Public Data: a New Substrate for Key Verification in DNSSEC

UCLA Computer Science Technical Report # 100020

Eric Osterweil
UCLA
eoster@cs.ucla.edu

Dan Massey
Colorado State University
massey@cs.colostate.edu

Beichuan Zhang
University of Arizona
bzhang@arizona.edu

Lixia Zhang
UCLA
lixia@cs.ucla.edu

September 7, 2010

Abstract

Motivated at least partly by operational problems associated with deploying global-scale PKIs, a growing number of alternative approaches propose to verify cryptographic keys by cross checking their consistency from topologically distinct locations and over time. These systems are experiencing growing operational use, but there has been no rigorous analysis to show the advantages and limitations. This paper provides a formal model of a consistency checking key learning and verification approach based on the concept of *Public Data*. Public Data offers a probabilistic description of users' risks based on the structure of their own deployments. A user of this framework can provision her own *Community of Trust* in such a way that she can reduce and accurately estimate the probability of being spoofed by an adversary. The results are applied specifically to the DNS Security (DNSSEC) problem and show that after reasonable provisioning, a user can force an adversary to pay an unrealistic cost to launch a successful attack.

1 Introduction

Cryptography is a powerful tool in the arsenal of the security community. However, the protection of a cryptosystem only begins after clients have securely learned and verified its cryptographic keys. One of the most common examples of a key learning system is the Public Key Infrastructure [15] (PKI), which is a hierarchical model that creates a very structured process for verifying keys. A growing number of efforts aim to deploy very large-scale public key cryptography systems in the Internet, but overall operational success has been limited [17]. One prominent attempt to deploy such a system is the DNS Security Extensions [2, 4, 3] (DNSSEC). DNSSEC uses a hierarchical key verification model which is similar to traditional PKIs. However, even though the global DNS root has deployed DNSSEC recently, operational evidence [18] is still quite strong that the hierarchical key verification model of DNSSEC is not coalescing. Roughly 98% [19] of the secure *islands of trust* in the DNSSEC deployment are isolated zones that do not have a secure delegation chain that allows resolvers to verify their keys, and many zones remain several delegations below the root and have no incentive mechanism to coerce their parent zones to deploy DNSSEC.

Motivated at least partly by pressing operational problems, a growing number of alternative key learning and verification approaches have begun to emerge in the literature [16, 23]. A common theme in these systems is to verify cryptographic keys by cross checking their consistency over

time and from topologically distinct locations. Proponents of this approach argue that the systems can offer a good deal of practical security and are ready to be used *now* to solve real operational problems. At the same time, others view these observation and consensus based approaches as ineffective, less than rigorous, and easy to subvert. These criticisms stem from the fact that there has been no rigorous analysis of the effectiveness of observation and consensus-based key verification systems.

This paper presents *Public Data*, the first formal model of a consistency checking key learning and verification approach for an Internet-scale system. The Public Data framework predicates its verification on the observation that the Internet's topological path diversity is what offers clients protection against an adversary, and the specific deployment provisioning that one uses is what determines how robust their protection is. Thus, the Public Data approach can offer a probabilistic description of users' risks based on the structure of their own deployments. This approach gives users the ability to tighten their own security through their operational decisions. Using DNSSEC as the subject of our model, we apply the Public Data framework to meet the needs of an actual Internet-scale cryptosystem. This work aims to provide a more rigorous analysis and verifies the analysis with large-scale simulations over an Internet-scale topology. The results indicate that a user of this framework can provision her own *Community of Trust* in such a way that she can reduce and accurately estimate the probability of being spoofed by an adversary. Further, we find that after reasonable provisioning, a user can force an adversary to pay an unrealistic cost to launch a successful attack.

The remainder of this paper is organized as follows: we begin by describing how the design of DNSSEC addresses key verification in Section 2. Section 3 defines our threat model. Section 4 both introduces the Public Data model, defines terminology needed for later analysis, and describes the verification model. Section 5 presents our security analysis and Section 6 evaluates the results. Finally, Section 7 discusses related work and Section 8 concludes the paper.

2 Background

The Domain Name System [14] (DNS) maps domain names such as `www.ucla.edu` to a wide range of data including IP addresses, email services, geographic locations, and more. These names form a tree-like hierarchical name space; each node in the tree, except the leaf nodes, is called a *domain*. At the top of the tree, the root domain delegates authority to *Top Level Domains* (TLDs) like `.com`, `.net`, `.org`, and `.edu`. The `.com` domain then delegates authority to create the `google.com` domain, `.edu` delegates authority to create `ucla.edu` domain, and so forth. The repository of information that makes up the domain database is divided up into logical name spaces called *zones*, which each belongs to a single administrative authority and is served by a set of *authoritative name servers*. This provides redundancy and fault tolerance through spacial diversity.

Security was not a primary objective when the DNS was designed in mid 80's and a number of well known vulnerabilities have been identified [6, 5]. The DNS Security Extensions' [2, 4, 3] (DNSSEC's) design goal is to prove that data in a DNS reply is authentic. In order to do this, each zone creates public/private key pairs and then uses the private portions to sign data. Its public keys are stored in DNS records called `DNSKEY`s, and all the signatures are stored in records called `RRSIG`s. In response to a query from a DNS resolver, an authoritative server returns both the requested data and its associated `RRSIG`s. A resolver that has learned and authenticated the `DNSKEY` of the requested zone can verify the *origin authenticity* and integrity of the reply data. To resist replay attacks, each signature carries a definitive expiration time.

In order to authenticate the `DNSKEY` for a given zone, say `ucla.edu`, the resolver needs to construct a *chain of trust* that follows the DNS hierarchy from a trusted root zone key down to the key of the zone in question (this is shown in Figure 1). In the ideal case, the public key of the DNS

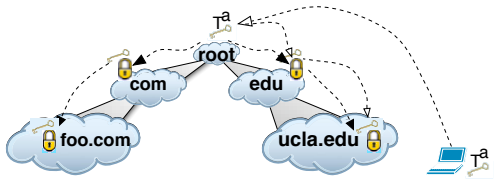

Figure 1: Using the root zone's key as a trust anchor (T^a) resolvers trace a "chain of trust" down the DNSSEC hierarchy to any zone.

root zone would be obtained offline in a secure way and stored at the resolver, so that the resolver can use it to authenticate the public key of `.edu`. Then, the public key of `.edu` would then be used to authenticate the public key of `ucla.edu`.

There are two challenges in building the chain of trust. First, a parent zone must encode the authentication of each of its child zone's public keys in the DNS. To accomplish this, the parent zone creates and signs a Delegation Signer (`DS`) record that is a hash (SHA-1, SHA26, etc.) of a `DNSKEY` record at the child zone, and signs it with *its own* key. This creates an authentication link from the parent to child. It is the child zone's responsibility to request an update to the `DS` record every time the child's `DNSKEY` changes. Although all the above procedures seem simple and straightforward, one must keep in mind that they are performed manually, and people inevitably make errors, especially when handling large zones that have hundreds or thousands of children zones.

Moreover, the parent and child zones belong to *different* administrative authorities, and each may decide independently if and when they turn on DNSSEC. This leads to the second and more problematic challenge. If the parent zone is not signed, there is no chain of trust leading to the child zone's `DNSKEY`. This orphaned key effectively becomes an isolated trust anchor for its subtree in the DNS hierarchy. To verify the data in these isolated DNSSEC zones, one has to obtain the keys for such isolated trust anchors offline in a secure manner. DNSSEC resolvers maintain a set of well-known "trust-anchor" keys (T^a) so that a chain of key sets + signatures (secure delegation chain) can be traced from some T^a to a DNSSEC key lower in the tree. The original DNSSEC design envisioned that its deployment would be rolled out in a top-down manner. Thus only the root zone's key would need to be configured in all resolvers' T^a sets and all secure delegations would follow the existing DNS hierarchy. However as of this writing, operators of large DNSSEC TLDs (like `.org`) still do not have provisions for accepting and maintaining `DS` records from their children zones because of the operational difficulties. A full description of these difficulties can be found in prior work [20]. Ultimately, without a secure delegation hierarchy, `DNSKEY` messages can be spoofed as easily as ordinary DNS messages.

3 Threat Model

We begin by identifying end-point participants of this system as users (such as Alice), the DNS resolvers, and zone name servers. In this analysis we assume that these endpoints are operating properly and truthfully. Since our goal is to verify cryptographic keys by cross checking their consistency from topologically distinct locations, we choose to focus on the Man in the Middle (MitM) attack.

To launch such an attack, an adversary (Eve) must be able to capture and replace data packets that are in transit. Thus, Eve must first be able to observe these packets from a *vantage* that she has access to, and must then be able to interpose in communications. For example, if Eve has

compromised a router in an Internet Service Provider (ISP) then she might be able to use Deep Packet Inspection (DPI) to identify when the router is forwarding either a DNS query for a `DNSKEY` or a DNS response, and would then need to replace it with an invalid value. Another example might be if Eve is on a collision domain (such as a WiFi network or on a hub) with any IP hop that is passing this information and she simply spoofs an answer. In this case she can passively listen to the broadcast domain and inject her own responses. Thus, the path that communications traverse can lead to an attack if an adversary is among the vantages the compose the path. This is sometimes referred to as an on-path attack in the literature.

However, the work that Eve must do to successfully subvert Alice is more complicated than intercepting a single point-to-point message. In DNSSEC, an attack is actually launched against both Alice's resolver and the set of all name servers that serve a DNS zone (not just one). In other words, Eve is attacking a set of servers who can each report a zone's genuine key. If she spoofs just one, then Alice can detect her attack by querying the other name servers that serve the zone. In addition Eve must also consider the operationally feasibility for her to launch a prolonged attack. Specifically, if her attack is detected, she can expect corrective action to be taken. Thus, Eve must consider attack scenarios in which she can subvert Alice while controlling the visibility of her attack.

In order to simplify our model we choose to focus our analysis on DNS query response messages, and not queries. The reason for this is that any attacks Eve can launch against queries from a resolver can be directly modeled as attacks launched on responses from a name server. For example:

- If Eve decided to drop a query, this would have appeared the same as if she had dropped a response message.
- If Eve replaced a query with another that elicited a different response from a name server, then she could also just replace a response with whatever she liked.
- If Eve delayed the query, she could also have just delayed the response.

Thus, for simplicity and without loss of generality, we assume that adversaries only tamper with DNS reply messages.

4 Public Data Model

The intuition of the *Public Data* model is that when a piece of data becomes widely known in public, it becomes difficult to spoof it. If we consider a particular user (say Alice) and we allow her to compare her own observation of a datum with observations witnessed at other topologically diverse locations, then the can rely on the topological diversity of Alice's witnesses to protect her against a spoofing attack. When one considers that Eve (a MitM adversary) can only succeed in her attack *if* she can convince Alice that a spoofed value is valid, then Eve's job becomes much harder if a datum's genuine value is observed from very diverse (and independent) vantage points. The following discussion illustrates the foundation of our analysis. It first outlines the basic components of the *Public Data* model, and then uses them to formalize its distributed verification process.

4.1 Components

The critical components of the Public Data model are the set of clients that query for data and the set of servers that provide data. Both clients and servers are associated specific locations in the network, referred to as *Vantages* below. In addition to defining Vantages below, we also formally define Public Data, its properties, and the types of messages exchanged in our model.

All Public Data is requested from, served from, and transferred through *Vantages*. We model the Internet as a set of network vantages whose connectivity to each other can be expressed as the graph $G = (V, E)$. A vantage is essentially a network node at an IP address, more precisely $v_i \in V$. Examples of vantages include DNS resolvers, name servers, IP routers, etc.

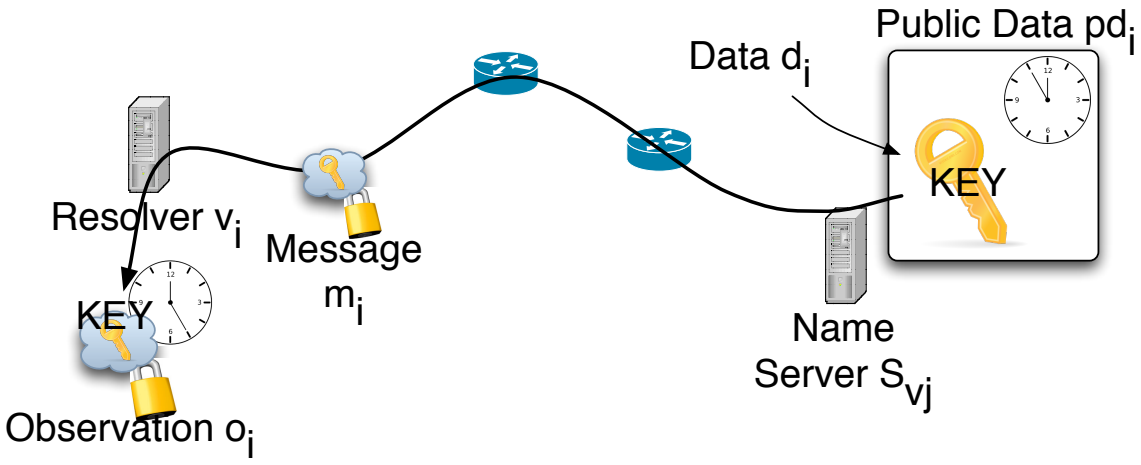

Figure 2: Here we can see that Data Source S_{v_j} has a public datum pd_i which represents the public representation of d_i, and the time at which it was created. A resolver at v_i queries for S_{v_j}'s key and uses the response message (which contains the current key d_i) to create an observation o_i, which contains the time at which v_i saw the data item.

Each DNS name server is located at some Vantage and is responsible for serving a zone's data. Therefore, we define these as *Data Sources* in our model. As seen in Figure 2, a Data Source at Vantage v_j is denoted S_{v_j}. When a datum d_i is served from a Data Source S_{v_j}, it becomes an immutable Public Datum, $pd_i = (d_i, t_k)$, where d_i is the datum itself and t_k is the inception time of d_i. A *Public Data Source* $S_{v_j} = \{pd_0, \ldots, pd_m\}$ is characterized by its network vantage of v_j and all of the public data it has ever served[1].

Data is exchanged using a message $m = (d_i, Sig_K(d_i))$ where d_i is an opaque data item, $Sig_K(d_i)$ is a signature covers d_i, and the signature can be verified by the crypto key K. A message m_i sent from v_j to v_k will traverse some single acyclic path of vantages denoted: $\sigma_{(j,k)} = (v_j, \ldots, v_k)$. When vantage v_j receives the message, it creates an observation: $o_i = (v_j, m_k, t_l)$ that contains the vantage that made it, the message, and the time at which it was received. Furthermore, if the recipient has already learned the verifying key K, then the message's authenticity can be checked. While this seems like a cyclic dependency for *learning and verifying* `DNSKEY`s, it can be useful for other communications sent by a peer vantage from whom the key has been learned out of band. This is discussed further in Section 4.2.

Finally, we also assume that even if a server S removes one piece of data d from its currently served set, it must be possible to prove that S served d in the past (a weak form of nonrepudiation). This mechanism is an important part of verifying Public Data and it will help protect client resolvers from malicious servers. For example, with this property a server will be unable issue a bogus DNS `A` record and then later claim that this data was never served to a client. Note this is already a property of DNSSEC, as all DNS record sets are sent with cryptographic signatures covering them.

4.2 Verification Processes

We begin the description of the Public Data verification processes by observing that the *validity* and *verifiability* are distinct concepts and are, therefore, treated as such. In our model, validity is

[1]The physical implementation of how data is stored and where is an important consideration, but for the purposes of clarity, we focus on the abstract model for now.

a notion of the true authenticity of data. To define validity, we introduce a message oracle Θ. The oracle is always able to determine validity with 100% certainty[2]. For a given datum d_i and time t_i, $valid(\Theta, d_i, t_i) = true$ iff 1) d_i came from the source it reported, 2) the time stamp t_i does not occur *before* the data was actually created at the source, and 3) d_i was still being served by the data source at t_i (i.e. it is not a replay of older data). Thus, $valid()$ only determines if an observation properly reflects data from a Public Data Source (this is not "ground truth" in the sense of whether the data's meaning holds any real-world significance). Based on this, we can define an attack as $valid(\Theta, d_i, t_i) = false$.

Since clients do not have access to Θ, they are instead concerned with the *verifiability* of data. A resolver at v_i creates a query message m_i at time t_0 and sends it to a name server S_{v_j} over the path $\sigma_{(i,j)}$. When the server S_{v_j} receives this query, it constructs a response message m_j (containing the data d_k from its current pd_k), and sends it back over the reverse path $\sigma_{(j,i)}$.

Our definition of *verifying* a data item d_i casts verification as a function of a continuous metric that uses a set of observations O and user-defined threshold value p. The observations (O) are used to perform consistency checking of the data items. Each observation in O could (for instance) be an observation made from Alice's vantage v_i to a specific name server $S_{v_j} \in V_Z$.

We can see, however, that as one adds more observations from different vantages into O there is an increased chance that this will add paths that do not intersect with those already in O. Therefore, adding independent paths *may* make the number of vantages that Eve must subvert larger too (in order to keep $verify()$ from converging on the valid answer)[3]. Thus, we define a *Community of Trust* V_{CoT} as a set of vantages (called *witnesses*) from which Alice can cull additional observations. The idea is to let Alice leverage her own judgment of real-world trust in other operators, or well known services to help add path diversity to her O. In this model we require Alice to have obtained the public keys for these witnesses through an out of band mechanism (in order to verify messages from them). By learning the keys from her V_{CoT} ahead of time Alice can verify the crypto signatures on communications with her witnesses, and rely on classical cryptography for protection of these messages.

When Alice queries her V_{CoT}, and some observations report different values, any datum with greater than a p majority is chosen as the verified datum. In other words, p is the proportion of nodes that must agree to call a data item "verified." The specific algorithm for $verify(O, p)$ is described in Algorithm 1.

5 Security Analysis

Compromising routers and end hosts requires some combination of (relatively) uncommon skills, access to people with these skills, and the means to enlist help. In our model, attacks *cost* resources, which might take the form of time, money, social leverage, etc. The goal of this model is to let users quantify their vulnerability to a MitM attack. This estimation is based on both the specifics of their Community of Trust (V_{CoT}) and the zones the are querying (V_Z). In other words, we want to let Alice estimate the probability that Eve can spoof her, and how much Eve would have to spend to do so.

To launch a successful attack, an adversary may have to target well protected vantages, and the cost may be non-negligible. We quantify an overall cost function $C(V_e, t)$ in terms of two component functions: i) the cost of *acquiring* nodes $c_a(V_e)$, and ii) *usage* time $c_t(V_e, t)$. We note that approximating the cost of this sort of activity becomes quite difficult as one attempts to make

[2]It is important to note that Θ is not accessible to any $v_i \in V$ and is only defined to disambiguate if $d_i \in pd_i$ or $d_i \notin pd_i$.

[3]We note that adding observations to O does not necessarily add path independence, and we address this in Section 5.1

Algorithm 1: $verif(O, p)$: Returns a datum with higher than p proportion of agreement, or the empty set.

```
begin
    Input: O
    Input: p
    Output: d^verif
    d^verif = ∅

    /* Count of the observation seen the most frequently */
    max_count = 0

    /* The data value with the most observations */
    d^max = ∅

    /* A hash of counts for each data item seen */
    counts[] = ∅
    foreach o_i ∈ O do
        /* Extract the data item contained in the observation */
        d_i =extract_data(o_i)
        counts[d_i]++

        /* If this is the most popular, set as "max"*/
        if counts[d_i] > max_count then
            max_count = counts[d_i]
            d^max = d_i
    /* If max variable exceeds user-specified threshold, return it */
    if max_count > (p × |O|) then
        d^verif = d^max
end
```

a precise estimate. Rather than attempting to achieve this elusive goal, we present this formulation as just a single high level candidate cost formulation, and we use it as a starting point for our analysis.

Acquisition We define the *Acquisition* cost in terms of the difficulty an adversary faces in compromising a node, and thus increasing the spread of her attack. For example, some nodes (such as core routers at large ISPs) may be difficult to access, and may take uncommon skill-sets to compromise. Previous work [13] has noted the existence and nature of an Internet blackmarket economy in which (among other things) routers are rented as a commodity. Here, it suffices to say that specific routers at specific locations (such as the core of a very large transit ISP) may not be for sale, or may be sold at a premium. While the level of effort needed to obtain *specific routers* can vary widely with different targets, we begin with a simple generic metric as an approximation for this difficulty.

When Eve wants to attack she intends to spoof answers between a data source S_{v_i} and a client v_j, and to do this she must control at least one vantage $v_e \in \sigma_{(i,j)}$. In order to succeed in an attack between the set of name servers for a zone V_Z and a CoT V_{CoT}, Eve must have a set of attack vantages V_e that can intercept response messages.

However, *discovering* what nodes need to be in V_e is a component of cost too, and there can be a cost in finding this out. For example, trying to identify the interfaces on an ISP's router might require social engineering, or possibly cost real money. Considering that each node in V_e

may have both different acquisition and discovery costs (depending on where it is, who owns it, etc), we propose Equation 1 as our candidate acquisition cost function.

$$c_a(V_e) = \sum_{i=0}^{|V_e|} c_{intr}(v_i) + c_{disc}(v_i) \tag{1}$$

This expression embraces the fact that each node may potentially have a different intrusion cost $c_{intr}()$ and a different discovery cost $c_{disc}()$. We, therefore, suggest that it is sufficiently nonconstrictive and high level that it can serve as a candidate representation that is even applicable to some complex cost models.

Usage The usage portion of Eve's cost logically models the notion that Eve may have recurring costs to maintain her V_e, or perhaps faces a cost that accrues over time, and that these costs may even be non-stationary (i.e. they may vary over time). For example, if Eve is snooping traffic on a router, then that router will have to inspect its traffic (DPI). This activity will result in increased CPU load, and she might eventually be *detected* when operators investigate why a router is overloaded. Clearly this problem is more pressing on large core routers at major ISPs than in small home offices (SOHO) routers. In this case, we make a broad generalization that Eve's cost is proportional to the rate of detection λ_{detect}. Alternately, in some cases, Eve might be paying rent for access to a router that was compromised by someone else. As above, if we make a general assumption that each element in V_e may have a different usage cost and that this cost may even vary over time, then we can model her usage cost between time $t = 0$ and time $t = n$ with Equation 2.

$$c_u(V_e, t) = \sum_{t=0}^{n} \sum_{i=0}^{|V_e|} \lambda_{detect}(v_i, t) + c_{rent}(v_i, t) \tag{2}$$

We can see, by inspection, that as an attack is launched for a prolonged period, or as the number of nodes needed to engage in the attack grows, the cost function does too.

5.1 The Impact of Acquisition Cost

We start our discussion of the security analysis by temporarily simplifying Eve's attack model slightly. We assume that Eve only needs to have sufficient spread to spoof Alice, and that the temporal component is unnecessary.

In order for Eve to fully subvert Alice's Community of Trust (V_{CoT}), she must be able to intercept all messages between V_{CoT} and the name servers Alice is trying to reach (V_Z). From this observation, we generalize that Eve needs to be in the position to be able to partition V_{CoT} from V_Z, and she would like to minimize her acquisition costs in doing so. The lower bound on the *number* of nodes she needs to compromise is on the order of the minimum cut set: $|V_e| = O(MinCut(v_j, V_Z))$. The intuition here is that Eve's vertices must be able to disconnect (or partition) all messages from V_Z to anyone in Alice's V_{CoT}. For example, in Figure 3 if Alice is at vantage A and $V_Z = \{L, M\}$, then her min-cut set is order 1 (if vantage F is cut then she is disconnected). However, if she adds (say) vantage B to V_{CoT}, then her cut set grows to the oval $\{F, G\}$. Here we note that even if V_Z grows to $V_Z = \{L, M, N\}$ the min-cut set remains the same. However, if Alice adds C to her V_{CoT} the cut set grows to the circle: $\{F, G, H\}$. Thus, the cut set may not grown every time a source or destination is added, but sometimes it can. We define $V_{cut} = MinCut()$, and use $|V_{cut}|$ as a lower bound on the number of nodes needed for Eve's attack to succeed.

Conversely, Alice's goal is to raise Eve's cost in every way she can. More specifically, her best defense is to increase the size of her min-cut set by increasing the size and topological diversity of her V_{CoT}. In determining how to spend her resources, Eve faces a tradeoff between paying to learn

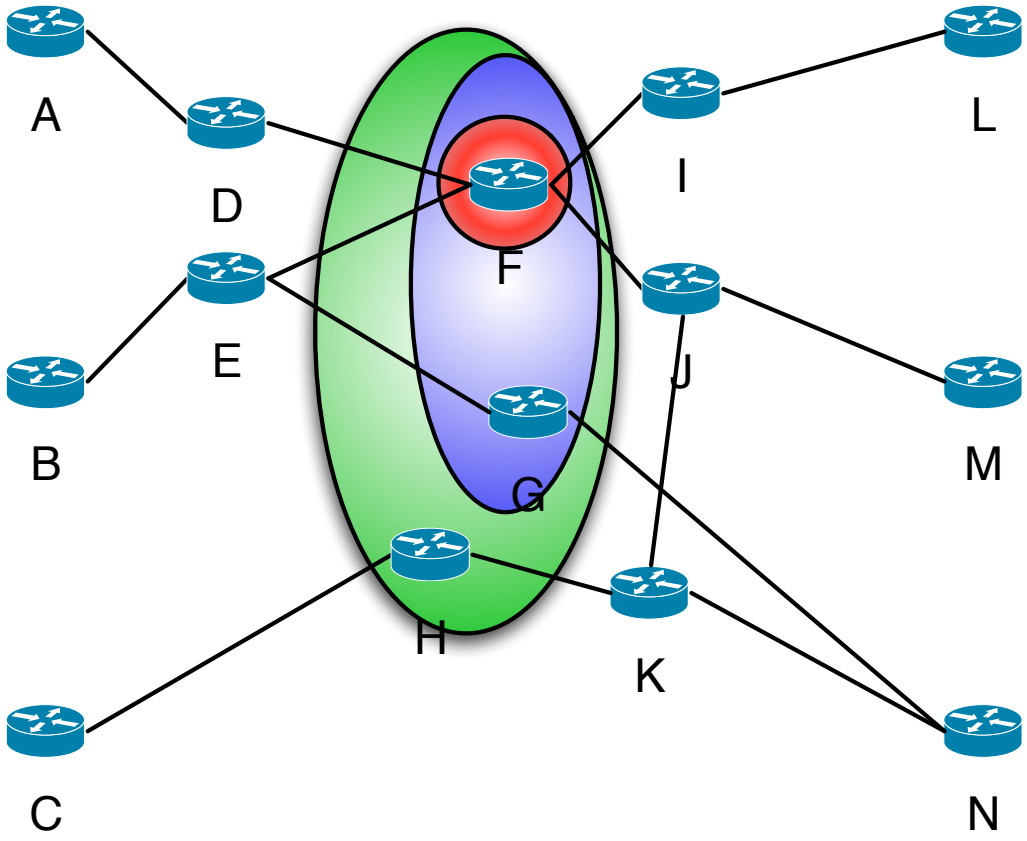

Figure 3: Here we can see that how adding either name servers or CoT members can increase the size of a min-cut set. if Alice is at vantage A and $V_Z = \{L, M\}$, then her min-cut set is order 1 (if vantage F is cut then she is disconnected). However, if she adds (say) vantage B to V_{CoT}, then her cut set grows to the oval $\{F, G\}$. Here we note that even if V_Z grows to $V_Z = \{L, M, N\}$ the min-cut set remains the same. However, if Alice adds C to her V_{CoT} the cut set grows to the circle: $\{F, G, H\}$.

which nodes need to be compromised in order to partition Alice from V_Z (which maximizes $c_{disc}()$), and uniformly compromising as many nodes as possible in hopes of partitioning Alice (maximizing $c_{intr}()$). Clearly, if Eve spends a lot on trying to discover which nodes to use then she may not have enough resources to cover V_{cut}. We conjecture that given a fixed target of just Alice (and not her V_{CoT}) Eve might map the BGP AS paths between v_i and V_Z, social engineer some view of intra-AS topology of each ISP on this path, then probe and try to compromise the specific routers in each of these ISPs. It is important to note that a heavy investment in discovering nodes ($c_{disc}()$) results in a high cost, and a strong possibility of failure. This is because determining Internet path information between arbitrary source / destination pairs is non-trivial, and remains an open research area even to Internet researchers [11, 10, 7, 9, 22, 12]. Complications range from the inherent difficulty in inferring either intra-AS or inter-AS topology, to routing path asymmetries ($\sigma_{(i,j)} \neq \sigma_{(j,i)}$), and more. Furthermore, paths are subject to both slow changes over time and changes in bursts.

Eve's first thought might be to focus her efforts on the upstream ISPs of V_Z (the zone's name

servers). However, this negatively impacts Eve in two ways: i) spending the resources on these servers does not allow her to subvert traffic to other zones (which makes her attack very focused, and not extensible to multiple targets), and ii) this makes her attack much easier to detect by the zone owners. If an operator for the zone has monitoring setup for their zone (such as SecSpider [1]), then a global attack is immediately visible and thus, much more likely to be shutdown. Clearly this is a feasible attack, but we consider attacks in which Eve is either focused on spoofing specific users, or at least spending her attack cost in such a way that she can be in position to attack multiple and changing sets of name servers for one or more zones.

To illustrate our security analysis we evaluate three different classifications of adversaries: i) General, ii) Targeted, and iii) Rank Order. Each of these adversary types is differentiated by the way that they choose which nodes to compromise.

General: In this model Eve's goal is to take the set of all possible vantages to compromise (V), and acquire as many of them as she can afford ($c_{disc}() = 0$). Of all the attack models we have considered, we propose that this one is the most relevant to general Internet users. Here, Eve has a general set of compromised nodes and may be focused on spoofing either a set of users (not just Alice), or may even be performing an unstructured attack against any target of opportunity.

We observe that Eve's chances of compromising Alice's specific V_{cut} set out of all sets of possible nodes V are the same as choosing a specific combination of $r = |V_{cut}|$ elements out of a set of size $|V|$:

$$Probability(V_{cut}) = \binom{|V|}{r}^{-1}$$

The intuition for this expression comes from the following reasoning: if there exists a min-cut set of size r, Eve has r chances to guess which routers are in this set (out of all $|V|$), and she is given no additional information, then she has an equal chance to guess this set among all other sets of size r.

We can extend this slightly to say that if Eve has $n = |V_e|$ choices (where $n > r$), then her chances are multiplied by the number of combinations that can be made with nodes outside the min-cut set $(n - r)$.

$$Probability_s(V_e) = \binom{|V|}{n}^{-1} \times \binom{|V - V_{cut}|}{n - |V_{cut}|} \tag{3}$$

Targeted: In this case, we presume that Even has learned all of the information about the path between Alice's v_i and V_Z. Though we have mentioned that this is generally an infeasible task, we give Eve the benefit of the doubt by assuming she has accomplished this some how. However, in this model, as assume that Eve does *not* have any path information between the rest of Alice's V_{CoT} and V_Z. This could be because members of Alice's V_{CoT} may not be common knowledge, or because it is generally unrealistic for anyone to learn this information about arbitrary paths in the Internet.

We will show that this type of attacker has roughly the same chances of compromising Alice's CoT as in the General case. This observation comes from the fact that as additions to Alice's CoT contribute independent paths, her min-cut set grows and changes. Therefore, Eve has the same type of moving target as in the general case, she just starts off with some good guesses.

Rank Order: Finally, we consider an adversary who chooses to spend her resources compromising the *largest* ISPs first, in the hopes of disconnecting Alice. Here the probability of success is related to the chance that the min-cut set is included in top n-most well connected ISPs (V_n), or $V_{cut} \subseteq V_n$. We will show that this approach is exorbitantly expensive, and attempting it is beyond

the capacity of all but the most well funded adversary, *and* that if Alice has bolstered her V_{CoT} sufficiently and is querying a large zone, then it is beyond reason for any adversary to be able to afford to successfully subvert her in this way.

5.2 The Impact of Usage Cost

When considering the likelihood that Eve will be able to spoof Alice, we observe that given sufficient spread her ability to spoof messages to Alice is also limited by a simple temporal function of opportunity. That is, if Alice makes an observation o_i at a specific time between t_0 and t_n, then Eve must have positioned her vantages and be attacking at that specific moment in the interval $T = t_n - t_0$. From this we can say that the probability that Eve can observe and spoof a message is a proportional to the amount of time that she spends observing T_e (as seen in Equation 4).

$$Probability_t(T_e) \propto \frac{T_e}{T} \tag{4}$$

We can see by inspection that as the amount time that Eve spends in T_e increases, the probability of intercepting a message increases too. However, the cost of using the nodes in V_e increases as well.

6 Evaluation

The goal of using Public Data to verify crypto keys is to provide a mechanism that allows users to bolster their security through enhancing their operational deployments. In this Section we begin by illustrating that as Alice increases her V_{CoT} and zone operators increase their V_Z, the cut set V_{cut} will grow too. Then, we analyze both the probability of compromise and the progression of cost for each of our attack classes. Due to space limitations, we focus our evaluation on the acquisition costs and probability models, and leave the evaluation of additional usage costs and probability to future work. We claim that this gives Eve a very large benefit, but will show that the Public Data approach still provides very solid assurances.

We performed our evaluation using a simulated Internet-like Autonomous System (AS) topology, in which each AS represents an ISP, educational institution, government agency, etc. This topology was generated by the Inet Topology Generator [8]. Using a topology of 22,000 nodes (which is similar in scale to the current Internet) we randomly chose vantages for Alice, her CoT, and for the name servers of a target zone. We varied Alice's CoT size (including her) from one to ten, and for each CoT size we varied the name server set size between one and ten. This gave us 100 combinations. We then ran ten simulation runs for each combination in which we varied the number of ASes that Eve had acquired from one to the full 22,000. In these attack simulations, we set the cost of compromising an AS node to be proportional to its degree (or connectivity). This is in the spirit of larger, more prominent ASes (like AT&T, Sprint, etc) have more internal routers and more internal path diversity. Thus, for an adversary to subvert an entire AS, she will have to secure more internal routers. Moreover, larger ASes (such as tier-1 provides) are likely harder to crack into than, say, smaller ISPs. Our belief is that this linear relationship between AS degree and cost is conservative in favor of Eve.

Cut-Set Using our topology, we mapped each combination of V_{CoT} and V_Z to the min-cut set V_{cut} between them. Figures 4, 5, 6 show a general trend of increase in min-cut set size as V_{CoT} and V_Z grow. Intuitively, the min-cut set should grow as the CoT increases. By adding more witnesses to the CoT, one gains more nodes and more paths. But note the addition of a new witness does not add value if its paths to the authoritative servers go through the previous cut set. The figures show how likely it is that when Alice adds a witness to V_{CoT}, her V_{cut} will grow.

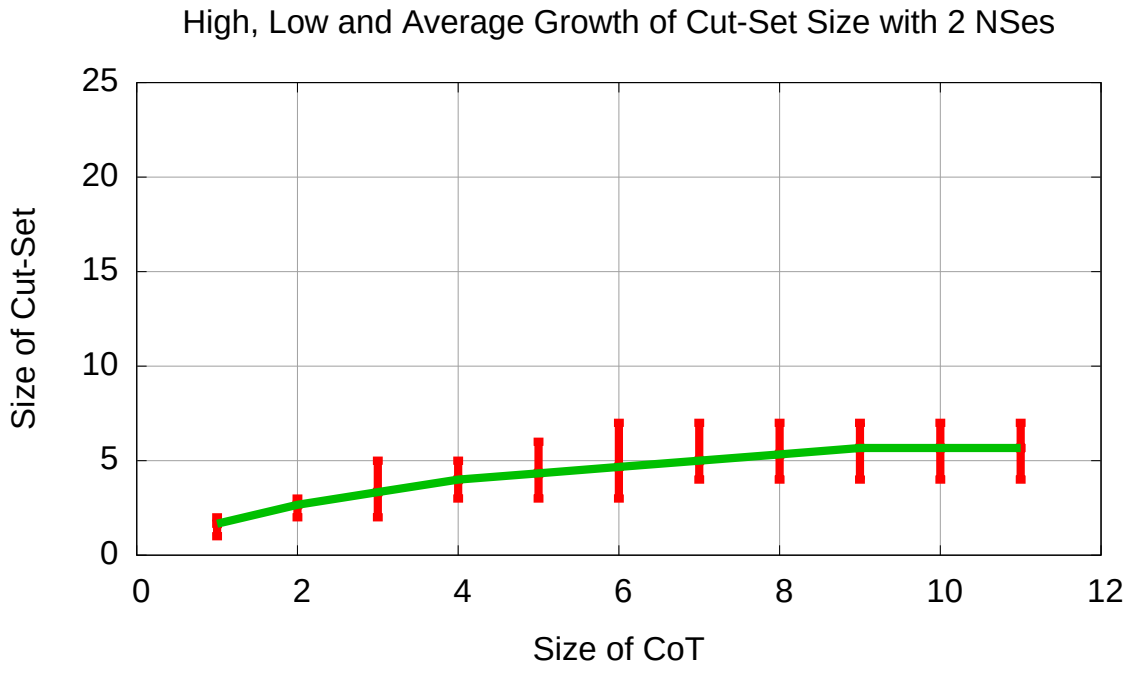

Figure 4: This figure shows the upper bound, lower bound, and average case of the size of V_{cut} when randomly choosing V_{CoT} and V_Z. Here we have the V_Z size chosen to represent small zones ($V_Z = 2$)

Cost and Probability Our approach in estimating Eve's costs is to try and be liberal in our beliefs. For example, in our evaluation we presume that for a certain "price" Eve can actually buy every AS in the Internet (which we normalized from 0 to 100). That is, we plot our simulations from one compromised AS all the way out to where every AS in the whole Internet belongs to Eve. While we claim that this extreme is wholly unrealistic, we ultimately leave the judgment of its believability to the reader. We simply use this extreme to illustrate the scaling properties of our model in the presence of such an all-powerful adversary. In this way we gain a working sense for how large Alice's min-cut sets will need to be to overcome adversaries ranging from impoverished to very powerful. We demonstrate this by simulating attacks and calculating the success rates of our three classes of adversaries: general, targeted, and rank order.

Figures 7, 8, and 9 illustrate that as a *General* adversary, Eve's success rate at subverting Alice closely follows our predictions in Equation 3. We can see that if Alice's V_{cut} is on the order of size two, then Eve has a roughly linear chance of spoofing her. While this is certainly important, we note that in Figures 4-6, we can see that Alice can generally increase her V_{cut} size by adding witnesses to her V_{CoT}. In other cases, even when V_{cut} is only size 6, Eve's investments yield only very marginal success rates until her cost approaches roughly 80% of the cost of the all ASes in the Internet.

We next consider the *Targeted* adversary. Recall that she has taken the time to learn Alice's AS path, and compromises ASes in it, before resorting to a general attack against the rest of Alice's CoT. From Figures 10, 11, and 12, we can see that while Alice has a very small min-cut set (presumably from a small CoT), Eve's success rate surpasses our model's predictions. However, as Alice's CoT grows, Eve's chances begin to closely resemble those of a general adversary. Additionally, we can see that even in the average case with a $|V_{cut}| = 8$, Eve's probability of success

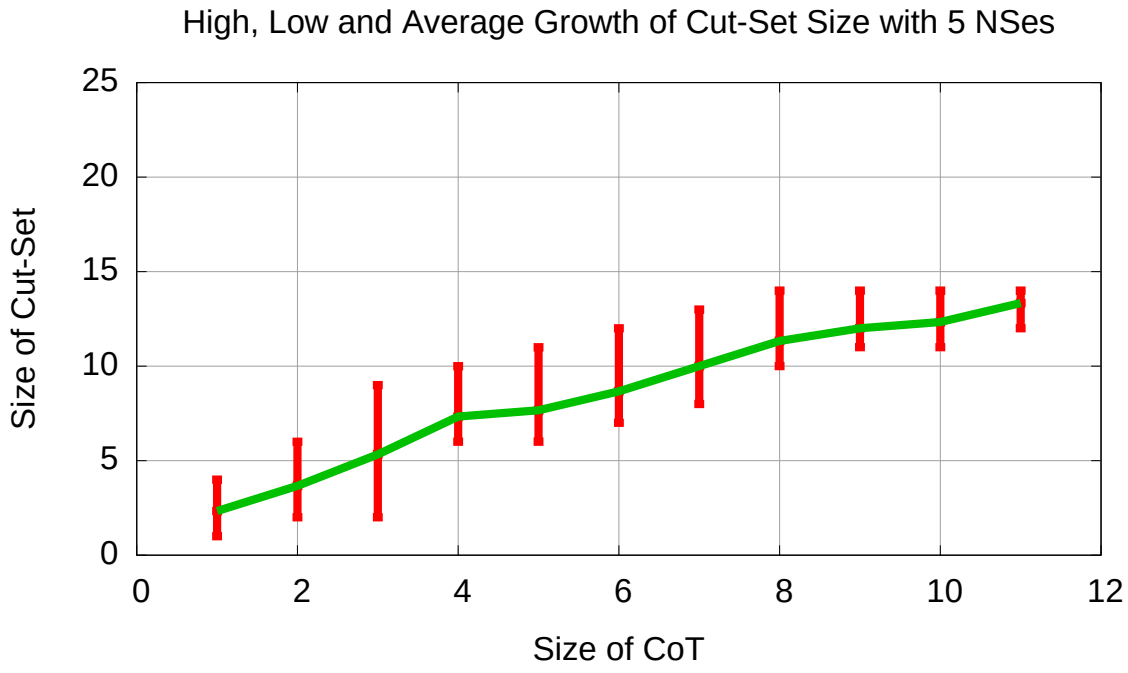

Figure 5: These figures show the upper bound, lower bound, and average case of the size of V_{cut} when randomly choosing V_{CoT} and V_Z. Here we have the V_Z size chosen to represent medium sized zones ($V_Z = 5$)

only begins to reach 25% after she has spent approximately 70% of the total cost to compromise the entire Internet. Thus, Alice's road to salivation, as with a general adversary, is through augmenting her CoT.

Lastly, we examine the Rank-Order adversary. Here Eve likely represents a well funded organization. This is a crucial observation because we can see from Figures 13, 14, and 15 that the cost to her becomes on the order of that to "purchase" the entire Internet within the first percentages of ASes that she compromises. Thus, one must either be facing an incredibly powerful adversary (possibly a nation-state), or an someone who has managed to permeate every AS in the world. Moreover, we can see that if Alice has a $|V_{cut}| \approx 14$, Eve spends 90% of the total cost of the Internet before subverting Alice in more than 18% of the cases. One interesting note from this class of adversary is the odd step-function-like behavior of the graphs. This behavior comes from the fact that there are high degree hub-nodes (which resemble tier-1 ISPs) that seem like they *should* belong to V_{cut} in almost any situation. However, the simulations show that with a sufficiently high path diversity (from a large V_{CoT}), connectivity between witnesses in V_{CoT} is pushed to peerings at the edge of the network. In *these* cases, a brute force attack on the Internet's core tier-1s will still not subvert the protections of Public Data verification until quite a high cost is paid.

7 Related Work

As noted in Section 1, a growing number of systems have begun to emerge that use distributed key learning approaches to verify cryptographic keys. While these systems include certain similarities and differences, the fundamental approach is very consistent.

SecSpider [16] is a distributed key learning system that is designed for DNSSEC. It polls for

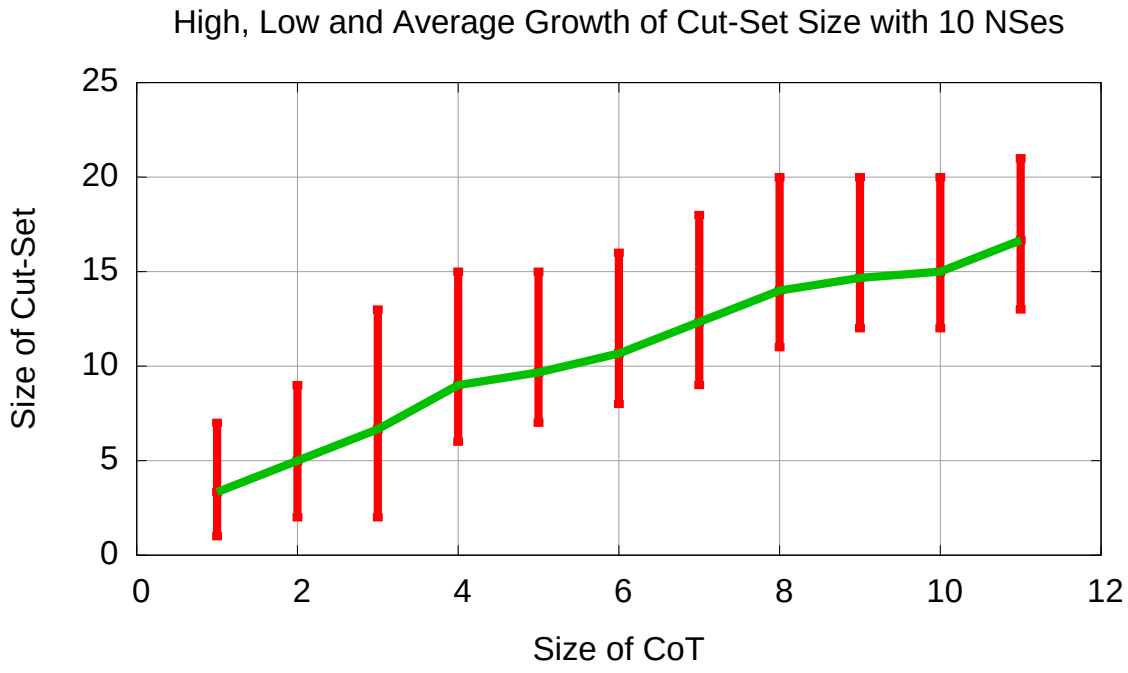

Figure 6: These figures show the upper bound, lower bound, and average case of the size of V_{cut} when randomly choosing V_{CoT} and V_Z. Here we have the V_Z size chosen to represent larger TLD/root zone sizes ($V_Z = 10$).

DNSKEYs from several distributed vantages throughout the world, and it has been operational as a distributed key system since 2007. In SecSpider, DNSKEYs are are archived and a full history has been kept of keys going back to when the system first started monitoring the DNSSEC rollout in 2005. However, SecSpider does not adhere to any formal model that describes the degree to which clients can be certain that an adversary has not spoofed its pollers.

Another operational system is Perspectives [23] which is designed to gather SSH public keys through a set of "notary" servers, which are deployed in different locations. This system is an on-demand system that also maintains a notion of how long SSH keys have been seen. However, like SecSpider, Perspectives' model does not attempt to quantify the level of trust users should have that the keys it verifies are genuine.

One other key learning approach that is relevant is SDSI [21]. SDSI is not an operational Internet key learning system (as those above are), but it is a framework in which the veracity of keys can be determined through local attestations. That is, when a user decides to give trust to an entity, that entity is then able to vouch for other keys, which allows for a grass-roots style of key verification.

8 Conclusion

In this paper we show that one is able to cast verifiability of keys as a function of both an adversary's cost to compromise and the probability that she will be able to succeed in an attack. By using topologically diverse witnesses, and a structured framework for key continuity we have created a model in which a key's presence as Public Data makes it, essentially, capable of self-verifying. We presented the formulation of both a cost model and a probability expression and experimented with

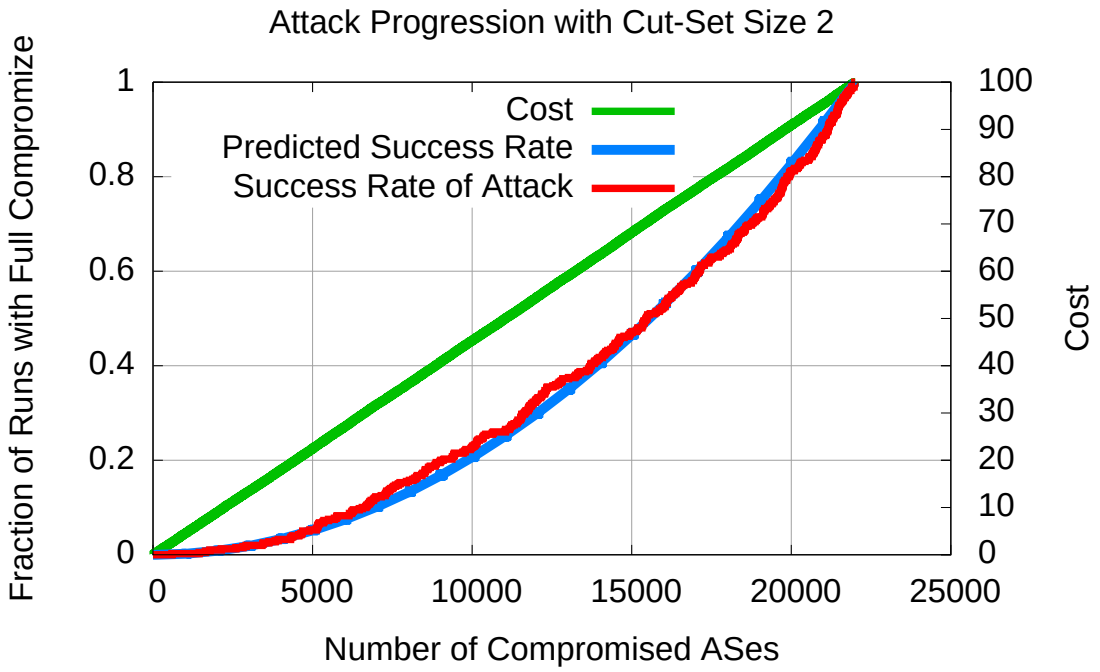

Figure 7: The predictions from Equation 3 very closely match the experimental results for the General type of adversary. Here we see a $|V_{cut}| = 2$.

Figure 8: The predictions from Equation 3 very closely match the experimental results for the General type of adversary. Here we see a $|V_{cut}| = 8$.

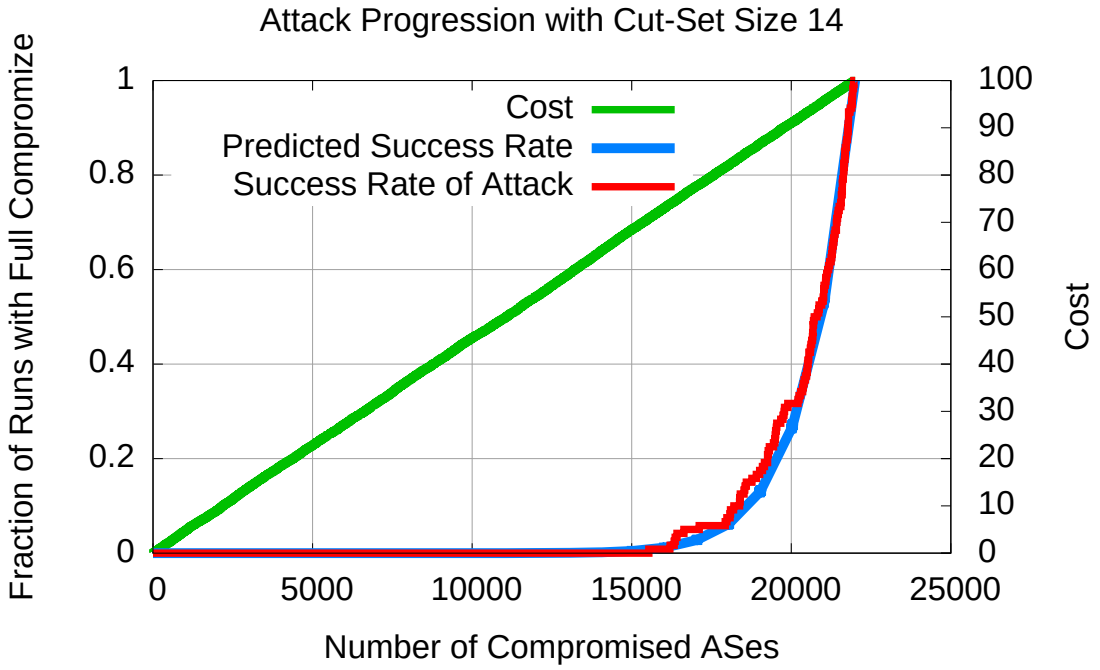

Figure 9: The predictions from Equation 3 very closely match the experimental results for the General type of adversary. Here we see a $|V_{cut}| = 14$.

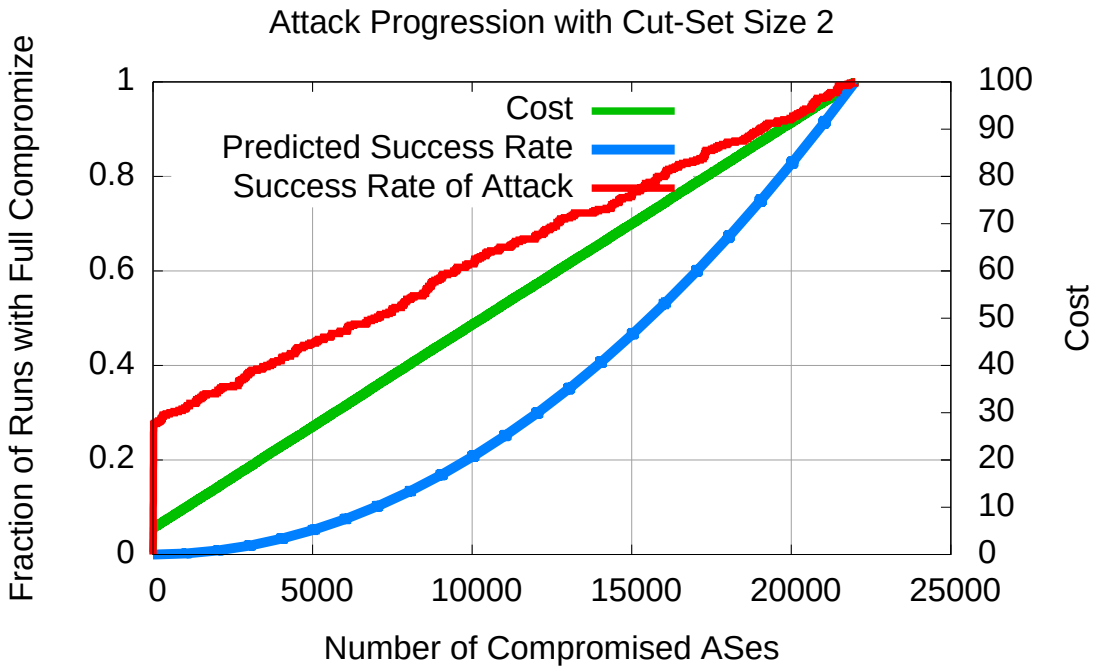

Figure 10: This figure shows that for a small $V_{cut} = 2$ the Targeted adversary has a better than predicted chance to spoof Alice.

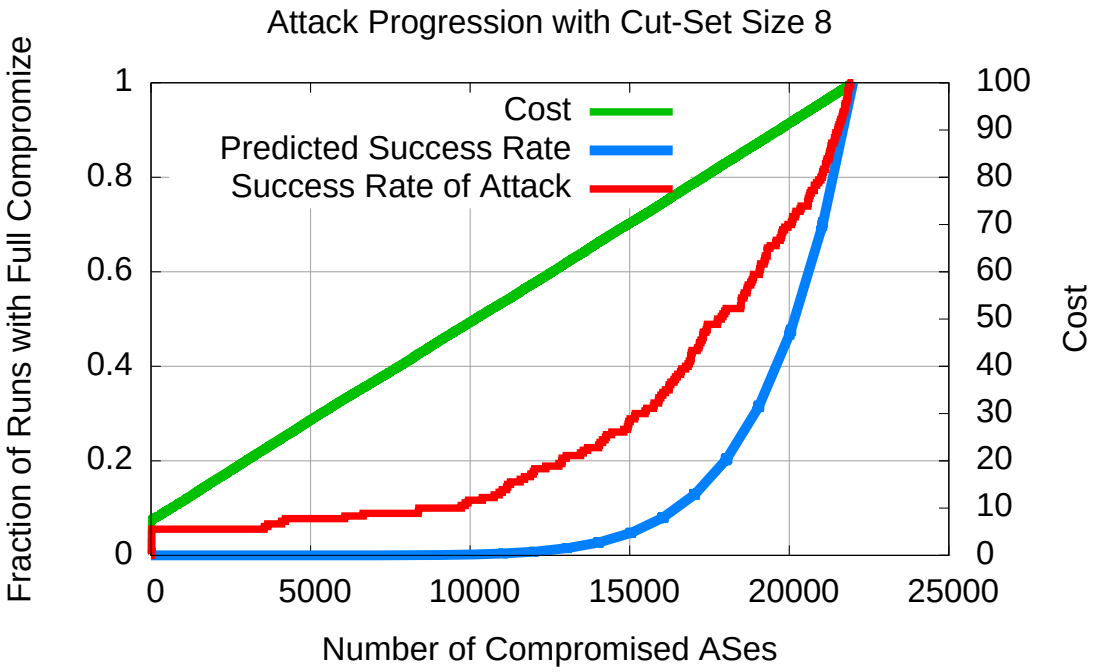

Figure 11: This figure shows that with a $V_{cut} = 8$, the adversary is already significantly less likely to be able to spoof Alice, though not quite as unlikely as predicted in the general case.

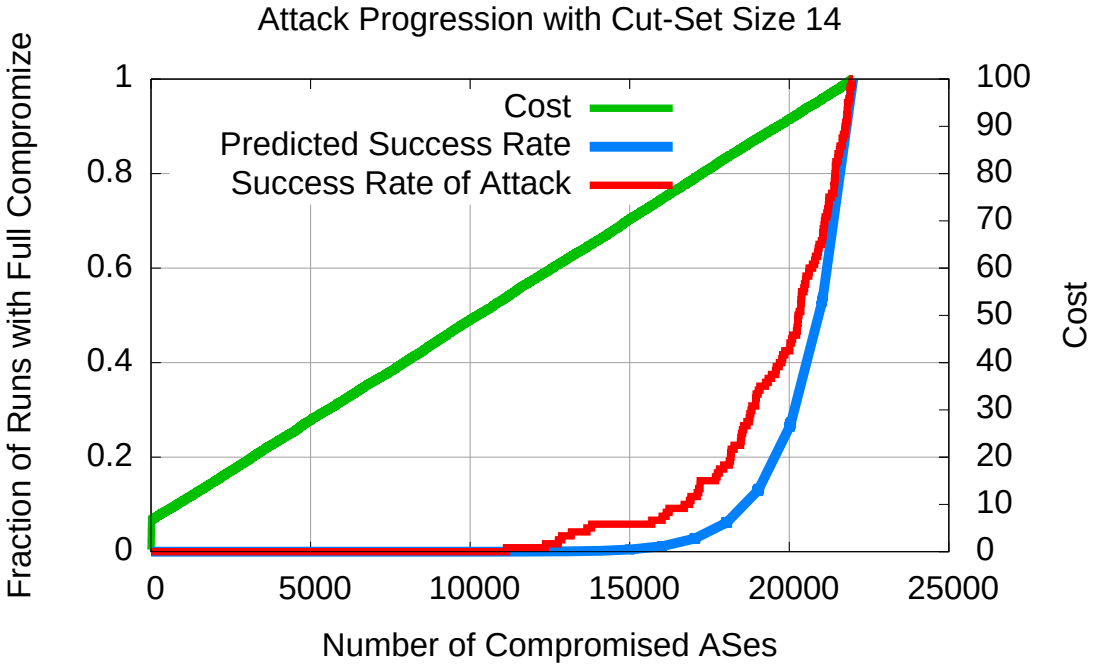

Figure 12: This figure shows that for a $V_{cut} = 14$, Alice's protection approaches the prediction of Equation 3.

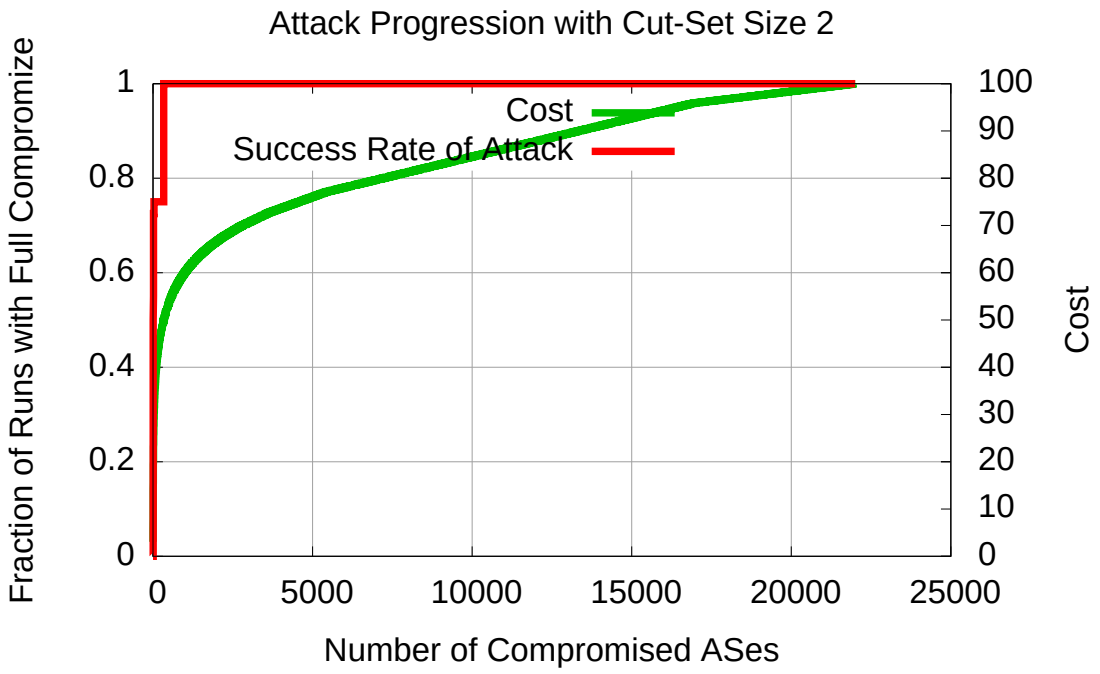

Figure 13: For a Rank-Order adversary, with a small $V_{cut} = 2$ Alice falls victim early in the attack.

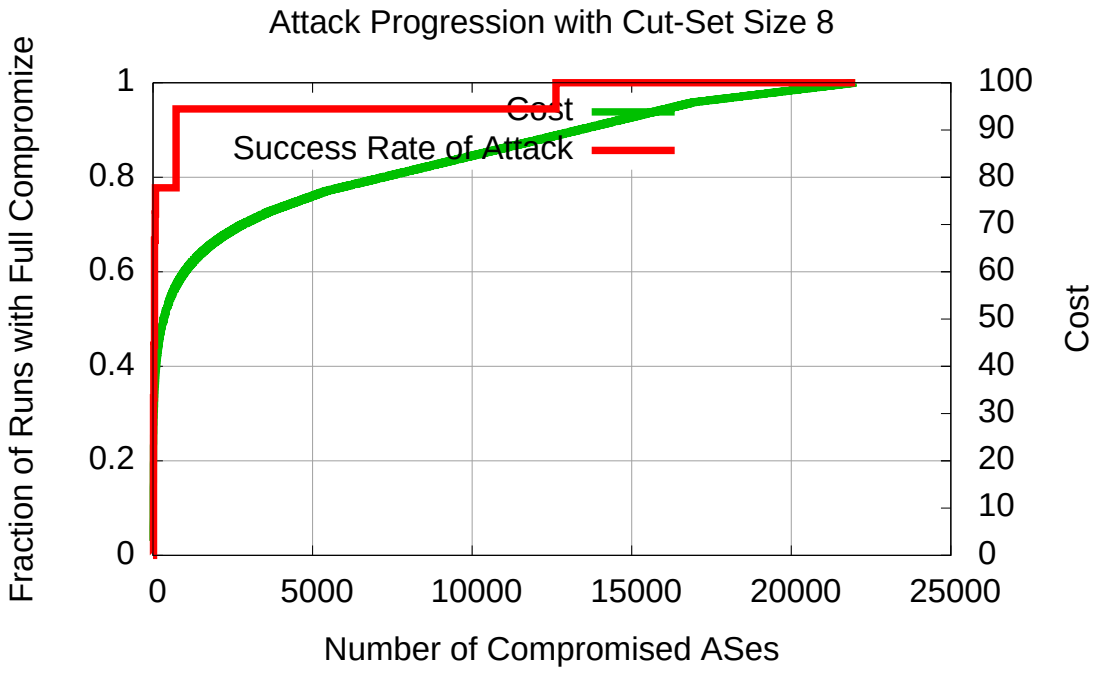

Figure 14: For a Rank-Order adversary, with a $V_{cut} = 8$ Alice is very likely to still fall victim to the attack early on.

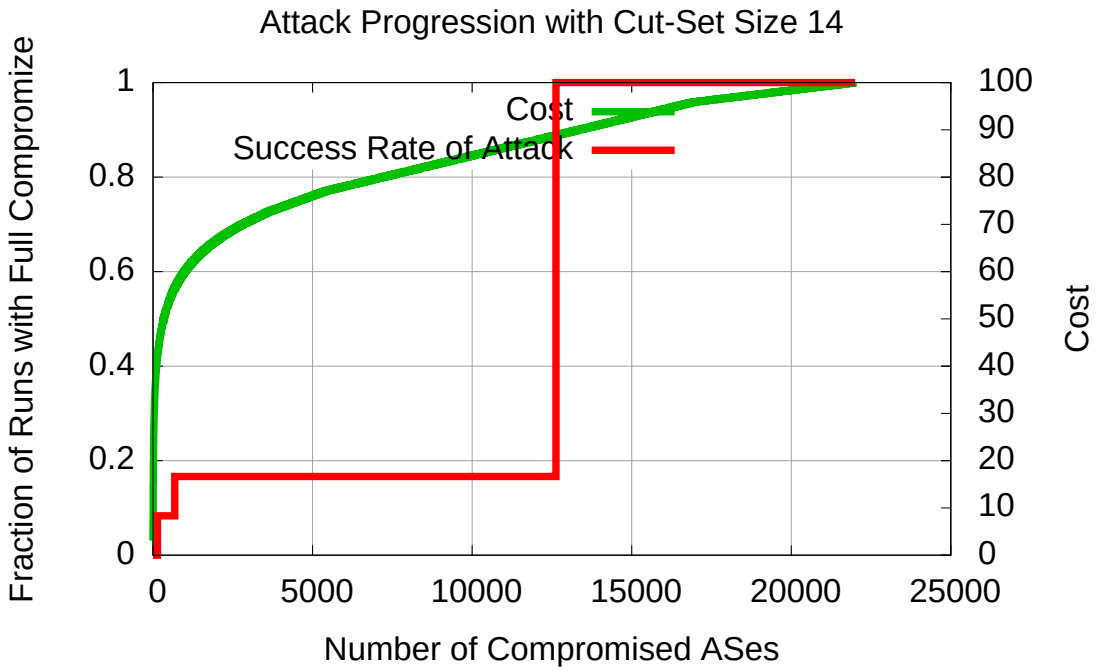

Figure 15: For a Rank-Order adversary, with a large $V_{cut} = 14$ Eve pays a huge cost and her success is staved off. This turns out to be because of the richness of connectivity at the edge of the network.

their accuracy using Internet-scale simulations.

Our findings are that users in this model can, indeed, defend themselves against adversaries. By proactively seeking diverse Communities of Trust in their deployments they can greatly reduce an adversary's chances to spoof them. Further, our general observation from these results is that the Internet's densely connected and distributed topology is a crucial benefit in the resilience of this approach. The high degree of connectivity at the edge of the network means that adversaries who may have compromised large hubs (like tier 1 ISPs) may not necessarily be able to subvert even moderate sized Communities of Trust. It is also noteworthy that in many Internet operational communities there exists growing anecdotal evidence that the Internet's edges are growing richer in this type of connectivity, suggesting that the performance of this approach may only become better in the future.

In the future we intend to augment this evaluation by adding in the usage cost and probability functions ($c_u()$ and $Pr_t()$) of our model. In addition, key rollover and revocation are both important issues. We believe that this model can fully support both of those operations by simply using forward verification signatures (where each key signs its successor). In the future we also plan to evolve this model to incorporate this explicitly. Finally, we have begun plans to deploy an actual `DNSKEY` verification system whose design is based on this model.

References

[1] SecSpider. `http://secspider.cs.ucla.edu/`.

[2] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. DNS Security Introduction and Requirement. RFC 4033, March 2005.

[3] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. Protocol Modifications for the DNS Security Extensions. RFC 4035, March 2005.

[4] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. Resource Records for the DNS Security Extensions. RFC 4034, March 2005.

[5] D. Atkins and D. Austein. Threat Analysis of the Domain Name System (DNS). RFC 3833, August 2004.

[6] Steven M. Bellovin. Using the domain name system for system break-ins. In *Proceedings of the Fifth Usenix Unix Security Symposium*, pages 199–208, 1995.

[7] Bradley Huffaker, Amogh Dhamdhere, Marina Fomenkov, and kc claffy. Toward Topology Dualism: Improving the Accuracy of AS Annotations for Routers. In *PAM 2010: Passive and Active Measurement Conference, Zurich*, 2010.

[8] C. Jin, Q. Chen, and S. Jamin. Inet topology generator. *Univ. Michigan, Ann Arbor, MI, Tech. Rep. CSE-TR-456-02*, 2000.

[9] E. Katz-Bassett, H.V. Madhyastha, V.K. Adhikari, A. Krishnamurthy, and T. Anderson. Practical reverse traceroute. NANOG, 2009.

[10] Ratul Mahajan, Neil Spring, David Wetherall, and Tom Anderson. Inferring link weights using end-to-end measurements. In *IMW '02: Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, pages 231–236, New York, NY, USA, 2002. ACM.

[11] Z. Morley Mao, Lili Qiu, Jia Wang, and Yin Zhang. On as-level path inference. In *SIGMETRICS '05: Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 339–349, New York, NY, USA, 2005. ACM.

[12] Zhuoqing Morley Mao, Jennifer Rexford, Jia Wang, and Randy H. Katz. Towards an accurate as-level traceroute tool. In *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 365–378, New York, NY, USA, 2003. ACM.

[13] J. Martin and R. Thomas. The underground economy: priceless. *USENIX; login*, 31(6), 2006.

[14] P. Mockapetris and K. J. Dunlap. Development of the domain name system. In *SIGCOMM '88*, pages 123–133, 1988.

[15] Rebecca Nielsen. Observations from the deployment of a large scale pki. In *4th Annual PKI R&D Workshop: Multiple Paths to Trust*, April 2005.

[16] E. Osterweil, D. Massey, and L. Zhang. Deploying and Monitoring DNS Security (DNSSEC). In *2009 Annual Computer Security Applications Conference*, pages 429–438. IEEE, 2009.

[17] Eric Osterweil, Dan Massey, and Lixia Zhang. Managing trusted keys in internet-scale systems. In *The First Workshop on Trust and Security in the Future Internet (FIST'09)*, 2009.

[18] Eric Osterweil, Daniel Massey, and Lixia Zhang. Observations from the DNSSEC Deployment. In *The 3rd workshop on Secure Network Protocols (NPSec)*, 2007.

[19] Eric Osterweil, Michael Ryan, Dan Massey, and Lixia Zhang. Quantifying the operational status of the dnssec deployment. In *IMC '08: Proceedings of the 8th ACM SIGCOMM conference on Internet measurement*. ACM, 2008.

[20] Eric Osterweil and Lixia Zhang. Interadministrative challenges in managing dnskeys. *IEEE Security and Privacy*, 7(5):44–51, 2009.

[21] R.L. Rivest and B. Lampson. SDSI–A simple distributed security infrastructure. Citeseer, 1996.

[22] Renata Teixeira, Keith Marzullo, Stefan Savage, and Geoffrey M. Voelker. Characterizing and measuring path diversity of internet topologies. In *SIGMETRICS '03: Proceedings of the 2003 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 304–305, New York, NY, USA, 2003. ACM.

[23] Dan Wendlandt, David Andersen, and Adrian Perrig. Perspectives: Improving SSH-style

host authentication with multi-path probing. In *Proc. USENIX Annual Technical Conference*, Boston, MA, June 2008.