Towards Attack-Tolerant Networks: Concurrent Multipath Routing and the Butterfly Network

Edward L. Platt
University of Michigan
elplatt@umich.edu

Daniel M. Romero
University of Michigan
drom@umich.edu

Abstract

Targeted attacks against network infrastructure are notoriously difficult to guard against. In the case of communication networks, such attacks can leave users vulnerable to censorship and surveillance, even when cryptography is used. Much of the existing work on network fault-tolerance focuses on random faults and does not apply to adversarial faults (attacks). Centralized networks have single points of failure by definition, leading to a growing popularity in decentralized architectures and protocols for greater fault-tolerance. However, centralized network *structure* can arise even when *protocols* are decentralized. Despite their decentralized protocols, the Internet and World-Wide Web have been shown both theoretically and historically to be highly susceptible to attack, in part due to emergent structural centralization. When single points of failure exist, they are potentially vulnerable to non-technological (i.e., coercive) attacks, suggesting the importance of a structural approach to attack-tolerance. We show how the assumption of partial trust transitivity, while more realistic than the assumption underlying webs of trust, can be used to quantify the effective redundancy of a network as a function of trust transitivity. We also prove that the effective redundancy of the wrap-around butterfly topology increases exponentially with trust transitivity and describe a novel concurrent multipath routing algorithm for constructing paths to utilize that redundancy. When portions of network structure can be dictated our results can be used to create scalable, attack-tolerant infrastructures. More generally, our results provide a theoretical formalism for evaluating the effects of network structure on adversarial fault-tolerance.

Keywords: Butterfly Topology, Fault Tolerance, Adversarial Faults, Multipath Routing, Censorship, Decentralization.

1 Introduction

> The Net interprets censorship as damage and routes around it.
>
> –*John Gilmore* [15]

Much of the world's infrastructure is networked: power grids, cellular networks, roads, and of course, the Internet. As with any critical infrastructure, the cost of failures can be immense, so methods for tolerating various kind of faults within networks are an important and ongoing area of research [46, 1, 40]. *Adversarial faults*, those in which an adversary can target attacks strategically, deserve special attention. Such attacks are both extremely difficult to guard against and often have important social implications. In particular, censorship and surveillance are often achieved by targeting central network locations and either blocking or capturing the information flowing through them. The Internet's decentralized design was motivated by the need to withstand targeted attacks, such as nuclear strikes [6]. But despite longstanding common wisdom [15], both theoretical results and recent events (described below) have demonstrated that the Internet can be surprisingly vulnerable to attack. Decentralization remains a promising approach towards building resilient networks, but there is a need to better understand the relationship between decentralized network structure and adversarial fault tolerance.

The Internet's vulnerability to censorship and other targeted attacks has been demonstrated by several recent events. In 2008, YouTube suffered a worldwide outage for several hours when a service provider in Pakistan advertised false routing information [19]. The action (known as a *black hole attack*) was intended to censor YouTube within Pakistan only, but resulted in a worldwide cascading failure. The action was initiated by government order in Pakistan, and spread beyond Pakistan when a router misconfiguration allowed the false routing information to propagate globally. While the government order and router misconfiguration initiated the outage, it was the structure of the Internet's router network that allowed a fault in a single router to propagate globally. And while the action was not an intentional attack against the global Internet, the ability of an attacker to succeed without even trying only highlights the Internet's vulnerability to adversarial faults.

In 2013, the Texas-based email provider Lavabit was ordered to disclose their private SSL keys to the FBI [36]. Rather than complying, Lavabit ceased operations in order to protect their users from surveillance. Once again, the attack was successful due to a highly centralized architecture: SSL keys under control of a single entity, in a single legal jurisdiction. While originally intended as surveillance, this action effectively became an act of censorship. It is also important to note that while Lavabit's cryptography worked as intended, the attack was still successful because the system was vulnerable to non-technical coercion. So we see that such vulnerabilities are not limited to any one system or protocol, but result from centralized structure itself.

Analysis of the Internet's router network has shown that while it is remarkably resilient against random faults, it is highly susceptible to adversarial faults [1]. These results have been attributed to the scale-free structure of the Internet's router network [4, 5]. In scale-free networks and other networks with heavy-tail degree distributions, random failures are highly likely to affect only low-degree nodes, thus having little effect. However, Adversarial faults target the few high-degree nodes, and therefore remove a large number of edges with each fault. So while the *protocols* of the Internet are decentralized, the *network structure* is somewhat centralized. In other words, the protocols of the Internet do not *require* centralization, but centralization may still emerge from the sociotechnical processes that create its network structure.

With strong theoretical and historical evidence that centralized network structure can create vulnerabilities, methods for analyzing structural vulnerabilities and for designing fault-tolerant networks are needed. This paper presents several contributions towards advancing those goals. While our motivation comes from the Internet router network and World Wide Web, our work is theoretical, focusing on abstract networks, and could potentially be applied to many different types of networks, whether made of physical wires, virtual tunnels, or other types of links.

We consider a setting in which a source node (Alice) in a network attempts to route a message to a target node (Bob) by forwarding it through the links of the network. A "link" in this context could represent any kind of connection (e.g., physical cables, encrypted channels). We assume that some nodes in the network may be compromised by an attacker (Mal). We also assume that Mal is an adversary of Alice specifically and targets nodes strategically with the goal of interfering with Alice's communications (rather than disrupting the network as a whole). This assumption applies to scalable network architectures that can be made large enough that an attacker must focus their resources in order to have a significant effect. Compromised nodes may behave incorrectly by blocking, altering, or incorrectly routing messages. We assume that Mal has full knowledge of the network structure, but has limited resources and thus can only compromise a fixed number of nodes.

We also assume that nodes *trust* their immediate neighbors. So Mal is unable to compromise Alice's node, or her direct neighbors. In the commonly used *web of trust* approach [45, 16], we would extend that trust transitively to the entire network. However, we make a weaker and more realistic assumption: that trust is extended transitively to nodes within a fixed number of hops. So only the beginning and end of a path between Alice and Bob is trusted against interference from Mal. We call this assumption *partial trust transitivity*, and refer to such paths as *partially trusted.*

Under the above assumptions, we show how to evaluate the influence of network structure on attack-tolerance, how to use local trust and redundancy to achieve greater attack-tolerance when no single path is fully trusted, and propose a novel routing algorithm for constructing such paths on the butterfly network topology. The butterfly topology is popular in parallel processing [26] and peer-to-peer [31, 25] applications, due to its regular structure, low degree, and high connectivity.

It is important to note that the butterfly is a highly structured and constrained network topology, very different form those found in social networks and other self-organized networks. The reader may wonder whether it is realistic or useful to assume such control over the network structure. We have already seen that whenever a single point of failure exists in the network, there is a potential for an attacker to exploit it through coercion, without needing to compromise the technology. So, *attack-tolerance cannot be achieved without the ability to influence network structure.* Luckily, there are scenarios in which network topology can be dictated. Examples include overlay networks [31, 25], formal organizations [33], government-regulated cellular networks [42], and call tree notification systems [35]. In general, *when the need for attack-tolerance is high enough to warrant investment in infrastructure, networks can be engineered and maintained as infrastructure.* It is also worth noting that attack-tolerant networks may be sub-components of larger, less-constrained systems. For example, a single server might be replaced by a distributed network of servers, each with different ownership, physical location, and legal jurisdiction, without placing any unrealistic constraints on the clients connecting to those servers. Additionally, there may be ways to achieve improved attack-tolerance from architectures more flexible than the butterfly, which is a potential area for future work.

We begin by describing how fault tolerance techniques can be adapted and evaluated in a network setting with partial trust transitivity and adversarial faults. Generally, faults in network paths can be correlated, preventing the application of standard fault tolerance techniques [3, 41], which assume independent faults. By constructing *independent paths*, which have no untrusted nodes in common, we show how to model communication across a complex network in the presence of correlated adversarial faults as communication across redundant simple channels with random errors. Redundant messages can be sent across these channels in parallel, a technique known as *concurrent multipath routing* [46, 37, 21], and used for fault tolerance. The receiver can then use the redundant messages to detect and/or correct errors. We formally evaluate the effects of network structure on attack-tolerance and show that the probability of an undetected error decreases exponentially with the number of independent paths between source and destination, even when no individual path is entirely trusted.

We also propose a novel concurrent multipath routing algorithm for the butterfly topology. The algorithm constructs independent paths, which can be combined with the fault-tolerant concurrent multipath routing scheme above to achieve a high level of adversarial fault tolerance on the butterfly topology.

Our main contributions are:

- We propose a novel method for extending standard fault tolerance techniques to *adversarial* faults in *complex networks*. We do so by modeling redundant independent paths with partial trust transitivity as a single virtual channel, and show that the probability of detecting adversarial faults approaches 1 exponentially with the number of paths;

- We prove that the number of independent paths between two nodes in a wrap-around butterfly network with partial trust transitivity increases exponentially with the trust radius;

- We present a scalable, efficient, and attack-tolerant concurrent multipath routing algorithm on the butterfly network topology.

This paper is organized as follows. Section 2 reviews background and related work. Section 3 describes adversarial fault tolerance on structured networks. Section 4 describes the concurrent multipath routing algorithm for the butterfly network topology Section 5 discusses the results. And Section 6 concludes.

2 Background and Related Work

There has been considerable work on trust in network security. Both centralized and decentralized approaches are commonly used to create trust infrastructures. Centralized approaches such as *public key infrastructure* (PKI) suffer from a number of vulnerabilities [14], which stem largely from the single points of failure inherent to centralization. The well-known and widely-used *web of trust* approach [45, 16] is a decentralized alternative. In a web of trust, individuals choose who they trust initially. Trust is then extended to new individuals if they are vouched for by a currently-trusted individual, making it possible to quickly establish a large group of trusted nodes. However, web of trust's assumption of infinite transitivity is unrealistic [10], and does not distinguish between paths of different lengths. Our work addresses both of these limitations by incorporating a more realistic assumption of partial transitivity.

Previous work on incorporating network structure into trust models has focused on authentication protocols, showing that independent paths can reduce an adversary's ability to impersonate a target [28]. Other work has shown that identifying independent paths in arbitrary networks is NP-hard and provided approximation algorithms [38]. Our work complements these by introducing the partial trust assumption extending the focus beyond authentication. When network topology can be controlled, we sidestep the NP-hard problem of finding independent paths on arbitrary networks by using the mathematical structure of the butterfly topology to construct provably independent paths.

Many distributed consensus protocols (such as those used by cryptocurrencies) are designed to tolerate arbitrary or adversarial faults. Byzantine agreement protocols [27, 8] provide tolerance against arbitrary faults (including attacks) under some circumstances, but are limited to small networks due to poor scalability. Proof-of-work [13, 34] and proof-of-stake [22] provide better scalability, but are wasteful of computational and energy resources. Federated Byzantine Agreement (FBA) [32] is scalable, allows for flexible trust, and is highly fault-tolerant on networks meeting a set of requirements. However, FBA does not provide a method for evaluating the fault tolerance properties of different network structures or for calculating the failure probabilities within a particular network.

There are relatively few attack-tolerance schemes that focus on network structure, compared to more popular cryptographic approaches [16]. All existing attack-tolerant networks we are aware of are content-addressable networks: data is routed to and from storage nodes rather than between sender and receiver. Fiat and Saia described a scheme that combines the butterfly topology with expander graphs to create a highly censorship-resistant, content-addressable network [17], although this scheme does not scale well and is impractical due to a high level of data replication. Perhaps the most mature structural solution is the Freenet collaboration [11]. Freenet uses secret sharing [39, 7] and small-world routing [44, 23] to create a content-addressable network with a high level of both confidentiality and censorship resistance. Freenet guarantees that data is stored redundantly, but still allows for centralized network structure, and thus single points of failure, as data travels from its origin to the redundant storage locations. Unlike the above content-addressable networks, our proposal is purely network based and does not require nodes to store data indefinitely. Our proposal also improves on the scalablity of Fiat and Saia's work, and does not rely on assumptions about existing social network structure.

Multipath routing protocols identify multiple paths between source and destination in contrast to traditional *unipath* routing, which uses a single path. The special case of *concurrent* multipath routing uses mutliple paths simultaneously. Multipath routing has many applications, including reduced congestion, increased throughput, and more reliability [37]. Many of these routing protocols offer increased confidentiality [46]. Some approaches utilize redundant paths as backups for increased fault tolerance [2], and some specifically protect against adversarial faults [24, 20, 30]. Most work on multipath routing has been motivated by applications related to wireless sensor networks (WSNs), and have thus focused on ad hoc, unstructured networks, often having a central base station. The method of Liu et al. [29] routes multiple messages first to random peers and then to a central base station, with the network edges constrained by sensors' physical location. We have found only few examples in the existing literature of applications of concurrent multipath routing to *adversarial* fault tolerance, and all have focused on ad-hoc wireless sensor networks, without attention to the role of network structure. The alogorithm we present for the butterfly topology complements existing work by addressing cases where links are not constrained by physical distance, and where network structure can be engineered for greater attack-tolerance.

Our proposed routing algorithm makes use of a *structured network*, in which link structure is predetermined. Structured networks have been a popular tool in parallel processing architectures [26]. More recently, peer-to-peer systems based on distributed hash tables have used structured *overlay networks* to map table keys to local TCP/IP routes [31, 25]. Such networks can be designed to have favorable structural and routing properties, which can be used to to improve attack-tolerance.

3 Trust Networks and Fault Tolerance

Within the field of *fault tolerance*, many techniques have been developed for building reliable systems out of unreliable components [3, 41]. We will make use of standard fault tolerance terminology, summarized here. A *fault* is said to occur when one component of a system behaves incorrectly (e.g., a routing node blocks or alters a message). The result of that fault (e.g., a recipient receiving conflicting messages) is called an *error* state. If the error is undetected or corrected to the wrong value, the system is said to have experienced a *failure* (e.g., an altered message is accepted as authentic). Note that when an error is detected but cannot be corrected, the system has still tolerated the fault because it has not accepted an error state. We are concerned in particular with *adversarial faults*, which are chosen strategically to maximize the likelihood of a failure.

3.1 Partial Trust Model

A central question in large-scale, secure communication is this: how can two parties communicate reliably and securely when no direct trusted link exists between them? The commonly-used web of trust approach [45, 16] extends trust infinitely transitively: to friends of friends, and friends of friends of friends, and so on. However, the assumption of infinitely transitive trust is unrealistic [10], and does not allow for the analysis of the effects of network structure.

An alternative assumption might be that each hop away from Alice in in the network reduces the probability that a node can resist compromise exponentially. Such a situation could occur if nodes more distant from Alice are more favorably disposed to Alice's adversary, more likely to cooperate with that adversary, or less likely to take proactive security measures against that adversary. The above model can be further approximated by assuming that nodes up to some fixed number of hops cannot be compromised, and that those beyond can. This simplified version is still more realistic than infinite transitivity and will be convenient for proving our results. We now proceed to define our model formally.

We define the *partial trust model* on an undirected graph $G = (V, E)$, although the model can easily be extended to directed multigraphs. Vertices representing commiunicating agents, and with edges representing mutually trusted communication links. Let $v \in V$ be an arbitrary sender (Alice) and $w \in V$ be an arbitrary receiver (Bob). We assume the presence of an adversary (Mal) who knows the full structure of the network, and who can compromise a fixed number of nodes, gaining complete control of their behavior. We also assume that

Mal is an adversary of Alice and/or Bob specifically, rather than the network as a whole. So Mal can compromise any node except for those trusted by Alice or Bob. We define a *trust radius* h such that nodes u and u' trust each other if their distance is less than h. For a given node u, we call the set of trusted nodes its *trusted neighborhood* $T_h(u)$, and all nodes at exactly distance h the *trust boundary* $B_h(u)$:

$$\begin{aligned} T_h(u) &= \{u' \mid d(u,u') < h\} && (1) \\ B_h(u) &= \{u' \mid d(u,u') = h\}\,. && (2) \end{aligned}$$

The trust boundary B_h plays an important role because these nodes are not trusted by u, and if compromised can entirely isolate u from the rest of the network. These trust assumptions imply that when Alice sends a message to Bob, Mal can only cause faults in the set of nodes outside both of their trusted neighborhoods: $V \setminus (T_h(v) \cup T_h(w))$. We refer to this set of nodes as the *untrusted region*.

3.2 Effective Redundancy

Our goal is to achieve fault tolerance through redundancy. In the network setting, redundnacy is achieved using *independent paths* [38], which have no common points of failure. Typically, it is assumed that paths must be disjoint in order to be independent. However, under the partial trust assumption, two non-disjoint paths can still be independent as long as their intersection contains only trusted nodes, greatly increasing the level of redundancy available. Under the partial trust assumption, the available redundancy thus depends on both the network structure and the level of trust.

We now quantify the *effective redundancy* between Alice and Bob when trust radius h is assumed. This quantity, $\delta_{v,w,h}$ is exactly the max-flow/min-cut of the graph after Alice's and Bob's trusted neighborhoods have been collapsed into single source/sink vertices. Each trust boundary forms a cut of the network and places an upper bound on the min-cut:

$$\delta_{v,w,h} \leq \min\left(\mid B_h(v) \mid, \mid B_h(w) \mid\right). \qquad (3)$$

Equality holds when there are no bottlenecks within the untrusted region, an indication that the network is decentralized. The efrective redundancy of the entire graph can be characterized by the minimum over all vertex pairs:

$$\delta_h(G) \equiv \min_{v,w \in V} \delta_{v,w,h}. \qquad (4)$$

Thus, for any pair of nodes in the network, at least δ_h independent, redundant paths can be constructed between them. δ_h is a purely structural network property, and places an upper bound on the effectiveness of any redundancy-based fault tolerance scheme. The more quickly δ_h grows with h, the better a network is at leveraging trust transitivity to create redundancy. Thus, the scaling of δ_h can be used to quantify a network's ability to withstand targeted attacks, even when the exact trust radius h is unknown.

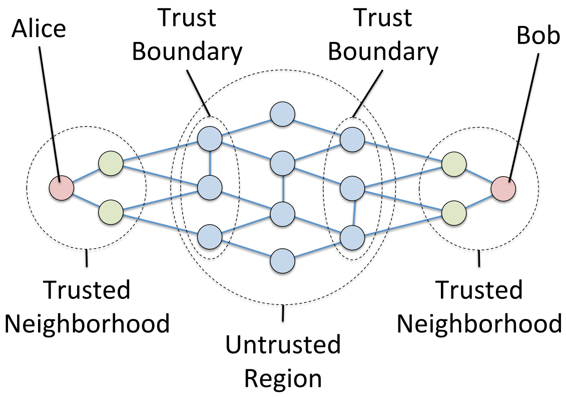

Figure 1: Illustration of a trusted communication network and the network properties used by the *partial trust model*. Edges represent mutually trusted communication links. The sender (Alice, v) and receiver (Bob, w) trust all nodes less than the *trust radius* h hops away. These nodes form their *trusted neighborhoods* $T_h(v)$ and $T_h(w)$. We assume that all faults occur in the remaining nodes: the *untrusted region*. The untrusted nodes in contact with the trusted neighborhoods for the *trust boundaries* $B_h(v)$ and $B_h(w)$, which (in the absence of central bottlenecks) determine the *effective redundancy* δ_h provided by the network. Alice and Bob can be modelled as connected by a direct link with at least δ_h redundant channels.

3.3 Multipath Fault Tolerance

Once we have determined a network's effective redundancy, we can apply redundancy-based fault tolerance techniques, by sending multiple copies of a message (*concurrent multipath routing*). Having found the effective redundancy between two nodes, we can simplify our model, replacing independent paths through the complex network with direct channels between the endpoints. We model our sender (Alice) and receiver (Bob) as communicating over δ_h direct and redundant virtual channels. The partial trust model allows us to make this simplifying assumption for analyzing a fault tolerance scheme, but implementing such a scheme will require a method for constructing specific network paths. We will return to the question of constructing paths in the next section. For now, we concern ourselves with the question: given that the network provides δ_h redundant channels between Alice and Bob, what is the probability that an adversary (Mal) causes an undetectable error after inducing a fixed number of faults?

Let us first consider the scenario in which Alice sends a message copy over each available channel. We can also assume that each message includes the number of messages sent, the full list of channels used, etc., making that information available to Bob. When Bob receives the messages, there are several possibilities. If some of the messages are missing or if some of the messages disagree, Bob knows that some of the messages were either blocked or altered, and he has successfully tolerated the fault(s). Bob can then take any of several actions: 1. request retransmission; 2. end receipts so Alice knows which paths have been compromised; or 3. attempt error correction using majority voting. If instead, Bob finds that all the messages are present and agree, there are two possible cases. The first case is that Mal has not compromised any of the messages, and Bob has correctly accepted them, so no failure has occurred. The second case is that Mal has compromised *all* of the messages, so Bob has accepted an erroneous message and a failure has occurred. In the present scenario, whether a failure occurs depends only on whether Mal has the resources to compromise all of the channels. In a more realistic scenario, both Alice and Mal have limited resources and are not able to use or compromise all available channels.

In a more sophisticated multipath fault tolerance scheme, Alice randomly chooses $k \leq \delta_h$ channels and sends a copy of her message on each. We assume that Mal is capable of compromising $l \leq \delta_h$ channels. Since Alice chooses channels randomly, all channels are equally likely to contain a message, so Mal can do no better than also choosing randomly. We can also return to the full network setting by noting that each of the δ_h independent paths in the network can serve as independent channels between Alice and Bob. Mal's best strategy is now to identify a minimum node cut in the network and randomly compromise nodes from that cut. With this strategy, each compromised node reduces effective redundancy by one, equivalent to compromising one of the channels between Alice and Bob. If $k > l$, at least one message will get through uncompromised and all errors are detectable. Otherwise, the probability of Mal producing an undetectable error is the probability that all of Alice's chosen channels are compromised:

$$p_f = \frac{l!(\delta_h - k)!}{\delta_h!(l-k)!}. \tag{5}$$

Letting $k = \alpha\delta_h$ and $l = \beta\delta_h$, then applying Stirling's approximation gives:

$$p_f \approx \frac{\sqrt{\beta(1-\alpha)}}{\sqrt{\beta-\alpha}} \left[\left(\frac{\beta-\alpha}{1-\alpha}\right)^{\alpha} \left(\frac{\beta}{\beta-\alpha}\right)^{\beta} (1-\alpha) \right]^{\delta_h} \tag{6}$$

Figure 2 shows the value of p_f as a function of k and l. Equation (6) shows that while p_f depends on the fractions of redundant paths actually utilized α and compromised β, it decreases exponentially with the effective redundancy δ_h (which we will later see increases exponentially in h in the butterfly topology). This result is significant because δ_h depends only on the network structure and the strength of trust transitivity. *Thus, the scheme can be effective, even when the number of copies sent k is a small fraction of the effective redundancy.* In other words, this scheme exhibits a *stabilizing asymmetry*: senders can tolerate attacks from significantly more powerful adversaries, as long as the network structure provides large δ_h.

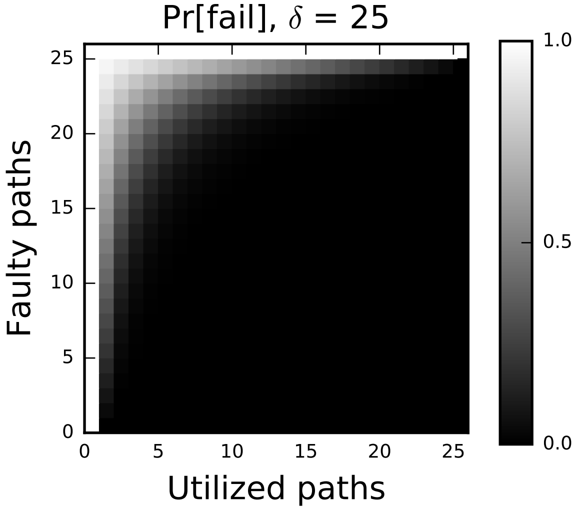

Figure 2: The probability of an undetectable error as a function of the number of message copies and the number of adversarial faults.

In order to derive the above results, we have assumed that Alice and all intermediary agents are able to identify specific, independent network paths that achieve the effective redundancy δ_h. We now proceed to describe a routing algorithm for doing so in the special case of the butterfly network topology.

4 Multipath Butterfly Routing

In previous sections, we showed that reliable communication across a network can be achieved even when any single message path might be compromised by an adversary, provided the network has sufficient redundancy, and provided the sender and intermediaries know how to route message copies along independent paths. In this section, we address both requirements by proposing a novel routing algorithm for constructing independent paths on the butterfly network topology. This architecture and routing algorithm achieve an effective redundancy that increases exponentially with the trust radius, allowing a very high level of adversarial fault tolerance.

The structure of the butterfly network is highly constrained, making it most suitable for applications where portions of the network structure can be designed or dictated. Examples of such networks include: overlay networks [31, 25], formal organizations [33], government-regulated cellular networks [42], and call tree notification systems [35]. However, when attack-tolerance is desired, it will always require control over network structure in order to eliminate single points of failure. The regular structure of the butterfly is not a limitation of our approach, but rather a reflection of the inherent difficulty of attack-tolerance. Lastly, we note that the partial trust model and multipath fault tolerance schemes of the previous section do not rely on any particular network topology or routing algorithm, and our choice of the butterfly topology is only one of many possible choices.

4.1 Butterfly Network Topology

We choose the butterfly topology [26] because of several desirable properties (described below) and because its structure allows for relatively straightforward design and analysis of routing algorithms. While several variations on the butterfly network exist, we utilize the wrap-around butterfly. We denote the m-dimensional, directed wrap-around butterfly as a graph wBF(m):

$$\begin{aligned} \mathrm{wBF}(m) &= (V, E_{\downarrow} \cup E_{\rightarrow}) & (7)\\ V &= \mathbb{Z}_m \times \mathbb{Z}_2^m & (8)\\ E_{\downarrow} &= \{((l,z),(l+1 \pmod{m}, z)\} & (9)\\ E_{\rightarrow} &= \{(l,z),(l+1 \pmod{m}, z \oplus 1_l\}, & (10)\end{aligned}$$

where $\mathbb{Z}_m$ is the set of integers modulo m, $\oplus$ represents componentwise addition modulo 2, and 1_l is a vector with a 1 in index l and 0 elsewhere. Each node is associated with a level l and an m-bit string z known as *the place-within-level*. There are two types of edges: down, and down-right (shown in Figure 3). Down edges ($E_{\downarrow}$) connect nodes sharing the same z value in a cycle of increasing level l. Down-right edges ($E_{\rightarrow}$) also link to a node of level $l+1$, but one having the place-within-level equal to z with the lth bit inverted.

The wrap-around butterfly network is known to have several of the properties we desire for scalable, decentralized communication networks:

Vertex-transitivity: Because the wrap-around butterfly is vertex transitive, it is maximally decentralized;

Small-diameter: For any two nodes, the length of the shortest path between them is $O(\log N)$, where N is the number of nodes in the network;

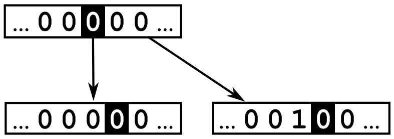

Figure 3: Schematic illustration of the two types of edges in a directed butterfly network. The node (l, z) is shown as the bit string z with a square around the lth bit. "Down" edges increment l, leaving z unchanged, while "down-right" edges increment l and invert the lth bit of z. In the wrap-around variant, the nodes with maximum l have down and down-right edges to the nodes with $l = 0$.

Sparsity: With a constant degree of 4, the wrap-around butterfly is extremely sparse, and can scale indefinitely without node degree becoming a limitation;

Redundancy: Multiple paths exist between any two nodes. Specifically, we will prove below that the number of independent paths between two nodes increases exponentially with the trust radius h.

The structure of the butterfly network lends itself to a well-known (unipath) routing algorithm, which we later extend to the multipath case. The unipath algorithm first follows a down or down-right edge at every step, increasing the level l by 1 and cycling through the indices of the place-within-level. If the current node's place-within-level matches the destination node's at index l, a down edge is chosen and the place-within-level does not change. Otherwise, a down-right edge is chosen and the lth component of the place-within-level is flipped, after which it matches the destination. After m iterations of this, all levels have been visited and the place-within-level matches that of the destination. Simply following down (or up) edges will then increment (decrement) the level until the destination node is reached.

4.2 Multipath Routing Algorithm

We now present a routing algorithm to construct 2^h independent paths between two nodes in a butterfly network, where h is the trust radius under the partial trust model. Informally, Alice sends each message to a distinct node on her trust boundary, then to a distinct intermediate node in the untrusted region, then to a distinct node on Bob's trust boundary, and finally to Bob. The intermediate nodes are in a sense "far" from each other and ensure that no two paths overlap in the untrusted region. Each path can be parameterized by a single integer s, which identifies the specific node on Alice's trust boundary (or equivalently the node on Bob's trust boundary, or the untrusted intermdiate).

The algorithm guarantees paths are independent by ensuring that (outside the trusted neighborhoods) they only include nodes that match the path parameter s at certain indexes in their place-within-level. Since each path has a unique parameter s, its set of untrusted nodes is disjoint from all other paths. As with the unipath routing algorithm, each of the multiple paths proceed from a source v to a destination u using down and down-right edges, cycling through levels one at a time. However, we cycle through the levels twice, once to route from v to a particular path's intermediary node, and again to route from the intermediary to w. Each cycle is divided into stages, with different properties used to prove independence at each stage (see Figure 4). In the first cycle (stages 1–4), path independence is guaranteed by ensuring that all nodes match the path parameter s in the first h bits of the place-within-level. Similarly, in the second cycle (stages 5–7), independence is guaranteed by ensuring that all paths match s in the h bits of the place-within-level preceding the destination index. A full example is illustrated in Figure 5.

4.2.1 Algorithm Specification

We now begin the formal specification of our multipath routing scheme for the wrap-around butterfly network. For convenience, the relevant variables are summarized in Table 1. Utilizing vertex transitivity, we label the source node as $(l^{(0)}, z^{(0)}) = (0, 0)$ and denote the destination node as $w = (l_w, z_w)$, without loss of generality.

Let s be an h-bit binary string with s_i denoting the bit at index i. There are 2^h such strings. Let $v_s^{(t)} = (l^{(t)}, z^{(t)})$ be the node at position t in the path parameterized by s. For convenience, we will omit the subscript s when it is obvious from context. We define three distinct partitions of m-bit binary strings. Let $Q_{v^{(0)}}$ be the set of m-bit strings in which the bits at all indices $h \le i < l_w - h$ match those of $z^{(0)}$, and let $\overline{Q_{v^{(0)}}}$ be its complement. Note that $Q_{v^{(0)}}$ is trivially all m-bit strings if $l_w < 2h$. Let R_s be the set of m-bit strings with the lowest h bits all matching the bits of s, and let $\overline{R_s}$ be its complement. Let S_s be the set of m-bit strings with the h bits preceding index l_w all matching the bits of $\tilde{s}$, where $\tilde{s}$ is a cyclic permutation of s:

$$\tilde{s}_i \quad = \quad s_{(i+l_w) \bmod h}, \tag{11}$$

and let $\overline{S_s}$ be its complement. We will make use of the fact that:

$$s \neq s' \quad \Longrightarrow \quad S_s \cap S_{s'} = R_s \cap R_{s'} = \emptyset. \tag{12}$$

Routes are constructed in 7 stages. The network topology dictates that $l^{(t+1)} = l^{(t)} + 1 \pmod{m}$, so we let $l = t \pmod{m}$. and that $z^{(t+1)}$ is equal to $z^{(t)}$ with or without the bit in index $l^{(t)}$ inverted, depending on whether the down or down-right edge was taken at step t.

	A	B	C	D
	h	$l_w - 2h$	h	$m - l_w$
start	$0\ldots$	$\ldots0\ldots$	$\ldots0\ldots$	$\ldots0$
1.	s	$\ldots0\ldots$	$\ldots0\ldots$	$\ldots0$
2.	s	$\ldots1\ldots$	$\ldots0\ldots$	$\ldots0$
3.	s	$\ldots1\ldots$	$\tilde{s}$	$\ldots0$
4.	s	$\ldots1\ldots$	$\tilde{s}$	$z_{w,D}$
5.	$z_{w,A}$	$\ldots1\ldots$	$\tilde{s}$	$z_{w,D}$
6.	$z_{w,A}$	$z_{w,B}$	$\tilde{s}$	$z_{w,D}$
7.	$z_{w,A}$	$z_{w,B}$	$z_{w,C}$	$z_{w,D}$

Figure 4: Progression of place-within-level z as the multipath routing algorithm cycles through the levels of the butterfly network.

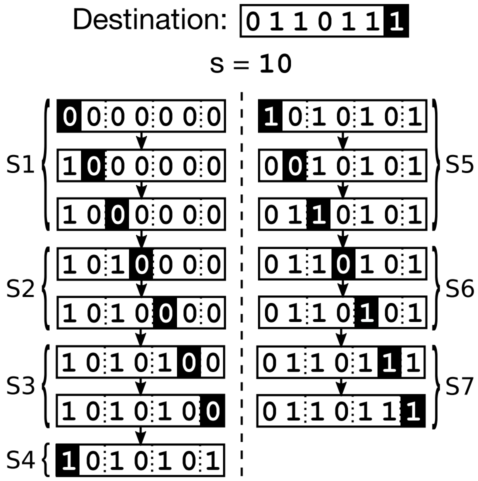

Figure 5: An example of one path as constructed by the proposed multipath routing algorithm. The path is shown for $s = 10_2$ and $w = (6, 0110111_2)$.

Table 1: Butterfly Multipath Routing Variables

NAME	VARIABLE
butterfly dimension	$m \in \mathbb{Z}_+$
node level	$l \in \mathbb{Z} : 0 \le l < m$
node place within level	$z \in \mathbb{Z}_2^m$
trust radius	$h \in \mathbb{Z} : 1 \le h \le \lfloor m/2 \rfloor$
path index	$s \in \mathbb{Z}_2^h$

Stage 1: $(0 \le t < h)$ Down or down-right edges are chosen such that the tth bit of $z^{(t+1)}$ is equal to the tth bit of s. Throughout Stage 1, all nodes are within the sender's trusted neighborhood. Throughout Stage 1, $z^{(t)} \in Q_{v^{(0)}}$. At the

end of Stage 1, $z^{(h)} \in S_s$, and $z^{(t)}$ will remain so until the level cycles to 0 at $t = m$.

Stage 2: $(h \leq t < l_w - h)$ Edges are chosen to make the tth bit of $z^{(t+1)}$ the inverse of the tth bit of $z^{(0)}$. Note that this stage does not occur when $l_w < 2h$. If this stage occurs, then $z^{(t)} \in \overline{Q_{v^{(0)}}}$ until these levels are reached again in stage 6.

Stage 3: $(l_w - h \leq t < l_w)$ The bits of $z^{(t)}$ are chosen to match $\tilde{s}$, such that after the stage is complete, $z^{(t)} \in R_s$.

Stage 4: $(l_w \leq t < m)$ Paths are chosen such that the tth bit of $z^{(t+1)}$ matches that of the destination node z_w. This stage will not occur if $l_w > m - h$.

Stage 5: $(m \leq t < m + h)$ There are two cases. If $2h < l_w < m - h$, then there is no overlap between the indices defining R_s and S_s. In this case, the first h bits of $z^{(t)}$ are set to match z_w. Otherwise there is some overlap between the indices defining R_s and S_s. In this case, the each of the first h bits of $z^{(t)}$ is either kept the same if $l_w - h \leq l < l_w$, or set to the corresponding bit of z_w otherwise. In this stage and after, $z^{(t)}$ is no longer guaranteed to be in R_s. However, $z^{(t)}$ remains in S_s during and after this stage.

Stage 6: $(m + h \leq t < m + l_w - h)$ In this stage, edges are chosen to set the bits of $z^{(t)}$ to their corresponding value in z_w. $z^{(t)} \in \overline{Q_{v^{(0)}}}$ throughout this stage, but not afterwards.

Stage 7: $(m + l_w - h \leq t < m + l_w)$ The h bits of $z^{(t)}$ preceding index l_w are set to match z_w. All nodes in this stage are within h hops of w and thus in its trusted neighborhood. After this stage, $v^{(m+l_w)} = w$ and routing is complete.

4.2.2 Proof of Path Independence

Theorem 1. *Given an m-bit wrap-around butterfly network ($m > 1$), and a radius h ($1 \leq h \leq \lfloor \frac{m}{2} \rfloor$), for all node pairs (v, w) such that $d(v, w) \geq 2h$, there exist 2^h paths v_s ($0 \leq s < 2^h$) from v to w such that $s \neq s' \implies v_s \cap v_{s'} \subset T_h(u) \cup T_h(v)$.*

Proof. Nodes from two paths can only coincide if their levels are the same. Nodes which share a level must either be in the same stage, or 4 stages apart. Let (a,a') denote a pair of sub-paths corresponding to stage a of one path and stage a' of another. Excluding paths that intersect in their trusted neighborhoods, (1,1) and (7,7), we have reduced the list of possible intersections to the following cases: (2,2), (3,3), (4,4), (5,5), (6,6), (1,5), (2,6), and (3,7). Nodes in stages 2–4 belong to R_s so cannot overlap with any stage 2–4 nodes from another path, eliminating (2,2), (3,3), and (4,4). Similarly, nodes in stages 4–6 belong to a unique S_s, eliminating (5,5) and (6,6). Nodes in stage 1 belong to $Q_{v^{(0)}}$ while those in stage 5 belong in its complement, eliminating (1,5). Similarly, for all l in stage 2, $z^{(l)}$ is equal to $z^{(0)}$, while in stage 6, $z^{(l)}$ is the inverse, eliminating (2,6) This leaves only (3,7), a collision which can occur only for only one path (with s matching the first h bits of z_w), and which enters the trusted neighborhood in stage 3. For this single path, we can proceed directly from stage 2 to stage 7, eliminating the last possible collision. □

Thus, assuming the partial trust model with trust transitive for h hops, we can construct 2^h paths on a wrap-around butterfly topology which do not intersect outside the trusted neighborhoods of the source and destination. Note that the node sequence $v_s^{(t)}$ can be calculated entirely from the source v, destination w, and path parameter s, meaning that with this information nodes are able to determine which neighbor to route a given message copy to. Furthermore, the existence of 2^h paths places a lower bound on the effective redundancy δ_h, showing that the decentralized, redundant, structured networks such as the butterfly can have a very low probability of failure when faced with adversarial faults, even from a very powerful attacker.

5 Discussion

While decentralized protocols have received much attention for their potential fault tolerance applications, centralized structures are always vulnerable to exploitation by non-technical means (i.e., coercion), and there is a need for a better understanding of the relationship between network structure and attack-tolerance. We have proposed a network-based scheme for *adversarial* fault tolerance on the butterfly topology, utilizing a novel concurrent multipath routing algorithm. We have also demonstrated how *partial trust transitivity*, in addition to being more realistic than infinite transitivity, provides a theoretical foundation for quantitative analysis of the relationship between trust, network structure, and attack tolerance.

Such attacks include many forms of censorship and surveillance, which have important social implications. We have already discussed two such cases: Pakistan's inadvertant censorship of YouTube [19] and the FBI's surveillance-turned-censorship of Lavabit [36]. The reader may wonder how our methods could be employed in scenarios such as large-scale state-sponsored censorship. Censorship-resistant infrastructure often replaces central servers (e.g., the router in the 2008 YouTube incident) with multiple servers across the world, synchronized through consensus protocols. The *directory authorities* used by the Tor project [12] are one example. However, the size of such authority networks is often limited by the number of trusted relationships (degree) each node can maintain, and the inherent insecurity of extending transitive trust to an ever-larger network. Our work fills a much-needed gap by quantifying the connection between network-structure, trust transitivity, and attack-tolerance. We provide both a theoretical framework and specific example of how network structure can be engineered to leverage trust for a high level of attack-tolerance, without sacrificing scalability.

Fault-tolerant network infrastructures have many direct applications. Areas such as cryptocurrency [32, 34, 22], secure multiparty computation [43, 9, 18], and wireless sensor networks [21] have immediate need for scalable, fault-tolerant infrastructures. Many Internet services (e.g., email, social networks, cloud storage) are still highly centralized and vulnerable to technical and non-technical (i.e., coercive) attacks. Fault tolerance using *both* decentralized protocols and decentralized network structures is one promising approach to securing these services.

We have focused primarily on adversarial faults that block or change messages (e.g., censorship). Existing cryptographic techniques for circumventing surveillance are relatively mature compared to those for tolerating censorship. However, the techniques presented in this paper are entirely compatible with, and in some cases could enhance, existing anti-surveillance techniques. For example, *man-in-the-middle* attacks exploit a privileged network position to attack otherwise secure cryptography, suggesting that structural approaches can complement cryptographic ones.

While our present proposal is specific to the butterfly topology, the multipath fault tolerance scheme could be applied to any network that has both sufficient redundancy and a routing algorithm to discover independent paths. For general networks, finding all such paths is NP-hard, but efficient, suboptimal algorithms exist [38]. However, we have argued that attack-tolerance requires the ability to influence network structure and reduce reilance on single points of failure. Our work is most applicable to cases where the need for attack-tolerance justifies investment in deliberate infrastructure. For example, a coalition of groups supporting free expression could use our work to construct a censorship-resistant communication network. In general, such groups would need to invest resources into vetting their neighbors to establish trust, but there are scenarios in which the attack-tolerance requirement would justify that investment. It is also worth noting that because faulty paths can be identified in our scheme, it may at times be appropriate to begin by assuming mutual trust and revoking that relationship if it is violated. Any entities dependent on the proper functioning of the network would have an incentive to resist attack in order to maintain their ability to participate.

Our work suggests several directions for future work towards developing practical, attack-tolerant communication infrastructure. The development of new multipath routing algorithms on other structured networks could achieve higher levels of redundancy. It is also desirable to identify dynamics that give rise to structured networks, and to evaluate whether our results can be generalized to unstructured or approximately structured networks. Finally, these results could be implemented to address specific applications, e.g., secure messaging, domain name resolution, or anonymous web browsing.

6 Conclusion

We have presented a novel concurrent multipath routing algorithm for the butterfly topology, as well as a scheme for using this algorithm to construct a highly attack-tolerant virtual channel between any two nodes, even when no fully-trusted path exists between them. Under this scheme, the probability of an adversary causing an undetectable error decreases exponentially with the network's effective redundancy. The effective redundancy, in the case of the butterfly topology, grows exponentially with the trust radius. Furthermore, a small increase in the number of messages sent can compensate for a large increase in the number of messages compromised by an adversary. We have also demonstrated how the assumption of partial trust transitivity can enable a quantitative analysis of the relationships between network structure, trust, and attack-tolerance. These results are directly applicable to systems in which the link structure can be imposed by the designer, and more generally, provide a theoretical foudnation that can be used more to evaluate the role of network structure, trust transitivity, and effective redundancy on attack-tolerance.

7 Acknowledgments

The authors would like to thank Tony Garnock-Jones, A. Frederick Dudley, and Nathaniel Bezanson for helpful conversations. This research was partly supported by the National Science Foundation under Grant No. IIS-1617820.

References

[1] R. Albert, H. Jeong, and A.-L. Barabási. Error and attack tolerance of complex networks. *Nature*, 406(6794):378–382, 2000.

[2] N. A. Alrajeh, M. S. Alabed, and M. S. Elwahiby. Secure ant-based routing protocol for wireless sensor network. *Int J Distrib Sens N*, 2013, 2013.

[3] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE T Depend Secure*, 1(1):11–33, 2004.

[4] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

[5] A.-L. Barabási and others. Scale-free networks: a decade and beyond. *Science*, 325(5939):412, 2009.

[6] P. Baran and others. On distributed communications. *Volumes I-XI, RAND Corporation Research Documents, August*, pages 637–648, 1964.

[7] G. R. Blakley. Safeguarding cryptographic keys. *P Natl Comp Conf*, 48:313–317, 1979.

[8] M. Castro, B. Liskov, and others. Practical Byzantine fault tolerance. In *OSDI*, volume 99, pages 173–186, 1999.

[9] D. Chaum, C. Crépeau, and I. Damgard. Multiparty unconditionally secure protocols. In *STOC*, pages 11–19. ACM, 1988.

[10] B. Christianson and W. S. Harbison. Why isn't trust transitive? In *Security protocols*, pages 171–176. Springer, 1997.

[11] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. Freenet: A distributed anonymous information storage and retrieval system. In *Designing Privacy Enhancing Technologies*, pages 46–66. Springer, 2001.

[12] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. Technical report, DTIC Document, 2004.

[13] C. Dwork and M. Naor. Pricing via processing or combatting junk mail. In *Advances in Cryptology*, pages 139–147. Springer, 1993.

[14] C. Ellison and B. Schneier. Ten risks of PKI: What you're not being told about public key infrastructure. *Comput Secur J*, 16(1):1–7, 2000.

[15] P. Elmer-Dewitt and D. Jackson. First nation in cyberspace. *Time*, 6:62–64, 1993.

[16] N. Ferguson and B. Schneier. *Practical cryptography.* Wiley, New York, 2003.

[17] A. Fiat and J. Saia. Censorship resistant peer-to-peer content addressable networks. In *SIAM SODA*, pages 94–103. ACM, 2002.

[18] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *STOC*, pages 218–229. ACM, 1987.

[19] P. Hunter. Pakistan YouTube block exposes fundamental internet security weakness: Concern that pakistani action affected youtube access elsewhere in world. *Computer Fraud & Security*, 2008(4):10–11, 2008.

[20] I. Khalil, S. Bagchi, C. N. Rotaru, and N. B. Shroff. UnMask: Utilizing neighbor monitoring for attack mitigation in multihop wireless sensor networks. *Ad Hoc Networks*, 8(2):148–164, 2010.

[21] S. R. Khiani, C. Dethe, and V. Thakare. Comparative Analysis of Multipath Routing Techniques and Design of Secure Energy Aware Routing Algorithm for Wireless Sensor Network. *IJACR*, 3(3):374, 2013.

[22] S. King and S. Nadal. *Ppcoin: Peer-to-peer cryptocurrency with proof-of-stake.* August, 2012.

[23] J. Kleinberg. The small-world phenomenon: An algorithmic perspective. In *STOC*, pages 163–170. ACM, 2000.

[24] E. Kohno, T. Okazaki, M. Takeuchi, T. Ohta, Y. Kakuda, and M. Aida. Improvement of assurance including security for wireless sensor networks using dispersed data transmission. *J Comp Sys Sci*, 78(6):1703–1715, 2012.

[25] D. Korzun and A. Gurtov. *Structured peer-to-peer systems: fundamentals of hierarchical organization, routing, scaling, and security.* Springer, New York, NY, 2013.

[26] A. D. Kshemkalyani and M. Singhal. *Distributed computing: principles, algorithms, and systems.* Cambridge University Press, 2008.

[27] L. Lamport, R. Shostak, and M. Pease. The Byzantine generals problem. *TOPLAS*, 4(3):382–401, 1982.

[28] R. Levien. Attack-resistant trust metrics. In *Computing with Social Trust*, pages 121–132. Springer, 2009.

[29] A. Liu, Z. Zheng, C. Zhang, Z. Chen, and X. Shen. Secure and energy-efficient disjoint multipath routing for WSNs. *IEEE T Veh Technol*, 61(7):3255–3265, 2012.

[30] W. Lou and Y. Kwon. H-SPREAD: a hybrid multipath scheme for secure and reliable data collection in wireless sensor networks. *IEEE T Veh Technol*, 55(4):1320–1330, 2006.

[31] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim. A survey and comparison of peer-to-peer overlay network schemes. *IEEE Commun Surv Tut*, 7(2):72–93, 2005.

[32] D. Mazières. *Stellar Consensus Protocol: A Federated Model for Internet-level Consensus.* 2015.

[33] L. B. Mohr. *Explaining organizational behavior.* Jossey-Bass, 1982.

[34] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *bitcoin.org*, page 28, 2008.

[35] J. V. Nickerson, B. Tversky, J. E. Corter, L. Yu, and D. Mason. Thinking with networks. In *CogSci*, volume 36, 2010.

[36] K. Poulsen. Edward Snowden's e-mail provider defied FBI demands to turn over crypto keys, documents show. *WIRED*, 2013.

[37] J. Qadir, A. Ali, K.-L. A. Yau, A. Sathiaseelan, and J. Crowcroft. Exploiting the power of multiplicity: a holistic survey of network-layer multipath. *IEEE Comm Surv Tut*, 17(4):2176–2213, 2015.

[38] M. K. Reiter and S. G. Stubblebine. Resilient authentication using path independence. *IEEE T Comput*, 47(12):1351–1362, 1998.

[39] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[40] J. P. Sterbenz, D. Hutchison, E. K. Çetinkaya, A. Jabbar, J. P. Rohrer, M. Schöller, and P. Smith. Resilience and survivability in communication networks: Strategies, principles, and survey of disciplines. *Computer Networks*, 54(8):1245–1265, 2010.

[41] J. Von Neumann. Probabilistic logics and the synthesis of reliable organisms from unreliable components. *Automata studies*, 34:43–98, 1956.

[42] D. C. Walker. *Mass notification and crisis communications: Planning, preparedness, and systems.* CRC Press, 2012.

[43] A. C. Yao. Protocols for secure computations. In *SFCS*, pages 160–164. IEEE, 1982.

[44] H. Zhang, A. Goel, and R. Govindan. Using the small-world model to improve freenet performance. In *INFOCOM*, volume 3, pages 1228–1237. IEEE, 2002.

[45] P. R. Zimmermann. *The official PGP user's guide.* MIT press, 1995.

[46] S. M. Zin, N. B. Anuar, M. L. M. M. Kiah, and I. Ahmedy. Survey of secure multipath routing protocols for WSNs. *J Netw Comput Appl*, 55:123–153, 2015.