	0	0
2	0	0
3	0	0
4	100	70
5	90	30
6	160	110

Table 4.4 Line data for PS_1 (and PS_2 with reordering)

Bus No.	**Bus No.**	**R(PU)**	**X(PU)**	$(1/2B)(PU)$
1	4	0	0.225	0.0065
1	5	0	0.105	0.0045
1	6	0	0.215	0.0055
2	4	0	0.035	0.0000
3	5	0	0.042	0.0000
4	6	0	0.125	0.0035
5	6	0	0.175	0.0300

simplifies to:

$$P_{e,i} = \sum_{j=1}^{n} |E_i| \times |E_j| \times [B_{ij} sin(\delta_i - \delta_j)]. \tag{4.6}$$

Define an energy function $V(\vec{\delta}, \vec{\omega})$, where $\vec{\delta} = [\delta_1, \ldots, \delta_n]^T$ and $\vec{\omega} = [\omega_1, \ldots, \omega_n]^T$, as follows:

$$V(\vec{\delta}, \vec{\omega}) = V_K(\vec{\omega}) + V_P(\vec{\delta}), \tag{4.7}$$

where,

$$V_K(\vec{\omega}) = \frac{1}{2} \sum_{i=1}^{n} M_i \omega_i^2 \tag{4.8}$$

$$V_P(\vec{\delta}) = -\sum_{i=1}^{n} P_{m,i} \delta_i - \sum_{i=1}^{n} \sum_{j=i}^{n} |E_i| \times |E_j| \times [B_{ij} \cos(\delta_i - \delta_j)]. \tag{4.9}$$

Table 4.5 Power flow solution for PS_1

Bus No.	**Voltage (Mag.)**	**Angle (Degree)**	**Load(MW)**	**Load(Mvar)**	**Generation(Mw)**	**Generation(Mvar)**
1	1.060	0.000	0.000	0.000	100.000	116.170
2	1.040	1.217	0.000	0.000	150.000	97.704
3	1.030	0.412	0.000	0.000	100.000	27.919
4	1.008	-1.653	100.000	70.000	0.000	0.000
5	1.019	-1.881	90.000	30.000	0.000	0.000
6	0.960	-6.368	160.000	110.000	0.000	0.000

Table 4.6 Power flow solution for PS_2

Bus No.	Voltage (Mag.)	Angle (Degree)	Load(MW)	Load(Mvar)	Generation(Mw)	Generation(Mvar)
1	1.060	0.000	0.000	0.000	100.000	82.304
2	1.040	-2.391	0.000	0.000	150.000	125.447
3	1.030	-0.370	0.000	0.000	100.000	26.545
4	0.977	-6.511	100.000	70.000	0.000	0.000
5	1.020	-2.661	90.000	30.000	0.000	0.000
6	0.999	-5.287	160.000	110.000	0.000	0.000

Note that $V_K(\vec{\omega})$ is kinetic energy of the generators and $V_P(\vec{\delta})$ is the potential energy of the system, that is stored in the inductive lines of the power grid network.

Using the potential energy function, the swing equations (4.1), (4.2) can be rewritten as follows:

$$\dot{\delta}_i = \omega_i \tag{4.10}$$

$$\dot{\omega}_i = \frac{1}{M_i}[-D_i\omega_i - \frac{\partial V_P}{\partial \delta_i}(\vec{\delta})]. \tag{4.11}$$

A point $(\vec{\delta}^e, 0)$ is an equilibrium of (4.10) and (4.11) if and only if $(\partial V_P/\partial\vec{\delta})(\vec{\delta}^e) = 0$. Since $\vec{\omega}^e = 0$, the energy function at the equilibrium is of form: $V(\vec{\delta}^e, \vec{\omega}^e) = V_P(\vec{\delta}^e)$ [Pai (1989)]. Then, the stability region of a power system can be equivalently studied in the $\vec{\delta}$ subspace.

Theorem 4 (Thomas and Thorp (1985))**.** *Consider the reduced-order gradient system:*

$$\dot{\vec{\delta}} = -\frac{\partial V_P}{\partial \vec{\delta}}(\vec{\delta}). \tag{4.12}$$

The stability boundary of (4.12) is the potential energy boundary surface of (4.10), (4.11).

For RoS computation, given a SEP $\vec{x}^e$ of a system, we propagate in time the boundary of the backward reachable set of $\vec{x}^e$, i.e., the set of states starting from where trajectories can reach the SEP, by solving the following *Hamilton-Jacobi-Isaacs* (HJI) PDE:

$$\phi_{\vec{x}}^T f(\vec{x}) + \phi_t = 0. \tag{4.13}$$

This PDE describes the propagation of the backward reachable set boundary, specified by $\phi(\vec{x}, t) = 0$, as a function of time, in which $\phi_{\vec{x}}^T = [\frac{\partial\phi}{\partial x_1}, ..., \frac{\partial\phi}{\partial x_n}]$, and with terminal conditions:

$$\phi(\vec{x}, 0) = ||\vec{x} - \vec{x}^e|| = 0.$$

The backward reachable set of the SEP $\vec{x}^e$ (computed using the toolbox of level set methods [Mitchell (2004)]) is always contained in the region of stability of the SEP $\vec{x}^e$, and as t goes to infinity, the backward reachable set approaches the true region of stability. Fig. 4.2 shows the propagation of the backward reachable set boundary of pre-fault system for PS_1 and its convergence to the RoS, which is as described in Theorem 4.

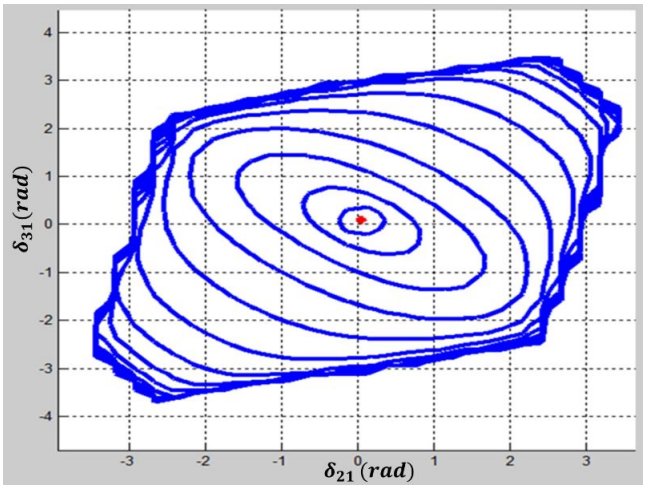

Figure 4.2 PS_1 pre-fault RoS computation using backward reachable set

A Modal-Region-of-Stability "*modal-RoS*" is a graphical representation that captures the evolution of RoS through the changes of systems modes of configurations under a sequence of fault and recovery actions. We denote a *modal-RoS* for a given sequence of fault and recovery actions as: $RoS_I \rightarrow RoS_{P1} \rightarrow ... \rightarrow RoS_{Pm}$, where I is the initial pre-fault configuration mode, "$\rightarrow$" designates mode change, $P1, ..., Pm$ are the new post-fault configuration modes as a sequence of m fault and recovery actions take place sequentially in time. Fig. 4.4(a) shows RoS evolution for PS_1 from its pre-fault mode I; its state trajectory when fault is applied at line L_{15} causing RoS to be lost is shown in Fig. 4.4(b); post-fault Region-of-Stability RoS_{P1} in Fig. 4.4(c); and post-fault state trajectory within RoS_{P1} when clearance is applied within critical time in Fig. 4.4(d). Then, repeating this process for a sequence of fault and recovery actions, a *modal-RoS* can be generated. Fig. 4.3 shows an attack scenario A_1 in which three lines are faulted in the sequence: $L_{15} \rightarrow L_{46} \rightarrow L_{56}$, along with recovery actions with certain clearance times. Accordingly, the *modal-RoS*: $RoS_I \rightarrow RoS_{P1} \rightarrow RoS_{P2}$, is shown in Fig. 4.5,

where $P1$ is post-fault mode after clearing line L_{15} fault, and $P2$ is post-fault mode after line L_{46} fault is cleared. Note the *RoS* is lost after the final L_{56} fault, and so *modal-RoS* terminates at this fault.

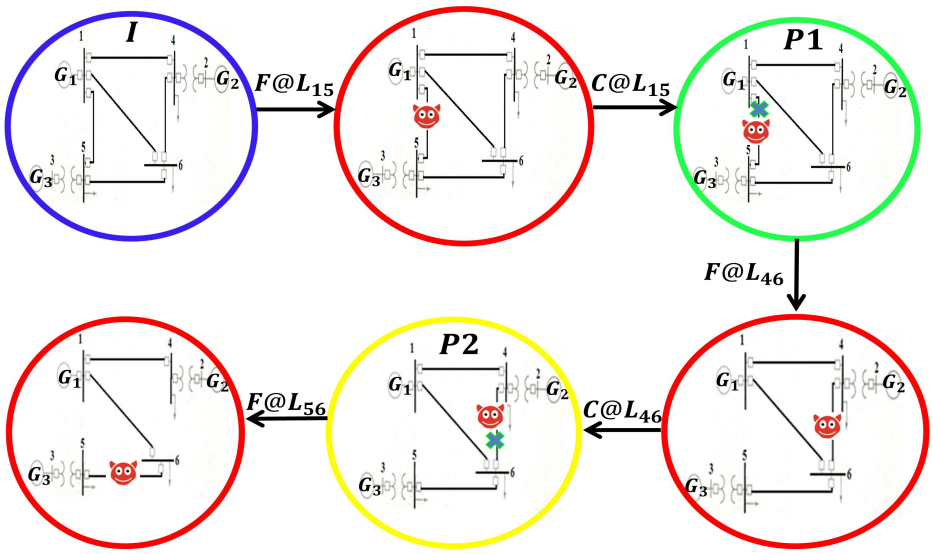

Figure 4.3 PS_1 topology evolution under A_1 (transition label F denotes fault, whereas C denotes its clearance)

Level-of-Resilience (LoR) is determined by comparing the reduction in size of *RoS*, denoted as RoS_R, and also the reduction in performance (LoP_R), as measured along the various modes of the *modal-RoS*. (For the moment, we ignore Recovery Time since this is the same for the two power systems.)

For a given attack scenario, we compute the LoR as follows. First for the initial pre-fault mode I, compute the *RoS* of its SEP $\vec{x}_I^e$. For any perturbation/low level disturbance (e.g., transient change in load) that does not switch the mode, the perturbed state must be within the *RoS* of the pre-fault system for the system to remain stable (state trajectory still converge to $\vec{x}_I^e$). When a fault/severe disturbance occurs affecting the system structure, a new Kron-reduced equivalent admittance matrix Y_F corresponding to faulted mode is generated; the system may become unstable without any recovery action rendering the *RoS* to be an empty

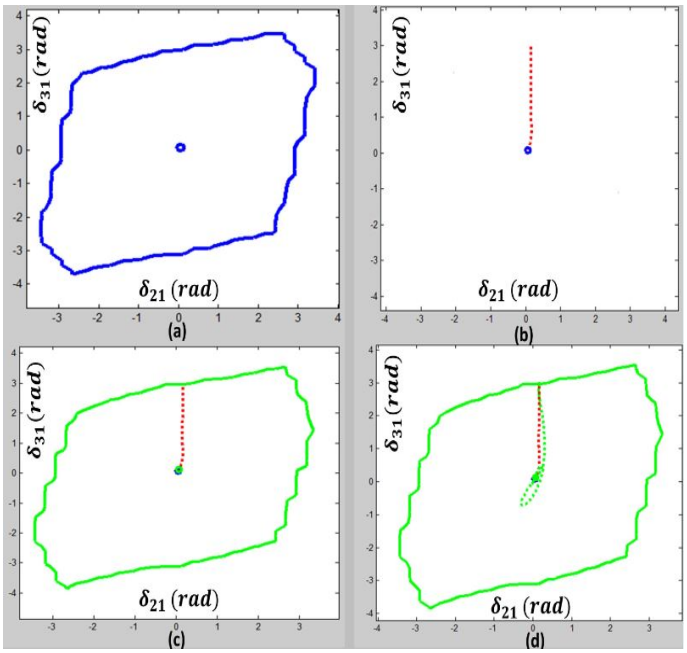

Figure 4.4 PS_1 RoS evolution (a) pre-fault RoS_I, (b) state trajectory when fault occurs at line L_{15}, (c) RoS_{P1} for mode $P1$ after fault is cleared, (d) post-fault state trajectory within RoS_{P1} when fault is cleared within critical time

set. The fault can be cleared by isolating the faulted line using circuit breakers. If the fault is not cleared within a critical time window (t_c^*), then overall system might no longer be stable. When fault is cleared within the critical time, a new Kron reduced admittance matrix Y_P is obtained, corresponding to post-fault mode P. The system will stabilize to a new equilibrium point $\vec{x}_P^e$ of mode P only if the forward reachable state trajectory under fault starting from pre-fault $\vec{x}_I^e$, *i.e.*, $Reach_f^+(\vec{x}_I^e)$ is within the RoS of $\vec{x}_P^e$. This is captured by the requirement:

$$t_c \text{ such that } Reach_f^+(\vec{x}_I^e, t_c) \in RoS(\vec{x}_P^e).$$

Such pre-fault and post-fault RoSs can continue to be sequenced for any subsequent attacks to yield a sequence of RoSs, a *modal-RoS*. Associated with each *modal-RoS* is a Level-of-Performance-Reduction (LoP_R), measured as the percentage of reduction of load served along the various modes of the *modal-RoS*. Using the size of RoS and the associated LoP_R for the eventual mode, as defined next, we can measure and compare Level-of-Resilience (LoR). Another aspect of resiliency metric is *Recovery-Time (RT)*, which is the time system takes to detect and recover from an attack or a fault.

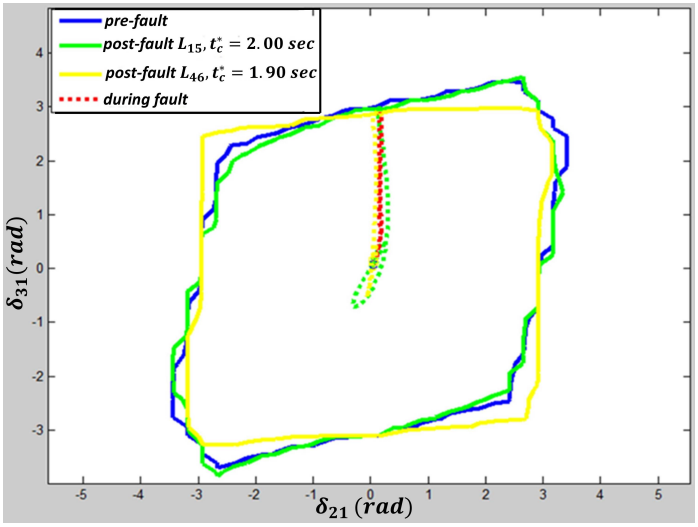

Figure 4.5 PS_1 *modal-RoS* under A_1

Definition 1. *Given a modal-RoS:* $RoS_I \rightarrow RoS_{P1} \rightarrow ... \rightarrow RoS_{Pm}$*, the percentage of RoS-Reduction,* RoS_R*, is given by,*

$$RoS_R = \frac{D_I - D_{Pm}}{D_I}\%, \tag{4.14}$$

where for a given RoS boundary Ω *in n-dimensional space with equilibrium* $\vec{x}^e$*, D can be computed as the shortest Euclidean distance between* Ω *and* $\vec{x}^e$*:*

$$D = min_{\vec{q}\in\Omega}||\vec{q} - \vec{x}^e|| = min_{\vec{q}\in\Omega}\sqrt{\sum_{k=1}^{n}(q_k - x_k^e)^2}. \tag{4.15}$$

Associated with each *modal-RoS* is a Level-of-Performance-Reduction (LoP_R), measured as the percentage of reduction of load served along the various modes of the *modal-RoS*.

Definition 2. *Given a modal-RoS:* $RoS_I \rightarrow RoS_{P1} \rightarrow ... \rightarrow RoS_{Pm}$*, Level-of-Performance-Reduction,* LoP_R*, is defined as percentage of reduction of load served under a given attack scenario:*

$$LoP_R = \frac{\sum_{\text{loads in mode } I} p_{\text{real}} - \sum_{\text{loads in mode } Pm} p_{\text{real}}}{\sum_{\text{loads in mode } I} p_{\text{real}}}\%.$$

Definition 3. *Given two systems* PS_1*,* PS_2*, and an attack scenario A,* $LoR(PS_1, A) > LoR(PS_2, A)$ *if:* $[RoS_R(PS_1, A) < RoS_R(PS_2, A)]$

$\vee\ [[RoS_R(PS_1, A) = RoS_R(PS_2, A)] \wedge [LoP_R(PS_1, A) < LoP_R(PS_2, A)]]$

$\vee$ $[[RoS_R(PS_1, A) = RoS_R(PS_2, A)] \wedge [LoP_R(PS_1, A) = LoP_R(PS_2, A)] \wedge [RT(PS_1, A) < RT(PS_2, A)]]$.

In power systems, local controls are used for detection and clearance of faults, and so RT is typically independent of system topology and hence not included in Definition 3, but could be included for general dynamical systems.

Remark 2. *Definition 3 compares Level-of-Resilience under a given attack scenario. This can be generalized to the case over all possible attack-scenarios by considering the* worst-case *or* average-case *values. This requires enumerating all attack-scenarios, which can be an outcome of model based generation of attack sequences as discussed in next chapter.*

4.3 Experimental Comparison of LoR

In this section, we simulate two different attack scenarios for two different power systems PS_1 (Fig. 4.1(a)) and PS_2 (Fig. 4.1(b)), with same generators and loads, but with different topology, and for each scenario we evaluate and compare their LoR. Consider the first attack scenario A_1, in which three lines are faulted in the sequence: $L_{15} \rightarrow L_{46} \rightarrow L_{56}$, and cleared with certain clearance times by the simultaneous opening of breakers at both ends of this line. Note that If we apply a third fault at line L_{56} near bus 5 at time 21 *sec*, for PS_1 then, regardless of the clearance time, machine 3 no longer runs in synchronism (i.e., its relative angle diverges).

Fig. 4.6 shows the corresponding relative angles under A_1. Fig. 4.5 shows the evolution of RoS for PS_1, yielding a *modal-RoS* as discussed earlier.

We simulate the same attack sequence A_1 for the second power system, PS_2, shown in Fig. 4.1(b). Note that applying a third fault at line L_{56} at time 21 *sec* causes machine 2 to fall out of synchronism, regardless of the clearance time. Fig. 4.7 shows the system relative angles with respect to time. Fig. 4.8 shows the RoS evolution for PS_2, yielding its own *modal-RoS*.

For PS_1 (respectively, PS_2), the protective relay elements across machine 3 (respectively, machine 2) would interpret the loss of synchronizing condition as an abnormal operating condition and trip machine 3 (respectively, machine 2) [Thompson (2012)], ensuring its protec-

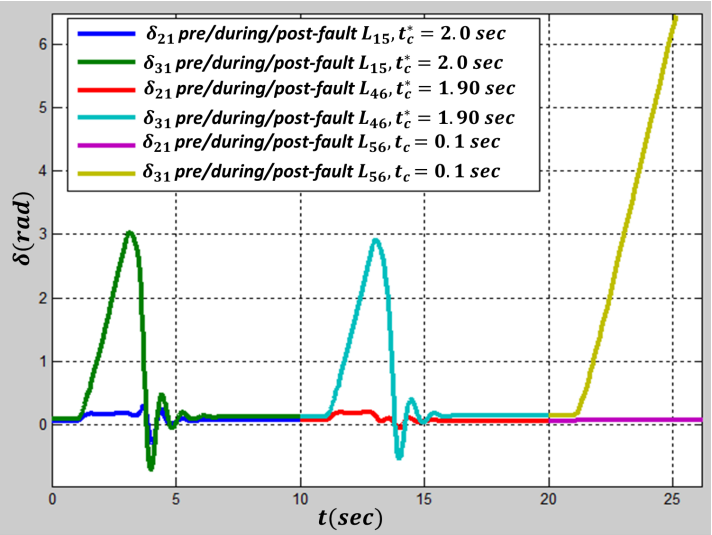

Figure 4.6 PS_1 relative angles under A_1

tion. In the end, PS_1 has Level-of-Performance-Reduction, $LoP_R = 25.71\%$, while for PS_2, $LoP_R = 45.71\%$.

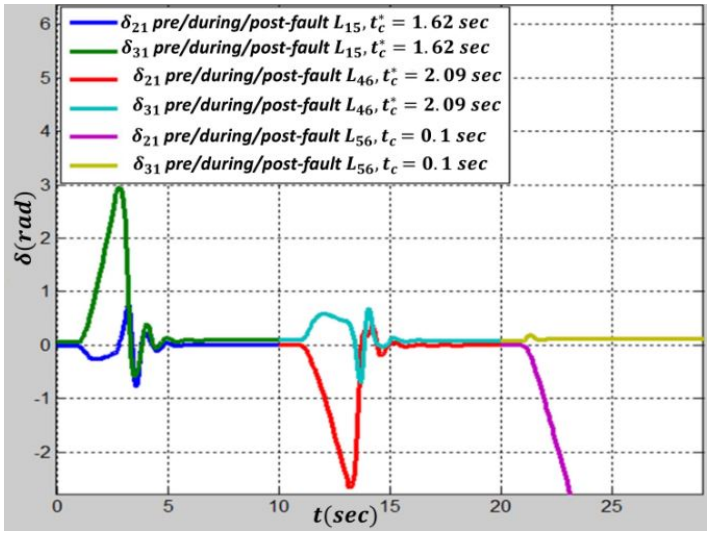

Figure 4.7 PS_2 relative angles under A_1

For comparing LoR of PS_1 and PS_2, under attack scenario A_1, Table 4.7 shows the nearest distance from equilibrium to boundary associated with each RoS, as well as RoS_R. PS_2 has higher RoS_R (15.018%) as opposed to PS_1 (4.507%). Also, $LoP_R(PS_1, A_1) < LoP_R(PS_2, A_1)$. Hence, $LoR(PS_1, A_1) > LoR(PS_2, A_1)$. Thus, PS_1 is more resilient than PS_2 to A_1.

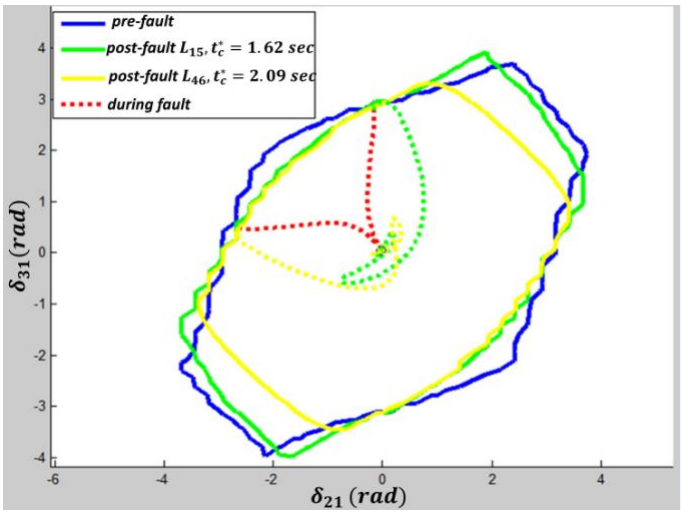

Figure 4.8 PS_2 *modal-RoS* under A_1

Table 4.7 Size of each RoS (rad) and $RoS_R(\%)$ under A_1

$A_1:$	D_I	$D_{P(L_{15})}$	$D_{P(L_{46})}$	RoS_R
PS_1	2.840	2.712	2.712	4.507
PS_2	2.710	2.374	2.303	15.018

Similarly, under another attack scenario, $A_2 : L_{56} \rightarrow L_{14} \rightarrow L_{46}$, *modal-RoS* for PS_1 (respectively, PS_2) are shown in Fig. 4.9 (respectively, Fig. 4.10).

Table 4.8 shows the nearest distance between equilibrium and boundary associated with each RoS, as well as RoS_R. PS_2 has higher RoS_R (0.996%) as opposed to PS_1 (-0.634%). Also, for PS_1, $LoP_R = 28.57\%$, while for PS_2, $LoP_R = 45.71\%$, so $LoR(PS_1, A_2) > LoR(PS_2, A_2)$, implying that topology PS_1 is more resilient as compared to PS_2, under both the attack scenarios.

Table 4.8 Size of each RoS (rad) and $RoS_R(\%)$ under A_2

$A_2:$	D_I	$D_{P(L_{56})}$	$D_{P(L_{14})}$	RoS_R
PS_1	2.840	2.878	2.858	-0.634
PS_2	2.710	2.776	2.683	0.996

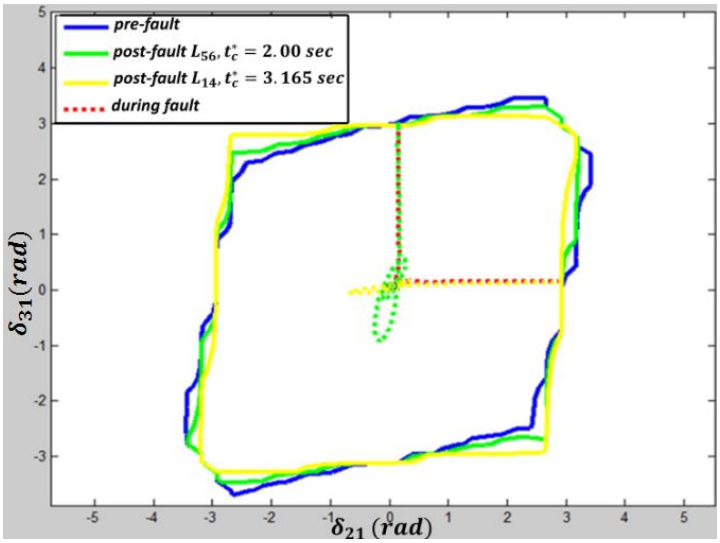

Figure 4.9 PS_1 *modal-RoS* under A_2

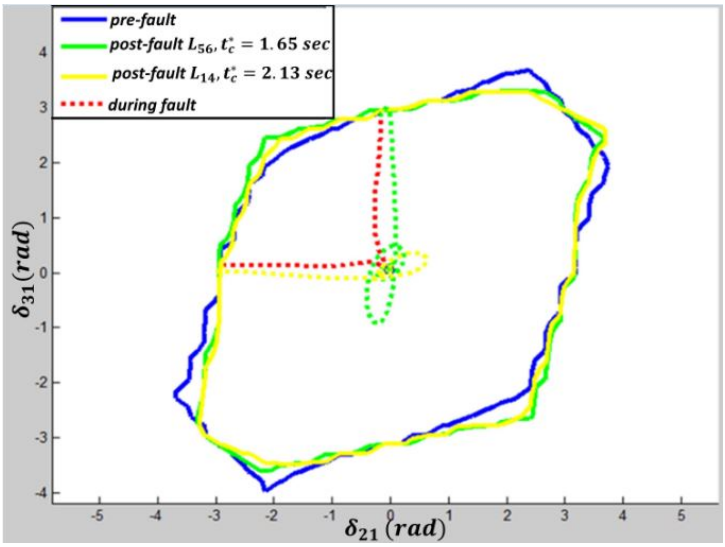

Figure 4.10 PS_2 *modal-RoS* under A_2

CHAPTER 5. MODEL-BASED GENERATION OF ATTACK SEQUENCES

In the previous chapter, we introduced the level of resilience of a given system, assessed under various attack scenarios. In this chapter we present a model-based approach for generating such attack scenarios. This requires a comprehensive description of the system model (describing architecture and connectivity, components and behaviors, assets, defenses, vulnerabilities, atomic attacks), as well as of security/resiliency properties being investigated. A state exploration based approach has been proposed to find all behaviors/paths of the model leading to those reachable states where the specified security/resiliency properties are violated. An *attack graph* is a collection of all paths from initial states to such reachable violating states. Once attack graphs are generated for systems, further analysis can be done to compare the resiliency levels against the attack scenarios within the graph. Each path in the graph is a single attack scenario (sequence of attack/recovery actions resulting in system compromise), and has an associated (LoR). Then, by enumerating against all possible attack scenarios, we can compare their worst case or average LoR. In the following sections, we present a model-based attack graph generation approach and its implementation.

5.1 System Description for Network Example

As a concrete example, we adapt the network system from [Jha et al. (2002); Sheyner et al. (2002)] shown in Fig. 5.1. There are three hosts: 0, where attacker is located, whereas 1 and 2 are two target hosts. Also, there exists a firewall separating the targets from the rest of the internet. Host 1 is running ftp, and $sshd$, while Host 2 is running $database$ and ftp. An IDS monitors the network traffic between the target hosts and the outside world.

There are 4 possible atomic attacks:

1. *sshd buffer overflow (sbo)*: This attack immediately gives a root shell on the victim machine to the remote user. This attack has both stealthy and detectable variants.

2. *ftp_rhosts (ftpr)*: Using an ftp vulnerability, the attacker creates an .rhosts file in the ftp home directory, creating a remote login trust relationship between its machine and the target machine. This attack is stealthy.

3. *remote login (rlog)*: Using an existing remote login trust relationship between two machines, the attacker logs into from one machine to another and gets a user shell without supplying a password. This attack is detectable provided there is an IDS assigned to it.

4. *local buffer overflow (lbo)*: Exploiting a buffer overflow vulnerability on a setuid root file gives attacker root access on a local machine. This attack is stealthy.

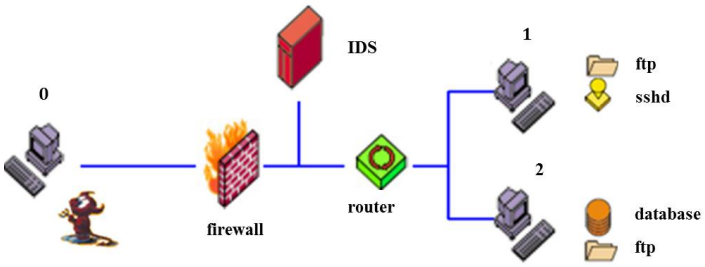

Figure 5.1 Network example

For a detectable atomic attack with an IDS assigned, the IDS triggers an alarm upon the detection; if an attack is stealthy, the IDS misses it. The hosts are subject to three vulnerabilities:

1. wdir: a writable ftp home directory (this vulnerability is exploitable by ftp_rhosts attack)

2. fshell: an executable command shell assigned to the ftp user name (this vulnerability is also exploitable by ftp_rhosts attack)

3. xterm: the xterm executable is vulnerable to the local buffer overflow attack.

The firewall does not place any access control restrictions on the flow of network traffic. The traffic flow between $(0;1)$ and between $(0;2)$ are monitored by a network-based IDS, whereas the flow between $(1;2)$ is not monitored.

5.1.1 Formal System Description for Network Example

We can formally specify the system as follows:

1. Set of hosts $H = 0, 1, 2$; variable $i \in \{0, 1, 2\}$ (static)
2. System connectivity, $C \subseteq H \times H$; Boolean $c_{ij} = 1$ iff host i connected to host j (static)
3. System services S; Boolean $s_i = 1$ iff service $s \in \{ftp, sshd, data\}$ is running on host i (dynamic)
4. System vulnerabilities V; Boolean $v_i = 1$ iff vulnerability $v \in \{dir, fshell, xterm\}$ exists on host i (static)
5. Attack instances $A_I \subseteq A \times H \times H$; labeled $a_{ij} \equiv$ attack a from source i to target j, $a \in \{sbo, ftpr, rlog, lbo\}$, sbo:= sshd buffer overflow, $ftpr$:=ftp_rhosts, $rlog$:=remote login, lbo:=local buffer overflow
6. Trust relation $T \subseteq H \times H$; Boolean $t_{ij} = 1$ iff i is trusted by j (dynamic)
7. Attacker level of privilege L on host i; variable $l_i \in \{none, user, root\}$ (dynamic)
8. Intrusion detection system $IDS : A \times H \times H \rightarrow \{0, 1\}$; Boolean $ids(a_{ij}) = 1$ iff attack a from source i to target j is detectable (static).
9. A global Boolean d_g tracks whether an IDS alarm has been triggered for any previously executed atomic attack (dynamic).
10. Attack pre-conditions:
    - $Pre_{(sbo_{ij})} \equiv c_{ij} = 1 \wedge (l_i \geq user) \wedge (l_j < root) \wedge (sshd_j = 1)$
    - $Pre_{(ftpr_{ij})} \equiv c_{ij} = 1 \wedge (l_i \geq user) \wedge (\exists k \in H : t_{kj} = 0) \wedge (ftp_j = 1) \wedge (wdir_j = 1 \wedge fshell_j = 1)$

- $Pre_{(rlog_{ij})} \equiv c_{ij} = 1 \wedge (l_i \geq user) \wedge (l_j = none) \wedge (t_{ij} = 1)$
- $Pre_{(lbo_{ij})} \equiv c_{ij} = 1 \wedge (l_j = user) \wedge (xterm_j = 1)$

11. Attack post-conditions:

- $Post_{(sbo_{ij})} \equiv (l_j = root) \wedge (sshd_j = 0) \wedge ((i = d_g = 0) \Rightarrow (d_g = 0)(d_g = 1))$
- $Post_{(ftpr_{ij})} \equiv (\forall k \in H : t_{kj} = 1)$
- $Post_{(rlog_{ij})} \equiv (l_j = user) \wedge (i = 0 \Rightarrow d_g = 1)$
- $Post_{(lbo_{ij})} \equiv (l_j = root)$

12. Initial States: $l_0 = root \wedge (l_1 = l_2 = none) \wedge (\forall ij \in H \times H : t_{ij} = 0) \wedge (ftp_1 = ftp_2 = sshd_1 = data_2 = 1) \wedge d_g = 0$.

13. The security property P is that the attacker on host 2 has privilege level below root or gets detected. This can then be described by a Computational Tree Logic (CTL): $P \equiv AG(l_2 < root \vee d_g = 1) \equiv AG(\neg(l_2 = root \wedge d_g = 0))$.

5.1.2 Formal System Model for Network Example

A finite state model M of this example is shown in parts in Fig. 5.2, where each state corresponds to values of dynamic variables, namely, privilege level, trust levels, services running, and attack detectability status, whereas each state transition corresponds to a single atomic attack. The model M depicts their evolution under the sequence of atomic attacks. A typical transition from state s_1 to state s_2 corresponds to an atomic attack whose preconditions are satisfied in s_1 and whose post-conditions hold in state s_2.

5.1.3 Model-based Attack Graph Generation for Network Example

An attack graph is a subgraph of the system model, consisting of all paths from the initial state leading to a final reachable state where the security property of interest is violated. Fig. 5.3 shows the attack graph, a subgraph of the model M of Fig. 5.2.

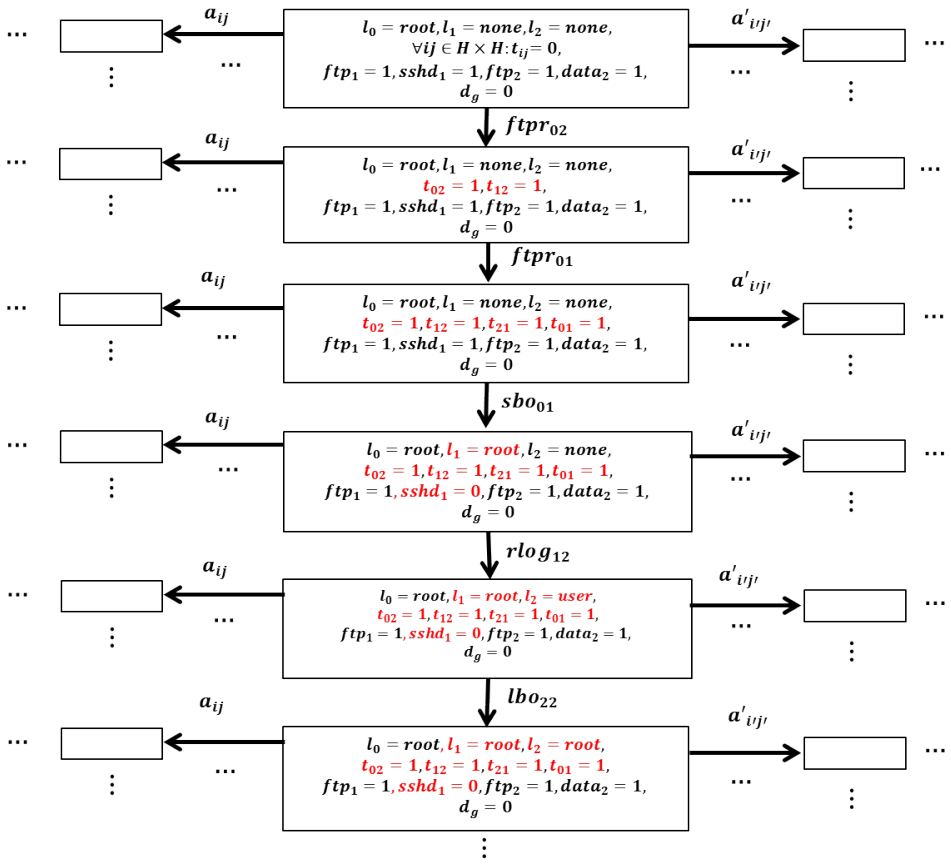

Figure 5.2 Network state model M

5.2 Initial work towards Implementation of Automated Attack Graph Generation

For implementation of automated attack graph generation, we have started to explore *Architecture Analysis and Design Language* (*AADL*) [SEI (2004)] for system and property description.

Fig. 5.4 shows a conceptual architecture of the main components and their interactions. An architecture model defined in *AADL* describes the system components and their interfaces/connections within the *Open-source AADL tool environment* (*Osate*2) [Carnegie-Mellon-University (2016)]. *Osate*2 is an upgraded *Eclipse* [Eclipse-Foundation (2004)] based platform, an open source development platform for building, deploying and managing Java, C/C++, and Personal Home Page(*PHP*) applications. *Osate*2 extends the *Eclipse* platform with textual

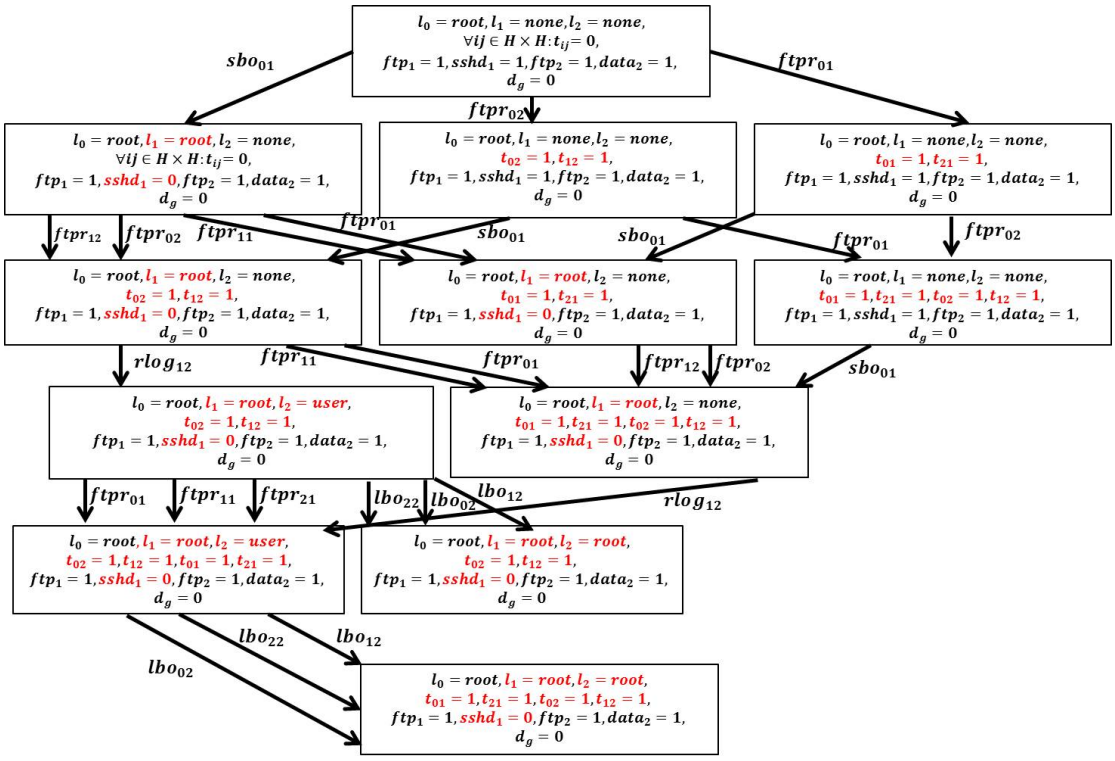

Figure 5.3 Network example attack graph

and graphical editors for *AADL*, and support for *Extensible Markup Language (XML)* based interchange formate [(W3C) (2004)] for *AADL*, based on its Meta model specification. *Osate*2 also supports *AADL* plug-ins [Feiler and Gluch (2012)].

Within an *AADL* architecture level model, extensions can be provided through *Annexes* [Feiler and Gluch (2012)]. An Annex sub-language can provide new categories of model elements such as behaviors and properties of the components. We use *AADL Annex AGREE* (*Assume Guarantee Reasoning Environment*) plug-in [Rockwell-Collins and Uof-Minnesota (2016a)], an assume-guarantee-style compositional reasoner that supports behavior and property description and provides *AADL* to *Luster* translation. *AGREE* relies on model-checking tool *JKind* [Sheeran et al. (2000)]- a k-induction bounded model checker for the verification of *Luster* model. The k-induction in turn uses a back-end Satisfiability Modulo Theories (*SMT*) solver *Z*3 [Rockwell-Collins and Uof-Minnesota (2016b)]. A verified property is guaranteed to be true for all runs of the system. A falsified property is reported with an explicit counterexample demonstrating the property violation.

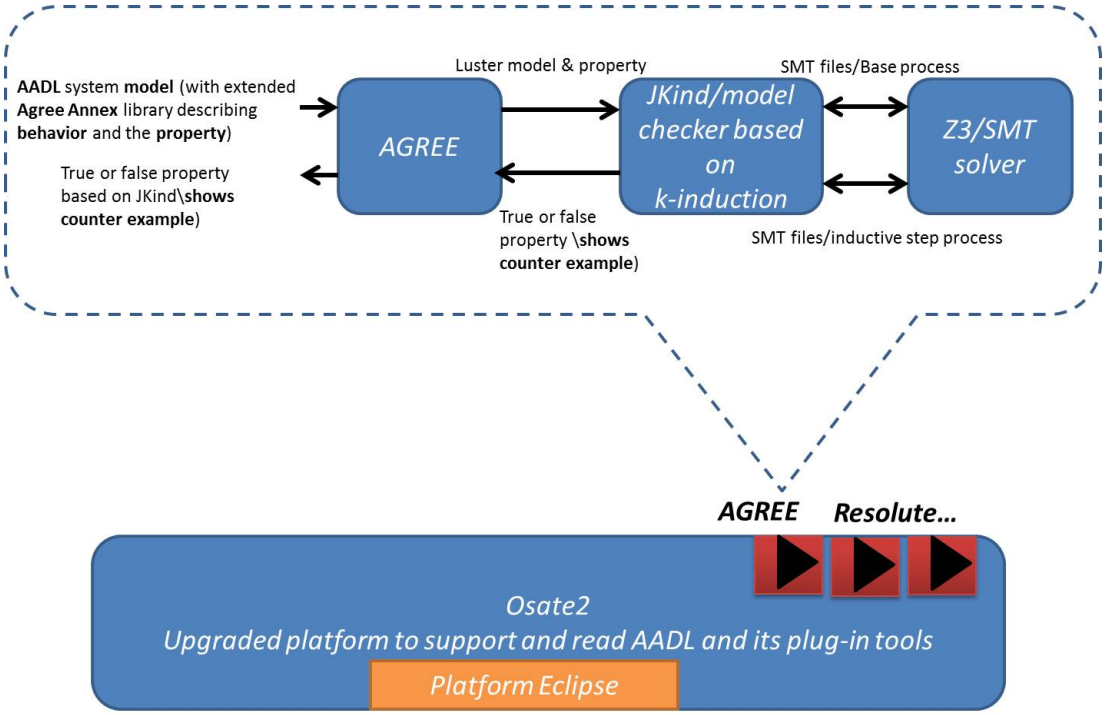

Figure 5.4 Conceptual architecture for generating an attack scenario

For our networked system application of Section 5.1, we are able to use the $AADL/AGREE$ tool to generate one instance of an attack scenario, i.e., counter example $CE_1 := sbo_{01} - > ftpr_{02} - > rlog_{12} - > lbo_{02}$ resulting in the violation of the security property P. By encoding this discovered counter example CE_1 in disjunct with the property P being checked, namely $P \vee CE_1$, a next counterexample satisfies $\neg(P \vee CE_1) = \neg P \wedge \neg CE_1$, i.e., which is a counterexample of P and is different from CE_1. This yielded a new counterexample $CE_2 := sbo_{01} - > ftpr_{12} - > rlog_{12} - > lbo_{02}$. By repeating this process multiple (but finitely in number) CEs can be found, yielding the entire "attack graph" for our application example. The corresponding $AADL/AGREE$ code is provided in Appendix B.

Here, we present some tools summarized in [Feiler and Gluch (2012)] that support the creation, analysis and validation of $AADL$ models. The $META$ tool-set [Rockwell-Collins (2012)] extends $OSATE$ and includes $SysML$-$AADL$ translator, a static model verification tool (Lute), and a compositional verification tool ($AGREE$). The tool-set ($TASTE$) [ESA (2016)] provides a set of graphical editors for creating the software/hardware architectures, and the

deployment of the software on the hardware. These editors provide a simplified user interface, eliminating the need to learn the textual syntax of AADL.

For dependability analysis, fault and fault propagation information can be added into an *AADL* model through the *Error Model Annex* [AS5506/1 (2006); Feiler and Rugina (2007)]. Error state machines, and error propagations can be introduced with component specifications to perform safety and reliability analyses, such as Functional Hazard Assessment (*FHA*), Failure Mode and Effect Analysis (*FMEA*), and Mean Time To Failure (*MTTF*) analysis, as demonstrated in [Redman et al. (2010); Feiler et al. (2010)].

The *Behavior Annex* [AS5506/2 (2011)] can be deployed as a way of adding behavioral specifications to components, it can be interfaced with various model checking tools via a common interchange format language for model checkers called *FIACRE* designed both as the target language of model transformation engines from various models such as *SDL*, *UML*, *AADL*, and as the source language of compilers into the targeted verification tool-boxes, namely *CADP* and *Tina*, which can be generated from *AADL* models with behavior specifications [Berthomieu et al. (2008)]

The *Hybrid Annex* [Ahmad et al. (2014)] defines an annex sub-language of *AADL* to allow continuous behavior specifications (e.g., power system dynamics) to be attached to AADL components. The *BLESS* Plug-in [Larson et al. (2012)] for *OSATE*, supports system modeling in *AADL*. The *BLESS Annex* is an extension to the *Behavior Annex*, to specify assertions, including time-related assertions, about system behavior and to verify them with a theorem prover.

CHAPTER 6. SUMMARY AND DISCUSSION

6.1 Summary of Dissertation

In this dissertation, we studied sececy and resiliency quentification for cyber-physical systems. The main contributions of this dissertation are summarized as follows.

1. An information theoretic measure of secrecy loss in systems modeled as partially-observed stochastic discrete event systems (stochastic PODES) in centralized setting, in the presence of a single attacker was presented. The statistical difference, in the form of the Jensen-Shannon Divergence, between the influence of secrets versus covers on the observations, is employed to quantify the loss of secrecy. We further showed that the proposed JSD measure for secrecy loss is equivalent to the mutual information between the distribution over possible observations and that over possible system status (secret versus cover).

2. The computation of the worst-case loss of secrecy as obtained in limit over longer and longer observations was proposed. This computation required developing a recursion relating JSD over length-n sequences to distributions over length-(n-1) sequences through the 1-step dynamics of underlying system model. We also presented an observer-based approach and used to aide the computation of JSD in the limit. Illustrative examples, including one with side channel attack, were provided to demonstrate the proposed computation approach.

3. We extended information theoretic measure for secrecy loss quantification in PODESs to the distributed collusive setting, in the presence of multiple attackers/observers exchanging their observations, respectively. Channel models are introduced to extend the system

model to capture the effect of exchange of observations, and the JSD computation of the centralized case is applied over the extended model to arrive at the measure for secrecy loss. Illustrative examples, were provided to demonstrate the proposed computation approach.

4. We proposed a measure for comparing Level-of-Resilience (LoR) for power systems. This measure is based on comparing systems characteristics: percentage of Region-of-stability-Reduction (RoS_R), percentage of Level-of-Performance-Reduction (LoP_R), and the system recovery-time, under the given attack scenarios. The system state trajectories and RoS evolution are tracked and captured in form of *modal-RoS*. Examples were illustrated to compare the LoR of two different power system topologies under two different attack scenarios. While the results were employed for power systems, they more generally apply to any hybrid dynamical system with both continuous and discrete dynamics, where the discrete state changes (i.e., mode switches) are caused by attack and/or recovery actions. A note about *recovery time*, within which a recovery action can be taken, is that for power systems it corresponds to the time taken to detect and clear faults, which does not vary dramatically from one system topology to another, and so not considered explicitly in the resiliency comparison measure of our example. For general hybrid dynamical systems, however, recovery time can also be included in the comparison measure. The work presented here provides a framework to do so.

5. The level of resilience of a given system is assessed under various attack scenarios. We presented a model-based approach for generating such attack scenarios in automated fashion. This requires a comprehensive description of the system model as well as of security/resiliency properties being investigated. A state exploration based approach has been proposed to find all behaviors/paths of the model leading to those reachable states where the specified security/resiliency properties are violated. Using the $AADL/AGREE$ tool, we demonstrated how this may be accomplished for a networked system application.

6.2 Future Work

1. We have studied the quantification of Centralized/Distributed Secrecy of a given stochastic discrete event system. Now when there is flexibility to exercise control in order to meet some desired secrecy level, designing such control strategies remains a future research direction.

2. Developing a software tool for JSD computation, and performing application studies is also a future possible direction: knowing the JSD value can help an engineer to perform secrecy analysis of a system, and revisit the system design for achieving satisfactory metric.

3. For the general adoption of the proposed measure for comparing Level-of-Resilience (LoR), one must further provide computationally efficient tool for LoR comparison, and this can be a subject of further study. For example, one could consider a simplified metric for measuring level of stability such as *stability margin*, which is easier to compute than region-of-stability.

4. Having proposed ways to measure how secure and resilient a given system is against passive/active attacks, we may next look for ways to enhance these properties, by first identifying the system weaknesses. Conducting a vulnerability assessment (VA) is a key step, that includes the process of identifying, quantifying, and prioritizing (or ranking) the vulnerabilities in a system. A potential new research direction can be to survey the existing VA methods and tools and propose new features/extensions/applications to expose vulnerabilities, so as to make fixes to enhance *secrecy/resiliency*. For instance, vulnerability identification can be done using various static analysis techniques such as static code analysis ranging from scanning the source text for simple patterns, to data flow analysis, to advanced model checking, and dynamic analysis such as fault injection/fuzzing. The severity of exploiting these vulnerabilities can be further quantified and prioritized. Examples of some well known projects/database libraries of system vulnerabilities/weaknesses are CCE [MITRE (2014)], CAPEC [MITRE (2015a)],

CVE [MITRE (2015b)], CWE [MITRE (2015c)], PLOVER [Christey (2006)], OWASP [OWASP-Foundation (2015)], and WASC [WASC (2005)]. Consider for example the vulnerability assessment for *Supervisory Control and Data Acquisition* (SCADA) systems. The vulnerability tests can be performed on Modbus (an application layer protocol used in SCADA networks) to discover weaknesses/vulnerabilities associated with the lack of encryption or any other security measures in such a protocol [Zhu et al. (2011)].

5. Using the $AADL/AGREE$ tool, we were able to generate multiple (but finitely in number) attack scenarios. Future work will involve the automatic generation of the entire "attack graph". $AGREE$ limits model-checker to $JKind$ which can model-check only the invariant properties (that hold in all states), and so one should also look for tools other than $AGREE$ that interface $AADL$ with different model-checkers. For example, one may consider the tool $COMPASS$ [Bozzano et al. (2009)] that interfaces $AADL$ to $NuSMV$.

6. In our work, we focused on the attack scenarios that are discrete in nature, e.g., the attack/recovery actions applied for power systems. Future work may also involve continuous attack actions. For instance, attacker may continuously vary the temperature magnitude, or a magnetic field. Such more general attack scenarios can be investigated in the future.

APPENDIX A. MATLAB CODE FOR POWER SYSTEM EXAMPLE OF CHAPTER 4

Here, we provide the Matlab source code adapted from [Saadat (1999)], and modified to simulate a sequence of attack/fault and recovery actions of attack scenario A_1, for two different power systems PS_1 (Fig. 4.1(a)), and PS_2 (Fig. 4.1(b)), to compare their LoR. In this attack scenario, three lines are faulted in the sequence: $L_{15} \rightarrow L_{46} \rightarrow L_{56}$, and cleared within certain clearance times by the simultaneous opening of breakers at both ends of these lines. We also provide a modified Matlab code used for generating the *modal RoS*. This code is adapted from [Jin et al. (2010)] tool, which generates the RoS for dynamical systems based on level set reachability analysis. This tool is a modified version of the tool-box of Level Set Methods [Mitchell (2004)].

%Program “trstab” is used in conjunction with the power flow program “lfnewton”.
%Power flow program provides the power, voltage magnitude and phase
%angle at each bus. Also, the load admittances are returned in a
%matrix named “yload”. In addition to the required power flow data,
%transient reactance, and inertia constant of each machine must be
%specified. This is defined in a matrix named “gendata”. Each row
%contains the bus number to which a generator is connected, armature
%resistance, transient reactance, and the machine inertia constant.
%Program “trstab” obtains the pre-fault bus admittance matrix including
%the load admittances. Voltage behind transient reactants are also obtained.
%The reduced admittance matrix before, during, and after fault are found.
%Machine equations are expressed in state variable form and the MATLAB

%ode23 is used to solve the multi-machine equations. The phase angle

%difference of each machine with respect to the slack bus is plotted.

%The simulation can be repeated for a different fault clearance time, or

%a different fault location.

%%%%%%%%%%%%%%%%%%%

We run the main program "trstab" for PS_1 (respectively, PS_2) three times, to simulate the attack sequence $L_{15} \rightarrow L_{46} \rightarrow L_{56}$, with clearance times: 3, 2.9, and $1.1sec$, and collect the data for relative rotor angle responses.

%%%%%%%%%%%%

CHP11EX7 %the system data in pre-fault mode, comment this function when running "removeL1L5" mode, and "removeL1L5L4L6" mode separately.

%removeL1L5%uncomment this function to insert new system data after removing line1-line5, then rerun "trstab".

%removeL1L5L4L6 %uncomment this function to insert new system data after removing line1-line5, and line4-line6, then rerun "trstab".

global Pm f H E Y th ngg

f=60;

ngr=gendata(:,1);

ngg=length(gendata(:,1));

for k=1:ngg

zdd(ngr(k))=gendata(k, 2)+j*gendata(k,3);

H(k)=gendata(k,4);

end

%%%Ep, and Pm are assumed constant, and equal to the pre-fault mode generated values, the following commands are commented when running"trstab" against $L_{46} \rightarrow L_{56}$ attacks/faults.

for k=1:ngg

I=conj(S(ngr(k)))/conj(V(ngr(k)));

Ep(k) = V(ngr(k))+zdd(ngr(k))*I;

Px(k)=real(S(ngr(k)));

end

E=abs(Ep); d0=angle(Ep);

%%%%%%%%%%

%the following values for Ep, and Pm are obtained from the pre-fault mode, uncomment them when running "trstab" against $L_{46} \rightarrow L_{56}$ attacks/faults.

%Ep=[1.2930 1.2006 1.1243];

%Pm=[-0.2123 0.0787 0.1335];

%E=Ep;

%d0=[0.1297 0.1940 0.2499];%the rotor angle data is obtained from the first run of "trstab" against the first fault L_{15}.

%the second fault in the sequence is applied with an initial point being the post fault L_{15} data.

%d0=[0.1233 0.1817 0.2614]; %the rotor angle data is obtained from the second run of "trstab" against the second fault L_{46}.

%the third fault in the sequence is applied with an initial point being the post fault L_{46} data.

%%%%%%%%%%

for k=1:ngg

nl(nbr+k) = nbus+k;

nr(nbr+k) = gendata(k, 1);

R(nbr+k) = real(zdd(ngr(k)));

X(nbr+k) = imag(zdd(ngr(k)));

Bc(nbr+k) = 0;

a(nbr+k) = 1.0;

yload(nbus+k)=0;

end

nbr1=nbr; nbus1=nbus;

nbrt=nbr+ngg;

nbust=nbus+ngg;

linedata=[nl, nr, R, X, -j*Bc, a];

[$Ybus, Ybf$]=ybusbf(linedata, yload, nbus1,nbust);

```
fprintf('\n Pre-fault reduced bus admittance matrix \n')
Ybf
Y=abs(Ybf); th=angle(Ybf);
%%%%%%%%%%
%comment the following commands when running "trstab" against L46 → L56 attacks/faults
Pm=zeros(1, ngg);
disp(['G(i)E''(i)d0(i)Pm(i)'])
for ii = 1:ngg
for jj = 1:ngg
Pm(ii) = Pm(ii) + E(ii)*E(jj)*imag(Ybf(ii, jj))*sin(d0(ii)-d0(jj));
end,
fprintf(' %g', ngr(ii)), fprintf(' %8.4f',E(ii)), fprintf(' %8.4f', 180/pi*d0(ii))
fprintf(' %8.4f \n',Pm(ii))
end
%%%%%%%%%%
respfl='y';
t3 = 0;
w3=zeros(1, length(d0));
x3 = [d0, w3];
tol=0.0001;
tspan=[t3, 1];
[t3, xm] =ode23('pfpek', tspan, x3);
while respfl =='y' | respfl=='Y'
nf=input('Enter faulted bus No. - > ');
fprintf('\n Faulted reduced bus admittance matrix \n')
Ydf=ybusdf(Ybus, nbus1, nbust, nf)
[Yaf]=YBUSAF(linedata, yload, nbus1,nbust, nbrt);
fprintf('\n Postfault reduced bus admittance matrix \n')
Yaf
```

```
resptc='y';
while resptc =='y' | resptc=='Y'
tc=input('Enter clearing time of fault in sec. tc = ');
tf=input('Enter final simulation time in sec. tf = ');
clear t x del
% during fault response
t0 = 1;
w0=zeros(1, length(d0));
x0 = [d0, w0];
tol=0.0001;
Y=abs(Ydf); th=angle(Ydf);
tspan=[t0, tc];
[t1, xf] =ode23('dfpek', tspan, x0);
%after fault response x0c =xf(length(xf), :);
Y=abs(Yaf); th=angle(Yaf);
tspan = [tc, tf]; [t2, xc] =ode23('afpek', tspan, x0c);
t =[t3; t1; t2]; x =[xm; xf; xc]; fprintf('\n Fault is cleared at %4.3f Sec. \n', tc)
for k=1:nbus
if kb(k)==1
ms=k; else, end
end
fprintf('\n Phase angle difference of each machine \n')
fprintf('with respect to the slack in rad.\n')
fprintf(' t - sec')
kk=0;
for k=1:ngg
if k
sim =ms
kk=kk+1;
```

del(:,kk)=(x(:,k)-x(:,ms));

fprintf(' d(%g,',ngr(k)), fprintf('%g)', ngr(ms))

else, end

end

fprintf(' \n')

disp([t, del])

h=figure; figure(h)

plot(t, del,'linewidth',3)

legend('del_{21}','del_{31}')

xlabel('t, sec'), ylabel('Delta, rad'), grid

resp=0;

while strcmp(resp, 'n')~ =1 & strcmp(resp, 'N')~ =1 & strcmp(resp, 'y')~ =1 & strcmp(resp, 'Y')~ =1

resp=input('Another clearing time of fault? Enter "y" or "n" within quotes − > ');

if strcmp(resp, 'n')~ =1 & strcmp(resp, 'N')~=1 & strcmp(resp, 'y')~ =1 & strcmp(resp, 'Y')~ =1

fprintf('\n Incorrect reply, try again \n\n'), end

end

resptc=resp;

end

resp2=0;

while strcmp(resp2, 'n')~ =1 & strcmp(resp2, 'N')~ =1 & strcmp(resp2, 'y')~ =1 & strcmp(resp2, 'Y')~ =1

resp2=input('Another fault location: Enter "y" or "n" within quotes − > ');

if strcmp(resp2, 'n')~ =1 & strcmp(resp2, 'N')~ =1 & strcmp(resp2, 'y')~ =1 & strcmp(resp2, 'Y')~ =1

fprintf('\n Incorrect reply, try again \n\n'), end

respf1=resp2;

end

```
if respf1==’n’ | respf1==’N’, return, else, end
end
%%%%%%%%%%%%%%%%%%%
CHP11EX7
%%%%%%%%%%%%%%%%%%%
basemva = 100; accuracy = 0.0001; maxiter = 10;
Busd=[ 1 1 1.06 0.0 00.00 00.00 0.00 00.00 0 0 0
2 2 1.04 0.0 00.00 00.00 150.00 00.00 0 140 0
3 2 1.03 0.0 00.00 00.00 100.00 00.00 0 90 0
4 0 1.0 0.0 100.00 70.00 00.00 00.00 0 0 0
5 0 1.0 0.0 90.00 30.00 00.00 00.00 0 0 0
6 0 1.0 0.0 160.00 110.00 00.00 00.00 0 0 0];
linedata=[1 4 0 0.225 0.0065 1.0
1 5 0 0.105 0.0045 1.0
1 6 0 0.215 0.0055 1.0
2 4 0 0.035 0.0000 1.0
3 5 0 0.042 0.0000 1.0
4 6 0 0.125 0.0035 1.0
5 6 0 0.175 0.0300 1.0];
Lfybus
Lfnewton
Busout
% Gen. Ra Xd’ H
gendata=[ 1 0 0.20 20
2 0 0.15 4
3 0 0.25 5];
%%%%%%%%%%%%%%%%%%%
%this program obtains th Bus Admittance Matrix for power flow solution [Saadat (1999)].
Lfybus
```

```
%%%%%%%%%%%%%%%%%%%
j=sqrt(-1); i = sqrt(-1);
nl = linedata(:,1); nr = linedata(:,2); R = linedata(:,3);
X = linedata(:,4); Bc = j*linedata(:,5); a = linedata(:, 6);
nbr=length(linedata(:,1)); nbus = max(max(nl), max(nr));
Z = R + j*X; y= ones(nbr,1)./Z;
for n = 1:nbr
if a(n) <= 0 a(n) = 1; else end
Ybus=zeros(nbus,nbus);
for k=1:nbr;
Ybus(nl(k),nr(k))=Ybus(nl(k),nr(k))-y(k)/a(k);
Ybus(nr(k),nl(k))=Ybus(nl(k),nr(k));
end
end
for n=1:nbus
for k=1:nbr
if nl(k)==n
Ybus(n,n) = Ybus(n,n)+y(k)/(a(k)^2) + Bc(k);
elseif nr(k)==n
Ybus(n,n) = Ybus(n,n)+y(k) +Bc(k);
else, end
end
end
clear Pgg
%%%%%%%%%%%%%%%%%%%
%power flow solution by Newton-Raphson method [Saadat (1999)].
Lfnewton
%%%%%%%%%%%%%%%%%%%
ns=0; ng=0; Vm=0; delta=0; yload=0; deltad=0;
```

```
nbus = length(Busd(:,1));
for k=1:nbus
n=Busd(k,1);
kb(n)=Busd(k,2); Vm(n)=Busd(k,3); delta(n)=Busd(k, 4);
Pd(n)=Busd(k,5); Qd(n)=Busd(k,6); Pg(n)=Busd(k,7); Qg(n) = Busd(k,8);
Qmin(n)=Busd(k, 9); Qmax(n)=Busd(k, 10);
Qsh(n)=Busd(k, 11);
if Vm(n) <= 0 Vm(n) = 1.0; V(n) = 1 + j*0;
else delta(n) = pi/180*delta(n);
V(n) = Vm(n)*(cos(delta(n)) + j*sin(delta(n)));
P(n)=(Pg(n)-Pd(n))/basemva;
Q(n)=(Qg(n)-Qd(n)+ Qsh(n))/basemva;
S(n) = P(n) + j*Q(n);
end
end
for k=1:nbus
if kb(k) == 1, ns = ns+1; else, end
if kb(k) == 2 ng = ng+1; else, end
ngs(k) = ng;
nss(k) = ns;
end
Ym=abs(Ybus); t = angle(Ybus);
m=2*nbus-ng-2*ns;
maxerror = 1; converge=1;
iter = 0;
clear A DC J DX
while maxerror >= accuracy & iter <= maxiter
for i=1:m
for k=1:m
```

```
A(i,k)=0;
end, end
iter = iter+1;
for n=1:nbus
nn=n-nss(n);
lm=nbus+n-ngs(n)-nss(n)-ns;
J11=0; J22=0; J33=0; J44=0;
for i=1:nbr
if nl(i) == n | nr(i) == n
if nl(i) == n, l = nr(i); end
if nr(i) == n, l = nl(i); end
J11=J11+ Vm(n)*Vm(l)*Ym(n,l)*sin(t(n,l)- delta(n) + delta(l));
J33=J33+ Vm(n)*Vm(l)*Ym(n,l)*cos(t(n,l)- delta(n) + delta(l));
if kb(n)~=1
J22=J22+ Vm(l)*Ym(n,l)*cos(t(n,l)- delta(n) + delta(l));
J44=J44+ Vm(l)*Ym(n,l)*sin(t(n,l)- delta(n) + delta(l));
else, end
if kb(n) ~= 1 & kb(l) ~=1
lk = nbus+l-ngs(l)-nss(l)-ns;
ll = l -nss(l);
A(nn, ll) =-Vm(n)*Vm(l)*Ym(n,l)*sin(t(n,l)- delta(n) + delta(l));
if kb(l) == 0
A(nn, lk) =Vm(n)*Ym(n,l)*cos(t(n,l)- delta(n) + delta(l));end
if kb(n) == 0
A(lm, ll) =-Vm(n)*Vm(l)*Ym(n,l)*cos(t(n,l)- delta(n)+delta(l)); end
if kb(n) == 0 & kb(l) == 0
A(lm, lk) =-Vm(n)*Ym(n,l)*sin(t(n,l)- delta(n) + delta(l));end
else end
else , end
```

```
end
Pk = Vm(n)^2*Ym(n,n)*cos(t(n,n))+J33;
Qk = -Vm(n)^2*Ym(n,n)*sin(t(n,n))-J11;
if kb(n) == 1 P(n)=Pk; Q(n) = Qk; end
if kb(n) == 2 Q(n)=Qk;
if Qmax(n) ~= 0
Qgc = Q(n)*basemva + Qd(n) - Qsh(n);
if iter <= 7
if iter4 > 2
if Qgc < Qmin(n),
Vm(n) = Vm(n) + 0.01;
elseif Qgc > Qmax(n),
Vm(n) = Vm(n) - 0.01;end
else, end
else,end
else,end
end
if kb(n) ~= 1
A(nn,nn) = J11;
DC(nn) = P(n)-Pk;
end
if kb(n) == 0
A(nn,lm) = 2*Vm(n)*Ym(n,n)*cos(t(n,n))+J22;
A(lm,nn)= J33;
A(lm,lm) =-2*Vm(n)*Ym(n,n)*sin(t(n,n))-J44;
DC(lm) = Q(n)-Qk;
end
end
DX=A\DC';
```

```
for n=1:nbus
nn=n-nss(n);
lm=nbus+n-ngs(n)-nss(n)-ns;
if kb(n) ~= 1
delta(n) = delta(n)+DX(nn); end
if kb(n) == 0
Vm(n)=Vm(n)+DX(lm); end
end
maxerror=max(abs(DC));
if iter == maxiter & maxerror > accuracy
fprintf('\nWARNING: Iterative solution did not converged after ')
fprintf('%g', iter), fprintf(' iterations.\n \n')
fprintf('Press Enter to terminate the iterations and print the results \n')
converge = 0; pause, else, end
end
if converge ~= 1
tech= (' ITERATIVE SOLUTION DID NOT CONVERGE'); else,
tech=(' Power Flow Solution by Newton-Raphson Method');
end
V = Vm.*cos(delta)+j*Vm.*sin(delta);
deltad=180/pi*delta;
i=sqrt(-1);
k=0;
for n = 1:nbus
if kb(n) == 1
k=k+1;
S(n)= P(n)+j*Q(n)
Pg(n) = P(n)*basemva + Pd(n)
Qg(n) = Q(n)*basemva + Qd(n) - Qsh(n);
```

```
Pgg(k)=Pg(n)
Qgg(k)=Qg(n);
elseif kb(n) ==2
k=k+1;
S(n)=P(n)+j*Q(n)
Qg(n) = Q(n)*basemva + Qd(n) - Qsh(n);
Pgg(k)=Pg(n)
Qgg(k)=Qg(n);
end
yload(n) = (Pd(n)- j*Qd(n)+j*Qsh(n))/(basemva*Vm(n)2̂);
end
Busd(:,3)=Vm'; Busd(:,4)=deltad';
Pgt = sum(Pg); Qgt = sum(Qg); Pdt = sum(Pd); Qdt = sum(Qd); Qsht = sum(Qsh);
%%%%%%%%%%%%%%%%%%
%this program prints the power flow solution in a tabulated form on the screen [Saadat (1999)].
Busout
%%%%%%%%%%%%%%%%%%
disp(tech)
fprintf(' Maximum Power Mismatch = %g \n', maxerror)
fprintf(' No. of Iterations = %g \n \n', iter)
head =[' Bus Voltage Angle ——Load—— —Generation— Injected'
' No. Mag. Degree MW Mvar MW Mvar Mvar '
' '];
disp(head)
for n=1:nbus
fprintf(' %5g', n), fprintf(' %7.3f', Vm(n)),
fprintf(' %8.3f', deltad(n)), fprintf(' %9.3f', Pd(n)),
fprintf(' %9.3f', Qd(n)), fprintf(' %9.3f', Pg(n)),
fprintf(%9.3f ', Qg(n)), fprintf(' %8.3f \n', Qsh(n))
```

```
end
fprintf(' \n'), fprintf(' Total ')
fprintf(' %9.3f', Pdt), fprintf(' %9.3f', Qdt),
fprintf(' %9.3f', Pgt), fprintf(' %9.3f', Qgt), fprintf(' %9.3f \n\n', Qsht)
%%%%%%%%%%%%%%%%%%
%this function forms the bus admittance matrix including load admittances before fault [Saadat (1999)].
ybusbf
%%%%%%%%%%%%%%%%%%
function [Ybus, Ybf] = ybusbf(linedata, yload, nbus1, nbust)
global Pm f H E Y th ngg
Lfybus
for k=1:nbust
Ybus(k,k)=Ybus(k,k)+yload(k);
end
YLL=Ybus(1:nbus1, 1:nbus1);
YGG = Ybus(nbus1+1:nbust, nbus1+1:nbust);
YLG = Ybus(1:nbus1, nbus1+1:nbust);
Ybf=YGG-YLG.'*inv(YLL)*YLG;
%%%%%%%%%%%%%%%%%%
%state variable representation of the multi-machine system of pre-fault, during-fault, and post-fault modes [Saadat (1999)].
pfpek
%dfpek
%afpek
%%%%%%%%%%%%%%%%%%
function xdot = pfpek(t,x)
global Pm f H E Y th ngg
Pe=zeros(1, ngg);
```

```
for ii = 1:ngg
for jj = 1:ngg
Pe(ii) = Pe(ii) + E(ii)*E(jj)*Y(ii, jj)*sin(th(ii, jj))*sin(x(ii)-x(jj));
%we assume that the system is lossless, so the transfer admittance is purely imaginary.
end, end
for k=1:ngg
xdot(k)=x(k+ngg);
xdot(k+ngg)=(pi*f)/H(k)*(Pm(k)-Pe(k)-0.12*x(k+ngg));
end
xdot=xdot';
%%%%%%%%%%%%%%%%%%
%this function forms the bus admittance matrix including load admittances during fault [Saadat (1999)].
ybusdf
%%%%%%%%%%%%%%%%%%
function Ypf=ybusdf(Ybus, nbus1, nbust, nf)
global Pm f H E Y th ngg
nbusf=nbust-1;
Ybus(:,nf:nbusf)=Ybus(:,nf+1:nbust);
Ybus(nf:nbusf,:)=Ybus(nf+1:nbust,:);
YLL=Ybus(1:nbus1-1, 1:nbus1-1)
YGG = Ybus(nbus1:nbusf, nbus1:nbusf)
YLG = Ybus(1:nbus1-1, nbus1:nbusf)
Ypf=YGG-YLG.'*inv(YLL)*YLG;
%%%%%%%%%%%%%%%%%%
%this function forms the bus admittance matrix including load admittances after removal of faulted line [Saadat (1999)].
ybusaf
%%%%%%%%%%%%%%%%%%
```

```
function [Yaf]=ybusaf(linedata, yload, nbus1,nbust, nbrt);
global Pm f H E Y th ngg
nl=linedata(:, 1); nr=linedata(:, 2);
remove = 0;
while remove ~= 1
fprintf('\n Fault is cleared by opening a line. The bus to bus nos. of the \n')
fprintf('line to be removed must be entered within brackets, e.g. [5, 7] \n')
fline=input('Enter the bus to bus Nos. of line to be removed - > ');
nlf=fline(1); nrf=fline(2);
for k=1:nbrt
if nl(k)==nlf & nr(k)==nrf
remove = 1;
m=k;
else, end
end
if remove ~= 1
fprintf('\n The line to be removed does not exist in the line data. try again. \n \n')
end
end
linedat2(1:m-1,:)= linedata(1:m-1,:);
linedat2(m:nbrt-1,:)=linedata(m+1:nbrt,:);
linedat0=linedata;
linedata=linedat2;
Lfybus
for k=1:nbust
Ybus(k,k)=Ybus(k,k)+yload(k);
end
YLL=Ybus(1:nbus1, 1:nbus1);
YGG = Ybus(nbus1+1:nbust, nbus1+1:nbust);
```

```
YLG = Ybus(1:nbus1, nbus1+1:nbust);
Yaf=YGG-YLG.'*inv(YLL)*YLG;
linedata=linedat0;
%%%%%%%%%%%%%%%%%%
```

The following modified Matlab code is used to generate the *modal-RoS* for PS_1 (respectively, PS_2) under attack scenario A_1, which is adapted from [Jin et al. (2010)] and [Mitchell (2004)].

```
%%%%%%%%%%%%%%%%%%
function [data, g, data0] = SingleInfinite(accuracy);
%parameters:
%reach Implicit surface function for unsafe (reach) set in mode 1.
%g Grid structure on which data is computed.
%avoid Implicit surface function for safe to switch set (avoid)
%in mode 1.
%data0 Implicit surface function for target set.
run ('addPathToKernel');
% Integration Parameters.
Tmax =5.0;
PlotSteps = 15;
T0 = 0;
SingleStep = 0;
tPlot = (Tmax - T0) / (PlotSteps - 1);
Small = 100 * eps;
DissType = 'global';
% What level set should we view?
level = 0;
fig = figure;
if (nargin <1)
accuracy = 'medium';
end
```

DisplayType = 'contour';

g.dim = 2;

g.min = $[-10; -10]$;

g.max = $[10; 10]$;

g.bdry = @addGhostExtrapolate;

g.N = $[80; 120]$;

g = processGrid(g);

%create initial conditions (rectangle).

data = ShapeSphere(g, [0.0560;0.0783], 0.3);%the pre-fault equilibrium point is obtained from program "trstab".

data0 = data;

data4 = ShapeSphere(g, [0.0643; 0.1202], .3);%the post-fault L_{15} equilibrium point is obtained from Program "trstab".

data00000=data4;

data2 = ShapeSphere(g, [0.0584;0.1381], .3);%the post-fault L_{46} equilibrium point is obtained from Program "trstab".

data000=data2;

%pre-fault mode dynamics following (4.12).

velocitypre = {1.781184*sin(g.xs{1})+(0.93686*sin(g.xs{1}-g.xs{2}))-0.0787; ...

1.438636* sin(g.xs{2})+(0.9368611*sin(g.xs{2}-g.xs{1}))-0.1335};

%post-fault L_{15} mode dynamics following (4.12).

velocitypostb15 = {1.9219432*sin(g.xs{1})+(0.801425*sin(g.xs{1}-g.xs{2}))-0.0787; ...

0.74031204* sin(g.xs{2})+(0.8014251*sin(g.xs{2}-g.xs{1}))-0.1335};

%post-fault L_{46} mode dynamics following (4.12).

velocitypostb46 ={1.78137464*sin(g.xs{1})+(0.3165135048*sin(g.xs{1}-g.xs{2}))-0.0787; ...

0.7870026682* sin(g.xs{2})+(0.3165135048*sin(g.xs{2}-g.xs{1}))-0.1335};

$[final, steptime]$ = findReachSet(g, data0, velocitypre, accuracy, ...

Tmax, tPlot, fig, 1, []);

$[final5, steptime]$ = findReachSet(g, data00000, velocitypostb15, accuracy, ...

Tmax, tPlot, fig, 1, []);

[$final3, steptime$] = findReachSet(g, data000, velocitypostb46, accuracy, ...

Tmax, tPlot, fig, 1, []);

%the collected data for the state variables of pre-fault, during-fault, and post-fault modes, obtained from program "trstab".

xm =[0.1464 0.2024 0.2248 0 0 0

0.1464 0.2024 0.2248 0.0000 0.0000 -0.0000

0.1464 0.2024 0.2248 0.0000 0.0000 -0.0000

0.1464 0.2024 0.2248 0.0000 0.0000 -0.0000

0.1464 0.2024 0.2248 0.0000 0.0000 0.0000

0.1464 0.2024 0.2248 0.0000 -0.0000 0.0000

0.1464 0.2024 0.2248 0.0000 -0.0000 0.0000

0.1464 0.2024 0.2248 0.0000 0.0000 0.0000

0.1464 0.2024 0.2248 0.0000 0.0000 -0.0000

0.1464 0.2024 0.2248 0.0000 0.0000 -0.0000

0.1464 0.2024 0.2248 0.0000 0.0000 -0.0000];

xf =[0.1464 0.2024 0.2248 0 0 0

0.1464 0.2024 0.2248 -0.0000 0.0000 0.0001

0.1464 0.2024 0.2248 -0.0001 0.0001 0.0005

0.1464 0.2024 0.2248 -0.0008 0.0007 0.0025

0.1464 0.2024 0.2248 -0.0038 0.0035 0.0124

0.1463 0.2025 0.2252 -0.0190 0.0171 0.0608

0.1457 0.2031 0.2271 -0.0470 0.0403 0.1459

0.1439 0.2046 0.2326 -0.0862 0.0675 0.2573

0.1395 0.2077 0.2453 -0.1384 0.0922 0.3947

0.1305 0.2129 0.2704 -0.2018 0.0978 0.5477

0.1209 0.2168 0.2961 -0.2457 0.0797 0.6474

0.1110 0.2192 0.3218 -0.2785 0.0494 0.7201

0.1013 0.2203 0.3466 -0.3033 0.0134 0.7749

0.0948 0.2203 0.3633 -0.3172 -0.0132 0.8057

0.0879 0.2197 0.3807 -0.3297 -0.0419 0.8337

0.0790 0.2181 0.4031 -0.3434 -0.0800 0.8651

0.0674 0.2146 0.4323 -0.3581 -0.1299 0.8996

0.0528 0.2082 0.4690 -0.3724 -0.1908 0.9349

0.0346 0.1973 0.5148 -0.3859 -0.2612 0.9696

0.0113 0.1795 0.5732 -0.3985 -0.3382 1.0033

-0.0212 0.1489 0.6553 -0.4121 -0.4185 1.0367

-0.0600 0.1073 0.7524 -0.4268 -0.4730 1.0628

-0.0975 0.0654 0.8449 -0.4417 -0.4932 1.0790

-0.1344 0.0248 0.9341 -0.4571 -0.4943 1.0895

-0.1744 -0.0174 1.0281 -0.4738 -0.4871 1.0970

-0.2217 -0.0646 1.1358 -0.4917 -0.4793 1.1026

-0.2774 -0.1178 1.2586 -0.5085 -0.4795 1.1066

-0.3450 -0.1813 1.4037 -0.5224 -0.4928 1.1094

-0.4077 -0.2410 1.5359 -0.5302 -0.5106 1.1108

-0.4797 -0.3114 1.6861 -0.5355 -0.5290 1.1117

-0.5873 -0.4190 1.9085 -0.5411 -0.5427 1.1123

-0.6814 -0.5133 2.1012 -0.5453 -0.5461 1.1125

-0.7320 -0.5639 2.2042 -0.5472 -0.5467 1.1126];

xc =[-0.7320 -0.5639 2.2042 -0.5472 -0.5467 1.1126

-0.7349 -0.5668 2.2099 -0.5311 -0.5191 1.0259

-0.7467 -0.5779 2.2302 -0.4633 -0.4222 0.6892

-0.7575 -0.5876 2.2438 -0.3985 -0.3582 0.4098

-0.7666 -0.5960 2.2512 -0.3391 -0.3211 0.1886

-0.7738 -0.6032 2.2535 -0.2867 -0.3012 0.0180

-0.7790 -0.6089 2.2527 -0.2453 -0.2908 -0.1033

-0.7827 -0.6137 2.2502 -0.2109 -0.2835 -0.1968

-0.7868 -0.6196 2.2448 -0.1679 -0.2736 -0.3069

-0.7905 -0.6266 2.2350 -0.1171 -0.2563 -0.4313

-0.7933 -0.6341 2.2195 -0.0593 -0.2232 -0.5723

-0.7942 -0.6411 2.1962 0.0052 -0.1619 -0.7386

-0.7932 -0.6449 2.1709 0.0615 -0.0844 -0.8984

-0.7907 -0.6461 2.1430 0.1149 0.0088 -1.0664

-0.7859 -0.6437 2.1045 0.1804 0.1427 -1.2938

-0.7781 -0.6355 2.0533 0.2600 0.3219 -1.5965

-0.7664 -0.6191 1.9855 0.3600 0.5520 -2.0009

-0.7494 -0.5912 1.8943 0.4910 0.8413 -2.5470

-0.7242 -0.5468 1.7674 0.6710 1.2014 -3.2962

-0.6846 -0.4761 1.5784 0.9365 1.6483 -4.3591

-0.6377 -0.3960 1.3655 1.2279 2.0208 -5.4341

-0.5890 -0.3196 1.1555 1.5019 2.2442 -6.3246

-0.5358 -0.2449 0.9376 1.7664 2.3153 -7.0322

-0.4729 -0.1697 0.6957 2.0287 2.1946 -7.5157

-0.4036 -0.1037 0.4498 2.2509 1.8572 -7.6343

-0.3361 -0.0565 0.2317 2.3985 1.3873 -7.3808

-0.2734 -0.0273 0.0483 2.4744 0.8828 -6.8665

-0.2209 -0.0133 -0.0906 2.4919 0.4477 -6.2611

-0.1762 -0.0085 -0.1977 2.4729 0.0914 -5.6396

-0.1483 -0.0087 -0.2593 2.4449 -0.1169 -5.2076

-0.1207 -0.0111 -0.3159 2.4045 -0.3067 -4.7520

-0.0931 -0.0157 -0.3683 2.3508 -0.4779 -4.2675

-0.0643 -0.0226 -0.4181 2.2798 -0.6318 -3.7340

-0.0338 -0.0322 -0.4649 2.1870 -0.7614 -3.1419

-0.0019 -0.0444 -0.5072 2.0679 -0.8548 -2.4890

0.0312 -0.0590 -0.5426 1.9177 -0.8968 -1.7782

0.0648 -0.0753 -0.5682 1.7316 -0.8704 -1.0168

0.0980 -0.0922 -0.5807 1.5044 -0.7583 -0.2152

0.1290 -0.1071 -0.5762 1.2359 -0.5489 0.5981

0.1517 -0.1158 -0.5571 0.9841 -0.3005 1.2573

0.1675 -0.1190 -0.5296 0.7585 -0.0486 1.7766

0.1774 -0.1181 -0.5003 0.5742 0.1717 2.1561

0.1836 -0.1148 -0.4715 0.4218 0.3605 2.4412

0.1889 -0.1069 -0.4291 0.2292 0.6042 2.7651

0.1912 -0.0921 -0.3710 0.0073 0.8861 3.0870

0.1889 -0.0699 -0.3013 -0.2143 1.1605 3.3484

0.1819 -0.0414 -0.2247 -0.4145 1.3912 3.5214

0.1701 -0.0068 -0.1416 -0.5891 1.5635 3.6024

0.1579 0.0234 -0.0738 -0.7010 1.6446 3.5969

0.1465 0.0490 -0.0187 -0.7725 1.6709 3.5482

0.1305 0.0821 0.0506 -0.8372 1.6516 3.4317

0.1127 0.1156 0.1201 -0.8723 1.5711 3.2512

0.0883 0.1569 0.2063 -0.8697 1.3781 2.9289

0.0624 0.1948 0.2899 -0.8082 1.0791 2.4896

0.0409 0.2208 0.3538 -0.7074 0.7579 2.0335

0.0238 0.2367 0.4016 -0.5844 0.4454 1.5814

0.0110 0.2441 0.4352 -0.4532 0.1650 1.1531

0.0040 0.2454 0.4527 -0.3540 -0.0193 0.8507

-0.0013 0.2435 0.4650 -0.2526 -0.1854 0.5548

-0.0052 0.2383 0.4727 -0.1392 -0.3455 0.2354

-0.0069 0.2276 0.4737 0.0005 -0.5038 -0.1452

-0.0056 0.2156 0.4673 0.1138 -0.5974 -0.4460

-0.0024 0.2033 0.4561 0.2080 -0.6470 -0.6910

0.0006 0.1950 0.4464 0.2637 -0.6619 -0.8341

0.0042 0.1865 0.4349 0.3150 -0.6638 -0.9641

0.0094 0.1766 0.4192 0.3696 -0.6506 -1.1008

0.0172 0.1643 0.3964 0.4291 -0.6115 -1.2462

0.0279 0.1508 0.3656 0.4845 -0.5372 -1.3754

0.0418 0.1376 0.3266 0.5265 -0.4231 -1.4615

0.0562 0.1280 0.2873 0.5447 -0.2967 -1.4789

0.0700 0.1220 0.2500 0.5427 -0.1727 -1.4373

0.0830 0.1192 0.2163 0.5251 -0.0608 -1.3500

0.0943 0.1189 0.1875 0.4975 0.0318 -1.2335

0.1030 0.1201 0.1664 0.4683 0.0970 -1.1178

0.1140 0.1235 0.1410 0.4198 0.1703 -0.9325

0.1275 0.1309 0.1131 0.3385 0.2397 -0.6316

0.1374 0.1394 0.0971 0.2558 0.2688 -0.3347

0.1439 0.1475 0.0911 0.1799 0.2702 -0.0711

0.1471 0.1530 0.0914 0.1295 0.2609 0.0977

0.1492 0.1581 0.0950 0.0812 0.2455 0.2539

0.1506 0.1642 0.1040 0.0232 0.2201 0.4315

0.1503 0.1705 0.1200 -0.0369 0.1871 0.5989

0.1487 0.1752 0.1377 -0.0817 0.1590 0.7066

0.1448 0.1804 0.1655 -0.1291 0.1260 0.7910

0.1385 0.1851 0.1995 -0.1631 0.0988 0.8021

0.1321 0.1885 0.2289 -0.1753 0.0829 0.7442

0.1267 0.1909 0.2505 -0.1735 0.0738 0.6602

0.1218 0.1929 0.2682 -0.1631 0.0662 0.5573

0.1172 0.1948 0.2830 -0.1450 0.0578 0.4371

0.1129 0.1965 0.2951 -0.1179 0.0462 0.2956

0.1094 0.1978 0.3026 -0.0845 0.0305 0.1496

0.1076 0.1984 0.3051 -0.0576 0.0167 0.0476

0.1067 0.1986 0.3053 -0.0365 0.0050 -0.0252

0.1063 0.1986 0.3046 -0.0210 -0.0040 -0.0749

0.1061 0.1985 0.3032 -0.0058 -0.0131 -0.1207

0.1061 0.1982 0.3006 0.0120 -0.0240 -0.1706

0.1065 0.1977 0.2972 0.0294 -0.0349 -0.2151

0.1075 0.1967 0.2914 0.0507 -0.0482 -0.2633

0.1092 0.1952 0.2836 0.0716 -0.0607 -0.3024

0.1118 0.1930 0.2734 0.0906 -0.0705 -0.3267

0.1159 0.1900 0.2596 0.1054 -0.0737 -0.3259

0.1206 0.1870 0.2463 0.1089 -0.0656 -0.2934

0.1245 0.1848 0.2364 0.1035 -0.0509 -0.2465

0.1275 0.1835 0.2295 0.0936 -0.0343 -0.1970

0.1299 0.1828 0.2247 0.0817 -0.0178 -0.1488

0.1318 0.1825 0.2217 0.0695 -0.0030 -0.1049

0.1330 0.1826 0.2199 0.0585 0.0090 -0.0689

0.1339 0.1828 0.2190 0.0491 0.0184 -0.0397

0.1349 0.1833 0.2186 0.0364 0.0297 -0.0026

0.1356 0.1842 0.2190 0.0228 0.0402 0.0348

0.1360 0.1852 0.2203 0.0094 0.0487 0.0691

0.1360 0.1867 0.2226 -0.0046 0.0551 0.1022

0.1358 0.1880 0.2252 -0.0147 0.0577 0.1240

0.1352 0.1897 0.2292 -0.0256 0.0578 0.1448

0.1340 0.1919 0.2353 -0.0363 0.0527 0.1596

0.1326 0.1936 0.2409 -0.0415 0.0442 0.1601

0.1312 0.1949 0.2462 -0.0430 0.0341 0.1510

0.1298 0.1959 0.2508 -0.0415 0.0236 0.1346

0.1285 0.1964 0.2547 -0.0378 0.0135 0.1133

0.1275 0.1967 0.2578 -0.0325 0.0046 0.0891

0.1266 0.1967 0.2600 -0.0264 -0.0028 0.0642

0.1260 0.1966 0.2613 -0.0206 -0.0080 0.0424

0.1255 0.1962 0.2622 -0.0128 -0.0130 0.0152

0.1253 0.1959 0.2623 -0.0074 -0.0155 -0.0027

0.1252 0.1956 0.2621 -0.0022 -0.0171 -0.0191

0.1252 0.1951 0.2614 0.0034 -0.0179 -0.0356

0.1253 0.1947 0.2605 0.0076 -0.0180 -0.0472

0.1256 0.1942 0.2589 0.0128 -0.0172 -0.0602

0.1261 0.1936 0.2566 0.0174 -0.0153 -0.0696

0.1268 0.1931 0.2540 0.0205 -0.0126 -0.0730

0.1277 0.1927 0.2512 0.0218 -0.0095 -0.0704

0.1285 0.1924 0.2485 0.0214 -0.0060 -0.0617

0.1293 0.1922 0.2464 0.0193 -0.0028 -0.0490

0.1300 0.1922 0.2448 0.0162 -0.0000 -0.0346

0.1305 0.1922 0.2438 0.0126 0.0023 -0.0202

0.1309 0.1923 0.2434 0.0090 0.0042 -0.0071

0.1310 0.1924 0.2433 0.0064 0.0053 0.0013

0.1311 0.1925 0.2434 0.0039 0.0062 0.0090

0.1312 0.1927 0.2438 0.0011 0.0071 0.0168

0.1312 0.1929 0.2443 -0.0016 0.0078 0.0237

0.1311 0.1931 0.2449 -0.0035 0.0082 0.0280

0.1310 0.1933 0.2458 -0.0058 0.0084 0.0323

0.1307 0.1937 0.2473 -0.0081 0.0080 0.0348

0.1303 0.1941 0.2489 -0.0092 0.0069 0.0331

0.1299 0.1943 0.2503 -0.0091 0.0053 0.0286

0.1295 0.1945 0.2513 -0.0082 0.0036 0.0225

0.1293 0.1946 0.2519 -0.0069 0.0019 0.0163

0.1291 0.1946 0.2523 -0.0055 0.0004 0.0106

0.1290 0.1946 0.2525 -0.0041 -0.0007 0.0059

0.1289 0.1946 0.2526 -0.0030 -0.0016 0.0021

0.1288 0.1946 0.2526 -0.0019 -0.0023 -0.0014

0.1288 0.1945 0.2526 -0.0008 -0.0029 -0.0045

0.1288 0.1944 0.2524 0.0004 -0.0035 -0.0080

0.1288 0.1943 0.2522 0.0014 -0.0039 -0.0105

0.1289 0.1942 0.2519 0.0025 -0.0042 -0.0128

0.1290 0.1941 0.2514 0.0036 -0.0041 -0.0149

0.1292 0.1939 0.2507 0.0044 -0.0037 -0.0155

0.1293 0.1938 0.2502 0.0047 -0.0030 -0.0147

0.1295 0.1937 0.2497 0.0046 -0.0021 -0.0131

0.1296 0.1936 0.2493 0.0042 -0.0013 -0.0110

0.1298 0.1936 0.2490 0.0037 -0.0006 -0.0086

0.1299 0.1936 0.2488 0.0031 0.0001 -0.0063

0.1299 0.1936 0.2486 0.0025 0.0007 -0.0039

0.1300 0.1937 0.2486 0.0017 0.0012 -0.0013

0.1300 0.1937 0.2485 0.0012 0.0015 0.0004

0.1301 0.1937 0.2486 0.0006 0.0018 0.0019

0.1301 0.1938 0.2486 0.0001 0.0019 0.0036

0.1301 0.1938 0.2488 -0.0006 0.0020 0.0051

0.1300 0.1939 0.2490 -0.0011 0.0020 0.0064

0.1300 0.1940 0.2492 -0.0016 0.0019 0.0072

0.1299 0.1940 0.2495 -0.0019 0.0016 0.0073

0.1298 0.1941 0.2498 -0.0020 0.0012 0.0068

0.1298 0.1941 0.2500 -0.0019 0.0008 0.0059

0.1297 0.1942 0.2502 -0.0017 0.0004 0.0046

0.1296 0.1942 0.2504 -0.0014 0.0001 0.0032

0.1296 0.1942 0.2505 -0.0010 -0.0002 0.0017

0.1296 0.1941 0.2505 -0.0007 -0.0004 0.0005

0.1295 0.1941 0.2505 -0.0003 -0.0006 -0.0005

0.1295 0.1941 0.2505 -0.0001 -0.0007 -0.0014

0.1295 0.1941 0.2504 0.0003 -0.0008 -0.0022

0.1296 0.1941 0.2503 0.0006 -0.0008 -0.0029

0.1296 0.1940 0.2502 0.0008 -0.0007 -0.0033

0.1296 0.1940 0.2500 0.0010 -0.0006 -0.0033

0.1297 0.1940 0.2499 0.0010 -0.0004 -0.0029

0.1297 0.1940 0.2498 0.0009 -0.0003 -0.0023

0.1297 0.1939 0.2497 0.0007 -0.0001 -0.0016

0.1298 0.1939 0.2497 0.0006 0.0001 -0.0010

0.1298 0.1939 0.2496 0.0004 0.0002 -0.0004

0.1298 0.1940 0.2496 0.0002 0.0003 0.0001

0.1298 0.1940 0.2496 0.0001 0.0003 0.0006

0.1298 0.1940 0.2497 -0.0001 0.0004 0.0011

0.1298 0.1940 0.2497 -0.0003 0.0004 0.0014

0.1298 0.1940 0.2498 -0.0004 0.0004 0.0016

0.1298 0.1940 0.2499 -0.0004 0.0003 0.0015

0.1297 0.1940 0.2499 -0.0004 0.0002 0.0013

0.1297 0.1941 0.2500 -0.0004 0.0001 0.0010

0.1297 0.1941 0.2500 -0.0003 0.0000 0.0007

0.1297 0.1941 0.2500 -0.0002 -0.0000 0.0004

0.1297 0.1941 0.2500 -0.0001 -0.0001 0.0001

0.1297 0.1940 0.2500 -0.0001 -0.0001 -0.0002

0.1297 0.1940 0.2500 0.0000 -0.0002 -0.0004

0.1297 0.1940 0.2500 0.0001 -0.0002 -0.0007

0.1297 0.1940 0.2500 0.0002 -0.0002 -0.0007

0.1297 0.1940 0.2499 0.0002 -0.0001 -0.0007

0.1297 0.1940 0.2499 0.0002 -0.0001 -0.0006

0.1297 0.1940 0.2499 0.0002 -0.0000 -0.0004

0.1297 0.1940 0.2499 0.0001 0.0000 -0.0002

0.1297 0.1940 0.2499 0.0001 0.0000 -0.0001

0.1297 0.1940 0.2499 0.0000 0.0001 0.0001

0.1297 0.1940 0.2499 -0.0000 0.0001 0.0003

0.1297 0.1940 0.2499 -0.0001 0.0001 0.0003

0.1297 0.1940 0.2499 -0.0001 0.0001 0.0003

0.1297 0.1940 0.2499 -0.0001 0.0000 0.0003

0.1297 0.1940 0.2499 -0.0001 0.0000 0.0002

0.1297 0.1940 0.2499 -0.0000 -0.0000 0.0001

0.1297 0.1940 0.2499 -0.0000 -0.0000 -0.0000

0.1297 0.1940 0.2499 0.0000 -0.0000 -0.0001

0.1297 0.1940 0.2499 0.0000 -0.0000 -0.0002

0.1297 0.1940 0.2499 0.0000 -0.0000 -0.0001

0.1297 0.1940 0.2499 0.0000 -0.0000 -0.0001

0.1297 0.1940 0.2499 0.0000 0.0000 -0.0000

0.1297 0.1940 0.2499 0.0000 0.0000 0.0000

0.1297 0.1940 0.2499 -0.0000 0.0000 0.0000];

xmm =[0.1297 0.1940 0.2499 -0.0000 -0.0001 -0.0001

0.1297 0.1940 0.2499 -0.0001 -0.0001 -0.0002

0.1297 0.1940 0.2499 -0.0001 -0.0001 -0.0002

0.1297 0.1940 0.2499 -0.0001 -0.0001 -0.0002

0.1297 0.1940 0.2498 -0.0002 -0.0002 -0.0001

0.1296 0.1939 0.2498 -0.0002 -0.0002 -0.0001

0.1296 0.1939 0.2498 -0.0002 -0.0002 -0.0002

0.1296 0.1939 0.2498 -0.0002 -0.0002 -0.0002

0.1296 0.1939 0.2498 -0.0002 -0.0002 -0.0002

0.1295 0.1939 0.2497 -0.0002 -0.0002 -0.0003];

xff =[0.1297 0.1940 0.2499 0 0 0

0.1297 0.1940 0.2499 -0.0000 0.0000 0.0001

0.1297 0.1940 0.2499 -0.0001 0.0001 0.0005

0.1297 0.1940 0.2499 -0.0008 0.0007 0.0025

0.1297 0.1940 0.2499 -0.0038 0.0034 0.0124

0.1296 0.1941 0.2503 -0.0189 0.0165 0.0608

0.1289 0.1947 0.2524 -0.0483 0.0398 0.1504

0.1269 0.1963 0.2585 -0.0900 0.0677 0.2693

0.1218 0.1998 0.2734 -0.1474 0.0927 0.4185

0.1142 0.2040 0.2943 -0.1994 0.0980 0.5423

0.1045 0.2080 0.3202 -0.2449 0.0843 0.6440

0.0940 0.2109 0.3476 -0.2812 0.0567 0.7218

0.0834 0.2123 0.3746 -0.3090 0.0218 0.7804

0.0763 0.2125 0.3924 -0.3243 -0.0037 0.8123

0.0689 0.2121 0.4108 -0.3380 -0.0316 0.8411

0.0602 0.2109 0.4325 -0.3518 -0.0656 0.8705

0.0484 0.2080 0.4615 -0.3672 -0.1115 0.9038

0.0334 0.2024 0.4983 -0.3827 -0.1690 0.9382

0.0147 0.1926 0.5442 -0.3973 -0.2370 0.9723

-0.0090 0.1765 0.6020 -0.4107 -0.3132 1.0049

-0.0399 0.1501 0.6778 -0.4234 -0.3929 1.0356

-0.0819 0.1080 0.7803 -0.4371 -0.4631 1.0632

-0.1232 0.0632 0.8800 -0.4503 -0.4963 1.0802

-0.1641 0.0181 0.9774 -0.4641 -0.5055 1.0910

-0.2072 -0.0278 1.0773 -0.4790 -0.5022 1.0983

-0.2561 -0.0778 1.1879 -0.4950 -0.4947 1.1035

-0.3198 -0.1399 1.3275 -0.5125 -0.4912 1.1075

-0.3956 -0.2120 1.4892 -0.5270 -0.5001 1.1099

-0.4665 -0.2796 1.6372 -0.5351 -0.5153 1.1111

-0.5449 -0.3561 1.7993 -0.5403 -0.5318 1.1118

-0.6479 -0.4586 2.0105 -0.5445 -0.5447 1.1122

-0.7005 -0.5113 2.1178 -0.5462 -0.5478 1.1123];

xcc=[-0.7005 -0.5113 2.1178 -0.5462 -0.5478 1.1123

-0.7036 -0.5145 2.1240 -0.5224 -0.5577 1.0255

-0.7165 -0.5305 2.1471 -0.4177 -0.6056 0.6710

-0.7275 -0.5496 2.1626 -0.3150 -0.6514 0.3628

-0.7353 -0.5687 2.1696 -0.2298 -0.6773 0.1326

-0.7392 -0.5816 2.1708 -0.1792 -0.6820 0.0041

-0.7422 -0.5945 2.1698 -0.1331 -0.6751 -0.1092

-0.7442 -0.6068 2.1669 -0.0928 -0.6571 -0.2072

-0.7458 -0.6219 2.1606 -0.0461 -0.6174 -0.3218

-0.7464 -0.6387 2.1495 0.0038 -0.5448 -0.4499

-0.7459 -0.6514 2.1369 0.0416 -0.4634 -0.5551

-0.7445 -0.6611 2.1229 0.0735 -0.3748 -0.6523

-0.7421 -0.6697 2.1036 0.1084 -0.2583 -0.7699

-0.7382 -0.6755 2.0781 0.1462 -0.1178 -0.9101

-0.7330 -0.6768 2.0472 0.1868 0.0330 -1.0708

-0.7277 -0.6743 2.0177 0.2235 0.1580 -1.2194

-0.7199 -0.6670 1.9763 0.2746 0.3068 -1.4241

-0.7066 -0.6500 1.9096 0.3593 0.4969 -1.7502

-0.6863 -0.6210 1.8142 0.4857 0.6975 -2.2079

-0.6515 -0.5726 1.6623 0.6939 0.9223 -2.9102

-0.6019 -0.5108 1.4623 0.9648 1.1327 -3.7617

-0.5471 -0.4500 1.2548 1.2280 1.2948 -4.5283

-0.4787 -0.3817 1.0096 1.5014 1.4280 -5.2381

-0.4050 -0.3149 0.7597 1.7253 1.4979 -5.6950

-0.3239 -0.2476 0.5004 1.8841 1.4943 -5.8255

-0.2330 -0.1791 0.2314 1.9473 1.4031 -5.5093

-0.1567 -0.1262 0.0254 1.9020 1.2709 -4.8751

-0.0974 -0.0876 -0.1192 1.8001 1.1444 -4.1551

-0.0480 -0.0567 -0.2270 1.6659 1.0303 -3.4030

0.0111 -0.0202 -0.3357 1.4354 0.8919 -2.3113

0.0562 0.0085 -0.3977 1.1927 0.7895 -1.3123

0.0870 0.0299 -0.4235 0.9784 0.7224 -0.5206

0.1143 0.0520 -0.4267 0.7320 0.6623 0.3042

0.1310 0.0690 -0.4103 0.5307 0.6208 0.9161

0.1454 0.0904 -0.3641 0.2719 0.5689 1.6212

0.1510 0.1111 -0.2903 0.0273 0.5099 2.1939

0.1487 0.1274 -0.2101 -0.1529 0.4486 2.5373

0.1416 0.1407 -0.1255 -0.2882 0.3803 2.7225

0.1299 0.1522 -0.0316 -0.3906 0.2972 2.7742

0.1206 0.1582 0.0305 -0.4346 0.2378 2.7320

0.1105 0.1629 0.0910 -0.4603 0.1762 2.6348

0.0972 0.1668 0.1634 -0.4680 0.0977 2.4428

0.0834 0.1685 0.2325 -0.4493 0.0181 2.1736

0.0748 0.1684 0.2734 -0.4241 -0.0310 1.9667

0.0668 0.1673 0.3100 -0.3902 -0.0761 1.7428

0.0577 0.1648 0.3494 -0.3378 -0.1254 1.4463

0.0483 0.1600 0.3887 -0.2596 -0.1741 1.0588

0.0409 0.1533 0.4184 -0.1633 -0.2077 0.6305

0.0370 0.1461 0.4330 -0.0694 -0.2176 0.2469

0.0362 0.1395 0.4356 0.0130 -0.2081 -0.0689

0.0373 0.1345 0.4310 0.0751 -0.1886 -0.2949

0.0402 0.1298 0.4198 0.1371 -0.1562 -0.5099

0.0458 0.1254 0.3994 0.1994 -0.1060 -0.7131

0.0524 0.1230 0.3762 0.2431 -0.0538 -0.8438

0.0591 0.1223 0.3533 0.2713 -0.0059 -0.9180

0.0650 0.1225 0.3336 0.2872 0.0330 -0.9514

0.0704 0.1235 0.3159 0.2961 0.0659 -0.9619

0.0776 0.1255 0.2927 0.3009 0.1058 -0.9507

0.0866 0.1294 0.2647 0.2969 0.1491 -0.9010

0.0977 0.1360 0.2322 0.2777 0.1903 -0.7898

0.1089 0.1448 0.2018 0.2416 0.2155 -0.6187

0.1179 0.1537 0.1804 0.1975 0.2190 -0.4295

0.1246 0.1620 0.1675 0.1511 0.2061 -0.2430

0.1292 0.1690 0.1618 0.1070 0.1828 -0.0752

0.1314 0.1731 0.1612 0.0787 0.1631 0.0278

0.1330 0.1767 0.1630 0.0516 0.1410 0.1220

0.1340 0.1805 0.1682 0.0206 0.1118 0.2244

0.1342 0.1839 0.1784 -0.0127 0.0755 0.3246

0.1334 0.1858 0.1899 -0.0367 0.0453 0.3872

0.1316 0.1869 0.2049 -0.0573 0.0153 0.4283

0.1292 0.1870 0.2213 -0.0710 -0.0095 0.4395

0.1260 0.1861 0.2397 -0.0779 -0.0297 0.4183

0.1228 0.1846 0.2560 -0.0768 -0.0411 0.3702

0.1200 0.1830 0.2687 -0.0703 -0.0452 0.3101

0.1177 0.1814 0.2786 -0.0602 -0.0448 0.2420

0.1158 0.1799 0.2859 -0.0475 -0.0411 0.1705

0.1143 0.1785 0.2908 -0.0329 -0.0349 0.0981

0.1134 0.1773 0.2931 -0.0174 -0.0272 0.0287

0.1131 0.1767 0.2933 -0.0069 -0.0216 -0.0145

0.1130 0.1762 0.2924 0.0031 -0.0158 -0.0529

0.1132 0.1759 0.2909 0.0114 -0.0108 -0.0826

0.1136 0.1757 0.2880 0.0208 -0.0049 -0.1132

0.1146 0.1757 0.2831 0.0311 0.0023 -0.1423

0.1160 0.1759 0.2771 0.0386 0.0087 -0.1579

0.1182 0.1765 0.2686 0.0435 0.0153 -0.1577

0.1202 0.1773 0.2617 0.0433 0.0192 -0.1416

0.1221 0.1782 0.2561 0.0398 0.0215 -0.1161

0.1236 0.1791 0.2518 0.0340 0.0224 -0.0859

0.1249 0.1800 0.2490 0.0270 0.0221 -0.0546

0.1257 0.1809 0.2475 0.0196 0.0209 -0.0253

0.1263 0.1816 0.2471 0.0127 0.0191 -0.0001

0.1266 0.1821 0.2473 0.0071 0.0171 0.0188

0.1267 0.1826 0.2482 0.0018 0.0148 0.0353

0.1267 0.1830 0.2495 -0.0031 0.0120 0.0491

0.1266 0.1833 0.2509 -0.0067 0.0095 0.0581

0.1262 0.1836 0.2535 -0.0109 0.0055 0.0665

0.1257 0.1837 0.2561 -0.0134 0.0019 0.0690

0.1254 0.1838 0.2578 -0.0143 -0.0002 0.0681

0.1250 0.1837 0.2595 -0.0147 -0.0022 0.0654

0.1246 0.1837 0.2612 -0.0145 -0.0040 0.0607

0.1241 0.1835 0.2632 -0.0136 -0.0060 0.0521

0.1236 0.1832 0.2652 -0.0115 -0.0076 0.0396

0.1232 0.1828 0.2666 -0.0088 -0.0082 0.0264

0.1229 0.1825 0.2673 -0.0059 -0.0080 0.0138

0.1227 0.1822 0.2676 -0.0032 -0.0072 0.0028

0.1227 0.1820 0.2676 -0.0009 -0.0061 -0.0058

0.1227 0.1819 0.2674 0.0009 -0.0050 -0.0120

0.1227 0.1818 0.2670 0.0022 -0.0040 -0.0164

0.1228 0.1817 0.2664 0.0038 -0.0025 -0.0214

0.1229 0.1816 0.2656 0.0051 -0.0010 -0.0247

0.1231 0.1816 0.2649 0.0059 0.0002 -0.0263

0.1232 0.1816 0.2642 0.0063 0.0012 -0.0267

0.1234 0.1817 0.2635 0.0066 0.0021 -0.0262

0.1237 0.1818 0.2625 0.0065 0.0031 -0.0242

0.1240 0.1819 0.2616 0.0059 0.0038 -0.0202

0.1242 0.1821 0.2608 0.0050 0.0041 -0.0153

0.1244 0.1822 0.2603 0.0039 0.0040 -0.0101

0.1245 0.1824 0.2600 0.0027 0.0036 -0.0052

0.1246 0.1825 0.2599 0.0016 0.0031 -0.0009

0.1246 0.1826 0.2599 0.0007 0.0025 0.0024

0.1246 0.1827 0.2600 -0.0000 0.0020 0.0050

0.1246 0.1827 0.2602 -0.0008 0.0014 0.0073

0.1246 0.1828 0.2605 -0.0015 0.0007 0.0092

0.1245 0.1828 0.2609 -0.0021 0.0001 0.0104

0.1244 0.1828 0.2613 -0.0024 -0.0005 0.0109

0.1243 0.1827 0.2617 -0.0026 -0.0010 0.0104

0.1242 0.1827 0.2622 -0.0026 -0.0013 0.0090

0.1241 0.1826 0.2625 -0.0023 -0.0015 0.0072

0.1240 0.1825 0.2628 -0.0020 -0.0015 0.0051

0.1239 0.1825 0.2630 -0.0015 -0.0015 0.0030

0.1239 0.1824 0.2630 -0.0011 -0.0013 0.0011

0.1238 0.1824 0.2630 -0.0008 -0.0012 -0.0002

0.1238 0.1824 0.2630 -0.0005 -0.0011 -0.0013

0.1238 0.1823 0.2630 -0.0001 -0.0009 -0.0024

0.1238 0.1823 0.2628 0.0002 -0.0006 -0.0036

0.1238 0.1823 0.2627 0.0005 -0.0004 -0.0043

0.1239 0.1823 0.2624 0.0007 -0.0001 -0.0046

0.1239 0.1823 0.2622 0.0008 0.0001 -0.0043

0.1239 0.1823 0.2620 0.0007 0.0002 -0.0037

0.1240 0.1823 0.2619 0.0006 0.0003 -0.0030

0.1240 0.1823 0.2618 0.0004 0.0003 -0.0021

0.1240 0.1823 0.2617 0.0002 0.0003 -0.0013

0.1240 0.1823 0.2617 0.0000 0.0002 -0.0005

0.1240 0.1823 0.2617 -0.0001 0.0002 0.0001

0.1240 0.1823 0.2617 -0.0003 0.0001 0.0007

0.1240 0.1823 0.2617 -0.0005 -0.0001 0.0011

0.1240 0.1823 0.2618 -0.0006 -0.0002 0.0015

0.1239 0.1823 0.2619 -0.0007 -0.0003 0.0015

0.1239 0.1823 0.2619 -0.0007 -0.0004 0.0013

0.1239 0.1823 0.2620 -0.0006 -0.0005 0.0011

0.1238 0.1823 0.2620 -0.0006 -0.0005 0.0008

0.1238 0.1822 0.2620 -0.0005 -0.0005 0.0005

0.1238 0.1822 0.2621 -0.0005 -0.0005 0.0002

0.1238 0.1822 0.2621 -0.0004 -0.0005 -0.0002

0.1238 0.1822 0.2620 -0.0003 -0.0004 -0.0005

0.1238 0.1822 0.2620 -0.0002 -0.0004 -0.0007

0.1237 0.1821 0.2619 -0.0001 -0.0003 -0.0010

0.1237 0.1821 0.2619 -0.0001 -0.0002 -0.0010

0.1237 0.1821 0.2618 -0.0001 -0.0002 -0.0009

0.1237 0.1821 0.2618 -0.0001 -0.0002 -0.0008

0.1237 0.1821 0.2618 -0.0002 -0.0002 -0.0006

0.1237 0.1821 0.2617 -0.0002 -0.0002 -0.0004

0.1237 0.1821 0.2617 -0.0002 -0.0002 -0.0003

0.1237 0.1821 0.2617 -0.0003 -0.0002 -0.0001

0.1237 0.1820 0.2617 -0.0003 -0.0003 -0.0000

0.1236 0.1820 0.2617 -0.0003 -0.0003 0.0000

0.1236 0.1820 0.2617 -0.0003 -0.0003 -0.0000

0.1236 0.1820 0.2617 -0.0003 -0.0003 -0.0001

0.1236 0.1819 0.2617 -0.0003 -0.0003 -0.0002

0.1235 0.1819 0.2617 -0.0003 -0.0003 -0.0003

0.1235 0.1819 0.2616 -0.0003 -0.0003 -0.0004

0.1235 0.1819 0.2616 -0.0003 -0.0003 -0.0004

0.1235 0.1819 0.2616 -0.0003 -0.0003 -0.0004

0.1235 0.1818 0.2615 -0.0003 -0.0003 -0.0003

0.1234 0.1818 0.2615 -0.0003 -0.0003 -0.0003

0.1234 0.1818 0.2615 -0.0003 -0.0003 -0.0002

0.1234 0.1817 0.2615 -0.0003 -0.0003 -0.0002

0.1233 0.1817 0.2614 -0.0003 -0.0003 -0.0002

0.1233 0.1817 0.2614 -0.0003 -0.0003 -0.0003];

xmmm =[0.1464 0.2024 0.2248 0 0 0

0.1464 0.2024 0.2248 0.0000 0.0000 -0.0000

0.1464 0.2024 0.2248 0.0000 0.0000 -0.0000

0.1464 0.2024 0.2248 0.0000 0.0000 -0.0000

0.1464 0.2024 0.2248 0.0000 0.0000 0.0000

0.1464 0.2024 0.2248 0.0000 -0.0000 0.0000

0.1464 0.2024 0.2248 0.0000 -0.0000 0.0000

0.1464 0.2024 0.2248 0.0000 0.0000 0.0000

0.1464 0.2024 0.2248 0.0000 0.0000 -0.0000

0.1464 0.2024 0.2248 0.0000 0.0000 -0.0000

0.1464 0.2024 0.2248 0.0000 0.0000 -0.0000];

xfff =[0.1464 0.2024 0.2248 0 0 0

0.1464 0.2024 0.2248 -0.0000 0.0000 0.0001

0.1464 0.2024 0.2248 -0.0001 0.0001 0.0005

0.1464 0.2024 0.2248 -0.0008 0.0007 0.0025

0.1464 0.2024 0.2248 -0.0038 0.0035 0.0124

0.1463 0.2025 0.2252 -0.0190 0.0171 0.0608

0.1457 0.2031 0.2271 -0.0470 0.0403 0.1459

0.1439 0.2046 0.2326 -0.0862 0.0675 0.2573

0.1395 0.2077 0.2453 -0.1384 0.0922 0.3947

0.1305 0.2129 0.2704 -0.2018 0.0978 0.5477

0.1209 0.2168 0.2961 -0.2457 0.0797 0.6474

0.1110 0.2192 0.3218 -0.2785 0.0494 0.7201

0.1013 0.2203 0.3466 -0.3033 0.0134 0.7749

0.0948 0.2203 0.3633 -0.3172 -0.0132 0.8057

0.0879 0.2197 0.3807 -0.3297 -0.0419 0.8337

0.0790 0.2181 0.4031 -0.3434 -0.0800 0.8651

0.0674 0.2146 0.4323 -0.3581 -0.1299 0.8996

0.0528 0.2082 0.4690 -0.3724 -0.1908 0.9349

0.0346 0.1973 0.5148 -0.3859 -0.2612 0.9696

0.0113 0.1795 0.5732 -0.3985 -0.3382 1.0033

-0.0212 0.1489 0.6553 -0.4121 -0.4185 1.0367

-0.0600 0.1073 0.7524 -0.4268 -0.4730 1.0628

-0.0975 0.0654 0.8449 -0.4417 -0.4932 1.0790

-0.1344 0.0248 0.9341 -0.4571 -0.4943 1.0895

-0.1744 -0.0174 1.0281 -0.4738 -0.4871 1.0970

-0.2217 -0.0646 1.1358 -0.4917 -0.4793 1.1026

-0.2774 -0.1178 1.2586 -0.5085 -0.4795 1.1066

-0.3450 -0.1813 1.4037 -0.5224 -0.4928 1.1094

-0.4077 -0.2410 1.5359 -0.5302 -0.5106 1.1108

-0.4797 -0.3114 1.6861 -0.5355 -0.5290 1.1117

-0.5877 -0.4194 1.9093 -0.5411 -0.5427 1.1123

-0.6816 -0.5136 2.1016 -0.5453 -0.5461 1.1125

-0.7868 -0.6186 2.3155 -0.5491 -0.5475 1.1126];

xccc =[-0.7868 -0.6186 2.3155 -0.5491 -0.5475 1.1126

-0.7934 -0.6253 2.3285 -0.5267 -0.5586 1.0328

-0.8070 -0.6411 2.3547 -0.4839 -0.6077 0.9152

-0.8213 -0.6610 2.3820 -0.4473 -0.6923 0.8684

-0.8373 -0.6888 2.4146 -0.4220 -0.8172 0.9176

-0.8586 -0.7353 2.4666 -0.4255 -1.0101 1.1575

-0.8804 -0.7894 2.5325 -0.4793 -1.2133 1.5804

-0.9061 -0.8536 2.6234 -0.5921 -1.4460 2.2133

-0.9389 -0.9304 2.7510 -0.7759 -1.7337 3.0973

-0.9895 -1.0365 2.9564 -1.0845 -2.1473 4.4423

-1.0341 -1.1209 3.1388 -1.3539 -2.4590 5.5276

-1.0894 -1.2165 3.3627 -1.6702 -2.7454 6.6914

-1.1494 -1.3098 3.5996 -1.9812 -2.9035 7.6840

-1.2144 -1.3986 3.8461 -2.2726 -2.8836 8.4144

-1.2928 -1.4884 4.1265 -2.5551 -2.6127 8.8194

-1.3663 -1.5547 4.3705 -2.7482 -2.1612 8.7599

-1.4442 -1.6062 4.6071 -2.8763 -1.5441 8.2931

-1.5111 -1.6350 4.7908 -2.9235 -0.9616 7.6180

-1.5685 -1.6489 4.9331 -2.9180 -0.4640 6.8766

-1.6181 -1.6533 5.0446 -2.8789 -0.0587 6.1364

-1.6588 -1.6520 5.1274 -2.8225 0.2427 5.4730

-1.6938 -1.6475 5.1923 -2.7561 0.4718 4.8683

-1.7410 -1.6369 5.2698 -2.6386 0.7260 4.0095

-1.8059 -1.6150 5.3564 -2.4193 0.9481 2.7579

-1.8598 -1.5920 5.4080 -2.1768 0.9943 1.6572

-1.9077 -1.5695 5.4346 -1.9026 0.9028 0.6281

-1.9473 -1.5512 5.4385 -1.6181 0.7070 -0.2741

-1.9781 -1.5391 5.4247 -1.3409 0.4518 -1.0318

-2.0013 -1.5330 5.3989 -1.0807 0.1738 -1.6559

-2.0137 -1.5321 5.3760 -0.9110 -0.0213 -2.0237

-2.0239 -1.5336 5.3488 -0.7420 -0.2228 -2.3619

-2.0328 -1.5382 5.3140 -0.5578 -0.4477 -2.7006

-2.0406 -1.5486 5.2629 -0.3265 -0.7333 -3.0838

-2.0447 -1.5679 5.1925 -0.0586 -1.0606 -3.4692

-2.0426 -1.5979 5.1036 0.2216 -1.3864 -3.8005

-2.0352 -1.6314 5.0170 0.4468 -1.6229 -4.0031

-2.0183 -1.6824 4.8972 0.6938 -1.8288 -4.1296

-1.9869 -1.7548 4.7389 0.9139 -1.8903 -4.0593

-1.9554 -1.8148 4.6101 1.0027 -1.7605 -3.8066

-1.9224 -1.8682 4.4921 1.0033 -1.4880 -3.4013

-1.8924 -1.9088 4.3945 0.9316 -1.1365 -2.9059

-1.8676 -1.9358 4.3196 0.8152 -0.7719 -2.3834

-1.8480 -1.9515 4.2637 0.6743 -0.4263 -1.8610

-1.8334 -1.9581 4.2245 0.5249 -0.1234 -1.3645

-1.8259 -1.9586 4.2055 0.4216 0.0572 -1.0416

-1.8200 -1.9564 4.1915 0.3161 0.2203 -0.7235

-1.8150 -1.9500 4.1808 0.1794 0.4009 -0.3245

-1.8125 -1.9370 4.1787 0.0097 0.5757 0.1544

-1.8139 -1.9228 4.1865 -0.1271 0.6732 0.5296

-1.8181 -1.9071 4.2021 -0.2493 0.7215 0.8574

-1.8281 -1.8847 4.2347 -0.3924 0.7163 1.2308

-1.8416 -1.8641 4.2759 -0.4992 0.6445 1.4968

-1.8569 -1.8473 4.3210 -0.5690 0.5323 1.6548

-1.8730 -1.8345 4.3673 -0.6075 0.4000 1.7206

-1.8892 -1.8258 4.4126 -0.6197 0.2634 1.7074

-1.9048 -1.8208 4.4550 -0.6103 0.1342 1.6293

-1.9193 -1.8190 4.4929 -0.5845 0.0212 1.5022

-1.9318 -1.8195 4.5243 -0.5489 -0.0686 1.3487

-1.9430 -1.8218 4.5511 -0.5056 -0.1410 1.1729

-1.9596 -1.8285 4.5872 -0.4172 -0.2287 0.8322

-1.9722 -1.8372 4.6096 -0.3228 -0.2727 0.4846

-1.9809 -1.8460 4.6197 -0.2327 -0.2846 0.1666

-1.9851 -1.8519 4.6211 -0.1734 -0.2805 -0.0342

-1.9881 -1.8576 4.6184 -0.1159 -0.2692 -0.2214

-1.9901 -1.8638 4.6107 -0.0533 -0.2501 -0.4152

-1.9906 -1.8712 4.5941 0.0214 -0.2197 -0.6275

-1.9892 -1.8769 4.5749 0.0775 -0.1923 -0.7676

-1.9854 -1.8830 4.5457 0.1353 -0.1603 -0.8815

-1.9789 -1.8889 4.5088 0.1808 -0.1310 -0.9232

-1.9711 -1.8937 4.4721 0.2034 -0.1097 -0.8780

-1.9640 -1.8973 4.4432 0.2053 -0.0955 -0.7822

-1.9575 -1.9002 4.4199 0.1939 -0.0832 -0.6572

-1.9514 -1.9027 4.4005 0.1711 -0.0693 -0.5058

-1.9458 -1.9049 4.3855 0.1363 -0.0509 -0.3288

-1.9418 -1.9063 4.3771 0.0966 -0.0295 -0.1610

-1.9399 -1.9068 4.3747 0.0677 -0.0131 -0.0534

-1.9389 -1.9069 4.3744 0.0459 -0.0004 0.0210

-1.9383 -1.9068 4.3753 0.0241 0.0125 0.0899

-1.9381 -1.9065 4.3774 0.0036 0.0250 0.1507

-1.9383 -1.9058 4.3812 -0.0209 0.0399 0.2169

-1.9389 -1.9049 4.3863 -0.0435 0.0536 0.2720

-1.9405 -1.9032 4.3946 -0.0701 0.0690 0.3278

-1.9432 -1.9007 4.4058 -0.0949 0.0815 0.3675

-1.9475 -1.8973 4.4211 -0.1158 0.0873 0.3810

-1.9529 -1.8935 4.4376 -0.1246 0.0800 0.3545

-1.9577 -1.8907 4.4504 -0.1209 0.0636 0.3042

-1.9614 -1.8889 4.4593 -0.1109 0.0445 0.2482

-1.9644 -1.8879 4.4656 -0.0978 0.0252 0.1918

-1.9667 -1.8875 4.4698 -0.0837 0.0074 0.1391

-1.9679 -1.8875 4.4717 -0.0746 -0.0029 0.1079

-1.9689 -1.8876 4.4731 -0.0651 -0.0128 0.0769

-1.9700 -1.8879 4.4741 -0.0531 -0.0243 0.0397

-1.9711 -1.8887 4.4745 -0.0373 -0.0377 -0.0065

-1.9718 -1.8897 4.4739 -0.0224 -0.0483 -0.0473

-1.9721 -1.8911 4.4721 -0.0063 -0.0574 -0.0884

-1.9721 -1.8927 4.4692 0.0089 -0.0632 -0.1240

-1.9718 -1.8941 4.4662 0.0196 -0.0652 -0.1470

-1.9710 -1.8962 4.4612 0.0325 -0.0641 -0.1709

-1.9695 -1.8985 4.4543 0.0433 -0.0573 -0.1845

-1.9679 -1.9004 4.4478 0.0486 -0.0474 -0.1831

-1.9662 -1.9018 4.4418 0.0496 -0.0358 -0.1708

-1.9646 -1.9027 4.4366 0.0474 -0.0240 -0.1506

-1.9632 -1.9033 4.4323 0.0427 -0.0129 -0.1248

-1.9620 -1.9035 4.4289 0.0363 -0.0031 -0.0961

-1.9610 -1.9035 4.4265 0.0291 0.0047 -0.0672

-1.9604 -1.9033 4.4251 0.0218 0.0105 -0.0405

-1.9598 -1.9029 4.4243 0.0131 0.0156 -0.0103

-1.9596 -1.9025 4.4243 0.0072 0.0180 0.0090

-1.9595 -1.9021 4.4247 0.0015 0.0195 0.0266

-1.9596 -1.9017 4.4256 -0.0046 0.0204 0.0443

-1.9597 -1.9012 4.4267 -0.0097 0.0204 0.0580

-1.9601 -1.9006 4.4288 -0.0157 0.0194 0.0724

-1.9608 -1.8999 4.4317 -0.0209 0.0171 0.0819

-1.9617 -1.8993 4.4348 -0.0240 0.0140 0.0841

-1.9626 -1.8988 4.4381 -0.0251 0.0103 0.0789

-1.9637 -1.8985 4.4411 -0.0239 0.0063 0.0669

-1.9645 -1.8983 4.4434 -0.0212 0.0027 0.0514

-1.9652 -1.8983 4.4450 -0.0174 -0.0005 0.0346

-1.9658 -1.8983 4.4459 -0.0131 -0.0031 0.0184

-1.9661 -1.8985 4.4462 -0.0090 -0.0053 0.0040

-1.9663 -1.8986 4.4462 -0.0053 -0.0068 -0.0074

-1.9664 -1.8988 4.4459 -0.0025 -0.0078 -0.0157

-1.9664 -1.8990 4.4453 0.0008 -0.0088 -0.0245

-1.9664 -1.8992 4.4447 0.0033 -0.0094 -0.0304

-1.9662 -1.8995 4.4438 0.0059 -0.0097 -0.0356

-1.9660 -1.8999 4.4423 0.0086 -0.0096 -0.0394

-1.9654 -1.9004 4.4402 0.0105 -0.0083 -0.0389

-1.9650 -1.9007 4.4387 0.0107 -0.0066 -0.0344

-1.9646 -1.9009 4.4374 0.0098 -0.0046 -0.0278

-1.9643 -1.9010 4.4366 0.0084 -0.0026 -0.0207

-1.9640 -1.9011 4.4360 0.0068 -0.0008 -0.0140

-1.9639 -1.9011 4.4358 0.0056 0.0002 -0.0096

-1.9638 -1.9011 4.4356 0.0044 0.0012 -0.0053

-1.9637 -1.9010 4.4355 0.0030 0.0023 -0.0008

-1.9636 -1.9010 4.4356 0.0015 0.0032 0.0038

-1.9636 -1.9009 4.4357 0.0001 0.0039 0.0077

-1.9636 -1.9008 4.4359 -0.0012 0.0044 0.0110

-1.9637 -1.9007 4.4362 -0.0023 0.0047 0.0136

-1.9638 -1.9005 4.4367 -0.0036 0.0048 0.0162

-1.9640 -1.9003 4.4375 -0.0048 0.0044 0.0178

-1.9642 -1.9002 4.4382 -0.0053 0.0037 0.0175

-1.9644 -1.9000 4.4388 -0.0053 0.0028 0.0159

-1.9645 -1.9000 4.4393 -0.0050 0.0018 0.0136

-1.9647 -1.8999 4.4397 -0.0045 0.0009 0.0109

-1.9648 -1.8999 4.4399 -0.0038 0.0001 0.0081

-1.9649 -1.8999 4.4401 -0.0031 -0.0005 0.0055

-1.9650 -1.8999 4.4402 -0.0023 -0.0011 0.0027

-1.9650 -1.9000 4.4403 -0.0015 -0.0017 -0.0001

-1.9651 -1.9000 4.4402 -0.0008 -0.0020 -0.0022

-1.9651 -1.9001 4.4402 -0.0001 -0.0022 -0.0041

-1.9651 -1.9001 4.4400 0.0006 -0.0023 -0.0058

-1.9650 -1.9002 4.4398 0.0012 -0.0023 -0.0072

-1.9650 -1.9003 4.4395 0.0018 -0.0022 -0.0082

-1.9649 -1.9004 4.4392 0.0022 -0.0018 -0.0085

-1.9648 -1.9004 4.4389 0.0023 -0.0014 -0.0080

-1.9647 -1.9005 4.4386 0.0023 -0.0010 -0.0069

-1.9646 -1.9005 4.4383 0.0020 -0.0006 -0.0055

-1.9646 -1.9005 4.4382 0.0017 -0.0002 -0.0039

-1.9645 -1.9005 4.4381 0.0013 0.0002 -0.0023

-1.9645 -1.9005 4.4380 0.0008 0.0005 -0.0008

-1.9645 -1.9005 4.4380 0.0005 0.0007 0.0005

-1.9645 -1.9005 4.4380 0.0001 0.0008 0.0014

-1.9645 -1.9005 4.4381 -0.0002 0.0009 0.0023

-1.9645 -1.9004 4.4382 -0.0006 0.0009 0.0032

-1.9645 -1.9004 4.4384 -0.0009 0.0009 0.0038

-1.9646 -1.9003 4.4385 -0.0011 0.0008 0.0038

-1.9646 -1.9003 4.4387 -0.0011 0.0006 0.0035

-1.9646 -1.9003 4.4388 -0.0011 0.0004 0.0029

-1.9647 -1.9003 4.4389 -0.0009 0.0002 0.0022

-1.9647 -1.9003 4.4390 -0.0007 0.0000 0.0014

-1.9647 -1.9003 4.4390 -0.0005 -0.0002 0.0007

-1.9648 -1.9003 4.4390 -0.0003 -0.0003 0.0001

-1.9648 -1.9003 4.4390 -0.0002 -0.0004 -0.0005

-1.9648 -1.9003 4.4390 0.0000 -0.0004 -0.0010

-1.9648 -1.9003 4.4390 0.0002 -0.0005 -0.0014

-1.9647 -1.9004 4.4389 0.0004 -0.0005 -0.0018

-1.9647 -1.9004 4.4388 0.0005 -0.0004 -0.0018

-1.9647 -1.9004 4.4387 0.0005 -0.0003 -0.0016

-1.9647 -1.9004 4.4386 0.0005 -0.0002 -0.0013

-1.9647 -1.9004 4.4386 0.0004 -0.0001 -0.0009

-1.9646 -1.9004 4.4386 0.0003 0.0000 -0.0006

-1.9646 -1.9004 4.4386 0.0002 0.0001 -0.0002

-1.9646 -1.9004 4.4386 0.0001 0.0002 0.0001

-1.9646 -1.9004 4.4386 -0.0000 0.0002 0.0004

-1.9646 -1.9004 4.4386 -0.0001 0.0002 0.0007

-1.9646 -1.9004 4.4386 -0.0002 0.0002 0.0008

-1.9646 -1.9004 4.4387 -0.0002 0.0002 0.0008

-1.9647 -1.9004 4.4387 -0.0002 0.0001 0.0007

```
-1.9647 -1.9004 4.4387 -0.0002 0.0001 0.0006
-1.9647 -1.9004 4.4388 -0.0002 0.0000 0.0004
-1.9647 -1.9004 4.4388 -0.0001 -0.0000 0.0002
-1.9647 -1.9004 4.4388 -0.0001 -0.0001 -0.0000
-1.9647 -1.9004 4.4388 -0.0000 -0.0001 -0.0002
-1.9647 -1.9004 4.4388 0.0001 -0.0001 -0.0003
-1.9647 -1.9004 4.4387 0.0001 -0.0001 -0.0004
-1.9647 -1.9004 4.4387 0.0001 -0.0001 -0.0004
-1.9647 -1.9004 4.4387 0.0001 -0.0000 -0.0003
-1.9647 -1.9004 4.4387 0.0001 -0.0000 -0.0002
-1.9647 -1.9004 4.4387 0.0000 0.0000 -0.0001
-1.9647 -1.9004 4.4387 0.0000 0.0000 0.0000
-1.9647 -1.9004 4.4387 -0.0000 0.0000 0.0001
-1.9647 -1.9004 4.4387 -0.0000 0.0000 0.0002
-1.9647 -1.9004 4.4387 -0.0000 0.0000 0.0002
-1.9647 -1.9004 4.4387 -0.0000 0.0000 0.0001
-1.9647 -1.9004 4.4387 -0.0000 -0.0000 0.0000
-1.9647 -1.9004 4.4387 -0.0000 -0.0000 -0.0000
-1.9647 -1.9004 4.4387 0.0000 -0.0000 -0.0001];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure;
level = [ 0 0 ];
contour(g.xs1, g.xs2, final, level,'b','linewidth',3);
hold on;
contour(g.xs1, g.xs2, final5, level, 'g','linewidth',3);
contour(g.xs1, g.xs2, final3, level, 'y','linewidth',3);
hold on;
plot(xf(:,2)-xf(:,1),xf(:,3)-xf(:,1),'r:','linewidth',3)
hold on;
```

```
plot(xc(:,2)-xc(:,1),xc(:,3)-xc(:,1),'g:','linewidth',3);
hold on;
plot(xm(:,2)-xm(:,1),xm(:,3)-xm(:,1),'b:')
legend('pre-fault','post-fault(L_{15}), t_c^*=2.00 sec', 'post-fault(L_{46}), t_c^*=1.90 sec','during fault')
hold on;
plot(xmm(:,2)-xmm(:,1),xmm(:,3)-xmm(:,1),'b:')
hold on;
plot(xff(:,2)-xff(:,1),xff(:,3)-xff(:,1),'r:','linewidth',3)
hold on;
plot(xcc(:,2)-xcc(:,1),xcc(:,3)-xcc(:,1),'y:','linewidth',3);
xlabel(' delta_{21}');
ylabel(' delta_{31}');
axis equal;
axis(g.axis);
hold on;
axis equal;
plot(0.0560,0.0783, 'bo');
hold on;
hold on;
plot(0.0643,0.1202, 'g+');
hold on;
plot(0.0584,0.1381,'ys');
```

APPENDIX B. AADL/AGREE CODE FOR NETWORK EXAMPLE OF CHAPTER 5

Here, we provide the *AADL/AGREE* source code for the implementation of the network example, adapted from [Jha et al. (2002); Sheyner et al. (2002)].

---

package data_types_implementation

public

with Base_Types;–built in data types package

–8 possible attack data name types from PC_0 to PC_1, and PC_2.

---

data sbo_01

end sbo_01;

data ftpr_01

end ftpr_01;

data rlog_01

end rlog_01;

data lbo_01

end lbo_01;

data sbo_02

end sbo_02;

data ftpr_02

end ftpr_02;

data rlog_02

end rlog_02;

data lbo_02

end lbo_02;

———————————————————————-

– 8 possible attack data implementation types from PC_0 to PC_1, and PC_2.

———————————————————————

data implementation sbo_01.impl

subcomponents

val: data Base_Types::boolean;

end sbo_01.impl;

data implementation ftpr_01.impl

subcomponents

val: data Base_Types::boolean;

end ftpr_01.impl;

data implementation rlog_01.impl

subcomponents

val: data Base_Types::boolean;

end rlog_01.impl;

data implementation lbo_01.impl

subcomponents

val: data Base_Types::boolean;

end lbo_01.impl;

data implementation sbo_02.impl

subcomponents

val: data Base_Types::boolean;

end sbo_02.impl;

data implementation ftpr_02.impl

subcomponents

val: data Base_Types::boolean;

end ftpr_02.impl;

data implementation rlog_02.impl

subcomponents

val: data Base_Types::boolean;

end rlog_02.impl;

data implementation lbo_02.impl

subcomponents

val: data Base_Types::boolean;

end lbo_02.impl;

———————————————-

–6 possible attack data name types from PC_1 to PC_2, and PC_1.

———————————————

data sbo_12

end sbo_12;

data ftpr_12

end ftpr_12;

data rlog_12

end rlog_12;

data lbo_12

end lbo_12;

data ftpr_11

end ftpr_11;

data lbo_11

end lbo_11;

———————————————————————-

–6 possible attack data implementation types from PC_1 to PC_2, and PC_1.

———————————————————————

data implementation sbo_12.impl

subcomponents

val: data Base_Types::boolean;

end sbo_12.impl;

data implementation ftpr_12.impl

subcomponents

val: data Base_Types::boolean;

end ftpr_12.impl;

data implementation rlog_12.impl

subcomponents

val: data Base_Types::boolean;

end rlog_12.impl;

data implementation lbo_12.impl

subcomponents

val: data Base_Types::boolean;

end lbo_12.impl;

data implementation ftpr_11.impl

subcomponents

val: data Base_Types::boolean;

end ftpr_11.impl;

data implementation lbo_11.impl

subcomponents

val: data Base_Types::boolean;

end lbo_11.impl;

———————————————-

–6 possible attack data name types from PC_2 to PC_1, and PC_2.

———————————————

data sbo_21

end sbo_21;

data ftpr_21

end ftpr_21;

data rlog_21

end rlog_21;

data lbo_21

end lbo_21;

data ftpr_22

end ftpr_22;

data lbo_22

end lbo_22;

–6 possible attack data implementation types from PC_2 to PC_1, and PC_2.

---

data implementation sbo_21.impl

subcomponents

val: data Base_Types::boolean;

end sbo_21.impl;

data implementation ftpr_21.impl

subcomponents

val: data Base_Types::boolean;

end ftpr_21.impl;

data implementation rlog_21.impl

subcomponents

val: data Base_Types::boolean;

end rlog_21.impl;

data implementation lbo_21.impl

subcomponents

val: data Base_Types::boolean;

end lbo_21.impl;

data implementation ftpr_22.impl

subcomponents

val: data Base_Types::boolean;

end ftpr_22.impl;

data implementation lbo_22.impl

subcomponents

val: data Base_Types::boolean;

end lbo_22.impl;

———————————————————————-

—4 possible Intrusion detection data name types.

———————————————————————

data id_sbo_01

end id_sbo_01;

data id_sbo_02

end id_sbo_02;

data id_rlog_01

end id_rlog_01;

data id_rlog_02

end id_rlog_02;

———————————————————-

–4 possible Intrusion detection data implementation types.

———————————————————

data implementation id_sbo_01.impl

subcomponents

val: data Base_Types::boolean;

end id_sbo_01.impl;

data implementation id_sbo_02.impl

subcomponents

val: data Base_Types::boolean;

end id_sbo_02.impl;

data implementation id_rlog_01.impl

subcomponents

val: data Base_Types::boolean;

end id_rlog_01.impl;

data implementation id_rlog_02.impl

subcomponents

val: data Base_Types::boolean;

end id_rlog_02.impl;

—the property of attacker level l_2.

---

data l_2_level

end l_2_level;

–the data implementation type of attacker level l_2.

---

data implementation l_2_level.impl

subcomponents

val: data Base_Types::integer;

end l_2_level.impl;

— the property of intrusion detection dg.

---

data dg_detection

end dg_detection;

–the data implementation type of intrusion detection dg.

---

data implementation dg_detection.impl

subcomponents

val: data Base_Types::integer;

end dg_detection.impl;

end data_types_implementation;

---

package system_model

public

with data_types_implementation;–this package defines all used data types/implementations.

–4 main components: PC_0, PC_1,PC_2,ID.

system PC_0 – attacker host.

features

thr_sbo_01: out data port data_types_implementation::sbo_01.impl;

thr_ftpr_01: out data port data_types_implementation::ftpr_01.impl;

thr_rlog_01: out data port data_types_implementation::rlog_01.impl;

thr_lbo_01: out data port data_types_implementation::lbo_01.impl;

thr_sbo_02: out data port data_types_implementation::sbo_02.impl;

thr_ftpr_02: out data port data_types_implementation::ftpr_02.impl;

thr_rlog_02: out data port data_types_implementation::rlog_02.impl;

thr_lbo_02: out data port data_types_implementation::lbo_02.impl;

---

annex agree {**

const l_0:int=2;–attacker level at PC_0 is root(2).

guarantee "sbo_01_threat": if (l_0≥1) then (thr_sbo_01.val=true or thr_sbo_01.val=false) else thr_sbo_01.val=false;

guarantee "ftpr_01_threat":if (l_0≥1) then (thr_ftpr_01.val=true or thr_ftpr_01.val=false) else thr_ftpr_01.val=false;

guarantee "rlog_01_threat": if (l_0≥1) then (thr_rlog_01.val=true or thr_rlog_01.val=false) else thr_rlog_01.val=false;

guarantee "sbo_02_threat":if (l_0≥1) then (thr_sbo_02.val=true or thr_sbo_02.val=false) else thr_sbo_02.val=false;

guarantee "ftpr_02_threat": if (l_0≥1) then (thr_ftpr_02.val=true or thr_ftpr_02.val=false) else thr_ftpr_02.val=false;

guarantee "rlog_02_threat": if (l_0≥1) then (thr_rlog_02.val=true or thr_rlog_02.val=false) else thr_rlog_02.val=false;

**};

end PC_0;

---

system PC_1 –host_1.

features

thr_sbo_01: in data port data_types_implementation::sbo_01.impl;

thr_ftpr_01: in data port data_types_implementation::ftpr_01.impl;

thr_rlog_01: in data port data_types_implementation::rlog_01.impl;

thr_lbo_01: in data port data_types_implementation::lbo_01.impl;

thr_sbo_12: out data port data_types_implementation::sbo_12.impl;

thr_ftpr_12: out data port data_types_implementation::ftpr_12.impl;

thr_rlog_12: out data port data_types_implementation::rlog_12.impl;

thr_lbo_12: out data port data_types_implementation::lbo_12.impl;

thr_lbo_11:out data port data_types_implementation::lbo_11.impl;

thr_ftpr_11:out data port data_types_implementation::ftpr_11.impl;

thr_sbo_21: in data port data_types_implementation::sbo_21.impl;

thr_ftpr_21: in data port data_types_implementation::ftpr_21.impl;

thr_rlog_21: in data port data_types_implementation::rlog_21.impl;

thr_lbo_21: in data port data_types_implementation::lbo_21.impl;

id_done_sbo_01:out data port data_types_implementation::id_sbo_01.impl;

id_done_rlog_01:out data port data_types_implementation::id_rlog_01.impl;

---

annex agree{**

eq done_sbo_01:bool= (thr_sbo_01.val) and (l_1<2) and (sshd_1);

–precondition/attack instance sbo_01.

eq done_ftpr_01:bool=(thr_ftpr_01.val)and (t_01=false or t_21=false) and ftp_1 and wdir_1 and fshell_1;

–precondition/attack instance ftpr_01.

eq done_rlog_01:bool=(thr_rlog_01.val)and (l_1=0) and (t_01);

–precondition/attack instance rlog_01.

eq done_lbo_01:bool=(thr_lbo_01.val)and (l_1=1)and xterm_1;

–precondition/attack instance lbo_01.

eq done_sbo_21:bool=(thr_sbo_21.val)and (l_1<2) and (sshd_1);

–precondition/attack instance sbo_21.

eq done_ftpr_21:bool=(thr_ftpr_21.val)and (t_01=false or t_21=false) and ftp_1 and wdir_1 and fshell_1;

eq done_rlog_21:bool=(thr_rlog_21.val)and (l_1=0) and (t_21);

eq done_lbo_21:bool=(thr_lbo_21.val)and (l_1=1)and xterm_1;

eq done_lbo_11:bool=(thr_lbo_11.val)and(l_1=1)and xterm_1;

eq done_ftpr_11:bool=(thr_ftpr_11.val)and(l_1≥1)and (t_01=false or t_21=false) and ftp_1 and wdir_1 and fshell_1;

————————define all vulnerabilities, and services of PC_1.

const xterm_1:bool=true; const ftp_1:bool=true;const wdir_1:bool=true;const fshell_1:bool=true;

eq l_1:int=(0− > if pre(done_sbo_01) then 2 —the attacker level at PC_1.

else if pre(done_rlog_01) then 1

else if pre(done_lbo_01) then 2

else if pre(done_sbo_21) then 2

else if pre(done_rlog_21) then 1

else if pre(done_lbo_21)then 2

else if pre(done_lbo_11) then 2

else pre(l_1));

eq sshd_1:bool=(true− > if pre(done_sbo_01) then false —the sshd service at PC_1.

else if pre(done_sbo_21)then false

else pre(sshd_1)

);

eq t_01:bool=(false− >if pre(done_ftpr_01) then true —PC_1 trusts PC_0.

else if pre(done_ftpr_21) then true

else if pre(done_ftpr_11) then true

else pre (t_01)

);

eq t_21:bool=(false− >if pre(done_ftpr_01) then true —PC_1 trusts PC_2

else if pre(done_ftpr_21) then true

else if pre(done_ftpr_11) then true

else pre (t_21)

);

guarantee "sbo_12_threat":if (l_1≥1) then (thr_sbo_12.val=true or thr_sbo_12.val=false) else thr_sbo_12.val=false;

guarantee "ftpr_12_threat":if (l_1≥1)then (thr_ftpr_12.val=true or thr_ftpr_12.val=false) else thr_ftpr_12.val=false;

guarantee "rlog_12_threat":if (l_1≥1)then (thr_rlog_12.val=true or thr_rlog_12.val=false) else thr_rlog_12.val=false;

guarantee "id_sbo_01_value":id_done_sbo_01.val= done_sbo_01;

guarantee "id_rlog_01_value":id_done_rlog_01.val= done_rlog_01;

**};

end PC_1;

---

system PC_2–host_2.

features

thr_lbo_02: in data port data_types_implementation::lbo_02.impl;

thr_sbo_02: in data port data_types_implementation::sbo_02.impl;

thr_ftpr_02: in data port data_types_implementation::ftpr_02.impl;

thr_rlog_02: in data port data_types_implementation::rlog_02.impl;

thr_sbo_12: in data port data_types_implementation::sbo_12.impl;

thr_ftpr_12: in data port data_types_implementation::ftpr_12.impl;

thr_rlog_12: in data port data_types_implementation::rlog_12.impl;

thr_lbo_12: in data port data_types_implementation::lbo_12.impl;

thr_sbo_21: out data port data_types_implementation::sbo_21.impl;

thr_ftpr_21: out data port data_types_implementation::ftpr_21.impl;

thr_rlog_21: out data port data_types_implementation::rlog_21.impl;

thr_lbo_21: out data port data_types_implementation::lbo_21.impl;

thr_lbo_22:out data port data_types_implementation::lbo_22.impl;

thr_ftpr_22:out data port data_types_implementation::ftpr_22.impl;

id_done_sbo_02: out data port data_types_implementation::id_sbo_02.impl;

id_done_rlog_02: out data port data_types_implementation::id_rlog_02.impl;

test_l_2:out data port data_types_implementation::l_2_level.impl ;

---

annex agree{**

eq done_sbo_02:bool=(thr_sbo_02.val)and (l_2<2) and sshd_2;

eq done_ftpr_02:bool=(thr_ftpr_02.val)and (t_02=false or t_12=false) and ftp_2 and wdir_2 and fshell_2;

eq done_rlog_02:bool=(thr_rlog_02.val)and (l_2=0) and (t_02);

eq done_lbo_02:bool=(thr_lbo_02.val)and (l_2=1) and xterm_2;

eq done_sbo_12:bool=(thr_sbo_12.val)and (l_2<2) and sshd_2;

eq done_ftpr_12:bool=(thr_ftpr_12.val)and (t_02=false or t_12=false) and ftp_2 and wdir_2 and fshell_2;

eq done_rlog_12:bool=(thr_rlog_12.val)and (l_2=0) and (t_12);

eq done_lbo_12:bool=(thr_lbo_12.val)and (l_2=1)and xterm_2;

eq done_lbo_22:bool= (thr_lbo_22.val)and(l_2=1)and xterm_2;

eq done_ftpr_22:bool= (thr_ftpr_22.val)and(l_2≥1)and (t_02=false or t_12=false) and ftp_2 and wdir_2 and fshell_2;

const xterm_2:bool=true; const sshd_2:bool=false;

eq l_2:int=(0− > if pre(done_sbo_02) then 2

else if pre(done_rlog_02) then 1

else if pre(done_lbo_02) then 2

else if pre(done_sbo_12) then 2

else if pre(done_rlog_12) then 1

else if pre(done_lbo_12) then 2

else if pre(done_lbo_22) then 2

else pre(l_2));

const ftp_2:bool=true;const wdir_2:bool=true;const fshell_2:bool=true; const data_2:bool=true;

eq t_02:bool=(false− >if pre(done_ftpr_02) then true

else if pre(done_ftpr_12) then true

else if pre(done_ftpr_22) then true

else pre (t_02)

);

eq t_12:bool=(false− >if pre(done_ftpr_02) then true

else if pre(done_ftpr_12) then true

else if pre(done_ftpr_22) then true

else pre (t_12)

);

guarantee "l_2_value":test_l_2.val= l_2;

guarantee "sbo_21_threat":if (l_2≥1) then (thr_sbo_21.val=true or thr_sbo_21.val=false) else thr_sbo_21.val=false;

guarantee "ftpr_21_threat": if (l_2≥1)then (thr_ftpr_21.val=true or thr_ftpr_21.val=false) else thr_ftpr_21.val=false;

guarantee "rlog_21_threat": if (l_2≥1)then (thr_rlog_21.val=true or thr_rlog_21.val=false) else thr_rlog_21.val=false;

guarantee ”id_sbo_02_value”:id_done_sbo_02.val= done_sbo_02;

guarantee ”id_rlog_02_value”:id_done_rlog_02.val= done_rlog_02;

**};

end PC_2;

---

system ID–the intrusion detection system.

features id_done_sbo_01:in data port data_types_implementation::id_sbo_01.impl;

id_done_rlog_01:in data port data_types_implementation::id_rlog_01.impl;

id_done_sbo_02: in data port data_types_implementation::id_sbo_02.impl;

id_done_rlog_02: in data port data_types_implementation::id_rlog_02.impl;

test_dg:out data port data_types_implementation::dg_detection.impl ;

---

annex agree{**

eq dg:int= 0;guarantee ”dg_value”:test_dg.val=dg− >if (id_done_sbo_01.val=true

and (pre(test_dg.val)=0))then (test_dg.val=0 or test_dg.val=1)

else if (id_done_sbo_02.val=true) and (pre(test_dg.val)=0) then (test_dg.val=0 or test_dg.val=1)

else if (id_done_rlog_01.val=true) then (test_dg.val=1)

else if (id_done_rlog_02.val=true) then (test_dg.val=1)

else (test_dg.val=pre(test_dg.val));

**};

end ID;

---

system Complete

features

test_l_2:out data port data_types_implementation::l_2_level.impl ;

test_dg:out data port data_types_implementation::dg_detection.impl ;

thr_sbo_01: out data port data_types_implementation::sbo_01.impl;

thr_ftpr_01: out data port data_types_implementation::ftpr_01.impl;

thr_rlog_01: out data port data_types_implementation::rlog_01.impl;

thr_lbo_01: out data port data_types_implementation::lbo_01.impl;

thr_sbo_02: out data port data_types_implementation::sbo_02.impl;

thr_ftpr_02: out data port data_types_implementation::ftpr_02.impl;

thr_rlog_02: out data port data_types_implementation::rlog_02.impl;

thr_lbo_02: out data port data_types_implementation::lbo_02.impl;

thr_sbo_12: out data port data_types_implementation::sbo_12.impl;

thr_ftpr_12: out data port data_types_implementation::ftpr_12.impl;

thr_rlog_12: out data port data_types_implementation::rlog_12.impl;

thr_lbo_12: out data port data_types_implementation::lbo_12.impl;

thr_lbo_11:out data port data_types_implementation::lbo_11.impl;

thr_ftpr_11: out data port data_types_implementation::ftpr_11.impl;

thr_sbo_21: out data port data_types_implementation::sbo_21.impl;

thr_ftpr_21: out data port data_types_implementation::ftpr_21.impl;

thr_rlog_21: out data port data_types_implementation::rlog_21.impl;

thr_lbo_21: out data port data_types_implementation::lbo_21.impl;

thr_lbo_22:out data port data_types_implementation::lbo_22.impl;

thr_ftpr_22: out data port data_types_implementation::ftpr_22.impl;

annex agree{**

—we test the following property under the assumption that only one attack instance can occur at a time.

guarantee "security property":(test_l_2.val<2 or test_dg.val=1);–attacker on host_2 has privilege level below root or gets detected.

—to generate a second attack scenario, we encode the discovered counter example CE_1, given below, in disjunct with the security property P being checked, namely $P \vee CE_1$. This yields a new counter example.

–The following formula describes the the discovered counter example CE_1, uncomment it to generate a second attack scenario.

– eq CE_1:bool=(thr_sbo_01.val=false and pre(thr_sbo_01.val=false)

and pre(pre(thr_sbo_01.val=false))

and pre(pre(pre(thr_sbo_01.val=false))) and pre(pre(pre(pre(thr_sbo_01.val=true)))))

–and(thr_ftpr_02.val=false and pre(thr_ftpr_02.val=false)and pre(pre(thr_ftpr_02.val=false)) and pre(pre(pre(thr_ftpr_02.val=true))) and pre(pre(pre(pre(thr_ftpr_02.val=false)))))

–and(thr_rlog_12.val=false and pre(thr_rlog_12.val=false)and pre(pre(thr_rlog_12.val=true)) and pre(pre(pre(thr_rlog_12.val=false))) and pre(pre(pre(pre(thr_rlog_12.val=false)))))

–and(thr_lbo_02.val=false and pre(thr_lbo_02.val=true)and pre(pre(thr_lbo_02.val=false)) and pre(pre(pre(thr_lbo_02.val=false))) and pre(pre(pre(pre(thr_lbo_02.val=false)))));

—to generate a second counter example, the security property is modified as follows (uncomment it):

– guarantee "security property":((test_l_2.val<2 or test_dg.val=1) or CE_1);

assume "one_threat_active":

(thr_ftpr_01.val=> (thr_ftpr_02.val or thr_sbo_01.val or thr_rlog_01.val

or thr_sbo_02.val or thr_rlog_02.val or thr_lbo_01.val or thr_lbo_02.val or thr_sbo_12.val

or thr_ftpr_12.val or thr_rlog_12.val or thr_lbo_12.val or thr_sbo_21.val or thr_ftpr_21.val or thr_rlog_21.val or thr_lbo_21.val or thr_lbo_11.val

or thr_lbo_22.val or thr_ftpr_11.val or thr_ftpr_22.val) =false)

and (thr_sbo_01.val=> (thr_ftpr_01.val or thr_rlog_01.val or thr_sbo_02.val or thr_ftpr_02.val

or thr_rlog_02.val or thr_lbo_01.val or thr_lbo_02.val or thr_sbo_12.val or thr_ftpr_12.val or thr_rlog_12.val or thr_lbo_12.val or thr_sbo_21.val

or thr_ftpr_21.val or thr_rlog_21.val

or thr_lbo_21.val or thr_lbo_11.val

or thr_lbo_22.val or thr_ftpr_11.val or thr_ftpr_22.val)=false)

and (thr_rlog_01.val=> (thr_ftpr_01.val

or thr_sbo_01.val or thr_sbo_02.val

or thr_ftpr_02.val or thr_rlog_02.val or thr_lbo_01.val or thr_lbo_02.val

or thr_sbo_12.val or thr_ftpr_12.val

or thr_rlog_12.val or thr_lbo_12.val or thr_sbo_21.val or thr_ftpr_21.val

or thr_rlog_21.val or thr_lbo_21.val

or thr_lbo_11.val or thr_lbo_22.val or thr_ftpr_11.val or thr_ftpr_22.val)=false)

and (thr_sbo_02.val=> (thr_rlog_01.val or thr_ftpr_01.val or thr_sbo_01.val or thr_ftpr_02.val

or thr_rlog_02.val or thr_lbo_01.val or thr_lbo_02.val or thr_sbo_12.val

or thr_ftpr_12.val or thr_rlog_12.val

or thr_lbo_12.val or thr_sbo_21.val or thr_ftpr_21.val or thr_rlog_21.val

or thr_lbo_21.val or thr_lbo_11.val

or thr_lbo_22.val or thr_ftpr_11.val or thr_ftpr_22.val)=false)

and (thr_ftpr_02.val => (thr_ftpr_01.val

or thr_rlog_01.val

or thr_sbo_01.val or thr_sbo_02.val or thr_rlog_02.val or thr_lbo_01.val or thr_lbo_02.val

or thr_sbo_12.val

or thr_ftpr_12.val or thr_rlog_12.val or thr_lbo_12.val or thr_sbo_21.val or thr_ftpr_21.va

l or thr_rlog_21.val or thr_lbo_21.val or thr_lbo_11.val or thr_lbo_22.val

or thr_ftpr_11.val or thr_ftpr_22.val)=false)

and (thr_rlog_02.val=> (thr_rlog_01.val or thr_ftpr_01.val

or thr_sbo_01.val or thr_sbo_02.val or thr_ftpr_02.val or thr_lbo_01.val or thr_lbo_02.val

or thr_sbo_12.val or thr_ftpr_12.val or thr_rlog_12.val or thr_lbo_12.val or thr_sbo_21.val or thr_ftpr_21.val or thr_rlog_21.val

or thr_lbo_21.val or thr_lbo_11.val or thr_lbo_22.val or thr_ftpr_11.val or thr_ftpr_22.val)=false)

and (thr_lbo_01.val=> (thr_rlog_01.val or thr_ftpr_01.val or thr_sbo_01.val

or thr_sbo_02.val or thr_ftpr_02.val or thr_rlog_02.val or thr_lbo_02.val or thr_sbo_12.val or thr_ftpr_12.val

or thr_rlog_12.val or thr_lbo_12.val or thr_sbo_21.val or thr_ftpr_21.val or thr_rlog_21.val or thr_lbo_21.val

or thr_lbo_11.val or thr_lbo_22.val or thr_ftpr_11.val or thr_ftpr_22.val)=false)

and (thr_lbo_02.val=> (thr_rlog_01.val or thr_ftpr_01.val or thr_sbo_01.val or thr_sbo_02.val

or thr_ftpr_02.val or thr_rlog_02.val or thr_lbo_01.val or thr_sbo_12.val or thr_ftpr_12.val or thr_rlog_12.val

or thr_lbo_12.val or thr_sbo_21.val or thr_ftpr_21.val or thr_rlog_21.val or thr_lbo_21.val or

thr_lbo_11.val

or thr_lbo_22.val or thr_ftpr_11.val or thr_ftpr_22.val)=false)

and (thr_sbo_12.val => (thr_ftpr_12.val or thr_rlog_12.val or thr_lbo_12.val

or thr_ftpr_01.val

or thr_ftpr_02.val or thr_sbo_01.val or thr_rlog_01.val or thr_sbo_02.val or thr_rlog_02.val

or thr_lbo_01.val

or thr_lbo_02.val or thr_sbo_21.val or thr_ftpr_21.val or thr_rlog_21.val or thr_lbo_21.val

or thr_lbo_11.val

or thr_lbo_22.val or thr_ftpr_11.val or thr_ftpr_22.val)=false)

and (thr_ftpr_12.val=>(thr_sbo_12.val or thr_rlog_12.val or thr_lbo_12.val or thr_ftpr_01.val

or thr_ftpr_02.val or thr_sbo_01.val or thr_rlog_01.val or thr_sbo_02.val

or thr_rlog_02.val or thr_lbo_01.val

or thr_lbo_02.val or thr_sbo_21.val or thr_ftpr_21.val or thr_rlog_21.val or thr_lbo_21.val or thr_lbo_11.val

or thr_lbo_22.val or thr_ftpr_11.val or thr_ftpr_22.val)=false)

and (thr_rlog_12.val=>(thr_ftpr_12.val or thr_sbo_12.val or thr_lbo_12.val or thr_ftpr_01.val

or thr_ftpr_02.val or thr_sbo_01.val or thr_rlog_01.val or thr_sbo_02.val or thr_rlog_02.val or thr_lbo_01.val

or thr_lbo_02.val or thr_sbo_21.val or thr_ftpr_21.val or thr_rlog_21.val or thr_lbo_21.val or thr_lbo_11.val

or thr_lbo_22.val or thr_ftpr_11.val or thr_ftpr_22.val)=false)

and (thr_lbo_12.val=>(thr_ftpr_12.val or thr_rlog_12.val or thr_sbo_12.val or thr_ftpr_01.val

or thr_ftpr_02.val or thr_sbo_01.val or thr_rlog_01.val or thr_sbo_02.val or thr_rlog_02.val or thr_lbo_01.val

or thr_lbo_02.val or thr_sbo_21.val or thr_ftpr_21.val or thr_rlog_21.val or thr_lbo_21.val or thr_lbo_11.val

or thr_lbo_22.val or thr_ftpr_11.val or thr_ftpr_22.val)=false)

and (thr_sbo_21.val =>(thr_ftpr_21.val or thr_rlog_21.val or thr_lbo_21.val or thr_ftpr_01.val

or thr_ftpr_02.val or thr_sbo_01.val or thr_rlog_01.val or thr_sbo_02.val or thr_rlog_02.val or

thr_lbo_01.val or thr_lbo_02.val

or thr_sbo_12.val or thr_ftpr_12.val or thr_rlog_12.val or thr_lbo_12.val or thr_lbo_11.val

or thr_lbo_22.val or thr_ftpr_11.val or thr_ftpr_22.val)=false)

and (thr_ftpr_21.val=>(thr_sbo_21.val or thr_rlog_21.val or thr_lbo_21.val or thr_ftpr_01.val

or thr_ftpr_02.val or thr_sbo_01.val or thr_rlog_01.val or thr_sbo_02.val or thr_rlog_02.val or thr_lbo_01.val or thr_lbo_02.val

or thr_sbo_12.val or thr_ftpr_12.val or thr_rlog_12.val or thr_lbo_12.val or thr_lbo_11.val

or thr_lbo_22.val or thr_ftpr_11.val or thr_ftpr_22.val)=false)

and (thr_rlog_21.val=>(thr_ftpr_21.val or thr_sbo_21.val or thr_lbo_21.val or thr_ftpr_01.val

or thr_ftpr_02.val or thr_sbo_01.val or thr_rlog_01.val or thr_sbo_02.val or thr_rlog_02.val or thr_lbo_01.val or thr_lbo_02.val

or thr_sbo_12.val or thr_ftpr_12.val or thr_rlog_12.val or thr_lbo_12.val or thr_lbo_11.val or thr_lbo_22.val

or thr_ftpr_11.val or thr_ftpr_22.val)=false)

and (thr_lbo_21.val=>(thr_ftpr_21.val or thr_rlog_21.val or thr_sbo_21.val or thr_ftpr_01.val

or thr_ftpr_02.val or thr_sbo_01.val or thr_rlog_01.val or thr_sbo_02.val or thr_rlog_02.val or thr_lbo_01.val

or thr_lbo_02.val or thr_sbo_12.val or thr_ftpr_12.val or thr_rlog_12.val or thr_lbo_12.val or thr_lbo_11.val

or thr_lbo_22.val or thr_ftpr_11.val or thr_ftpr_22.val)=false)

and (thr_lbo_11.val=> (thr_lbo_22.val or thr_ftpr_11.val or thr_ftpr_22.val or thr_ftpr_01.val or thr_ftpr_02.val

or thr_sbo_01.val or thr_rlog_01.val or thr_sbo_02.val or thr_rlog_02.val or thr_lbo_01.val or thr_lbo_02.val

or thr_sbo_12.val or thr_ftpr_12.val or thr_rlog_12.val or thr_lbo_12.val or thr_sbo_21.val or thr_ftpr_21.val

or thr_rlog_21.val or thr_lbo_21.val)=false)

and (thr_lbo_22.val=> (thr_lbo_11.val or thr_ftpr_11.val or thr_ftpr_22.val or thr_ftpr_01.val or thr_ftpr_02.val

or thr_sbo_01.val or thr_rlog_01.val or thr_sbo_02.val or thr_rlog_02.val or thr_lbo_01.val or thr_lbo_02.val

or thr_sbo_12.val or thr_ftpr_12.val or thr_rlog_12.val or thr_lbo_12.val or thr_sbo_21.val or thr_ftpr_21.val

or thr_rlog_21.val or thr_lbo_21.val)=false)

and (thr_ftpr_11.val=> (thr_lbo_11.val or thr_lbo_22.val or thr_ftpr_22.val or thr_ftpr_01.val or thr_ftpr_02.val

or thr_sbo_01.val or thr_rlog_01.val or thr_sbo_02.val or thr_rlog_02.val or thr_lbo_01.val or thr_lbo_02.val

or thr_sbo_12.val or thr_ftpr_12.val or thr_rlog_12.val or thr_lbo_12.val or thr_sbo_21.val or thr_ftpr_21.val

or thr_rlog_21.val or thr_lbo_21.val)=false)

and (thr_ftpr_22.val=> (thr_lbo_11.val or thr_lbo_22.val or thr_ftpr_11.val or thr_ftpr_01.val or thr_ftpr_02.val

or thr_sbo_01.val or thr_rlog_01.val or thr_sbo_02.val or thr_rlog_02.val or thr_lbo_01.val or thr_lbo_02.val

or thr_sbo_12.val or thr_ftpr_12.val or thr_rlog_12.val or thr_lbo_12.val or thr_sbo_21.val or thr_ftpr_21.val

or thr_rlog_21.val or thr_lbo_21.val)=false) ;

**};

end Complete;

system implementation Complete.Impl

subcomponents

A_sub : system PC_0 ;

H1_sub : system PC_1 ;

H2_sub : system PC_2;

Int_det: system ID;

connections

test_l_2_output: port H2_sub.test_l_2− >test_l_2

{*Communication_Properties* :: *Timing* => *immediate*; };

test_dg_output: port Int_det.test_dg− >test_dg

{*Communication_Properties* :: *Timing* => *immediate*; };

pc1_id_sbo_01:port H1_sub.id_done_sbo_01− >Int_det.id_done_sbo_01

{*Communication_Properties* :: *Timing* => *immediate*; };

pc2_id_sbo_02:port H2_sub.id_done_sbo_02− >Int_det.id_done_sbo_02

{*Communication_Properties* :: *Timing* => *immediate*; };

pc1_id_rlog_01:port H1_sub.id_done_rlog_01− >Int_det.id_done_rlog_01

{*Communication_Properties* :: *Timing* => *immediate*; };

pc2_id_rlog_02:port H2_sub.id_done_rlog_02− >Int_det.id_done_rlog_02

{*Communication_Properties* :: *Timing* => *immediate*; };

sbo_01_T : port A_sub.thr_sbo_01− > H1_sub.thr_sbo_01

{*Communication_Properties* :: *Timing* => *immediate*; };

ftpr_01_T : port A_sub.thr_ftpr_01− > H1_sub.thr_ftpr_01

{*Communication_Properties* :: *Timing* => *immediate*; };

rlog_01_T : port A_sub.thr_rlog_01− > H1_sub.thr_rlog_01

{*Communication_Properties* :: *Timing* => *immediate*; };

lbo_01_T: port A_sub.thr_lbo_01− > H1_sub.thr_lbo_01

{*Communication_Properties* :: *Timing* => *immediate*; };

sbo_02_T : port A_sub.thr_sbo_02− > H2_sub.thr_sbo_02

{*Communication_Properties* :: *Timing* => *immediate*; };

ftpr_02_T : port A_sub.thr_ftpr_02− > H2_sub.thr_ftpr_02

{*Communication_Properties* :: *Timing* => *immediate*; };

rlog_02_T : port A_sub.thr_rlog_02− > H2_sub.thr_rlog_02

{*Communication_Properties* :: *Timing* => *immediate*; };

lbo_02_T: port A_sub.thr_lbo_02− > H2_sub.thr_lbo_02

{*Communication_Properties* :: *Timing* => *immediate*; };

sbo_12_T : port H1_sub.thr_sbo_12− > H2_sub.thr_sbo_12

{*Communication_Properties* :: *Timing* => *immediate*; };

ftpr_12_T : port H1_sub.thr_ftpr_12− > H2_sub.thr_ftpr_12

{*Communication_Properties* :: *Timing* => *immediate*; };

rlog_12_T : port H1_sub.thr_rlog_12− > H2_sub.thr_rlog_12

{*Communication_Properties* :: *Timing* => *immediate*; };

lbo_12_T: port H1_sub.thr_lbo_12− > H2_sub.thr_lbo_12

{*Communication_Properties* :: *Timing* => *immediate*; };

sbo_21_T : port H2_sub.thr_sbo_21− > H1_sub.thr_sbo_21

{*Communication_Properties* :: *Timing* => *immediate*; };

ftpr_21_T : port H2_sub.thr_ftpr_21− > H1_sub.thr_ftpr_21

{*Communication_Properties* :: *Timing* => *immediate*; };

rlog_21_T : port H2_sub.thr_rlog_21− > H1_sub.thr_rlog_21

{*Communication_Properties* :: *Timing* => *immediate*; };

lbo_21_T: port H2_sub.thr_lbo_21− > H1_sub.thr_lbo_21

{*Communication_Properties* :: *Timing* => *immediate*; };

sbo_01:port A_sub.thr_sbo_01− >thr_sbo_01

{*Communication_Properties* :: *Timing* => *immediate*; };

ftpr_01:port A_sub.thr_ftpr_01− >thr_ftpr_01

{*Communication_Properties* :: *Timing* => *immediate*; };

rlog_01:port A_sub.thr_rlog_01− >thr_rlog_01

{*Communication_Properties* :: *Timing* => *immediate*; };

lbo_01: port A_sub.thr_lbo_01− >thr_lbo_01

{*Communication_Properties* :: *Timing* => *immediate*; };

sbo_02:port A_sub.thr_sbo_02− >thr_sbo_02

{*Communication_Properties* :: *Timing* => *immediate*; };

ftpr_02:port A_sub.thr_ftpr_02− >thr_ftpr_02

{*Communication_Properties* :: *Timing* => *immediate*; };

rlog_02:port A_sub.thr_rlog_02− >thr_rlog_02

{*Communication_Properties* :: *Timing* => *immediate*; };

lbo_02: port A_sub.thr_lbo_02− >thr_lbo_02

{*Communication_Properties* :: *Timing* => *immediate*; };

sbo_12:port H1_sub.thr_sbo_12− >thr_sbo_12

{*Communication_Properties* :: *Timing* => *immediate*; };

ftpr_12:port H1_sub.thr_ftpr_12− >thr_ftpr_12

{*Communication_Properties* :: *Timing* => *immediate*; };

rlog_12:port H1_sub.thr_rlog_12− >thr_rlog_12

{*Communication_Properties* :: *Timing* => *immediate*; };

lbo_12:port H1_sub.thr_lbo_12− >thr_lbo_12

{*Communication_Properties* :: *Timing* => *immediate*; };

sbo_21:port H2_sub.thr_sbo_21− >thr_sbo_21

{*Communication_Properties* :: *Timing* => *immediate*; };

ftpr_21:port H2_sub.thr_ftpr_21− >thr_ftpr_21

{*Communication_Properties* :: *Timing* => *immediate*; };

rlog_21:port H2_sub.thr_rlog_21− >thr_rlog_21

{*Communication_Properties* :: *Timing* => *immediate*; };

lbo_21:port H2_sub.thr_lbo_21− >thr_lbo_21

{*Communication_Properties* :: *Timing* => *immediate*; };

lbo_11:port H1_sub.thr_lbo_11− > thr_lbo_11

{*Communication_Properties* :: *Timing* => *immediate*; };

ftpr_11: port H1_sub.thr_ftpr_11− >thr_ftpr_11

{*Communication_Properties* :: *Timing* => *immediate*; };

lbo_22:port H2_sub.thr_lbo_22− > thr_lbo_22

{*Communication_Properties* :: *Timing* => *immediate*; };

ftpr_22: port H2_sub.thr_ftpr_22− >thr_ftpr_22

{*Communication_Properties* :: *Timing* => *immediate*; };

end Complete.Impl;

end system_model;

BIBLIOGRAPHY

Ahmad, E., Larson, B. R., Barrett, S. C., Zhan, N., and Dong, Y. (2014). Hybrid annex: An aadl extension for continuous behavior and cyber-physical interaction modeling. In *Proc. of the 2014 ACM SIGAda annual conference on High integrity language technology*, pages 29–38.

AS5506/1, S. I. S. (2006). Architecture analysis & design language (aadl) annex volume 1: Annex e: Error model annex. [online]. Available: http://standards.sae.org/as5506/1/.

AS5506/2, S. I. S. (2011). Architecture analysis & design language (aadl) annex volume 2: Annex d: Behavior annex. [online]. Available: http://standards.sae.org/as5506/2/.

Berthomieu, B., Bodeveix, J.-P., Farail, P., Filali, M., Garavel, H., Gaufillet, P., Lang, F., and Vernadat, F. (2008). Fiacre: an intermediate language for model verification in the topcased environment. In *Proc. of 4th International Congress on Embedded Real-Time Systems*.

Bozzano, M., Cimatti, A., Katoen, J.-P., Nguyen, V. Y., Noll, T., and Roveri, M. (2009). *The COMPASS Approach: Correctness,Modelling and Performability of Aerospace Systems*, pages 173–186. Springer Berlin Heidelberg, Berlin, Heidelberg.

Bryans, J., Koutny, M., and Mu, C. (2011). Towards quantitative analysis of opacity. *Technical Reports Series, Newcastle University*.

Carnegie-Mellon-University (2016). Open source aadl tool environment for the SAE architecture analysis and design language (AADL). [online]. Available: http://osate.github.io/index.html.

Chen, J., Ibrahim, M., and Kumar, R. (accepted (Sept. 2015)). Quantification of secrecy in partially observed stochastic discrete event systems. *IEEE Transactions on Automation Science and Engineering*.

Chen, J. and Kumar, R. (2015). Failure detection framework for stochastic discrete event systems with guaranteed error bounds. *IEEE Trans. on Automatic Control*, 60(6):1542–1553.

Christey, S. (2006). Preliminary list of vulnerability examples for researchers (PLOVER). [online]. Available: https://cwe.mitre.org/documents/sources/PLOVER.pdf.

Christian S. Collberg, C. T. (2002). Watermarking, tamper-proofing, and obfuscation-tools for software protection. *IEEE transactions on software engineering*, 28(8):735–746.

Cover, T. M. and Thomas, J. A. (2012). *Elements of information theory*. John Wiley and Sons.

Daemen, J. and Rijmen, V. (1999). AES proposal: Rijndael, version 2, AES submission.

Eclipse-Foundation (2004). Eclipse. [online]. Available: http://eclipse.org.

ESA, E. S. A. (2016). Taste (the assert set of tools for engineering). [online]. Available: http://download.tuxfamily.org/taste/.

Espinoza, B. and Smith, G. (2013). Min-entropy as a resource. *Information and Computation*, 226:57–75.

Feiler, P. H. and Gluch, D. P. (2012). *Model-Based Engineering with AADL: An Introduction to the SAE Architecture Analysis & Design Language*. Addison-Wesley.

Feiler, P. H., Hansson, J., de Niz, D., and Wrage, L. (2010). System architecture virtual integration: A case study. In *Proc. of Embedded Real-time Software and Systems Conference (ERTS2010)*.

Feiler, P. H. and Rugina, A. (2007). Dependability modeling with the architecture analysis & design language (aadl). [online]. Available: http://www.sei.cmu.edu/library/abstracts/reports/07tn043.cfm.

Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., and Waters, B. (2013). Candidate indistinguishability obfuscation and functional encryption for all circuits. In *Proc. of IEEE*

54th Annual Symposium on Foundations of Computer Science (FOCS' 2013), pages 40–49, Berkeley, CA.

Garg, V. K., Kumar, R., and Marcus, S. I. (1999). A probabilistic language formalism for stochastic discrete-event systems. 44(2):280–293.

Ibrahim, M., Chen, J., and Kumar, R. (2014). Secrecy in stochastic discrete event systems. In *Proc. of the 11th IEEE International Conference on Networking, Sensing and Control (ICNSC'14)*, pages 48–53, Miami, FL.

Ibrahim, M., Chen, J., and Kumar, R. (2015). An information theoretic measure for secrecy loss in stochastic discrete event systems. In *Proc. of the 7th International Conference on Electronics, Computers and Artificial Intelligence (ECAI'15)*, pages 1–6, Bucharest.

Ibrahim, M., Chen, J., and Kumar, R. (2016a). Quantication of distributed secrecy loss in stochastic discrete event systems under bounded-delay communications. In *Proc. of the 13th International Workshop on Discrete Event Systems*, Xian, China.

Ibrahim, M., Chen, J., and Kumar, R. (2016b). *Quantification of Centralized/Distributed Secrecy in Stochastic Discrete Event Systems*, pages 21–40. Springer International Publishing, Switzerland.

Ibrahim, M., Chen, J., and Kumar, R. (2016c). A resiliency measure for electrical power systems. In *Proc. of the 13th International Workshop on Discrete Event Systems*, Xian, China.

Jacob, R., Lesage, J.-J., and Faure, J.-M. (2015). Opacity of discrete event systems: models, validation and quantification. In *Proc. of the 5th international workshop on Dependable Control of Discrete Systems (DCDS'15), hal-01139890*, Cancun, Mexico.

Jha, S., Sheyner, O., and Wing, J. (2002). Two formal analyses of attack graphs. In *Proc. of the 15th IEEE Computer Security Foundations Workshop*.

Jin, L., Kumar, R., and Elia, N. (2010). Reachability analysis based transient stability design in power systems. *Electrical Power and Energy Systems*, 32(7):782–787.

Kaijser, T. (1975). A limit theorem for partially observed markov chains. *The Annals of Probability*, 3(4):677–696.

Kak, A. (2015). AES: Lecture notes in computer and network security. Purdue University, [online]. Available: https://engineering.purdue.edu/kak/compsec/NewLectures/Lecture8.pdf (accessed 1 May 2015).

Khalil, H. K. (2002). *Nonlinear Systems*. Prentice Hall, 3 edition.

Kinney, R., Crucitti, P., Albert, R., and Latora, V. (2005). Modeling cascading failures in the north american power grid. *The European Physical Journal B*, 46:101–107.

Kundur, D. and Ahsan, K. (2003). Practical internet steganography: Data hiding in ip. In *Proc. of Texas Workshop on Security of Information Systems*.

Larson, B. R., Chalin, P., and Hatclif, J. (2012). Bless: Formal specification and verification of behaviors for embedded systems with software. [online]. Available: http://bless.santoslab.org/node/5.

Latora, V. and Marchiori, M. (2001). Efficient behavior of small-world networks. *Physical Review Letters*, 87(19):198701.

Lin, J. (1991). Divergence measures based on the shannon entropy. *IEEE Trans. on Information Theory*, 37(1):145–151.

Liu, C.-W. and Thorp, J. S. (1997). A novel method to compute the closest unstable equilibrium point for transient stability region estimate in power systems. *IEEE Trans. Circuits and Systems-I: Fundamental Theory and Applications*, 44(7):630–635.

Maliszewski, P. J. and Perrings, C. (2012). Factors in the resilience of electrical power distribution infrastructures. *Applied Geography*, 32:668–679.

Mitchell, I. (2004). A toolbox of level set methods version 1.0. [online]. Available: http://www.cs.ubc.ca/ mitchell/ToolboxLS/.

MITRE (2014). Common configuration enumeration (CCE). [online]. Available: http://cce.mitre.org/about/index.html.

MITRE (2015a). Common attack pattern enumeration and classification (CAPEC). [online]. Available: https://capec.mitre.org/.

MITRE (2015b). Common vulnerabilities and exposures (CVE). [online]. Available: https://cve.mitre.org/.

MITRE (2015c). Common weakness enumeration (CWE). [online]. Available: https://cwe.mitre.org/.

OWASP-Foundation (2015). Open web application security project (OWASP). [online]. Available: https://www.owasp.org/index.php/Main_Page.

Pai, M. A. (1989). *Energy Function Analysis for Power System Stability*. Kluwer academic publishers.

Qiu, W. and Kumar, R. (2008). Distributed diagnosis under bounded-delay communication of immediately forwarded local observations. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*.

Redman, D., Ward, D., Chilenski, J., and Pollari, G. (2010). Virtual integration for improved system design. In *Proc. of the first Analytic Virtual Integration of Cyber-Physical Systems Workshop in conjunction with RTSS 2010*.

Ren, J. and Wu, J. (2010). Survey on anonymous communications in computer networks. *Computer Communications*, 33:420–431.

Rockwell-Collins (2012). The rockwell collins meta toolset. [online]. Availabl: https://wiki.sei.cmu.edu/aadl/index.php/RC_META.

Rockwell-Collins and Uof-Minnesota (2016a). The assume guarantee reasoning environment. [online]. Available: http://loonwerks.com/tools/agree.html.

Rockwell-Collins and Uof-Minnesota (2016b). Z3prover/z3. [online]. Available: https://github.com/Z3Prover/z3.

Saadat, H. (1999). *Power System Analysis*. McGraw-Hill Education.

SEI, S. E. I. (2004). Architecture analysis and design language. Carnegie-Mellon University, Pittsburgh, Pennsylvania, USA, [online]. Available: http://standards.sae.org/as5506/.

Sheeran, M., Singh, S., and StålmarckSheeran, G. (2000). *Checking Safety Properties Using Induction and a SAT-Solver*, pages 127–144. Springer Berlin Heidelberg, Berlin, Heidelberg.

Sheyner, O., Haines, J., Jha, S., Lippmann, R., and Wing, J. (2002). Automated generation and analysis of attack graphs. In *Proc. of IEEE Symposium on Security and Privacy*.

Shoukry, Y., Martin, P., Tabuada1, P., and Srivastava, M. (2013). Non-invasive spoofing attacks for anti-lock braking systems. In *Proc. of the 15th international conference on Cryptographic Hardware and Embedded Systems (CHES'13)*, Santa Barbara.

Smith, G. (2009). On the foundations of quantitative information flow. In *Proc. of Int. Conf. Foundations of Software Science and Computation Structures (FoSSaCS 09)*, pages 288–302.

Takai, S. and Kumar, R. (2009). Verification and synthesis for secrecy in discrete-event systems. In *Proc. of 2009*, pages 4741–4746, St. Louis, MO.

Thomas, R. J. and Thorp, J. S. (1985). Towards a direct test for large scale electric power system instabilities. In *Proc. of the 24th Conference on Decision and Control*, pages 65–69, Lauderdale, FL.

Thompson, M. J. (2012). Fundamentals and advancements in generator synchronizing systems. In *Proc. of the 65th Annual Conference for Protective Relay Engineers*, pages 203–214, College Station, TX.

Varaiya, P., Wu, F. F., and Chen, R.-L. (1985). Direct methods for transient stability analysis of power systems: Recent results. *IEEE Trans. Circuits and Systems-I: Fundamental Theory and Applications*, 73(12):1703–1715.

(W3C), W. W. C. (2004). Extensible markup language (XML)1.0 (third edition). [online]. Available: http://w3.org/TR/4004/REC-xml-20040204/.

Wang, X. and Ray, A. (2004). A language measure for performance evaluation of discrete-event supervisory control systems. *Applied Math. Modelling*, 28(9):817–833.

WASC (2005). Web application security consortium (WASC). [online]. Available: http://www.webappsec.org/.

Willis, H. H. and Loa, K. (2015). Measuring the resilience of energy distribution systems. *RAND Corporation, Santa Monica, Calif.*

Zhang, T. and ruby B. Lee (2014). Secure cache modeling for measuring side-channel leakage. *Technical Report, Princeton University.*

Zhu, B., Joseph, A., and Sastry, S. (2011). A taxonomy of cyber attacks on scada systems. In *Proc. of IEEE International Conferences on Internet of Things, and Cyber, Physical and Social Computing*, San Diego, California.