6 Name Resolution

DANIELE ASONI, YIH-CHUN HU,
RAPHAEL M. REISCHUK, BRIAN TRAMMELL

While the path resolution process is necessary to turn a destination address into a set of paths, this is not sufficient for establishing communication between SCION-connected endpoints: we also need a way to turn an Internet name into a SCION address. As name resolution and path establishment are separate processes, with different timescales and triggered by separate events, we design a dedicated infrastructure that is optimized for each purpose.

We begin with an analysis of what a name resolution service is good for. At its core a naming service must provide a few basic functions, but in essence associate a human-understandable name with machine-understandable information. Although the Internet's Domain Name System (DNS) has been used and abused as a general-purpose distributed database, a useful Internet naming service need only provide information that is necessary to establish and maintain communication with an Internet-connected entity: addresses, namespace delegations, service information, certificates, and auxiliary information. There are two entities in this ideal naming service: (a) The *querier* is a client who wants to establish communication across the Internet with a named entity. That client uses the naming service to retrieve the necessary information. (b) The *authority* is an entity with the right to make assertions about names within parts of the Internet namespace. Before looking at the specific interplay between queriers and authorities, we start by discussing various possible resolution types.

Chapter Contents

6.1 Background

6.1.1 Resolution Types

The assertions for name resolution are essentially mappings of various forms:

- **Name-to-address:** given a name, return associated addresses.
- **Name-to-name:** given a name, return equivalent names.
- **Name-to-service:** given a name representing a service, return a name and transport-layer ports for connecting to the service.
- **Name-to-certificate:** given a name representing a host or service, return an end-entity certificate representing the named entity, for authentication of a subsequent connection attempt with the named entity.
- **Name-to-delegation:** given a name representing a zone within the namespace, return the public key used to verify assertions in the zone.
- **Name-to-auxiliary-information:** given a name representing an organization-level zone within the namespace, return information about the zone and the organization behind it, analogous to the WHOIS service; as well as any restrictions on names in the zone (e.g., for confusability reduction).
- **Address-to-name:** given an address, return associated names, analogous to reverse DNS.

6.1.2 Properties of an Ideal Naming Service

We consider a set of properties of an ideal Internet naming service as background to selecting a design for SCION name resolution. A more in-depth discussion of these properties, enumerated below, is given in an IETF draft [236]. An ideal naming service must

- provide for names which are meaningful to human users;
- guarantee that different names are distinguishable by its users;
- allow for authority over names to be federated;
- allow a unitary authority for any given name to be transparently determined;
- operate without requiring trust in the operators of the name server infrastructure;
- provide for revocation of authority over a given name;
- allow assertions about names, and the nonexistence of a mapping for a name, to be unambiguously authenticated;
- provide for consistency, and predictability in the presence of changes to assertions about names, but

- allow for explicit inconsistency when necessary, and global transparency of this inconsistency;
- perform acceptably, in terms of availability, latency, and bandwidth efficiency;
- allow clients to specify tradeoffs between privacy and performance.

Since an Internet naming service is designed to provide information about Internet-connected hosts and services for the purposes of establishing a connection, note that assertion confidentiality (usually referred to in DNS literature in terms of zone enumeration) is a non-goal of our ideal naming service. If assertion confidentiality is required, an alternative service can be established that provides access control to the information that should remain secret.

6.1.3 Notation

Throughout the chapter, we make use of the terms, abbreviations, and resource record (RR) types defined in Table 6.1.

Term	Definition
Assertion	Mapping between a name and a property of that name.
Shard	Set of assertions for some authority and context.
Zone	Set of all shards and assertions for some authority and context.
NCO	Naming Consistency Observer (see Page 124)
RR	Resource Record (fundamental data unit, see table below)
TLD	Top-Level Domain (such as `.ch` or `.com`)
ZK	Zone Key (key to sign assertions for the respective zone)
RZK	Root Zone Key (special zone key used for the root zone)

(a) Terms and Abbreviations

RR Type	Content
`A`	32-bit IPv4 address
`AAAA`	128-bit IPv6 address
`CNAME`	canonical name for a given alias
`NS`	responsible name server
`PTR`	pointer to domain name (e.g., address-to-name mapping)
`SRV`	service locator (locates service/protocol for a given domain)
`TLSA`	certificate or public key association (see DANE [60])

(b) Resource Record (RR) Types

Table 6.1: Notation used in the context of name resolution.

6.2 Name Resolution Architecture

We now turn our attention to designing protocols to provide this ideal naming service, which we call RAINS (a recursive acronym for "RAINS, another Internet naming service") [237].

Why not just use DNS?

We note that the DNS protocol as used in the present Internet, when deployed with the mandatory usage of DNS Security Extensions (DNSSEC) and one root per ISD, meets most of the properties of our ideal name system. Only explicit tradeoffs and explicit inconsistency are not well-supported by DNS with DNSSEC. An initial approach to providing name resolution for SCION could therefore be to borrow DNS.

Using DNS would have the following advantages:

- It leverages an existing, widely deployed protocol, with which there is widespread operational experience.
- It allows names for SCION-enabled nodes to be registered in the same name resolution system as the non-SCION Internet, which should make incremental deployment easier.

It would also have some serious disadvantages:

- DNS has no concept of explicit inconsistency or explicit tradeoff, especially for privacy.
- Even with DNSSEC, DNS has poor operational security properties, specifically lack of query anonymity and vulnerability to abuse as an amplification attack vector.
- Since DNSSEC would be mandatory for SCION RRs, SCION-enabled nodes could only use signed top-level domains (TLDs). Many country-code TLDs remain unsigned.
- Support for extension mechanisms for DNS (EDNS0) and DNSSEC varies widely among stub and recursive resolvers, which negates the incremental deployment advantage above: lack of interoperability of the minimum DNS required for SCION and other DNS-supporting software and hardware would lead to difficult-to-debug issues.

The final disadvantage is the most troubling, and led to our decision to build a new name resolution protocol for SCION.

The RAINS architecture is simple, and resembles the architecture of DNS. A RAINS server is an entity that provides transient and/or permanent storage for **assertions** about names, and a lookup function that finds assertions for a given **query** about a name, either by searching local storage or by delegating to another RAINS server. RAINS servers can take on any or all of three roles:

- **authority service**, acting on behalf of an authority to ensure properly signed assertions are made available to the system;
- **query service**, acting on behalf of a client to answer queries with relevant assertions, and to validate assertions on the client's behalf; and/or

- **intermediary service**, acting on behalf of neither but providing storage and lookup for assertions with certain properties for query and authority servers.

RAINS servers use the RAINS protocol, described in this section, to exchange queries and assertions. RAINS clients use a subset variant of the RAINS protocol (called the RAINS client protocol) to interact with RAINS servers providing query services on their behalf. RAINS protocol connections among servers are encrypted and authenticated. RAINS client protocol connections between clients and query servers are encrypted and optionally authenticated. In addition, the RAINS protocol provides object-level authentication. Section 6.4.1 provides details on bootstrapping trust using RAINS.

Authority service in RAINS resembles the role of authoritative servers in the present DNS. Query service resembles the role of recursive resolvers. Intermediate service resembles the role of caching resolvers. RAINS is therefore a drop-in replacement for the present DNS with better support for contexts and tradeoffs and with mandatory delegation and authentication by signature chain. As with DNS, a given RAINS server may play both the authority server and query server roles at any given time, depending on configuration. However, future implementations of RAINS could use other mechanisms for matching queries and assertions, and moving assertions to where they can be most efficiently matched with queries.

From the basic building blocks of these three services, any number of naming service architectures could be built. Within SCION, RAINS authority services are generally operated by TLDs (as isolation context root authority servers) as well as domain name registrants or ISPs acting on their behalf. Intermediate and query services are operated by ISPs and enterprise networks, and ASes make query servers available via service anycast (see Section 7.5 on Page 162).

RAINS also integrates into SCION's authentication infrastructure. End-entity certificates for named hosts can be stored in RAINS, and RAINS intermediate and query services supports assertions signed via ARPKI (see Section 4.4).

There is an inherent tension between SCION's architectural principle of isolation and the need for a globally consistent namespace. RAINS on SCION resolves this by supporting *isolation transparency*. Queries and assertions can cross ISD boundaries, which is the basis of the Naming Consistency Observer (NCO) described in detail in Section 6.5 on Page 124.

So what is new in RAINS?

Though designed as a drop-in replacement, RAINS makes several radical departures from DNS as presently specified and implemented:

- Delegation from a superordinate zone to a subordinate zone is accomplished solely with cryptography: a superordinate defines the key(s) that are valid for signing assertions in the subordinate during a particular time interval. Assertions about names can therefore safely be served from any infrastructure.
- All time references in RAINS are absolute: instead of a time to live, each assertion's temporal validity is defined by the temporal validity of the signature(s) on it.
- All assertions have validity within a specific context. A context determines the rules for chaining signatures to verify the validity of an assertion. Within SCION, publicly-available names within an ISD exist within that ISD's native isolation context. The use of context explicitly separates global usage of the DNS from local usage thereof, and allows other application-specific naming constraints to be bound to names; see Section 6.3.3.
- Explicit information about registrars and registrants is available in the naming system at runtime, combining the functionality of WHOIS with the naming service.
- Sets of valid characters and rules for valid names are defined on a per-zone basis, and can be verified at runtime.
- Reverse lookups are performed on a completely separate tree, supporting delegations of any prefix length, in accordance with classless inter-domain routing (CIDR) and the IPv6 addressing architecture.

6.3 Naming Information Model

Here we describe the information model for messages in the RAINS protocols. For simplicity of description, we omit details on error handling and parts of the information model necessary for protocol implementation and operation. The detailed protocol specification is in our IETF draft [237].

RAINS operates on two different basic types of information: **assertions** (Section 6.3.1) are mappings between a name and some property of the name, which can be grouped into *shards* and *zones* (Section 6.3.2) for performance and operational optimizations; and **queries** (Section 6.3.5) are expressions of interest about certain types of information about a name. RAINS matches queries to assertions that answer them.

6.3.1 Assertions

An assertion consists of the following elements:

- **Context**: name of the isolation context in which the assertion is valid. Section 6.3.3 provides more details.
- **Subject**: the non-qualified name about which the assertion is made. A non-qualified name is a local, not necessarily globally unambiguous identifier (e.g., `'foo'`), which — in combination with the zone name (e.g., `'example.com'`) — yields the fully-qualified (i.e., unambiguous) name (e.g., `'foo.example.com'`). The domain name separator here is `'.'` and separates subject and zone.
- **Zone**: the name of the zone (e.g., `'example.com'`) in which the assertion is made.
- **Object**: the data associated with the name of the given type.
- **Type**: the type of information about the subject contained in the assertion. Each assertion is about a single type of data. Supported types include:
  - **Delegation**: the authority associated with the zone identified by the name (replaces the `NS` DNS record type for cryptographic delegation; see below)
  - **Redirection**: the authority servers for the zone identified by the name (analogous to the `NS` DNS record type)
  - **Address**: one or more addresses associated with the name, given an address family (analogous to the `A` and `AAAA` DNS record types).
  - **Service-info**: one or more layer 4 ports associated with the name, if the name identifies a service (analogous to the `SRV` DNS record type).
  - **Name**: one or more names associated with the name (analogous to the `CNAME` and the `PTR` DNS record types: a `PTR`-analog lookup is defined by the zone in which the lookup is made).

  - **Certificate**: an end-entity certificate representing the named entity, for authentication of a subsequent connection attempt with the named entity (analogous to the `TLSA` DNS record type).
  - **Nameset**: an expression of the set of names allowed within a zone; e.g., Unicode scripts or codepages in which names in the zone may be issued. An assertion about a subject within a zone whose name is not allowed by a valid signed nameset expression is taken to be invalid.
  - **Registrar**: a string identifying the registrar responsible for the appearance of a delegation within a zone, for TLDs that allow multiple organizations to modify their zones.
  - **Registrant**: a string containing information about the registrant of a zone within a TLD (analogous to the WHOIS service).
  - **Infrastructure-key**: a public key by which a RAINS server can be identified, for object security on RAINS messages.
  - **External-key**: a public key by which assertions in a zone can be verified outside the delegation hierarchy, e.g., via an SCP as described in Section 4.4.3.
- **Issued**: a timestamp at which the assertion was made.
- **Expires**: a timestamp after which the assertion is no longer valid.
- **Signature**: a signature generated by the authority, to authenticate the assertion. This signature covers all elements within the assertion except the signatures themselves. An assertion may have multiple concurrently-valid signatures.

Issued and expired timestamps are always expressed in terms of UTC. Since the signature protects the timestamps as well, it is necessary to sign new assertions before old ones expire. At a single point in time, it is possible to have multiple active valid assertions with overlapping validity times for a given $\langle$subject, zone, context, type$\rangle$ tuple. The union of the object values of all of these assertions is considered to be the set of valid values at that point in time.

6.3.2 Grouping Assertions: Shards and Zones

An authenticated assertion with a valid signature provides a proof of existence of a name. Another mechanism is necessary to provide a proof of nonexistence; otherwise, malicious intermediate and query services could cause false negatives for queries by simply refusing to forward matching assertions. RAINS provides shards for this purpose.

A **shard** is a set of assertions for the same authority within the same context, protected by an additional signature over all assertions within the shard, which has the property that, given a subject and an authenticated shard, then either an assertion of a given type exists within the shard, or does not exist at all. We

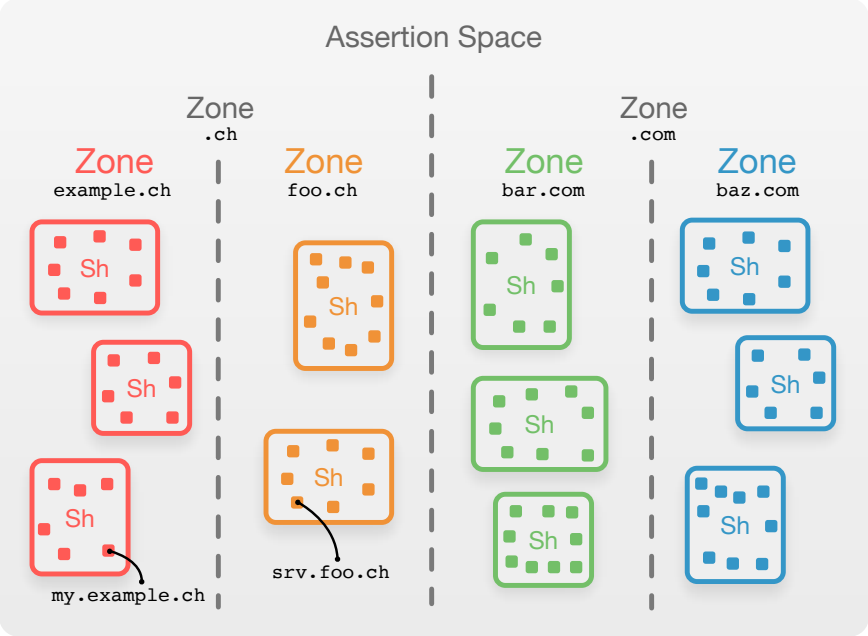

Figure 6.1: Hierarchical zones with shards and assertions.

achieve this property by associating the shard with an exclusive **shard range** of names appearing in a shard: a shard with the range ‘a’ to ‘b’ contains all names in the zone and context that sort after ‘a’ and before ‘b’. This property allows efficient verification of the nonexistence of an assertion for a given name at the query.

Example. Consider a zone containing the names ‘aaa’, ‘aab’, ‘baa’, ‘cat’, ‘dog’, ‘nap’, ‘yyz’, and ‘zzz’, as illustrated in Figure 6.2. This

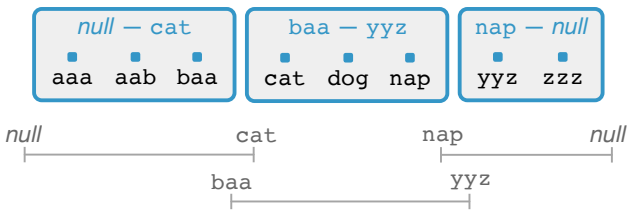

Figure 6.2: Eight assertions aligned in three shards with overlapping ranges.

zone could be split into three shards: {aaa, aab, baa}, {cat, dog, nap}, and {yyz, zzz}. To ensure that a proof of nonexistence can be given for any name other than these eight using only one of these shards, the shard ranges overlap: the first shard has the range {null - cat}, the second the range {baa, yyz}, and the third the range {nap - null}. Note that names falling between the names in the shards can be disproved using either of the neighboring shards.

A **zone** is the entire set of *shards* and assertions for a given authority within a given context. Figure 6.1 shows two zones ('.ch' and '.com') with two subordinated zones each. A zone may also contain assertions about the zone itself; this is especially useful for self-signing root zones.

6.3.3 Isolation Contexts

All assertions are held to be valid within an explicitly named *assertion context*. Assertion contexts are used to determine the validity of the signature by the declared authority. There are two broad uses for assertion contexts: *isolation* and *local assertion*. Isolation contexts allow assertions and queries about an ISD other than that from which a query was made.

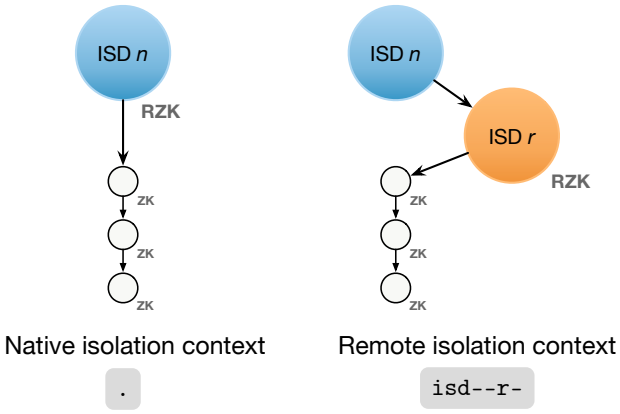

Figure 6.3: Isolation Contexts. RZK is the root zone key for an ISD; ZK is the zone key for a given zone.

There are two kinds of isolation context (as illustrated in Figure 6.3):

- The **native isolation context** is identified by the special context name '.'. Assertions in the native isolation context are signed by the authority for the subject name, with a signature chain rooted at the root authority for the ISD in which the assertion is made, such that the authority resides within that ISD (see also Figure 3.1 on Page 53).

- A **remote isolation context** is identified by the special context name `'isd--r-'`, where r is the number of the ISD at which the context is rooted. Assertions in a remote isolation context are signed by the authority for the subject name, with a signature chain rooted at the root authority for the isolation domain identified by the context, such that the authority resides within that ISD. Remote isolation contexts can be used to make assertions about names as seen within other ISDs.

Assertions in an isolation context are intended to be publicly available throughout the Internet. Since these assertions are made available to support connections to public services, resistance to zone enumeration is explicitly not a design goal of the RAINS protocol.

Example. The following examples illustrate how contexts work. Consider the name `'simplon.inf.ethz.ch'` in the (default) context `'.'`. This context is the native isolation context, so the signature chain is determined from the name itself, rooted at the TRC for the current ISD. If the assertion is issued by an authority in ISD 33, the chain is as follows:

$$TRC_{33} \rightarrow RZK_{33} \rightarrow ZK_{\texttt{ch}} \rightarrow ZK_{\texttt{ethz}} \rightarrow ZK_{\texttt{inf}} \rightarrow assertion$$

where TRC_{33} denotes the TRC of ISD 33, RZK_{33} its root zone key, and ZK the zone key for a given delegation.

Now consider the name `'simplon.inf.ethz.ch'` in the context `'--isd-44'`. This is the remote isolation context for ISD 44, so the signature chain is again determined from the name itself, but rooted at ISD 44's TRC, as authenticated against the current ISD's TRC, as follows:

$$TRC_{33} \rightarrow TRC_{44} \rightarrow RZK_{44} \rightarrow ZK_{\texttt{ch}} \rightarrow ZK_{\texttt{ethz}} \rightarrow ZK_{\texttt{inf}} \rightarrow assertion$$

Note that both ISD 33 and ISD 44 use the same authority for the top-level domain `'.ch'`, but the verification path depends on the initial root of trust for each ISD. Other arrangements are possible; see Section 6.5 for more.

6.3.4 Local Assertion Contexts

Isolation contexts are useful for names pertaining to services made available to the Internet at large. The basic mechanism isolation context uses – providing an alternate signature chain to the root of a namespace – can be generalized. RAINS provides for **local assertion contexts** to make the intentional inconsistency often implemented in the current DNS authenticatable and transparent.

A local assertion context is equivalent to a RAINS subject name designating the namespace within which the assertion is made. When a local assertion context is present on an assertion, the assertion is verified by following the

delegation chain from the root through the names in the context before following the delegation chain for the name. Each context is then essentially an alternate root for a new namespace. While the same effect could be achieved simply by concatenating names together, separating this information into explicit subject name and context name allows the semantically meaningful part, which should be presented to the user (the subject name), from the namespace designator, which should be user-accessible but otherwise is a matter of system configuration.

Example: Split DNS. Consider an organization that places its workstations in their own top-level namespace. A workstation named `simplon` might carry the full name `simplon.workstations`. In the current DNS, this would be achieved through "split DNS"; i.e., answering queries about the `workstations` zone only on certain networks. This arrangement, however, is operationally brittle and can lead to leakage both of queries and names beyond their intended scope.

To implement this split within RAINS, ETH Informatik could place assertions about its workstations in the local assertion context `isg.ethz.ch`, in essence creating a local root namespace containing the `workstations` zone. The signature chain for these assertions start with the name components in the context before considering the subject name:

$$TRC_{33} \rightarrow RZK_{33} \rightarrow ZK_{\texttt{ch}} \rightarrow ZK_{\texttt{ethz}} \rightarrow ZK_{\texttt{isg}} \rightarrow ZK_{\texttt{workstations}} \rightarrow \textit{assertion}$$

Note that the zone key for `workstations` above is local to `isg.ethz.ch`, unrelated to the zone key for the TLD `workstations`, if it exists.

Additional information can be placed in a context beyond the name of the local root. This additional information is separated from the authority part by a *context marker*, the special name `cx--`. Additional information in a context is used to group assertions signed by the same local root, and to provide a way to attach contextual information to queries.

Example: CDN zones. Consider a content delivery network (CDN) separating content into zones (data centers from which content is served) based on geography. It creates a local assertion context `some-cdn.com`, and places information about the zone in the additional context part: e.g., the local assertion context `zrh.cx--.some-cdn.com` names servers hosting content in a CDN's Zurich data center. A client could represent its desire to find content nearby by making queries in the `zrh.cx--.some-cdn.com`, `fra.cx--.some-cdn.com` (Frankfurt), and `ams.cx--.some-cdn.com` (Amsterdam) contexts. Note that in this case that assertions in each of these content zones will be signed by the same delegation chain `.some-cdn.com`.

Local assertion contexts can be combined with remote isolation contexts, as well; here, the remote isolation is inserted into the signature chain before the name components in the context.

Example: Combining local and isolation contexts. . Consider the name `example.com` within the context `zrh.cx--.some-cdn.com.--isd-44`, asserted within ISD 33. Here, the signature chain for the context is rooted at ISD 44's TRC, then follows the authority part of the local isolation context before looking for names in the root:

$$TRC_{33} \rightarrow TRC_{44} \rightarrow RZK_{44} \rightarrow ZK^{1}_{\texttt{com}} \rightarrow ZK_{\texttt{some-cdn}} \rightarrow ZK^{2}_{\texttt{com}} \rightarrow \textit{assertion}$$

Here $ZK^{1}_{\texttt{com}}$ is the zone key for the top-level domain `.com`, while $ZK^{2}_{\texttt{com}}$ is a local key signed by `some-cdn.com`.

6.3.5 Queries

A query is a request for a set of assertions, shards, and zones supporting a conclusion about a given subject-object mapping. It consists of the following information elements:

- **Context:** the isolation context or local context in which responses will be accepted. A query may also name a special *any* context, signifying a willingness to receive information about names in any context available at the query server.
- **Subject:** the name about which the query is made; in contrast to assertions, the subject name here is fully qualified.
- **Types:** a list of the types of information about the subject that the query requests.
- **Valid-until:** a client-generated timestamp for the query after which it expires and should not be answered.
- **Token:** a client-generated token for the query, which can be used in the response to refer to the query.
- **Options:** a set of options by which a client may specify tradeoffs (e.g., reduced performance for improved privacy).

A response to a query consists of a message containing a set of assertions bound to the token supplied by the client in the query.

When used with the RAINS client protocol, the query server performing verification may sign the entire response; this is an assertion that the query server has verified the signatures from the appropriate roots, allowing the client only to verify the query server's signature on the whole response.

6.3.6 Registrar and Registrant Assertions

The registrant object type in the RAINS data model associates civil information about a name's registrant (organization or legal personality owning an entry under a top-level domain), and in essence integrates WHOIS into the naming service. The presence of a registrant object on a name identifies that name as a registrant-level domain; i.e., a name that exists due to a contractual relationship with a domain name registrar. This integration has two advantages: first, it provides authentication of WHOIS information. Second, it allows operational decisions to be taken based on WHOIS information.

The registrar object type identifies the registrar responsible for a given name's existence. This allows operational decisions to be taken based on registrar; e.g., to block a registrar that is predominantly responsible for malware domains.

6.3.7 Augmented Assertion Authentication

To verify the authenticity of an assertion, a client or a query server can verify the signature against the delegation for the zone containing the assertion. The delegation assertion for that zone can be verified against the delegation from the zone containing it, and so on all the way back to the root delegation from the TRC for the isolation context. This delegation chain authentication has identical properties to the verification of an RRSIG in DNSSEC. It also has identical drawbacks: each level of delegation must be trusted in order to verify a name at the leaf.

RAINS provides for signatures by *external keys* on assertions, i.e., those outside the delegation hierarchy, to provide additional and/or parallel verification of the authenticity of the assertion. This facility, together with the certificate object type for storage of end-entity certificates, provides two-way integration between RAINS and ARPKI (see Section 4.3.2 on Page 89).

6.3.8 Address-to-Name Mapping

Information about addresses in RAINS is stored in a separate tree, indexed by address and prefix. An address assertion is similar to a name assertion, but is indexed by subject address as opposed to subject name, and the hierarchy of names is built upon delegation from less-specific to more-specific prefixes. Address assertions may only contain delegation, redirection, name, and registrant type objects.

Contexts are also available for address assertions, but the native isolation context may only contain assertions for SCION addresses within its ISD, remote isolation contexts may only contain assertions for SCION addresses within the remote ISD, and local contexts may only contain assertions for non- routable

addresses within the address family (e.g., RFC1918 [208] or unique local addresses (ULAs) [109]).