# Authentication Challenges in a Global Environment

Stephanos Matsumoto, Carnegie Mellon University/ETH Zurich
Raphael M. Reischuk, ETH Zurich
Pawel Szalachowski, ETH Zurich
Tiffany Hyun-Jin Kim, HRL Laboratories
Adrian Perrig, ETH Zurich

We address the problem of scaling authentication for naming, routing, and end-entity certification to a global environment in which authentication policies and users' sets of trust roots vary widely. The current mechanisms for authenticating names (DNSSEC), routes (BGPSEC), and end-entity certificates (TLS) do not support a coexistence of authentication policies, affect the entire Internet when compromised, cannot update trust root information efficiently, and do not provide users with the ability to make flexible trust decisions. We propose a Scalable Authentication Infrastructure for Next-generation Trust (SAINT), which partitions the Internet into groups with common, local trust roots, and isolates the effects of a compromised trust root. SAINT requires groups with direct routing connections to cross-sign each other for authentication purposes, allowing diverse authentication policies while keeping all entities' authentication information globally discoverable. SAINT makes trust root management a central part of the network architecture, enabling trust root updates within seconds and allowing users to make flexible trust decisions. SAINT operates without a significant performance penalty and can be deployed alongside existing infrastructures.

**ACM Reference Format:**
Stephanos Matsumoto, Raphael M. Reischuk, Pawel Szalachowski, Tiffany Hyun-Jin Kim, and Adrian Perrig, 2016. Authentication Challenges in a Global Environment. *ACM Trans. Info. Syst. Sec.* V, N, Article XXXX (November 2016), 30 pages.
DOI: http://dx.doi.org/10.1145/0000000.0000000

## 1. INTRODUCTION

Alice lives in the Republic of Mythuania and frequently travels around the world for business purposes. In her business dealings, she frequently communicates with clients and banks located all over the world. For the security of these communications, Alice primarily relies on three mechanisms: DNSSEC to authenticate name-to-address mappings in DNS records [Arends et al. 2005], RPKI and BGPSEC to authenticate routes used to reach the sites [Lepinski and Kent 2012; Lepinski 2013], and TLS to authenticate the public keys used to establish a confidential connection to the sites [Dierks and Rescorla 2008]. The authenticity of information is anchored in each mechanisms's respective *trust roots*, and each mechanism follows one of two trust root models: *monopoly* (the entire system has a single or a small number of trust roots) and *oligopoly* (the system has a multitude of trust roots of equal authority).

Unfortunately, both models suffer from shortcomings that can undermine the authenticity of public keys in the PKI. The monopoly model unrealistically expects users to anchor their trust in a single global entity, such as ICANN for DNSSEC or IANA for RPKI/BGPSEC, effectively forcing Alice to trust the global root to authenticate information. Alice and Bob may differ in whom they trust, but under the monopoly model neither one has a choice of trust roots and must use the single global root. Moreover, the global root is a single point of failure in the monopoly model. Though for example the DNSSEC root zone key is well-secured [Dillow 2010], state-level attackers may have the capacity to compromise these keys if they wish [Borger 2013; Gellman and Poitras 2013; Weston et al. 2013].

The oligopoly model, on the other hand, gives all trust roots equal and global authority, allowing every trust root to issue a certificate for any entity in the Internet. For example, root certificate authorities (CAs) in TLS can issue a certificate for any domain name and can even delegate this global authority to their child CAs. This unfettered global power of root CAs highlights the lack of *scoped authority*:[1] authorities such as these CAs have unrestricted control over who they can

[1] In this paper, we focus on the properties highlighted in orange.

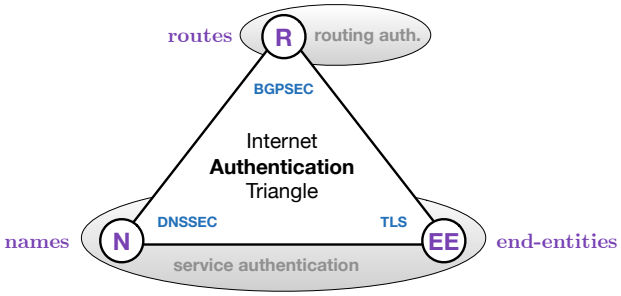


Fig. 1. Authentication triangle for routes, names, and end-entities.

certify. Therefore, in the oligopoly model, any compromised trust root can affect authentication for any entity in the world, leading to *weakest-link security*.

In an oligopoly system with many trust roots, Alice may have difficulties evaluating the trustworthiness of the many possible trust roots and may not even know which entities she is ultimately trusting. In other words, Alice lacks *trust root transparency*. Moreover, her choice of trust roots in a system such as TLS can prevent her from verifying the certificates of sites not certified by her trust roots. She therefore lacks two crucial properties: *trust agility*, which allows Alice to easily choose and change her trust roots [Marlinspike 2011], and *global authentication*, which allows Alice to find a chain of certificates to any reachable destination regardless of her choice of trust roots.

Finally, trust roots' public keys can change over time due to revocation, key rollover, or addition of new trust roots. In order to accurately evaluate her confidence in a chain of certificates, Alice needs to have access to the latest trust root information, particularly in light of a compromise. Thus Alice should be able to learn of any changes to her trust roots and obtain up-to-date information quickly and easily (i.e., *update efficiency*).

To address the above problems, we propose the Scalable Authentication Infrastructure for Next-generation Trust (SAINT), a new authentication architecture that addresses problems in DNSSEC, BGPSEC, and TLS. Using Alice's example, we demonstrate how authentication can be augmented to provide stronger guarantees and additional properties over today's authentication mechanisms. In designing SAINT, we make the following contributions:

— We propose a holistic approach for scoping trust root authority, dividing the Internet into groups sharing a common set of trust roots for routing, naming, and end-entity certification as shown in Figure 1. We do this by extending isolation domains (ISDs) from SCION [Zhang et al. 2011; Barrera et al. 2015] to include scoped trust roots for naming and end-entity certification. The configuration of ISDs allows Alice to select an ISD as her set of trust roots, enabling *trust agility*, and restricting trust roots to a single ISD provides *scoped authority*.
— We design **trust root configuration (TRC) files**, which contain trust root information. The distribution channel of these files is the same as that of routing messages, providing *update efficiency*. Additionally, these files allow Alice to quickly obtain and use new trust root information, enabling *trust agility*.
— We define **cross-signing** mechanisms of trust roots between ISDs sharing a routing link using special certificates. This mechanism enables *global authentication*, so that Alice can verify any entity to which she has a routing path. The certificates used for cross-signing also allow Alice to see the ISDs (and hence the set of trust roots) through which she is authenticating her destination, a property we call *trust root transparency*.
— We separate authentication for **routing** and **service** entities, preventing circular dependencies in authenticating routes. This separation also enables *trust agility* and *global authentication*, allowing Alice's trust decisions for service entities to apply anywhere in the world.

## 2. BACKGROUND AND RELATED WORK

In this section, we provide a brief overview of authentication infrastructures, including existing work in authenticating naming, routing, and end-entity certification and how they relate to the challenges presented in Section 1. For each aspect of authentication, we also include a list of the shortcomings of previous work in this area. We also describe the SCION architecture in detail (see Section 2.5), as SAINT relies on SCION mechanisms and extends its authentication framework.

### 2.1. Naming

Because end users often use domain names as the primary way to identify sites, it is important to ensure the authenticity of responses to domain name queries. **DNSSEC** [Arends et al. 2005] was created to authenticate DNS responses and thus to prevent cache poisoning and other attacks against DNS security. ICANN operates the DNSSEC root signing key, which for security is split among seven people [Dillow 2010], five of which must authorize a new root key. The root key authenticates the public keys associated with `.com`, `.org`, etc. In turn, these keys authenticate the next level of the DNS hierarchy. Clients can authenticate the DNS responses by starting from the root key and validating step-by-step the entire signature chain. **Self-certifying names** [Mazieres et al. 1999; Moskowitz et al. 2008; Andersen et al. 2008] provide another approach to improving the security of naming, deriving the name of an entity from its public key. Self-certifying names provide the advantage that they do not require an external PKI.

*Shortcomings.* In DNSSEC, everyone must trust the root key, which is controlled by ICANN, and represents a single point of failure in the system. In addition, updating the root key is a slow process, as five people from around the world must gather to update the key, and some DNSSEC-enabled devices have the current root key hardcoded into them and thus must be updated each time the root key is updated. Self-certifying names avoid this problem, but often involve another level of indirection to provide human-readable names, and require a name change each time an entity changes its public key.

### 2.2. Routing

In order to deliver packets to the correct destination without allowing large-scale interception of traffic, it is important to ensure the authenticity of addresses and routes. Together, **RPKI** [Lepinski and Kent 2012] and **BGPSEC** [Lepinski 2013] provide the means to authenticate addresses and AS numbers (RPKI) and routes (BGPSEC), protecting both against spoofing or unauthorized modification.

In RPKI, each Regional Internet Registry (RIR) serves as a trust anchor and signs certificates corresponding to resources, such as Autonomous System (AS) numbers and IP addresses, issued by that regional registry. For example, ARIN signs a delegation for an address space provided to AT&T, which in turn signs a delegation to a customer of its subspace. The same process occurs with AS numbers. Verifiers use the trust anchor managed by each RIR to verify the delegation chain of certificates for AS numbers and address spaces. Before the owner of an address block advertises a prefix in BGPSEC, it must use the address block certificate to sign a Route Origin Authorization (ROA) to an initial AS. Each AS on the path adds a signature of its own and the following AS number, called a route attestation in S-BGP [Kent et al. 2000]. The route attestations, together with AS number and address block certificates, enable validation of the path in BGPSEC.

The idea of aggregating hosts and routers into an abstracted routing entity has been proposed by others. The Nimrod routing architecture [Castineyra et al. 1996] describes a hierarchy of "clusters" of hosts, routers, or networks that can reach each other via a path contained within the cluster. FARA [Clark et al. 2003] generalizes the notion of an "entity" to also include clusters of computers that can be reached as a network communication endpoint.

*Shortcomings.* RPKI's validation process in BGPSEC suffers from circular dependencies. To transfer routing information, BGPSEC peers use the UPDATE message which contains signatures. The certificate chains for the signature validation are stored at each RIR's RPKI server. Hence, vali-

dation of the UPDATE message requires each BGPSEC router to fetch the certificates directly from the RIRs or from its local server, resulting in slow update propagation [Cooper et al. 2013].

### 2.3. End-Entity Certification

To ensure the authenticity and confidentiality of end-to-end communication between a client and a server, it is necessary for a server to authenticate its identity (i.e., its public key) to a client. **SSL/TLS** [Dierks and Rescorla 2008] secures web connections between browsers and web servers. A web site is authenticated via an X.509 [Cooper et al. 2008] certificate that the web site obtains from a Certification Authority (CA). Each browser relies on a set of root public keys, either its own or one provided by the operating system. The CAs controlling these keys, called the root CAs, can issue certificates granting CA powers, as well as issue certificates to end-entities such as websites.

There have been several proposal aiming to limit the scope of domains that CAs can certify. For example, CAge [Kasten et al. 2013] proposes to restrict CAs to signing domains in a small number of TLDs and treat other certificates as suspicious, and the US government has recently considered this proposal [Upton et al. 2015]. Abadi et al. suggest a policy engine to empower clients or ISPs to specify acceptance criteria for certificates [Abadi et al. 2013].

There have also been efforts to publicize public key via alternative mechanisms. DANE [Hoffman and Schlyter 2012] leverages the DNSSEC infrastructure to authenticate TLS public keys. Its goals are to tie TLS public keys to DNS names, use DNS to distribute these public keys, and to leverage the hierarchical authentication structure of DNSSEC to restrict the scope of CAs' authority. Certificate Transparency (CT) [Laurie et al. 2013] and the Accountable Key Infrastructure (AKI) [Kim et al. 2013] expose all CA operations to the public via auditable log servers, and modify clients to only accept certificates that have been publicized through such logs. Further work in this area includes CIRT [Ryan 2014], which provides more efficient log proofs and leverages these for an end-to-end mail service, PoliCert [Szalachowski et al. 2014], which decouples AKI policies from certificates, and ARPKI [Basin et al. 2014], which provides formally-proven security guarantees for AKI. All of these log-based approaches rely on gossiping protocols [Chuat et al. 2015] to disseminate misbehavior.

*Shortcomings.* Numerous security issues exist with the standard TLS CA ecosystem. Current browsers trust around 650 organizations [Eckersley and Burns 2010], several hundred of which are root certificates. CAs have global jurisdiction and delegate this power to intermediate CAs; consequently, any compromised CA can issue a fake certificate for any site in the entire Internet. Recent attacks on CAs have underscored the fact that even the most widely-used CAs suffer from such vulnerabilities, leading to Man-in-the-Middle (MitM) attacks on high-profile sites [Matsumoto and Reischuk 2015]. DANE relies on the security of DNSSEC, and while DNSSEC has notably avoided major compromises, a compromised DNSSEC key can be used to specify arbitrary trust anchors or bypass X.509 certificate validation, making a site's DNS nameserver a single point of failure for authentication. Log-based PKIs place the burden of scoping CA authority on individual domains rather than scoping CA authority by design.

### 2.4. Authentication Infrastructure Design

Other work has addressed problems that are not specific to any one of naming, routing, or end-entity certification. For example, IPA [Li et al. 2011] focuses on incremental deployment in the current Internet and leverages DNSSEC as a lightweight PKI to enable host authentication. IPA distributes AS certificates via S-BGP routing update messages, avoiding circular dependencies.

Previous work has also addressed authentication in a distributed, large scale network without any global trust infrastructure. For example, one proposal uses an authenticated path through the name space to make explicit trust relationships among entities [Birrell et al. 1986], and another describes an authentication theory based on the name space or the communication channel from which the other entity's authority can be deduced [Lampson et al. 1991]. There has also been work in this space to define a policy for inter-realm authentication trust based on trust hierarchies that can support transparent name authentication [Gligor et al. 1992].

Within the space of broadcasting messages such as key updates through a PKI, there has been work proposed techniques for using multiple MAC keys to authenticate both single-source multicast and group broadcast communications [Canetti et al. 1999]. Within a distributed infrastructure providing isolated realms of communication, these techniques could be used to update important trust root information.

*Shortcomings.* IPA relies on a single global root of trust, which creates a single point of failure and does not offer a choice of trust roots. Work in distributed authentication does not address how authentication across domains should work, except that all domains should certify one another's public keys. This approach does not take into account the different authentication policies of domains and cannot scale to large numbers of domains. The techniques by Canetti et al. require the storage of many keys; given the expected size and frequency of trust root updates, the memory requirements outweighs the savings in computational and bandwidth overhead.

### 2.5. SCION Architecture

**SCION** [Zhang et al. 2011; Barrera et al. 2015] is an isolation architecture for inter-domain routing in the Internet. SCION allows so-called **isolation domains (ISDs)** to distinguish between connections originating from inside or outside the domain, and can guarantee that the path of communication between two entities in a domain remains completely in that domain. ISDs are formed from ASes that are naturally grouped along jurisdictional boundaries and can agree on common roots of trust for routing information. These boundaries protect misbehavior in one ISD from affecting routing in another ISD. Each ISD is administered by a *core* of multiple tier-1 ASes (called *core ASes*).

SCION introduces path discovery messages called *Path Construction Beacons (PCBs)*, which are used in constructing both intra-ISD and inter-ISD paths. Core ASes periodically announce and disseminate PCBs to other ASes. Each AS in turn selects a subset of these PCBs according to its local policies to forward to downstream ASes (who then repeat this process). ASes also register the selected PCBs as path segments. These path segments can then be assembled to form an end-to-end path. PCBs and path segments convey AS-level path information (i.e., the traversed path), and are cryptographically protected by ASes' private keys. The intra-ISD dissemination follows customer-provider relationships, while inter-ISD dissemination is flood-based (among core ASes only).

ASes in SCION maintain the following control-plane infrastructure:

***Beacon Servers***. Beacon servers are responsible for disseminating and discovering path information in the network. A core AS's beacon server periodically announces PCBs and disseminates them to the AS's customers (to explore intra-ISD paths) or among neighboring core ASes (to explore inter-ISD paths). Non-core beacon servers receive PCBs from their providers, select the most desirable PCBs according to local policy, disseminate these PCBs downstream to their customers, and register them as path segments at local and core path servers (making the ASes accessible to local and remote end hosts).

***Path Servers***. Path servers maintain a database of registered path segments and make these segments accessible to end hosts. In essence, path servers form a caching system that stores mappings between ASes and path segments to those ASes. Path servers are queried by end hosts to deliver a set of path segments to desired destinations.

***Certificate Servers***. Certificate servers manage and distribute cryptographic key material and certificates within their ASes. In particular, certificate servers of core ASes keep a mapping between all issued *AS identifiers* (ASIDs) and ASes' certificates from their corresponding ISDs.

An end-host address in SCION is a 3-tuple of the form $(\mathcal{I}, A, e)$, where $\mathcal{I}$ represents an ISD identifier, $A$ represents an AS identifier, and $e$ represents an *end-host identifier* (EID). In contrast to the current Internet, AS identifiers and EIDs do not have significance outside of their respective ISDs and ASes, and thus can have any format. An EID, for example, can be an IPv4, IPv6, MAC, or self-certifying address.

## 3. PROBLEM DEFINITION

Our goal is to design a global authentication infrastructure enabling users to authenticate routes, names, and End-Entity (EE) certificates (such as TLS certificates) for servers in the network. We consider this problem in a global environment that has many trust roots operating in different jurisdictions, only some of which the client trusts. Furthermore, the client and server may not share any common trust roots. Each trust root helps authenticate a certain *type* of information: some are responsible for routing information, and others for EE certificates.

**Desired properties.** In order to effectively address the goals above, a network architecture should have the following properties:

— ***Global authentication.*** Any client that can reach a server can obtain and verify a chain of valid certificates from the client trust root to the server's root, name or EE certificates, regardless of the client's choice of trust root or location in the network.
— ***Scoped authority.*** Trust roots should be limited in scope, so that a compromise of a trust root does not affect any entity outside of that scope. In particular, if a client and server share trust roots for some information, no other trust root should be able to affect authentication of that information.
— ***Trust agility.*** All clients should have a choice over their trust roots that is easily modifiable at any time, with changes taking effect immediately (within seconds).
— ***Update efficiency.*** Changes to trust root information (e.g., new keys and revocations) should take effect quickly (within minutes). In particular, clients should be able to *automatically* detect and obtain new trust root information, without installing a browser/OS update or manually configuring new trust roots.
— ***Trust root transparency.*** Clients should know when trust roots other than their own are certifying information that they verify. In particular, for a chain of signatures, clients should be able to determine which trust roots are responsible for each signature.

In Section 9, we formalize these properties and show how SAINT provides each one.

**Network assumptions.** We assume SCION as the underlying network architecture. SAINT relies on SCION's infrastructure and mechanisms, and extends its authentication framework.

**Adversary model.** Our adversary is an individual or organization whose goal is to convince clients of false information for a route, name, or EE certificate. To achieve this goal, the adversary can actively suppress, change, replay, or inject messages into client-server communication, and might also gain access to the private keys of trust roots in one domain. However, besides these capabilities, the adversary cannot break cryptographic primitives such as public-key encryption and hash functions.

**Other assumptions.** We assume loose time synchronization (i.e., to within a few minutes) in the network. In order for clients to successfully verify authentication information, they must also be able to verify an initial set of trusted public keys (possibly through an out-of-band mechanism) which can then be used to bootstrap trust in other keys used during authentication.

## 4. SAINT OVERVIEW

In this section, we highlight important features of SAINT. Returning to the example of Alice and Bob from Section 1, we provide intuitive explanations of how these features accomplish the desired properties mentioned in Section 3 and how they fit into the overall SAINT architecture.

### 4.1. Leveraging SCION's Isolation Domains (ISDs)

The Internet consists of a diverse assortment of groups, or domains, each with its own set of trusted parties and individual policies regarding routing, naming, and EE certification. We achieve *scoped authority* by making these differences an explicit part of the SAINT architecture. Using SCION's isolation domains, we group hosts, routers, and networks as shown in Figure 2 and leverage SCION's routing infrastructure to provide additional authentication mechanisms for naming and EE certification.

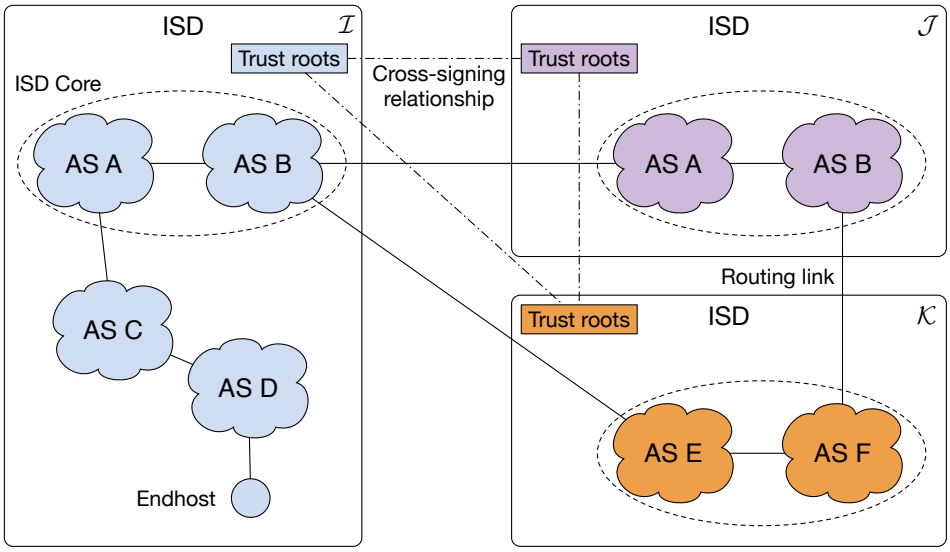


Fig. 2. A network of isolation domains, each with its own ISD core and trust roots. The ISDs connect with one another through links between core ASes, as well as through peering links (not shown). The cross-signing among the trust roots of the ISDs allow authentication across ISD boundaries. During deployment, we expect inter-ISD communication to take place on an IP overlay.

The trust roots of an ISD (see Figure 4) manage authentication within their ISD, including management of routing, naming, and EE certification policies, but are not authorities outside of the ISDs. The structure of ISDs attempts to capture existing trust relationships between humans by grouping those with similar trust decisions together and by protecting users from misconfigurations or breaches of trust outside these "circles of trust" where other trust decisions and policies hold. Thanks to the concept of ISDs, Alice (from the example in Section 1) can select her roots of trust. As long as there are no compromised entities on her authentication paths, Alice can securely authenticate servers. Multipath authentication can further reduce the effect of compromised entities.

In practice, ISDs can represent groups of various scales, such as companies, conglomerates, or countries. ISD-level policies will vary greatly by the scale of the ISD, since corporate policies often contain much more detail than country-wide laws. In this article, we use countries as examples of ISDs for several reasons: (1) international boundaries approximately map to DNS naming boundaries, which are also separated in SAINT, (2) national data privacy laws provide a reasonable example of security policies in top-level ISDs, and (3) the resulting set of domains represents an easily-understood choice among possible sets of trust roots, since users can more easily understand what it means to evaluate and trust a country (representing a set of trust roots) rather than doing so with individual trust roots.

### 4.2. Trust Root Configuration (TRC) Files

Trust root management in SAINT is handled by *TRC files*, which contain information about an ISD's trust roots, such as their public keys. TRC files can be fetched by a user to authenticate services in SAINT. They provide *update efficiency* by following the same channels as routing messages and DNS responses (see Section 5.3). Moreover, because routing messages are required to maintain connectivity, new TRC files can quickly propagate throughout the network in case a trust root is compromised. Finally, this mechanism allows Alice to quickly obtain up-to-date trust root information.

In addition to the above distribution mechanisms, TRC files can also be downloaded and chosen by users as a new trust root. Since a TRC file contains all of the necessary trust root information,

SAINT provides *trust agility* in that a user like Alice can easily (and at any time) switch to a different set of trust roots by simply obtaining and selecting a different TRC file.

TRC files contain trust root information for a given ISD and thereby enable *trust root transparency*. Namely, when a trust root signs the information of another ISD's trust root (as explained below), it does so by signing the TRC file of the other ISD. Thus a chain of signatures clearly indicates domain boundaries by design. Alice can use this knowledge of ISD boundaries to evaluate the trustworthiness of this signature chain and determine whether or not to accept the authenticated information. Section 5.2 provides further details on TRC files.

### 4.3. Cross-Signing Trust Roots

In a global trust environment, it is unrealistic to expect that all ISDs will cross-certify one another. Rather, we ensure *global authentication* by only requiring the trust roots of two ISDs to cross-certify one another if they share routing links, that is, if they are physically connected and route traffic through one another. This requirement ensures that the existence of a routing path implies the existence of a chain of signatures for a name or EE certificate, allowing Alice to verify this information for any entity she can reach. No matter where users are located, they can authenticate service information (names and EE certificates) starting from their own trust roots (named in their "home" TRC file) to the ISD of the entity whose information they are verifying. Thus as long as Alice can reach her home ISD from the ISD in which she is located, she can use her existing trust decision for authentication anywhere in the world. Section 5 provides more information on trust roots, and Section 6 provides more details on their cross-signing.

### 4.4. Separation of Authentication Types

SAINT separates routing authentication from service authentication (which certifies names and EE certificates). Because authentic routes are required to fetch necessary information during name lookups and EE certificate handshakes, we treat routing as a separate authentication mechanism. Moreover, we note that authentication of routes cannot rely on fetching external information, as this would itself require authentic routes and thus create a circular dependency.

The separation of routing and service authentication also helps to provide *global authentication* in SAINT. We observe that a user's physical locations indeed influences her routing authentication; in particular, a route from Alice to Bob must be authenticated by trust roots of the ISDs in which Alice and Bob are located. However, this requirement does not hold for service authentication; thus Alice can use the trust roots of an ISD of her choosing to completely bypass the ISD in which she is located to authenticate names or EE certificates, providing global authentication and greater resilience against MitM attacks.

### 4.5. DNS Namespace

Each ISD has the autonomy to manage its own namespace. We structure SAINT's global namespace as a collection of top-level domains (TLDs), as shown in Figure 3, rather than by a global root zone as is done in DNSSEC. However, SAINT's name resolution process is similar to that of DNSSEC.

Each SAINT DNS root server primarily answers queries for hosts within its ISD. An ISD's root servers support one of two top-level domain types:

— **Regional TLDs** in SAINT correspond to a specific ISD. In the example of Figure 3, the TLD `.us` represents the United States ISD and `.uk` represents the United Kingdom ISD. In order to provide transparency, the DNS server responsible for a regional TLD guarantees that any address record (similar to `A` records in DNS) maps to an address within the corresponding ISD.
— **Generic TLDs** (such as `.com` in today's Internet) are managed by a global ISD and can name an entity located *anywhere* in the world. However, a name in a generic TLD is implemented as a redirection to another name, thereby ensuring that only names under regional TLDs map to addresses (and only within the TLD's corresponding ISD). This guarantee provides domain

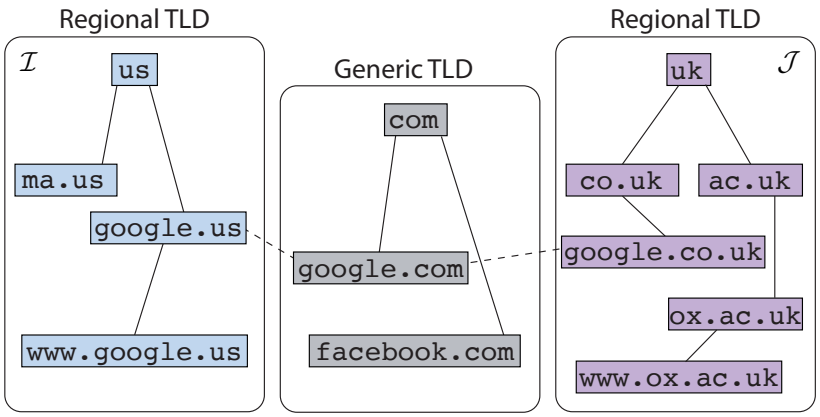


Fig. 3. Namespace structure in SAINT. Solid lines indicate hierarchical relations, and dashed lines indicate redirections.

transparency during DNS lookups. Details about name resolution for generic TLDs are in Appendix A.

We expect that today's ccTLDs such as `.us` and `.uk` will continue to operate as regional TLDs under SAINT. Countries such as Tuvalu (whose ccTLD is the popular `.tv`) may choose to operate as a generic TLD and continue to sell names that map all over the world, but must do so through redirections to other names.

### 4.6. Trust Anchor ISDs

Trust anchors in the current Internet, such as IANA for RPKI and BGPSEC, ICANN for DNSSEC, and root CAs in TLS, represent starting points for authenticating information. Similarly, trust anchor ISDs are starting points for authenticating routes, names, and EE certification in SAINT. When Alice performs a route lookup, name lookup, or EE certificate verification, she relies on the trust roots of her *trust anchor ISD* to authenticate the resulting route, name or certificate.

By default, Alice anchors all of her trust for authenticating both routing and service information in the ISD in which she is currently located. Due to the separation of authentication by type, however, Alice can benefit from *trust agility* and anchor her trust for authenticating routing and service information in separate ISDs if she chooses. As discussed in Section 4.4, for routing purposes the trust roots of the ISD's in which Alice is currently located must certify all of her routes. However, Alice can select the trust roots of any ISD to authenticate service information (names and EE certificates).

Alice thus has a trust anchor ISD for routing and for service authentication. By default, these are the same, meaning that a normal user of SAINT does not need to configure any trust anchor ISD to use SAINT. However, should Alice wish to select a different trust anchor ISD for service authentication (e.g., if she lives in a country whose service trust roots she does not trust), she can easily do so by obtaining a new TRC (see Section 5.3 and Section 8 for more details and an example).

## 5. TRUST ROOTS

In this section, we cover what entities serve as trust roots and how trust roots are configured for an ISD. We also discuss how we update trust root information using network-level messages, and how this incorporation of trust management into the network allows for fast updates (i.e., *update efficiency*) of trust root information. Finally, we describe our scheme of separating trust roots by ISD, and how separated categories of authentication enable *trust agility*.

### 5.1. Trusted Parties

Trust roots sign authentication information for routes, names, and EE certificates, and set policies governing the ISD. These policies may include information such as preferences for certain encryp-

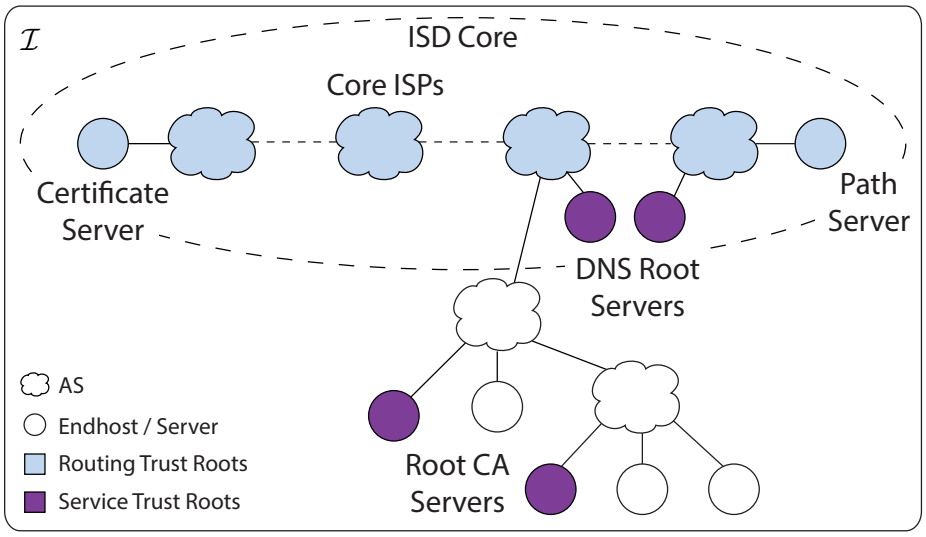


Fig. 4. Logical and physical placements of trust roots in an ISD.

tion and signature algorithms or constraints on certificate validity. As illustrated in Figure 4, a trust root is an authority for either routing or service authentication.

The **routing trust roots** consist of the following parties:

— The *core ISPs* are responsible for sending out route announcements, which are propagated from providers to customers and establish cryptographically signed down-path segments from the recipient to the core.

— The *path servers* store and provide a lookup service for mappings between an ASID and the AS's down-path segments. These path segments are registered by the ASes at the core. The path servers are co-located with and operated by the core ISPs.

— The *certificate servers* issue and store *AS certificates* binding an ASID to its public key (called its *AS key*), which are used to verify the signed paths provided by the path servers. Like the path servers, the certificate servers are co-located with and operated by the core ISPs.

The **service trust roots** consist of the trust roots for naming and for EE certification. The *DNS root* is the starting point for verifying all names in the ISD's namespace, and also sets ISD-wide naming policies such as reserved or forbidden domain names and signature algorithms to sign records. Because the failure of the DNS root can block user connectivity in an ISD, the DNS root should be highly robust and available, using mechanisms such as distributed anycast schemes and placing servers in the ISD core where they can be reached through highly available top-tier ASes.

The core ISPs and DNS root servers need to maintain high availability for all entities, even those outside the ISD, in order to ensure that the path lookup and DNS services function for sources both within and outside of the ISD. In order to enable efficient updating of the TRC file (see Section 5.2), all trust roots must have highly available channels of communication with each other (though no availability for entities outside the ISD is necessary). This allows them to quickly send messages and signatures to one another in the event that the TRC file must be updated.

The *root CAs* are the starting points for verifying EE certification information in an ISD. Root CAs in SAINT serve the same purpose as they do in today's PKIs by signing TLS public-key certificates. However, they are restricted to only signing EE certificates in ISDs in which they are root CAs. They can also sign intermediate CA certificates as in today's TLS PKI. If the ISD uses other public-key infrastructures such as CT or AKI (see Section 12), then the trust roots for EE certification also include trusted parties of those PKIs, such as public logs and auditors/validators.

### 5.2. Trust Root Configuration (TRC) Files

A TRC file provides *trust root transparency* by specifying the trust roots for an ISD, the public keys of those trust roots, and the authentication policies of the ISD. It also specifies the locations of the DNS root and TRC servers (described in Section 5.3) to allow users to reach these servers without performing DNS lookups. TRCs are created and managed by an ISD's trust roots and distributed through the routing mechanism. A threshold of trust roots is required to sign a new or updated TRC file, and the core ISPs distribute the TRC file within the ISD through a broadcast mechanism that we describe below.

The quorum of trust roots required to update the TRC file is specified in the TRC file itself, providing the trust roots with the autonomy to set their own threshold for altering the TRC file. A higher threshold is more secure to a compromise of multiple trust roots, but also reduces the efficiency in updating TRC files. The TRC file also specifies a quorum of trust roots that must sign a cross-signing certificate to authenticate another ISD's trust roots. Cross-signing certificates are described in more detail in Section 6.

**TRC format.** A TRC file is encoded as an XML file with the fields shown in Figure 5. The version number and timestamp ensure that users can verify information using recent policies and trust root information. The public keys of the ISD's trust roots provide starting points for verifying routes, names, and EE certificates. The TRC file may also contain the public keys of additional entities (e.g., public logs in CT [Laurie et al. 2013] or AKI [Kim et al. 2013]). In order to allow users to easily reach the DNS root of an ISD, the TRC also contains one or more addresses for the ISD's DNS root.

| Field | Description |
|---|---|
| `isd` | ISD identifier |
| `version` | version of TRC file |
| `time` | timestamp |
| `coreISPs` | list of core ISPs and their public keys |
| `certServKey` | root certificate server's public key |
| `pathKey` | path server's public key |
| `rootCAs` | list of root CAs and their public keys |
| `rootDNSkey` | DNS root's public key |
| `rootDNSaddr` | DNS root's address |
| `trcServer` | TRC server's address |
| `quorum` | number of trust roots that must sign new TRC |
| `trcQuorum` | number of trust roots that must sign an ISD cross-signing cert |
| `policies` | additional management policies for the ISD |
| `signatures` | signatures by a quorum of trust roots |

Fig. 5. Fields in a TRC file.

**Policies.** A TRC file can also specify additional policies related to ISD management. For example, these policies might specify a minimum key length or required encryption algorithms for all EE certificates in the ISD. Systems such as PoliCert [Szalachowski et al. 2014] have proposed similar policies on a per-domain basis; we leave a detailed design of additional ISD-wide policies to future work.

**Updating the TRC file.** To update the TRC file, the trust roots of the appropriate ISD must confer and agree on the changes to make to the TRC file. For example, if a trust root needs to be added or removed, or the quorum changed, the trust roots must reach a consensus on the proposed changes first. The trust roots can reach this consensus out of band or by using an in-band scheme such as PBFT [Castro and Liskov 1999], Paxos [Lamport 1998], or Raft [Ongaro and Ousterhout 2014]. However, in addition to reaching consensus via such a protocol, a quorum (whose size is at least `quorum` from the *old* TRC file) of the trust roots must also individually sign the new TRC

file to enable *external* authentication of the changes. Each of these signatures is appended to the `signatures` section of the new TRC file, and the new file is sent when a quorum of trust roots signs the file. The trust roots can also use group signatures [Chaum and Van Heyst 1991] or threshold signatures [Shoup 2000] to update the TRC file.

### 5.3. TRC Distribution and Management

In our running example, we envision that clients like Alice will most commonly obtain an initial TRC file of her provider's ISD when forming a service agreement. If Alice wants to obtain a different ISD's TRC file, she can contact the *TRC server* of that ISD, a server that stores the TRC files and cross-signing certificates (see Section 6) of other ISDs. The TRC server's address is in the TRC file of the ISD, allowing Alice to directly query the server for other TRC files. In an extreme case where Alice does not trust the provider or ISD, she may download a TRC file from a publicly-accessible mirror site or obtain one in person from a trusted colleague or organization. Alice can also obtain a TRC file *a priori* if she plans to join such an ISD with a new device.

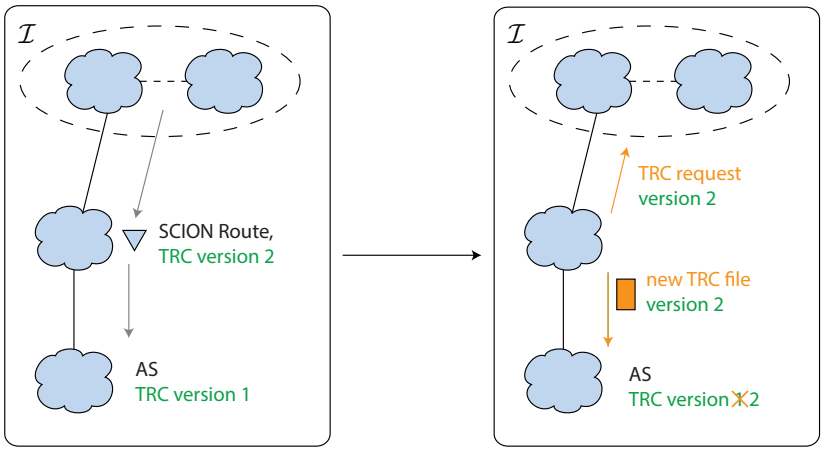


Fig. 6. Distribution mechanism for updated TRCs. Arrows indicate sent network messages.

**Obtaining updated TRC files.** ASes and users in an ISD are informed of the latest version of the TRC file with each routing announcement and DNS response. Thus, as long as Alice has an Internet connection and performs DNS lookups, she can quickly detect and obtain a TRC update, providing *update efficiency*. The version number is part of each routing announcement, and a timestamped message signed by the trust roots accompanies each DNS answer (to avoid re-signing every DNS record in the ISD upon updating the TRC file). When Alice detects a new TRC file, she can fetch the new file from the provider or DNS root (see Figure 6). Obtaining an updated TRC file does not invalidate existing cached routes, names, or EE certificates as long as the trust roots of the new version of the TRC file still certify the information. This property is due to the fact that the TRC file simply certifies the *trust roots* for routes, names, and EE certificates. Thus a new TRC version can be propagated to revoke information certified by misbehaving trust roots without disrupting other information that is still valid.

**Changing trust anchor ISDs.** Besides the ability to select trust roots as described in Section 4.6, trust agility also provides the ability to easily and quickly modify this selection. The above methods of obtaining TRC files provide this notion of *trust agility*, as Alice can change trust anchor ISDs by simply obtaining a new TRC file. Under normal circumstances, Alice can simply download a new TRC file from her trust anchor ISD's TRC server, but if for example she discovers that her

trust anchor ISD has been conducting state-level surveillance, she can instead obtain the TRC file manually or from an external server as described above.

## 6. CROSS-SIGNING

In this section, we use the example of Alice and Bob to provide more details on cross-signing in SAINT. We begin by describing cross-signing certificates, and then explain how these are used to enable inter-ISD authentication and in particular, *global authentication*. We then discuss the tradeoff between global authentication and the trustworthiness of authenticated information, and how the authentication policies expressed in an ISD's TRC file fits this tradeoff.

### 6.1. Cross-Signing Certificates

In order to enable global authentication, we require ISDs to issue a *cross-signing certificate* for each neighboring ISD, that is, ISDs to which they connect. As with TRC files, the trust roots of an ISD are responsible for issuing cross-signing certificates. The resulting web of cross-signing between ISDs ensures that by following a route from Alice's ISD to Bob's ISD, a corresponding chain of signatures from Alice's trust roots to Bob's trust roots will exist, forming a chain of signatures from Alice's trust roots to Bob and providing *trust root transparency*. ISDs without direct routing connections can also issue cross-signing certificates to one another, forming further chains of signatures to enable authentication between ISDs that do not share direct routing connections.

A cross-signing certificate issued by $\mathcal{I}$ for $\mathcal{J}$'s trust roots contains: (a) a timestamp, (b) the name of the issuer $\mathcal{I}$, (c) the current version number of $T_{\mathcal{I}}$, (d) the name of the receiver $\mathcal{J}$, (e) the current version number of $T_{\mathcal{J}}$, (f) a hash of $T_{\mathcal{J}}$, and (g) a signature by a quorum of $\mathcal{I}$'s trust roots. Figure 7 presents an explanation of the notation used. The version numbers of the TRC files ensure that the trust roots' public keys can be checked against the appropriate versions of the TRC files.

Each ISD stores these certificates in its TRC server for its users and also propagates the certificates along its inter-ISD routing links to provide each ISD with the necessary information to form a chain of signatures to a given destination ISD. Alice can then query her trust anchor ISD to obtain these chains of signatures and select one to authenticate information in Bob's ISD.

### 6.2. Inter-ISD Authentication

When Alice, whose trust anchor ISD is $\mathcal{K}$, wants to authenticate Bob, who is in another ISD $\mathcal{M}$, she needs to obtain cross-signing certificates to form a chain of signatures from $\mathcal{K}$ to $\mathcal{M}$. While verifying Bob's routes, name, and EE certificate, she obtains the appropriate cross-signing certificates from $\mathcal{K}$'s TRC server. If Alice is in an ISD $\mathcal{I}$, then every route from her to Bob will have a chain of signatures starting at $\mathcal{K}$, proceeding to the trust roots of $\mathcal{I}$, then to the trust roots of $\mathcal{M}$, and finally to Bob's AS.

If $\mathcal{K}$ and $\mathcal{M}$ do not share routing links but have issued cross-signing certificates to each other, Alice can verify Bob's name and EE certificate using $\mathcal{K}$'s cross-signing certificate for $\mathcal{M}$. These cross-signing "shortcuts" allow Alice to authenticate Bob's information with fewer ISDs authenticating information "in transit," providing fewer opportunities for a compromised trust root to disrupt authentication.

If Alice has routes from $\mathcal{I}$ to $\mathcal{K}$ and from $\mathcal{K}$ to $\mathcal{M}$, then she can find a route to Bob and thus a chain of signatures to Bob through the trust roots of $\mathcal{M}$. Since she must be able to contact $\mathcal{K}$ from $\mathcal{I}$ to obtain the appropriate cross-signing certificates, she has a route between $\mathcal{K}$ and $\mathcal{I}$ and can thus obtain cross-signing certificates from $\mathcal{K}$ to $\mathcal{I}$, and similarly for $\mathcal{I}$ and $\mathcal{M}$. Though a chain of signatures may cross many ISDs, Alice is guaranteed to find at least one such chain.

Note that cross-signing certificates do not necessarily indicate a trust relationship between ISDs; a cross-signing certificate instead only states: "These are the public keys of the trust roots for the following ISD." Alice can use trust root transparency to evaluate the trustworthiness of an ISD or chain of signatures, or she can trust an external organization such as the EFF to carry this evaluation out for her. While we leave a specific metric for evaluating the trustworthiness of a chain of cross-signing certificates to future work, we can use a number of factors, such as chain length or the

number of disjoint paths Alice has to Bob [Reiter and Stubblebine 1998]. Using disjoint paths is particularly useful, since Alice can choose to avoid specific ISDs in her authentication paths. Since cross-signing certificates are propagated along inter-ISD links, we anticipate that Alice will be able to obtain disjoint chains of cross-signing certificates to Bob regardless of her trust anchor ISD.

### 6.3. Authentication Policies

The above cross-signing requirement ensures that Alice can authenticate Bob's information regardless of which ISD he is in. While a compromised trust root on a chain of signatures from Alice to Bob can adversely affect authentication by certifying false information, Alice's trust anchor ISD $\mathcal{K}$ can mitigate this risk through the use of ISD-wide policies in the TRC file. These policies can also blacklist public keys, such as those contained in known unauthorized certificates or those of compromised trusted authorities. Using such policies, $\mathcal{K}$ can protect Alice from compromises in other ISDs. If others with $\mathcal{K}$ as their trust anchor ISD frequently contact Bob or other destinations in $\mathcal{M}$, then $\mathcal{K}$ may form a cross-signing relationship with $\mathcal{M}$ to minimize the risk of compromised trust roots in other ISDs.

ISDs face a tradeoff between enabling global authentication (providing signature chains that clients can accept) and protecting their users from compromises in other domains. The default behavior in SAINT is to provide global authentication, allowing the user or client to make the decision. As illustrated above, an ISD must explicitly state any exceptions to this behavior in the policy field of its TRC file. The ability to restrict the authentication of known false information through policies provides a mechanism by which an ISD can protect not only its own users, but also users for whom a chain of signatures passes through the ISD.

## 7. SEPARATION OF AUTHENTICATION INTO ROUTING AND SERVICE LAYER

In this section, we describe how SAINT separates routing and service authentication. We first describe our motivation for separating these two types of authentication into two *authentication layers*, and then discuss how this separation provides global authentication.

### 7.1. Routing and Service Authentication

Authentication in SAINT is classified and separated into the routing and the service layer (see Figure 1). We make this separation in part because we observe that the authentication of route information fundamentally differs from the authentication of service information. In particular, the routing layer cannot assume the existence of secure routes to obtain any external information, and therefore an entity must rely on pre-verified paths or be able to verify paths without fetching external information. By contrast, the service layer assumes the existence of authentic routes and thus allows contacting external entities to obtain authentication information.

Routing messages in SCION propagate beginning from the ISD core and follow provider-customer AS links. Unlike in RPKI and BGPSEC, all necessary information (e.g., AS certificates) are sent with the routing message, allowing an AS to verify routing messages directly upon arrival. Moreover, information such as AS certificates are short-lived, eliminating the need to propagate revocation information for AS keys.

By contrast, a DNS lookup, which falls under the service layer, must use a route to reach one or more nameservers and fetch the appropriate information for verifying name-to-address mappings. TLS may also require contacting an external entity to determine the validity of an EE certificate (e.g., an OCSP responder or CT [Laurie et al. 2013] log). Due to this dependence, Alice in our example must verify routes to the ISD core of her current ISD $\mathcal{I}$, and form and verify routing paths from $\mathcal{I}$ to Bob's ISD $\mathcal{M}$ before she can authenticate Bob's service information.

### 7.2. Global Authentication

Separating routing and service authentication also enables the *global authentication* property. Suppose that Alice checks into a hotel in Oceania, a known surveillance state, and attempts to connect to her hotel's wireless Internet. If the Oceanian trust roots are controlled by the government, then

| Symbol | Name | Use |
|---|---|---|
| **Identifiers** | | |
| $X$ | AS | AS with ASID $X$ |
| $y$ | Endhost | an end-entity such as a client or server |
| $e_y$ | EID | locate endhost $y$ within its AS and ISD |
| $\mathcal{Z}$ | ISD | ISD with identifier $\mathcal{Z}$ |
| **Certificates** | | |
| $AC_X$ | AS cert | bind $X$ to $AK_X$ (signed by $RS$ of $X$'s ISD) |
| $EC_y$ | End-entity cert | store CA-signed public key information during connection setup |
| $DC_y$ | CERT RR | store CA-signed DNS binding between $y$ and $DK_y$ |
| **Keys** | | |
| $AK_X$ | AS key | sign paths that can be used to reach $X$ |
| $DK_y$ | DNSKEY RR | sign DNS resource records in DNSSEC |
| $EK_y$ | End-entity key | set up secure end-to-end connections, e.g., via TLS |
| $K_y^{-1}$ | Private key | private key for public key $K_y$ |
| **Servers** | | |
| $PS_Y$ | AS Path server | contact ISD path server for clients in $Y$ |
| $PS_{\mathcal{Z}}$ | ISD Path server | maintain database of signed paths for ASes in $\mathcal{Z}$ |
| $CS_{\mathcal{Z}}$ | Certificate server | assign ASIDs and AS numbers in $\mathcal{Z}$ |
| **Messages** | | |
| $P_X$ | Signed path set | sent to $PS$ of $X$'s ISD to register paths to reach $X$ |
| $T_{\mathcal{Z}}$ | TRC file | provide trust root information for $\mathcal{Z}$ |

Fig. 7. Notation.

it is inevitable for Alice today that the government can examine all of her packets. In other words, Oceanian trust roots must certify her routes out of the Oceanian ISD and thus these trust roots must be on the chain of signatures for routes from Alice to any destination in the Internet.

With SAINT, however, Alice can choose $\mathcal{K}$ as her trust anchor ISD for service authentication, since SAINT separates routing and service authentication. Moreover, this choice does not depend on Alice's current location and thus applies wherever Alice is in the Internet. In our example, this means that Alice does not have to rely on signatures from the Oceanian trust roots to verify Bob's name or EE certificate, even if she is connecting to the Internet from an Oceanian hotel.

## 8. AUTHENTICATION EXAMPLE

We now discuss all steps carried out when a client, say Alice, is authenticating a server, say Bob, in SAINT. We first describe the setup steps for the server, such as joining an ISD and registering domain names, routing paths, and EE certificates. We then describe how client Alice checks the information that she receives about server Bob.

Figure 7 provides a list of the notation used. We use $a$ to denote Alice and $b$ to denote Bob. As previously mentioned, Alice's trust anchor ISD is $\mathcal{K}$. Bob is part of the AS $B$ in Mythuania $\mathcal{M}$, whose ccTLD is `.my`.

### 8.1. AS Setup

Figure 8 depicts the steps of the AS setup process for AS $B$ in the Mythuanian ISD $\mathcal{M}$:
(1) $CS_{\mathcal{M}}$ assigns the ASID $B$ to Bob's AS.
(2) $B$ creates an AS key pair $(AK_B, AK_B^{-1})$.
(3) $B$ sends $\{B, AK_B\}$ to $CS_{\mathcal{M}}$.
(4) $CS_{\mathcal{M}}$ issues $B$ an AS certificate $AC_B$.
(5) $B$ receives the TRC file $T_{\mathcal{M}}$ from its parent AS.
(6) $B$ receives SCION routing messages from its parent AS.
(7) $B$ selects a set of paths $P_B$ (signed with $AK_B^{-1}$) and sends $\{P_B, AC_B\}$ to $PS_{\mathcal{M}}$.

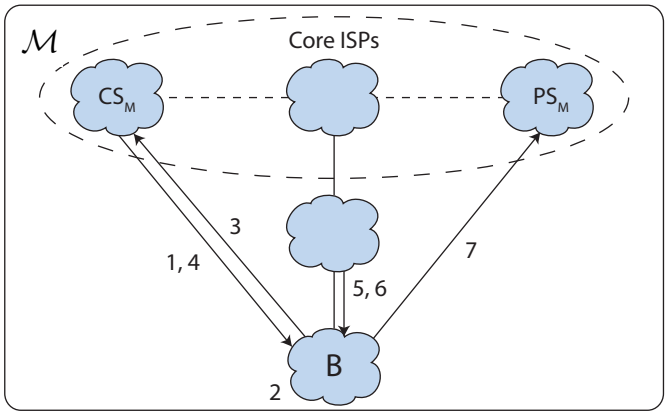


Fig. 8. Diagram for AS setup steps (Section 8.1).

### 8.2. Server Setup

Figure 9 shows the steps of the server setup process for Bob (as described below), Figure 10 shows the established relations after that setup.

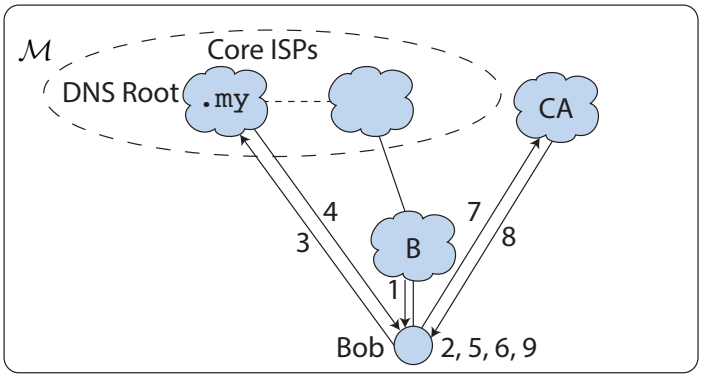


Fig. 9. Diagram for server setup steps (Section 8.2).

(1) AS $B$ assigns Bob the EID $e_b$, making his fully-specified SCION address $(\mathcal{M}, B, e_b)$.
(2) Bob chooses the name `b.my` and creates a domain-name key pair $(DK_b, DK_b^{-1})$.
(3) Bob sends `b.my` and $DK_b$ to the `.my` operator to register his name and key out-of-band.[2]
(4) The `.my` operator creates a delegation signer (DS) record to point to $DK_b$ from the `.my` zone, as well as a record mapping `b.my` to Bob's nameserver.
(5) On his nameserver, Bob creates a DNS public key (DNSKEY) record that encodes his domain-name public key $DK_b$, and a record that maps `www.b.my` to his address $(\mathcal{M}, B, e_b)$. These records are signed with the private key $DK_b^{-1}$, and the signatures are placed in the two corresponding resource record signature (RRSIG) records.

[2]In practice, Bob will create multiple key pairs and use one of the private keys to sign the others, but for simplicity we assume here that Bob uses $DK_b$ both to sign his DNS zone information and to self-sign $DK_b$.

(6) Bob creates an end-entity key pair $(EK_b, EK_b^{-1})$.
(7) Bob sends $\{\texttt{www.b.my}, EK_b\}$ to a CA in $\mathcal{M}$.
(8) The CA verifies[3] whether Bob is indeed an owner of `www.b.my` and issues Bob an EE certificate $EC_b = \{\texttt{www.b.my}, EK_b\}_{K_{CA}^{-1}}$.
(9) Bob creates a certificate $DC_b = \{\texttt{www.b.my}, DK_b\}_{EK_b^{-1}}$, and stores $DC_b$ along with $EC_b$ as a certificate (CERT) record in his nameserver.

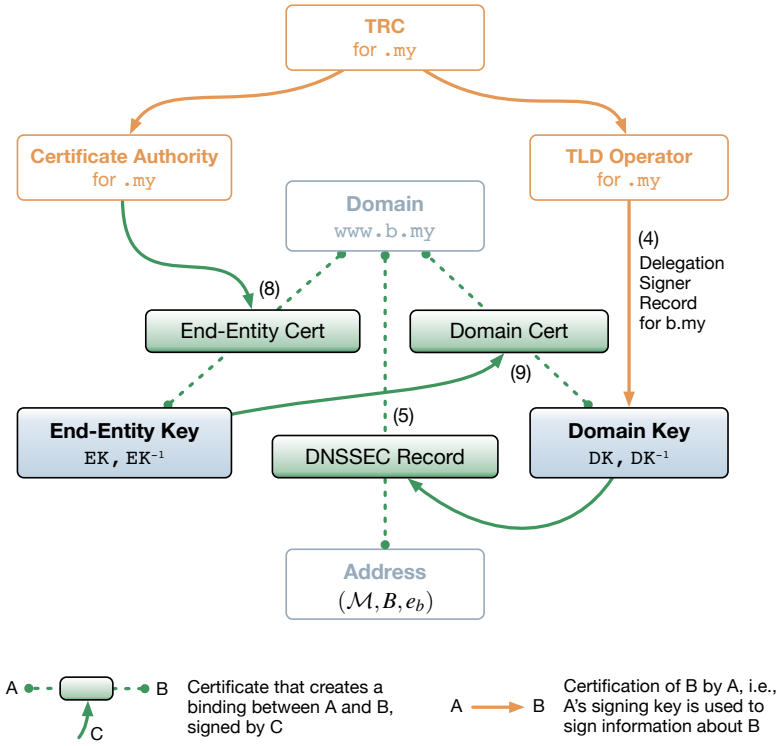


Fig. 10. Authentication dependencies.

Figure 10 shows the dependencies between keys, certificates, and authorities. All authentication starts at the TRC file for a given top-level domain.

### 8.3. Client Setup

In order for Alice to verify information, she must first possess a TRC file to configure her set of trust roots. Even if she does not have a TRC file, we assume that she can verify a TRC file from her trust anchor ISD $\mathcal{K}$. In fact, she can verify this TRC file in any ISD $\mathcal{I}$ — even if she does not trust $\mathcal{I}$. As illustrated in Figure 11, after connecting to the Internet in $\mathcal{I}$, Alice does the following to obtain and verify a TRC file:
(1) Alice's ISP (AS $C$) assigns her the EID $e_a$, making her address $(\mathcal{I}, C, e_a)$. $C$ also sends her the latest TRC file $T_{\mathcal{I}}$ for the ISD $\mathcal{I}$.

[3]The most popular ways of this verification are *domain validation*, where Bob's identity is validated by proving some control over his domain name; and *extended validation*, where Bob proves that he is a legal entity controlling his domain name.

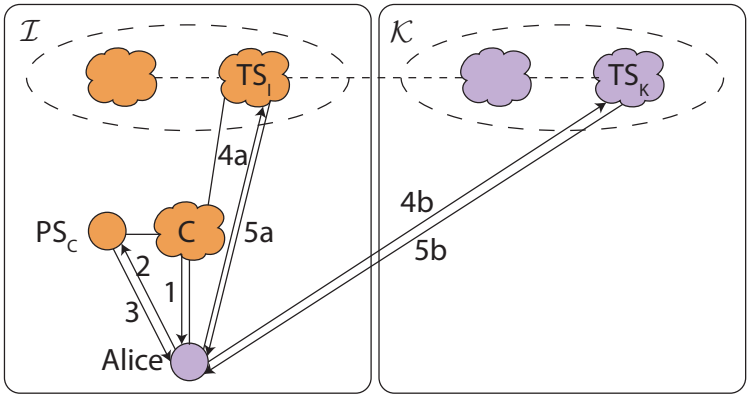


Fig. 11. Diagram for obtaining a TRC file (Section 8.3).

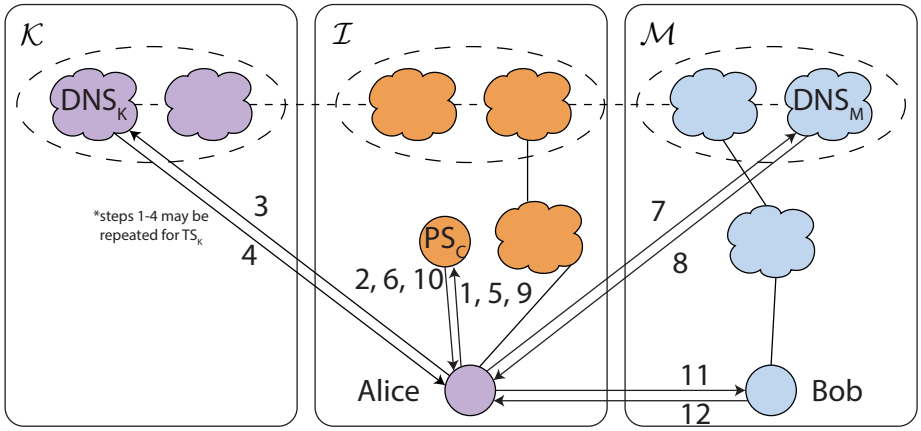


Fig. 12. Client lookup and verification of server's name, route, and EE certificate (Section 8.4).

(2) Alice requests from $PS_C$ a path to the TRC server $TS_{\mathcal{I}}$ of $\mathcal{I}$ or $TS_{\mathcal{K}}$ of $\mathcal{K}$ (if she knows the address).
(3) $PS_C$ returns to Alice the path she requested.
(4) Alice now contacts either $TS_{\mathcal{I}}$ (4a in Figure 11) or $TS_{\mathcal{K}}$ (4b) and requests $T_{\mathcal{K}}$. In the case of 4b, Alice also requests a cross-signing certificate for $\mathcal{I}$ from $TS_{\mathcal{K}}$ to ensure that all authentication (even for routes) begins from $T_{\mathcal{K}}$.
(5) If Alice contacted $TS_{\mathcal{I}}$, she receives $T_{\mathcal{K}}$ (5a). Otherwise, she receives $T_{\mathcal{K}}$ and the cross-signing certificate for $\mathcal{I}$ from $TS_{\mathcal{K}}$ (5b).

We assume that Alice verifies the authenticity of $T_{\mathcal{K}}$ through an out-of-band mechanism, e.g., if she makes plans to travel to $\mathcal{I}$ and considers it a "hostile" ISD, then Alice can obtain a hash of the public keys of $\mathcal{K}$'s trust roots *a priori*, or she can obtain this information in an embassy of $\mathcal{K}$ within $\mathcal{I}$.

### 8.4. Client Verification

Figure 12 illustrates the complete process that Alice executes to authenticate Bob. We assume that Alice has completed the client setup process and thus has $T_{\mathcal{K}}$ to use as the starting point for authenticating Bob. The authentication process is as follows:

(1) Alice begins contacting the local path server $PS_C$ to obtain a path to her DNS root $DNS_{\mathcal{K}}$.
(2) The local path server $PS_C$ returns to Alice a set of paths that she can use to reach $DNS_{\mathcal{K}}$.
(3) Alice contacts $DNS_{\mathcal{K}}$ to query `www.b.my`.
(4) $DNS_{\mathcal{K}}$ responds that `.my` is in $\mathcal{M}$. If necessary, Alice repeats the above steps to contact $TS_{\mathcal{K}}$ in order to obtain $T_{\mathcal{K}}$ and the cross-signing certificate from $\mathcal{K}$ to $\mathcal{M}$.
(5) Alice then contacts path server $PS_C$ to request a path to the DNS root $DNS_{\mathcal{M}}$ of $\mathcal{M}$, whose address she has from $T_{\mathcal{M}}$.
(6) $PS_C$ returns to Alice a set of paths to $\mathcal{M}$'s DNS root.
(7) Alice contacts $DNS_{\mathcal{M}}$ to query `www.b.my`.
(8) $DNS_{\mathcal{M}}$ provides Bob's address $(\mathcal{M}, B, e_b)$, Bob's domain certificate $DC_b$ (authenticating $DK_b$ used to verify the address), and his EE certificate $EC_b$ (authenticating $EK_b$ used to verify $DC_b$). Recall that these were established in Section 8.2.
(9) Alice requests a path to Bob's address $(\mathcal{M}, B, e_b)$ from $PS_C$.
(10) $PS_C$ returns to Alice Bob's AS certificate $AC_B$ and a set of paths $P_B$ to reach $B$.
(11) Alice contacts Bob to initiate the TLS handshake.
(12) Bob sends Alice the EE certificate $EC_b$, which binds `b.my` to the key $EK_b$. This key certifies the domain certificate $DC_b$, which binds `b.my` to the domain key $DK_b$, and can also be used during the TLS handshake.

Alice verifies that Bob's EE public key $EK_b$ contained in the EE certificate she obtained from $\mathcal{M}$'s DNS root matches the EE public key she receives during the TLS handshake. If the keys match, she proceeds with the TLS handshake to establish a secure end-to-end connection with Bob.

Throughout this process, Alice verifies that valid authentication paths exist for each entity she contacts: $PS_C$, $TS_{\mathcal{K}}$, $\mathcal{M}$'s DNS root, $B$, and Bob. When she receives information signed by the trust roots of an ISD other than $\mathcal{K}$, Alice uses the appropriate cross-signing certificate to verify the public keys of the ISD's trust roots, thus ensuring that all authentication ultimately begins with trust roots listed in the TRC of her trust anchor ISD $\mathcal{K}$.

**Error handling.** A verification failure at any stage in the authentication process will prevent Alice from authenticating and establishing a connection to Bob. In the event that the verification of a routing path fails, Alice will not be able to reach Bob or entities such as DNS roots and TRC servers. However, Alice likely cannot detect this failure from her browser. In the event that the verification of Bob's name-to-address mapping fails, Alice will not know the address at which she can reach Bob. While most modern browsers indicate such a failure, Alice cannot proceed with verification after such a failure. From the perspective of Alice's browser, a failure to verify Bob's EE certificate is the most informative, as most modern browsers display the type of error that occurred and in some cases provide the option to continue with the connection anyway.

## 9. ANALYSIS

In this section, we present a formal model of SAINT and analyze how our model achieves the desired properties presented in Section 3.

### 9.1. Model

We begin our model by defining an ISD and the global network in which we consider ISDs.

*Definition* 9.1. An *ISD* is an ordered triple $\mathcal{I} = (\mathsf{id}, \mathsf{TR}, \mathsf{M})$ where $\mathsf{id}$ is a string called the *identifier*, $\mathsf{TR}$ is a set of *trust root public keys*, and $\mathsf{M}$ is a set of *members* of the ISD. When dealing with multiple ISDs, we write the attributes of an ISD $\mathcal{I}$ as $\mathsf{id}_{\mathcal{I}}$, $\mathsf{TR}_{\mathcal{I}}$, and $\mathsf{M}_{\mathcal{I}}$, respectively.

Pairs of ISDs can share direct routing connections, i.e., if $\mathcal{I}$ and $\mathcal{J}$ share a direct routing connection, then there exists a router in $\mathcal{I}$ that can send a packet to a router in $\mathcal{J}$ such that no router in any other ISD logically handles the packet.

Pairs of ISDs can also issue cross-signing certificates to each other. For the purposes of our model, we represent a cross-signing certificate by the trust root public keys of an ISD, signed by the trust root private keys of another ISD. We formalize this definition as follows:

*Definition* 9.2. A *cross-signing certificate* issued by $\mathcal{I}$ for $\mathcal{J}$ is a 4-tuple $\mathsf{CSC}(\mathcal{I},\mathcal{J}) = (\mathsf{id}_{\mathcal{I}},\mathsf{id}_{\mathcal{J}},\mathsf{TR}_{\mathcal{J}},\sigma_{\mathcal{I}\mathcal{J}})$, where $\sigma_{\mathcal{I}\mathcal{J}}$ is the *signature* of the certificate, signed using $\mathcal{I}$'s signing keys.

We make several assumptions in the above definitions. Notably, we assume that routing connections and cross-signing relations are symmetric, i.e., whenever $\mathsf{CSC}(\mathcal{I},\mathcal{J})$ exists, then also $\mathsf{CSC}(\mathcal{J},\mathcal{I})$ has been issued. We also assume that a routing connection implies a cross-signing relation. With these considerations in mind, we define a global network as follows:

*Definition* 9.3. The *global network* is an ordered triple $(\mathscr{I},\mathscr{N},\mathscr{C})$ where $\mathscr{I}$ is a set of ISDs, $\mathscr{N}$ is a set of unordered ISD pairs, and $\mathscr{C}$ is a set of cross-signing certificates, such that if $(\mathcal{I},\mathcal{J}) \in \mathscr{N}$, then $\mathsf{CSC}(\mathcal{I},\mathcal{J}) \in \mathscr{C}$ and if $\mathsf{CSC}(\mathcal{I},\mathcal{J}) \in \mathscr{C}$, then $\mathsf{CSC}(\mathcal{J},\mathcal{I}) \in \mathscr{C}$. We call $\mathscr{I}$ the *ISD set* of the network, $\mathscr{N}$ the *neighbor set*, and $\mathscr{C}$ the *cross-signing set*.

We thus model the global network as a graph with two sets of edges representing routing connections and cross-signing relationships, respectively. The cross-signing relationships $\mathscr{C}$ are a superset of routing connections $\mathscr{N}$, as they may include cross-signing relationships that exist independently of an underlying routing connection. In the global network, we define paths of routing connections or cross-signing certificates as follows:

*Definition* 9.4. A *path* in a global network $(\mathscr{I},\mathscr{N},\mathscr{C})$ from $\mathcal{X}_1$ to $\mathcal{X}_n$ is a sequence of ISDs $(\mathcal{X}_1,\ldots,\mathcal{X}_n)$ where for all $1 \le i \le n$, $\mathcal{X}_i \in \mathscr{I}$ and for all $1 \le i < n$, $(\mathcal{X}_i,\mathcal{X}_{i+1}) \in \mathscr{N}$. We call $\mathcal{X}_1$ and $\mathcal{X}_n$ the *source* and *destination* of the path, respectively.

*Definition* 9.5. A *certificate chain* in a global network $(\mathscr{I},\mathscr{N},\mathscr{C})$ from $\mathcal{X}_1$ to $\mathcal{X}_n$ is a sequence of ISDs $\mathbf{C} = \langle \mathcal{X}_1,\ldots,\mathcal{X}_n\rangle$ where for all $1 \le i \le n$, $\mathcal{X}_i \in \mathscr{I}$ and for all $1 \le i < n$, $\mathsf{CSC}(\mathcal{X}_i,\mathcal{X}_{i+1}) \in \mathscr{C}$. Similarly to Definition 9.4, a certificate chain also has a source and destination.

We now define clients, who reside in ISDs, as well as evaluation functions, used by clients to determine the validity of a certificate chain.

*Definition* 9.6. A *client* is a triple $A = (\mathcal{R},\mathcal{T},\mathsf{eval})$ where $\mathcal{R}$ is an ISD called the *routing source ISD*, $\mathcal{T}$ is an ISD called the *trust root ISD*, and $\mathsf{eval} : \{\mathbf{C}\} \to \{0,1\}$ is an *evaluation function* used to evaluate a certificate chain.

Intuitively, eval represents the method used by $A$ to determine whether to trust a given certificate chain or not. Trusting a chain means behaving as if a route, name, or EE certificate signed by the destination ISD's trust roots is correct. Thus a client $A$ who accepts a certificate chain with destination $\mathcal{D}$ will, for example, negotiate a session key with a destination $B$ located in $\mathcal{D}$ to communicate confidentially with $B$. We elaborate on the evaluation function with the following definition:

*Definition* 9.7. Let eval be the evaluation function of a client $A = (\mathcal{R},\mathcal{T},\mathsf{eval})$. We call eval *valid* if for a certificate chain $\mathbf{C} = \langle \mathcal{X}_1,\ldots,\mathcal{X}_n\rangle$, we have $\mathsf{eval}(\mathbf{C}) = 1$ when $\mathcal{X}_1 = \mathcal{T}$ and for all $1 \le i < n$, $\mathsf{Vrfy}_{K_{\mathcal{X}_i}}\left(\mathsf{TR}_{\mathcal{X}_{i+1}},\sigma_{\mathcal{X}_i\mathcal{X}_{i+1}}\right) = 1$, where $\mathsf{Vrfy}_K(m,\sigma)$ denotes verification of message $m$ and signature $\sigma$ with public key $K$.

We now state two lemmas that will be used in our analysis of SAINT's desired properties. The proofs are straightforward and thus left to the reader.

LEMMA 9.8. *In any global network $(\mathscr{I},\mathscr{N},\mathscr{C})$, the existence of a path $(\mathcal{X}_1,\ldots,\mathcal{X}_n)$ implies the existence of a certificate chain $\langle \mathcal{X}_1,\ldots,\mathcal{X}_n\rangle$.*

LEMMA 9.9. *Paths are closed under the substring operation, symmetric, and transitive. That is, in a global network $(\mathscr{I},\mathscr{N},\mathscr{C})$, the following statements hold for all paths $(\mathcal{X}_1,\ldots,\mathcal{X}_m)$ and $(\mathcal{Y}_1,\ldots,\mathcal{Y}_n)$:*

(1) *for all i and j where* $1 \leq i \leq j \leq m$, $(\mathcal{X}_i, \ldots, \mathcal{X}_j)$ *is a path.*
(2) $(\mathcal{X}_m, \ldots, \mathcal{X}_1)$ *is a path.*
(3) *if* $\mathcal{X}_m = \mathcal{Y}_1$, *then* $(\mathcal{X}_1, \ldots, \mathcal{X}_m, \mathcal{Y}_2, \ldots, \mathcal{Y}_n)$ *is a path.*

We can combine Lemmas 9.8 and 9.9 to show that certificate chains are also closed under the substring operation, symmetric, and transitive.

### 9.2. Analysis of Desired Properties

We now analyze our model in terms of the desired properties presented in Section 3. We present precise definitions for all properties (except trust root transparency), and proofs of global authentication and scoped authority. For the remaining properties, we argue that our design of SAINT provides these properties.

**Global authentication.** Intuitively, the global authentication property states that a client can obtain a certificate chain from its trust root to any destination ISD that has a route from its source ISD.

*Definition* 9.10. Let $A$ be a client with routing source ISD $\mathcal{S}$ and trust anchor ISD $\mathcal{T}$. We say that *global authentication* holds if for any destination ISD $\mathcal{D}$ where there exists a route from $\mathcal{S}$ to $\mathcal{D}$, $A$ can obtain a certificate chain with source $\mathcal{T}$ and destination $\mathcal{D}$.

We show that if client $A$ knows a path to its trust anchor ISD $\mathcal{T}$, then $A$ knows a certificate chain from $\mathcal{T}$ to $\mathcal{D}$, i.e., $A$ can authenticate $\mathcal{D}$.

THEOREM 9.11. *Let A be a client with a routing source ISD* $\mathcal{S}$ *and trust anchor ISD* $\mathcal{T}$. *Then for all domains* $\mathcal{D}$ *that have a route from* $\mathcal{S}$, *if A knows a route from* $\mathcal{S}$ *to* $\mathcal{T}$, *A can construct a certificate chain from its trust root* $\mathcal{T}$ *to destination* $\mathcal{D}$.

PROOF. There exist routes from $\mathcal{S}$ to $\mathcal{T}$ and from $\mathcal{S}$ to $\mathcal{D}$. By symmetry of routes (Lemma 9.9), there exists a route from $\mathcal{T}$ to $\mathcal{S}$. Then by transitivity of routes (Lemma 9.9), we conclude that there exists a route from $\mathcal{T}$ to $\mathcal{D}$. Then by Lemma 9.8, a certificate chain from $\mathcal{T}$ to $\mathcal{D}$ exists. □

We note that $A$ does not need to communicate with $\mathcal{T}$ at all in this process — with paths to $\mathcal{T}$ and $\mathcal{D}$, client $A$ has all it needs to construct the certificate chain to authenticate $\mathcal{D}$ based on $\mathcal{T}$. If $A$ can reach $\mathcal{T}$ from $\mathcal{S}$, then it may be able to obtain a shorter certificate chain than one going through $\mathcal{S}$. Finally, even if $A$ cannot reach $\mathcal{T}$ from $\mathcal{S}$, it may be able to obtain or construct a certificate chain from $\mathcal{T}$ to $\mathcal{D}$, using an out-of-band method or by using cached cross-signing certificates. Without making assumptions about the graph of ISDs, we argue that given today's network connectivity, $A$ can likely reach $\mathcal{T}$ and thus obtain a certificate chain to $\mathcal{D}$.

**Scoped authority.** Intuitively, the scoped authority property states that given a client $A$ with a valid evaluation function $\mathsf{eval}_A$, no off-path adversary can forge a certificate chain that $A$ will trust. To formally define this property, we design the following game:
(1) A set $\mathscr{I}$ of ISDs, along with their public keys, routing relations $\mathscr{N}$, and cross-signing relations $\mathscr{C}$, is generated via a setup algorithm, producing a global network.
(2) The adversary is given the generated global network $(\mathscr{I}, \mathscr{N}, \mathscr{C})$, a trust anchor ISD $\mathcal{T} \in \mathscr{I}$, and access to a signing oracle that takes as input a message-ISD pair $(m, \mathcal{I})$ and returns a signature $\mathsf{Sign}_{K_{\mathcal{I}}^{-1}}(m)$. Let $Q$ be the set of all queries sent from the adversary to the oracle.
(3) The adversary finally outputs a sequence $\mathbf{C} = \langle \mathcal{X}_1, \ldots, \mathcal{X}_n \rangle$.
(4) The adversary succeeds if
   (a) $\mathsf{eval}_A(\mathbf{C}) = 1$ for a client $A$ anchoring its trust in $\mathcal{T}$,
   (b) for all $1 \leq i < n$, $(\mathsf{TR}_{\mathcal{X}_{i+1}}, \mathcal{X}_i) \notin Q$, and
   (c) $\mathcal{X}_n \notin \mathscr{I}$, that is, the adversary has certified a public key not belonging to any legitimate ISD in the network.

We can use this game to formalize the concept of scoped authority as follows:

*Definition* 9.12. Let $(\mathscr{I},\mathscr{N},\mathscr{C})$ be a global network. We say that *scoped authority* holds if for all certificate chains in the network, the probability of the adversary winning the above game is low.

We prove this by showing that in essence, an adversary must forge a signature to place its public key in a certificate chain.

THEOREM 9.13. *Suppose that in a global network, the signature $\sigma_{\mathcal{I}\mathcal{J}}$ in a cross-signing certificate $\mathsf{CSC}(\mathcal{I},\mathcal{J})$ is set to $\mathsf{Sign}_{K_{\mathcal{I}}^{-1}}(\mathsf{TR}_{\mathcal{J}})$, where the signature algorithm is unforgeable under a chosen message attack (UF-CMA). Then scoped authority holds in the network.*

PROOF. Suppose that an adversary $M$ exists who wins in the above scoped authority game with high probability. We then show that there exists an adversary $M'$ who successfully attacks the unforgeability of the underlying signature scheme.

Assume the global network consists of a number of honest ISDs $\mathscr{I}$.

Let adversary $M$ output a certificate chain $\mathbf{C}=\langle\mathcal{X}_1,\ldots,\mathcal{X}_n\rangle$ such that $\mathsf{eval}_A(\mathbf{C})=1$ for a client $A$ anchoring its trust in $\mathcal{T}$. To let $M$ be successful, we assume that for all $1\leq i<n$, $(\mathsf{TR}_{\mathcal{X}_{i+1}},\mathcal{X}_i)\notin Q$ and, most importantly, $\mathcal{X}_n\notin\mathscr{I}$, that is, $M$ has certified a public key not belonging to any legitimate ISD in the network.

We let adversary $M'$ simulate adversary $M$, that is, $M'$ obtains the certificate chain $\mathbf{C}$ from $M$ and then outputs the last signature of $\mathbf{C}$. This constitutes a forgery and is thus an attack against UF-CMA. Since $(\mathsf{TR}_{\mathcal{X}_n},\mathcal{X}_{n-1})\notin Q$, and since $\mathcal{X}_{n-1}$ has not signed the last chain element, but the signature of the last element is valid according to the validation of $A$, we know that $M$ must have returned a valid signature to $M'$ without having access to the key and without collaborating with an honest ISD in $\mathscr{I}$. □

We thus conclude that a very simple method for signing certificates and computing $\mathsf{eval}_A(\mathbf{C})$ can provide scoped authority. While we expect that in practice other methods could be used (such as those that rely on multiple certificate chains), we consider these methods out of scope for our analysis, and instead propose challenges for future work to investigate these methods.

**Trust agility.** Intuitively, the trust agility property states that $A$ has a choice of trust root ISDs to verify a destination ISD and that $A$ can easily modify this choice at any time. More precisely, $A$ should be able to download the TRC file for any ISD $\mathcal{T}$ and begin using it as a trust anchor ISD.

*Definition* 9.14. Let $\mathscr{I}$ be the set of ISDs in a SAINT network, and let $A$ be a client. We say that *trust agility* holds if for all $\mathcal{T}\in\mathscr{I}$, $A$ can obtain and verify the TRC file of $\mathcal{T}$.

When analyzing trust agility in SAINT, we consider whether or not $A$ has an existing trust anchor ISD $\mathcal{T}_0$, and whether or not $A$ has a path to $\mathcal{T}$. We thus analyze four cases:

(1) If $A$ anchors its trust in $\mathcal{T}_0$ and has a path to $\mathcal{T}$, then by Theorem 9.11 $A$ can verify a certificate chain from $\mathcal{T}_0$ to $\mathcal{T}$.
(2) If $A$ anchors its trust in $\mathcal{T}_0$ but does not have a path to $\mathcal{T}$, then $A$ can obtain the TRC file of $\mathcal{T}$ from $\mathcal{T}_0$'s TRC server, or contact other ISDs that $A$ can reach to obtain the TRC file.
(3) $A$ has no existing trust anchor but has a path to $\mathcal{T}$, then $A$ can trust its routing source ISD by default to verify its route to $\mathcal{T}$, and in the process obtain a certificate chain to $\mathcal{T}$ that can be verified.
(4) If $A$ has no existing trust anchor and no path to $\mathcal{T}$, then $A$ must obtain $\mathcal{T}$'s TRC file out of band.

As with global authentication, we note that $A$ may rely on cached cross-signing certificates or out-of-band communication to obtain and verify $\mathcal{T}$'s TRC file. As stated in Section 5.3, $A$ can obtain an initial TRC file in a variety of ways. In fact, how $A$ obtains the TRC file is irrelevant, and in the absence of any certificate chains to verify the TRC file, $A$ can rely on out-of-band information for the verification as well.

**Update efficiency.** Intuitively, update efficiency means that a client automatically obtains the latest trust root information. The purpose of providing update efficiency is so that relying parties such as

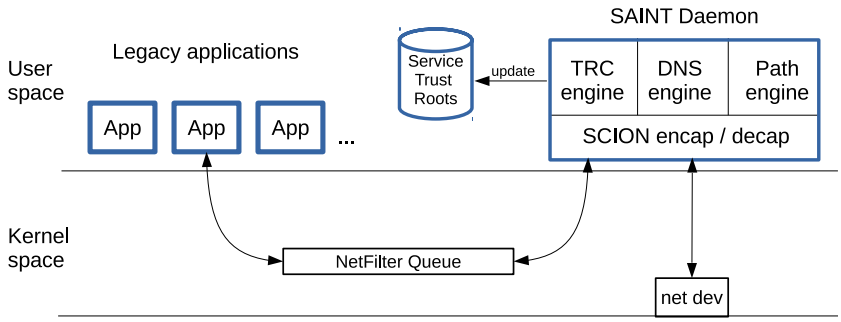


Fig. 13. Architecture for endhost implementation.

clients, servers, and ISDs can have the latest trust root information when verifying a route, name, or EE certificate. We observe, however, that a relying party does not need to have the latest trust root information until it must use the information for verification. We can thus concretely define update efficiency in the following manner:

*Definition* 9.15. Let $A$ be a relying party. We say that *update efficiency* holds if $A$ processes at most one route, name record, or EE certificate each with outdated trust root information.

We can argue that this definition holds in SAINT if $A$ performs a single request for a route, name, or EE certificate at a time (i.e., $A$ does not request multiple routes, names, or certificates at once). Upon receiving a route, name, or certificate with a newer TRC version, $A$ simply performs an additional round trip to its parent or to its trust anchor ISD to obtain the latest TRC file.

**Trust root transparency.** The idea of trust root transparency is that a client can determine what ISD's trust roots are responsible for a part of the certificate chain. The argument to show that SAINT provides trust root transparency is simple: we observe that a client can simply start with its trust root ISD and traverse the certificate chain. For each cross-signing certificate observed, the client can conclude that the ISD whose trust roots are authenticated in the cross-signing certificate is now responsible for all subsequent certificates until the next cross-signing certificate is observed.

## 10. IMPLEMENTATION AND EVALUATION

In this section, we describe our prototype implementation of SAINT. We used this implementation to evaluate the performance of authentication and trust root management functions; we also discuss our evaluation results here.

### 10.1. Implementation

We implemented the endhost side of SAINT (see Figure 13). The main component of our implementation is the *SAINT daemon*, which acts as a gateway between applications and the network. The SAINT daemon includes SCION layer support for packet encapsulation and decapsulation, a path engine for route management and verification, a name lookup engine for SAINT name queries, and a TRC engine, which allows users to obtain and verify TRC files.

Traffic generated by the applications is delivered to the SAINT daemon by the NetFilter queue, allowing legacy applications to deploy SAINT without requiring any changes. We ran our simulation on an Intel Core i5-3380M CPU at 2.90 GHz, 16 GB of RAM, Python 3.4, and gcc 4.8.2. We used Ed25519 [Bernstein et al. 2012] as our signature scheme for name and path verification, and RSA-2048 for TLS certificates.

### 10.2. TRC Updates

We measured the efficiency of TRC distribution and updates by simulating the propagation through the current AS topology. We used the CAIDA inferred AS relationships dataset from October

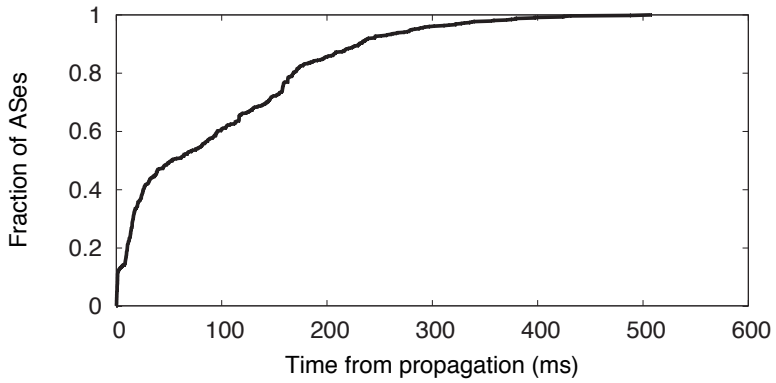


Fig. 14. CDF of the percentage of vantage point ASes using a new TRC after propagation from the core ISPs.

| Measurement | Min | Max | Med | Avg |
|---|---|---|---|---|
| DNS verification | 199 | 967 | 557 | 500 |
| Path verification | 348 | 1 691 | 691 | 652 |
| Certificate validation (TLS) | 210 | 1 123 | 222 | 233 |
| Certificate validation (TLS+CRL) | 401 | 1 775 | 440 | 460 |

Fig. 15. Intra-ISD operation latencies in microseconds, ignoring network delay.

| Measurement | Min | Max | Med | Avg |
|---|---|---|---|---|
| TRC Lookup | 2.6 | 5512.3 | 71.0 | 180.5 |
| DNS Lookup | 2.6 | 5513.2 | 97.3 | 229.8 |
| Path Lookup | 0.2 | 2313.8 | 7.5 | 49.3 |
| TLS Handshake | 5.1 | 5562.7 | 121.1 | 251.9 |
| Current Internet Total | 12.2 | 7968.0 | 275.3 | 528.5 |
| SAINT Total (Trust Anchor ISD) | 13.9 | 11081.0 | 279.8 | 622.5 |
| SAINT Total (Remote ISD) | 16.2 | 16595.0 | 380.8 | 848.7 |

Fig. 16. Latency of operations in SAINT in milliseconds, along with a comparison of end-to-end connection establishment latencies in the current Internet and SAINT, with the client connecting from its trust anchor ISD or a remote ISD.

2014 [CAIDA 2014] as our model of the current topology, and used traceroutes from iPlane datasets[4] to estimate inter-AS latency. From each of iPlane's vantage point ASes (distributed throughout the world), we identified the latency (half of the round trip time) to each of the top-tier ASes identified using the CAIDA-based topology. Since we expect that these top-tier ASes will be core ISPs in SAINT, these latencies provide a reasonable estimate for the latency of endhosts to their respective ISD cores.

Our evaluation demonstrates that more than half of our vantage point ASes receive a new TRC file within 100 ms of the file being sent from the core ISPs (see Figure 14). Moreover, all vantage point ASes receive the new TRC file within 600 ms. Our vantage point ASes included stub ASes (that is, ASes with no customer ASes), demonstrating that end users around the world can quickly receive updated TRC files. These results show that our TRC propagation mechanism is significantly more efficient than the current trust root update mechanisms in browsers (for which the root CA update latency is on the order of days).

### 10.3. Authentication Overhead and Performance

We first tested the speed of cryptographic verifications within an ISD in SAINT by measuring 300 end-to-end secure connection establishments on a sample SCION topology of virtual machines. Figure 15 shows the timing results, which do not take into account the network delay. However, our results give us an insight into the overhead of SAINT: all functions take less than 1 ms on average, which is significantly less than the end-to-end round trip time in an actual connection establishment.

To estimate the performance of SAINT with network delay and compare this performance with that of the current Internet, we measured end-to-end connection establishments to 100 HTTPS sites on the current Internet randomly selected from `httpsnow.org`. We observed 412 separate TLS connection establishments. Using DNSSEC, BGPSEC, and TLS, we measured the latency of the total page loading time, which included blocking of the connection request (not taken into account for comparison), the DNS lookup, the connect, send, and wait, and receive times, and the TLS handshake. Based on the operations described in Section 8, we then broke the connection establishment process into a TRC file lookup, DNS lookup, path lookup, and TLS handshake. We conservatively estimated that a path lookup latency is similar to that of a DNS lookup in the current Internet.

We then calculated the total connection establishment latency for the current Internet, for SAINT within the client's trust anchor ISD, and for SAINT where the client is in a remote ISD. In particular, we note that in its trust anchor ISD, the client does not need to perform a TRC file lookup and enjoys shorter latencies for reaching its DNS root. In contrast, if the client is in a remote ISD, the client must make multiple round trips to remote ISDs to fetch the TRC files necessary to verify the destination ISD, routes, and to perform the DNS query, as shown in Figure 12.

Figure 16 shows the results of our measurements. As expected, end-to-end connection establishment in SAINT takes longer. In particular, for local connection establishment the latency increase is 18% and for remote connection establishment the latency increase is 60% over the current Internet. We note that this is only for connection establishment; end-to-end latency after connection establishment is likely to be similar to in the current Internet, since paths in SCION are not significantly longer than those established by BGP [Zhang et al. 2011]. While these results indicate a potentially high increase in latency, it is worth noting that we performed these measurements without caching, and with caching we may be able to bring the latency closer to that of today's Internet, since we can avoid the expensive round trips to remote ISDs.

## 11. DEPLOYING SAINT

SAINT is an extension of SCION's authentication infrastructure, and therefore we envision that SAINT will be incrementally deployed alongside SCION. As the deployment of SCION is already underway with interested adopters, our approach provides a smooth transition plan for the deployment of SAINT. We can additionally leverage the availability and control-plane isolation benefits provided by SCION.

We now discuss several deployment challenges for SAINT and propose several solutions to facilitate the deployment of SAINT in the current Internet. Specifically, we discuss SAINT's interoperability with the current Internet, and describe how ISDs can be initially deployed. We then propose a method for the initial distribution of TRC files.

**The Legacy ISD.** To facilitate the incremental deployment of SAINT, we propose a special ISD called the *legacy ISD* $\mathcal{L}$, which represents the current Internet. The legacy ISD relies on DNSSEC, BGPSEC, and TLS and thus provides only the guarantees of the existing Internet's authentication mechanisms. Addressing in SAINT would retain legacy AS numbers and IP addresses with the ISD $\mathcal{L}$. For example, suppose that `a.com` maps to the IP address 1.2.3.4 in the current Internet, which is located in AS 567, which has not yet deployed SAINT. The domain `a.com` would then correspond to the address $(\mathcal{L}, 567, 1.2.3.4)$.

[4] http://iplane.cs.washington.edu/data/data.html

One challenge we face in practice is that names in SAINT's generic TLDs may already exist in the legacy ISD. Not only do such collisions cause a problem for name resolution, but they also create vulnerabilities to downgrade attacks in SAINT's name lookup mechanism, since an adversary could simply return an unsecured DNS response in the legacy ISD for a query with a name collision.

One simple way to prevent this problem is to enforce a collision-free policy in SAINT so that no TLDs that exist in the legacy ISD are allowed. There is still a chance that ICANN may cause collisions in the future by adding a TLD that already exists in the SAINT namespace, but if we use countries as ISDs, we can likely avoid this problem, as even with the recent TLD additions, ICANN requires regional names to undergo stricter approval processes [ICANN 2012].

Furthermore, we propose three mechanisms to remedy this solution as we transition from the current Internet to SAINT. First, we require domains that have names both in SAINT and in the legacy DNS to signal this through the legacy DNS. For example, a domain could create an experimental resource record with a SAINT address that can be queried by SAINT clients, a CNAME redirect to a SAINT name, or a TLSA record that indicates a public key that the domain only provides through SAINT. Each method has its particular drawback; for example, an experimental record type would need to be queried by clients for all legacy domains, and a CNAME or TLSA record may inadvertently break legacy clients.

Second, name resolvers in SAINT can query both the legacy namespace and the SAINT namespace in parallel, signaling to the client when there is a conflict in the namespace. These conflicts can be cached by the resolver or by clients to prevent excessive parallel lookups. The advantage of this method is that it provides transparency, highlighting conflicts where they occur rather than selecting the results from one namespace to present to the client. However, false positives are possible where accidental collisions in the namespace occur. For example, `b.my` in the Mythuanian namespace and `b.my` in the legacy namespace (the Malaysian ccTLD) may be completely unrelated sites even though their names collide.

Finally, we can require SAINT name resolvers to query the legacy ISD as a last resort, and only with a proof of the name's absence in the SAINT namespace (such as an NSEC3 record). This mechanism avoids downgrade attacks and can be useful further in deployment when few legacy names are queried, but in the early stages can potentially disrupt a large number of lookups. However, if we take measure to avoid conflicts with the legacy namespace as described above, we can minimize the disruption caused with this method.

In order to maintain connectivity to the current Internet, servers and clients must support legacy authentication. In particular, clients and servers must continue to support DNSSEC, BGPSEC, and TLS. Additionally, when servers receive incoming connections from the legacy ISD, they should not respond with SAINT-specific messages such as signed path sets or cross-signing certificates. However, TRC files will be made available through the legacy ISD in order to support the initial bootstrapping of trust roots.

**ISD Deployment.** SAINT offers benefits even for early deployers of ISDs, namely isolation of compromises in the ISD and protection from compromises in the legacy ISD. Each ISD can also reap benefits of deploying an alternative PKI without requiring global deployment. An ISD can specify its choice of PKI in the policy field of the TRC file, preventing protocol downgrade attacks. If using countries as ISDs, deploying an ISD involves simply attaching to the existing namespace at its corresponding ccTLD. This construction allows the DNS to provide a scaffolding during the deployment of SAINT and also allows the DNS in SAINT to distribute TRC files.

However, we recognize the challenges that come with using countries as ISDs. In particular, the deployment of such a scheme would require the core ISPs, root CAs, and Internet registries in each country to create a federation of trust roots. In practice, we may see corporations rather than countries form ISDs, which may require IP tunnels in order to form inter-ISD routing relationships. Additionally, since there are far more corporations than countries, and the number of cross-signing relationships may be grow proportionally to the square of the number of ISDs, the scalability of SAINT could fail with corporations as ISDs. In practice, however, we anticipate that corporations

will only maintain connectivity and business relationships with a small fraction of other ISD-scale corporations, and thus we do not rule out the possibility of corporations being ISDs.

**Initial TRC Distribution.** The initial distribution of TRCs must occur securely since TRCs anchor all authentication in SAINT. Many trust roots in the current Internet may continue to serve as trust roots in SAINT, and thus may be able to "inherit" user trust in SAINT that they already have in the current Internet. However, SAINT will likely result in the creation of new trust roots, and thus must have a mechanism for bootstrapping trust in the initial public keys of these roots.

To address this challenge, we suggest to perform the initial distribution of SAINT TRC files through DNSSEC. Since ISDs can deploy by attaching to specific ccTLDs in the current DNS namespace, an ISD can create a reserved domain name such as `trc.us`, whose DNS record contains the TRC (e.g., in a TXT record). Clients can then fetch the TRC by looking up the appropriate domain name. Additional work has been done in distributing authentication information through out-of-band means such as over public radio [Schulman et al. 2014], but these strategies are beyond the scope of this paper.

## 12. DISCUSSION

**Feasibility of country-based ISDs.** In order to determine the feasibility of having countries as ISDs in SAINT as described in Section 4.1, we mapped AS numbers to countries and examined the resulting inter-ISD relationships. We used the AS relationships database from CAIDA [CAIDA 2014] and Team Cymru's IP to AS number mapping tool,[5] to map AS numbers to countries. We identified 228 "countries" in total, including the EU and ZZ (indicating that the AS's country was unknown). We identified 2 636 unique country pairs between which an inter-AS link existed. These links signify direct routing connections, and thus we expect cross-signing for each ISD pair. The most prolific cross-signing ISDs were the US (196), the EU (135), and the UK (124), but half of the ISDs cross-sign on the order of tens of other ISDs.

**Political Concerns.** Given concerns over governmental nation-wide surveillance, one may be concerned about centralizing trust roots in a large ISD. While we acknowledge that states may compromise these entities on a large scale, SAINT's trust agility allows users to protect themselves from the interception of sensitive connections. Additionally, our efficient method of updating trust root information allows ISDs to quickly recover from a compromised CA.

Another concern may be that SAINT encourages fragmentation in the Internet. While ISDs separate the control planes, SAINT is designed to preserve global reachability while simultaneously protecting users through the use of ISDs and trust agility. We thus structure inter-ISD authentication to provide users with the best of both worlds.

**Optimizations.** As described in Section 8, the authentication process involves six round-trip connections from the client Alice. Each communication with the path server requires verification of the path server's signature, the signed set of routing paths returned by the path server, the destination AS certificate, and possibly cross-signing certificates for the path server and destination AS's ISDs. To contact a destination outside her trust anchor ISD, Alice must also obtain and verify a cross-signing certificate. Contacting the DNS server requires verification of at least the DNS root key, server DNS key, and signed DNS record, and contacting the server (Bob) requires verification of at least the server certificate.

However, in practice many of these verifications may not be necessary. Caching cross-signing certificates, for example, eliminates the need to reach a TRC server and to verify a cross-signing certificate with each end-to-end connection establishment. Additionally, some of these verifications can be handled by entities other than the client; for example, paths can be verified by the client's AS, and DNS records by a trusted DNS stub resolver. Since ASes and stub resolvers serve multiple

[5] http://www.team-cymru.org/Services/ip-to-asn.html

clients, caching verification results can further reduce the connection latency, especially for popular names, routes, and EE certificates.

In order to further decrease the size of messages sent in the network, we can also split the TRC file into routing and a service TRC files. The routing TRC can then be propagated along AS links, and the service TRC can be obtained from the TRC server. This scheme ensures that users only receive the portion of the TRC that they need for a particular type of authentication, thus reducing the size of TRC files sent in the network.

## 13. CONCLUSIONS

By explicitly separating and scoping trusted authorities in the Internet, we allow users to choose their trust roots and protect users from CA compromises. By distributing trust root information as network messages, we allow users to quickly obtain up-to-date information about compromised or updated trust roots. By mandating cross-signing relationships based on routing connections, we ensure that users can authenticate information throughout the Internet. By separating routing and service authentication, we allow users' trust root decisions to apply anywhere in the world. These ideas address fundamental shortcomings of current authentication and secure the communications of clients throughout the world regardless of their choice of trust roots.

### Acknowledgments

We would like to thank Yih-Chun Hu, Virgil Gligor, Ari Juels, Burt Kaliski, and Gene Tsudik for their insightful comments and guidance on drafts of this paper. The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement 617605.

We also gratefully acknowledge support from the NSF under grants CNS-1040801 and DGE-1252522, from ETH Zurich, and from Google.

## REFERENCES

Martin Abadi, Andrew Birrel, Ilya Mironov, Ted Wobber, and Yinglian Xie. 2013. Global Authentication in an Untrustworthy World. In *HotOS*.

David G. Andersen, Hari Balakrishnan, Nick Feamster, Teemu Koponen, Daekyeong Moon, and Scott Shenker. 2008. Accountable Internet Protocol (AIP). In *SIGCOMM*.

R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. 2005. DNS Security Introduction and Requirements. RFC 4033. (2005).

David Barrera, Raphael M. Reischuk, Pawel Szalachowski, and Adrian Perrig. 2015. SCION Five Years Later: Revisiting Scalability, Control, and Isolation on Next-Generation Networks. *arXiv e-prints* (Aug. 2015). http://arxiv.org/pdf/1508.01651

David Basin, Cas Cremers, Tiffany Hyun-Jin Kim, Adrian Perrig, Ralf Sasse, and Pawel Szalachowski. 2014. ARPKI: Attack Resilient Public-Key Infrastructure. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 382–393.

Daniel J Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. 2012. High-speed high-security signatures. *Journal of Cryptographic Engineering* 2, 2 (2012).

Andrew D. Birrell, Butler W. Lampson, Roger M. Needham, and Michael D. Schroeder. 1986. A Global Authentication Service without Global Trust. In *IEEE S&P*.

Julian Borger. 2013. GCHQ and European spy agencies worked together on mass surveillance. http://www.theguardian.com/uk-news/2013/nov/01/gchq-europe-spy-agencies-mass-surveillance-snowden. (2013).

CAIDA. 2014. The CAIDA AS Relationships Dataset. http://www.caida.org/data/as-relationships/. (2014).

Ran Canetti, Juan Garay, Gene Itkis, Daniele Micciancio, Moni Naor, and Benny Pinkas. 1999. Multicast security: A taxonomy and some efficient constructions. In *IEEE International Conference on Computer Communications (INFOCOM)*, Vol. 2. 708–716.

I. Castineyra, N. Chiappa, and M. Steenstrup. 1996. The Nimrod Routing Architecture. RFC 1992. (1996).

Miguel Castro and Barbara Liskov. 1999. Practical Byzantine Fault Tolerance. In *Third Symposium on Operating System Design and Implementation (OSDI)*.

David Chaum and Eugène Van Heyst. 1991. Group signatures. In *EUROCRYPT*.

Laurent Chuat, Pawel Szalachowski, Adrian Perrig, Ben Laurie, and Eran Messeri. 2015. Efficient Gossip Protocols for Verifying the Consistency of Certificate Logs. In *IEEE Conference on Communications and Network Security (CNS)*. 415–423.

D. Clark, R. Braden, A. Falk, and V. Pingali. 2003. FARA: reorganizing the addressing architecture. *SIGCOMM CCR* 33, 4 (2003).

Danny Cooper, Ethan Heilman, Kyle Brogle, Leonid Reyzin, and Sharon Goldberg. 2013. On the risk of misbehaving RPKI authorities. In *HotNets*. ACM.

David Cooper, Stefan Santesson, Stephen Farrell, Sharon Boeyen, Russell Housley, and Tim Polk. 2008. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280. (May 2008).

Tim Dierks and Eric Rescorla. 2008. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246. (2008).

C. Dillow. 2010. An Order of Seven Global Cyber-Guardians Now Hold Keys to the Internet. http://www.popsci.com/technology/article/2010-07/order-seven-cyber-guardians-around-world-now-hold-keys-internet. (2010).

Peter Eckersley and Jesse Burns. 2010. Is the SSLiverse a Safe Place? CCC 2010. (December 2010).

Barton Gellman and Laura Poitras. 2013. U.S., British intelligence mining data from nine U.S. Internet companies in broad secret program. http://www.washingtonpost.com/investigations/us-intelligence-mining-data-from-nine-us-internet-companies-in-broad-secret-program/2013/06/06/3a0c0da8-cebf-11e2-8845-d970ccb04497_story.html. (2013).

Virgil D. Gligor, Shyh-Wei Luan, and Joseph N. Pato. 1992. On Inter-realm Authentication in Large Distributed Systems. In *S&P*.

P. Hoffman and J. Schlyter. 2012. The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA. RFC 6698. (2012).

ICANN. 2012. gTLD Applicant Guidebook. (June 2012).

James Kasten, Eric Wustrow, and J Alex Halderman. 2013. CAge: Taming certificate authorities by inferring restricted scopes. In *FC*.

Stephen Kent, Charles Lynn, and Karen Seo. 2000. Secure Border Gateway Protocol (S-BGP). 18, 4 (2000).

Tiffany Hyun-Jin Kim, Lin-Shung Huang, Adrian Perrig, Collin Jackson, and Virgil Gligor. 2013. Accountable Key Infrastructure (AKI): A Proposal for a Public-Key Validation Infrastructure. In *WWW*.

Leslie Lamport. 1998. The part-time parliament. *Transactions on Computer Systems (TOCS)* 16, 2 (1998), 133–169.

Butler Lampson, Martin Abadi, Michael Burrows, and Edward Wober. 1991. Authentication in Distributed Systems: Theory and Practice. In *SOSP*.

Ben Laurie, Adam Langley, and Emilia Kasper. 2013. Certificate Transparency. RFC 6962. (2013).

M. Lepinski. 2013. BGPSEC Protocol Specification. https://tools.ietf.org/html/draft-ietf-sidr-bgpsec-protocol-07. (2013).

M. Lepinski and S. Kent. 2012. An Infrastructure to Support Secure Internet Routing. RFC 6480. (2012).

Ang Li, Xin Liu, and Xiaowei Yang. 2011. Bootstrapping accountability in the internet we have. *NSDI* (2011).

Moxie Marlinspike. 2011. SSL and the Future of Authenticity. http://www.thoughtcrime.org/blog/ssl-and-the-future-of-authenticity/. (2011).

Stephanos Matsumoto and Raphael M. Reischuk. 2015. Certificates-as-an-Insurance: Incentivizing Accountability in SSL/TLS. In *NDSS SENT*. http://internetsociety.org/sites/default/files/01_6.pdf

David Mazieres, Michael Kaminsky, M. Frans Kaashoek, and Emmett Witchel. 1999. Separating key management from file system security. In *SOSP*.

R. Moskowitz, T. Heer, P. Jokela, and T. Henderson. 2008. Host Identity Protocol. RFC 5201. (2008).

Diego Ongaro and John Ousterhout. 2014. In search of an understandable consensus algorithm. In *USENIX Annual Technical Conference (ATC)*. 305–319.

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

Mark D Ryan. 2014. Enhanced certificate transparency and end-to-end encrypted mail. *NDSS* (2014).

Aaron Schulman, Dave Levin, and Neil Spring. 2014. RevCert: Fast, Private Certificate Revocation over FM Radio. *CCS* (2014).

Victor Shoup. 2000. Practical threshold signatures. In *EUROCRYPT*.

Pawel Szalachowski, Stephanos Matsumoto, and Adrian Perrig. 2014. PoliCert: Secure and Flexible TLS Certificate Management. In *CCS*.

Fred Upton, Tim Murphy, Greg Walden, and Michael C. Burgess. 2015. Letters to Browsers Regarding Government Certificate Authorities. https://energycommerce.house.gov/news-center/letters/letters-browsers-regarding-government-certificate-authorities. (June 2015).

Greg Weston, Glenn Greenwald, and Ryan Gallagher. 2013. Snowden document shows Canada set up spy posts for NSA. http://www.cbc.ca/news/politics/snowden-document-shows-canada-set-up-spy-posts-for-nsa-1.2456886. (2013).

Xin Zhang, Hsu-Chun Hsiao, Geoffrey Hasker, Haowen Chan, Adrian Perrig, and David G. Andersen. 2011. SCION: Scalability, Control, and Isolation On Next-Generation Networks. In *IEEE S&P*.

## A. DNS RESOLUTION FOR GENERIC TLDS

As stated in Section 4.5, for reasons of transparency and security, each domain name with a generic TLD is resolved to a regional domain name (rather than to an address). Assume, for example, a company $r$ wants to register the domain `r.com`. Instead of registering a DNS tuple $(\texttt{r.com}, IP)$, the company registers one or more CNAME-like records[6] that point to other (usually regional) domain names:

$$\texttt{r.com} \rightarrow \{\texttt{r.us},\ \texttt{r.de},\ \texttt{r-swiss.ch},\ \texttt{r-italia.it}\} \tag{1}$$

Upon a DNS resolution request for a generic domain $D$ from a client, the DNS server for `.com` returns all regional names for $D$ in the order specified by the registrant $r$. The client then either chooses a domain name that is within its own ISD, or it chooses any other domain name in the provided list.

In order to ensure the authenticity of generic DNS records, SAINT requires a minimal setup as follows: any registrant $r$ must first register its DNS public key $DK_r$ with the generic DNS server $S$.

$$r \xrightarrow{DK_r} S\ (\texttt{.com})$$

$S$ stores and signs the public key $DK_r$, and returns the signature $\{r, DK_r\}_{K_S^{-1}}$.

$$r \xleftarrow{\{r,DK_r\}_{K_S^{-1}}} S\ (\texttt{.com})$$

After this initial step, registrant $r$ registers its domain name $D$ by providing the list of regional domain names $\{...\}$ together with a signature $\{D, \{\ldots\}\}_{DK_r^{-1}}$.

$$r \xrightarrow{\{D,\{...\}\}_{DK_r^{-1}}} S\ (\texttt{.com})$$

**Verification.** Consider client $c$, who wants to authenticate a DNS response for a generic domain $D$.

$$c \xrightarrow{D?} S\ (\texttt{.com})$$

$$\xleftarrow{\{D,\{...\}\}\quad \{D,\{...\}\}_{DK_r^{-1}}\quad DK_r\quad \{r,DK_r\}_{K_S^{-1}}}$$

The client verifies the signature $\{r, DK_r\}_{K_S^{-1}}$ and caches the public verification key $DK_r$ of the registrant $r$ of domain $D$. The client then verifies the authenticity of the record $\{D, \{\ldots\}\}$ using the registrant's public key $DK_r$. If the verification was successful, the client choses one of the specified regional domain names and resolves its actual address $(\mathcal{I}, A, E)$.

**Performance.** As in the current DNSSEC, the DNS server $S$ for generic TLDs does not sign the stored CNAME-like records itself; rather, it signs the public keys of the registrants. The reasons include performance considerations: whenever a registrant $r$ wants to register a new generic domain or whenever $r$ wants to extend an existing record, the DNS server has minimal effort in that it does not need to validate or sign the new records. Using CNAME-like records also keeps the performance close to existing lookups: CNAME records add only 13% latency to a DNS name resolution on average.[7]

**Availability.** Whenever a client needs to resolve a generic domain name, the client first contacts its local DNS server. If the local DNS server has no cached entry for the generic domain, the request is redirected to the DNS server of the generic TLD. This one step of indirection is at least as robust as today's DNS system: in case the DNS server for a generic domain is unavailable, then only the availability of that single TLD is constricted. There is hence no single point of failure for the entire DNS system. As today, caching of generic domain name records by local DNS servers further increases the robustness and performance for generic DNS lookups.

**Security.** The record for a generic domain $D$ is signed by the owner of $D$ (i.e., the registrant $r$) using an asymmetric signature scheme and the private signing key of $r$. The public verification key of $r$ is signed by $r$'s ISD and by the DNS server for `.com`. Client $C$ can hence base its trust on the ISD of $r$ or (in case $C$ does not trust this ISD) on the DNS server for `.com`.

Another positive aspect of this design is the fact that a key compromise of `.com`'s DNS server does not directly affect the security of an end-to-end connection: an attacker would additionally need to compromise the DNS server of a regional ISD, which is used for the second lookup, the regional lookup for the actual address $(\mathcal{I}, A, E)$. Despite being unlikely, this attack only works under the assumption that a client uses the verification key of `.com` (rather than the verification key of the resolved regional CNAME domain).

[6]CNAME records cannot point to multiple names, but the general idea of our records is the same.

[7]This result is based on our private discussions with Verisign Labs.