On Stochastic Modeling Applications to Cybersecurity:

Loss, Attack, and Detection

by

Axel La Salle

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

ARIZONA STATE UNIVERSITY

August 2023

ABSTRACT

The main objective of this work is to study novel stochastic modeling applications to cybersecurity aspects across three dimensions: Loss, attack, and detection. First, motivated by recent spatial stochastic models with cyber insurance applications, the first and second moments of the size of a typical cluster of bond percolation on finite graphs are studied. More precisely, having a finite graph where edges are independently open with the same probability p and a vertex x chosen uniformly at random, the goal is to find the first and second moments of the number of vertices in the cluster of open edges containing x. Exact expressions for the first and second moments of the size distribution of a bond percolation cluster on essential building blocks of hybrid graphs: the ring, the path, the random star, and regular graphs are derived. Upper bounds for the moments are obtained by using a coupling argument to compare the percolation model with branching processes when the graph is the random rooted tree with a given offspring distribution and a given finite radius. Second, the Petri Net modeling framework for performance analysis is well established; extensions provide enough flexibility to examine the behavior of a permissioned blockchain platform in the context of an ongoing cyberattack via simulation. The relationship between system performance and cyberattack configuration is analyzed. The simulations vary the blockchain's parameters and network structure, revealing the factors that contribute positively or negatively to a Sybil attack through the performance impact of the system. Lastly, the denoising diffusion probabilistic models (DDPM) ability for synthetic tabular data augmentation is studied. DDPMs surpass generative adversarial networks in improving computer vision classification tasks and image generation, for example, stable diffusion. Recent research and open-source implementations point to a strong quality of synthetic tabular data generation for classification and regression tasks.

Unfortunately, the present state of literature concerning tabular data augmentation with DDPM for classification is lacking. Further, cyber datasets commonly have highly unbalanced distributions complicating training. Synthetic tabular data augmentation is investigated with cyber datasets and performance of well-known metrics in machine learning classification tasks improve with augmentation and balancing.

DEDICATION

To my family and friends.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

INTRODUCTION

Cybersecurity risks, losses, and attacks are constant threats affecting every industry and government in the world. Recently, novel stochastic modeling applications have emerged in the academic and industry setting, evolving ever-increasingly. A recent definition of cyber risk appears in [1] stating that 'cyber risk means any risk of financial loss, disruption or damage to an organization's reputation from some sort of failure of its information technology systems.'. First, however, we focus on the reformulation in [2]. Motivated by the lack of loss distributions in the cyber insurance setting, Jevtic and Lanchier introduced a realistic framework based on random graphs and percolation theory to quantify the loss distribution due to cyberattacks on Local Area Networks in small and medium size companies [2]. In the context of Blockchain 2.0, their framework is generalized for cyber risk in the context of smart contracts [3]. Furthermore, their modeling framework is applicable to many industry cases. The modeling framework for cyber risk is adapted in [4] for the healthcare hospital setting. Additional industry-specific case studies can be found in [5]. In Chapter 1, we study the first and second moments of bond percolation clusters on a class of finite graphs that can be seen as building blocks to more general hybrid graphs. The network infrastructures arising from smart technologies or enterprise networks are typically more complex than these graphs. These complex structures, however, often consist of the composition of these simple graphs, so studying these classes of graphs above is a good starting point.

Hyperledger Fabric (HLF) is an open-source enterprise-grade permissioned distributed ledger technology (DLT) platform established under the Linux Foundation. In the business context, it is the most highly used private blockchain platform today [6]. Unfortunately, the advent of this new technology is rightfully accompanied by the realization of its potential cyber vulnerabilities and worries about its future implications for the stability of our economic system and economic transactions relying on this new technology. We can categorize HLF vulnerabilities into internal and external issues and attacks. Internal vulnerabilities are specific to the blockchain protocol and lie in configuration, consensus and endorsement, membership and access, and chaincode. In contrast, external vulnerabilities include physical machine attacks and endpoint security. Regarding HLF performance modeling and threat modeling, Place Transition Net (PN) models are taking a prominent role. To expand the HLF Petri Net modeling literature work, we first use Petri Nets to model HLF. Next, we develop a model of a cyber attack. Finally, we present a joint model of HLF and a cyber attack. In the context of the simulation environment we develop to mimic real-world situations, we investigate the joint behavior of effectively two joint systems and see how their implementations affect mutual performance.

Finally, when it comes to data, the successful application of machine learning models in various fields strongly depends on the data quality used for training. The ability to obtain additional data can improve a machine learning model's performance. Unfortunately, this could prove challenging, costly, or, in some cases, impossible. More specifically, leveraging additional data can be crucial for achieving satisfactory performance for applications involving classification tasks in the unbalanced setting and anomaly detection tasks via unsupervised and semi-supervised methods. Originally developed for image generation, diffusion models have shown promise in generating realistic synthetic tabular data. Emerging evidence points to the eventual triumph of diffusion models over generative adversarial networks analogous to what has happened in the computer vision field with image generation. This chapter explores the adaptation of recent open-source implementations of diffusion models for synthetic tabular data generation. We discuss the principles of diffusion models, their application to tabular data, and the challenges and future directions in this field in the context of cybersecurity and anomaly detection. We aim to provide insights and empirical evidence into the effectiveness of diffusion models for generating high-quality synthetic tabular datasets focusing on cyber and anomaly detection datasets for performance improvements through synthetic data augmentation in the tabular setting.

Chapter 1

SIZE DISTRIBUTION OF BOND PERCOLATION CLUSTERS ON FINITE GRAPHS AND INSURANCE PREMIUM

1.1 First and Second Moments of the Size Distribution of Bond Percolation Clusters on Finite Graphs

1.1.1 Introduction

The bond percolation cluster on a connected graph is a collection of independent Bernoulli random variables with the same success probability p indexed by the set of edges, with the edges associated to a success being referred to as open edges, and the ones associated to a failure being referred to as closed edges. The open cluster containing a vertex x is the random subset of vertices that are connected to vertex x by a path of open edges. Bond percolation is traditionally studied on infinite graphs such as the d-dimensional integer lattices, and the quantity of interest is the percolation probability, the probability that the cluster containing the origin is infinite [7, 8]. The original motivation was to determine the probability that the center of a porous stone immersed in water is wet, but bond percolation can more generally be seen as the simplest spatially explicit invasion model and now has a number of applications.

In this chapter, we are interested in bond percolation on finite graphs. In particular, all the open clusters are finite so whether the open cluster containing a given vertex is infinite or not becomes irrelevant. Instead, we investigate the size distribution of the open cluster containing a vertex chosen uniformly at random. Studying the size of a typical cluster is natural from a mathematical point of view but our main motivation originates from cyber insurance.

Due to the increasing use of smart technologies, our economy is now exposed to cyber risks caused by system failures, computer viruses, cyber attacks, etc. thus the need for customers to look for an insurance. Insurance premiums (the amount of money an individual or business must pay for an insurance policy) are computed based on the mean and variance of the aggregate loss distribution over a given period of time which, in turn, are typically estimated from past data. However, due to the lack of data in the context of cyber risk, insurance companies are unable for the moment to have a good idea of what their premiums should be. Motivated by this problem, Jectić and Lanchier [2] recently introduced a stochastic modeling framework to evaluate insurance premiums in the context of cyber risk. In their model, the system to be insured consist of a finite random graph equipped with a cost topology representing the network infrastructure of components that are attributed a certain monetary value. Losses occur at the times of a Poisson process with a fixed intensity, and each loss is modeled from a realization of bond percolation and quantified as the combined cost of all the components in the open cluster containing a component chosen uniformly at random. Using some conditionings, one can prove that the mean and variance of the aggregate loss, which are key quantities to compute insurance premiums, can be expressed using the first and second moments of the size of the open cluster containing a vertex chosen uniformly at random, the two quantities we study in this chapter.

1.1.2 Percolation Model for the Loss Distribution

To quantify the aggregate loss in a given time window, Jevtic and Lanchier [2, 3] introduced a class of stochastic models that consists of the combination of a Poisson process, independent realizations of a random graph equipped with a random cost topology, and independent realizations of bond percolation processes on these graphs. More precisely, the process can be constructed using the following components:

- a Poisson process (N_t) with intensity λ representing the times at which losses occur,
- a finite random graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ representing the network infrastructure,
- a probability $p \in (0, 1)$ representing the vulnerability of the network,
- a positive distribution C representing the cost distribution.

The aggregate loss $\mathscr{L}_t$ at time t is determined as follows. At the arrival times

$$T_i = \inf\{t : N_t = i\} \text{ for all } i = 1, 2, ...$$

of the Poisson process, we let

$$\mathscr{G}_i = (\mathscr{V}_i, \mathscr{E}_i) \text{ for all } i = 1, 2, ...$$

be independent (and identically distributed) realizations of the random graph $\mathscr{G}$, and assume that an infection resulting in a loss spreads at time T_i on the graph $\mathscr{G}_i$. To model the spread of the infection and quantify the loss, we use bond percolation with parameter p, i.e., we let

$$\xi_i(e) = \text{Bernoulli}(p) \text{ for all } e \in \mathscr{E}_i \text{ and } i = 1, 2, ...$$

be independent. As previously mentioned, edges with $\xi_i(e) = 1$ are referred to as open edges while the other edges are referred to as closed edges. Assuming in addition that the consecutive infections start at a vertex U_i chosen uniformly at random, i.e.,

$$U_i = \text{Uniform}(\mathscr{V}_i) \text{ for all } i = 1, 2, ...$$

the set of vertices that get infected at time T_i is

$$\mathscr{C}_i(U_i) = \{y \in \mathscr{V}_i : \text{ there is a path of open edges connecting } U_i \text{ and } y\},$$

the open cluster containing U_i. Finally, to quantify the financial or reputational loss resulting from the infections, we attach a random cost to each vertex by letting

$$\mathscr{C}_i(y) \text{ for all } y \in \mathscr{V}_i \text{ and } i = 1, 2, ...$$

be independent and identically distributed with distribution C. We then define the size of the infections and the losses resulting from the infections as

$$S_i = \text{card}(\mathscr{C}_i(U_i)) \text{ and } L_i = \sum_{y \in \mathscr{V}_i} C_i \mathbf{1}\{y \in \mathscr{C}_i(U_i) = \sum_{\mathscr{C}_i(U_i)} C_i(y)$$

the number of infected vertices and the cumulative cost of all these vertices. The aggregate loss at time t is the sum of the all losses by time t, i.e.,

$$\mathscr{L}_t = \sum_{i=1}^{N_t} L_i = \sum_{i=1}^{N_t} \sum_{y \in \mathscr{C}_i(U_i)} C_i$$

As previously mentioned, insurance premiums are computed from the mean and variance of the aggregate loss $\mathscr{L}_t$. Because the consecutive losses L_i are defined from independent and identically distributed random objects, they are themselves independent and identically distributed. In particular, to avoid cumbersome notations, we can drop all the subscripts i referring to the number of the infections. By conditioning on the number of infections, we get

$$E(\mathscr{L}_t) = E(E(\mathscr{L}_t | N_t)) = E(N_t E(L)) = E(N_t)E(L).$$

Using also independence and the law of total variance, we obtain

$$\begin{aligned} \text{Var}(\mathscr{L}_t) &= E(\text{Var}(\mathscr{L}_t)|N_t) + \text{Var}(E(\mathscr{L}_t)|N_t) \\ &= E(N_t \text{Var}(L)) + \text{Var}(N_t E(L)) \\ &= E(N_t)\text{Var}(L) + \text{Var}(N_t)(E(L))^2 \end{aligned}$$

Because $N_t = \text{Poisson}(\lambda t)$, we deduce that

$$E(\mathscr{L}_t) = \lambda t E(L) \quad \text{and} \quad \text{Var}(\mathscr{L}_t) = \lambda t \text{Var}(L) + \lambda t (E(L))^2 = \lambda t E(L^2) \tag{1.1}$$

showing that the mean and variance of the aggregate loss can be conveniently expressed using the first and second moments of the loss L resulting from a single infection. Similarly, conditioning on the size of the infection and using independence of the local costs, we get

$$\begin{aligned} E(L) &= E(E(L|S)) = E(SE(C)) = E(S)E(C) \\ E(L^2) &= E(E(L^2|S)) = E(SE(C^2) + S(S-1)(E(C)^2)) \\ &= E(S)E(C^2) + E(S^2)(E(C))^2 - E(S)(E(C))^2 \\ &= E(S)\text{Var}(C) + E(S^2)(E(C))^2 \end{aligned} \tag{1.2}$$

Combining (1.1) and (1.2), we deduce that

$$\begin{aligned} E(\mathscr{L}_t) &= \lambda t E(S)E(C) \\ \text{Var}(\mathscr{L}_t) &= \lambda t E(S)\text{Var}(C) + \lambda t E(S^2)(E(C))^2 \end{aligned} \tag{1.3}$$

which shows that computing the mean and variance of the aggregate loss in a given time window reduces to computing the first and second moments of the size distribution of the bond percolation cluster containing a vertex chosen uniformly at random. Studying the size distribution, however, is mathematically challenging due to the presence of spatial correlations: the fact that the states at different vertices (infected or not) are not independent. In particular, computing the first and second moments of the size of a percolation cluster is graph-specific. In the next section, we state our main results, focusing on four different classes of finite graphs.

1.1.3 Size Distribution of the Percolation Clusters

Motivated by (1.3) and (1.4), our main results focus on estimates for the first and second moments of the size distribution of bond percolation cluster on four classes of finite graphs: deterministic rings, deterministic paths, random stars, and random rooted trees with a fixed radius. The network infrastructures arising from smart technologies are typically more complex than these graphs. These complex structures, however, often consist of the composition of these simple graphs, so studying the four classes of graphs above is a good starting point.

Ring. To begin with, we focus on the ring with R edges (and R vertices). Because there are two self-avoiding paths connecting any two vertices, one needs to use the inclusion-exclusion identity to determine the probability of a path of open edges connecting a given vertex to the origin of the infection, but the calculations are simplified due to spherical symmetry.

Theorem 1 – *For the ring with R edges,*

$$\begin{aligned} E(S) &= \frac{1+p}{1-p} - \left(\frac{2}{1-p} + (R-1)\right) p^R \\ E(S^2) &= 1 + \frac{6p}{(1-p)^2} - \left[\left(\frac{6}{1-p}\right)\left(R + \frac{p}{1-p}\right) + (R-1)(2R-1)\right] p^R \end{aligned}$$

Path. We now look at the path with R edges (and $R+1$ vertices). In contrast with the ring, there is now a unique self-avoiding path connecting any two vertices, so computing the probability that a given vertex is connected to the origin of the infection by a path of open edges (and therefore infected) is simpler, but spherical symmetry is lost.

Theorem 2 – *For the path with R edges,*

$$\begin{aligned} E(S) &= \left(\frac{1+p}{1-p}\right) - \left(\frac{2p}{R+1}\right) \frac{1-p^{R+1}}{(1-p)^2} \\ E(S^2) &= 1 + \frac{6p}{(1-p)^2}\left[1 - \left(\frac{1+p}{R+1}\right)\left(\frac{1-p^{R+1}}{1-p}\right) + p^{R+1}\right] \end{aligned}$$

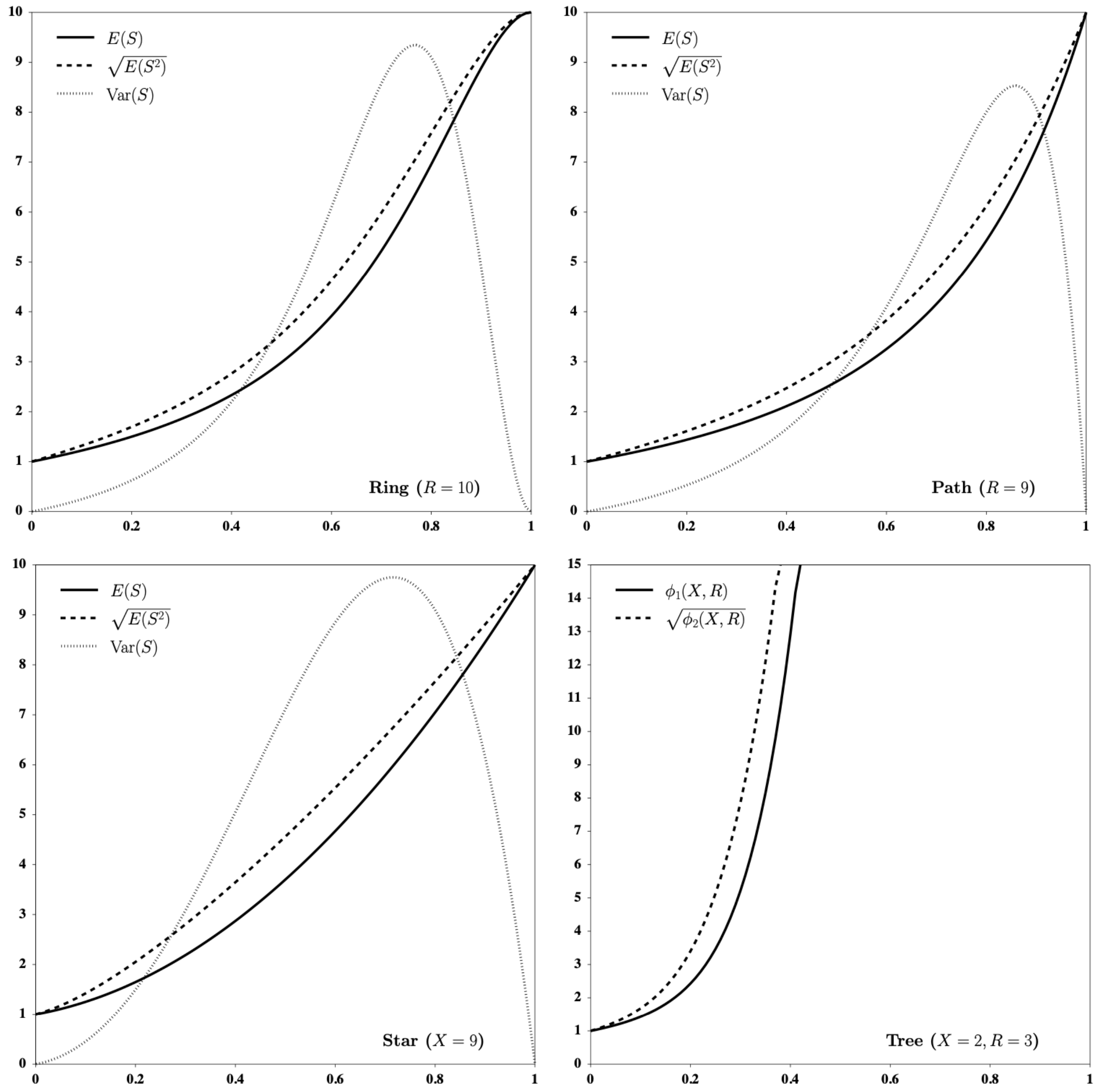


Figure 1. First Moment, Square Root of the Second Moment, and Variance of the Size of the Infection on the Ring, Path and (Deterministic) Star with 10 Vertices as a Function of p. The Last Figure Shows the Bound ϕ_1 and the Square Root of the Bound ϕ_2 given in Theorem 4 for the Binary Tree with Radius 3 and 1+2+4+8 = 15 Vertices.

Note that, for bond percolation on the integers, the open cluster containing the origin spans from vertex $-X_-$ to vertex X_+ where $X_\pm$ are two independent shifted geometric random variables (number of failures before the first success) with success probability $1-p$, the probability that an edge is closed. In particular, for bond percolation on the integers,

$$E(S) = E(1 + X_+ + X_-) = 1 + 2\Big(\frac{1}{1-p} - 1\Big) = \frac{2}{1-p} - 1 = \frac{1+p}{1-p}$$

which, in agreement with our results, corresponds to the (common) limit as $R \to \infty$ of the first moment in the two theorems. Similarly, for bond percolation on the integers,

$$\mathrm{Var}(S) = 2\mathrm{Var}(X_+) = \frac{2p}{(1-p)^2} \text{ and } E(S^2) = \frac{2p}{(1-p)^2} + \Big((\frac{1+p}{1-p})^2\Big) = 1 + \frac{6p}{(1-p)^2}$$

the (common) limit as $R \to \infty$ of the second moment in both theorems.

Star. The next graph is the random star with X edges (and $X+1$ vertices) where X is a positive random variable with the probability mass function $(p_k)_{k=1}^{\infty}$ and finite mean and variance

$$\mu = E(X) = \sum_{k=0}^{\infty} kp_k < \infty \text{ and } \sigma^2 = \mathrm{Var}(X) = \sum_{k=0}^{\infty} (k-\mu)^2 p_k < \infty$$

As for the path, there is a unique self-avoiding path connecting any two vertices, which facilitates the analysis. Looking separately at the size of the open clusters containing the center and containing a leaf, and using spherical symmetry, we get the following theorem.

Theorem 3 – *For the (random) star with X branches,*

$$\begin{aligned} E(S) &= 1 + 2E\left(\frac{X}{X+1}\right)p + E\left(\frac{X(X-1)}{X+1}\right)p^2 \\ E(S^2) &= 1 + 6E\left(\frac{X}{X+1}\right)p + 6E\left(\frac{X(X-1)}{X+1}\right)p^2 + E\left(\frac{X(X-1)(X-2)}{X+1}\right)p^3 \end{aligned}$$

Tree. Finally, we study the size of the bond percolation clusters on the random rooted tree with offspring distribution X and radius R. To construct this random graph, we draw k edges with probability p_k starting from a root, and additional edges starting from each of the subsequent vertices using the same probability distribution. The construction stops after R steps, or generations, so the tree has radius R. Bond percolation of this graph has already been studied in [2] where it is proved, using combinatorial arguments to count the number of self-avoiding paths of a given length starting from a given vertex, that the conditional mean is equal to

$$E_r(S) = \frac{1}{1-\mu p}\Big(1 + p(1-p^r) - \frac{(\mu p)^{R-r+1}(1-p^2(1+(\mu-1)(\mu p^2)^r))}{1-\mu p^2}\Big)$$

for all $0 \leq r \leq R$. Here, the subscript r refers to the conditional mean given that the infection starts at distance r from the root. See [2, Theorem 4] with $p_0 = 0$. In addition, uniformly in all possible realizations of the random tree, we have the following upper bounds for the conditional variance and the conditional second moment:

$$\mathrm{Var}_r(S) \leq (p^{-2R} - 1)(E_r(S))^2 \text{ therefore } E_r(S^2) \leq (p^{-R} E_r(S))^2.$$

See [2, Theorem 5]. Motivated by the lack of information about the origin of the infection, rather than computing conditional quantities that might not be relevant for insurance premiums, we study their unconditional counterparts when the infection starts from a vertex chosen uniformly at random, just like we did for the other three graphs. Using coupling arguments to compare the open cluster of bond percolation with a certain branching process, we obtain the following upper bounds for the unconditional first and second moments.

Theorem 4 – *For the (random) rooted tree with radius R,*

$$E(S) \leq \phi_1(X,R) = \frac{1-\nu_+^{2R+1}}{1-\nu_+}$$
$$E(S^2) \leq \phi_2(X,R) = \frac{\Sigma_+^2}{(1-\nu_+)^2}\Big(\frac{1-\nu^{4R+1}}{1-\nu_+} - (4R+1)\nu_+^{2R}\Big) + \Big(\frac{1-\nu_+^{2R+1}}{1-\nu_+}\Big)^2$$

where $\nu_+ = p(\mu+1)$ *and* $\Sigma_+^2 = p(1-p)(\mu+1) + p^2\sigma^2$.

Based on their probabilistic interpretation, the upper bounds ϕ_1 and ϕ_2 in the theorem must be increasing with respect to the radius. In particular, the theorem immediately implies that, when $\nu_+ < 1$, we have the following (simpler) upper bounds that are uniform in the radius:

$$E(S) \leq \lim_{R\to\infty}\phi_1(X,R) = \frac{1}{1-\nu_+}$$
$$E(S^2) \leq \lim_{R\to\infty}\phi_2(X,R) = \frac{\Sigma_+^2}{(1-\nu)^3} + \frac{1}{(1-\nu_+)^2}$$

Figure 1 shows the first moment, the square root of the second moment, and the variance of the size distribution computed from the first three theorems as a function of the probability p. The last panel shows the upper bounds for the first moment and the square root of the second moment in the last theorem as a function of p. The reason for looking at the square root of the second moment is just a convenient choice to plot the functions as the first moment and the square root of the second moment both range from one when $p = 0$ to the number of vertices when $p = 1$. The rest of the section is devoted to proofs. The next section explains how the moments of the size distribution of a random percolation cluster are related to the (joint) probability of open paths connecting pairs of vertices. These results hold for all graphs and are applied in the three subsequent sections to rings, paths, and stars in order to prove the first three theorems. The proof of the last theorem relies in addition on the properties of a certain branching process and coupling arguments to compare this branching process to the percolation clusters.

1.1.4 Preliminary Results

To begin with, we collect a few general results that will be used repeatedly across the section regardless of the graph under consideration.

Having a realization $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ of the random graph and a vertex $x \in \mathscr{V}$, probabilities, expected values, and variances with a subscript x refer to conditional counterparts given that the infection starts at vertex $U = x$. Having another vertex $y \in \mathscr{V}$, we write $x \leftrightarrow y$ to indicate that the two vertices are connected by a path of open edges. That is, there exist

$$\{x_0, x_1, x_2, ..., x_n \subset \mathscr{V}\} \text{ with } x_0 = x \text{ and } x_n = y$$

such that, for all $j = 0, 1, ..., n - 1$,

$$(x_j, x_{j+1}) \in \mathscr{E} \text{ and } \xi((x_j, x_{j+1})) = 1$$

On the event that the infection starts at vertex $U = x$, we can keep track of the state (infected or not) at each vertex by defining the function

$$\zeta : \mathscr{V} \to \{0, 1\} \text{ with } \zeta(y) = \begin{cases} 1 & \text{if } y \in \mathscr{C}(x) \\ 0 & \text{if } y \notin \mathscr{C}(x) \end{cases}$$

Observing that $y \in \mathscr{C}(x)$, meaning that y is infected, if and only if there is a path of open edges connecting x, the origin of the infection, and vertex y, we deduce that the conditional kth moment of the size distribution can be expressed using the probability of open paths as

$$\begin{aligned} E_x(S^K) = E_x\Big(\sum_{y \in \mathscr{V}} \zeta(y)\Big)^k &= E_x\Big(\sum_{X \in \mathscr{V}^k} \zeta(X_1) \cdots \zeta(X_k)\Big) \\ = \sum_{X \in \mathscr{V}^k} P_x(\zeta(X_1) = 1 \text{ for all } i) &= \sum_{X \in \mathscr{V}^k} P(x \leftrightarrow X_i \text{ for all } i) \end{aligned} \tag{1.4}$$

Taking $k = 1, 2$, and conditioning on U, we get

$$E(S) = \sum_{x \in \mathscr{V}} E_x(S) P(U = x) = \frac{1}{\text{card}(\mathscr{V})} \sum P(x \leftrightarrow y) \tag{1.5}$$

while the second moment can be expressed as

$$E(S^2) = \sum_{x \in \mathscr{V}} E_x(S^2) P(U = x) = \frac{1}{\text{card}(\mathscr{V})} \sum P(x \leftrightarrow y, x \leftrightarrow z) \tag{1.6}$$

1.1.5 Proof of Theorem 1

In this section, $\mathscr{G}$ is the ring with R edges, and we write $\mathscr{V} = \{0, 1, ..., R-1\}$. The probability that two given vertices are connected by an open path is slightly more difficult to compute for this graph than for the path, the star and the tree because there are two (rather than one) self-avoiding paths connecting the two vertices. The analysis, however, is simpler overall because of the obvious symmetry of the graph that implies that

$$E(S^k) = \sum_{x=0}^{R-1} E_x(S^k) P(U = x) = \frac{1}{\text{card}(\mathscr{V})} \sum_{x=0}^{R-1} E_x(s^k) = E_0(S^k) \tag{1.7}$$

The following two lemmas give respectively the first and second moments of the size distribution of the cluster of open edges containing vertex zero.

Lemma 5 – *For all* $p \in (0, 1)$,

$$E_0(S) = \frac{1+p}{1-p} - \Big(\frac{2}{1-p} + (R-1)\Big) p^R$$

Proof. Let $y \in \mathscr{V}$. As previously mentioned, there are exactly two self-avoiding paths going from zero to y, and we let A_1 and A_2 be the events that the path is going clockwise.counter-clockwise is open. Because each edge is independently open with probability p,

$$\begin{aligned} P(0 \leftrightarrow y) &= P(A_1 \cup A_2) \\ &= P(A_1) + P(A_2) - P(A_1 \cap A_2) = p^{d(0,y)} + p^{R-d(0,y)} - p^R \end{aligned} \tag{1.8}$$

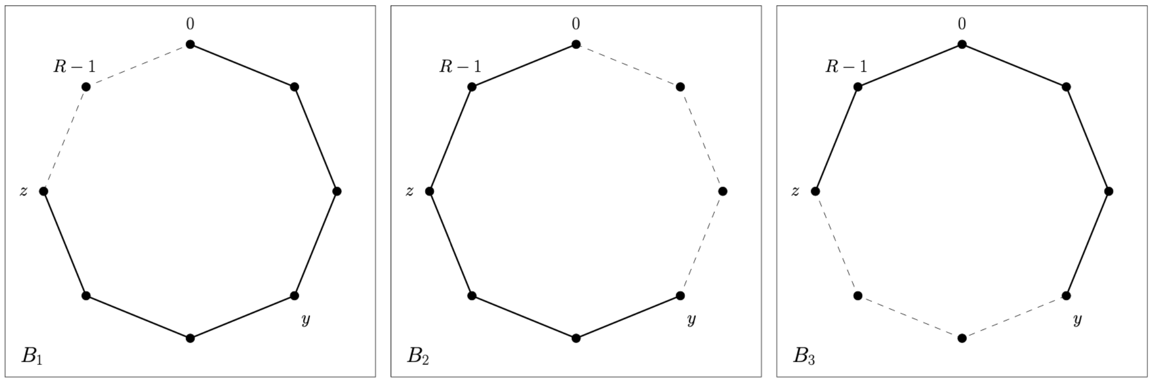


Figure 2. Illustration of the Events B_1, B_2, B_3 in (12). The Solid Lines Represent the Edges That Must Be Open Whereas the Dashed Lines Represent the Edges That Can Be Either Open or Closed.

Combining (4) and (8), we deduce that

$$E_0(S) = \sum_{y \in \mathscr{V}} P(0 \leftrightarrow y) = \sum_{y=0}^{R-1} (p^y + p^{R-y} - p^R)$$

Then, using symmetry, we conclude that

$$\begin{aligned} E_0(S) = 1 = \sum_{y=1}^{R-1} (p^y + p^{R-y} - p^R) &= 1 + \sum_{y=1}^{R-1} (2p^y) - (R-1)p^R \\ = 1 + 2p\Big(\frac{1-p^{R-1}}{1-p}\Big) - (R-1)p^R &= \Big(\frac{1+p}{1-p}\Big) - \Big(\frac{2}{1-p} + (R-1)\Big)p^R \end{aligned} \tag{1.9}$$

This completes the proof. $\square$

Lemma 6 – *For all* $p \in (0,1)$,

$$E_0(S^2) = 1 + \frac{6p}{(1-p)^2} - \Big[\Big(\frac{6}{1-p}\Big)\Big(R + \frac{p}{1-p} + (R-1)(2R-1)\Big)\Big]p^R$$

Proof. According to (1.4), the second moment is

$$E_0(S^2) = \sum_{y,z \in \mathscr{V}} P(0 \leftrightarrow y, 0 \leftrightarrow z)$$

Decomposing depending on whether card$\{0, y, z\} = 1, 2, 3$, we get

$$E_0(S^2) = 1 + 3\Big(\sum_{y \neq 0} P(0 \leftrightarrow y)\Big) + 2\Big(\sum_{0<y<z} P(0 \leftrightarrow y, 0 \leftrightarrow z)\Big) \tag{1.10}$$

Applying Lemma 5, the first sum in (1.10) is

$$\begin{aligned}\sum_{y\neq 0} P(0 \leftrightarrow y) = E_0(S) - 1 &= \Big(\frac{1+p}{1-p}\Big) - \Big(\frac{2}{1-p} + (R-1)\Big)p^R - 1 \\ &= \Big(\frac{2p}{1-p}\Big) - \Big(\frac{2}{1-p} + (R-1)\Big)p^R\end{aligned} \tag{1.11}$$

To compute the last sum in (1.10), let $0 < y < z$, and define the events

$$\begin{aligned} B_1 &= \text{ path } (0, 1, ..., y, ..., z-1, z) \text{ is open,} \\ B_2 &= \text{ path } (y, y+1, ..., z, ..., R-1, 0) \text{ is open,} \\ B_3 &= \text{ path } (z, z+1, ..., 0, ..., y-1, y) \text{ is open,}\end{aligned} \tag{1.12}$$

See Figure 2 above. Observe that $0 \leftrightarrow y$ and $0 \leftrightarrow z$ is and only if at least one of the three events above occurs which, applying the inclusion-exclusion identity, gives

$$\begin{aligned} P(0 \leftrightarrow y, 0 \leftrightarrow z) &= P(B_1) + P(B_2) + P(B_3) - P(B_1 \cap B_2) \\ &\quad - P(B_2 \cap B_3) - P(B_3 \cap B_1) + P(B_1 \cap B_2 \cap B_3) \\ &= p^z + p^{R-y} + p^{R+y-z} - p^R - p^R - p^R + p^R \\ &= p^z + p^{R-y} + p^{R+y-z} - 2p^R \end{aligned} \tag{1.13}$$

Summing over all $0 < y < z$, we get

$$\begin{aligned} \sum_{0<y<z} p^z = \sum_{y=1}^{R-2} \sum_{z=y+1}^{R-1} p^z = \sum_{y=1}^{R-2} \Big(\frac{1-p^{R-y-1}}{1-p}\Big) \\ = \frac{1}{1-p} \sum_{y=1}^{R-2} (p^{y+1} - p^R) = \frac{1}{1-p}\Big[p^2\Big(\frac{1-p^{R-2}}{1-p}\Big) - (R-2)p^R\Big] \end{aligned} \tag{1.14}$$

Similarly, we prove that

$$\sum_{0<y<z} p^{R-y} = \sum_{0<y<z} p^{R+y-z} = \frac{1}{1-p}\Big[p^2\Big(\frac{1-p^{R-2}}{1-p}\Big) - (R-2)p^R\Big] \tag{1.15}$$

while for the last term in (1.13),

$$\sum_{0<y<z} 2p^R = 2\binom{R-1}{2} p^R = (R-1)(R-2)p^R \tag{1.16}$$

Combining (1.13)-(1.16), we deduce that

$$\sum_{0<y<z} P(0 \leftrightarrow y, 0 \leftrightarrow z) = \frac{3}{1-p}\Big[p^2\Big(\frac{1-p^{R-2}}{1-p}\Big) - (R-2)p^R\Big] - (R-1)(R-2)p^R$$

then using also (1.10) and (1.11),

$$\begin{aligned} E_0(S^2) = 1 &+ \Big(\frac{6p}{1-p}\Big) - \Big(\frac{6}{1-p} + 3(R-1)\Big)p^R \\ &+ \Big(\frac{6}{1-p}\Big)\Big[p^2\Big(\frac{1-p^{R-2}}{1-p}\Big) - (R-2)p^R\Big] - 2(R-1)(R-2)p^R \end{aligned}$$

Rearranging the terms and simplifying, we conclude that

$$\begin{aligned} E_0(S^2) &= 1 + \Big(\frac{6p}{1-p} + \frac{6p^2}{(1-p)^2}\Big) - \Big(3(R-1) + 2(R-1)(R-2)\Big)p^R \\ &\qquad - \Big(\frac{6}{1-p}\Big)\Big(1 + \frac{1}{1-p} + R - 2\Big)p^R \\ &= 1 + \frac{6p}{(1-p)^2} - \Big[\Big(\frac{6}{1-p}\Big)\Big(R + \frac{p}{1-p}\Big) + (R-1)(2R-1)\Big]p^R \end{aligned}$$

This completes the proof. $\square$

Theorem 1 immediately follows from (1.7) and Lemmas 5 and 6.

1.1.6 Proof of Theorem 2

We now assume that $\mathscr{G}$ is the path with R edges, and we write $\mathscr{V} = \{0, 1, ..., R\}$. To compute the unconditional first and second moments, we start by computing their conditional counterparts given that the infection starts at $x \in \mathscr{V}$ using (1.4). The next lemma focuses on the first moment and shows the first part of the theorem.

Lemma 7 – *For all* $p \in (0,1)$,

$$E(S) = \Big(\frac{1+p}{1-p}\Big) - \Big(\frac{2p}{R+1}\Big)\Big(\frac{1-p^{R+1}}{(1-p)^2}\Big)$$

PROOF. Let $x \in \mathscr{V}$. Due to the absence of cycle symmetry, we now have

$$P(x \leftrightarrow y) = p^{d(x,y)} \text{ for all } y \in \mathscr{V}$$

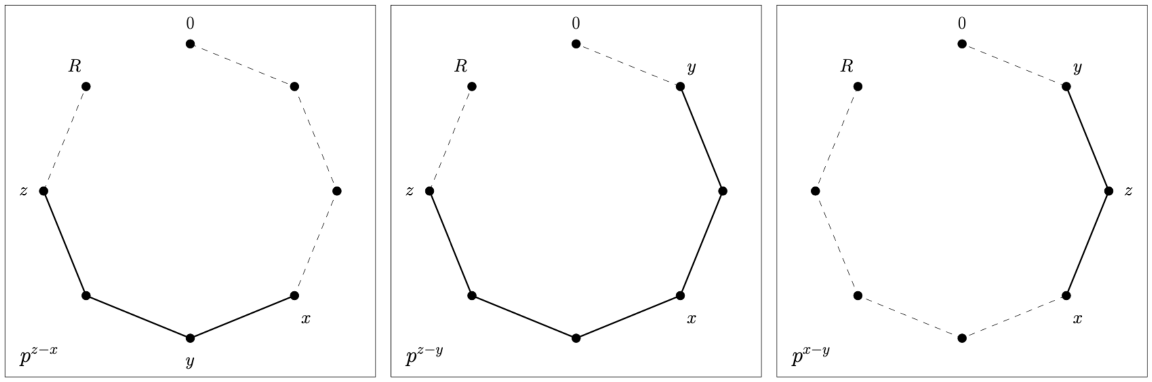


Figure 3. Illustration of the Three Orderings in (18). The Solid Lines Represent the Edges That Must Be Open Whereas the Dashed Lines Represent the Edges That Can Be Either Open or Closed.

This together with (1.4), implies that

$$\begin{aligned}E_x(S) &= \sum_{y\in\mathscr{V}} p^{d(x,y)} = 1 + \sum y = 0^{x-1} p^{x-y} + \sum_{y=x+1}^{R} p^{y-x} \\ &= 1 + \sum_{y=1}^{x} p^y + \sum_{y=1}^{R-x} p^y = 1 + p\Big(\frac{1-p^x}{1-p}\Big) + p\Big(\frac{1-p^{R-x}}{1-p}\Big)\end{aligned} \tag{1.17}$$

Note that the previous equation also holds for $x = 0$ and $x = R$. In particular, combining (5) and (17), we deduce that the first moment is given by

$$\begin{aligned}E(S) &= \frac{1}{R+1}\sum_{x=0}^{R}\Big(1 + p\Big(\frac{1-p^x}{1-p}\Big) + p\Big(\frac{1-p^{R-x}}{1-p}\Big)\Big) \\ &= 1 + \Big(\frac{2p}{1-p}\Big) - \Big(\frac{1}{R+1}\Big)\Big(\frac{p}{1-p}\Big)\sum_{x=0}^{R}(p^x + p^{R-x})\end{aligned}$$

Using some evident symmetry, and simplifying,

$$\begin{aligned}E(S) &= 1 + \Big(\frac{2p}{1-p}\Big)\Big[1 - \Big(\frac{1}{R+1}\Big)\sum_{x=0}^{R} p^x\Big] \\ &= 1 + \Big(\frac{2p}{1-p}\Big)\Big[1 - \Big(\frac{1}{R+1}\Big)\Big(\frac{1-p^{R+1}}{1-p}\Big)\Big] = \Big(\frac{1+p}{1-p}\Big) - \Big(\frac{2p}{R+1}\Big)\Big(\frac{1-p^{R+1}}{(1-p)^2}\Big)\end{aligned}$$

This completes the proof. $\square$

We now look at the second moment of the size distribution.

Lemma 8 – *For all* $p \in (0,1)$,

$$E(S^2) = 1 + \frac{6p}{(1-p)^2}\Big[1 - \Big(\frac{1+p}{R+1}\Big)\Big(\frac{1-p^{R+1}}{1-p} + p^{R+1}\Big)\Big]$$

PROOF. Note that, for all $x, y, x \in \mathscr{V}$,

$$P(x \leftrightarrow y, x \leftrightarrow z) = \begin{cases} p^{d(x,z)} = p^{z-x} & \text{when } x \le y \le z, \\ p^{d(y,z)} = p^{z-y} & \text{when } y \le x \le z, \\ p^{d(x,y)} = p^{x-y} & \text{when } y \le z \le x \end{cases} \tag{1.18}$$

See Figure 3 for an illustration of these three cases. In particular, partitioning based on the possible orderings of the vertices, accounting for symmetry, and using (4), we deduce that

$$E_x(S^2) = 2\Big(\sum_{x\le y\le z} p^{z-x} + \sum_{y\le z\le x} p^{x-y} + \sum_{y\le x\le z} p^{z-y}\Big) + \Big(\sum_{y\le x} p^{x-y} + \sum_{x\le z} p^{z-x}\Big) \tag{1.19}$$

For all $x < R$, the first sum in (1.19) is

$$\begin{aligned} \sum_{x\le y\le z} p^{z-x} &= \sum_{y=x}^{R-1}\sum_{z=y+1}^{R} p^{z-x} = \sum_{y=x}^{R-1} p^{y-x+1}\Big(\frac{1-p^{R-y}}{1-p}\Big) \\ &= \frac{p}{1-p}\sum_{R-1}^{y=x}(p^{y-x} - p^{R-x}) = \frac{p}{1-p}\Big(\frac{1-p^{R-x}}{1-p} - (R-x)p^{R-x}\Big) \end{aligned} \tag{1.20}$$

The left-hand side and right-hand side are both equal to zero when $x = R$ therefore (20) also holds in this case. Similarly, the second sum in (19) is given by

$$\sum_{y\le x\le z} p^{z-y} = \sum_{z=1}^{x}\sum_{y=0}^{z-1} p^{x-y} = \frac{p}{1-p}\Big(\frac{1-p^x}{1-p} - xp^x\Big) \tag{1.21}$$

for all $x \in \mathscr{V}$. The third sum in (1.19) is

$$\begin{aligned} \sum_{y\le x\le z} p^{z-y} &= \sum_{y=0}^{x-1}\sum_{z=x+1}^{R} p^{z-y} = \sum_{y=0}^{x-1} p^{x-y+1}\Big(\frac{1-p^{R-x}}{1-p}\Big) \\ &= \Big(\frac{1-p^{R-x}}{1-p}\Big)\sum_{y=2}^{x+1} p^y = p^2\Big(\frac{1-p^{R-x}}{1-p}\Big)\Big(\frac{1-p^x}{1-p}\Big) \end{aligned} \tag{1.22}$$

for all $x \in \mathscr{V}$. Finally, the last two sums are given by

$$\sum_{y \leq x} p^{x-y} = \frac{1-p^{x+1}}{1-p} = 1 + p\Big(\frac{1-p^x}{1-p}\Big) \text{ and } \sum_{x<z} p^{z-x} = p\Big(\frac{1-p^{R-x}}{1-p}\Big) \tag{1.23}$$

Combining (1.19)-(1.23), we deduce that

$$\begin{aligned} E_x(S^2) = 1 &+ \Big(p\frac{2p}{1-p}\Big)\Big(\frac{1-p^x}{1-p} + \frac{1-p^{R-x}}{1-p}\Big) \\ &+ 2p^2\Big(\frac{1-p^x}{1-p}\Big)\Big(\frac{1-p^{R-x}}{1-p}\Big) - \Big(\frac{2p}{1-p}\Big)(xp^x + (R-x)p^{R-x}) \end{aligned}$$

Then, using (1.6), we get

$$\begin{aligned} E(S^2) = 1 &+ \Big(p + \frac{2p}{1-p}\Big)\Big(\frac{1}{1-p}\Big)\Big[2 - \Big(\frac{1}{R+1}\Big)\sum_{x=0}^{R}(p^x + p^{R-x})\Big] \\ &+ 2\Big(\frac{p}{1-p}^2\Big[(1+p^R) - \Big(\frac{1}{R+1}\Big)\sum_{x=0}^{R}(p^x + p^{R-x})\Big]\Big) \\ &- \Big(\frac{2p}{1-p}\Big)\Big(\frac{1}{R+1}\Big)\sum_{x=0}^{R}(xp^x + (R-x)p^{R-x}) \end{aligned} \tag{1.24}$$

Observe also that

$$\begin{aligned} \sum_{x=0}^{R} xp^x &= p\sum_{x=0}^{R}\frac{\partial(p^x)}{\partial p} = p\frac{\partial}{\partial p}\Big(\sum_{x=0}^{R}\Big) = p\frac{\partial}{\partial p}\Big(\frac{1-p^{R+1}}{1-p}\Big) \\ &= p\Big(\frac{Rp^{R+1} - (R+1)p^R + 1}{(1-p)^2}\Big) = \frac{p}{1-p}\Big[\Big(\frac{1-p^{R+1}}{1-p}\Big) - (R+1)p^R\Big] \end{aligned} \tag{1.25}$$

Combining (1.24) and (1.25), and using symmetry,

$$\begin{aligned} E(S^2) = 1 &+ \Big(p + \frac{2p}{1-p}\Big)\Big(\frac{2}{1-p}\Big)\Big[1 - \Big(\frac{1}{R+1}\Big)\Big(\frac{1-p^{R+1}}{1-p}\Big)\Big] \\ &+ 2\Big(\frac{2}{1-p}\Big)^2\Big[(1+p^R) - \Big(\frac{2}{R+1}\Big)\Big(\frac{1-p^{R+1}}{1-p}\Big)\Big] \\ &+ \Big(\frac{2p}{1-p}\Big)^2\Big[p^R - \Big(\frac{1}{R+1}\Big)\Big(\frac{1-p^{R+1}}{1-p}\Big)\Big] \end{aligned}$$

then simplifying, we conclude that

$$E(S^2) = 1 + \frac{6p}{(1-p)^2}\Big[1 - \Big(\frac{1+p}{R+1}\Big)\Big(\frac{1-p^{R+1}}{1-p}\Big) + p^{R+1}\Big]$$

This completes the proof. $\square$

1.1.7 Proof of Theorem 3

In this section, we study the size distribution of the clusters on the random star with X branches, where X is a positive integer-valued random variable. We write $\mathscr{V} = \{0, 1, ..., X\}$ with vertex 0 referring to the center of the star. Because the center plays a special role, to find the moments of the size distribution, we first look at their conditional counterparts given that the infection starts from the center, and given that the infection starts from a leaf, respectively.

Lemma 9 – *For all $k \geq 1$ and $p \in (0,1)$,*

$$E_0(S|X=k) = 1 + kp \quad \textit{and} \quad E_0(S^2|X=k) = 1 + 3kp + k(k-1)p^2$$

Proof. When the infection starts at the center, the probability that leaf y gets infected is the probability p that edge $(0,y)$ is open. This and (4) imply that

$$E_0(S|X=k) = \sum_{y=0}^{k} P(0 \leftrightarrow y) = 1 \sum_{y=1}^{k} P(0 \leftrightarrow y) = 1 + kp$$

To deal with the second moment, observe that

$$P(0 \leftrightarrow y, 0 \leftrightarrow z) = \begin{cases} p^0 & \text{when } \operatorname{card}\{0,y,z\} = 1, \\ p^1 & \text{when } \operatorname{card}\{0,y,z\} = 2, \\ p^2 & \text{when } \operatorname{card}\{0,y,z\} = 3 \end{cases} \tag{1.26}$$

Using again (1.4), and decomposing based on the cardinal,

$$\begin{aligned} E_0(S^2|X=k) &= \sum_{y,z \in \mathscr{V}} P(0 \leftrightarrow y, 0 \leftrightarrow z) \\ &= P(0 \leftrightarrow 0) + 3\Big(\sum_{y \neq 0} P(0 \leftrightarrow y)\Big) + 2\Big(\sum_{0<y<z} P(0 \leftrightarrow y, 0 \leftrightarrow z)\Big) \end{aligned} \tag{1.27}$$

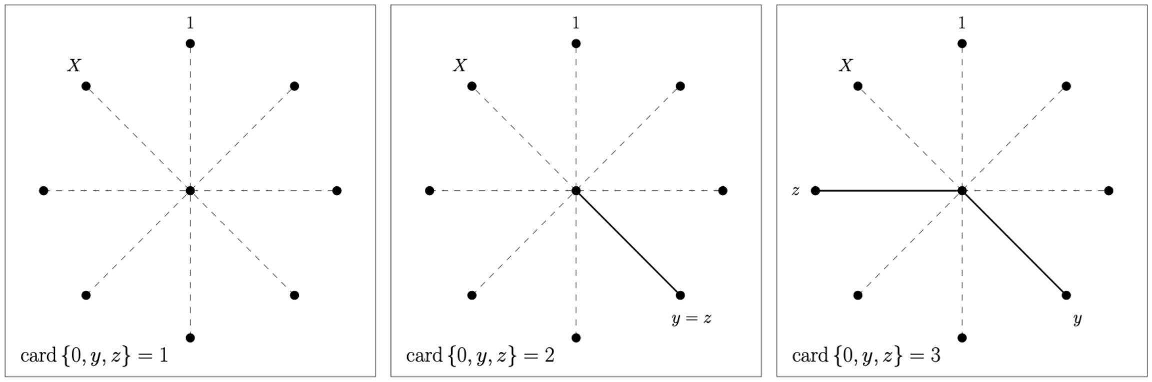


Figure 4. Illustration of the Three Cases in (26). The Solid Lines Represent the Edges That Must Be Open Whereas the Dashed Lines Represent the Edges That Can Be Either Open or Closed.

Counting the number of vertices and combining (1.26) and (1.27),

$$\begin{aligned} E_0(S^2|X=k) &= P(0 \leftrightarrow 0) + 3kP(0 \leftrightarrow 1) + k(k-1)P(0 \leftrightarrow 1, 0 \leftrightarrow 2) \\ &= 1 + 3kp + k(k-1)p^2 \end{aligned}$$

This completes the proof. □

We now look at the size distribution given that the infection starts from a leaf $x \neq 0$.

Lemma 10 – *For all $k \geq 1$ and $p \in (0,1)$,*

$$E_x(S) = 1 + p + (k-1)p^2$$
$$E_x(S^2) = 1 + 3p + 5(k-1)p^2 + (k-1)(k-2)p^3$$

Proof. When the infection starts at $x \neq 0$, the probability that a leaf $y \neq x$ gets infected is the probability p^2 that both $(0,x)$ and $(0,y)$ are open. This and (4) imply that

$$E_x(S|X=k) = \sum_{y=0}^{k} P(x \leftrightarrow y) = 1 + p + \sum_{y \neq 0,x} P(x \leftrightarrow y) = 1 + p + (k-1)p^2$$

Now, for $y = 0$ and/or $z = 0$,

$$P(x \leftrightarrow y, x \leftrightarrow z) = \begin{cases} p^1 & \text{when } \operatorname{card}\{x, y, z\} = 2, \\ p^2 & \text{when } \operatorname{card}\{x, y, z\} = 3 \end{cases} \tag{1.28}$$

while for all $y, z \neq 0$,

$$P(x \leftrightarrow y, x \leftrightarrow z) = \begin{cases} p^0 & \text{when } \operatorname{card}\{x, y, z\} = 1, \\ p^2 & \text{when } \operatorname{card}\{x, y, z\} = 2, \\ p^3 & \text{when } \operatorname{card}\{x, y, z\} = 3 \end{cases} \tag{1.29}$$

In addition, using (1.4) and spherical symmetry, we have

$$\begin{aligned} E_x(S^2|X = k) = 3P(1 \leftrightarrow 0) + 2(k-1)P(1 \leftrightarrow 0, 1 \leftrightarrow 2) + P(1 \leftrightarrow 1) \\ + 3(k-1)P(1 \leftrightarrow 2) + (k-1)(k-2)P(1 \leftrightarrow 2, 1 \leftrightarrow 3) \end{aligned} \tag{1.30}$$

Combining (1.28)-(1.30), and simplifying, we get

$$E_x(S^2|X = k)1 + 3p + 5(k-1)p^2 + (k-1)(k-2)p^3$$

This completes the proof. $\square$

Combining Lemmas 9 and 10, we can now compute the unconditional first and second moments given in the theorem when the infection starts at a vertex chosen uniformly at random. The two parts of the theorem are proved in the following two lemmas, respectively.

Lemma 11 – *For all* $p \in (0, 1)$*,*

$$E(S) = 1 + 2E\Big(\frac{X}{X+1}\Big)p + E\Big(\frac{X(X-1)}{X+1}\Big)p^2$$

Proof. Conditioning on the origin $U = \text{Uniform}(\mathscr{V})$, we get

$$E(S|X) = \sum_{x=0}^{X} E(S|X, U = x)P(U = x) = \frac{1}{X+1}\sum_{x=0}^{X} E_x(S|X)$$

Then, applying Lemmas 9 and 10, we deduce that

$$\begin{aligned} E(S|X) &= \frac{E_0(S|X) + XE_1(S|X)}{X+1} = \frac{1+Xp}{X+1} + \frac{X(1+p+(X-1)p^2)}{X+1} \\ &= 1 + 2\Big(\frac{X}{X+1}\Big)p + \Big(\frac{X(X-1)}{X+1}\Big)p^2 \end{aligned}$$

Taking the expected value on both sides gives the result. □

Lemma 12 – *For all $p \in (0,1)$,*

$$E(S^2) = 1 + 6E\Big(\frac{X}{X+1}\Big)p + 6E\Big(\frac{X(X-1)}{X+1}\Big)p^2 + E\Big(\frac{X(X-1)(X-2)}{X+1}\Big)p^3$$

Proof. Proceeding as in the proof of Lemma 11, we get

$$E(S^2|X) = \frac{E_0(S^2|X) + XE_1(S^2|X)}{X+1}$$

Using again Lemmas 9 and 10, we deduce that

$$\begin{aligned} E(S^2|X) &= \frac{1+3Xp+X(X-1)p^2}{X+1} \\ &+ \frac{X(1+3p+5(X-1)p^2+(X-1)(X-2)p^3)}{X+1} \\ &= 1 + 6\Big(\frac{X}{X+1}\Big)p + 6\Big(\frac{X(X-1)}{X+1}\Big)p^2 + \Big(\frac{X(X-1)(X-2)}{X+1}\Big)p^3 \end{aligned}$$

which gives the result. □

1.1.8 Proof of Theorem 4

1.1.8.1 Number of Individuals in a Branching Process

This section is devoted to collecting preliminary results about branching processes that will be used later to prove Theorem 4. More precisely, we compute the first and second moments of the number of individuals up to a given generation in a

Galton-Watson branching process (X_n) with offspring distribution $Y_{n,i}$, i.e, the process starts with $X_0 = 1$ individual and

$$X_{n+1} = Y_{n,1} + Y_{n,2} + \cdots Y_{n,X_n} \text{ for all } n \geq 0 \tag{1.31}$$

where the random variables $Y_{n,i}$ are independent and identically distributed with finite mean and finite variance. The main objective is to study

$$\bar{X}_R = X_0 + X_1 + \cdots + X_R,$$

the cumulative number of individuals up to generation R. The first and second moments of this random variable can be conveniently expressed using the mean and variance

$$E(Y_{n,i}) = \nu < \infty \text{ and } \mathrm{Var}(Y_{n,i}) = \Sigma^2 < \infty$$

The next lemma gives the first moment.

Lemma 13 – *For all $\nu \neq 1$,*

$$E(\hat{X}_R) = \sum_{n=0}^{R} \nu^n = \frac{1 - \nu^{R+1}}{1 - \nu}$$

PROOF. We proceed by induction. The lemma clearly holds for $R = 0$. Now, assume that the result holds up to generation R. It follows from (31) that

$$\begin{aligned} E(X_{n+1}) &= E(E(X_{n+1}|X_n)) = E(E(Y_{n,1} + Y_{n,2} + \cdots Y_{n,X_n}|X_n)) \\ &= E(X_n E(Y_{n,i})) = E(X_n)E(Y_{n,i}) = \nu E(X_n) \end{aligned} \tag{1.32}$$

In particular, an inductive argument gives

$$E(X_n) = \nu E(X_{n-1}) = \nu^2 E(X_{n-2}) = \cdots = \nu^n E(X_0) = \nu^n, \tag{1.33}$$

from which we deduce that

$$E(\bar{X}_{R+1}) = E(\bar{X}_R + X_{R+1}) = \Big(\sum_{n=0}^{R} \nu^n\Big) + \nu^{R+1} = \sum_{n=0}^{R+1} \nu^n$$

This completes the proof. $\square$

We now compute the second moment of the number of individuals at a given generation as well as the expected value of the product of the number of individuals at two different generations. These two quantities will be used to compute the second moment of $\bar{X}_R$.

Lemma 14 – *For all $\nu \neq 1$ and $\Sigma^2 > 0$,*

$$E(X_n^2) = \Sigma\nu^{n-1}\Big(\sum_{k=0}^{n-1}\nu^k\Big) + \nu^{2n} = \Sigma\nu^{n-1}\Big(\frac{1-\nu^n}{1-\nu}\Big) + \nu^{2n}$$

Proof. We again proceed by induction. The result is clear for $n = 0$. Now, assume that this holds at generation n. Using the fact that $Y_{n,i}$ are independent, we get

$$\begin{aligned} E(X_{n+1}^2) &= E(E(X_{n+1}^2|X_n)) = E(E((Y_{n,1} + \cdots Y_{n,X_n})|X_n)) \\ &= E(X_nE(Y_{n,i}^2) + X_n(X_n - 1)(E(Y_{n,i}))^2) = E(X_n\mathrm{Var}(Y_{n,i}) + (X_nE(Y_{n,i}))^2) \\ &= E(\Sigma^2 X_n + \nu^2 X_2^n) = \Sigma^2 E(X_n) + \nu^2 E(X_n^2) \end{aligned}$$

In particular, using (33) and our assumption, we conclude that

$$\begin{aligned} E(X_{n+1}^2) &= \Sigma^2\nu^n + \nu^2\Big(\Sigma^2\nu^{n-1}\Big(\sum_{k=0}^{n-1}\Big) + \nu^{2n}\Big) \\ &= \Sigma^2\nu^n\Big(1 + \nu\Big(\sum_{k=0}^{n-1}\nu^k\Big)\Big) + \nu^{2n+2} = \Sigma^2\nu^n\Big(\sum_{k=0}^{n}\Big) + \nu^{2n+2} \end{aligned}$$

This completes the proof. $\square$

Lemma 15 – *For all $\nu \neq 1$ and $\Sigma^2 > 0$,*

$$E(X_nX_m) = \nu^{m-n}E(X_n^2) = \Sigma^2\nu^{m-1}\Big(\frac{1-\nu^n}{1-\nu}\Big) + \nu^{m+n} \text{ for all } n \le m$$

Proof. Conditioning on X_n and using (32), we get

$$\begin{aligned} E(X_nX_m) &= E(E(X_nX_m|X_n)) = \mathscr{E}(X_nE(X_m|X_n)) \\ &= E(X_n(\nu^{m-n}X_n)) = \nu^{m-n}E(X_n^2) \end{aligned}$$

The second equality follows by using Lemma 14. $\square$

Lemma 16 – *For all $\nu \neq 1$and $\Sigma^2 > 0$,*

$$E(\bar{X}_n^2) = \frac{\Sigma^2}{(1-\nu)^2}\Big(\frac{1-\nu^{2R+1}}{1-\nu} - (2R+1)\nu^R\Big) + \Big(\frac{1-\nu^{R+1}}{1-\nu}\Big)^2$$

Proof. Combining Lemmas 14 and 15 implies that

$$\begin{aligned} E(\bar{X}_R^2) &= E\Big(\sum_{n=0}^{R}\sum_{m=0}^{R} X_n X_m\Big) = 2\sum_{n=0}^{R}\sum_{m=0}^{R} E(X_n X_m) - \sum_{n=0}^{R} E(X_n^2) \\ &= \sum_{n=0}^{R}\Big[\Big(\sum_{m=n}^{R} 2\nu^{m-n}\Big) - 1\Big]E(X_n^2) = \sum_{n=0}^{R}\Big[2\Big(\frac{1-\nu^{R-n+1}}{1-\nu}\Big) - 1\Big]E(X_n^2) \quad (1.34) \\ &= \sum_{n=0}^{R}\Big(\frac{1+\nu-2\nu^{R-n+1}}{1-\nu}\Big)\Big(\Sigma^2\nu^{n-1}\Big(\frac{1-\nu^n}{1-\nu}\Big) + \nu^{2n}\Big) \end{aligned}$$

Now observe that

$$\begin{aligned} &\sum_{n=0}^{R}\Big(\frac{1+\nu-2\nu^{R-n+1}}{1-\nu}\Big)\Big(\Sigma^2\nu^{n-1}\Big(\frac{1-\nu^n}{1-\nu}\Big)\Big) \\ &\quad= \frac{\Sigma^2}{(1-\nu)^2}\sum_{n=0}^{R}\Big((1+\nu)(\nu^{n-1}-\nu^{2n-1}) - 2\nu^R(1-\nu^n)\Big) \\ &\quad= \frac{\Sigma^2}{(1-\nu)^2}\Big((1+\nu)\sum_{n=0}^{R-1}(\nu^n - \nu\nu^{2n}) - 2\nu^R\sum_{n=0}^{R}(1-\nu^n)\Big) \end{aligned}$$

Computing these sums explicitly and simplifying,

$$\begin{aligned} &\sum_{n=0}^{R}\Big(\frac{1+\nu-2\nu^{R-n+1}}{1-\nu}\Big)\Big(\Sigma^2\nu^{n-1}\Big(\frac{1-\nu^n}{1-\nu}\Big)\Big) \\ &\quad= \frac{\Sigma^2}{(1-\nu^2)}\Big[(1+\nu)\Big(\frac{1-\nu^R}{1-\nu} - \nu\Big(\frac{1-\nu^{2R}}{1-\nu^2}\Big)\Big) - 2\nu^R\Big(R+1-\frac{1-\nu^{R+1}}{1-\nu}\Big)\Big] \\ &\quad= \frac{\Sigma^2}{(1-\nu^2)}\Big[\frac{(1+\nu)(1-\nu^R)}{1-\nu} - \frac{\nu(1-\nu^{2R})}{1-\nu} + \frac{2\nu^R(1-\nu^{R+1})}{1-\nu} - 2(R+1)\nu^R\Big] \\ &\quad= \frac{\Sigma^2}{(1-\nu^2)}\Big(\frac{1-\nu^{2R+1}}{1-\nu} - (2R+1)\nu^R\Big) \end{aligned} \quad (1.35)$$

In addition, we have

$$\begin{aligned}\sum_{n=0}^{R}\Big(\frac{1+\nu-2R^{R-n+1}}{1-\nu}\Big)\nu^{2n} &= \Big(\frac{1}{1-\nu}\Big)\Big[(1+\nu)\sum_{n=0}^{R}\nu^{2n}-2\nu^{R+1}\sum_{n=0}^{R}\Big] \\ &= \Big(\frac{1}{1-\nu}\Big)\Big[(1+\nu)\Big(\frac{1-\nu^{2R+2}}{1-\nu^2}\Big)-2\nu^{R+1}\Big(\frac{1-\nu^{R+1}}{1-\nu}\Big)\Big] \\ &= \frac{1-2\nu^{R+1}+\nu^{2R+2}}{(1-\nu)^2}=\Big(\frac{1-\nu^{R+1}}{1-\nu}\Big)^2\end{aligned} \qquad (1.36)$$

Combining (1.34)-(1.36) implies the result. $\square$

1.1.9 Proof of Theorem 4

The key to proving the upper bounds in Theorem 4 is to compare the bond percolation process on the random rooted tree with the branching process studied in the previous section. To compare both processes, the first step is to find a stochastic upper bound for the random tree and the size of the infection. To do this, we consider the following random tree.

$\mathscr{T}_+ =$ rooted tree with radius $2R$ in which the number of edges starting from each vertex (and moving away from the root) is $k+1$ with probability p_k

We further define a contagion process on this tree using again bond percolation by assuming that the edges are independently open with probability p, and let

$\mathscr{S}_+ =$ number of vertices connected to the root of T_+ by an open path

which is also the size of the bond percolation cluster containing the root of the tree. The tree is designed in such a way that S_+ provides a stochastic upper bound for the size distribution of the infection on the original random tree, as shown in the next lemma.

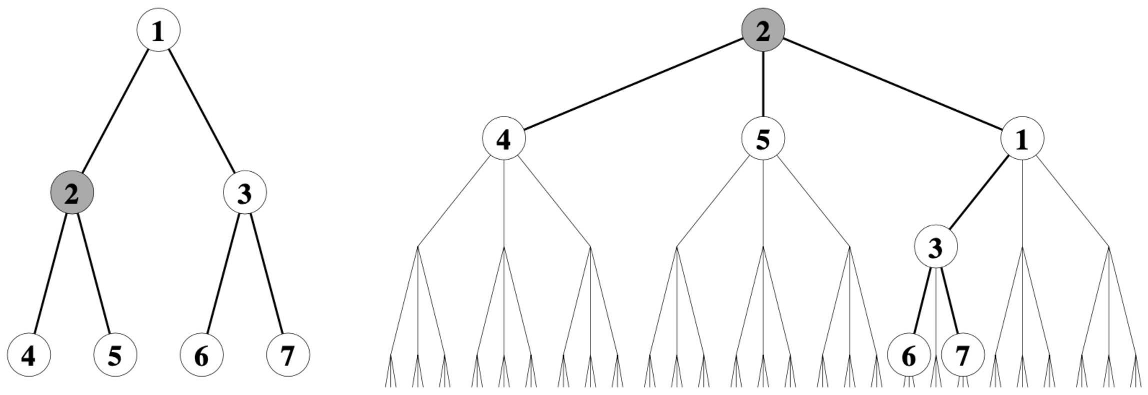


Figure 5. Illustration of the Coupling Used in the Proof of Lemma 17 When $\mathscr{T}$ Is the Deterministic Binary Tree with Radius Two. The Picture Shows That, for Any given Vertex (Vertex 2 in Our Example), This Tree Can Be Embedded in the Rooted Tree with Degree 3 and Radius 4 in Such a Way That the Designated Vertex Becomes the Root. Note That the Subtree Drawn in Thick Lines on the Right-hand Side Is Indeed Isomorphic to the Binary Tree on the Left-hand Side. In Particular, the Size of an Infection Starting at Vertex 2 on the Tree $\mathscr{T}$ Is Dominated Stochastically by the Size of an Infection Starting at the Root on the Tree $\mathscr{T}_+$ Provided the Probability of Contagion p Is the Same.

Lemma 17 – *Letting* 0_+ *be the root of the tree* $\mathscr{T}_+$ *we have,*

$$P_x(S > s) \leq P_{0_+}(S_+ > s) \text{ for all } s > 0 \text{ and } x \in \mathscr{V}$$

Proof. Let $\mathscr{T} = (\mathscr{V}, \mathscr{E})$ be a realization of the original random tree and $x = \text{Uniform}(\mathscr{V})$. First, we use a coupling argument to prove that $\mathscr{T}_+ = (\mathscr{V}_+, \mathscr{E}_+)$ can be constructed such that

P1 - the root of $\mathscr{T}_+$ coincides with x and

P2 - the random tree $\mathscr{T}$ is a subgraph of $\mathscr{T}_+$ i.e. $\mathscr{V} \subset \mathscr{V}_+$ and $\mathscr{E} \subset \mathscr{E}_+$

There is a natural approach to construct $\mathscr{T}_+$ which is illustrated in Figure 5 when the original rooted tree is the deterministic binary tree with radius two. To turn the picture into a rigorous construction, we proceed recursively by drawing edges starting from the root x of the tree $\mathscr{T}_+$, then edges starting from each of the neighbors of x (and moving away from x), and so on until generation $2R$, the radius of $\mathscr{T}_+$. To specify the set of edges at each generation, we let

$$\mathscr{E}(y) = \{z \in \mathscr{V} : d(y,z) = 1 \text{ and } d(x,z) = d(x,y) + 1\} \text{ for all } y \in \mathscr{V}$$

be the set of neighbors of y that are one unit further away form x than y. Then, the edges in $\mathscr{E}_+$ starting from the (new) root of x consists of

- the edges in $\mathscr{E}(x)$ when $0 < d(0,x), R$,
- the edges in $\mathscr{E}(x)$ and one more edge when $d(0,x) = 0$,
- the edges in $\mathscr{E}(x)$ and k more edges with probability p_k when $d(0,x) = R$,

where all the additional edges are not in $\mathscr{E}$ and their numbers are chosen independently. Assuming that all the vertices in $\mathscr{V}_+$ that are at a distance $0 < n < 2R$ from x have been constructed, for each $y \in \mathscr{V}_+$ at generation n, the set of edges starting from y consisting of

- the edges in $\mathscr{E}(y)$ and one more edge when $0 < d(0,y) < R$,
- the edges in $\mathscr{E}(y)$ and two more edges when $d(0,y) = 0$,
- a set of $k+1$ edges with probability p_k when $d(0,y) = R$ or $y \in \mathscr{V}_+ \setminus \mathscr{V}$.

where as previously the additional edges are not in $\mathscr{E}$ and chosen independently. The construction stops at generation $2R$. This construction implies that $\mathscr{T}_+$ is indeed the random rooted tree with radius $2R$ and the desired degree distribution. In addition, because

$$d(x,y) \leq 2R \text{ for all } x, y \in \mathscr{V},$$

after at most $2R$ steps, all the vertices and edges in $\mathscr{T}$ have been included in the graph $\mathscr{T}_+$, therefore the two properties P1 and P2 above hold. Next, we define bond percolation processes with the same parameter p on each of the two trees by again using a coupling. More precisely, we let

$$\xi(e) = \text{ Bernoulli}(p) \text{ for all } e \in \mathscr{E}_+$$

be independent, and declare edge e to be open if $\xi(e) = 1$. Then, each edge in $\mathscr{E}$ that is open is also an edge in $\mathscr{E}_+$ that is open. Recalling also that the new root is $0_+ = x$, we get

$$\begin{aligned} \mathscr{C}(x) &= \{y \in \mathscr{V} : \text{ there is a path } x \leftrightarrow y \text{ of open edges in } \mathscr{E}\} \\ &\subset \{y \in \mathscr{V}_+ : \text{ there is a path } 0_+ \leftrightarrow y \text{ of open edges in } \mathscr{E}_+\} = \mathscr{C}_+(0_+) \end{aligned}$$

Using monotonicity, we conclude that

$$P_x(S > s) = P(\text{card}(\mathscr{C}(x)) > s) \leq P(\text{card}(\mathscr{C}_+(0_+)) > s) = P_{0_+}(S_+ > s)$$

This completes the proof. $\square$

Motivated by Lemma 17, we now use the branching processes results from the previous section to study the first and second moments of the size distribution of the percolation cluster starting at the root of the tree $\mathscr{T}$. Spherical symmetry due to the fact that the infection starts from the root makes the calculations of the first and second moments easier.

Lemma 18 – *Let* $\nu_+ = p(\mu + 1)$ *and* $\Sigma_+^2 = p(1-p)(\mu+1) + p^2\sigma^2$. *Then,*

$$\begin{aligned} E_{0_+}(S_+) &= \frac{1 - \nu^{2R+1}}{1 - \nu_+} \\ E_{0_+}(S_+^2) &= \frac{\Sigma_+^2}{(1-\nu_+)^2}\Big(\frac{1 - \nu^{4R+1}}{1 - \nu_+} - (4R+1)\nu_+^{2R}\Big)^2 \end{aligned}$$

Proof. Because for the tree $\mathscr{T}_+$ the number of edges moving away from the root has the same distribution as $X + 1$ and each edge is independently open with probability p,

$$S_n = \text{card}\{y \in \mathscr{C}_+(0_+) : d(0_+, y) = n\}, n = 0, 1, ..., 2R,$$

the number of infected vertices at distance n from the root 0_+, is equal (in distribution) to the number of individuals at generation n in the branching process whose offspring distribution consists of the sum of $X + 1$ independent Bernoulli random variables:

$$Y_{n,i} = B_{n,i,0} + B_{n,i,1} + \cdots + B_{n,i,X} \text{ where } B_{n,i,j} = \text{Bernoulli}(p)$$

In particular, to prove the result, the key is to compute the mean and variance of the offspring distribution and then apply Lemmas 13 and 16. Conditioning on X, we get

$$E(Y_{n,i}|X) = E(B_{n,i,0} + \cdots + B_{n,i,X}|X) = (X + 1)E(B_{n,i,j}) = p(X + 1)$$

then taking expected value,

$$E(Y_{n,i}) = E(E(Y_{n,i}|X)) = pE(X + 1) = p(\mu + 1) = \nu_+$$

In addition, because the $B_{n,i,j}$ are independent,

$$\begin{aligned}\text{Var}(Y_{n,i}|X) &= \text{Var}(B_{n,i,0} = \cdots + B_{n,i,X}|X)\\ &= (X + 1)\text{Var}(B_{n,i,j}) = p(1 - p)(X + 1)\end{aligned}$$

This, together with (37) and the law of total variance, implies that

$$\begin{aligned}\text{Var}(Y_{n,i}) &= E(\text{Var}(Y_{n,i}|X)) + \ \text{Var}(E(Y_{n,i}|X))\\ &= p(1 - p)E(X + 1) + p^2 textVar(X + 1) = p(1 - p)(\mu + 1) + p^2\sigma^2 = \Sigma_+^2\end{aligned}$$

In addition, using that S_n is the number of individuals in the branching process with offspring distribution $Y_{n,i}$ and recalling that the random tree has radius $2R$, we obtain

$$E_{0_+}(S_+) = E(\bar{X}_{2R}) \text{ and } E_{0_+}(S_+^2) = E(\bar{X}_{2R}^2)$$

The lemma follows from (1.38)-(1.40) and Lemmas 13 and 16. $\square$

According to Lemmas 17 and 18,

$$E(S) \le \sup_{x\in\mathscr{V}} E_x(S) \le E_{0_+}(S_+) = \frac{1 - \nu^{2R+1}}{1 - \nu_+}$$

where $\nu_+ = p(\mu+1)$, so $E(S) \leq \phi_1(X,R)$. Similarly,

$$E(S^2) \leq E_{0_+}(S_+^2) = \frac{\Sigma_+^2}{(1-\nu)^2}\Big(\frac{1-\nu^{4R+1}}{1-\nu_+} - (4R+1)\nu_+^{2R}\Big) + \Big(\frac{1-\nu_+^{2R+1}}{1-\nu_+}\Big)^2$$

where $\Sigma_+^2 = p(1-p)(\mu+1) + p^2\sigma^2$, so $E(S^2) \leq \phi_2(X,R)$. This shows the theorem.

1.2 First and Second Moments of the Size Distribution of Bond Percolation Clusters on Regular graphs

1.2.1 Introduction

Bond percolation consists of a collection of independent Bernoulli random variables with the same success probability p indexed by the edges of a graph, with the edges associated to a success being open and the ones associated to a failure being closed. The percolation clusters are the connected components of the subgraph induced by the open edges. This process was introduced by [7] to study the spread of a fluid through a medium. Bond percolation has been extensively studied on infinite graphs such as integer lattices [8] in which case the process typically exhibits a phase transition for the density p of open edges from a subcritical phase where all the percolation clusters are finite to a supercritical phase where at least one percolation cluster is infinite. Bond percolation has also been studied along increasing sequences of finite graphs: as the size of the graphs tends to infinity, the supercritical phase is now characterized by the existence of a giant connected component of open edges whose size scales like the size of the graph. Important examples are the complete graph, in which case the set of open edges consists of the Erdos-Rényi random graph [9], and the hypercube [10]. Much less attention has been paid to bond percolation on fixed finite graphs in spite of its growing importance in terms of applications.

Indeed, the first moment of the size of the percolation clusters on finite graphs is closely related to the notion of network resilience in computer network theory [11]. Similarly, the modeling framework introduced by [2] shows that, in the context of cyber security, both the first and the second moments of the cluster size are keys to computing insurance premiums. Motivated by these aspects, [12] studied the first and second moments of the cluster size on elementary graphs: the path, the ring and the star. In both contexts (network resilience and cyber insurance), the underlying graph represents a local area network, i.e., a finite group of computers (the vertices) along with the way these computers are connected (the edges). In this work, we study the first and second moments of the cluster size on general finite regular graphs that model local area networks more realistically than paths, rings or stars.

1.2.2 Model Description

Throughout this section, $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ is a finite connected $D-$regular graph with N vertices. Let x be a vertex chosen uniformly at random and assume that the edges are independently open with probability p. The main objective of this section is to study the first and second moments of

$$S = \operatorname{card}(\mathscr{C}_x)$$

where

$$\mathscr{C}_x = \{y \in \mathscr{V} : \text{there is a path of open edges connecting } x \text{ and } y\}$$

the random number of vertices in the percolation cluster starting at x.

1.2.3 Main Results

In this section, we prove the following upper bounds for the first and second moments.

Theorem 19 – *For every D-regular graph with N vertices,*

$$\begin{aligned} E(S) &\leq 1 + Dp\left(\frac{1-\nu^R}{1-\nu}\right) \\ E(S^2) &\leq \left(1 + Dp\left(\frac{1-\nu^R}{1-\nu}\right)\right)^2 + \frac{Dp(1-p)}{(1-\nu)^2}\left(\frac{(1-\nu^R)(1+\nu^{R+1})}{1-\nu} - 2R\nu^R\right) \end{aligned}$$

where $\nu = (D-1)p$ and $R = N-1$.

To prove the theorem, the idea is to think of the cluster $\mathscr{C}_x$ as a dynamical object described by a birth process starting with one particle at x and in which particles give birth with probability p onto vacant adjacent vertices. The size of the cluster is equal to the ultimate number of particles in the birth process which, in turn, is dominated stochastically by the number of individuals up to generation $\mathrm{card}(\mathscr{V}) - 1 = N - 1$ in a certain branching process.

Taking D and N in the theorem to be the degree and the number of vertices in each of the Platonic solids, we get the solid curves in Figures 8–10, Section 3. Note that these upper bounds are only accurate for p small. To have upper bounds that are accurate for p large, we use that a vertex $y \neq x$ cannot be in the percolation cluster $\mathscr{C}_x$ when all the edges incident to x are closed. This gives the following result that again applies to all finite regular graphs.

Theorem 20 – *For every D-regular graph with N vertices,*

$$E(S) \leq N - (N-1)(1-p)^D$$

$$E(S^2) \leq N^2 - (N-1)(2N-1)(1-p)^D + (N-1)(N-2)(1-p)^{2D-1}.$$

1.2.4 Proof of Theorem 1

Having a vertex $x \in \mathscr{V}$ and a realization of bond percolation with parameter p on the graph, we consider the following discrete-time birth process (ξ_n). The state at time n is a spatial configuration of particles on the vertices:

$$\xi_n \subset \mathscr{V} \text{ where } \xi_n = \text{set of vertices occupied by a particle at time } n$$

The process starts at generation 0 with one particle at x, therefore $\xi_0 = \{x\}$. Then,

- for each vertex y adjacent to vertex x, the particle at x gives birth to a particle sent to vertex y if and only if the edge (x, y) is open.

These are the particles at generation 1. Assume the process has been defined up to generation $n > 0$, and let $Y_n = \text{card}(\xi_n \setminus \xi_{n-1})$ be the number of particles of generation n. Label $1, 2, ..., Y_n$ these particles and let $x_{n,1}, x_{n,2}, ..., x_{n,Y_n}$ be their locations so that

$$\xi_n \setminus \xi_{n-1} = \{x_{n,1}, x_{n,2}, ..., x_{n,Y_n}\}$$

Then, generation $n+1$ is defined sequentially from step 1 to step Y_n where, at step i,

- for each vertex y adjacent to vertex $x_{n,i}$, the ith particle of generation n gives birth to a particle sent to vertex y if and only if vertex y is empty and the edge $(x_{n,i}, y)$ is open.

Note that two particles i and j with $i < j$ might share a common neighbor y in which case a child of particle i sent to y prevents particle j from giving birth onto y. For a construction of the birth process from a realization of bond percolation on the dodecahedron, we refer to Figure 6. The process is designed so that particles ultimately occupy the open cluster starting at x. In particular, the total number of particles equals the cluster size, as proved in the next lemma.

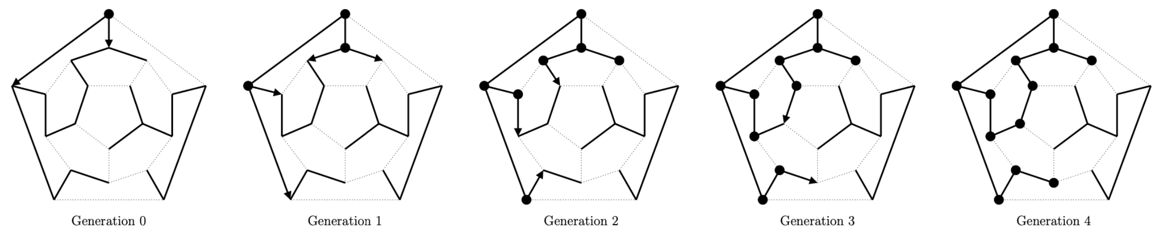


Figure 6. Example of a Construction of the Birth Process from a Realization of Bond Percolation on the Dodecahedron. The Thick Lines Represent the Open Edges, the Black Dots Represent the Vertices Occupied by a Particle at Each Generation, and the Arrows Represent the Birth Events, from Parent to Children.

Lemma 21 – *The cluster size is given by*

$$S = \operatorname{card}(\mathscr{C}_x) = \operatorname{card}(\xi_{N-1}) = Y_0 + Y_1 + \cdots + Y_{N-1} \quad \textit{where} \quad N = \operatorname{card}(\mathscr{V}).$$

PROOF. To begin with, we observe that

- Because particles can only give birth to another particle sent to an empty vertex, each vertex is ultimately occupied by at most one particle.
- The open cluster containing x can be written as

$$\mathscr{C}_x = \{y \in \mathscr{V} : \text{there is a self-avoiding path of open edges connecting vertex } x \text{ and vertex } y\}.$$

- The set of vertices occupied by a particle of generation n is

$$\xi_n \setminus \xi_{n-1} = \{y \in \mathscr{C}_x : \text{the shortest self-avoiding path of open edges connecting } x \text{ and } y \text{ has length } n\}.$$

These three properties imply that all the vertices in the open cluster $\mathscr{C}_x$ are ultimately occupied by exactly one particle whereas the vertices outside the cluster remain empty therefore

$$\begin{aligned} S = \operatorname{card}(\mathscr{C}_x) &= \operatorname{card}(\xi_0) + \operatorname{card}\left(\bigcup_{n=1}^{\infty} (\xi_n \setminus \xi_{n-1})\right) \\ &= \operatorname{card}(\xi_0) + \sum_{n=1}^{\infty} \operatorname{card}(\xi_n \setminus \xi_{n-1}) = \sum_{n=0}^{\infty} Y_n. \end{aligned} \tag{1.37}$$

In addition, because the graph has N vertices, the shortest self-avoiding path on this graph must have at most $N-1$ edges, from which it follows that

$$\xi_n = \xi_{n-1} \quad \text{and} \quad Y_n = \operatorname{card}(\xi_n \setminus \xi_{n-1}) = 0 \quad \text{for all} \quad n \geq N. \tag{1.38}$$

Combining (1.37) and (1.38) gives the result. $\square$

Coupling with a branching process. The next step is to compare the number of particles in the birth process with the number of individuals in a branching process (X_n). The process coincides with the birth process when the graph is a tree, and is defined as

$$X_0 = 1 \quad \text{and} \quad X_{n+1} = X_{n,1} + X_{n,2} + \cdots + X_{n,X_n} \quad \text{for all} \quad n \geq 0$$

where the random variables $X_{n,i}$ representing the offspring distribution (number of offspring of individual i at time n) are independent and have probability mass function

$$X_{0,1} = \text{Binomial}\,(D, p) \quad \text{and} \quad X_{n,i} = \text{Binomial}\,(D-1, p) \quad \text{for all} \quad n, i \geq 1.$$

This branching process can be visualized as the number of particles in the birth process above modified so that births onto already occupied vertices are allowed. In particular, the branching process dominates stochastically the birth process.

Lemma 22 – *For all $n \geq 0$, we have the stochastic domination $Y_n \preceq X_n$.*

Proof. As for the branching process, for all $n \geq 0$ and $i \leq Y_n$, we let

$$Y_{n,i} = \#\ \text{offspring of the } i\text{th particle of generation } n \text{ in the birth process.}$$

Because the edges are independently open with the same probability p and there are exactly D edges starting from each vertex, the number of offspring of the first particle is

$$Y_1 = Y_{0,1} = \text{Binomial}\,(D, p). \tag{1.39}$$

For each subsequent particle, say the particle located at z, we distinguish two types of edges starting from z just before the particle gives birth.

- There are m edges (z, y) that are connected to an occupied vertex y. Because parent and offspring are located on adjacent vertices, we must have $m \geq 1$.
- There are $D - m$ edges (z, y) that are connected to an empty vertex y. These edges have not been used yet in the construction of the birth process, i.e., there has been no previous attempt to give birth through these edges, therefore each of these edges is open with probability p independently of the past of the process.

From the previous two properties, we deduce that, for all $n > 0$ and $i \leq Y_n$,

$$\begin{aligned} P(Y_{n,i} \geq k) &= E(P(Y_{n,i} \geq k \,|\, Y_{0,1}, Y_{1,1}, \ldots, Y_{n,i-1})) \\ &\leq P(\text{Binomial}\,(D-1, p) \geq k) = P(X_{n,i} \geq k). \end{aligned} \tag{1.40}$$

The stochastic domination follows from (1.39) and (1.40). $\square$

Number of individuals. It directly follows from Lemmas 21 and 22 that

$$E(S^k) = E((Y_0 + Y_1 + \cdots + Y_{N-1})^k) \leq E((X_0 + X_1 + \cdots + X_{N-1})^k) \tag{1.41}$$

for all $k > 0$. In view of (1.41), the last step to complete the proof of Theorem 19 is to show that the upper bounds in the theorem are in fact the first and second moments of the total number of individuals up to generation $R = N - 1$ in the branching process:

$$E(\bar{X}_R) \quad \text{and} \quad E(\bar{X}_R^2) \quad \text{where} \quad \bar{X}_R = X_0 + X_1 + \cdots + X_R.$$

The rest of this section is devoted to computing these moments.

Lemma 23 – *Let $\nu = (D-1)p$. Then,*

$$E(\bar{X}_R) = 1 + Dp\left(\frac{1 - \nu^R}{1 - \nu}\right) \quad \textit{for all} \quad R > 0.$$

PROOF. For $i = 1, 2, \dots, X_1$, let

$$\bar{Z}_i = \text{number of descendants of the } i\text{th offspring of the first individual up to generation } R\text{, including the offspring.}$$

Then $\bar{X}_R = 1 + \bar{Z}_1 + \cdots + \bar{Z}_{X_1}$ and the $\bar{Z}_i$ are independent of X_1 so

$$\begin{aligned} E(\bar{X}_R) &= E(E(\bar{X}_R \mid X_1)) = E(E(1 + \bar{Z}_1 + \cdots + \bar{Z}_{X_1} \mid X_1)) \\ &= E(1 + X_1 E(\bar{Z}_i)) = 1 + E(X_1)E(\bar{Z}_i) = 1 + DpE(\bar{Z}_i). \end{aligned}$$

Because $\bar{Z}_i$ is the number of individuals up to generation $R-1$ in a branching process with offspring distribution Binomial $(D-1, p)$, we deduce from [2, Theorem 2] that

$$E(\bar{X}_R) = 1 + Dp\left(\frac{1-(\mu p)^R}{1-\mu p}\right) = 1 + Dp\left(\frac{1-\nu^R}{1-\nu}\right) \quad \text{where} \quad \nu = \mu p = (D-1)p.$$

This completes the proof. $\square$

Using the same decomposition as in the previous lemma, we now compute the second moment of the number of individuals up to generation $R = N-1$.

Lemma 24 – *Let $\nu = (D-1)p$. Then, for all $R > 0$,*

$$E(\bar{X}_R^2) = \left(1 + Dp\left(\frac{1-\nu^R}{1-\nu}\right)\right)^2 + \frac{Dp(1-p)}{(1-\nu)^2}\left(\frac{(1-\nu^R)(1+\nu^{R+1})}{1-\nu} - 2R\nu^R\right).$$

PROOF. Using again $\bar{X}_R = 1 + \bar{Z}_1 + \cdots + \bar{Z}_{X_1}$ and independence, we get

$$\begin{aligned} E(\bar{X}_R^2) &= E(E((1 + \bar{Z}_1 + \cdots + \bar{Z}_{X_1})^2 \mid X_1)) \\ &= E(E(1 + 2(\bar{Z}_1 + \cdots + \bar{Z}_{X_1}) + (\bar{Z}_1 + \cdots + \bar{Z}_{X_1})^2 \mid X_1)) \\ &= E(1 + 2X_1E(\bar{Z}_i) + X_1E(\bar{Z}_i^2) + X_1(X_1-1)(E(Z_i))^2) \\ &= 1 + 2E(X_1)E(\bar{Z}_i) + E(X_1)E(\bar{Z}_i^2) + E(X_1(X_1-1))(E(Z_i))^2. \end{aligned} \tag{1.42}$$

In addition, using that $X_1 = \text{Binomial}\,(D, p)$, we get

$$\begin{aligned} E(X_1(X_1-1)) &= \text{Var}(X_1) + (E(X_1))^2 - E(X_1) \\ &= Dp(1-p) + D^2p^2 - Dp = D(D-1)p^2. \end{aligned} \tag{1.43}$$

Combining (1.42) and (1.43) gives

$$\begin{aligned} E(\bar{X}_R^2) &= 1 + 2DpE(\bar{Z}_i) + DpE(\bar{Z}_i^2) + D(D-1)p^2(E(\bar{Z}_i))^2 \\ &= 1 + 2DpE(\bar{Z}_i) + Dp(\text{Var}(\bar{Z}_i) + (E(\bar{Z}_i))^2) + D(D-1)p^2(E(\bar{Z}_i))^2 \\ &= 1 + 2DpE(\bar{Z}_i) + Dp(Dp + 1 - p)(E(\bar{Z}_i))^2 + Dp\,\text{Var}(\bar{Z}_i) \\ &= (1 + DpE(\bar{Z}_i))^2 + Dp(1-p)(E(\bar{Z}_i))^2 + Dp\,\text{Var}(\bar{Z}_i). \end{aligned}$$

Then, applying [2, Theorem 2] with $\mu = D - 1$ and $\sigma^2 = 0$, we get

$$\begin{aligned} E(\bar{X}_R^2) = \left(1 + Dp\left(\frac{1-\nu^R}{1-\nu}\right)\right)^2 &+ Dp(1-p)\left(\frac{1-\nu^R}{1-\nu}\right)^2 \\ &+ Dp\,\frac{\nu(1-p)}{(1-\nu)^2}\left(\frac{1-\nu^{2R-1}}{1-\nu} - (2R-1)\nu^{R-1}\right). \end{aligned}$$

Observing also that

$$\begin{aligned} Dp\,(1-p)\quad\left(\frac{1-\nu^R}{1-\nu}\right)^2 &+ Dp\,\frac{\nu(1-p)}{(1-\nu)^2}\left(\frac{1-\nu^{2R-1}}{1-\nu} - (2R-1)\nu^{R-1}\right) \\ &= \frac{Dp(1-p)}{(1-\nu)^2}\left(\frac{(1-\nu)(1-\nu^R)^2 + \nu(1-\nu^{2R-1})}{1-\nu} - (2R-1)\nu^R\right) \\ &= \frac{Dp(1-p)}{(1-\nu)^2}\left(\frac{1 - 2\nu^R + 2\nu^{R+1} - \nu^{2R+1}}{1-\nu} - (2R-1)\nu^R\right) \\ &= \frac{Dp(1-p)}{(1-\nu)^2}\left(\frac{1 - 2\nu^R + 2\nu^{R+1} - \nu^{2R+1} + (1-\nu)\nu^R}{1-\nu} - 2R\nu^R\right) \\ &= \frac{Dp(1-p)}{(1-\nu)^2}\left(\frac{(1-\nu^R)(1+\nu^{R+1})}{1-\nu} - 2R\nu^R\right) \end{aligned}$$

completes the proof. □

Theorem 19 directly follows from (1.41), and from Lemmas 23 and 24.

1.2.5 Proof of Theorem 2

Theorem 1 gives a good upper bounds when the probability is p is small. To complement this result, we now give a second set of upper bounds that are accurate when p is close to one.

Theorem 20 relies on the following observation: vertex $y \neq x$ cannot be in the percolation cluster starting at x when all the edges incident to y are closed. In contrast with the comparison with branching processes, this result leads to a good approximation of the moments of the size distribution when the probability p approaches one. To prove the theorem, note that

$$\begin{aligned} E(S^k) = E\bigg(\sum_{y \in \mathscr{V}} \mathbf{1}\{y \in \mathscr{C}_x\}\bigg)^k &= \sum_{y_1, \ldots, y_k \in \mathscr{V}} E(\mathbf{1}\{y_1 \in \mathscr{C}_x\} \ \cdots \ \mathbf{1}\{y_k \in \mathscr{C}_x\}) \\ &= \sum_{y_1, \ldots, y_k \in \mathscr{V}} P(x \leftrightarrow y_1, \ldots, x \leftrightarrow y_k) \end{aligned} \tag{1.44}$$

for all integers k. To estimate the last sum, we let B_y be the event that all the edges incident to y are closed. Using that there are exactly D edges incident to each vertex, and that there is at most one edge connecting any two different vertices, say $y \neq z$, we get

$$\begin{aligned} &P(B_y) = (1-p)^D \\ &P(B_y \cup B_z) = P(B_y) + P(B_z) - P(B_y \cap B_z) \geq 2(1-p)^D - (1-p)^{2D-1}. \end{aligned} \tag{1.45}$$

In addition, we have the inclusion of events

$$B_y \subset \{x \not\leftrightarrow y\} \quad \text{for all} \quad y \neq x. \tag{1.46}$$

Combining (1.45) and (1.46), we get

$$P(x \not\leftrightarrow y \text{ or } x \not\leftrightarrow z) \geq \begin{cases} (1-p)^D & \text{when} \quad \operatorname{card}\{x, y, z\} = 2 \\ 2(1-p)^D - (1-p)^{2D-1} & \text{when} \quad \operatorname{card}\{x, y, z\} = 3. \end{cases} \tag{1.47}$$

Using (1.44) with $k = 1$ and (1.47), we deduce that

$$\begin{aligned} E(S) &= 1 + \textstyle\sum_{y \neq x} P(x \leftrightarrow y) = 1 + \sum_{y \neq x} (1 - P(x \not\leftrightarrow y)) \\ &\leq 1 + \textstyle\sum_{y \neq x} (1 - (1-p)^D) \\ &= 1 + (N-1)(1 - (1-p)^D) = N - (N-1)(1-p)^D. \end{aligned}$$

Similarly, applying (1.44) with $k = 2$, observing that

$$\operatorname{card}\{(y,z) \in \mathscr{V}^2 : \operatorname{card}\{x,y,z\} = 2\} = 3(N-1)$$

$$\operatorname{card}\{(y,z) \in \mathscr{V}^2 : \operatorname{card}\{x,y,z\} = 3\} = (N-1)(N-2),$$

and using (1.45) and (1.47), we deduce that

$$\begin{aligned} E(S^2) &\leq 1 + 3(N-1)(1-(1-p)^D) + (N-1)(N-2)(1 - 2(1-p)^D + (1-p)^{2D-1}) \\ &= N^2 - 3(N-1)(1-p)^D - (N-1)(N-2)(2(1-p)^D - (1-p)^{2D-1}) \\ &= N^2 - (N-1)(2N-1)(1-p)^D + (N-1)(N-2)(1-p)^{2D-1}. \end{aligned}$$

This completes the proof of Theorem 20.

1.3 Exact and Approximations to First and Second Moments of the Size Distribution of Bond Percolation Cluster on the Platonic Solids

1.3.1 Introduction

Having a simple finite undirected graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$, let

$$x = \text{Uniform}(\mathscr{V}) \quad \text{and} \quad \zeta(e) = \text{Bernoulli}(p),\ e \in \mathscr{E},$$

be a vertex chosen uniformly at random and a collection of Bernoulli random variables with the same success probability p on the set of edges. The edges with $\zeta(e) = 1$ are said to be open while the edges with $\zeta(e) = 0$ are said to be closed, and we let

$$\mathscr{C}_x = \{y \in \mathscr{V} : \text{there is a path of open edges connecting } x \text{ and } y\}$$

be the percolation cluster containing x. The main objective of this section is to study the first and second moments of $S = \text{card}(\mathscr{C}_x)$ = the size of this percolation cluster when the graph $\mathscr{G}$ consists of each of the five Platonic solids depicted in Figure 7.

Our last results are specific to the five Platonic solids and we denote by S_f the size of a percolation cluster on the solid with f faces. To obtain these results, we first observe that the mean cluster size can be easily expressed using the probability that each vertex belongs to the open cluster $\mathscr{C}_x$ which, in turn, is equal to the probability that at least one of the self-avoiding paths connecting x to this vertex is open. In particular, identifying all the self-avoiding paths connecting x to any other vertex and using the inclusion-exclusion identity give an exact expression for the first moment. The same holds for the second moment looking instead at all the pairs of paths connecting x to two other vertices. This approach also shows that the first and second moments of the cluster size are polynomials in p with integer coefficients and degree (at most) the total number of edges so, to state our next results and shorten the notation, we let

$$P_k = (p^0, p^1, \dots, p^k) \text{ for all } k \in \mathbb{N}.$$

The main difficulties following this strategy is to identify all the self-avoiding paths and compute the probability that any sub-collection of paths are simultaneously open.

1.3.2 Main Results

Theorem 1 (tetrahedron) – *For all* $p \in (0, 1)$,

$$E(S_4) = (1, 3, 6, 0, -21, 21, -6) \cdot P_6 \quad \textit{and} \quad E(S_4^2) = (1, 9, 36, 30, -171, 153, -42) \cdot P_6.$$

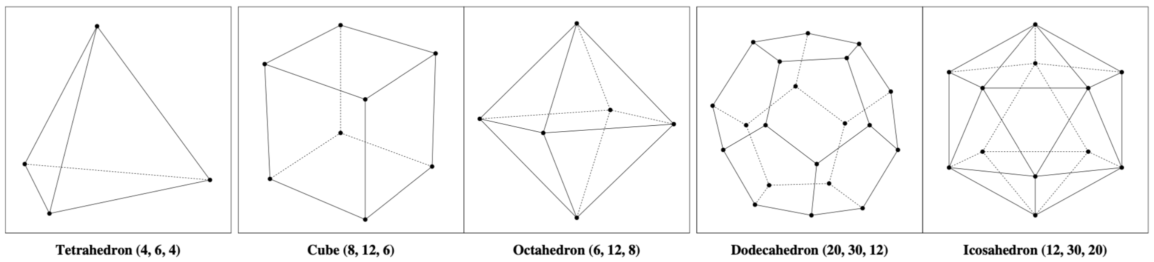


Figure 7. Picture of the Five Platonic Solids. The Number Between Parentheses Refer to the Number of Vertices, the Number of Edges, and the Number of Faces, Respectively. Note That the Tetrahedron Is Dual to Itself, the Cube and the Octahedron Are Dual to Each Other, and the Dodecahedron and Icosahedron Are Dual to Each Other.

Theorem 2 (cube and octahedron) – *For all* $p \in (0,1)$,

$$E(S_6) = (1,3,6,12,9,12,-81,-75,69,473,-777,447,-91) \cdot P_{12}$$

$$E(S_8) = (1,4,12,20,-14,-196,12,1316,-2815,2824,-1564,464,-58) \cdot P_{12}.$$

The dodecahedron and the icosahedron both have thirty edges. For these two solids, even writing down all the self-avoiding paths connecting two vertices is beyond human capability so we only focus on the paths of length at most five for the dodecahedron and of length at most three for the icosahedron.

Using that two vertices are in the same open cluster if (but not only if) at least one of the paths is open, we get the following lower bounds for the mean cluster size.

Theorem 3 (dodecahedron and icosahedron) – *For all* $p \in (0,1)$,

$$\begin{aligned} E(S_{12}) \geq\ & (1,3,6,12,24,30,-24,-30,-36,3,-6,42, \\ & -6,18,-21,14,0,-6,-9,0,0,6,0,0,-1,0,0,0,0,0,0) \cdot P_{30} \end{aligned}$$

$$\begin{aligned} E(S_{20}) \geq\ & (1,5,20,60,-90,-75,0,190,-10,-80,-60,10, \\ & -5,120,-35,-88,35,40,-35,10,-1,0,0,0,0,0,0,0,0,0,0) \cdot P_{30}. \end{aligned}$$

The two moments in Theorem 1 and the first moments in Theorem 2 are represented by the thick solid curves in Figures 8 and 9. These curves fit perfectly with numerical

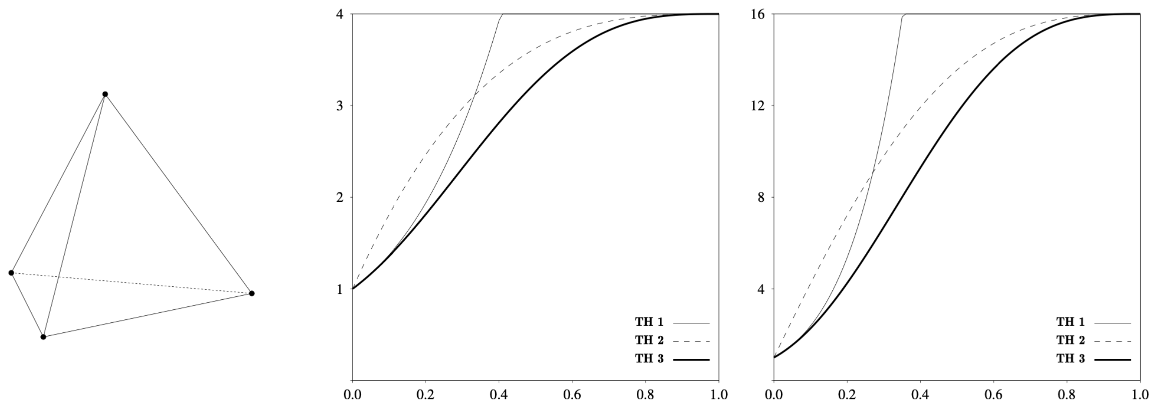


Figure 8. First Moment on the Left and Second Moment on the Right of the Size Distribution of Bond Percolation Clusters on the Tetrahedron as Function of p. The Thick Solid Lines Show the Exact Expressions in Theorem 1 of This Section While the Other Curves, Labeled Th 3, and Theorems 1, 2 from the Previous Section Labeled as Th1, and Th 2 Respectively.

solutions obtained from one hundred thousands independent realizations of the percolation process. The lower bounds for the first moments in Theorem 3 are represented by the dotted curves in Figure 10.

1.3.3 Preliminary Results

First, we prove a result (see (1.51) below) that holds not only for the Platonic solids but also a larger class of regular graphs. Fix a vertex $x \in \mathscr{V}$, let r be the radius of the graph, and define

$$\Lambda_s = \{y \in \mathscr{V} : d(x,y) = s\} \quad \text{and} \quad N_s = \text{card}\,(\Lambda_s) \quad \text{for} \quad s = 0, 1, \ldots, r.$$

At least for the Platonic solids, N_s does not depend on the choice of x. Fixing

$$y_s \in \Lambda_s \quad \text{for all} \quad s = 0, 1, \ldots, r,$$

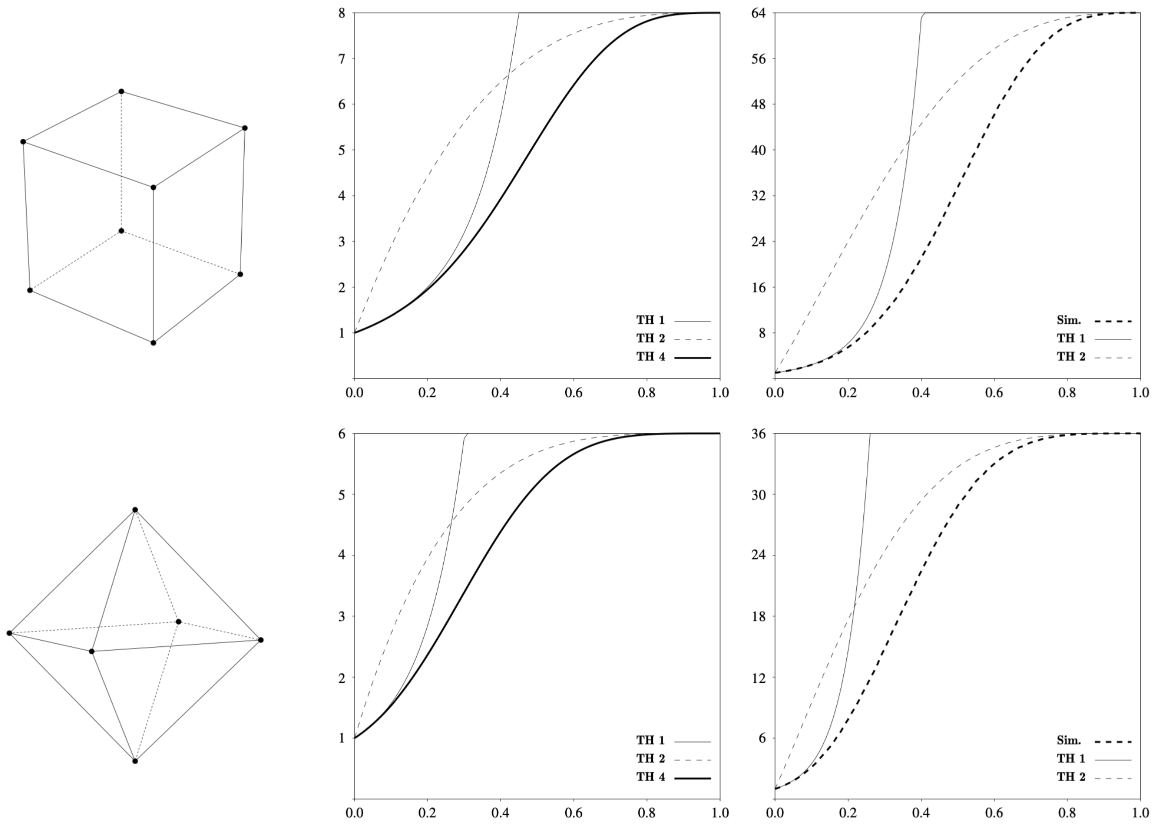


Figure 9. First Moment on the Left and Second Moment on the Right of the Size Distribution of Bond Percolation Clusters on the Cube (Top) and the Octahedron (Bottom) as Functions of the Probability p. The Thick Solid Lines Show the Exact Expressions Found in Theorem 2 of This Section, the Thick Dashed Lines Show the Second Moment Obtained from the Average of One Hundred Thousand Independent Realizations of the Process for Various Values of p, and the Other Curves Show the Upper Bounds in Th 1 and Th 2 from the Previous Section for the Appropriate Values of D and N.

and applying (1.44) with $k = 1$, we get

$$E(S) = \sum_{y \in \mathscr{V}} P(x \leftrightarrow y) = \sum_{s=0}^{r} \sum_{y \in \Lambda_s} P(x \leftrightarrow y) = \sum_{s=0}^{r} N_s P(x \leftrightarrow y_s). \tag{1.48}$$

To compute the probabilities $p_s = P(x \leftrightarrow y_s)$, we label the edges $0, 1, \ldots, n-1$, think of each self-avoiding path π as the collection of its edges, and let

$$\pi_1(y_s), \ldots, \pi_{K_s}(y_s) = \text{all the self-avoiding paths } x \to y_s$$

$$A_i = \text{the event that } \pi_i(y_s) \text{ is an open path for } i = 1, 2, \ldots, K_s.$$

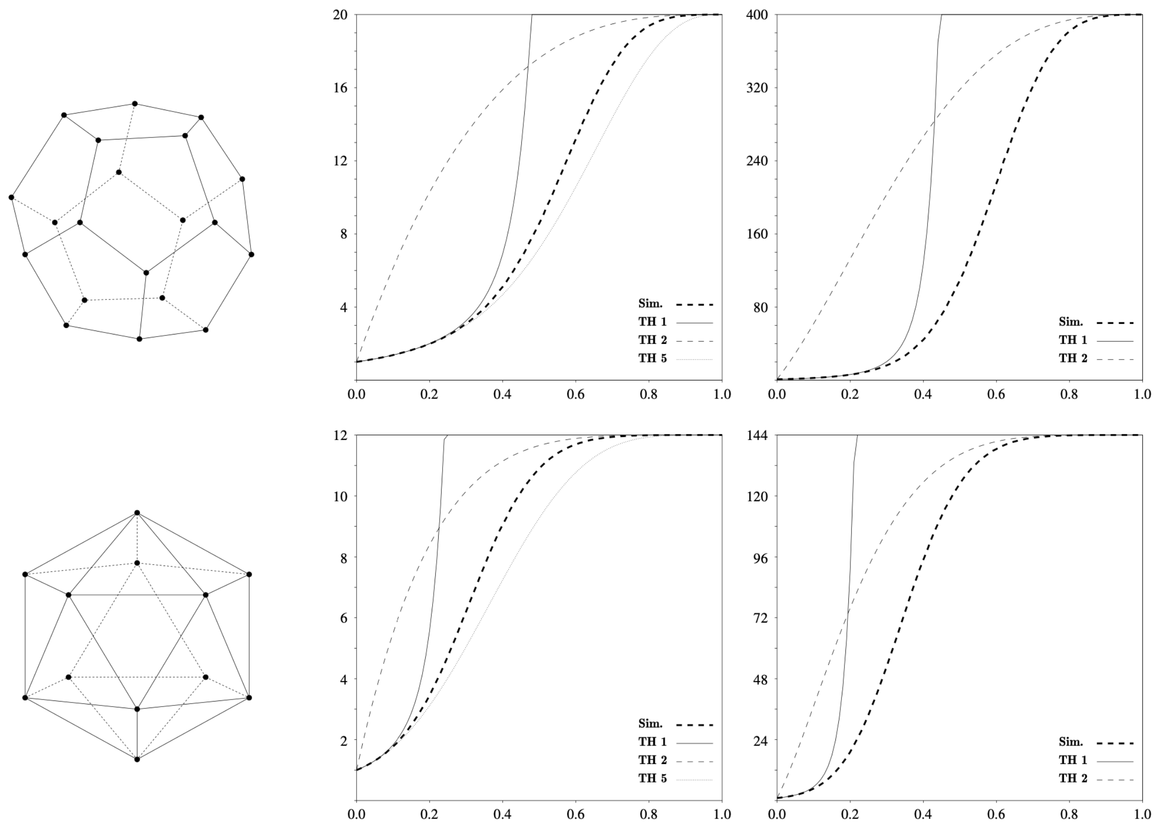


Figure 10. First Moment on the Left and Second Moment on the Right of the Size Distribution of Bond Percolation Clusters on the Dodecahedron (Top) and Icosahedron (Bottom) as Functions of the Probability p. The Thick Dashed Lines Show the Second Moment Obtained from the Average of One Hundred Thousand Independent Realizations of the Process for Various Values of p While the Other Curves Show the Upper Bounds in Th 1 and Th 2 from the Previous Section for the Appropriate Values of the Degree D and the Number of Vertices N, and the Lower Bounds in Th 3.

Because the edges are independently open with the same probability p,

$$\begin{aligned} P(A_{i_1} \cap \cdots \cap A_{i_j}) &= P(\pi_{i_1}(y_s), \ldots, \pi_{i_j}(y_s) \text{ are open paths}) \\ &= P(e \text{ is open for all } e \in \pi_{i_1}(y_s) \cup \cdots \cup \pi_{i_j}(y_s)) \\ &= p^{\operatorname{card}(\pi_{i_1}(y_s) \cup \cdots \cup \pi_{i_j}(y_s))} \end{aligned}$$

for all $0 < i_1 < \cdots < i_j \leq K_s$. Here card refers to the number of edges in the subgraph that consists of the union of the self-avoiding paths. Using that $x \leftrightarrow y_s$ if and only if at least one of the paths connecting the vertices is open, it follows from the inclusion-exclusion identity that

$$\begin{aligned} P(x \leftrightarrow y_s) &= P\bigg(\bigcup_{j=1}^{K_s} A_j\bigg) = \sum_{j=1}^{K_s} (-1)^{j+1} \sum_{0<i_1<\cdots<i_j\le K_s} P(A_{i_1}\cap\cdots\cap A_{i_j}) \\ &= \sum_{j=1}^{K_s} (-1)^{j+1} \sum_{0<i_1<\cdots<i_j\le K_s} p^{\operatorname{card}(\pi_{i_1}(y_s)\cup\cdots\cup\pi_{i_j}(y_s))}. \end{aligned} \tag{1.49}$$

Note that, in the previous expression, the index j corresponds to the number of self-avoiding paths while the second sum is over all possible choices of j paths. In particular, the double sum consists in looking at all the possible nonempty sub-collections of the K_s self-avoiding paths, therefore the right-hand side of (1.49) can be rewritten as

$$P(x \leftrightarrow y_s) = \sum_{B\subset[K_s]:B\neq\varnothing} (-1)^{\operatorname{card}(B)+1}\, p^{\operatorname{card}\left(\bigcup_{i\in B}\pi_i(y_s)\right)} \tag{1.50}$$

where $[K_s] = \{1, 2, \dots, K_s\}$. Combining (1.48) and (1.50) gives

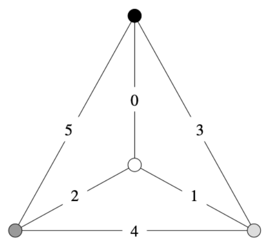
$$E(S) = \sum_{s=0}^{r} N_s \Bigg(\sum_{B\subset[K_s]:B\neq\varnothing} (-1)^{\operatorname{card}(B)+1}\, p^{\operatorname{card}\left(\bigcup_{i\in B}\pi_i(y_s)\right)}\Bigg). \tag{1.51}$$

The previous equation shows that computing the mean cluster size reduces to finding the self-avoiding paths that connect any two vertices of the graph. We now apply (1.51) to each of the five Platonic solids in order to prove Theorems 1–3.

1.3.4 Proof of Theorem 1

For the tetrahedron, all the vertices are distance one apart and there are exactly five self-avoiding paths connecting any two vertices (see first table in Figure 11). Calling these paths $\pi_1, \dots, \pi_5$ in the order they are listed in the table, and writing

$$\operatorname{card}(\pi_{i_1}\cup\pi_{i_2}\cup\cdots\cup\pi_{i_j}) = |\pi_{i_1,i_2,\dots,i_j}|$$

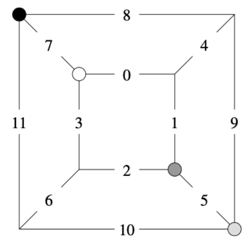


1	0
2	1, 3
	2, 5
3	1, 4, 5
	2, 3, 4

● ⟷ ○

2	3, 4
	3, 5
	4, 5
3	0, 1, 2
	0, 1, 4
	0, 1, 5
	0, 2, 3
	0, 2, 4
	1, 2, 3
	1, 2, 5

◉ ⟷ ● ⟷ ◎



1	7
3	0, 4, 8
	3, 6, 11
5	0, 1, 2, 6, 11
	0, 1, 5, 8, 9
	0, 1, 5, 10, 11
	0, 4, 9, 10, 11
	1, 2, 3, 4, 8
	2, 3, 5, 8, 9
	2, 3, 5, 10, 11
	3, 6, 8, 9, 10
7	0, 1, 2, 6, 8, 9, 10
	0, 2, 4, 5, 6, 9, 11
	1, 2, 3, 4, 9, 10, 11
	1, 3, 4, 5, 6, 8, 10

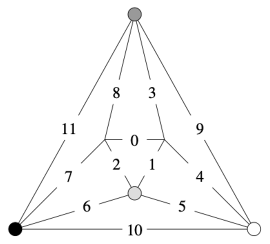
● ⟷ ○

2	8, 9
	10, 11
4	0, 1, 5, 7
	0, 4, 7, 9
	1, 4, 5, 8
	2, 3, 5, 7
	2, 5, 6, 11
	3, 6, 7, 10
6	0, 1, 2, 6, 7, 10
	0, 1, 3, 5, 6, 11
	0, 2, 3, 4, 5, 8
	0, 3, 4, 6, 8, 10
	0, 3, 4, 6, 9, 11
	1, 2, 3, 4, 7, 9
	1, 2, 4, 6, 8, 10
	1, 2, 4, 6, 9, 11

● ⟷ ◎

3	0, 1, 7
	1, 4, 8
	2, 3, 7
	2, 6, 11
	5, 8, 9
	5, 10, 11
5	0, 1, 3, 6, 11
	0, 2, 3, 4, 8
	0, 4, 5, 7, 9
	1, 4, 9, 10, 11
	2, 6, 8, 9, 10
	3, 5, 6, 7, 10
7	0, 1, 3, 6, 8, 9, 10
	0, 2, 3, 4, 9, 10, 11
	0, 2, 4, 6, 7, 9, 10
	0, 3, 4, 5, 6, 8, 10
	0, 3, 4, 5, 6, 9, 11
	1, 3, 4, 6, 7, 9, 10

● ⟷ ◉



1	10
2	5, 6
	9, 11
3	0, 4, 7
	1, 4, 6
	2, 5, 7
	3, 4, 11
	7, 8, 9
4	0, 1, 5, 7
	0, 2, 4, 6
	0, 3, 7, 9
	0, 4, 8, 11
	1, 2, 4, 7

● ⟷ ○

4	1, 3, 5, 11
	1, 3, 6, 9
	2, 5, 8, 11
	2, 6, 8, 9
	3, 4, 7, 8
5	0, 1, 5, 8, 11
	0, 1, 6, 8, 9
	0, 2, 3, 5, 11
	0, 2, 3, 6, 9
	1, 2, 3, 7, 9
	1, 2, 4, 8, 11
	1, 3, 5, 7, 8
	2, 3, 4, 6, 8

● ⟷ ○

2	1, 3
	2, 8
	5, 9
	6, 11
3	0, 1, 8
	0, 2, 3
	1, 4, 9
	2, 7, 11
	3, 4, 5
	5, 10, 11
	6, 7, 8
	6, 9, 10
4	0, 1, 7, 11
	0, 2, 4, 9

◉ ⟷ ◎

4	0, 3, 6, 7
	0, 4, 5, 8
	1, 4, 10, 11
	2, 7, 9, 10
	3, 4, 6, 10
	5, 7, 8, 10
5	0, 1, 7, 9, 10
	0, 2, 4, 10, 11
	0, 3, 5, 7, 10
	0, 4, 5, 7, 11
	0, 4, 6, 7, 9
	0, 4, 6, 8, 10
	1, 4, 7, 8, 10
	2, 3, 4, 7, 10

◉ ⟷ ◎

Figure 11. The Three Pictures on the Left Show Planar Representations of the Tetrahedron, the Cube and the Octahedron, along with an Arbitrary Labeling of Their Edges. The Tables on the Right Give the List of the Self-avoiding Paths Connecting the Two Vertices (or Pairs of Self-avoiding Paths Connecting the Three Vertices) Represented by the Black, Dark Grey, Light Grey And/Or White Dots in the Pictures. Each Path Is Represented by the Collection of Its Edges Using the Labels Shown in the Pictures. The Numbers in the First Column of Each Table Indicate the Length of the Paths.

for short, one can easily check that

$$\begin{array}{lllll}
|\pi_1| = 1 & |\pi_{1,2}| = 3 & |\pi_{1,2,3}| = 5 & |\pi_{1,2,3,4}| = 6 & |\pi_{1,2,3,4,5}| = 6 \\
|\pi_2| = 2 & |\pi_{1,3}| = 3 & |\pi_{1,2,4}| = 5 & |\pi_{1,2,3,5}| = 6 & \\
|\pi_3| = 2 & |\pi_{1,4}| = 4 & |\pi_{1,2,5}| = 5 & |\pi_{1,2,4,5}| = 6 & \\
|\pi_4| = 3 & |\pi_{1,5}| = 4 & |\pi_{1,3,4}| = 5 & |\pi_{1,3,4,5}| = 6 & \\
|\pi_5| = 3 & |\pi_{2,3}| = 4 & |\pi_{1,3,5}| = 5 & |\pi_{2,3,4,5}| = 5 & \\
 & |\pi_{2,4}| = 4 & |\pi_{1,4,5}| = 6 & & \\
 & |\pi_{2,5}| = 4 & |\pi_{2,3,4}| = 5 & & \\
 & |\pi_{3,4}| = 4 & |\pi_{2,3,5}| = 5 & & \\
 & |\pi_{3,5}| = 4 & |\pi_{2,4,5}| = 5 & & \\
 & |\pi_{4,5}| = 5 & |\pi_{3,4,5}| = 5 & &
\end{array}$$

This, together with (1.50), implies that, for all $x \neq y$,

$$\begin{aligned}
P(x \leftrightarrow y) &= (p + 2p^2 + 2p^3) - (2p^3 + 7p^4 + p^5) + (9p^5 + p^6) - (p^5 + 4p^6) + p^6 \\
&= p + 2p^2 - 7p^4 + 7p^5 - 2p^6 = (0, 1, 2, 0, -7, 7, -2) \cdot P_6.
\end{aligned} \tag{1.52}$$

Using also (1.51) and that $N_1 = 3$ for the tetrahedron, we conclude that

$$E(S_4) = 1 + 3\,(0, 1, 2, 0, -7, 7, -2) \cdot P_6 = (1, 3, 6, 0, -21, 21, -6) \cdot P_6$$

which proves the first part of Theorem 1.

To compute the second moment, we observe that any three distinct vertices of the tetrahedron always form a triangle (regardless of the choice of the vertices) and, for all $x \in \mathscr{V}$,

$$\begin{aligned}
\text{card}\,\{(y, z) \in \mathscr{V}^2 : \text{card}\,\{x, y, z\} = 2\} &= 3 \times 3 = 9 \\
\text{card}\,\{(y, z) \in \mathscr{V}^2 : \text{card}\,\{x, y, z\} = 3\} &= 3 \times 2 = 6.
\end{aligned}$$

Using also (1.44) with $k = 2$, we get

$$E(S_4^2) = P(x \leftrightarrow x) + 9P(x \leftrightarrow y) + 6P(x \leftrightarrow y, x \leftrightarrow z) \tag{1.53}$$

where vertices x, y, z are arbitrary but all three distinct. In addition, letting $\gamma_1, \gamma_2, \ldots, \gamma_K$ be the pairs of self-avoiding paths connecting all three vertices, and using the same argument as before based on the inclusion-exclusion identity, we get

$$P(x \leftrightarrow y, x \leftrightarrow z) = \sum_{B \subset [K] : B \neq \varnothing} (-1)^{\operatorname{card}(B)+1} \; p^{\operatorname{card}\left(\bigcup_{i \in B} \gamma_i\right)} \tag{1.54}$$

which can be viewed as the analog of (1.50). For the tetrahedron, there are $K = 10$ such paths (see the second table in Figure 11). As previously, computing

$$\operatorname{card}\left(\bigcup_{i \in B} \gamma_i\right) \quad \text{for every } B \subset [10] = \{1, 2, \ldots, 10\}$$

is straightforward in the sense that it does not require any abstract reasoning. However, having ten self-avoiding paths, the sum in (1.54) is now over

$$2^{10} - 1 = 1{,}023 \ \text{ terms}$$

and is therefore unrealistic to compute by hand. Also, to compute (1.54), we designed a computer program that goes through all the possible subsets $B \subset [10]$ and returns seven (= 1 + number of edges of the tetrahedron) coefficients $a_0, a_1, \ldots, a_6$. These seven coefficients are initially set to zero and increase or decrease by one according to the following algorithm:

$$\begin{aligned} &\text{replace } a_j \to a_j + 1 \text{ each time } \operatorname{card}\left(\textstyle\bigcup_{i \in B} \gamma_i\right) = j \text{ and } \operatorname{card}(B) \text{ is odd} \\ &\text{replace } a_j \to a_j - 1 \text{ each time } \operatorname{card}\left(\textstyle\bigcup_{i \in B} \gamma_i\right) = j \text{ and } \operatorname{card}(B) \text{ is even.} \end{aligned} \tag{1.55}$$

In other words, because the tetrahedron contains six edges, the right-hand side of (1.54) is a polynomial with degree at most six, and the algorithm returns the value of the seven coefficients of this polynomial. We point out that the values we obtain are exact because the computer is simply used to add a large number of integers, not to simulate the percolation process. The input of the program is the ten self-avoiding paths represented by the subsets of edges in the second table of Figure 11, and the output of the program is

$$a_0 = 0, \quad a_1 = 0, \quad a_2 = 3, \quad a_3 = 5, \quad a_4 = -18, \quad a_5 = 15, \quad a_6 = -4.$$

This, together with (1.52) and (1.53), implies that

$$\begin{aligned} E(S_4^2) &= 1 + 9\,(0,1,2,0,-7,7,-2) \cdot P_6 + 6\,(a_0,a_1,a_2,a_3,a_4,a_5,a_6) \cdot P_6 \\ &= 1 + 9\,(0,1,2,0,-7,7,-2) \cdot P_6 + 6\,(0,0,3,5,-18,15,-4) \cdot P_6 \\ &= (1,9,36,30,-171,153,-42) \cdot P_6. \end{aligned}$$

This completes the proof of Theorem 1.

1.3.5 Proof of Theorem 2

The idea is again to compute the sum (1.51) explicitly by first collecting the self-avoiding paths connecting two vertices.

Cube. For the cube, there are respectively fifteen, sixteen and eighteen self-avoiding paths connecting any two vertices at distance one, two, and three from each other, as shown in Figure 11. Because the cube has twelve edges, the sum consists of a polynomial with degree 12. The first four columns in the first table of Figure 12 show the coefficients. The first column means that, with probability one, a vertex is in the open cluster starting from itself while the second column means that a vertex at

	$s=0$	$s=1$	$s=2$	$s=3$	
	$N_s=1$	$N_s=3$	$N_s=3$	$N_s=1$	$E(S_6)$
0	1	0	0	0	1
1	0	1	0	0	3
2	0	0	2	0	6
3	0	2	0	6	12
4	0	−2	5	0	9
5	0	8	−4	0	12
6	0	−15	−5	−21	−81
7	0	−5	−22	6	−75
8	0	0	24	−3	69
9	0	67	62	86	473
10	0	−99	−115	−135	−777
11	0	55	68	78	447
12	0	−11	−14	−16	−91

	$s=0$	$s=1$	$s=2$	
	$N_s=1$	$N_s=4$	$N_s=1$	$E(S_8)$
0	1	0	0	1
1	0	1	0	4
2	0	2	4	12
3	0	3	8	20
4	0	0	−14	−14
5	0	−39	−40	−196
6	0	−1	16	12
7	0	257	288	1316
8	0	−540	−655	−2815
9	0	538	672	2824
10	0	−297	−376	−1564
11	0	88	112	464
12	0	−11	−14	−58

Figure 12. Coefficients Returned by Algorithm (1.55) for the Dodecahedron (Left) and the Icosahedron (Right) Using the Self-avoiding Paths Represented in Figure 11. The Last Column of Each Table Is Equal to the Linear Combination of the Other Columns with Weight given by the Value of The N_s in the Second Row. Because We Only Look at a Subset of the Self-avoiding Paths Connecting Two Vertices the Last Column Now Gives the Coefficients of a Polynomial In p That Is Smaller than the First Moment of the Cluster Size.

distance one from x is in the open cluster starting at x with probability

$$\begin{aligned}&(0,1,0,2,-2,8,-15,-5,0,67,-99,55,-11)\cdot P_{12}\\&\quad= p+2p^3-2p^4+8p^5-15p^6-5p^7+67p^9-99p^{10}+55p^{11}-11p^{12}.\end{aligned}$$

The second row in the first table of Figure 12 shows the value of N_s for the cube. The last column is the linear combination of the first four columns where column s has weight N_s. By (1.51), this is the expected value of the cluster size so the proof for the cube is complete.

Octahedron. Because the radius of the octahedron is two, two distinct vertices can only be at distance one or two apart. There are respectively twenty-six and twenty-eight self-avoiding paths connecting any two vertices at distance one and two from each other (see Figure 11). The sum again consists of a polynomial with degree 12, the common number of edges in the cube and the octahedron, whose coefficients are reported in the second table of Figure 12. The rest of the proof is exactly the same as for the cube.

1.3.6 Proof of Theorem 3

For the dodecahedron and the icosahedron, not only the sum (1.51) cannot be computed by hand, but also the number of self-avoiding paths connecting two vertices is beyond human capability. However, we can find lower bounds for the mean cluster size by only taking into account a subset of paths. More precisely, given $x \neq y$, and letting

- $\pi_1, \pi_2, \ldots, \pi_J$ be the self-avoiding paths of length $\leq c$ connecting x and y,
- $\pi_{J+1}, \pi_{J+2}, \ldots, \pi_K$ be the self-avoiding paths of length $> c$ connecting x and y,

we deduce from (1.50) that

$$\begin{aligned} P(x \leftrightarrow y) &= \sum_{B \subset [K] : B \neq \varnothing} (-1)^{\operatorname{card}(B)+1} \, p^{\operatorname{card}\left(\bigcup_{i \in B} \pi_i\right)} \\ &\geq \sum_{B \subset [J] : B \neq \varnothing} (-1)^{\operatorname{card}(B)+1} \, p^{\operatorname{card}\left(\bigcup_{i \in B} \pi_i\right)}. \end{aligned} \tag{1.56}$$

The inequality follows from an inclusion of events: if at least one of the first J paths is open then at least one of the K paths is open. For both the dodecahedron and the icosahedron, we choose the cutoff c to be the radius of the graph, meaning that

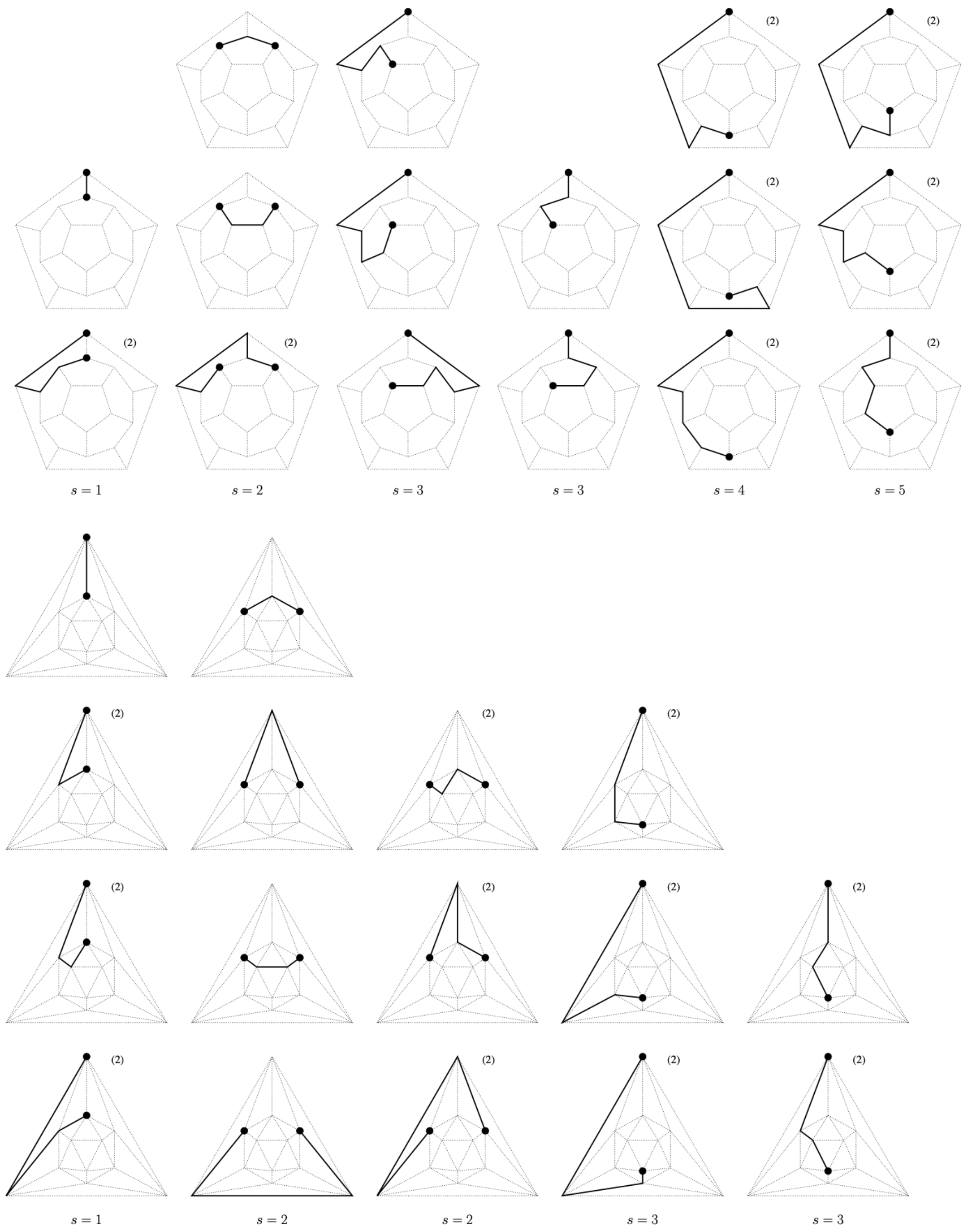


Figure 13. Picture of the Self-avoiding Paths with Length at Most Five Connecting Two Vertices of the Dodecahedron at Respectively Distance 1, 2, 3, 4, and 5, of Each Other, and Picture of the Self-avoiding Paths with Length at Most Three Connecting Two Vertices of the Icosahedron at Respectively Distance 1, 2, and 3, of Each Other. The Label (2) next to Some Pictures Means That the Mirror Image of the Path Is Another Path Connecting the Same Two Vertices.

we only consider self-avoiding paths with length at most five for the dodecahedron and self-avoiding paths with length at most three for the icosahedron. These paths are drawn in Figure 13. Because both graphs have thirty edges, the right-hand side of (1.56) is a polynomial with degree at most 30. The probabilities that two vertices at a given distance from each other are connected by an open path are reported in Figure 14. As previously, multiplying each column by the appropriate N_s listed in the first row of each table gives the coefficients of the polynomial on the right-hand side of (1.56), which completes the proof.

	$s=0$	$s=1$	$s=2$	$s=3$	$s=4$	$s=5$	
	$N_s=1$	$N_s=3$	$N_s=6$	$N_s=6$	$N_s=3$	$N_s=1$	$E(S_{12})\geq$
0	1	0	0	0	0	0	1
1	0	1	0	0	0	0	3
2	0	0	1	0	0	0	6
3	0	0	1	1	0	0	12
4	0	2	0	2	2	0	24
5	0	−2	1	2	4	6	30
6	0	0	−2	−2	0	0	−24
7	0	0	0	−2	−6	0	−30
8	0	−1	−2	−3	−1	0	−36
9	0	1	2	1	−4	−6	3
10	0	0	0	−1	3	−9	−6
11	0	0	0	4	6	0	42
12	0	0	0	0	−2	0	−6
13	0	0	0	1	2	6	18
14	0	0	0	−3	−5	12	−21
15	0	0	0	1	2	2	14
16	0	0	0	0	0	0	0
17	0	0	0	0	0	−6	−6
18	0	0	0	0	0	−9	−9
19	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0
21	0	0	0	0	0	6	6
22	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0
24	0	0	0	0	0	−1	−1

	$s=0$	$s=1$	$s=2$	$s=3$	
	$N_s=1$	$N_s=5$	$N_s=5$	$N_s=1$	$E(S_{20})\geq$
0	1	0	0	0	1
1	0	1	0	0	5
2	0	2	2	0	20
3	0	2	8	10	60
4	0	−9	−9	0	−90
5	0	1	−14	−10	−75
6	0	4	3	−35	0
7	0	8	28	10	190
8	0	−12	−2	60	−10
9	0	4	−28	40	−80
10	0	0	3	−75	−60
11	0	0	20	−90	10
12	0	0	−12	55	−5
13	0	0	2	110	120
14	0	0	0	−35	−35
15	0	0	0	−88	−88
16	0	0	0	35	35
17	0	0	0	40	40
18	0	0	0	−35	−35
19	0	0	0	10	10
20	0	0	0	−1	−1
21	0	0	0	0	0
22	0	0	0	0	0
23	0	0	0	0	0
24	0	0	0	0	0

Figure 14. Coefficients Returned by Algorithm (19) for the Dodecahedron (Left) and the Icosahedron (Right) Using the Self-avoiding Paths Represented in Figure 8. The Last Column of Each Table Is Equal to the Linear Combination of the Other Columns with Weight given by the Value of the N_s in the Second Row. Because We Only Look at a Subset of the Self-avoiding Paths Connecting Two Vertices the Last Column Now Gives the Coefficients of a Polynomial in p That Is Smaller than the First Moment of the Cluster Size.

Chapter 2

JOINT MODELING OF HLF AND SYBIL ATTACK: PETRI NET APPROACH

2.1 Introduction

Hyperledger Fabric (HLF) is an open-source enterprise-grade permissioned distributed ledger technology (DLT) platform established under the Linux Foundation. It is governed by a diverse technical steering committee and has a large community of over 35 organizations and nearly 200 developers since its earliest commits [13]. Among global corporate brands that use HLF technology, the adopters' list includes at least 20 largest global companies, including Allianz SE, Amazon, BNP Paribas, Intel, Microsoft, Siemens, State Farm, Visa, and Walmart [14]. Their economic activities represent an important part of total economic activity, and the prospect of increasingly channeling it through blockchain technology promises a fantastic future for this technology and its reach.

Unfortunately, the advent of this new technology is rightfully accompanied by the realization of its potential cyber vulnerabilities and worries about their future implications for the stability of our economic system and economic transactions relying on this new technology. To understand the backdrop of the emergency of this new technology, one needs to go no further than the summary of the Government Accountability Office, which notes that federal executive branch civilian agencies reported over 28,000 cybersecurity-related incidents to the Department of Homeland Security in the fiscal year 2019 [15].

When it comes to understanding the cyber vulnerabilities of HLF, or more general permissioned blockchains, despite the growing need and certain urgency given the current level of adoption, the present state of academic literature is its nascent form and offers little guidance. That is why the purpose of this report is to introduce the approach, based on graph-theoretical mathematical formalism for modeling discrete event systems, to improve the understanding of HLF behavior in the context of ongoing cyber attacks. This work's significant implications shed light on the interplay between the implementation of HLF and the implementation of a cyber attack in the context of the resilience of HLF which we view as the structural properties, policy adoptions and network management of a HLF's implementation leading to lower risk for potential cyber attacks, and the effectiveness of the Sybil cyber attack as our example of choice. To our knowledge, no other works of similar nature exist in current academic literature.

To achieve the stated purpose of this work we first use Petri Nets to model HLF. Next, we develop a model of a cyber attack. Finally, we present a joint model of HLF and a cyber attack. In the context of the simulation environment we develop to mimic real-world situations, we investigate the joint behavior of effectively two joint systems and see how their implementations affect mutual performance. The current literature that showcases practical mathematical modeling of HLF is reflected in papers [16], [17], and [18] that focus on understanding the performance of HLF configurations. The attention by Sukhwani et al. [18] is on performance modeling of the consensus phase under Practical Byzantine Fault Tolerance (PBFT) algorithm in HLF version v0.6. In turn, the work by Sukhwani et al. [16] presents a performance model of Hyperledger Fabric v1.0+ using Stochastic Reward Nets (SRN). In the context of their model, they compute the throughput, utilization, and mean queue length at each peer and processing stages within a peer. The authors in Yuan et al. [17], in the context of their HLF model, analyze the performance of the system by using Generalized Stochastic Petri Nets (GSPN). Their model decomposes a transaction flow into multiple phases and uses a simulation approach to find the system latency and throughput. Further, they analyze the impact of different configurations of the

Ordering service on system performance to find out the bottleneck. Further HLF performance analysis can be found in [19], [20], [21], [22], [23], and [24]. Performance analysis on HLF version 1.4 include [19], [20]. With HLF version 2.0, [21] and in HLF version 2.2 [22]. HLF smart contract "chain code" performance analysis is studied in [25].

HLF capabilities for extending next-generation secure and intelligent communication in Cyber-Physical Systems (CPS), are studied in Lohachab et al. [26], where performance analysis to identify bottlenecks and best configurations are investigated for the energy trading CPS use case. More generally, research investigating detection, resilience and control against cyber attacks can be found in [27], [28], [29], [30]. In You et al. [27] methods for sensor-reading modification (SM) attacks are studied. Ultimately, a method is proposed that synthesizes a liveness-enforcing supervisor tolerant to SM attacks considering the CPS as a bounded Petri Net. The subsequent papers are outside the scope of Petri Net modeling formalism but within cybersecurity and explore a variety of attacks. The work in Zhao et al. [28] investigates load frequency schemes for resilience against DoS attacks in CPS. The class of false data injection attacks in CPS aims to damage CPS by injecting false data like sensor measurements or control signals. A defense framework can be found in Zhao et al. [29]. For modeling large complex embedded systems, the state explosion is a difficult problem for Petri Net representation for embedded systems. In Xia and Li [30] synthesis methods are proposed to avoid the state space explosion problem. For modeling blockchain-enabled IoT networks, Lee et al. [31] proposes a model of HLF latency and provides insight into minimizing the latency for HLF-enabled IoT which is critical for securely handling time-sensitive data. Within the field of Machine Learning, advancements in the sub-field of Deep Learning (DL) along with high-quality, publicly available datasets have contributed to improvements in methods to detect cyber attacks successfully in CPS. An excellent survey listing new DL architectures for detection of cyber attacks can be found in Zhang et al. [32].

The literature related to cyber vulnerabilities of HLF is very scarce. The hypothesized attacks and potential vulnerabilities of HLF are presented in the works of Dabholkar and Saraswat [33], Hasanova et al. [34], Lagarde [35], Benedikt and Günther [36], and Wang [37]. The work of Dabholkar and Saraswat [33] lists or hypothesizes attacks when a membership service provider is compromised or in the presence of malicious ordering service, malicious validating nodes, external attacks, protocol-based attacks, chain code attacks, and implementation/architectural attacks. The work of Lagarde [35] provides an excellent concrete overview of cybersecurity vulnerabilities for HLF on various layers and components of the platform. It includes specific cybersecurity recommendations at the protocol level as well as at the architectural level. A discussion on the security implications arising from working in a permissioned DLT setting can be found in the work of Benedikt and Günther [36]. Key observations from their work are that the challenges emanating "from the inside" in a permissioned DLT setting, referred to as insider threats, are intertwined with the level of trust in architectural components and software layers of the platform, and therefore these technologies should not be regarded as "trust-free". The authors investigate how "insiders" can exploit trust and carry out attacks on the platform. In the work of Wang [37] the impact of malicious behavior is analyzed in the specific context of HLF. The author designs malicious behavior patterns and tests the blockchain performance of HLF under scenarios like PBFT with malicious behavior, denial-of-service attacks specifically high-rate spam transactions, and transactions with infinite loops. In the context of Petri Net approaches, when it comes to high-level modeling of cyberattacks, mostly on public blockchains, only the paper of Shahriar et al. [38] offers guidance. In terms of modeling complexity, amiability to our modeling approach, and its impact, the Sybil attack, in our view, takes an important role. In this attack, the attacker undermines a peer-to-peer network's reputation system and uses a large number of pseudonymous identities to achieve its goals [33]. That is why in this work, as a leading example of our approach, we use the Sybil attack as an attack of choice for the investigated joint model.

The main contributions of this paper are two-fold: (1) To expand the HLF Petri Net modeling literature by abstracting HLF's network transaction flow using a realistic majority endorsement policy under a more recent version of HLF v2.2. (2) Introduce a novel joint model of HLF under a Sybil attack scenario along with extensive numerical simulations of the joint model involving numerous parameter variations and network structure considerations to quantify the level of performance impact that the Sybil attack exerts on the HLF network.

The remainder of the paper is organized as follows. In Section 2, we briefly describe Hyperledger Fabric Version V2.x with an emphasis on transaction flow. In Section 2, we explain the basics of Petri Net mathematical formalism used in the modeling of HLF. Section 2 presents a few facts regarding the evolution of the HLF platform and a high-level overview of the steps involved in creating a HLF network. Section 2 presents our approach for Joint Modeling of Hyperledger Fabric and Sybil attack. In Section 2, we describe our simulation settings, which are used as a basis for numerical experiments whose results are presented in Section 2. Next, in Section 2, we discuss the HLF's model parametrization and validation. Finally, we conclude this work in Section 2 with a summary of our main findings and give suggestions for the continuation of the research.

2.2 Hyperledger Fabric Version V2.x Description

HLF was designed from the very beginning for enterprise use. One of the most critical features of HLF is the facilitation of a network of networks via channels. Generally, participants are not anonymous, and while absolute trust may not be present between participants, the HLF architecture allows for operation under a governance model built on the degree of trust that is present between them. The concept of "channels" aids when a subset of participants of the original network identifies additional applications of interests such that transactions stemming from these newly identified applications be kept private. HLF provides the capability for channel creation for this new activity of transactions. This new channel acts as a subnetwork of the HLF network and includes a separate ledger thereby guaranteeing data privacy of transactions involving the members of the channel from the rest of the HLF network. Multiple channels can be created in a HLF network and intuitively this corresponds to multiple subnetworks of clients within the HLF network. To summarize more precisely, according to the HLF documentation "a channel is a private blockchain overlay which allows for data isolation and confidentiality. A channel-specific ledger is shared across the peers in the channel, and the transacting parties must be authenticated to a channel in order to interact with it" [39].

Hyperledger Fabric has a highly modular and configurable architecture by design. It supports pluggable consensus protocols that allow the platform to adjust to many use cases and trust models [13].

When it comes to Hyperledger Fabric's Transaction Flow, it follows the execute-validate-order approach that can be summarized as follows.

- Transaction proposal: Channel client creates and submits a transaction proposal to peers designated with the endorsing role.
- Endorsement: The transaction proposal is simulated in each endorsing peer for the correctness of the transaction proposal. The endorsing peers return the proposal response with the needed data along with its signature.
- Ordering: After the client has collected enough endorsements for its transaction proposal from network peers, the proposal is sent to the ordering service. The ordering service verifies the transaction proposal against the channel policies and definitions. After that, transactions are bundled into blocks thus putting order to transactions.
- Validation: The validating set of peers perform the validation logic. First, a validation system chaincode (VSCC) check is performed. After that, a Multi-version concurrency control check is also performed. These checks essentially validate the identities that signed the transaction, verify the signature of the endorsing peers, and verify that the transactions' endorsing policy matches the chaincode definition. Finally, the block is committed to the ledger and local copies are updated.

2.3 Petri Nets Basics

In this work, when it comes to modeling HLF, the Petri Net model is used. Within the modeling framework of discrete-event dynamic systems, a place/transition net is a mathematical model for the characterization of distributed systems. Introduced by Carl Adam Petri [40] in 1962 and more commonly known as Petri Nets (PN), these models consist of places, transitions, arcs, and tokens [41] whose intuitive meaning and visual characteristics can be summarized as follows:

- Places – visually represented as circular nodes, the places abstract logical conditions or system states.
- Transitions – visually represented via rectangular boxes, they denote transitions abstracting a change in the status of the system. When a transition is enabled and fires, it "transitions" the system into a new state. The transitions can be seen to represent some production process that transforms resources from one place and stores them in another.
- Arcs – visually represented asymmetrical links exclusively between places and transitions never between places or between transitions. They denote the explicit relationship between the objects of the system where resources/logic can be consumed from and produced/transferred to.
- Tokens – visually represented as dots, they represent a numerical quantity of resources and are expressed in form of tokens inside a place.

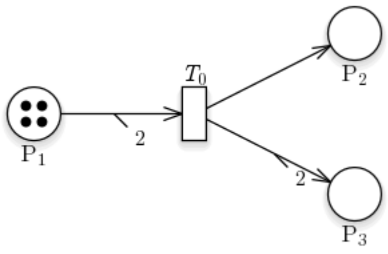
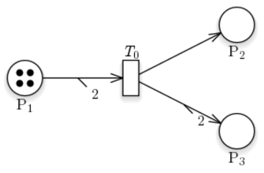


Figure 15. A Simple Petri Net Consisting of Three Places with Four Tokens in P_1, One Transition T_0 and Three Arcs

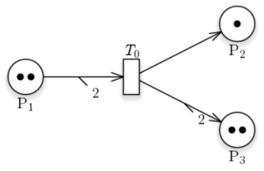
In graph-theoretic terms, Petri Nets are bipartite graphs. Formally, a Petri Net is defined as a tuple $N = (P, T, F, M_0)$ where P is a set of places, T is a set of transitions with P and T are disjoint sets. The function $F : (P \times T) \cup (T \times P) \rightarrow \mathbb{N}$ assigns a weight to each arc in the PN. Here M_0 is the initial marking, a function from the set of places to the positive integers. Markings $M_i, i \in \mathbb{N}$ represent the distribution/configurations of tokens over the Petri Net graph. The arcs connect a place to a transition or vice-versa. The places in the former case are called input places, and in the latter case, output places. Arcs can be assigned weights with non-negative integer values. A transition t becomes enabled when the number of tokens in the input place is at least the arc's weight, that is, when for all $s, M_i(P) \geq F(P, T)$. When the transition fires, it will consume a total number of tokens matching the arc weight from each input place $i, F(i, t)$ and produce a total number of tokens matching the weight of the arc into each output place $o, F(t, o)$. This setting represents the resource requirements for some production processes to take place. The distribution of tokens over the places represents a configuration or marking of the net.

For a concrete example of the basic PN dynamics consider Figure 16. The PN consists of three places, P_1, P_2, P_3, one transition T_0, and three arcs with cardinalities, $F((P_1, T_0)) = 2, F((T_0, P_2)) = 1, F((T_0, P_3)) = 2$.

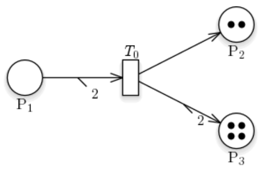
1. The initial marking $M_0 = (4, 0, 0)$ corresponds to four tokens in P_0, 0 tokens in P_1, and zero tokens in P_2. Transition T_0 is enabled since $M_0(P_0) = 4 \geq F((P_1, T_0)) = 2$.
2. After the first firing of T_0 the marking of the PN is $M_1 = (2, 1, 2)$. Transition T_0 is still enabled in the current marking since $M_1(P_0) = 2 \geq F((P_1, T_0)) = 2$.
3. After the second firing of transition T_0 the marking of the PN is $M_2 = (0, 2, 4)$. Transition T_0 is not enabled in the current marking since $M_2(P_0) = 0$ is less than $F((P_1, T_0)) = 2$.



(a) Initial Marking of PN



(b) Marking After First Firing of Transition T_0



(c) Marking After Second Firing of Transition T_0

Figure 16. Firing Sequence of Petri Net

Petri Net is a mathematical formalism with precise definitions and foundations but it also possesses intuitive graphical representation and visualization. There are a vast number of extensions and variations of Petri Nets each providing unique modeling power and sophistication [42]. In the following three subsections we introduce some of the most prominent classes of PN.

2.3.1 Stochastic Petri Net

Within the theory of Petri Nets, one of the most prominent examples is Stochastic Petri Net (SPN) which associates an exponential distribution to the firing of transitions thereby establishing a link with the theory of Markov Chains. Formally, a stochastic Petri Net $SPN = (PN, \Lambda)$ is formed from a Petri Net $PN = (P, T, F, M_0)$ by augmenting the definition to include the set $\Lambda = (\lambda_1, ..., \lambda_m)$ as part of the definition. The λ_i is the transition rate of transition t_i. Thus the firing time is exponentially distributed and the cumulative distribution of χ_i the firing time of transition t_i is given by

$$F_{\chi_i}(x) = 1 - e^{-\lambda_i x}.$$

SPNs are used to generate the, usually large, Markov chain based on a concise description of the system using the SPN modeling language. This allows the study of the corresponding Markov model of the system. One major challenge of Markov models is the cardinality of their state space. Deciding the appropriate PN formalism for modeling is one of the steps manifesting a great impact in regards to determining model complexity and state space largeness.

2.3.2 Generalized Stochastic Petri Net

An extension to SPN, the class of Generalized Stochastic Petri Net (GSPN) includes additional features to further extend the modeling power beyond SPNs. GSPNs add inhibitor arcs and immediate transitions. An inhibitor arc is an arc from a place to a transition that inhibits the firing of the transition when a token is present in the input place.

Given a marking of the PN, more than one transition may be simultaneously enabled. The tie between simultaneously enabled transitions can be broken by specifying priorities, by specifying probabilities, or by a race involving the corresponding exponential random variables of firing times i.e. $\min(F_{\chi_i}, F_{\chi_j}, ..., F_{\chi_k})$. Priorities are non-negative integers that determine an ordering of transitions. Whenever a transition with a priority μ is enabled, all transitions with priorities less than μ are inhibited from firing. If probabilities are assigned to the transitions, they are interpreted as the weights of each transition. With some probability, any of the enabled transitions may be the first to fire; any enabled transition with positive probability may fire subsequently. Immediate transitions which can be simultaneously enabled must have either priorities or probabilities specified in order to avoid errors in the definition of the PN. For timed transitions, the decision as to which transition fires next can be decided by a race; the transition with the minimal delay prior to firing will fire next. The markings of a GSPN are classified into two categories, tangible or vanishing. A marking is called tangible if the only transitions enabled are timed transitions, and a marking is called vanishing if one or more immediate transitions are enabled in the marking.

2.3.3 Stochastic Reward Net

Our modeling focuses on Stochastic Reward Nets (SRN) [43], a marking-dependent oriented formalism derived from GSPNs. SRNs substantially increase the modeling power of the GSPN by adding guard functions, marking dependent arc multiplicities, general transition priorities, and reward rates at the net level.

A guard function is a boolean function associated with the definition of a transition. Whenever the transition satisfies all the input and inhibitor conditions in a marking M, the guard is evaluated. The transition is considered enabled only if the guard boolean function evaluates to true.

Marking dependent arc multiplicities allow either the number of tokens required for the transition to be enabled, or the number of tokens removed from the input place, or the number of tokens placed in an output place to be a function of the current marking of the PN. Such arcs are called variable cardinality arcs. These capabilities provide powerful modeling flexibility to capture Hyperledger Fabric's modular architecture.

2.4 Hyperledger Fabric Platform

2.4.1 Version Releases

The earliest release of HLF dates back to September 2016 with release version v0.6. Currently, HLF stands with release v2.3, as of the time of current writing. Major milestones have been achieved over the years strengthening the security and design of the platform. In this section, we summarize the major changes to the platform.

- HLF v0.6 was shown in [37] vulnerable from attacks to the PBFT consensus as well as to variations of denial-of-service attacks.
- In HLF v1.0+, the execute-order-validate paradigm was adopted to help mitigate the scenarios like the ones mentioned above. But in HLF v1.0+ the issue of centralization of governance of chaincode (smart contracts) is still present.
- In HLF v2.0+, decentralized governance of chaincode was implemented. From this version onwards, multiple organizations must agree on the parameters of a chaincode definition. [13].

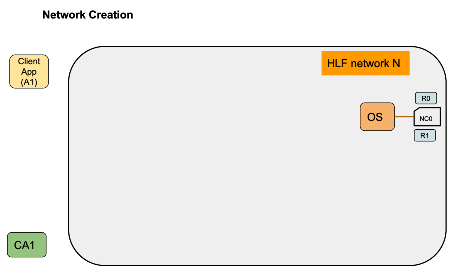
2.4.2 HLF Network Initialization

In this section, we describe the creation and initialization of the conceptual Hyperledger Fabric network leading to our proposed cybersecurity scenario. We refer the reader to the Hyperledger Fabric documentation [13] for a glossary of all the terms used throughout this paper. We remark that HLF, being a permissioned DLT platform, gains security from the construct of having verifiable identities and actors on the network. But the risk can emanate from poor governance, flawed policy administration, or configuration of the HLF network.

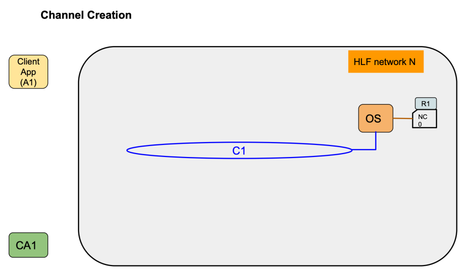
We now outline the steps involved in constructing the network with the cybersecurity scenario CSS2:

1. The ordering service (OS) is started by an administrator from organization $R0$. The configuration $NC0$ contains the policies that describe the initial administrative capabilities of the HLF network N.
2. Organization R0 updates the network configuration NC0 to give the organization R1 an administrator role. After this, both R0, R1 have equal rights over the HLF network N.
3. Network administrator R1 defines CA1 as the certificate authority for this organization.
4. A channel C1 has been created for R1 using the consortium definition X1. The channel is governed according to the channel configuration CC1 completely disjoint from the network configuration file NC0. CC1 is managed by R1 and R0 has no rights in channel C1.

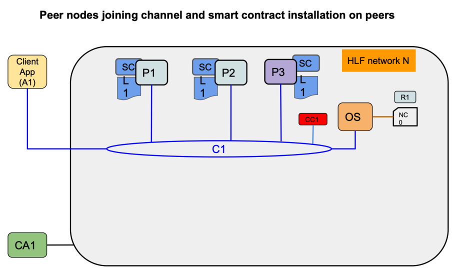
5. Peer nodes P1, P2, P3 (with designated endorsing roles) have joined channel C1. Peers P1, P2, P3 physically host a copy of the ledger L1. The set of peers and the ordering service can communicate with each other via channel C1. The certificate authority CA1 provides the certificates for the set of peers which determines their permission levels on channel C1.
6. A smart contract SC has been installed onto peers P1, P2, P3. Client application A1 in organization R1 can use SC to access the ledger via peer nodes P1, P2, P3.



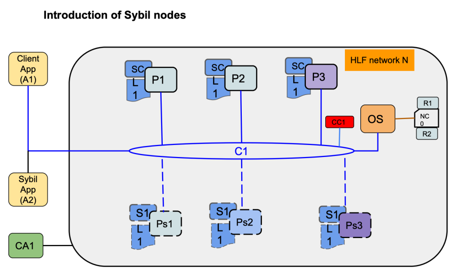
Steps 1, 2



Steps 3, 4


Peer nodes joining channel and smart contract installation on peers
Client App (A1)
SC
L 1
P1
P2
P3
HLF network N
R1
CC1
OS
NC 0
C1
CA1

Steps 5, 6



Step 7

Figure 17. HLF Network Initialization

The application A1, peers P1, P2, P3, and ordering service OS are all joined to channel C1 and thereby able to use resources from it.

7. Organization R2 is added to channel 1. Peers Ps1, Ps2, Ps3 (with designated endorsing roles) have joined channel 1.

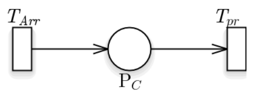
2.5 SRN Hyperledger Fabric Model

In this section, we introduce the Stochastic Reward Net model to Hyperledger fabric version V2.x under our proposed cyber-security scenario. The main advantages of using this Stochastic Petri Net formalism lie in the ability to easily add and remove system details which is crucial to study different scenarios given Hyperledger Fabric's architecture. We first describe the model for the HLF network with one regular client, then, by employing HLF's modular nature, we describe the updated HLF network, with overwritten policies, under which a malicious client has joined the channel along with a set of endorsing peers for his transaction proposals. The SRN model that captures this set-up is shown in Figure 20. We proceed in two stages: pre-cyber-security scenario (CSS1) and post-cyber-security scenario (CSS2).

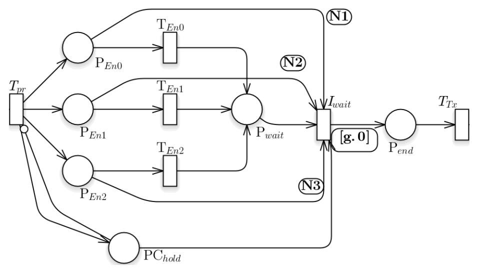
First, for CSS1, the SRN model is abstracting a regular functioning single-channel fabric network with one regular client, three endorsing peers, and a single peer running the validation logic. The endorsement policy has been established as a majority endorsement, that is, at least two out of the three peers must provide their endorsement to a transaction proposal. Figure 18 illustrates the decomposition of the model into four phases. In the remainder of this section, we describe each phase before presenting Figure 19 containing the diagram for the baseline model CSS1 of the HLF network.

Transaction requests from the regular client follow a Poisson arrival process with rate λ_{PC} and this phase corresponds to Figure 18 (a). The client prepares a transaction proposal, requesting endorsement from designated endorsing peers P_{En_i} for $i = 0, 1, 2$ (transition T_{Pr}, client processing time). The peers endorse the transaction proposal (transitions T_{En_i} for $i = 0, 1, 2$); this phase is shown in Figure 18 (b). Further details on how the majority endorsement policy is modeled with SRNs are given in subsection 2.

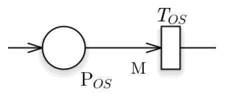
When the client receives enough endorsements, as specified by the endorsement policy, it sends the transaction proposal to the ordering service (transition T_{T_x}, transmission to ordering service, becomes enabled). In the channel configuration, a



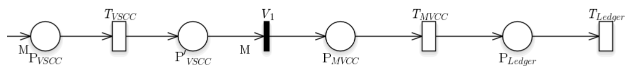
(a) Transaction Proposal



(b) Endorsement



(c) Ordering Service



(d) Block Validation & commit

Figure 18. Decomposition of HLF model

designated value for a bundle of transactions has been defined to be M, this is the block size. Now M pending transaction proposals are bundled into a "block" and sent to the committing peers of the channel (transition T_{OS}); this phase corresponds to Figure 18 (c).

In the final phase corresponding to Figure 18 (d), a Validation System Chaincode (VSCC, T_{VSCC}) check is performed to evaluate and confirm all the transactions endorsements in the block against the endorsement policy of the channel (the rate is limited by the CPU limitations of the peer). After, a Multi-version Concurrency Control check is performed to ensure that the versions of the keys read by the transaction during the endorsement phase are consistent with the current state at commit time (transition T_{MVCC}). At last, all transactions are written to the local copies of the ledger (T_{ledWrite}). We now present the complete diagram for the model made up of the previously presented parts.

For the second part CSS2, we consider the scenario where a second client (malicious

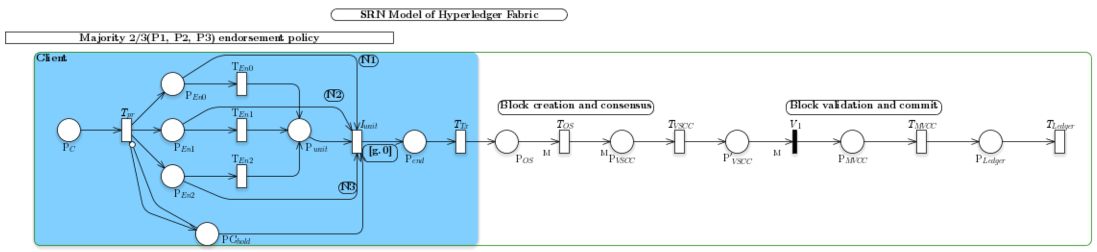


Figure 19. Petri Net Model for Baseline Scenario When There Is No Sybil Attack.

client) has joined the network and channel with appropriate permissions. Transaction requests from the malicious client follow a Poisson arrival process with rate λ_{PsC}. A set of three endorsing peers have joined the channel. The channel endorsing policy is updated to 2/3 of P_{En_i} OR 2/3 of Ps_{En_i}. The malicious client requests endorsement to its transaction proposals from the set Ps_{En_i}. After collecting the required endorsement, its transaction proposals are delivered to the channel's ordering service. The rest of the transaction flow of the network continues in the same way as described above for the HLF network in CSS1.

The complete diagram for this scenario is presented below in Figure 20

2.5.1 SRN Model Features

The main SRN features that allow multiple endorsement policies to be easily implemented in SPNP are: variable arcs and enabling guards.

These are defined in the SPNP manual 6.0 as viarc(t, p, func) and voarc(t, p, func). These functions define, respectively, an input arc from place p to transition t, an output arc from transition t to place p with multiplicity given by the marking-dependent function func.

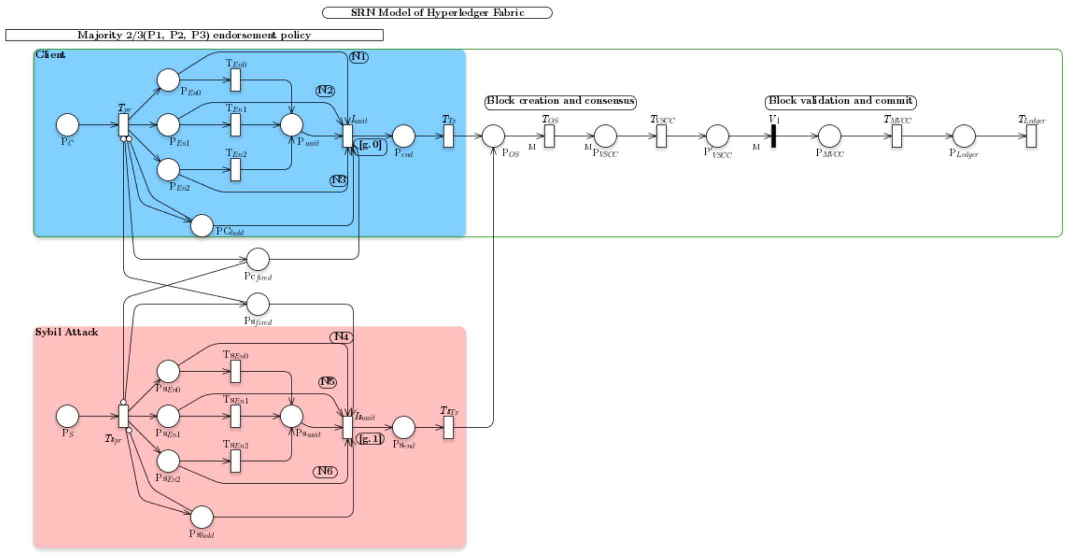

SRN Model of Hyperledger Fabric
Majority 2/3(P1, P2, P3) endorsement policy
Client
Sybil Attack
Block creation and consensus
Block validation and commit
N1
N2
N3
N4
N5
N6
[g. 0]
[g. 1]

Figure 20. HLF SRN Model of Sybil Attack Scenario.

The variable arc function definitions from Figure 19, 20 are:

$$N1 = \text{mark}(P_{En0}) \qquad N4 = \text{mark}(Ps_{En0})$$

$$N2 = \text{mark}(P_{En1}) \qquad N5 = \text{mark}(Ps_{En1})$$

$$N3 = \text{mark}(P_{En2}) \qquad N6 = \text{mark}(Ps_{En2})$$

The weight of the variable arcs above, $N1, ..., N6$, are defined to be the current mark of the input place.

The enabling guard definitions from Figure 19, 20 are:

$$[g.0] = \begin{cases} 1 & \text{mark}(P_{wait}) > 1 \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

$$[g.1] = \begin{cases} 1 & \text{mark}(Ps_{wait}) > 1 \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

Here the enabling guards (1), and (2) enforce the endorsement policy. Increasing the number in the logical expression with the mark of the place will translate to requiring more endorsements for a transaction proposal. In our case, we require at least two endorsements (out of three possible). The variable output arcs, $N1, ..., N6$, guarantee that as soon as the endorsement policy is satisfied the transition I_{wait} and Is_{wait} will become enabled therefore correctly capturing the majority endorsement policy since only a subset of peers are required to endorse. By leveraging variable arcs and enabling guards, numerous endorsement policies can be implemented without introducing much complexity: AND, OR, K/N, K/N OR K'/N', K/N AND K'/N'.

A Sybil attack involves creating a set of pseudoanonymous identities that joined the network in order to undermine the functionality of the network. By modifying the number of sybil peers (places Ps_i, and transitions Ts_i), the SRN models of HLF presented in this paper are able to capture a large space of Sybil attack configurations.

The structure of the SRN net from Figure 20 consists of: places (21), immediate transitions (1), timed transitions (16), constant input arcs (19), constant output arcs (24), and variable input arcs (8).

2.6 Simulations

Since the state space of the SRN is considerable, we must rely on discrete-event simulations using the software package SPNP 6.0 [44]. We carry out a set of experiments varying the following parameters rates and structure of the SRN as presented in Table 1:

- regular client transaction arrivals rate corresponding to the transition T_{Pr},
- malicious client transaction arrivals rate corresponding to the transition Ts_{Pr},
- regular client endorsement rate corresponding to the transitions T_{En_i},
- malicious client endorsement rate corresponding to the transitions T_{SEn_i},
- the number of regular client endorsing peers (in blue, places P_{En_i}), and

HLF Setup and parameters (variable)			
Component	Rate (s^{-1})	Mean Time (s)	Number
Regular Client Transactions arrival rate	180	0.0055	-
	200	0.0050	-
	220	0.0045	-
Regular Client Endorsement rate	270	0.0037	-
	300	0.0033	-
	330	0.0030	-
Number of Regular Client peers	-	-	3
	-	-	6
Malicious Client Transactions arrival rate	180	0.0055	-
	200	0.0050	-
	220	0.0045	-
Malicious Client Endorsement rate	270	0.0037	-
	300	0.0033	-
	330	0.0030	-
Number of Malicious Client peers	-	-	3
	-	-	6

Table 1. The Rate Parameters Varied and the Structural Variations of the SRN Net (Endorsing Peers) in the Set of Simulations.

- the number of malicious client endorsing peers (in red, places Ps_{En_i}).

During the simulations, the regular client submits a total of 200 transaction proposals. The malicious client submits a total of 50 transaction proposals. In the ordering phase a bundle of five transactions results in a block, so $M = 5$. Both clients wait for endorsement completion on a transaction proposal before requesting a new endorsement for new transactions. Further, both sets of endorsing phases are mutually exclusive in the sense that at a point in time only one of the two client's transaction proposals is in the endorsement phase.

The cumulative performance measures recorded for transitions are:

Simulation specific options - SPNP 6.0		
Option	Value	Description
IOP_SIMULATION	VAL_YES	Simulation procedure on
FOP_SIM_LENGTH	2000	Length of each simulation run
IOP_SIM_CUMULATIVE	VAL_YES	Cumulatively data from 0 to 2000
FOP_SIM_CONFIDENCE	0.9	Confidence interval
FOP_SIM_RUNS	100000	Max number of simulations runs
FOP_SIM_LENGTH	{0.1,0.2,...,4.8,4.9}	Length of each simulation run

Table 2. SPNP Options for Discrete-Event Simulations Used.

- Utilization – the weighted average (by the probability of each marking) that the function is enabled.
- Average throughput – defined for a transition as

$$E(T_s) = \sum_{i \in R(s)} p(i) \cdot \rho(s, i)$$

  where $R(a)$ is the subset of reachable markings in which s is enabled, $p(i)$ is the probability of the marking i and $\rho(s, i)$ is the rate of transition s in the marking i (SPNP marking-dependent function rateval in our set-up).

The simulation options for our sets of experiments are in Table 2 above. Since the structure of an SRN cannot vary dynamically, we implemented a python script to generate all the different configurations (C ANSI files) we are interested in. By varying the parameters listed above we obtain a total of 15,876 SRN nets that represent all the combinations from Table 1. We have focused on examining the average number of tokens passing through transitions I$_{wait}$ and Is$_{wait}$, which are the endorsed transactions from the endorsement phase that are being sent back to the client and ultimately to the ordering service, then to the rest of the network, to compare them as the parameters change. These are measured from simulation start time 0 to simulation run-time of 4.9 via discrete increments of 0.1.

HLF Setup and parameters (constant)			
Component	Rate (s^{-1})	Mean Time (s)	Number of Peers
OS	12	0.0833	1
VSCC	397	0.006	1
MVCC	196	0.005	1
Ledger write	48	0.0201	1

Table 3. The Rate Parameters Kept Constant During the Set of Simulations.

We seek to collect performance metrics under specific scenarios and gain insight into how the system performance is affected in the event of a cybersecurity risk scenario.

More precisely, letting $(AvT_{0.1}, AvT_{0.2}, ..., AvT_{4.9})$ be a vector containing the average throughput, AvT_i, at transitions I_{wait} for $i \in \{0.1, ..., 4.9\}$, $(0.1, 0.2, ..., 4.9)$ be the run-time increments , and $\#(I^i_{wait})$ represent the average number of tokens flowing through the transition at run-time i then

$$(AvT_{0.1}, AvT_{0.2}, ..., AvT_{4.9}) \cdot (0.1, 0.2, ..., 4.9)^T = (\#(I^{0.1}_{wait}), ..., \#(I^{4.9}_{wait}))$$

represents a vector of measurements of the average number of tokens flowing through the transition at run-time $i \in \{0.1, ..., 4.9\}$. We study this measurement for the impact on the transaction flow for different configurations of the HLF network over time in the next section. Table 3 provides a summary of parameters kept constant in the set of simulations.

2.7 Numerical Results

Our numerical investigations reveal the impact that changes in transaction arrival rate, peer endorsement rate, and the number of endorsement peers have[1]. The scenario labels for the simulation settings are given in Table 4 for Figures 21-27.

HLF Setup and parameters (variable)			
Component	Scenario Label	Rate (category)	Number
Regular Client Transactions arrival rate	RCTARL	Low	-
	RCTARN	Normal	-
	RCTARH	High	-
Regular Client Endorsement rate	RCERL	Low	-
	RCERN	Normal	-
	RCERH	High	-
Number of Regular Client peers	NRCP3	-	3
	NRCP6	-	6
Malicious Client Transactions arrival rate	MCTARL	Low	-
	MCTARN	Normal	-
	MCTARH	High	-
Malicious Client Endorsement rate	MCERL	Low	-
	MCERN	Normal	-
	MCERH	High	-
Number of Malicious Client peers	NMCP3	-	3
	NMCP6	-	6

Table 4. The Rate Parameters Categories and the Structural Variations of the SRN Net (Number of Endorsing Peers) in the Set of Simulations.

First, we consider the setting of 3 regular client peers (NRCP3), low regular client transaction arrival rate (RCTARL), and low malicious client transaction arrival rate (MCTARL). In this setting, we consider 3 and 6 malicious client peers (NMCP3 and NMCP6), as well as change malicious client endorsement rates from low, to normal and high (MCERL, MCERN and MCERH respectively). We present our finding in Figure 21 and Figure 22. Figure 21 shows that increasing the number of peers increases overall malicious transaction endorsement time. Also, increasing

[1]In the interest of effective visual presentation only the time till the average number of transactions reaches 49.5 for Sybil attack and 199.5 for regular transactions, will be considered.

the malicious peer endorsement rate increases the overall malicious transactions endorsement time, however, the impact of an increase of peers from 3 to 6 is larger. The relative impact of increase or decrease of malicious peers' endorsement rate is smaller for 6 peers than for 3 peers. Figure 22 shows that both increase in malicious peer transaction rate and an increase in the number of malicious peers positively affect the overall regular transactions processing time. This is an intuitive result as the sooner malicious transactions are endorsed less they can interfere with the endorsement of regular transactions. More malicious peers allow for faster reaching of consensus. The impact of their increase is found to be stronger. The kink in the slope of endorsement of regular transactions in time could serve as an indicator that malicious transactions have interfered with the rate of endorsement of regular transactions. Figure 23 shows the impact under, the same scenario settings, of the system with and without the cyberattack component.

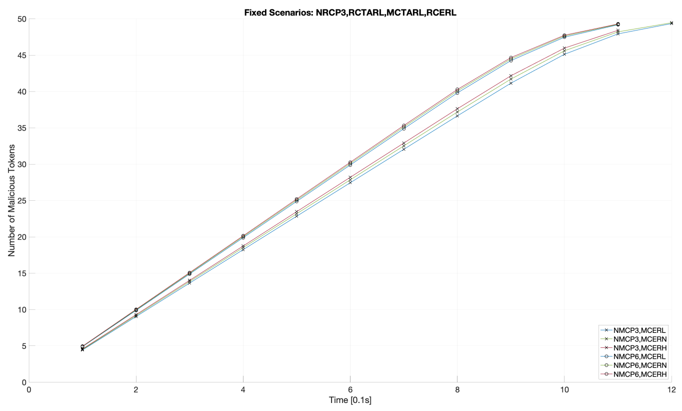


Figure 21. Average Endorsement Time of Malicious Transactions (Tokens) in Time - Comparison Between the Change of the Number of Malicious Peers Vs. Change in Malicious Peers Endorsement Rate.

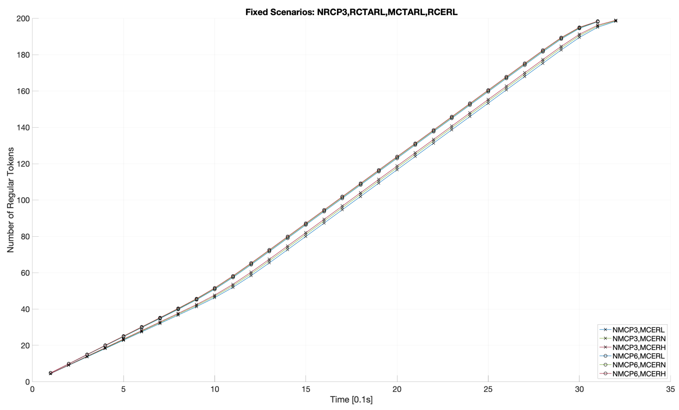


Figure 22. Average Endorsement of Regular Transactions (Tokens) in Time - Comparison Between the Change of the Number of Malicious Peers Vs. Change in Malicious Peers Endorsement Rate.

Second, we consider setting of 3 malicious client peers (MRCP3), low regular client transaction arrival rate (RCTARL) and low malicious client transaction arrival rate (MCTARL) as well as low regular and malicious transactions endorsement rate (RCERL and MCERL). In this setting, we consider 3 and 6 regular client peers (NRCP3 and NRCP6). We present our finding in Figure 24c. Figure 24a shows that increase in the number of regular peers has a positive overall effect on the processing time of a majority of regular transactions. However, from Figure 24b we learn that there will be also a positive overall impact on the overall endorsement time of most malicious transactions. This is intuitive, and due to the symmetry of our scheme. The faster the majority consensus is achieved, the faster the regular transactions will be endorsed and the less they will interfere endorsement of malicious transactions.

Third, we consider the setting of 3 regular and 3 malicious client peers (NRCP3 and NMCP3), low regular client transaction arrival rate (RCTARL), and low malicious client transaction arrival rate (MCTARL) as well as low malicious transactions endorsement rates (MCERL). In this setting, we consider low and high regular client

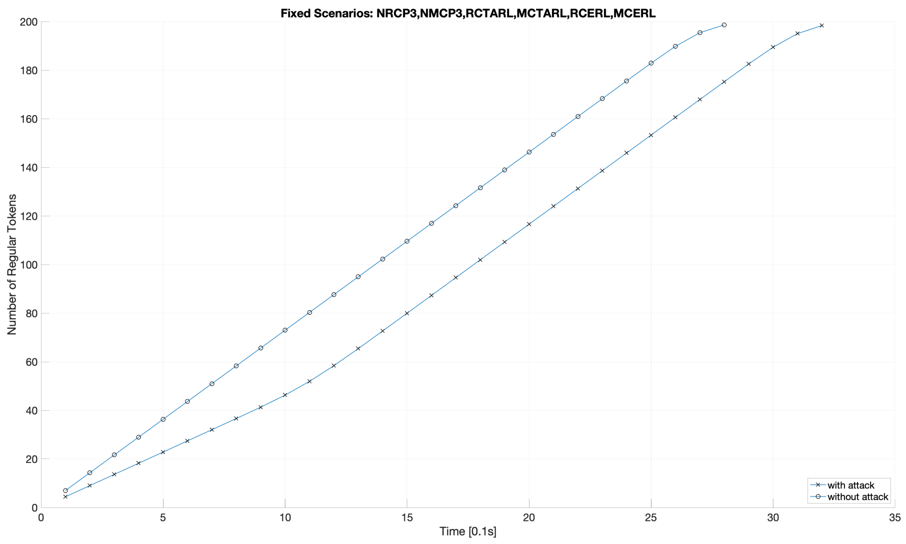
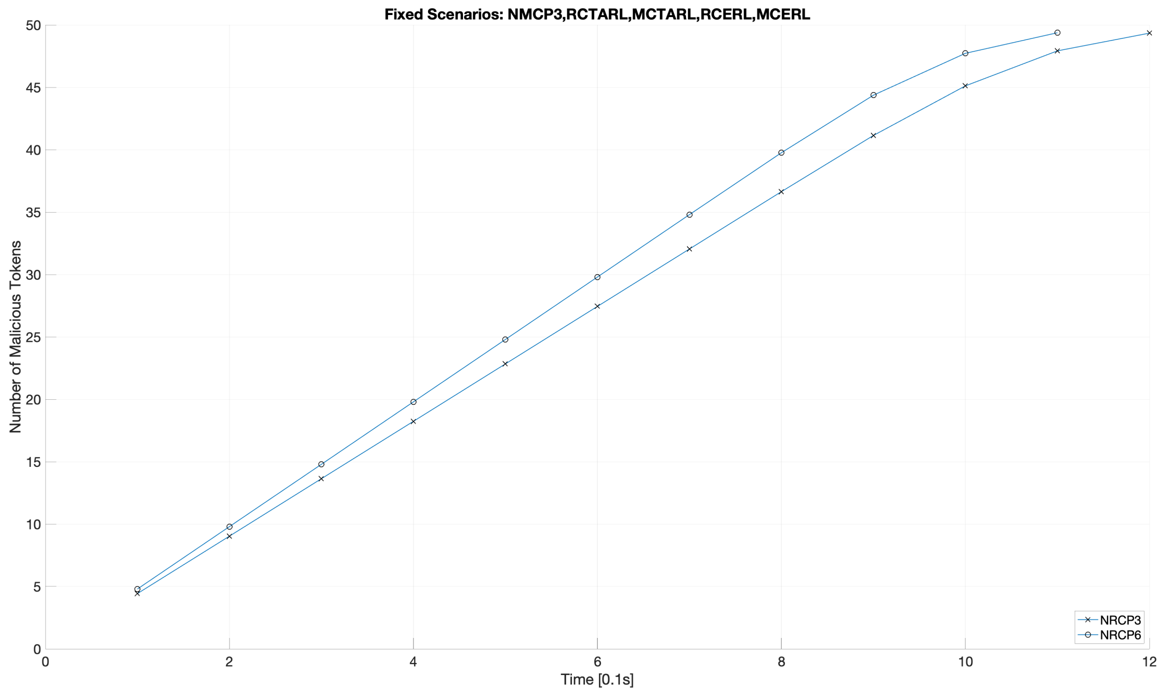


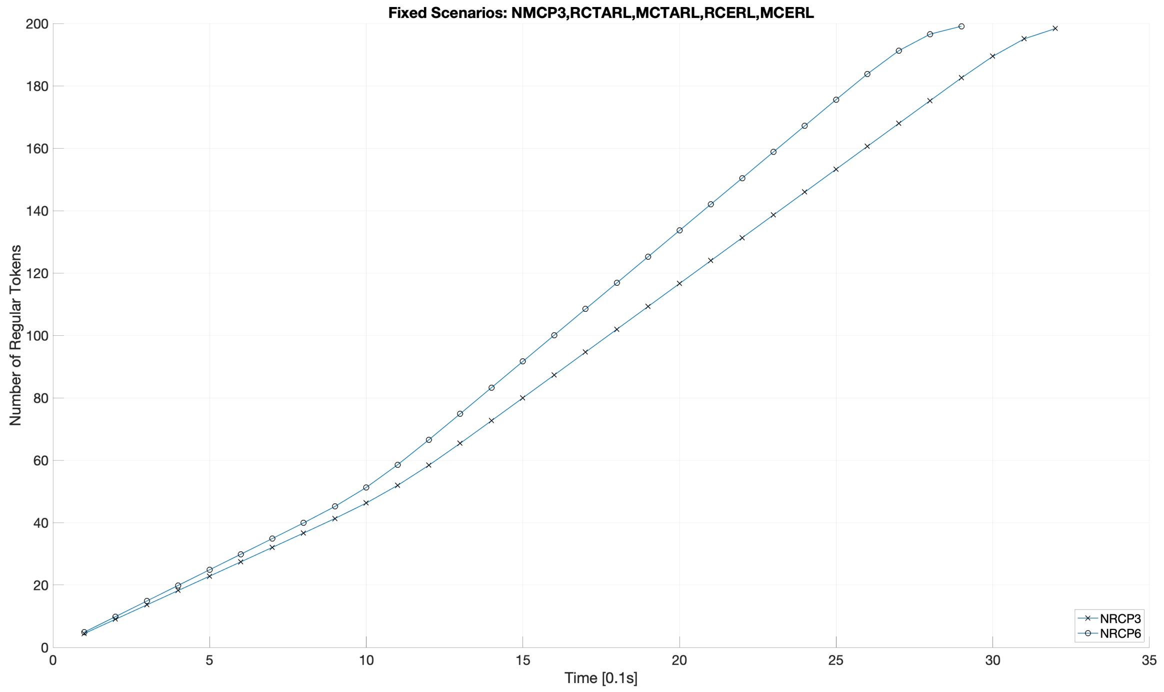
Figure 23. Average Endorsement of Regular Transactions (Tokens) in Time - with and Without Cyber Attack Component.

endorsement rates (RCERL and RCERH). We present our finding in Figure 25. Figure 25a shows that an increase in the regular client endorsement rate has a positive overall effect on the processing time of a majority of regular transactions. For similar reasons as in the previous paragraph Figure 25b show a positive impact on the overall malicious client's endorsement rate. However, comparing Figures 24c and 25 we see that impact created by the increase in the number of regular clients is stronger compared to the increase in regular client endorsement rate.

Fourth, we consider setting of 3 regular and 3 malicious client peers (NRCP3 and NMCP3), low regular client transaction arrival rate (RCTARL) as well as low regular and malicious transactions endorsement rate (RCERL and MCERL). In this setting, we consider low and high malicious client transaction arrival rates (MCTARL and MCTARH). We present our findings in Figure 26. Figure 26a shows a positive impact on the overall endorsement time of malicious transactions as expected. However, Figure 26b shows a negative impact on endorsement times of part of the regular

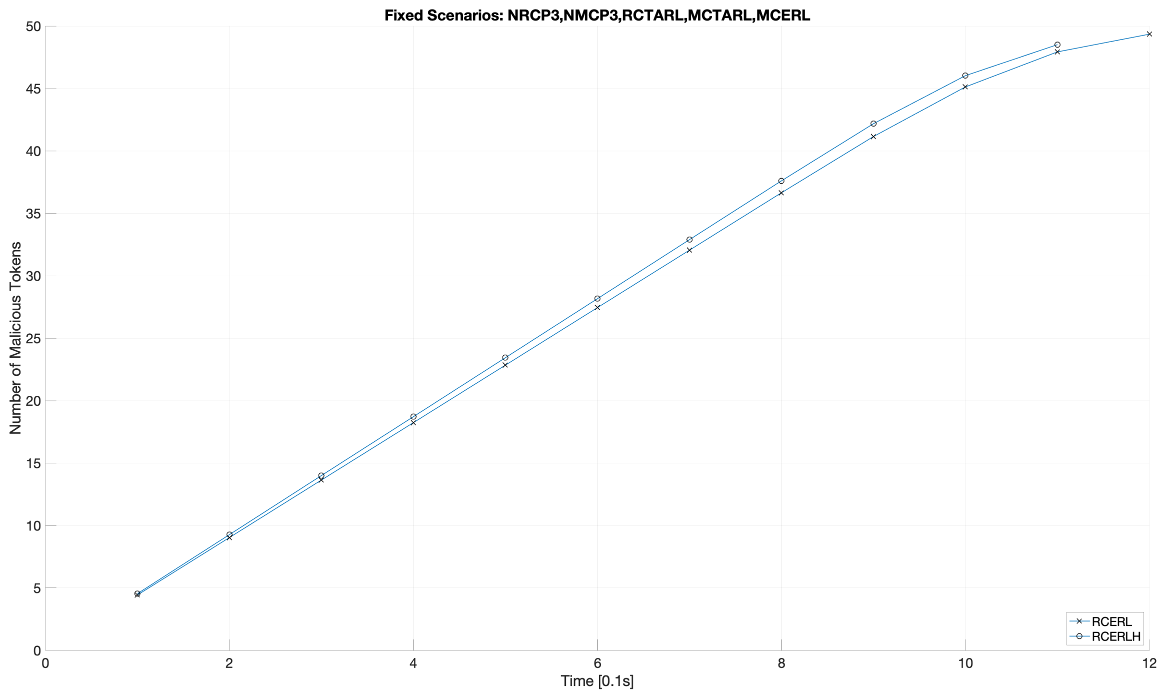


(a) Malicious Transactions - Impact of Number of Regular Client Peers Change.

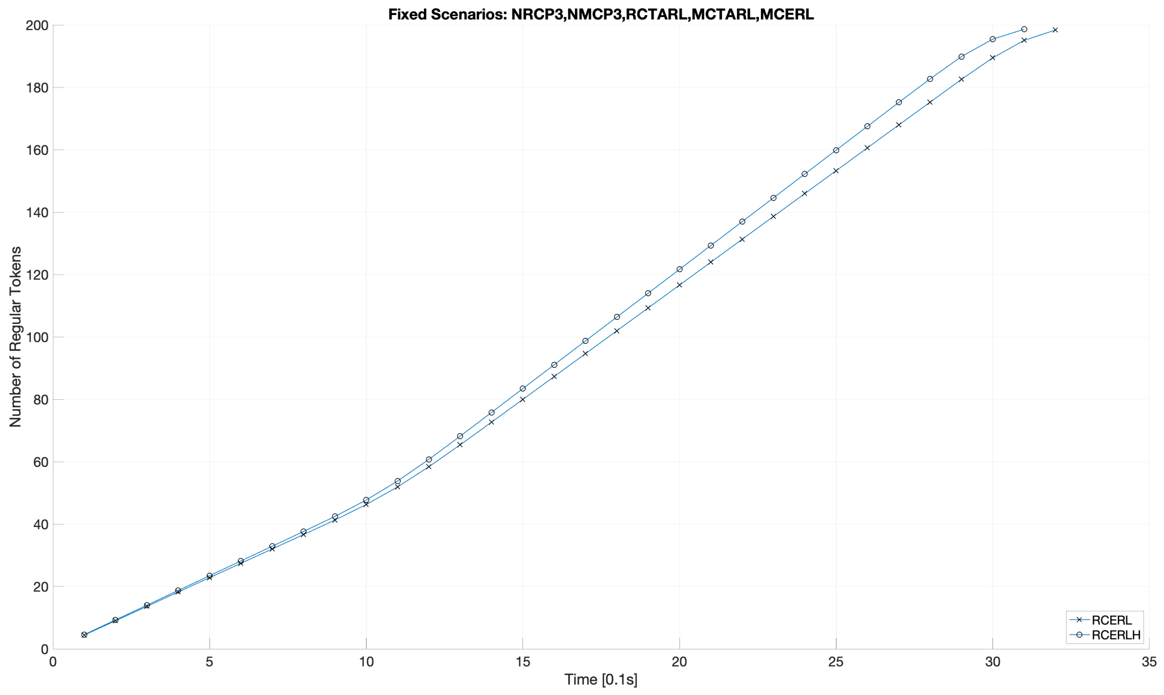


(b) Regular Transactions - Impact of Number of Regular Client Peers Change.

(c) Average Endorsement of Transactions (Tokens) in Time.



(a) Malicious Transactions - Impact of Regular Client Endorsement Rate Change.

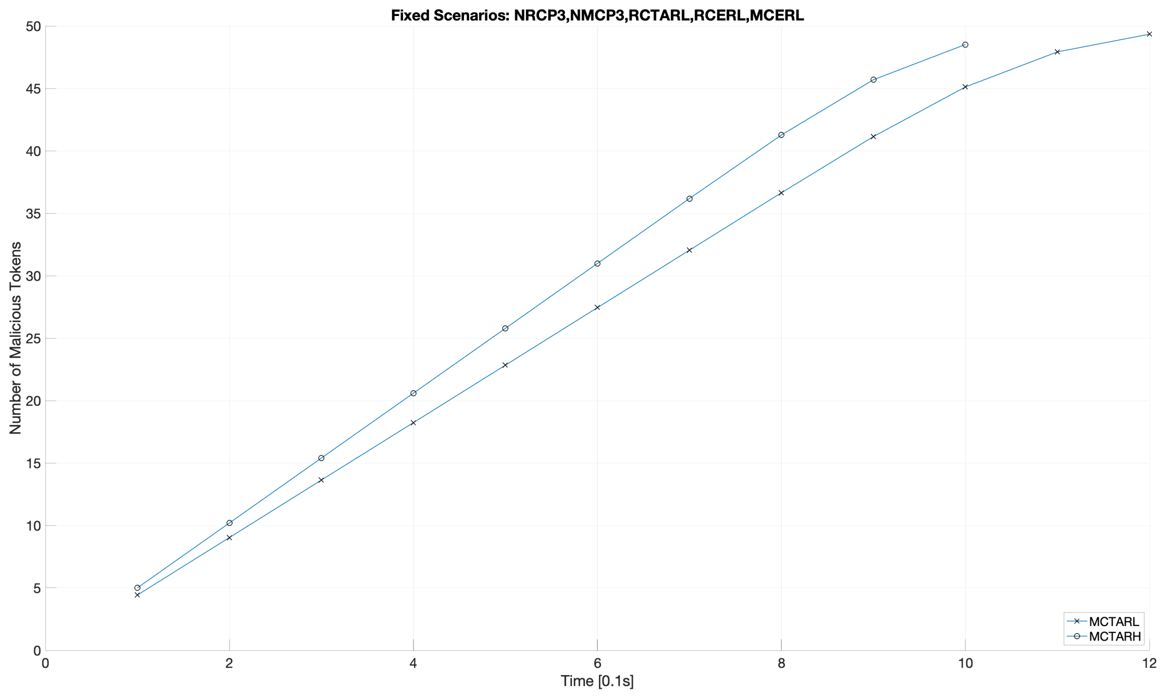


(b) Regular Transactions - Impact of Regular Client Endorsement Rate Change.

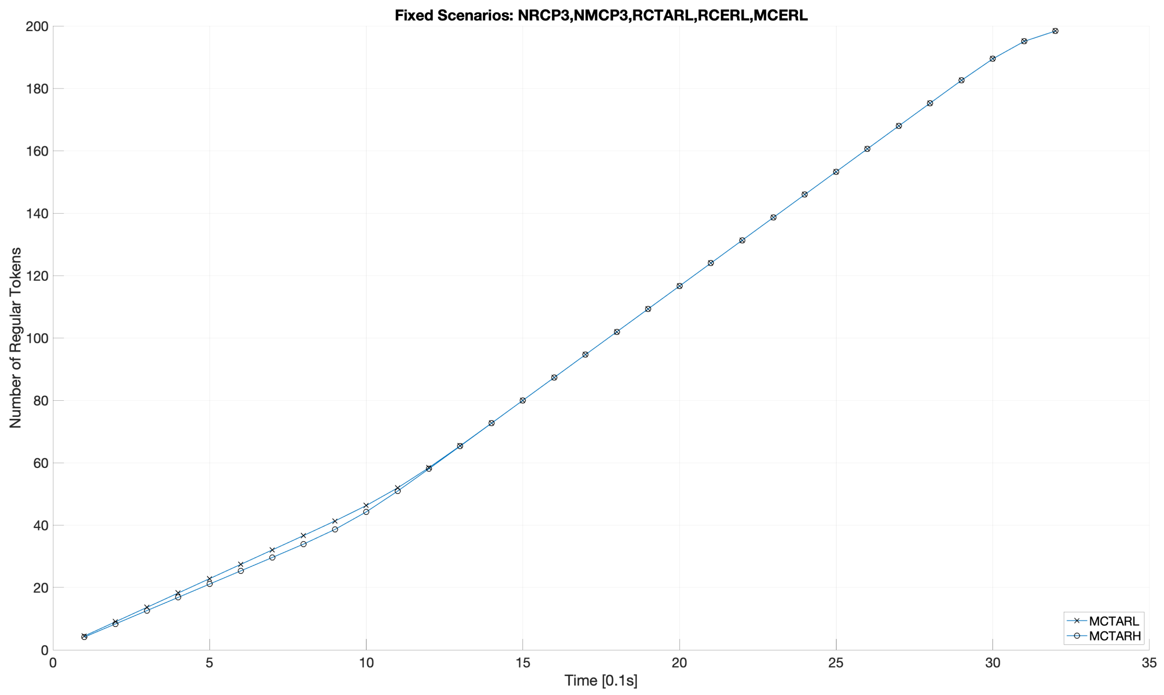
Figure 25. Average Endorsement of Transactions (Tokens) in Time.

(initial) transaction. This suggests that this approach could have adverse effects on HLF in case of a denial of service attack.

Finally, we consider the setting of 3 regular and 3 malicious client peers (NRCP3 and NMCP3), low malicious client transaction arrival rate (MCTARL) as well as low regular and malicious transactions endorsement rate (RCERL and MCERL). In this setting, we consider low and high regular client transaction arrival rates (RCTARL and RCTARH). We present our findings in Figure 27. Figure 27a shows a positive impact on the overall endorsement time of regular transactions as expected. However, Figure 27b shows a negative impact on endorsement times of part of the malicious transactions. This suggests that this approach could be a potential antidote that HLF can deploy in case of a denial of service attack.

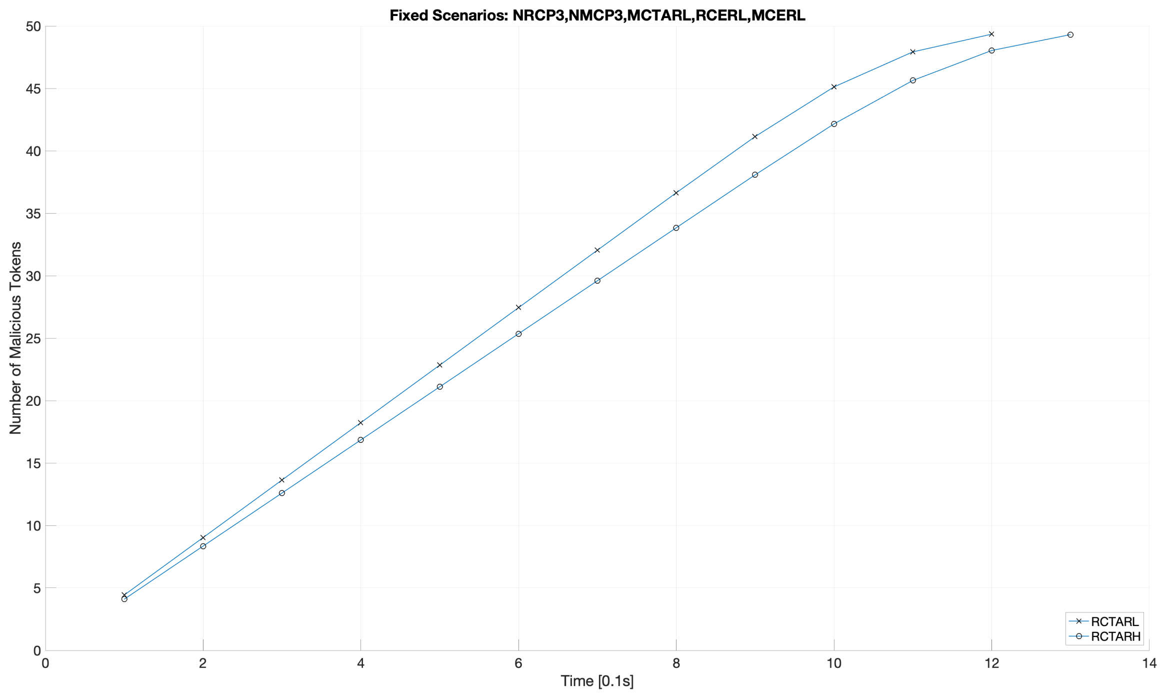


(a) Malicious Transactions - Impact of Malicious Transactions Arrival Rate Change.

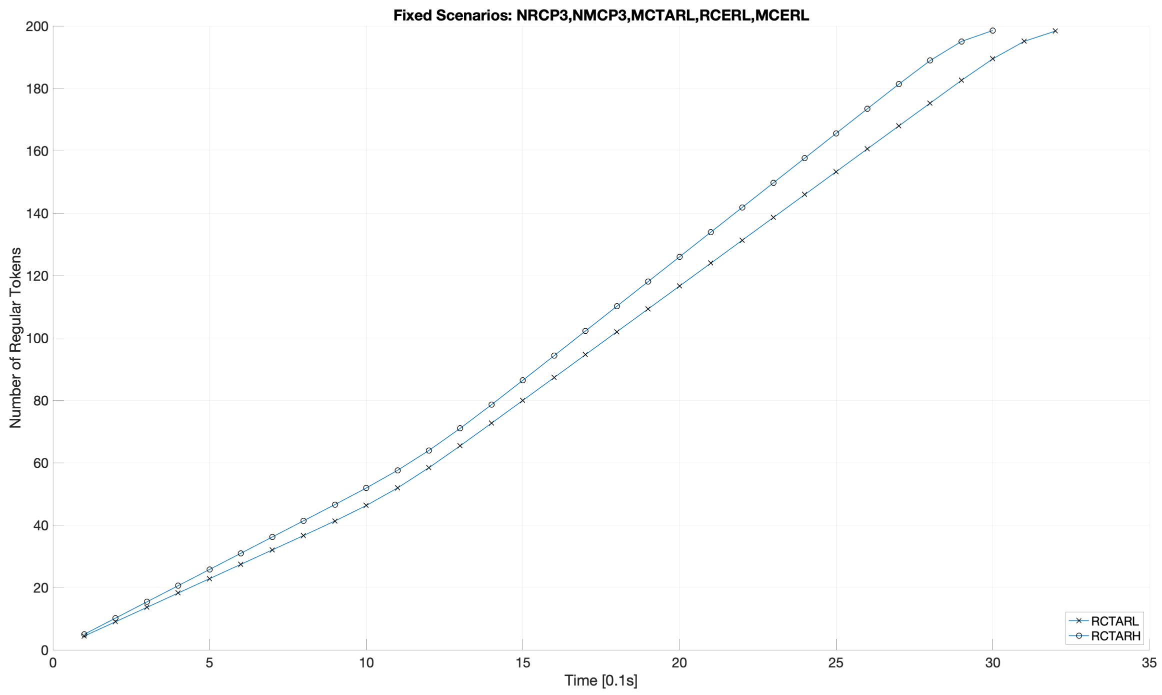


(b) Regular Transactions - Impact of Malicious Transactions Arrival Rate Change.

Figure 26. Average Endorsement of Transactions (Tokens) in Time.



(a) Malicious Transactions - Impact of Regular Transactions Arrival Rate Change.



(b) Regular Transactions - Impact of Regular Transactions Arrival Rate Change.

Figure 27. Average Endorsement of Transactions (Tokens) in Time.

2.8 Validation

We parametrize our SRN model using data collected from a HLF network setup in our cloud environment using the Google Cloud Platform (GCP). The ultimate goal of this campaign is to collect performance measures subject to realistic traffic patterns for validation purposes. In this section, we provide details of network setup, tools used, and our procedure for data collection and analysis. Below, we expand and detail these steps: network setup, chaincode test application, measurements, Caliper measurement procedure, and key parameters.

Network Setup. The data collected to parametrize our model is obtained from a HLF network setup deployed on a computing instance over the GCP. The computing instance is running a Linux-based machine (typec2-standard-8, 60vCPUs, 236 GB RAM) having the image Debian-10-buster-v2021072. Intel's Cascade Lake is the CPU platform chosen for emulation. The boot disk is set as a Balanced Persistent disk.

The HLF network setup is achieved using a minikube (v1.22.0) cluster over Kubernetes (v1.21.3). Kubernetes and minikube can be classified as container tools. Kubernetes[2] is an open-source orchestration system for Docker containers. It handles the scheduling aspect of nodes and actively manages workloads in a computing cluster. Minikube is a local Kubernetes, a single node cluster contained in a virtual machine (VM). We deploy the HLF over minikube using locally built Docker images. To obtain the needed parameters from the network, we make changes to the HLF (v2.2) source code and build the binaries locally on the GCP machine with Golang (v1.16.6) and Node (v10.24.0) installed. All binaries are hosted over Docker inside minikube. Within Kubernetes, each pod has a unique IP address for communication with other pods.

[2]See https://kubernetes.io/docs/tutorials/kubernetes-basics/.

The HLF network consists of four peers, and the channel's endorsement policy is set as a majority endorsement policy. Three peers have an endorsing role and the last remaining peer has the committing role (see Figure 28). The ordering service contains one ordering service node (OSN), 3 ZooKeeper nodes, and 4 Kafka brokers. The choice of Kafka ordering service is made following [16]. Since emulation is performed for a very small network and the ordering service is crash fault-tolerant, our choice is justified, however in future research RAFT can be considered. Hyperledger Caliper (v0.4.2) is acting as a client application to send transactions to the HLF network. An initial attempt was made to make changes in HLF (v1.x) but due to deprecated support from various Node and Golang libraries, it was abandoned in favor of HLF (v2.2).

All peer nodes (including the ordering service) are connected over a single channel using separate computational resources (8 CPUs each). Overall, each network entity is launched as a containerized application and connected to other network components using a single channel. Containers corresponding to each network entity (peer, certificate of authority, etc.) run over separate pods with a single channel thread that interacts with the network using locally installed Fabric SDK.

To execute the workload for the HLF network, we use Hyperledger Caliper (HC), a blockchain benchmarking tool under the Hyperledger project from the Linux foundation. HC allows users to measure the performance of a blockchain implementation. One of the many benefits of using HC is that it automatically handles the complex workflow performed by the client application. It features various standard rate control functions but for our purposes, we have implemented a new rate-control function to generate traffic following a Poisson arrival process. The literature on HLF modeling is still missing a distribution study analysis to choose the best-fit distribution for each transaction. However, consistently good validation results presented in this chapter and earlier works [16] certainly justify the choice of the exponential distribution for all transaction rates.

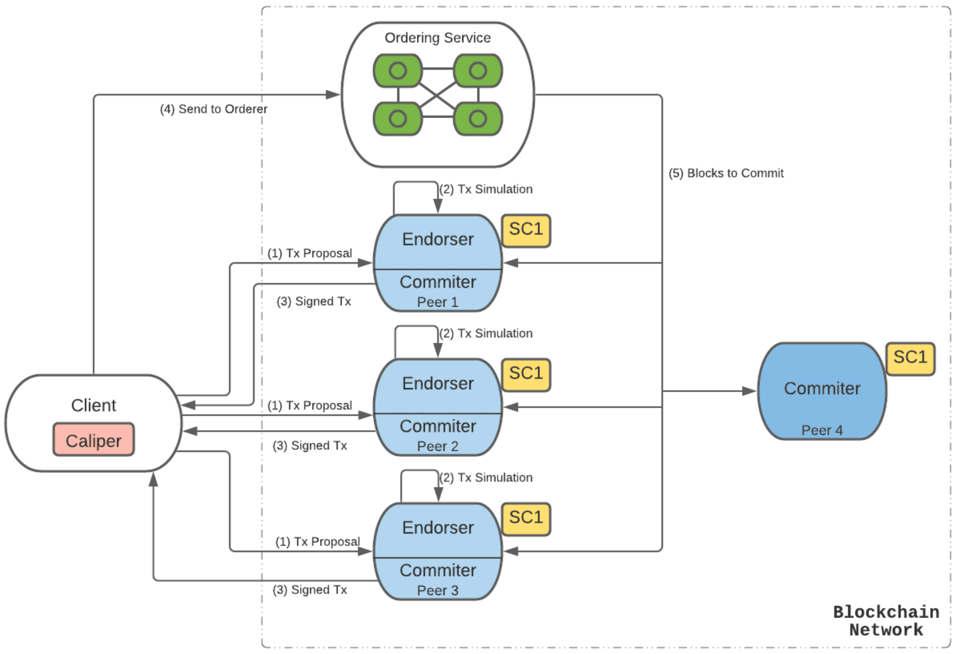


Figure 28. Hyperledger Fabric Network Set-Up

Chaincode Test Application. For performance measurements, we leverage a simple smart contract ('chaincode' SC1) and make necessary changes for our purposes of model validation. The function of SC1 is to maintain the account balance of users and it mainly performs two functions, 'query' and 'transfer'. The 'query' function checks if an account exists, and if not, creates a new account with a starting balance. It performs a total of two operations, one read and one write to the key-value repository. The 'transfer' function performs a balance transfer from one account to another. It performs two read and two write operations to the key-value repository. After a suitable amount of time running 'query' functions, the key-value repository will be populated for 'transfer' operations to complete successfully. Both functions on the client-side receive a random account number as input thus there is no dependency between consecutive transactions. This makes transactions rarely fail the validation stage. In this manner, we generate the workload of valid transactions following the desired process.

Measurements. We measure the time elapsed to perform the critical steps in the transaction flow by analyzing the HC output, logs files from the peers, and log files from the ordering service. We modify the HLF code by adding additional log entries to obtain the time when a transaction enters or leaves any particular phase. In any distinct phase, the weighted-time average of the number of transactions gives the mean queue length. The parameters of interest are measured for different block sizes with various transaction arrival rates.

Caliper Measurement Procedure. We run Hyperledger Caliper with a single client thread and a test duration of 240 seconds. To validate our model for various configuration settings, we vary three parameters in our network implementation: transaction type ('query', 'transfer'), block size, and transaction arrival rate. The parameters in Table 5 are used by the simulation in SPNP.

Key Parameters. We perform multiple test runs by varying the set of parameters described in the previous sections, collect log files, and derive the required parameters. Each test run consists of 9k to 20k transactions, with firing times following an exponential distribution that we enforce via mean transaction arrival rate. The key measurements for a 'query' type transaction are summarized in Table 6 given below. The MVCC and ledWrite are captured at the block level. For the validation effort, we compute the mean queue length at various stages within a peer: OS, VSCC, MVCC, and ledWrite. Mean queue length provides an intuitive insight into the performance of the HLF system.

The results given in Table 6 are measurements from the simulation (SPNP column) and real HLF network deployment (Validating Framework column) baseline scenario. The rows are mean queue length measurements of the four stages for a particular arrival rate and block size combination considered. To validate our model, we follow the common validation procedure in the literature [16], [17] by computing and comparing the mean queue length (MQL) at the stages: OS, VSCC, MVCC, and ledWrite with empirical results obtained from the validation framework. The MQL measure at these four stages between simulation and validation framework is an intuitive validation metric that establishes a correspondence between their transaction flow. For all stages, our differences, deduced from Table 6, between empirical and simulated values are comparable to the validation criteria in [16], where similar differences between SPNP and validating framework MQL measurements hold as arrival rate and block size are varied. Overall, we have similar differences and validating characteristics between simulation and empirical results as was shown in [16]. Therefore, in the context of validation purposes existing in the literature, we consider our model validated, that is,

Parameter (ms)	**Block 40**	**Block 60**	**Block 80**	**Block 200**
Client Processing	4.990	4.964	4.993	4.992
Endorsement	1.289	1.278	1.251	1.193
Transmit to Ordering Service	1.577	1.597	1.582	1.585
Block Creation and Delivery	76.196	114.747	156.616	381.808
VSCC	0.0401	0.0580	0.0718	0.0260
MVCC	4.938	7.475	9.535	18.777
Ledger Write:	39.641	47.432	53.237	81.484
Arrival Rate (TPS)	235.5	236.6	237.9	237.2

Table 5. Model Parameters Obtained from Validating Framework. See Section 2 For a Description of Model Parameters.

Mean Queue Length Validation Set - Baseline Scenario			
(Arrival Rate, Block Size)	Stage	SPNP	Validating Framework
(235.5, 40)	OS	27.503	26.324
	VSCC	0.0111	0.0145
	MVCC	0.516	0.577
	Ledger Write	4.175	4.677
(236.6, 60)	OS	41.602	40.725
	VSCC	0.0236	0.0241
	MVCC	0.785	0.881
	Ledger Write	4.950	5.638
(237.9, 80)	OS	56.127	58.474
	VSCC	0.0385	0.0278
	MVCC	1.009	1.125
	Ledger Write	5.574	6.358

Table 6. Mean Queue Length Comparison Between Spnp and Validating Framework under Baseline Scenario.

the model captures the transaction flow of a real HLF network deployment with a majority endorsement policy.

2.9 Conclusion

In this work, we contribute to the literature a novel approach in joint modeling of HLF, together with a cyberattack, here Sybil attack. Our findings show that this approach can yield a novel understanding of vulnerabilities of HLF and the interplay between structure and parametrization of HLF and, structure and parametrization of Sybil cyberattack.

Our simulations can capture the overall impact, at the transaction level, that a Sybil attack exerts on a HLF network. In all simulations involving the Sybil attack, there is a quantifiable degradation of performance on the HLF network. Moreover, the simulations provide a high level of detail capturing performance degradation from the beginning of a Sybil attack until its conclusion for all parameter and network structure combinations considered. The classification of the specific ways the network's performance is affected serves as a basis for advancing the understanding of Sybil attacks on HLF.

In view of increasing the chances of a successful Sybil attack and forcing malicious transactions through the network, the numerical results indicate: (1) A Sybil attack configuration that involves six Sybil endorsing peers is a favorable configuration over having three endorsing peers for each, the regular client and Sybil client. Additionally, the Sybil attack will cause less performance degradation to the HLF network with six endorsing peers than with three. (2) A Sybil attack that involves a high Sybil client transaction proposal rate will be favorable over having a lower Sybil client transaction proposal rate. Moreover, a high Sybil client transaction proposal rate will cause less performance degradation than a lower Sybil client transaction proposal rate. The favorable configurations for a Sybil attack in (1) and (2) also imply a higher difficulty in detecting such an attack via performance metrics.

From the point of view of decreasing the chances of allowing malicious transactions through the HLF network in a Sybil attack scenario, the numerical results reveal: (3) Having fewer regular client endorsing peers, three, compared to six is more favorable in hindering the number of Sybil transactions committed to the ledger. (4) Maintaining a lower client transaction proposal rate will decrease a Sybil client's ability to propose transactions and ultimately commit them to the ledger. The configurations and restrictions from (3) and (4) imply a lower chance of a successful Sybil attack.

When it comes to future work, the work presented in this paper is generalizable in at least two directions. First, it can be a template for understanding the behavior of other cyber-physical systems during a cyberattack. Various structural and contextual choices can be made within a particular attack, making for a potentially large space of attacks and their variations. As an added complexity, the efficiency of an attack is very dependent on the structure of HLF, and its implementation. An interesting analysis would look at how for a particular attack, the characteristics of HLF fabric affect its effectiveness. Second, it can be expanded to be a platform for understanding the performance of various types of cyberattacks and devising ways for their detection. A particular structural choice of attack, for example, the various configurations of the Sybil attack presented in this paper, offers multiple ways of information extraction over time. Thus multiple signal formulations can be considered to discern the possibility of their detection within the space of measurable variables within HLF.

Chapter 3

EFFECT OF MACHINE LEARNING PERFORMANCE WITH SYNTHETIC DATA AUGMENTATION FROM DIFUSSION MODELS ON CYBER-INTRUTION DATASETS

3.1 Introduction

The successful application of machine learning models in various fields strongly depends on the quality of the data used for training. Obtaining additional data can improve a machine learning model's performance. Unfortunately, this could prove challenging, costly, or, in some cases, impossible. More specifically, leveraging additional data can be crucial for achieving satisfactory performance for applications involving classification tasks in the imbalanced setting and anomaly detection tasks via unsupervised and semi-supervised methods. Originally developed for image generation, diffusion models have shown promise in generating realistic synthetic tabular data. Emerging evidence points to the eventual triumph of diffusion models over generative adversarial networks analogous to what has happened in the computer vision field with image generation. This chapter explores the adaptation of recent open-source implementations of diffusion models for synthetic tabular data generation. We discuss the principles of diffusion models, their application to tabular data, and the challenges and future directions in this field in the context of cybersecurity and anomaly detection. We aim to provide insights and empirical evidence into the effectiveness of diffusion models for generating high-quality synthetic tabular datasets focusing on cyber and anomaly detection datasets for performance improvements through synthetic data augmentation in the tabular setting.

3.2 Literature Review

The cumulative number of deep learning papers has been increasing steadily over the last decade, according to [45], showing no signs of stopping. Even subfields of deep learning, like generative AI, have recently had an explosion of research activity on ArXiv. Further narrower in scope, diffusion models have increased their numbers rapidly and intensely. Diffusion models have a wide range of applications, including computer vision, natural language processing, reinforcement learning anomaly detection, and more. More unconventional applications include protein design [46], medical image synthesis [47], inverse problems [48] and stenography [49].

One of the first works in denoising diffusion probabilistic models for image synthesis we refer the reader to is [50]. The focus of this chapter is on the adaptation of diffusion models for synthetic tabular data generation. Our primary reference is [51], where the authors introduced a single diffusion model for tabular datasets capable of handling numerical and categorical and thus can be universally applied to any tabular dataset. The model performed strongly in extensive tests on a wide set of datasets for regression and classification tasks. The authors show that the superiority of the diffusion model's synthetic data surpasses that of GAN/VAE alternatives, which is consistent with the advantage of diffusion models in other fields like computer vision. An earlier effort [52] tackled the inherent inhomogeneous nature of tabular data by separating discrete and continuous variables using two separate conditional diffusion models. Outside the general generation of synthetic data, the authors in [53] considered the problem of the imputation of tabular data with diffusion models with comparable and effective performance with well-known existing methods. For a complete textbook introduction to the field of synthetic data, we refer the reader to [54].

3.3 Methods

We augment an original dataset with synthetic samples to measure the effectiveness

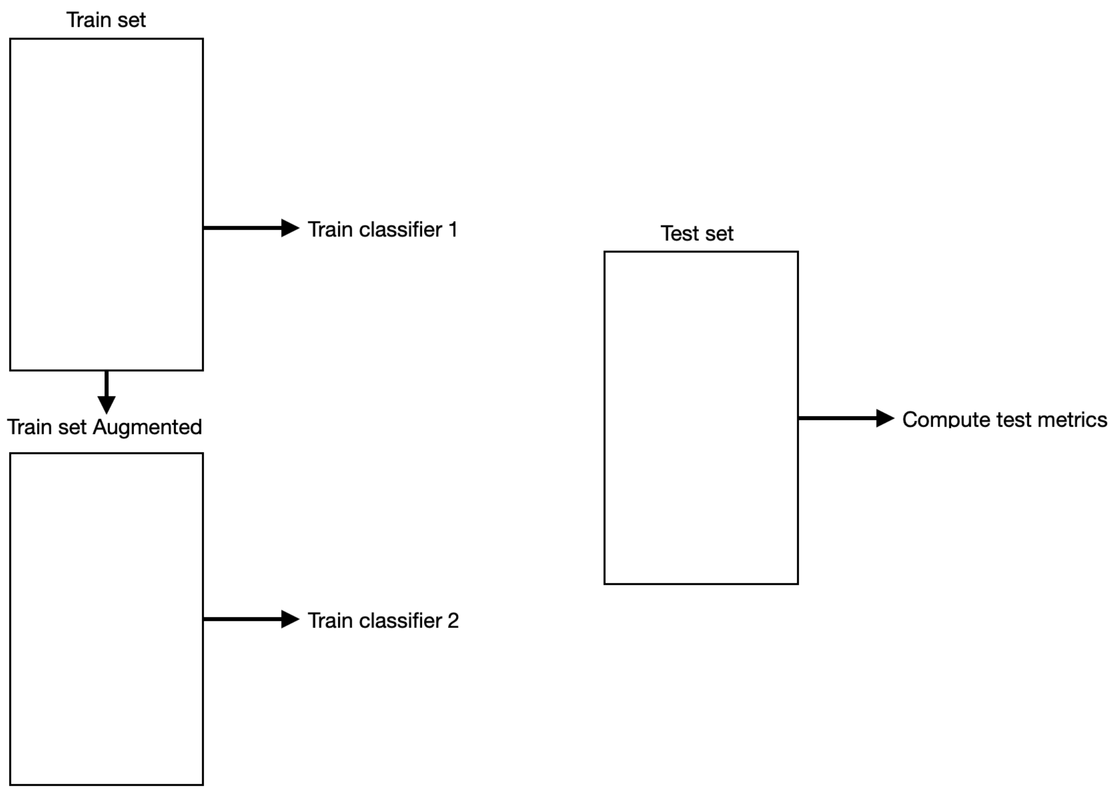


Figure 29. After Separating into a Training and Testing the Training Set Is Used for Synthetic Data Generation. The Test Set Is Not Leaked.

of synthetic tabular data generated by TabDDPM on cyber datasets. We train machine learning classifiers on the original and augmented datasets. We then use a test set to compute well-known classification metrics and asses the performance improvements obtained. The frameworks for synthetic data generation used in this chapter are open-source and available at [51, 55].

3.4 Evaluation Metrics

In order to evaluate these models, we compute the following:

- True Negatives (TN): number of correctly classified negative (normal) samples
- True Positives (TP): number of correctly classified positive (fraudulent) samples

- False Negatives (FN): number of positive (fraudulent) samples incorrectly classified as negative (normal)
- False Positives (FP): number of negative (normal) samples incorrectly classified as positive (fraudulent).

The relationship between the predicted and the actual classifications can be represented by a confusion matrix as shown in Table 7.

Table 7. Confusion matrix for two classes, negative (non-fraud) and positive (frauds).

		Predicted Class	
		Negative	Positive
Actual Class	Negative	True Negative (TN)	False Positive (FP)
	Positive	False Negative (FN)	True Positive (TP)

There are several metrics that combine these quantities in some way to evaluate a model, the most common being accuracy. Accuracy measures the number of correctly classified samples (TP + TN) out of the total number of samples.

$$\text{accuracy} = \frac{\text{TN} + \text{TP}}{\text{TP} + \text{FN} + \text{FP} + \text{FN}}.$$

The accuracy can range from 0 to 1, where 1 represents a perfect-performance model. However, the accuracy sometimes may not be a good indicator for imbalanced data because a model that classifies all samples as normal transactions will still obtain an accuracy of more than 99%, even though it completely disregards the fraudulent samples. In such cases, balanced accuracy is used instead because it places equal weight on the correct classification of each class.

$$\text{balanced accuracy} = \frac{1}{2}\left(\frac{\text{TP}}{\text{TP} + \text{FN}} + \frac{\text{TN}}{\text{TN} + \text{FP}}\right).$$

The balanced accuracy can range from 0 to 1, where 1 represents a model with perfect performance.

3.4.1 ROC Curve

The Receiving operating characteristic (ROC) curve plots the True Positive Rate (TPR) against False Positive Rate (FPR) at all possible classification thresholds. True Positive Rate (TPR) is a synonym for recall and is computed as

$$\text{True Positive Rate} = \frac{\text{TP}}{\text{TP} + \text{FN}}.$$

False Positive Rate (FPR) measures the number of incorrectly classified positive (fraudulent) samples out of the total number of samples actually belonging to the negative class. It is defined as

$$\text{False Positive Rate} = \frac{\text{FP}}{\text{FP} + \text{TN}}.$$

The ROC curve considers negative (normal) and positive (fraudulent) classes. Both metrics are computed for various probability thresholds τ in order to construct a curve. The area under this curve (AUC) can be used for aggregating the measure of TPR and FPR across all possible thresholds and returns a single score. The AUC score can range from 0 to 1, with 1 indicating perfect performance.

3.4.2 Precision-Recall Curve

The precision-recall curve is constructed using the metrics of precision and recall. Precision measures the number of correctly classified positive (fraudulent) samples out of the total number of samples classified as coming from the positive class. It is defined as

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}.$$

Recall measures the number of correctly classified positive (fraudulent) samples out of the total number of samples actually belonging to the positive class. It is defined as

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}.$$

Unlike the ROC-AUC Curve, precision and recall are only concerned with the positive (fraudulent) class and therefore evaluate a model's performance primarily based on the minority class. Likewise, both metrics are obtained with various probability thresholds in the ROC curve. We could then compute the AUC to obtain the model performance.

3.4.3 F1-Score

F1-Score is the weighted average of precision and recall, and is used for model comparison. It is defined as,

$$\text{F1-score} = \frac{2 \cdot (\text{precision} * \text{recall})}{\text{precision} + \text{recall}}.$$

F1-Score can fall between 0 and 1, with 0 meaning either precision or recall are zero.

3.4.4 Matthew's Correlation Coefficient

The Matthew's Correlation Coefficient (MCC) is used to assess machine learning predictions in binary and multiclass settings [56, 57]. The correlation coefficient is between +1 and -1, and the correlation coefficient is 0 when the prediction is entirely random. The MCC is defined as,

$$\text{MCC} = \frac{\text{TP} \times \text{TN} - \text{FP} \times \text{FN}}{\sqrt{(\text{TP+FP})(\text{TP+FN})(\text{TN+FP})(\text{TN+FN})}}$$

3.5 Datasets

3.5.1 Credit Card Fraud Detection

The dataset we use is from the machine learning group at the Université Libre de Bruxelles, Belgium, publicly available via Kaggle and contains credit card transactions made by European cardholders over the span of two days in September of 2013. This dataset has 492 frauds out of 284,807 total transactions, i.e., the positive class (frauds) comprises 0.172% of all transactions. There are 31 features. The other two are 'Time', which specifies the seconds elapsed between each transaction and the first transaction in the dataset, and 'Amount', which is the transaction amount. The column 'Class' contains the target labels, which take a value of 1 for fraudulent transactions and 0 otherwise.

3.5.2 Ethereum Transactions

The second dataset we use is also from Kaggle. This dataset consists of 7,662 non-fraud transactions and 2,179 frauds out of 9,841 total transactions made over Ethereum, i.e., the positive class (frauds) accounts for 22.14% of all transactions, which is much higher than the Credit Card dataset. There are 51 features where, 3 of which are categorical, and the others are numeric. In this section, we only use numeric

	Credit-card	Ehetereum
size (N)	$284,807$	$9,841$
proportion fraud	0.17%	22%
number features	29	45

Table 8. Summary of the Two Datasets.

features for computation efficiency. The feature 'FLAG' contains the target labels, which take a value of 1 for fraudulent transactions and 0 otherwise. Table 8 shows the summary of the two datasets.

3.5.3 UNSW-NB15

UNSW-NB15 is a comprehensive dataset for network intrusion detection systems released by the Intelligent Security Group at UNSW Canberra [58]. The dataset consists of a hybrid of real modern normal activities and synthetic contemporary attack behaviors. It includes nine types of attacks: worms, shellcode, reconnaissance, generic, exploits, denial of service, backdoors, analysis, and fuzzers. It has 42 features and the attack or normal label. The original dataset is considerably large ($\approx$ 2.5 million records), but the authors released a partitioned training set and a test set containing 82,332 and 175,341 records, respectively. The number of instances added for DoS, analysis, backdoor, exploit, fuzzers, reconnaissance, shellcode, and worm are: 7000, 7000, 6989, 2000, 9809, 3450, 12000, and 1000 respectively.

3.5.4 BRL IoT Device Management over Blockchain

This dataset is from the Blockchain Research Laboratory at Arizona State University. It contains static and dynamic parameters from an IoT device logging data to a Hyperledger Fabric platform (permissioned blockchain). The goal is to create a

Class	Train	(%)	Test	(%)
Normal	37000	75.5	56000	32.0
DoS	4089	8.3	12264	7.0
Generic	18871	38.5	40000	22.8
Exploit	11132	22.7	33393	2.0
Fuzzers	6062	12.4	18184	10.4
Reconnaissance	3496	7.1	10491	6.0
Analysis	677	1.4	2000	1.2
Backdoor	583	1.2	1746	1.0
Shellcode	378	0.77	1133	0.6
Worms	44	0.09	130	0.1

Table 9. UNSW NB15 Data Distribution for Train and Test Sets.

digital twin stored on an immutable ledger to asses levels of cybersecurity by deploying machine learning models on the historical data to keep track of changes in the device's state. The dataset subset used for augmentation contains four features: usedMemory, bufferCache, availableMemory, and Memory Utilization_kbcommit. The target column contains the device state with four unique values, giving a multi-classification dataset.

3.6 Results

This section presents the results concerning performance improvements from metrics introduced in Section 3. We remark that during the training process, the test set is not, in any way, used or leaked into training. The augmentation for these two datasets is with positive and negative classes with the generation following the distribution of the train set. All models use the default hyperparameters from the Scikit-learn module [59].

The ML classifiers considered in this section are briefly described below:

XGBoost: XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible, and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solves many data science problems in a fast and accurate way [60].

Decision Tree: Are a class of non-parametric supervised learning algorithms used for regression and classification. We focus on the latter task; in this case, decision trees are called classification trees. The classification tree predicts the value of a target variable by learning decision rules inferred from the data features.

Random Forest A random forest is an estimator that trains a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control any over-fitting.

3.6.1 Augmenting with Synthetic Data

In this section, the objective is augmentation with synthetic data, that is, data rows (instances) with positive and negative labels following the training set's distribution. The quantity added is usually a multiple of the training set's size. Classifiers are trained independently on the training and augmented training set, respectively, and their performance determined from the test set. ROC AUC is one of the most widely used threshold-free metrics for binary classification [57]. Table 10, shows the ROC AUC on the test set by XGBoost classifiers trained on the training and training augmented with synthetic data. The train, validation, and test split for both dataset is 70%, 15%, and 15% respectively.

ROC AUC	Credit-card	Ethereum
Original	0.967	0.983
Augmented	**0.981**	**0.997**

Table 10. Summary of ROC AUC Performance Metric for an XGBoost Classifier.

Results for the BRL IoT dataset are shown in table 11. Synthetic data augmentation is used, meaning instances are added following the training data's distribution. A total of 1,100 instances are added. On this dataset, all performance metrics improve with synthetic data augmentation. The accuracy score improved from 0.6356 to 0.6639.

	Original				TabDDPM Augmented			
	Precision	Recall	F1-score	MCC	Precision	Recall	F1-score	MCC
XGB	0.6391	0.6356	0.6354	0.5149	**0.6771**	**0.6705**	**0.6718**	**0.5621**

Table 11. Summary of Performance Metrics for the BRL IoT Dataset with XGBoost Classifier.

3.6.2 Balancing with Synthetic Data

In this section, we balance the original training set with labels from classes that contain a low number of instances. Then, classifiers are independently trained on the original training set and the balanced training set, and their performance is assessed on a test set. Table 12 presents the precision, recall, f1-score, and MCC scores for the UNSW NB15 dataset. Performance gains exist in the decision tree (DT) and XGBoost (XGB) classifiers. Slight performance degradation in performance exists for the random forest classifier.

Balancing with synthetic data for the credit card dataset yields improvements in ROC AUC, recall, and F1-score. Table 14 summarizes the results. The ROC AUC

	Original				TabDDPM Balanced			
	Precision	Recall	F1-score	MCC	Precision	Recall	F1-score	MCC
XGB	0.7524	0.7699	0.7253	0.7089	**0.7651**	**0.7712**	**0.7378**	**0.7094**
RF	0.7458	0.7470	0.7111	0.6792	**0.7483**	**0.7534**	**0.7178**	**0.6865**
DT	0.6682	0.6847	0.6468	0.6002	**0.6990**	**0.7103**	**0.6801**	**0.6359**

Table 12. Summary of Results for UNSW NB15 Dataset on Test set. Metrics for Model Trained with Original Train and Balanced Train Set Shown.

metric improved from 0.872 to 0.904. The balanced accuracy increased from 0.8717 to 0.9005. The train, test split is 80%, 20% respectively.

	Original				TabDDPM Balanced			
	Precision	Recall	F1-score	MCC	Precision	Recall	F1-score	MCC
XGB	**0.951**	0.744	0.835	0.841	0.933	**0.808**	**0.866**	**0.868**

Table 13. Summary of Performance Metrics for the Credit Card Dataset with XGBoost Classifier.

Adding 500 positive instances to the Ethereum dataset yields slight performance improvements. The ROC AUC increased from 0.962 to 0.969. The balanced accuracy slightly rose from 0.9616 to 0.9687. The train, test split is 70%, 30% respectively.

	Original				TabDDPM Balanced			
	Precision	Recall	F1-score	MCC	Precision	Recall	F1-score	MCC
XGB	0.9866	0.9866	0.9865	0.9470	**0.9882**	**0.9883**	**0.9882**	**0.9537**

Table 14. Summary of Performance Metrics for the Ethereum Dataset with XGBoost Classifier.

The main objective of conditional tabular generative adversarial network (CTGAN) model introduced in [61] is to tackle the inherent mix of continuous and discrete columns or features in commonly found in tabular datasets. CTGAN competes with tabular variational autoencoders (TVAE), and is able to provide better performance in several datasets. We find TabDDPM providing, in turn, comparable performance to CTGAN [62] and ultimately providing slight performance boost in five of the eight metrics we consider across two models: decision trees and random forests on the UNSW NB15 dataset.

	CTGAN				TabDDPM			
	Precision	Recall	F1-score	MCC	Precision	Recall	F1-score	MCC
DT	2.296%	2.519%	3.333%	3.690%	**4.609**%	**3.738**%	**5.148**%	**5.948**%
RF	**1.323**%	**1.227**%	0.470%	**1.720**%	0.335%	0.856%	**0.942**%	1.074%

Table 15. Percent Improvement of Performance Metrics with Decision Tree and Random Forest Classifiers under Synthetic Tabular Data Balancing from CTGAN and TabDDPM on the UNSW NB15 Dataset.

3.7 Conclusion

Performance obtained from leveraging a small number of labels ($\approx 1\%$) with semi-supervised methods can surpass the current state-of-the-art unsupervised learning methods [63]. In ML cybersecurity applications where, labels of rare instances, cyberattacks, intrusion logs, and many more., can be challenging to obtain initially, synthetic data augmentation can be crucial for successful classification tasks. Thus, synthetic tabular data generation via denoising diffusion models naturally finds its place as components to be integrated early in the design of ML classifiers to aid when small amounts of the label are available, and increasing the availability of such is virtually impossible. The authors in [51] have demonstrated the good quality of synthetic data generation by TabDDPM in a wide range of datasets with a combination of numerical, categorical, or both for regression and classification tasks. The results in [62] show improvements in the performance metrics of ML classifiers via balancing with conditional generative adversarial networks for tabular data. At the moment, conditional GAN models posses an advantage in the capabilities to balance tabular data since their data generation can be guided towards the desired labels of low quantities. The diffusion model for tabular data considered in this chapter is not conditional and thus one has to resort to filtering data generated for a certain constraint of interest, for example, the class labels that are imbalanced. In summary, we have applied synthetic tabular data augmentation to cyber datasets, which are characterized as being highly imbalanced, which makes training challenging. We have found that synthetic data augmentation can improve standard classification metrics commonly used for anomaly detection and classification tasks in these datasets. These results firmly position denoising diffusion probabilistic models for additional research in applications to anomaly detection, especially with MLP and NN-based algorithms architectures. Future research will investigate the performance improvements synthetic data augmentation can provide with custom losses, architecture, and ensemble models.

REFERENCES

[1] 'Cyber risk and risk management.' In: *The Institute of Risk Management* (2018).

[2] Jevtić, P. and Lanchier, N. 'Dynamic structural percolation model of loss distribution for cyber risk of small and medium-sized enterprises for tree-based LAN topology'. In: *Insurance Math. Econom.* 91 (2020), pp. 209–223. DOI: 10.1016/j.insmatheco.2020.02.005. URL: https://doi-org.ezproxy1.lib.asu.edu/10.1016/j.insmatheco.2020.02.005.

[3] Lanchier, Nicolas et al. 'Probabilistic Framework For Loss Distribution Of Smart Contract Risk'. In: *Advances in Complex Systems (ACS)* 24.07n08 (2021), pp. 1–31.

[4] Chiaradonna, Stefano, Jevtić, Petar, and Lanchier, Nicolas. 'Framework for cyber risk loss distribution of hospital infrastructure: Bond percolation on mixed random graphs approach'. In: *Risk Analysis* (2023). DOI: https://doi.org/10.1111/risa.14127. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/risa.14127. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/risa.14127.

[5] Chiaradonna, Stefano et al. 'Framework for cyber risk loss distribution of client-server networks: a bond percolation model and industry specific case studies.' In: Submitted. (2023).

[6] Androulaki, Elli et al. 'Hyperledger fabric: a distributed operating system for permissioned blockchains'. In: *Proceedings of the thirteenth EuroSys conference*. 2018, pp. 1–15.

[7] Broadbent, S. R. and Hammersley, J. M. 'Percolation processes. I. Crystals and mazes'. In: *Proc. Cambridge Philos. Soc.* 53 (1957), pp. 629–641. DOI: 10.1017/s0305004100032680. URL: https://doi-org.ezproxy1.lib.asu.edu/10.1017/s0305004100032680.

[8] Grimmett, G. *Percolation*. Second. Vol. 321. Grundlehren der Mathematischen Wissenschaften. Springer-Verlag, Berlin, 1999, pp. xiv+444. DOI: 10.1007/978-3-662-03981-6. URL: https://doi-org.ezproxy1.lib.asu.edu/10.1007/978-3-662-03981-6.

[9] Erdős, P. and Rényi, A. 'On random graphs. I'. In: *Publ. Math. Debrecen* 6 (1959), pp. 290–297.

[10] Ajtai, M., Komlós, J., and Szemerédi, E. 'Largest random component of a k-cube'. In: *Combinatorica* 2.1 (1982), pp. 1–7. DOI: 10.1007/BF02579276. URL: https://doi-org.ezproxy1.lib.asu.edu/10.1007/BF02579276.

[11] Kott, A. and Linkov, I. *Cyber resilience of systems and networks.* Springer International Publishing, 2019.

[12] Jevtić, P., Lanchier, N., and La Salle, A. 'First and second moments of the size distribution of bond percolation clusters on rings, paths and stars'. In: *Statist. Probab. Lett.* 161 (2020), pp. 108714, 6. DOI: 10.1016/j.spl.2020.108714. URL: https://doi-org.ezproxy1.lib.asu.edu/10.1016/j.spl.2020.108714.

[13] Foundation, Linux. *A Blockchain Platform for the Enterprise - Hyperledger Fabric Read the Docs.* URL: https://hyperledger-fabric.readthedocs.io/en/release-2.2/.

[14] Castillo, Michael del and Schifrin., Matt. *Forbes BLOCKCHAIN 50.* URL: https://www.forbes.com/sites/michaeldelcastillo/2020/02/19/blockchain-50/?sh=398cb51e7553.

[15] *Federal Government Needs to Urgently Pursue Critical Actions to Address Major Cybersecurity Challenges – High Risk Issue.* URL: https://www.gao.gov/assets/gao-21-288.pdf.

[16] H. Sukhwani N. Wang, K. S. Trivedi and Rindos., A. 'Performance Modeling of Hyperledger Fabric (Permissioned Blockchain Network).' In: *2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)* (2018).

[17] Yuan, Pu et al. 'Performance modeling and analysis of a Hyperledger-based system using GSPN'. In: *Computer Communications* 153 (Feb. 2020). DOI: 10.1016/j.comcom.2020.01.073.

[18] Sukhwani, H. et al. 'Performance Modeling of PBFT Consensus Process for Permissioned Blockchain Network (Hyperledger Fabric)'. In: *2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS)* (2017), pp. 253–255.

[19] Jiang, Lili et al. 'Performance analysis of Hyperledger Fabric platform: A hierarchical model approach'. In: *Peer-to-Peer Networking and Applications* 13.3 (2020), pp. 1014–1025.

[20] Xu, Xiaoqiong et al. 'Latency performance modeling and analysis for hyperledger fabric blockchain network'. In: *Information Processing & Management* 58.1 (2021), p. 102436.

[21] Wu, Ou et al. 'Performance Modeling of Hyperledger Fabric 2.0'. In: *The International Conference on Evaluation and Assessment in Software Engineering 2022*. 2022, pp. 357–365.

[22] Khattar Krish Mittal Dev, Tyagi Shobha. 'Performance analysis of hyperledger fabric blockchain network 2.2'. In: *JIMS8I International Journal of Information Communication and Computing Technology* 10 (2022), pp. 553–556. DOI: http://dx.doi.org/10.5958/2347-7202.2022.00003.2.

[23] Melo, Carlos et al. 'Performance and availability evaluation of the blockchain platform hyperledger fabric'. In: *The Journal of Supercomputing* (2022), pp. 1–23.

[24] Kuzlu, Murat et al. 'Performance analysis of a hyperledger fabric blockchain framework: throughput, latency and scalability'. In: *2019 IEEE international conference on blockchain (Blockchain)*. IEEE. 2019, pp. 536–540.

[25] Foschini, Luca et al. 'HyperLedger fabric blockchain: chaincode performance analysis'. In: *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE. 2020, pp. 1–6.

[26] Lohachab, Ankur et al. 'Performance evaluation of Hyperledger Fabric-enabled framework for pervasive peer-to-peer energy trading in smart Cyber–Physical Systems'. In: *Future Generation Computer Systems* 118 (2021), pp. 392–416.

[27] You, Dan, Wang, Shouguang, and Seatzu, Carla. 'A liveness-enforcing supervisor tolerant to sensor-reading modification attacks'. In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 52.4 (2021), pp. 2398–2411.

[28] Zhao, Xin, Zou, Suli, and Ma, Zhongjing. 'Decentralized Resilient H Load Frequency Control for Cyber-Physical Power Systems Under DoS Attacks'. In: *IEEE/CAA Journal of Automatica Sinica* 8.11 (2021), pp. 1737–1751.

[29] Zhao, Yue et al. 'Passivity-based robust control against quantified false data injection attacks in cyber-physical systems'. In: *IEEE/CAA Journal of Automatica Sinica* 8.8 (2021), pp. 1440–1450.

[30] Xia, Chuanliang and Li, Chengdong. 'Property preservation of Petri synthesis net based representation for embedded systems'. In: *IEEE/CAA Journal of Automatica Sinica* 8.4 (2020), pp. 905–915.

[31] Lee, Sungho et al. 'Latency modeling of hyperledger fabric for blockchain-enabled IoT networks'. In: *arXiv preprint arXiv:2102.09166* (2021).

[32] Zhang, Jun et al. 'Deep learning based attack detection for cyber-physical system cybersecurity: A survey'. In: *IEEE/CAA Journal of Automatica Sinica* 9.3 (2021), pp. 377–391.

[33] Dabholkar, Ahaan and Saraswat., Vishal. 'Ripping the fabric: Attacks and mitigations on hyperledger fabric.' In: *International Conference on Applications and Techniques in Information Security, Springer.* (2019).

[34] Hasanova, Huru et al. 'A Survey on Blockchain Cybersecurity Vulnerabilities and Possible Countermeasures'. In: *Int. J. Netw. Manag.* 29.2 (Mar. 2019). URL: https://doi.org/10.1002/nem.2060.

[35] Lagarde, Marie-Jeanne. 'Security Assessment of Authentication and Authorization Mechanisms in Ethereum,Quorum, Hyperledger Fabric and Corda'. In: (2019).

[36] Putz, Benedikt and Pernul, Günther. 'Trust Factors and Insider Threats in Permissioned Distributed Ledgers'. In: *Transactions on Large-Scale Data- and Knowledge-Centered Systems XLII.* Ed. by Abdelkader Hameurlain and Roland Wagner. Berlin, Heidelberg: Springer Berlin Heidelberg, 2019, pp. 25–50.

[37] Wang, Shuo. 'Performance Evaluation of Hyperledger Fabric with Malicious Behaviour'. In: *Blockchain ICBC 2019, Springer* (2019), pp. 211–219.

[38] Shahriar, M. A. et al. 'Modelling Attacks in Blockchain Systems using Petri Nets'. In: *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. 2020, pp. 1069–1078.

[39] *Glossary.* URL: https://hyperledger-fabric.readthedocs.io/en/release-2.2/glossary.html.

[40] Petri, Carl Adam. 'Communication with automata'. In: *Schriften des Instituts fur Instrumentelle Mathematik, Bonn,* (1966).

[41] Davidrajuh, Reggie. *Modeling Discrete Event Systems with GPenSIM: An introduction.* Springer, 2020.

[42] Haustermann, Michael. URL: https://www2.informatik.uni-hamburg.de/TGI/PetriNets/index.php.

[43] Ciardo, Gianfranco and Trivedi, Kishor. 'Stochastic Reward Nets for Reliability Prediction'. In: *Communications in Reliability, Maintainability and Serviceability* 1 (Oct. 2002).

[44] Tivedi., K.S. 'SPNP User's Manual - Version 6.0.' In: *International Journal of Network Management* (1999).

[45] Castelle, M. 'Deep learning as an epistemic ensemble'. In: *https://castelle.org/pages/deep- learning- as- an- epistemic- ensemble.html* (2018).

[46] Gruver, Nate et al. 'Protein Design with Guided Discrete Diffusion'. In: *arXiv preprint arXiv:2305.20009* (2023).

[47] Dorjsembe, Zolnamar et al. 'Conditional Diffusion Models for Semantic 3D Medical Image Synthesis'. In: *arXiv preprint arXiv:2305.18453* (2023).

[48] Mardani, Morteza et al. 'A Variational Perspective on Solving Inverse Problems with Diffusion Models'. In: *arXiv preprint arXiv:2305.04391* (2023).

[49] Kim, Daegyu et al. 'Diffusion-Stego: Training-free Diffusion Generative Steganography via Message Projection'. In: *arXiv preprint arXiv:2305.18726* (2023).

[50] Ho, Jonathan, Jain, Ajay, and Abbeel, Pieter. 'Denoising diffusion probabilistic models'. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 6840–6851.

[51] Kotelnikov, Akim et al. 'TabDDPM: Modelling Tabular Data with Diffusion Models'. In: *arXiv preprint arXiv:2209.15421* (2022).

[52] Lee, Chaejeong, Kim, Jayoung, and Park, Noseong. 'CoDi: Co-evolving Contrastive Diffusion Models for Mixed-type Tabular Synthesis'. In: *arXiv preprint arXiv:2304.12654* (2023).

[53] Zheng, Shuhan and Charoenphakdee, Nontawat. 'Diffusion models for missing value imputation in tabular data'. In: *arXiv preprint arXiv:2210.17128* (2022).

[54] Nikolenko, Sergey I. *Synthetic data for deep learning*. Vol. 174. Springer, 2021.

[55] Qian, Zhaozhi, Cebere, Bogdan-Constantin, and Schaar, Mihaela van der. *Synthcity: facilitating innovative use cases of synthetic data in different data modalities.*

2023. DOI: 10.48550/ARXIV.2301.07573. URL: https://arxiv.org/abs/2301.07573.

[56] Baldi, Pierre et al. 'Assessing the accuracy of prediction algorithms for classification: an overview'. In: *Bioinformatics* 16.5 (2000), pp. 412–424.

[57] Jurman, Giuseppe, Riccadonna, Samantha, and Furlanello, Cesare. 'A comparison of MCC and CEN error measures in multi-class prediction'. In: (2012).

[58] Moustafa, R and Slay, J. 'A comprehensive data set for network intrusion detection systems'. In: *School of Engineering and Information Technology University of New South Wales at the Australian Defense Force Academy Canberra, Australia, UNSW-NB15* (2015).

[59] Pedregosa, F. et al. 'Scikit-learn: Machine Learning in Python'. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[60] Chen, Tianqi and Guestrin, Carlos. 'Xgboost: A scalable tree boosting system'. In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 2016, pp. 785–794.

[61] Xu, Lei et al. 'Modeling tabular data using conditional gan'. In: *Advances in neural information processing systems* 32 (2019).

[62] Dina, Ayesha Siddiqua, Siddique, AB, and Manivannan, D. 'Effect of balancing data using synthetic data on the performance of machine learning classifiers for intrusion detection in computer networks'. In: *IEEE Access* 10 (2022), pp. 96731–96747.

[63] Han, Songqiao et al. 'ADBench: Anomaly Detection Benchmark'. In: *Neural Information Processing Systems (NeurIPS)*. 2022.