ABSTRACT

DAMON KOHLER. Improving Webs of Trust Through Predetermined Graph Structure. (Under the direction of Professor S. Purushothaman Iyer).

Parallel computing topographies and webs of trust (WoTs) share many of the same goals: minimum distance routing, an abundance of quickly determinable parallel paths, uniform structure, and fault tolerance. The structure of WoTs follows that of the social interaction between members of the WoT and are thus appropriately modeled by random graphs. However, it is the random structure of WoTs that contributes significantly to their insecurity. When using a WoT in a small, closed or secret society, such as a darknet, the random structure can be replaced with certain orderly structures, like the hypercube, which are commonly used for parallel computing networks. Imposing structure on the WoT, at its inception and throughout its lifetime, improves both security and the efficiency. To this end, I define the hypercube of trust.

Improving Webs of Trust Through Predetermined Graph Structure

by

Damon Kohler

A thesis submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Computer Science

Raleigh, NC

2006

Approved By:

Dr. Khaled Harfoush	Dr. Ting Yu

Dr. S. Purushothaman Iyer
Chair of Advisory Committee

For my dad.

Biography

Damon Kohler is from Caswell Beach, North Carolina. He received his Bachelor of Science degree in Computer Science from North Carolina State University and is pursuing an offer of employment as a software engineer for Google.

Acknowledgements

I would like to thank my friends and family for helping me along the way towards completing this work. I would especially like to thank Professor Iyer for his guidance and patience as I waded through many thesis topics before choosing this one, Kevin Damm for his tireless encouragement and help with developing these ideas, and Laura for her support and invaluable proofreading talents.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Kaufman, Perlman, and Speciner define public key infrastructure (PKI) as "consist[ing] of the components necessary to securely distribute public keys" [2]. There are many different models for PKI. However, PKI is generally associated with the *oligarchy model* or the use of centralized certificate authorities (CAs). In this scenario, users trust a single third party to authenticate each new user's identity and sign his certificate. In this way, the CA serves as a *trust anchor*, or a source that is well known and trusted. There are several pitfalls associated with centralized PKI, most notable of which is a single point of failure.

If a trust anchor is compromised, the entire system is compromised, since any certificate can be easily forged or manipulated. In addition, the CA has a significant amount of control over its users. Users are dependent on their CA to provide up-to-date signed certificates in order to prove their own certificate authenticity to other users. Because certificates expire over time, or may be revoked, users pay the CA for its services periodically. Typically this arrangement is not an acceptable solution for a small group of users who

simply wish to communicate securely over the Internet due it being either too costly or inflexible.

Because of their inherent trust in their organization, user groups may decide to run their own CA. By using their own CA, it is possible to eliminate costly dealings with commercial, third-party CAs. However, difficulties occur when two organizations that do not necessarily trust each other wish to communicate securely on some limited access basis. In this case, no single CA would be trusted by all parties involved, thus necessitating the use of one of many possible extensions to the oligarchy model that allow for authentication across multiple CAs or domains. Another solution, however, is to abandon the centralized PKI concept and establish a distributed PKI.

1.1 Distributed Public Key Infrastructures

Distributed PKIs are an instance of the *anarchy model* and are referred to as *webs of trust* (WoT). Whereas CAs are responsible for verifying the identity of their users and for controlling who receives signed certificates, WoTs typically employ a certificate vetting scheme that is controlled by individual users. Existing members of the WoT, known as *introducers*, create *binding certificates* for member candidates by signing the candidates' certificates. A binding certificate indicates the introducer's trust in the validity of the candidate's certificate and thus links the candidate into the WoT.

Members of the WoT calculate trust using a *trust metric*. For example, consider two WoT members, Alice and Bob. Alice tries to determine if she trusts the authenticity of Bob's certificate. If Bob's certificate is signed by introducers who Alice trusts and who

she is familiar with (*trusted introducers*) then her trust in Bob's certificate is greater than if Bob's certificate were signed by introducers with whom she is not familiar (*partially trusted introducers*). In much the same way, Alice may maintain a *web of distrust* where she places the certificates she explicitly distrusts. Thresholds on the number of trusted and partially trusted introducers required to convince Alice that Bob's certificate is valid can be used to automate trust calculations. By adjusting these thresholds, Alice's web of trust may be customized and may be as restrictive or as open as she desires.

The introducer system for extending a WoT results in organic growth. The structure of the WoT matches the structure of the social connections among its members. A common method of building WoTs is a social function known as a *key party*. At a key party, members are able to physically interact with one another, exchange certificates in person, and thus verifying the certificate/identity pairing. While this method of building WoTs is natural and simple, it results in a WoT most accurately modeled by a random graph.

To enable the discussion of the use and effects of structure, policies, and algorithms in relation to certificate use and validation, WoTs are typically represented by directed graphs. Nodes in the graph represent certificate/identity pairs. Edges represent binding certificates. A collection of binding certificates along a multi-node path is called a *certificate chain*. The method of evaluating certificate chains from a source node to a target node, in order to determine the validity or trustworthiness of the target node according to the source node, is the trust metric.

The use of trust metrics and WoTs is becoming increasingly popular as peer-to-peer (P2P) applications become more prevalent. One such P2P application type that

depends heavily on a robust WoT is a darknet.

1.2 Darknets

In 2002, several Microsoft employees coined the term "darknet" at the ACM Workshop on Digital Rights Management. They defined a darknet as "a collection of networks and technologies used to share digital content" [1]. Since then, the explosion of peer-to-peer (P2P) networks has narrowed the definition of darknets.

Modern darknets consist of a small group of trusted users, typically less than fifty, connected over an encrypted channel. Since all the users implicitly trust each other, information can pass freely between nodes without end-to-end encryption.

Darknets require certificate authentication in order to maintain the security of the network. Existing darknets fail to manage this effectively. In the absence of a centralized certificate authority, public key encryption is used with little concern for certificate authenticity, and as a result, darknets are currently rarely used in industry or academia. Instead, darknets are primarily used by security conscious file traders who are unaware of the need to securely authenticate certificate/identity pairs.

I believe that as the cost of security increases and as users become more aware of malicious users inhabiting the Internet, darknets will become increasingly popular and thus their security will become increasingly tested. Their flexibility and small, mobile nature are what make them appealing. It is important for the security concepts involved in their construction to be solid before more naive users begin to put their faith in the currently limited security that darknets offer.

1.3 Contributions

Existing research into increasing the efficiency and robustness of parallel computing through improved network topographies is directly applicable in many ways to the computations involved with the evaluation of trust in a distributed PKI (see Table 1.1). This work explores this previously unrecognized links between parallel computer topographies, their highly developed routing and fail-over algorithms, and webs of trust in two general contexts:

- *Improving the robustness of WoTs through structure.* Existing WoTs have many deficiencies that significantly degrade their security. Many of these weaknesses can be overcome by creating and maintaining structured WoTs with parallel computing topographies.

- *Improving the efficiency of WoTs through structure.* Existing methods use heuristics to solve the NP-hard problems involved in the use of WoTs. These algorithms typically run in polynomial time. The algorithms developed in this work, which are designed for structured WoTs, run in linear or sub-linear time and are exact.

The conjecture which precludes this solution is that the extra costs involved in constructing and maintaining a structured WoT, as opposed to an unstructured WoT, are acceptable given the increased security and increased efficiency of the WoT. This is the case with darknets, where the number of users is relatively small.

Throughout this thesis, the hypercube is addressed as an example of a well known parallel computing network topology. I define a hypercube of trust (HoT) as a sample

Table 1.1: A summary of corresponding motivations between parallel computing and webs of trust.

Parallel Computing	Webs of Trust
Fault tolerance	Attack-resistance
Efficient parallel routing	Certificate chain verification
Minimum distance routing	Cryptographic overhead
Uniform structure	Avoiding centralized nodes

application of the discoveries described in this thesis.

Chapter 2

Preliminaries

The following material is a survey of the theoretical building blocks used throughout the remainder of this thesis. These definitions, propositions, and theories will be referred to later rather than being restated.

2.1 Graph Theory

Definition 1 ([12]). A graph G is a triple consisting of a vertex set $V(G)$, an edge set $E(G)$, and a relation that associates with each edge two vertices (not necessarily distinct) called endpoints.

Definition 2 ([12]). A directed graph or digraph G is a triple consisting of a vertex set $V(G)$, an edge set $E(G)$, and a function assigning each edge an ordered pair of vertices. The first vertex of the ordered pair is the tail of the edge, and the second is the head; together, they are the endpoints. I say that an edges is an edge from its tail to its head.

Definition 3. A path or walk is a sequence of vertices $v_1, v_2, \ldots v_n$ such that for v_i where

$1 \leq i \leq n$ then $(v_i, v_{i+1}) \in E$. A cycle is a path $v_1 \dots v_n$ such that $v_1 = v_n$. A trail is a path with no repeated edge. A walk or trail is closed if its endpoints are the same and is thus a cycle. The length of a walk, trail, path, or cycle is the number of edges it contains.

Definition 4 ([12]). A clique in a graph is a set of pairwise adjacent vertices.

Definition 5 ([12]). A graph G is bipartite if $V(G)$ is the union of two disjoint (possibly empty) independent sets called partite sets of G.

Definition 6 ([12]). The chromatic number of a graph G, written $\chi(G)$, is the minimum number of colors needed to label the vertices so that adjacent vertices receive different colors. A graph G is k-partite if $V(G)$ can be expressed as the union of k (possibly empty) independent sets.

Definition 7 ([12]). Let v be a vertex in a digraph. The outdegree $d^+(v)$ is the number of edges with tail v. The indegree $d^-(v)$ is the number of edges with head v. The out-neighborhood or successor set $N^+(v)$ is $\{x \in V(G) \mid (v, x) \in E(G)\}$. The in-neighborhood or predecessor set $N^-(v)$ is $\{x \in V(G) \mid (x, v) \in E(G)\}$. The minimum and maximum indegree are $\omega^-(G)$ and $\Delta^-(G)$; for outdegree we use $\omega^+(G)$ and $\Delta^+(G)$.

Definition 8 ([12]). If G has a u, v-path, then the distance from u to v, written $d_G(u, v)$ or simply $d(u, v)$, is the least length of a u, v-path. If G has no such path, then $d(u, v) = \infty$. The diameter (diam G) is $\max_{u,v \epsilon V(G)} d(u, v)$.

The eccentricity of a vertex u, written $\epsilon(u)$, is $\max_{v \epsilon V(G)} d(u, v)$. The radius of a graph G, written rad G, is $\min_{u \epsilon V(G)} \epsilon(u)$.

Definition 9 ([12]). A network is a digraph with a non-negative capacity $c(e)$ on each edge e and a distinguished source vertex s and a sink vertex t. Vertices are also called nodes.

A flow f assigns a value $f(e)$ to each edge e. We write $f^+(v)$ for the total flow on edges leaving v and $f^-(v)$ for the total flow on edges entering v. A flow is feasible if it satisfies the capacity constraints $0 < f(e) < c(e)$ for each edge and the conservation constraints $f^+(v) = f^-(v)$ for each node $v \notin \{s, t\}$.

Definition 10 ([12]). The value of a flow f is the net flow $f^-(t) - f^+(t)$ into the sink. A maximum flow is a feasible flow of maximum value.

2.2 Certificate Chains

Definition 11 ([8]). Certificates are represented as triples $\langle x/j, k, s_{k'} \rangle$, where x is an identity, j is a key index number, k is a public key, and $s_{k'}$ is the digital signature over the combination of x/j and k.

A *user* of the system is defined by an *identity*, and each user of the system is assumed to have exactly one *true identity*. Any other identity which does not belong to exactly one user is a *false identity*. In real world situations, it may be the case that a user has more than one valid public key. Each of these public keys are represented by a key index number. However in this thesis, users are assumed to have a single valid public key and so the key index number is always 0 and can be ignored [5].

Definition 12 ([5]). Given a certificate $C = \langle x/j, k, s_{k'} \rangle$, if (i) the identity x is a true identity, and (ii) the user with identity x says k is his or her public key, then C is a true certificate and k is a true public key for x. Otherwise, C is a false certificate and k is a false public key for x.

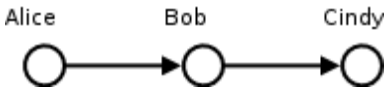

Figure 2.1: A certificate chain from Alice to Cindy. Alice has signed Bob's certificate and Bob has signed Cindy's certificate.

Definition 13 ([5]). If $s_{k'}$ in $\langle x/j, k, s_{k'} \rangle$ is generated by y, I say the certificate is issued by y and $s_{k'} = s_y$.

Definition 14 ([5]). If all certificates issued by y are true certificates, then y is a good user.

Definition 15 ([5]). If there exists at least one false certificate issued by y, then y is a malicious user.

Definition 16. If a certificate $\langle x/j, k, s_a \rangle$ is issued by a and a certificate $\langle x/j, k', s_b \rangle$ is issued by b then if $k = k'$ the two certificates are in agreement. If not, then there is a conflict between these two certificates.

Definition 17 ([4]). A certificate chain is a sequence of certificates where the public key of one certificate can be used to verify the digital signature associated with the previous certificate of the chain. See figure 2.2.

Definition 18 ([10]). Given a directed graph G, distinguished nodes s and t, and a path bound b, the *bound disjoint paths* (BDPs) between s and t are the paths contained in the maximum set of mutually disjoint b-bounded paths from s to t.

In this thesis, vertex disjoint paths are assumed to also be identity disjoint. Thus, all BDPs are identity disjoint.

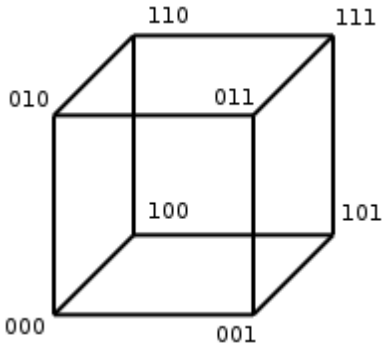

Figure 2.2: 3-D view of the 3-cube.

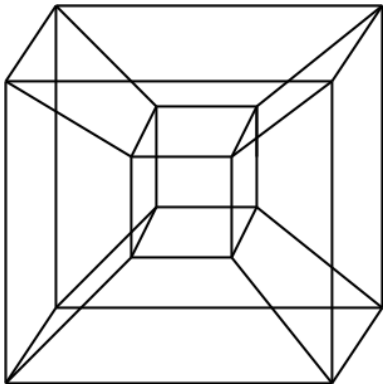

Figure 2.3: 3-D view of the 4-cube.

2.3 Hypercubes

Definition 19 ([11]). An n-cube graph, or H_n, is an undirected graph consisting of $k = 2^n$ vertices labeled from 0 to $2^n - 1$ and such that there is an edge between any two vertices if and only if the binary representations of their labels differ by one and only one bit. See figures 2.2 and 2.3.

Proposition 1 ([3]). The n-cube is a connected graph of diameter n.

Proposition 2 ([11]). The minimum distance between the nodes A and B is equal to the

number of bits that differ between A and B, i.e., to the Hamming distance $H(A, B)$.

Proposition 3 ([11]). There are n different ways of tearing an n-cube, i.e., of splitting it into two $(n-1)$-subcubes so that their respective vertices are connected in a one-to-one way. Given the labeling defined in Definition 19, each different tearing corresponds to splitting the n-cube graph into two sub-graphs: one whose node labels have a one in the position i and one whose node labels have a zero in position i.

Proposition 4 ([11]). Any two adjacent nodes A and B of an n-cube are such that the nodes adjacent to A and those adjacent to B are connected in a one-to-one fashion.

Proposition 5 ([11]). Let A, B be any two nodes of an n-cube and assume that $H(A, B) < n$. Then there are n parallel paths between A and B. Moreover, the length of each path is at most $H(A, B) + 2$.

Definition 20. Faulty nodes are nodes in the network which are not capable of communication with the rest of the network. For example, a shutdown computer would be considered a faulty node.

Chapter 3

Summary of Literature

The trust metrics devised in previous works evaluate trust as either a continuous value or as a Boolean value. The primary goal of these metrics is typically to maximize the number of certificates that can be proved to be true. Secondly, it is also useful for the trust metric to be resilient to attacks.

3.1 Continuous Trust Metrics

In continuous trust metrics, trust between nodes in a graph is calculated as a real value between zero and one. One such instance of this technique is described by Levien [6]. Levien's method uses a weighted digraph and maxflow to create an attack-resistant trust metric. He assumes that most nodes are trusted and that there is a constant indegree n for each node in the graph. The result is a trust metric that is significantly more resistant to attack than metrics like shortest path and Maurer [7]. In fact, given his assumptions, his maxflow trust metric is optimal.

3.2 Boolean Trust Metrics

Continuous trust metrics can be converted into Boolean trust metrics by simply setting a threshold value, σ, for trust evaluation. This is frequently done to automate the evaluation of trust metrics. Given a trust value t_k for key k, if $t_k < \sigma$ then k is considered false otherwise it is considered true. However, related work by Reiter and Stubblebine and Jiang [10, 5], propose methods that are strictly Boolean. Rather than assuming a constant indegree, these methods assume that a limited number of malicious users exist in the WoT. According to Levien, the attack resistance of Reiter and Stubblebine's method is nearly optimal [6].

3.3 Attacking Trust Metrics

Attacking a WoT is equivalent to inserting false certificates or *forging nodes*. Levien defines two types of attacks: node attacks and edge attacks [6]. In a node attack, the attacker compromises the target node and is then able to generate arbitrary binding certificates from the compromised node. For example, this happens when a node's private key, or password, is stolen. In an edge attack, the attacker is able to convince an uncompromised node to add a binding certificate to an untrustworthy node, or a forgery. In both attacks, it is desirable to choose a target node that is trusted by the greatest number of nodes in the graph, as this is an indicator of how much trust is associated with the target node.

3.4 Measuring Trust Metric Attack-Resistance

The success of an attack is measured by the fraction of nodes that accept or trust the forged node. The goal of an attack-resistant trust metric is to maximize the amount of work necessary to launch such an attack. Thus, the attack-resistance of a trust metric is measured in terms of the amount of work required, or the number of nodes which must be compromised, in order to successfully gain trust from a specified fraction of the nodes. Levien makes the following assertions and summarizes the attack resistance of published trust metrics in Table 3.1 where α is a factor indicating the amount of sharing of certification keys, generally in the range of [0.5..1] and where d is the number of certificates issued for each key in the system [6]:

- No trust metric can protect against attacks on d keys or more, where d is the minimum number of binding certificates on any widely accepted key.
- There is an optimal trust metric based on maximum network flow. Such a metric protects against almost any attack on fewer than d keys.
- Of previously published trust metrics, [10] is close to optimal while [7] is easily attacked.

3.5 Certificate Chains

In Reiter and Stubblebine's work, multiple identity independent certificate chains are used to verify the certificate of the target [10]. It is assumed that each identity/certificate pair is represented by exactly one node in the certificate graph. No identity is associated

Table 3.1: A comparison of trust metrics [6].

metric	# nodes for node attack	# nodes for edge attack	# edges for edge attack
shortest path	1	1	1
Maurer	1	1	d
Reiter & Stubblebine	d	d	d
Maxflow	d	d	d
Maxflow-edge	d	αd^2	αd^2
best case	d	αd^2	αd^2

with more than one public key. Jiang, expanding on Reiter and Stubblebine's work, defines the following two theorems regarding the use of independent certificate chains.

Theorem 1 ([5]). *Given an unlimited number of non-colluding malicious users, having* 2 *or more identity disjoint paths between k_0 and k_t proves the validity of the head certificate.*

Theorem 2 ([5]). *Given a limited number, m, of colluding malicious users, having $m+1$ identity disjoint paths between k_0 and k_t proves the validity of the head certificate.*

If certificate graphs are assumed to represent a WoT where users are only allowed to have a single public key, finding identity independent paths is the same as finding vertex independent paths. Reiter and Stubblebine refer to these paths as *bounded disjoint paths* (BDPs) [10]. BDPs are vertex disjoint paths of a length less than or equal to the bound. Finding BDPs is an NP-hard problem on random graphs. Because there is no efficient solution, approximations must be used.

3.6 Certificate Conflicts

Jiang expands on Reiter and Stubblebine's BDP based metric by analyzing certificate conflicts. He shows that the identification of conflicting certificates can contribute

to the verification of additional certificates in the WoT [5]. Since conflicts are indicative of malicious users, such users can be located and removed from the WoT. Jiang uses certificate conflicts to create *suspect sets*.

Definition 21 ([5]). A suspect set is a set of identities that contains at least one malicious user.

Jiang defines the following set of rules for determining suspect sets in a WoT containing a limited number of colluding malicious users [5].

1. Given a certificate chain whose head certificate is false, construct a new suspect set that contains all the identities (except the identity in the head certificate) in the certificates of the chain.

2. Given two certificate chains whose head certificates are conflicting with each other, if one of the head certificates is true and the other is undetermined, construct a new suspect set that contains all the identities in the certificates of the chain whose head certificate is undetermined.

3. Given two certificate chains whose head certificates are conflicting with each other, if both head certificates are undetermined, construct a new suspect set that contains all the identities in the certificates of the two chains.

These rules are designed to be applied in order and each increases in the generality. It may not be desirable to use all the rules since the size of the suspect sets may grow to be too large [5].

Jiang explains that given a maximum disjoint sets (MDS) from the generated suspect sets, the union of those a MDS, L, contains at least a malicious users [5]. Assuming there are b malicious users in the WoT, removing all certificates issued by identities contained in L from the WoT reduces the number of malicious users by a to $b-a$. By applying Theorem 2 to the resulting graph, additional certificates can be proved to be true. For each k_t, if there are $b - a + 1$ BDPs, k_t is true. However, determining the number of MDS is an NP-hard problem.

Chapter 4

Problem Statement

A major problem present in typical WoTs is that their structure is arbitrary. To counter this, Levien requires a constant indegree for each node in the graph [6]. Jiang proposes a recommendation system for adding certificates to an existing graph in order to increase its robustness [4]. The random nature of WoT structures presents difficulties that are only overcome by establishing restrictions on or making modifications to the structure.

For a given WoT it is possible to validate identity/key pairs through an analysis of multiple identity independent paths [10]. It is also possible to increase the number of identity/key pairs that can be validated using this method by analyzing certificate conflicts within the WoT [5]. This work intends to improve the efficiency of these methods and make additional improvements to the robustness of WoTs. This is done under the following assumptions:

Assumption 1. There is a maximum of m malicious users in the system.

This is the same assumption used in both Jiang's work and in Reiter and Stub-

blebine's work [5, 10]. When using structured graphs, it is easy to determine the maximum m malicious users that could potentially be detected. The assumption is important because, if the maximum is exceeded, the trust metric is no longer guaranteed to be accurate.

Assumption 2. There may exist malicious users who are in collusion.

Uncoordinated attacks on any WoT have very little chance of succeeding. It would be naive to assume that malicious users are incapable of collusion. This assumption increases the number of BDPs required for node authentication.

Assumption 3. Each identity has exactly one public key.

While this may be an unrealistic assumption in the case of very large WoTs, for darknets, which are much smaller by comparison, it is a reasonable assumption that non-malicious users will maintain a single public key.

Existing methods for performing trust metric calculations depend on NP-hard algorithms, such as determining maximum independent sets, and focus on the use of existing, large-scale WoTs. Thus, random graphs are studied as an approximation of a worst-case WoT. Even with optimal trust metrics, unstructured WoTs are subject to attacks where a disproportionately small amount of work can result in a significant success rate. Because of this, some constraints, such as constant indegree and unique identities for each vertex, are suggested in previous works to improve robustness [6, 10].

By imposing specific structures on a WoT as it is constructed, and throughout its lifetime, it is possible to improve the efficiency and the effectiveness of existing algorithms. In addition, this constraint also improves the robustness of the WoT by limiting malicious collusion and by limiting the spread of forgeries.

The structures of interest in this work are those that are commonly associated with parallel computing topographies. The following desirable qualities of network topologies found in parallel computing also make good WoT topologies:

- Minimum distance routing.
- An abundance of quickly determinable parallel paths.
- Uniform structure.
- Fault tolerance.

The following problems are solved by using existing, well-studied parallel computing topologies to construct WoTs:

Problem 1. Collusion between malicious users reduces the number of certificates that can be validated. Without collusion, validation requires only 2 certificate chains. However, with m malicious users in collusion, $m + 1$ certificate chains are required to validate the target certificate. How can collusion between malicious users be limited while maintaining the distributed nature of the WoT?

Problem 2. Malicious users are revealed by analyzing certificate conflicts. How can conflicting certificates be found efficiently in the WoT?

Problem 3. Finding identity disjoint certificate chains, or BDPs, in a random graph is an NP-hard problem. How can these certificate chains be found efficiently?

The HoT is introduced and used as an example of a structured WoT. Constructing and maintaining a HoT is a new problem and is also addressed in this work.

Problem 4. Hypercubes are known for their fault tolerance. Their hierarchical structure also makes it possible to monitor the connectivity of the graph in a distributed fashion. Given a probability that nodes are malicious, what is the probability that the hypercube will remain connected?

Problem 5. Since hypercubes always have 2^d nodes and there may not always be 2^d users, there are some complications in the creation and maintenance of a HoT. How can changes to the structure of a HoT be managed while maintaining the distributed nature of the HoT?

By solving these problems, darknets, which typically consist of less than approximately fifty nodes, can be made more robust, more resilient to attack, and more efficient with regards to performing trust metric calculations. Combined, these improvements make darknets a more viable method of secure, small-group interaction across untrusted mediums.

Chapter 5

Malicious Collusion

Given the vernacular definition of "malicious," it is important to note that malicious users may not necessarily intend to be malicious. Such users could be the victims of social engineering or other types of attack on the network. However, for the purposes of this thesis, such a user is a threat to the overall security of the WoT and thus considered to be malicious.

5.1 Random Graphs

The reason that WoTs are typically modeled by random graphs is that random graphs are considered the worst-case WoT. In addition, existing, large-scale WoTs are quite random and thus represent the worse-case scenario. The structure of WoTs takes the shape of the social network of users in the system. In order to make the WoT more robust, users try to verify and be verified by as many other users as possible. While this would eventually be effective, for example by achieving a fully connected graph, it is costly (it takes a lot of

key parties) and inefficient.

5.2 Centralized Nodes

The high number of binding certificates for centralized nodes makes them trusted by a disproportionately large number of users in the WoT. As such, the certificates generated by centralized nodes influence an equally disproportionately large portion of the WoT that now suffers from the same single point-of-failure problem found in centralized PKIs. Depending on the trust metric, compromising a single centralized node can lead to a catastrophic failure of the system.

This is not the case for Reiter and Stubblebine's or Jiang's trust metrics, which require $m + 1$ parallel certificate chains for verification. However, if a malicious takeover is successful, that is $m + 1$ or more certificate paths are under malicious control, and the malicious control stems from $m + 1$ compromised centralized nodes, then such a major system failure can occur.

Theorem 3. *Let $p > 0.5$ be the fraction of the WoT which accepts a centralized nodes as being valid. Given a WoT with c centralized nodes, the fraction of nodes, i, which are adjacent to all centralized nodes is in the range $cp - (c - 1) \leq i \leq p$.*

For example, in a WoT with 2 centralized nodes, each connected to a fraction $p \geq 0.8$ of the WoT, compromising those nodes results in compromising a minimum fraction $i \geq cp - (c - 1) = 2 * 0.8 - (2 - 1) = 0.6$ of the WoT. The maximum fraction would be 0.8.

Proof.

(i) If a fraction p of nodes are adjacent to a centralized node c_0, then $1-p$ of the nodes are not adjacent to c_0. According to the pigeon hole principle, any centralized node c_1, also adjacent to a fraction p of nodes, can avoid being adjacent to a maximum of $n*(1-p)$ of the nodes already adjacent to c_0 and must be adjacent to at least $p-(1-p)=2p-1$ of the nodes already adjacent to c_0. In essence, the minimum shared nodes between c_0 and c_1 is the minimum fraction of adjacent nodes for each minus the fraction of remaining nodes.

(ii)

$$2p-1=p-(1-p) \text{ for } n=2 \tag{5.1}$$

$$np-(n-1)=p-(1-(p-(1-p))) \text{ for } n=3 \tag{5.2}$$

$$p-(1-(np-(n-1)))=(n+1)p-((n+1)-1) \text{ for } n=n+1 \tag{5.3}$$

$$p-1+(np-(n-1))= \tag{5.4}$$

$$p-1+np-n+1= \tag{5.5}$$

$$p+np-n= \tag{5.6}$$

$$(n+1)p-n= \tag{5.7}$$

$$\tag{5.8}$$

□

5.3 Structured Graphs

If, instead of a random structure, the WoT's structure is orderly, predetermined, and agreed upon by the users in the system, robustness can be achieved quickly and ef-

ficiently. In addition, imposing a structure on the WoT helps to limit collusion between potentially malicious users on two fronts:

- Structure ensures a uniform distribution of nodes.
- Structure provides a social restriction.

As discussed previously, centralized nodes are a significant detriment to security. A uniform arrangement of nodes ensures a constant indegree and thus prevents centralized nodes.

In addition, requiring users to certify and be certified by specific other users, as defined by the WoT's structure, instead of allowing users to certify and acquire certification at their own discretion, limits the possibility of collusion. It would be possible to mount a coordinated attack with careful planning and monitoring of the WoT so that colluding malicious users could join at the right time to maximize the number of their shared neighbors. However, this requires significantly more work than carrying out an attack on a typical WoT. The social aspects of key exchange in this new restrictive manner are outside the scope of this paper.

5.4 Hypercubes

Hypercubes have no centralized nodes. The graph has a constant indegree for each node and no cliques larger than 2 nodes. This uniform distribution minimizes the effect of a malicious user at any particular node and forces the attack resistance to be linear.

Theorem 4. *The maximum number of colluding malicious users that can be detected in a d-HoT is* $r = \lfloor \frac{d-1}{2} \rfloor$.

Proof. Theorem 2 states that given m malicious users, $m+1$ identity independent certificate chains are required to verify the target certificate.

Proposition 5 states that given a complete d-cube, there are d BDPs between any two nodes.

In order for there to be more certificate chains than there are malicious users, more than one half the number of BDPs between any two nodes must contain only valid users. Thus, the maximum number of paths that can contain malicious users is one less than half the total number of BDPs.

$$\frac{d-1}{2} < \frac{d}{2} \tag{5.9}$$

$$\lfloor \frac{d-1}{2} \rfloor \leq \frac{d-1}{2} \tag{5.10}$$

$$2 * (\lfloor \frac{d-1}{2} \rfloor) + 1 = d \tag{5.11}$$

Therefore, the maximum number of malicious users that can be detected in a hypercube is $\lfloor \frac{d-1}{2} \rfloor$. □

The robustness of a HoT using conflict detection grows linearly with the dimension of the hypercube, d, while the number of users, n, grows exponentially as $n = 2^d$. Because of this, it is important that malicious collusion be limited by other means. While it is possible, with a random graph, to simply add more edges and increase the minimum indegree of the graph vertices, it is not possible to do this with a HoT. Despite this, there are inherent

properties of hypercubes that help to limit malicious collusion.

Theorem 5. *In a d-HoT, forging a single node requires compromising* $d - \lfloor \frac{d-1}{2} \rfloor$ *neighbor nodes of the target. Forging additional nodes requires at least* $d - \lfloor \frac{d-1}{2} \rfloor - 2$ *nodes for each new forged node.*

Proof.

(i) First, the maximum number of shared neighboring nodes between any two nodes in a d-cube is 2. All nodes in a d-cube can be labeled using a binary string $b_1 b_2 \dots b_d$. Nodes are adjacent, i.e. they are neighbors, if their binary addresses differ by a single bit. Two nodes share two neighbors if and only if their binary addresses differ by exactly 2 bits. If their addresses differ by more than two bits, the nodes share no neighbors since all single bit modifications of their addresses will always differ by at least 2 bits.

(ii) Second, $m > 2$ nodes share a maximum of 1 neighbor nodes. m nodes share a neighbor only if their binary addresses each differ from the shared neighbor node by a single bit. Any 2 nodes from the group may share a second neighbor since their addresses differ by exactly 2 bits. However, the addresses of any group of nodes greater than 2 will differ by more than 2 bits.

Given a successfully attacked node, the maximum number of malicious nodes already surrounding a second potential target is 2. An additional $d - \lfloor \frac{d-1}{2} \rfloor - 2$ nodes are required to forge the second node.

□

Table 5.1: The number of forged nodes versus the number of compromised nodes in a d-HoT.

d	# forged nodes	# compromised nodes
5	15	17
6	20	43
7	42	86
8	63	192
9	127	384
10	204	819
11	409	1638
12	682	3413
13	1365	6827
14	2340	14043
15	4680	28087

The number of forged nodes in a HoT being subjected to such an attack would be a small fraction of the number of nodes which have been compromised. If all nodes in the HoT were either forged or compromised, the number of forged nodes would be $f = \frac{2^d-2}{d-\lfloor\frac{d-1}{2}\rfloor-1}$ and the number of compromised nodes would be $c = 2^d - f$. This relation is depicted in Table 5.1. In essence, the real security deficiency with such a HoT is not protecting the nodes themselves from being compromised.

Chapter 6

Finding Certificate Chains

Finding parallel certificate chains, or BDPs, on a random graph is an NP-hard problem. However, for graphs with specific structures where the path information is encoded in the graph structure itself, BDPs can be found much faster, often in sub-linear time. Parallel computing network topographies are typically designed with this in mind. Table 6.1 provides a comparison.

In addition to parallel routing, the minimum distance routing properties of parallel computing topographies also reduce the amount of work necessary to evaluate a certificate chain. Reducing the number of certificates in the chain results in fewer decryption tasks during the end-point verification stage of communication. Levien's trust metric does not make any guarantees about certificate chain length nor does he address the problem [6]. Reiter and Stubblebine, however, do address the issue and their BDP algorithm allows for the length of the disjoint paths to be arbitrarily bounded.

Table 6.1: A comparison of trust metric algorithms and graph topographies for a graph $G(V, E)$.

Topology	Time	Max Distance
Random (Levien)	$V * E^2$	Undefined
Random (Reiter & Stubblebine)	$V * E$	Arbitrary bound
d-HoT	$\log^2 V$	$d + 2$

6.1 Hypercubes

In a hypercube, BDPs are easily calculated through address manipulation. There are exactly d BDPs between any two nodes, each of length no greater than $d + 2$, in a hypercube [11]. The minimum path length between any two nodes in a hypercube is equal to the Hamming distance between those two nodes, $H(k_0, k_t)$. Algorithm 1 is a proposed algorithm for finding all BDPs in a complete d-cube between k_0 and k_t. For simplicity, the algorithm assumes that the differing bits between the binary addresses of k_0 and k_t make up the first $H(k_0, k_t)$ bits of the addresses. For a complete d-cube with $n = 2^d$ nodes, there are d paths between k_0 and k_t. Thus, the running time for Algorithm 1 is $O(\log^2 n)$.

Algorithm 1 Find all BDPs in a complete d-cube between k_0 and k_t.

Require: $i \leftarrow H(k_0, k_t)$ {Hamming distance between k_0 and k_t}
Ensure: p' is the set of all BDPs between k_0 and k_t
  $j \leftarrow 1$ {Index of current differing bit}
  while $j \leq i$ **do** {Find all paths of length i}
    $p \leftarrow \{k_0\}$ {The current BDP being built}
    $n \leftarrow j$
    while $n \leq i$ **do**
      $q \leftarrow$ correct bit n of peek(p) according to k_t
      Push q on to p
      $n \leftarrow n + 1$
    end while
    $n \leftarrow 1$
    while $n < j$ **do**
      $q \leftarrow$ correct bit n of peek(p) according to k_t
      Push q on to p
      $n \leftarrow n + 1$
    end while
    Push p on to p'
    $j \leftarrow j + 1$
  end while
  if $i < d$ **then**
    for $j = i + 1$ to $j = d$ **do** {Find remaining paths of length $i + 2$}
      $r \leftarrow p'[j - i]$
      $p \leftarrow \{k_0\}$
      $q \leftarrow$ flip bit j of peek(p)
      Push q on to p
      for $n = 2$ to $n = i + 1$ **do** {For each intermediate node in path r}
        $q \leftarrow r[n]_{1\ldots i}\text{peek}(p)_{i+1\ldots d}$ {Replace the first i bits of peek(p) with the first i bits of $r[n]$.}
        Push q on to p
      end for
      $q \leftarrow$ flip bit j of peek(p)
      Push q on to p
      Push p on to p'
    end for
  end if
  return p'

Chapter 7

Detecting Certificate Conflicts

Algorithms for finding conflicting certificates are not addressed in Jiang's work. Here I address the issue and show that it is more efficient using structured graphs because the identification of certificate conflicts can be accomplished in a distributed fashion. Finding conflicting certificates enables the removal of malicious users and decreases the number of valid certificate chains required to validate a target certificate.

To find conflicts in a random graph requires a full search of the graph space. Given a random graph with a minimum indegree i, the time required to perform a complete breadth first search is $O(i * V^2)$ in the best-case and $O((V-1) * V^2)$ in the worst-case, i.e. for a complete graph. However, given a hierarchical structure, as typically found in parallel computing networks, it is trivial to establish distributed monitoring for conflicts and act on them as they occur. For example, the following theorem applies to HoTs:

Theorem 6. *Given a certificate conflict c in a k-sub-HoT, the d-HoT will contain the same conflict.*

Proof. Given a k-sub-HoT $H_k(V_k, E_k)$ of a HoT $H(V, E)$, by definition $V_k \subset V$ and $E_k \subset E$. If there are conflicting certificates in H_k, it is not possible to remove those conflicts by adding additional certificates.

By adding dimensions to the k-sub-HoT, certificates are added to V_k, E_k that approach V, E. No certificates are removed in the process of expanding the k-sub-HoT.

Therefore there exists conflicts in H. □

Since local k-sub-HoT conflicts imply global d-HoT conflicts, distributed monitoring for conflicts is possible. If a node detects a local conflict, it can then quickly investigate all the neighbors of the node exhibiting the conflicts and potentially identify a malicious user. By sharing this information with the rest of the HoT, the malicious member can be removed, quickly, efficiently, and automatically.

7.1 Using Certificate Conflicts

Certificates fall into one of three categories: true, false, or unknown. If a certificate is proved to be true, then any certificates that conflict with it are false. When there is not enough information to prove that a certificate is true, then its status is unknown. Thus, rather than using Jiang's rules for constructing suspect sets, it is possible to determine a more concise set of only malicious users using a simple rule:

Theorem 7. *Given a set of certificates for key x/j with indegree d*

$$C_{x/j} = \{\langle x/j, k, s_{k'_0}\rangle, \langle x/j, k, s_{k'_1}\rangle, \ldots, \langle x/j, k, s_{k'_d}\rangle\} \tag{7.1}$$

and a WoT with m malicious users, at least $m+1$ BDPs with head certificates in $C_{x/j}$

must be in agreement for x/j to be true. Any remaining certificates in $C_{x/j}$ which are not in agreement are false. If there are less than $m+1$ certificates in agreement, then x/j is unknown.

This rule directly follows Definition 15 of a malicious user. With this rule it is possible to create a set of purely malicious users known as the *malicious set*.

Proof. According to Theorem 2, given m malicious users, $m+1$ BDPs that do not contain malicious users are required to verify the target certificate. Thus, if the target is valid, $v \geq m+1$ certificates in $C_{x/j}$ must be valid and the remaining certificates must be invalid. If there are less than $m+1$ certificates in agreement, then it is not possible to determine the validity of the target and thus the validity of all the certificates in $C_{x/j}$ and the validity of the target is unknown. □

For example, according to Theorem 4, the maximum number of colluding malicious users which can be detected in a d-HoT is $\lfloor \frac{d-1}{2} \rfloor$. Thus, the number of valid certificates required to verify the target is $d - \lfloor \frac{d-1}{2} \rfloor$. If there exists $c \geq d - \lfloor \frac{d-1}{2} \rfloor$ certificates to the target that are in agreement, the target is considered valid and the c certificates that are in agreement are considered valid as well. Since only c paths are valid, then the $d-c$ remaining certificates must be invalid. If there do not exist $c \geq d - \lfloor \frac{d-1}{2} \rfloor$ certificates to the target that are in agreement, then it is not possible to determine the validity of the target. Thus, the validity of all the certificates in question, including the target, is unknown.

7.2 Using The Malicious Set

As malicious users are identified and the malicious set, M, is constructed, the malicious users are removed from the WoT. Assuming there are b malicious users in the system, the number of malicious users in the system is reduced by $|M|$ and thus the number of parallel certificate chains required to prove a certificate is true has been reduced to $b - |M| + 1$. As a result, it is possible to validate more certificates and additionally increase the size of the malicious set as new conflicts are detected. This is the same method used by Jiang to increase the number of verifiable certificates using suspect sets [5].

Chapter 8

Hypercubes of Trust

In this chapter, I further discuss the HoT as an application of the ideas presented above. I begin by discussing methods for predicting connectivity using the concept of strong fault-tolerance in hypercubes and then finish with method definitions for the construction and maintenance of HoTs.

8.1 Strong Fault-Tolerance

From the definition of strong fault-tolerance, which the hypercube fits, $d-2$ nodes may be faulty and there will still be $\min(\deg u, \deg v)$ BDPs between any two non-faulty nodes u and v [9]. Using this information, it is possible to determine the probability of the HoT maintaining a state of *sufficient connectedness*.

Definition 22. A HoT is *sufficiently connected* when all non-malicious nodes have enough BDPs to all other non-malicious nodes to enable their validation.

Theorem 8. *The probability of the d-HoT with $n = 2^d$ nodes being sufficiently connected*

Table 8.1: Probability of sufficient connectedness for d-HoTs given probability p of members being malicious.

p	0.10	0.05	0.01
5-HoT	0.367	0.786	0.996
6-HoT	0.039	0.373	0.973
7-HoT	0.001	0.113	0.960
8-HoT	0.000	0.001	0.745

given a constant p probability of nodes being malicious is $C(p) = \sum_{i=0}^{d-2} \binom{n}{i}(p^i)(1-p)^{n-i}$.

This is the case when the number of malicious users, m, is in the detectable range, $m < \lfloor \frac{d-1}{2} \rfloor$. In this case, $m < d-2$ for $d > 3$ and so there are $\min(\deg u, \deg v) \geq d - \lfloor \frac{d-1}{2} \rfloor$ BDPs between any two members u, v in the HoT. Thus the d-HoT is sufficiently connected.

Proof. Suppose a d-HoT with $d > 3$ has $m \leq \lfloor \frac{d-1}{2} \rfloor$ malicious users and has p be the probability of a node being malicious.

(i) First, I address $\min(\deg u, \deg v) \geq d - \lfloor \frac{d-1}{2} \rfloor$ for all node pairs u, v. The degree of every vertex in a hypercube is d. Since m is the maximum number of malicious nodes, then there will never be a node with more than m malicious neighbors. Thus the degree of each node is at least $d - m$.

(ii) Finally, I address the polynomial $C(p)$. The probability of the nodes being sufficiently connected is equal to the probability of having $m \leq \lfloor \frac{d-1}{2} \rfloor$ given p. Thus it is the probability that for each possible arrangement of $m \leq \lfloor \frac{d-1}{2} \rfloor$ nodes, $\binom{n}{m}$, $n-m$ nodes are not malicious, $(1-p)^{n-m}$ and m nodes are malicious, p^m.

□

8.2 Adding and Removing Members

One of the convenient properties of hypercubes is that their construction is relatively simple. A hypercube is most easily constructed through by creating a copy of an existing hypercube and then linking associated vertices. This process is known as *doubling*. Hypercubes always have 2^d vertices. Obviously, however, there may not always be 2^d users. This presents some problems in an environment where doubling the number of nodes may not always be desired, like when adding a single new user.

8.2.1 Adding Members

Since the number of nodes will almost always exceed the number of members, it is likely that there will not be enough existing neighbor nodes to fully introduce a new member. To remedy this, a member may need to represent more than one address at a time. However, in an effort to maintain proper structure of the HoT, each member should not represent more than one additional address at a time.

Method 1. Given a node in a d-HoT with address $b_0 b_1 \dots b_d$, the node should act as a surrogate for the node at address $b_0 b_1 \dots b_d 1$ until such time as that address is to occupied by a new member.

For new user Alice to join a HoT, she needs to create a public/private key pair and either be the first person in the HoT or be invited by another user, Bob, already in the HoT. The current dimension of the HoT, d, will dictate the number of introducers, also d, which Alice will require in order to join. It is Bob's responsibility to organize these introducers and to place them in contact with Alice or visa-versa. Since Alice's introduction

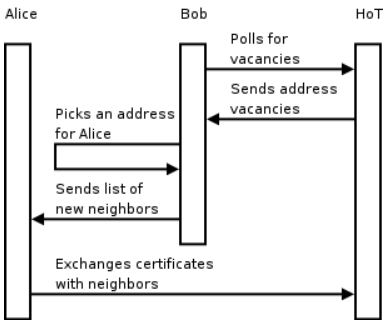

Figure 8.1: Bob invites Alice to join the HoT.

into the HoT is contingent on the acceptance of d members, Bob is not able to introduce an arbitrarily large number of users himself. The rest of the HoT can regulate him as they wish.

Method 2. Given a new member, Alice, to be introduced, HoT members are polled for available addresses. Alice's address is chosen arbitrarily from the responses. Alice then generates her key pair and proceeds to acquire verification from her new neighbors. Acquiring verification requires Alice to sign her neighbors' certificates and Alice's neighbors to sign her certificate. See figure 8.1.

It is important to note that since Alice requires verification from her neighbors, it is difficult for a malicious user to arbitrarily increase the dimension of the HoT by assigning Alice an address outside the current dimension. Existing members can choose not to verify Alice in a new dimension if they believe that another empty address within the current dimension is a better choice. In order to join the HoT, it may be necessary for Alice to negotiate with members about which address she should occupy if her choice is not satisfactory to existing members.

8.2.2 Removing Members

Removing a member from the HoT will create a hole. At this point, an existing node needs to take over for it.

Method 3. Given a member, Alice, to be removed from the HoT, Alice must be replaced with either an immediately available new member or the existing member who was previously the surrogate for Alice's address.

Once a member, Bob, takes over for the removed member, he must acquire verification from the neighbors of this newly acquired address. Using this method, it is possible that Bob will hold more than a single additional address. Because of this, it is in the best interest of the HoT to fill internal vacancies before filling vacancies in the outermost dimension in order to maintain uniform node distribution.

Chapter 9

Discussion

Current darknet implementations which use WoTs are insecure because there is little control over the dissemination, revocation, and authentication of certificates. For example, the popular open-source darknet application WASTE automatically establishes a fully connected WoT by broadcasting new user certificates to all existing users. Upon receiving the new certificate, an existing user is prompted to accept or, more frequently, automatically accepts the certificate. Thus, a single user is capable of, and responsible for, the addition of a new user who all existing users will likely trust immediately. In addition, there is no support for revoking certificates.

These problems can be overcome by forcing the WoT into a predetermined structure at its inception and throughout its existence. Security improvements achieved through structure include requiring multiple users to verify each new user's certificate and enabling the revocation of certificates. While this can be accomplished in unstructured WoTs, using certain structures, specifically those commonly used as parallel computing topographies,

improves trust metric efficiency and limits malicious collusion. Trust metric calculations become highly efficient, running in either linear or sub-linear time.

The attack-resistance of structured WoTs can be calculated probabilistically. Given a fixed probability of any member being malicious, it is possible to calculate the probability of the other members being able to maintain secure communication, or of the WoT maintaining a state of sufficient connectedness. Example calculations for HoTs show strong attack-resistance for small groups on the order of fifty members. In addition, HoTs provide a means for distributed monitoring of certificate conflicts enabling members to actively combat malicious users automatically.

An implementation of a darknet application which supports the HoT concept is planned as future work. Such a darknet application could appeal to groups that are currently forced to use other heavy-weight or inflexible solutions for the sake of increased security.

Bibliography

[1] P. Biddle, P. England, M. Peinado, and B. Willman. The darknet and the future of content distribution. In *2002 ACM Workshop on Digital Rights Management*, 2002.

[2] Mike Speciner Charlie Kaufman, Radia Perlman. *Network Security, Private Communication in a Public World*. Prentice Hall PTR, 2002.

[3] Jianer Chen, Iyad A. Kanj, and Guojun Wang. Hypercube network fault tolerance: A probabilistic approach, 2002.

[4] Qinglin Jiang. *Improving the Robustness of Webs of Trust*. PhD thesis, North Carolina State University, 2006.

[5] Qinglin Jiang, Douglas S. Reeves, and Peng Ning. Improving robustness of pgp keyrings by conflict detection. In Tatsuaki Okamoto, editor, *Topics in Cryptology CT-RSA 2004*, volume 2964, pages 194–207. Springer-Verlag GmbH, 2004.

[6] Raph Levien and Alex Aiken. Attack-resistant trust metrics for public key certification. pages 229–242.

[7] Ueli Maurer. Modelling a public-key infrastructure. In *ESORICS: European Symposium on Research in Computer Security*. LNCS, Springer-Verlag, 1996.

[8] A. Menezes, P. Van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.

[9] Eunseuk Oh and Jianer Chen. Parallel routing in hypercube networks with faulty nodes. In *ICPADS*, pages 338–345, 2001.

[10] Michael K. Reiter and Stuart G. Stubblebine. Resilient authentication using path independence. *IEEE Transactions on Computers*, 47(12):1351–1362, 1998.

[11] Youcef Saad and Martin H. Schultz. Topological properties of hypercubes. *IEEE Transaction on Computers*, 37(7):867–872, July 1988.

[12] Douglas B. West. *Introduction to Graph Theory*. Prentice-Hall, Inc., 2 edition, 2001.

APPENDIX

Appendix A

Python Code

The file "sufficient-connectedness.py"calculates the probability of sufficient connectedness for HoTs and outputs a LaTeXtable.

```
import math

def factorial(n):
    """Returns n-factorial"""
    if n < 0:
        return 0
    elif n > 0:
        return n * factorial(n - 1)
    else:
        return 1

def num_combos(m, n):
    """Returns the number of ways to sample m items from a population n"""
    return factorial(n) / (factorial(m) * factorial(n - m))

def sufficient_connectedness(d, p):
    """Returns the probability of sufficient connectedness
    d is the dimension of the hypercube
    p is the probability of nodes being malicious

    """
    n = 2 ** d
    maxmal = int(math.floor((d - 1) / 2)) # max malicious users

    # calculate the probability that m <= maxmal
    sump = 0
    for i in range(0, maxmal + 1):
        sump += num_combos(i, n) * (p ** i) * ((1 - p) ** (n - i))
```

```
    return sump

ps = [0.1, 0.05, 0.01]
ds = range(5, 9)

scps = []

for i, d in enumerate(ds):
    scps.append([])
    for j, p in enumerate(ps):
        scps[i].append(sufficient_connectedness(d, p))

# print a latex table of sufficient connectedness probabilities
print "\\begin{tabular}{lccc}"
print "  $p$ & %s \\\\" % " & ".join(["%0.2f" % p for p in ps])
print "  \hline \\\\"
for i, d in enumerate(ds):
    print "  $%d$-HoT & %s \\\\" % (d, " & ".join(["%0.3f" % p for p in scps[i]]))
print "\end{tabular}"
```