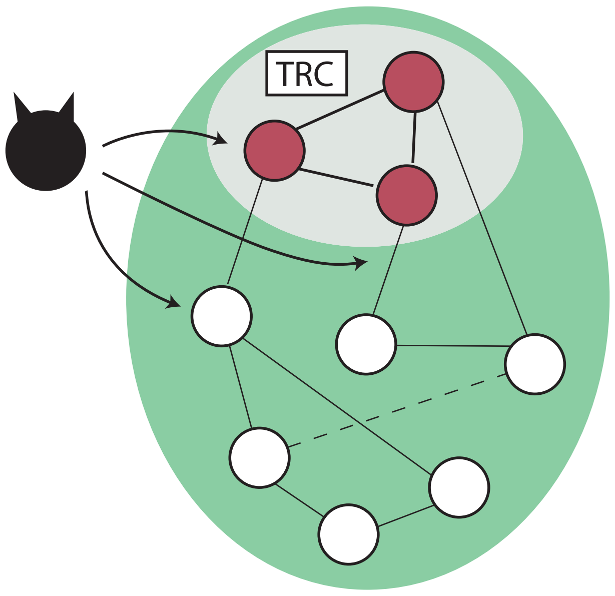
Security Analysis of a Future Internet Architecture



Benjamin Rothenberger

Master Thesis
Spring 2016

Supervisors:

Dr. David Barrera
Prof. Dr. Adrian Perrig

Department of Computer Science, ETH Zürich

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

SCION

Abstract

We perform an in-depth security analysis of the SCION future Internet architecture, which is designed with security as a core feature. We assume an adversarial position and analyse the architecture piece by piece by looking at the design and the implementation of SCION. This analysis includes attacks on infrastructure services, attacks on routing protocols incorporated in the SCION control plane, attacks thwarting confidentiality of the SCION data plane, and attacks against the availability of the SCION architecture. We propose and implement defenses for discovered vulnerabilities. To aid in identifying and defending attacks in SCION, we introduce an attack taxonomy. Further, we compare trust related issues between BGPsec respectively DNSSEC and the trust architecture chosen by SCION, and demonstrate that hierarchical, centralized trust infrastructures give parents unilateral control over delegated resources.

We show that a clean-slate design of an Internet architecture such as SCION improves multiple security aspects compared to today's Internet, and that many lessons from iterating over the design of SCION can help in the design of other architectures.

Acknowledgments

I would like to take the opportunity to thank all the people who accompanied and supported me during this project.

First, I would like to thank my thesis advisor Dr. *David* Barrera for dedicating a lot of time to supervise my work. I could profit from his research experience. He always had an open door whenever I ran into problems or had a question. Whenever I needed guidance, he steered me in the right direction.

Secondly, I would like to thank Prof. Dr. *Adrian* Perrig for his help and constructive comments, the possibility to continue researching in the area of future Internet architectures, and for investing his precious time for discussions about the project. Further, I would like to thank the group members of the Network Security group who welcomed me with open arms, specifically *Daniele* Asoni who was co-author for a part of my project.

Finally, I would like to express my very profound gratitude to my *family* for providing me with unfailing support, both emotionally and financially, and a stable environment during my studies. A mention deserves my girlfriend *Bettina*, who always found a way to encourage me and supplied me with energy during the months of work on this thesis.

Zürich, 1. August 2016

B. Rothenberger

Benjamin Rothenberger

Contents

1 Introduction

"Using encryption on the Internet is the equivalent of arranging an armored car to deliver credit card information from someone living in a cardboard box to someone living on a park bench."

Gene Spafford, *IT Security Professor, Purdue University, 2001*

The early design of the today's Internet considered the security of its protocols and architecture, but did not explicitly embed security as one of its primary features for various reasons. Since only a handful of nodes were connected to the network, the architects relied on strong accountability. In addition, the limited computing power and physical isolation of military networks made security by design appear as an unjustified cost. Decades later, the Internet scaled up to billions of users has started to show increasing concerns in terms of scalability, accountability, manageability and security. Today, not all participants of the Internet are trusted and the demands for scalability and security have changed.

The aforementioned reasons have lead to a widespread patching practice, where established protocols get updated by more secure variants. However, patching offers an undesirable long-term solution to the mentioned problems. The benefit of upgrading to a secure protocol is often scant compared to the overhead and sometimes leads to opposition from the community due to additional complexity. Some fundamental security issues on the current Internet, such as surveillance of individuals or entire masses, attacks on availability of specific endpoints or domains, and compromises in trust architectures that allow man-in-the middle attacks, cannot be resolved using patches. The problem is that even a "fully" secure Internet architecture does not lead to a perfect environment, or as Gene Spafford puts it: even if data lines are protected by strong cryptography, there exist still weak endpoint that are typically easier to compromise by an attacker. Following, researchers have investigated alternative designs for next-generation Internet architecture to overcome the patching practice [75].

Future Internet architectures (FIAs) tackle the mentioned issues by introducing alternative network structures with radically different, and largely backward-incompatible designs. One of these FIAs is SCION (Scalability, Control, and Isolation on next-generation Networks [11]), a clean slate redesign that includes security as a cornerstone. SCION tries to achieve availability even in presence of adversaries, offer transparency and control of forwarding paths, and enable trust agility. The design focuses on scalability, efficiency and also introduces new features such as multi-path communication. Further, it allows gradual deployment that can use most of the available physical architecture.

Even tough an architecture is built for security, it does not mean that the architecture automatically gains those security properties. It is important from a research and deployment perspective to analyse the security of these network architectures. The security of any of its components, including routing protocols, trust infrastructure and infrastructure services is dependent on many factors. For example, is the protocol correctly implemented? How does non-intentional misconfiguration affect the security of the system? Should an adversary that can eavesdrop on a large portion of network links be considered in a threat model? There exist a multitude of factors which may not only affect the security of a specific component, but also impact the security of another. Therefore, the security analysis should not only target independent components, but also analyse their interactions.

In this thesis, we perform an in-depth analysis of the future Internet architecture SCION. This analysis spans all architectural components, routing protocols, cryptographic primitives, and the current software implementation. We analyse the architecture piece by piece and try to find attack vectors from an adversarial point-of-view. For each of the vulnerabilities discovered, we investigate possible solutions to mitigate them and integrate those solutions into the main SCION codebase. Furthermore, we analyse differences between current trust architectures in BPGsec respectively DNSSEC and SCION. For code analysis, we employ static and dynamic software verification to identify software flaws. Finally, we discuss what characteristics a future-proof Internet architecture should have and identify how the design of SCION complies with them. Where possible, we make security recommendations to SCION and other architectures.

The results of this thesis have lead to improvements in the design and implementation of SCION, resolving many security vulnerabilities. Other suggestions made throughout the thesis should contribute a way forward to improve future versions of SCION.

1.1. Organization

The remainder of this thesis is organized as follows. First, we provide necessary background and related work about the current Internet architecture such as attacks, basic cryptography, and security problems. Chapter 3 gives a brief description of the SCION Internet Architecture and highlights all relevant details for the analysis. In Chapter 4, we analyse the security of the SCION infrastructure services, as well as the cryptographic primitives that are used in the current implementation. We illustrate attacks on SCION in Chapter 5, where we also deliver an attack taxonomy to classify the described attacks. Chapter 6 discusses trust related issues in the current Internet, the so-called adversarial kill-switches and evaluates the problem setting for SCION. Chapter 7 outlines future work and concludes.

2 Background

In this chapter we provide background on systems used in the current Internet architectures and known attacks on them. We follow up with a short description of the cryptographic primitives that will be used, and discuss security aspects of the current Internet. Readers who are already familiar with these topics may want to skip this chapter, and return to it later through references if needed.

2.1. DNS / DNSSEC

The *Domain Name System (DNS)* [71] is a global, decentralized naming system that stores mappings of domain names to network addresses. The DNS namespace is hierarchically organized as a tree whose nodes are labels of zones and whose root is the empty string (often denoted as '.'). The name of a node in the tree can be resolved by concatenating labels on the path separated by a dot [89]. For example: `inf.ethz.ch`, where `.ch` is a top-level domain (TLD), `.ethz` a second-level domain (SLD), and `.inf` is a sub domain.

All information is stored using a basic datastructure called resource records (RR). There exist different type of records that are permissible in zone files (e.g., `A` record for IPv4 addresses and `AAAA` record for IPv6 addresses). These records are logically organized in sets, the so called RRsets. Because the DNS protocol does not provide authentication or protect the integrity of resource records, an active attacker can perform a man-in-the-middle attack and inject or modify answers which in turn get accepted as legitimate answers. This is known as DNS spoofing. Further, any party with a privileged network position is able to modify DNS records to point wherever they choose. Due to DNS's caching at recursive resolvers, a single spoofed response may be sufficient to poison responses for the same RRset (see [84]).

The *Domain Name System Security Extensions (DNSSEC)* [6] aim to overcome the security limitations of DNS. It provides authentication using digitally signed DNS resource record sets (RRsets). The DNSSEC trust chain is a sequence of records that identify either a public key or a signature of a set of resource records. The root of this chain of trust is the root key, which is managed by the operators of the DNS root.

DNSSEC introduces the following new RR types:

- DNSKEY: public key which is used to sign a RRset
- RRSIG: resource record signature including a validity window and the name of the signing key's owner.
- DS: delegation signer, hash of a key used to link signed zones

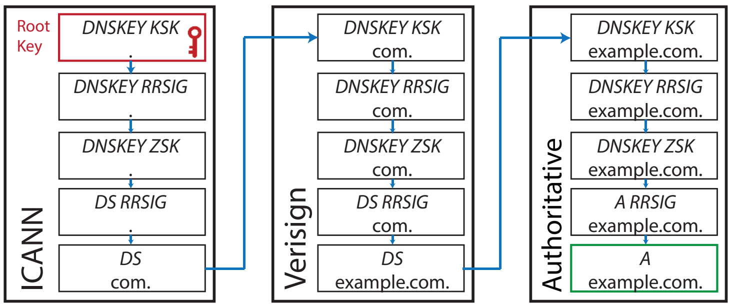


Figure 2.1.: Chain of trust in DNSSEC for example.com. *DNSKEY* entries contain a public key, which private counterpart is used to sign an RRset. The signature is then stored in a *RRSIG* record. *DS* records store a hash of a public key of the child zone and is used to link them.

- NSEC(3): next secure record, for explicit denial-of-existence of a DNS record

Each record is signed by either a Key Signing Key (KSK) or a Zone Signing Key (ZSK). The former is used to sign DNSKEY records, while the latter is used to sign all other records in the authoritative domain. To validate an RRset, a DNS client constructs a chain of trust from the root of trust to the RRset (see Fig. 2.1). If it succeeds, the information in that RRset is cryptographically authenticated [61]. The root keys themselves are stored on hardware security modules (HSM) in multiple redundant facilities with high physical protection [62]. Smartcards are necessary to activate and decrypt the HSMs, and these cards are stored in physical safe deposit boxes. *Crypto Officers (CO)* are given a physical key to these boxes. In case of an emergency situation where the COs are unable to travel to the facility, the responsible authorities can break into the boxes. If all HSMs become inaccessible, five out of seven *Recovery Key Share Holders (RKSH)* are required to reconstruct the secret backup key used for encryption of the backup HSMs [49].

2.2. BGP / BGPSEC

The *Border Gateway Protocol (BGP)* [64] is a incremental path vector protocol used for inter-domain routing in the current Internet. BGP-speaking routers send announcement messages when a new route is available, and withdrawal messages when a route no longer exists. Before advertising a route, each AS adds its globally unique AS-number to the beginning of the AS-path. Each router selects a single preferred BGP route for each destination prefix.

Among operational problems such as scalability or routing stability, BGP has major limitations to address security adequately [17]. The protocol assumes that all BGP speakers are trusted, and as such it does not ensure that maliciously acting routers use the AS number they were allocated. Routers can thus advertise prefixes from an unassigned address space or belonging to another AS; this is known as *Prefix Hijacking* [9] (see Section 2.3.3).

Securing inter-domain routing requires that IP address blocks and AS numbers used for routing advertisements are valid and that entities disseminating these advertisements are authorized to do so. The *Resource Public Key Infrastructure (RPKI)* [59] provides a trusted mapping from allocated IP prefixes to ASes authorized to originate them in BGP. It establishes a cryptographic hierarchy of authorities that sub-allocate IP address spaces and authorize their use in BGP (see Fig. 2.2). Authorities issue signed Route Origin Authorizations (ROAs) allowing an AS to originate one of their prefixes.

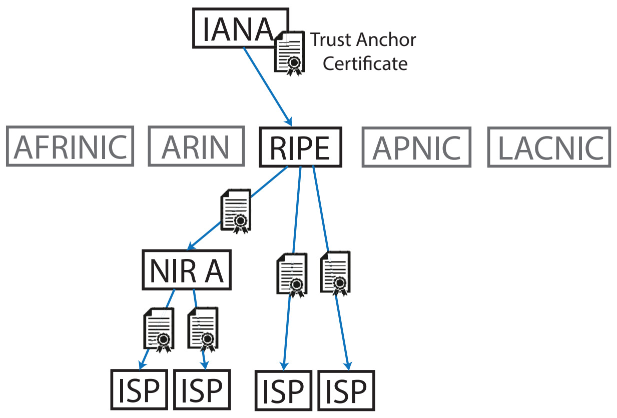


Figure 2.2.: RPKI resource allocation hierarchy.

The RPKI hierarchy is rooted at the regional Internet registries (RIRs). There is roughly one RIR per continent (RIPE in Europe, ARIN in North America, LACNIC in Latin America, AfriNIC in Africa, and APNIC in Asia Pacific). RPKI does not require any modifications to BGP message formats nor any online cryptography during routing. Each AS regularly syncs their local cache with the public repositories, locally verifies and pushes the resulting whitelist to its border routers.

While RPKI can help authenticate path origins, it cannot fully protect against some classes of attacks such as route leaks or path-shortening attacks [22]. To prevent these attacks, *BGPsec* [85] provides path integrity validation by building on RPKI. BGPsec adds cryptographic signatures to BGP messages and requires each AS to sign its outgoing BGP messages [35]. The signature covers the prefix and AS-level path, the local AS number, the AS number to which the update is being sent and includes all the signed messages received from the previous ASes on the path.

2.3. Network attacks

2.3.1. Denial of Service

Definition. "The prevention of authorised access to resources or the delaying of time-critical operations" [36].

Explanation. A *Denial of Service (DoS)* attack is an attempt to prevent legitimate users from using a service. Properties of today's networks make the execution of DoS attacks relatively simple. A typical example is a flooding attack, where the attacker effectively disables a server by overwhelming it with connection requests. A more serious type of this threat is the *Distributed Denial of Service (DDoS)* attack. An attacker recruits or uses a large number of hosts to launch an attack in a simultaneous or coordinated fashion. These hosts are compromised with a software which allows an attacker to control them (*zombies*). The attacker coordinates and triggers the zombies to attack the target in a **direct** or in a **reflected** way. The latter case adds another layer of uncompromised machines called *reflectors*. An adversary misuses these reflectors by sending spoofed packets that contain the victim's address as source address to them. The reflectors respond with packets directed to the victim's machine. Tracing or filtering such attacks is difficult because the requests come from widely dispersed uninfected machines [86, Ch. 21.10].

Attacks, as described above, can target either end hosts, network links or even entire domains. DoS attacks on end hosts increasingly target the application layer [65]. Attacks such as Coremelt [88] and Crossfire [32] generate seemingly benign traffic that traverses specific inter-AS links. These links are often high-capacity, and their saturation or lack of availability can impact large portions of Internet traffic.

Countermeasures. Even tough availability is one of the most important aspect of computer security we still lack of security mechanisms for handling this problem. Most of the mechanisms can themselves lead to denial of service.
Countermeasures are often classified by their activity level [19]. First line of defense is attack *prevention*, which attempts to either eliminate attacks or allow potential victims to endure the attack without denying services to legitimate users (e.g. provide backup resources on demand). There exists an entire industry based on content distribution networks, and cloud-based DoS mitigations. Unfortunately, even if these systems are deployed, the attacked systems may become unavailable if the adversary can generate more traffic it can handle, or can exploit a flaw or limitation in the defense's system. The second line is *reactive* behaviour during attacks. These mechanisms attempt to detect the attack as it begins and respond immediately [70]. Typically, victims must be able filter malicious requests, which is difficult in most cases, or rapidly add additional bandwidth capacity when an attack is noticed.

A prospective solution to mitigate DoS attacks on inter-domain links is SIBRA [12]. It provides a botnet-size independent way of resource allocations, that enables two end hosts to set up communication regardless of the size of distributed botnets. SIBRA is based resource reservation renewal, flow monitoring, and policing. For further details, we refer the reader to Basescu et al [12].

2.3.2. Man in the Middle (MitM)

Definition. "A form of active wiretapping attack in which the attacker intercepts and selectively modifies communicated data in order to masquerade as one or more of the entities involved in a communication" [86, p.896].

Explanation. The man-in-the middle attack intercepts a communication between two systems. The attacker secretly relays and possibly alters the communication between two involved parties who believe they are directly communicating with each other. To intercept traffic the attacker is placed either physically or logically between the communications partners. This strategy can be used against many cryptographic protocols which lack mutual authentication.
Assume that Alice wants to communicate with Bob, while Mallory wants to intercept, eavesdrop and possibly alter the communication. Alice asks Bob for his public key. If Mallory is able to intercept the message containing Bobs public key, she can replace it with her own key and claim the message was originally sent by Bob. Alice encrypts her message with the received key and sends the encrypted message to Bob. Mallory again intercepts, deciphers the message using her private key and encrypts it using the public key Bob. When Bob receives the message, he believes it came from Alice. Assuming Mallory doesn't cause any noticeable network delays, Alice and Bob have no idea that someone sitting between them is reading all of their supposedly secret communications [82, p.48].

Countermeasures. The most effective countermeasure is cryptographic authentication. If Alice and Bob check that they are in fact talking to each other, then no MitM attack can succeed unless the attacker is able to defeat the authentication mechanism.

2.3.3. Prefix Hijacking

Definition. "Illegitimate takeover of groups of IP addresses by corrupting Internet routing tables for manipulation, misdirection or interception of traffic".

Explanation. *Border Gateway Protocol (BGP)* is the dominant interdomain routing protocol. BPG works well due to its operational simplicity, despite not having any performance or security guarantees. It is an incremental protocol, which means that an edge router send announcement messages when a new route is available and withdrawal messages when a route no longer exists. To reduce the number of route advertisements and the size of routing tables all addresses are allocated in blocks. Such blocks are addressed using a prefix. Since prefixes have variable length, one IP prefix may be completely contained within another. If an IP router needs to redirect towards an address which is contained by two or more prefixes, it takes the longest prefix (more specific address block). A BGP-speaking router is not bound to its own prefix and can announce any destination prefix, including very small address blocks and address blocks it does not hold. This makes the routing system extremely vulnerable to misconfiguration or malicious attack. An autonomous system (AS) can advertise an unassigned or non-owned prefix and direct traffic towards the wrong AS. This action is known as *Prefix Hijacking* (also referred to as IP hijacking or BGP hijacking). If an offending AS simply drops all packets destined to the hijacked address the destination seems to be unreachable. This effect is called *Black Hole*. By redirecting traffic to hosts under an attackers control, the consequences can be much more severe, e.g. host impersonation or interception attacks. [17]

Countermeasures. A currently widely deployed countermeasure to bogus BGP announcements is *defensive filtering* using policies. These policies filter special documented and unassigned prefixes. An AS can also filter announcements containing private address blocks, exceptionally long AS paths and routes for small subnets (to limit the size of global routing tables). Filtering rules are fundamentally limited by heuristics and can not replace a strong security architecture. Furthermore detecting and disregarding bogus routes is challenging if the information is originated multiple AS hops away. Thus, filtering is accompanied by a *routing registry* which represents a global, shared view of routing information such as prefix ownership and routing policies. However, the registry itself needs to be secure, complete and correct, otherwise the route filters generated will not be accurate.
Other currently implemented security solutions that consider protection of BGP are limited in their effectiveness and involve other disadvantages. The most often deployed solution is *BGPSEC* which is described in section 2.1.

2.3.4. Mass Surveillance

Definition. "Mass surveillance is the subjection of a population or significant component of a group to indiscriminate monitoring. It involves a systematic interference with people's right to privacy. Any system that generates and collects data on individuals without attempting to limit the data set to well-defined targeted individuals is a form of mass surveillance" [50].

Explanation. In the past decade, mass surveillance (as an attack on privacy) has become known to the public. As document leaks have repeatedly shown, state-level adversaries are interested in monitoring, recording, and analyzing user data wherever possible, both passively and actively, mainly under the pretense of terror prevention [38]. In *passive mass surveillance*, an attacker records traffic at multiple observation points such as Internet exchange points or at physically unsecured cables (e.g., submarine communications cable). As storage costs decrease and processing decreases, collecting full communication logs and analyzing them becomes possible. *Active mass surveillance* tries to manipulate globally deployed protocols to attract traffic toward specific observation points (e.g., using Prefix hijacking as described in section 2.3.3). A recent example has been executed in China. The attackers disrupted global routing tables and channeled traffic through their ISPs for 18 minutes [2].

Countermeasures. To counteract mass surveillance, an individual should use end-to-end encryption. If an adversary wants to analyze the snooped, encrypted data, they would need to break the encryption scheme or compromise one of the endpoints [37]. Another solution is the use of anonymity networks such as the Tor network [25] to enable anonymous communication. Their use is intended to protect the personal privacy of users, as well as their freedom and ability to conduct confidential communication by keeping their Internet activities from being monitored.

2.4. Cryptography

2.4.1. Message Authentication Codes (MAC)

Definition. Message Authentication Code (MAC) algorithms are keyed-hash functions whose specific purpose is message authentication [68].

Explanation. A MAC function takes as input a key and a string of arbitrary length, which is the message that has to be authenticated, and returns a bit string of length k, which is called MAC. It must resist existential forgery under chosen-plaintext attacks (CPA) to be considered secure. This means that even if an attacker has access to an oracle which possesses the secret key and generates MACs for messages of the attacker's choosing, the attacker cannot guess the MAC for other messages (which were not used to query the oracle) without performing infeasible amounts of computation. If an attacker has modified the protected data, the verification process fails.

CBC-based MACs. The most commonly used MAC algorithms are based on a block ciphers. They make use of cipher- block-chaining (CBC). In this mode, the plaintext is padded and split into separate n-bit blocks $x_1 \dots x_t$. Letting Enc_k denote encryption using Enc with key k and $\oplus$ the bitwise xor-operation; the output blocks H are computed as follows:

$$H_1 \leftarrow Enc_k(x_1 \oplus IV), \quad IV = 0 \dots 0 \tag{2.1}$$

$$H_i \leftarrow Enc_k(x_i \oplus H_{i-1}), \quad i = 2 \dots t \tag{2.2}$$

The MAC is the n-bit block H_t [68].

2.4.2. Rainbow Tables

A rainbow table (RT) is a precomputed datastructure to reverse cryptographic hash functions. They are typically used in searching tasks (e.g., password recovery consisting of a limited set of characters). In searching tasks, one could make use of the time-memory trade-off by using less processing time T but more memory M than a brute-force attack to calculate a hash, but more processing time and less memory than a simple table lookup with one hash per entry, provided that the size of the search space is $N = T * M$. RTs form a compact representation of password sequences, called chains. Each chain starts with an initial value (start point SP) which is piped into a hash function H. The resulting value is lead through a reduction function R to possibly determine a new plaintext value. This process is iterated for specified times and the last hash value (end point EP) is stored jointly with the initial plaintext. Using more iterations results in a smaller table. For a single iteration the table contains all plaintext and hash-value combinations.

Using a single reduction function might lead to collisions, merges, and loops. They reduce the probability of full coverage and lead to unnecessary computation. This is solved using a different reduction function $R_1 \dots R_n$ at each column and makes use of all benefits of distinguished points without their limitations.

Finally, this process leads to a rainbow table as the following:

$$\begin{aligned}
SP_1 \quad &= x_{1,1} \xrightarrow{H} h_{1,1} \xrightarrow{R_1} x_{1,2} \xrightarrow{H} \dots \xrightarrow{R_n} x_{1,n} \quad &= EP_1 \\
SP_2 \quad &= x_{2,1} \xrightarrow{H} h_{2,1} \xrightarrow{R_1} x_{2,2} \xrightarrow{H} \dots \xrightarrow{R_n} x_{2,n} \quad &= EP_2 \\
\vdots \quad & \qquad\qquad\qquad \vdots & \vdots \\
SP_m \quad &= x_{m,1} \xrightarrow{H} h_{m,1} \xrightarrow{R_1} x_{m,2} \xrightarrow{H} \dots \xrightarrow{R_n} x_{m,n} \quad &= EP_m
\end{aligned}$$

To reverse a hash function, the obtained hash value is processed using the described hash-reduction sequence until the result of a reduction is found in the table. This process leads to the initial value of the hash-chain. In a second step, the initial value is also processed using hash and reduce to find the pre-image of the hash value.

2.5. Security on State of the Art Internet

2.5.1. Weaknesses of the Current Internet Architecture

The current Internet architecture yielded to some innovative design principles functionality that are implemented by endpoints. Particularly, the endpoints handle reliability through redundancy and recovery. The application state is mainly held at the end-points, which means that intermediate nodes can crash and recover without loss of the application state. Furthermore, the Internet was designed around one ’thin-waist’ (IP). This makes it easy to achieve connectivity for multiple application (i.e., implement IP), but also limits the Internet in its primary usage as an information distribution network [20].

Even tough security was considered early in the design of Internet architecture and protocols [80], it has not been embedded due to various reasons. Even if limited computing power would not have been an issue, the absence of risks (or rather understanding of risks) did not justify the deployment of cryptographic security. There was no banking business on the internet and military secrets were isolated on a separate network. The architects relied on strong accountability as only a few nodes were connected to the network. Decades later, the Internet protocols scaled up to a billion of users, but accountability did not.

Nowadays, TLS (Transport Layer Security) and its predecessor SSL (Secure Sockets Layer) are the only Internet security protocols to have achieved ubiquitous use, but even these typically only use authentication on one end (server-side) and suffer from design flaws such as dubious trust in certification authorities. Other protocols such as BGPsec and DNSSEC struggle in their deployment and becoming daily used. Apart from political issues in the security community (e.g. PGP versus S/MIME), protocols fail to address the users requirements. These requirements are not fixed and change over time.

Protocols like BGP are operated globally, running across organizational and national borders. No single centralized authority can mandate the deployment of a security solution, but instead, each AS can autonomously decide which solutions are deployed in their network. To successfully establish a new security standard becomes a coordination game among thousands of independently operated networks. Further, some security solutions which seems to be well-suited in theory suffer from substantial limitations in practice and do not work well unless a large number of networks deploy them.

2.5.2. Security Principles and Techniques for Next Generation Architectures

Security is going through a large development process in the recent years. In the following, we outline security principles and techniques that have proven their reliability [74].

Principles.

- **Privilege separation** is used to divide a structure into separate parts which are limited to specific privileges. A common method to implement privilege separation is to have a computer program fork into two processes, where a small process performs privileged actions. In a next-generation network architecture, separating the control plane makes an attacker unable to execute an attack from the data plane.
- **Privilege reduction** describes a principle where the abstraction layer of an architecture must be able to access only the information and resources that are necessary for its legitimate purpose. Each abstraction layer is therefore isolated, which improves the security of each part as it cannot be exploited by an adversary [81]. For a next generation architecture, parts of the infrastructure must be logically isolated in order to deny an attacker control over the entire domain. This principle makes it more stable and resilient to global attacks.
- **Verification of message integrity**. Each message should be cryptographically verified in order to protect it from alteration and corruption. Otherwise an on-path adversary can alter bypassing messages. This principle can be achieved using symmetric cryptography (e.g., Message Authentication Codes, where an attacker must be unable to create a forged tag using a chosen-plaintext attack).
- **Transparency** implies actions of openness and accountability. It allows a monitoring entity to observe actions, which makes the system less prone to misconfiguration. Transparency in terms of security also means that even if an adversary knows the entire system, it still remains secure (cf. Kerckhoffs principle [54]).
- **Trust agility**. This defined by giving control to each entity to select which other entities it wants to trust. These trust relations are used to validate bindings between named entities and their public keys. In information security, a component is typically validated using a chain-of-trust, whereof the root of this chain is known to the component. Providing trust agility, an entity can select or exclude the roots of trust they want to rely upon [67].
- **Easy-to-update**. As errors are inevitable in any development process, the system must be easy to update in order to fix them. It should be kept as modular as possible and provide a mechanism to safely roll-out updates.

Techniques.

- **Usage of type-safe language**. A type-safe programming language prevents type errors, which can be caused by a discrepancy between differing data types between constant, variables and function calls. Type enforcement can be static (at compile time) or dynamic (at run-time). For example, Python is a dynamically typed language and type safe as it throws exceptions to signal type errors that occur during execution [42].

- **Usage of memory-safe language**. Flaws in memory access causes major security vulnerabilities such as buffer-overflows and dangling pointers. Computer languages that support arbitrary pointer arithmetic, casting, and deallocation are typically not memory safe. Memory safety can be achieved by using automated theorem proving as part of static analysis or by disallowing the aforementioned features [90].
- **Unit testing**. The goal of unit testing is to isolate each part of the program and show that the individual parts are correct. It finds both bugs in the implementation as well as flaws in the specification of the unit, but will not catch every error in the program, because it cannot evaluate every execution path in any but the most trivial programs [46].
- **Fuzz testing and dynamic analysis**. It is a often automated or semi-automated way of testing by providing invalid, unexpected or random data and commonly used to test software for security flaws. Input that crosses the trust boundary is often the most interesting [91].
- **Frequent and signed updates**. The software should be updated regularly to patch bugs found. These updates must be signed to prevent them from being altered or corrupted.
- **Resilient API against developers confusion or misconfiguration**. A good example are AEAD ciphers, which simultaneously provide confidentiality, integrity, and authenticity assurances on the data. Similarly, an API should be built in a way, that a developer not fully understanding the logic behind a call uses it in a secure way or gets warned otherwise.

2.6. Related Work

2.6.1. Next Generation Networks

In the past years, several efforts on redesigning the Internet have been made to satisfy the newly emerged requirements of Internet-based applications. These include increased mobility, availability, routing and naming problems, etc. These new designs are generally called Future Internet Architectures (FIAs). We discuss the most important FIAs in terms of security and show their contribution.

The authors of *MobilityFirst* [78] have proposed an architecture that maps billions of identities to locations without fundamentally changing the underlying forwarding architecture of today's Internet in terms of security and availability.

Nebula [3] takes a different approach, where a sender can only send packets if an approved path to a service is available, in order to address security problems of the current Internet. The architecture assists the service in verifying that the packet indeed followed the approved path. However, this property is achieved using expensive path verification for every single packet at each router, which needs to keep a per-flow state. This high cost limits it usage to highly specialized services only.

Other recent examples such as *NIRA* [97], *Pathlets* [33] or SCION [11] are all based on the principle of source-routing, which means that, unlike in the current Internet, the source endpoint can determine the path that the packets get routed through. NIRA focuses on scalability and introduces the idea of splitting the path into a sender part and a destination part. Unlike NIRA, Pathlets focuses on flexibility of routing policies. This flexibility is achieved using simple configuration rules.

2.6.2. Classification of Attacks

To provide a better understanding of the problem behind an attack and the available solution space, the use of taxonomies to classify attacks has established. In this section, we provide references to relevant taxonomies that assist with identifying attacks.

Howard's taxonomy [45] classifies attacks by events. The events are described as attacks at a specific target intended to result in a changed state and involves the action and the target. An attack consists of five steps that an attacker needs to perform. Those steps are the tools the malicious entity uses, the vulnerability that has been exploited, the action performed, the target of the attack, and the unauthorized result.

Mirkovic and Reihner [70] provide a taxonomy of Distributed Denial of Service (DDos) attack and defense mechanisms to classify attacks and scope the defense strategies. The taxonomy is categorized by *Degree of Automation*, *Exploited Weaknesses*, *Source Address Validity*, *Impact on Victim*, as well as other smaller categories.

Hansman and Hunt [39] propose a taxonomy with four unique dimensions that provide classification of computer and network attacks. These dimensions are the attack vector that is used to classify the attack, the target of the attack, the vulnerability classification number, and the payload or effect involved in the attack. They aim at consistency in language with attack description.

Simmons et al. [83] focus on classifying cyber attacks using five major classifiers, the attack vector, the operational impact which describes high-level information about the attack, defense possibilities, information impact describing the impact on sensitive information, and the target of the attack. The authors state that their system is also able to classify "blended" attacks, for example a series of attack vectors exploited by the malicious entity.

2.6.3. Attacks on the current Internet generation

There exists a wide band of attacks on the current Internet generation targeting almost every available layer or protocol.

Butler et al. [17] consider a survey about vulnerabilities in the Border Gateway Protocol (BGP). It fails to address security adequately and frustrated past efforts on securing the interdomain routing system. The attacks range from prefix hijacking to attacks on availability using black-holing or flooding attacks. Even when the interdomain routing is secured using BGPsec, there exist still other attacks that interfere with routing availability, loop-free routing, and path predictability. These attacks are outlined by Qi Li et al. [60].

Also the naming system DNS and its secure variant DNSSEC offer attack vectors to malicious entities. DNS is vulnerable to MitM attacks as well as a range of other attacks such as cache poisoning. Ariyapperuma and Mitchell [8] present an analysis of security vulnerabilities in these protocols. DNSSEC can also be misused to perform DDOS attacks, as shown in a study by Rijswijk et al. [95]. In the worst case the authors could observe an amplification factor of over 50 compared to the original query.

Coremelt [88] and Crossfire [32] describe attacks on the availability of specific network links. The former presents an attack mechanism, where attackers only send traffic between each other, and not towards a victim host, resulting in no "unwanted" traffic. In the latter, a small set of bots directs flows to publicly accessible servers over a concentrated number of links, which effectively floods the chosen links. The attack remains undetectable for a targeted server as they affect network links.

Cohen et al. [21] focus on scale-free networks and study the tolerance of random network to intentional attacks from a physical point-of-view. They find that even poorly connected networks that are resilient to random removal of sites, are vulnerable to intentional attacks.

3 The SCION Architecture

Today's Internet suffers from numerous security and availability problems, which are undesirably solved by the current patching practice. Therefore a major redesign is necessary. This chapter describes the next generation architecture **SCION** [11] (**S**calability, **C**ontrol, and **I**solation **O**n next-generation **N**etworks), aims to solve security, scalability and availability issues which current networks suffer from. It provides a point-to-point communication framework to achieve the desired properties. This infrastructure should remain available even in the presence of powerful attackers. The authors state that, as long as an attacker-free path between endpoints exists, that path should be discovered and used with guaranteed bandwidth between these endpoints. In the following, we give a summary of SCION's key features and security mechanisms.

3.1. Overview

An **Autonomous System (AS)** is a collection of networks administrated as a single entity and internally connected using a common routing protocol. A set of ASes is logically grouped into **Isolation Domains (ISD)**, a fundamental building block of the SCION architecture, to provide transparency and a heterogeneous trust environment. An ISD is administered by one or more ASes, which form the ISD Core. Every ISD has an associated human-understandable, globally unique namespace and contains a file that defines the roots of trust used to validate bindings between entities and their public keys or addresses, the **Trust Root Configuration (TRC)**. In order to join an ISD an AS needs to accept the ISD's TRC file. Following, all ASes within an ISD agree on entities that operate the trust root and determine an internal policy. ISDs can also overlap or be hierarchically structured using sub-ISDs. Figure 3.1 shows an illustration of such a network with its fundamental building blocks [11].

Compared to the current Internet architecture, SCION is separated into smaller, logical pieces which operate independently of each other (e.g. no global root of trust). From a security perspective the authors follow the principle of privilege separation. All control mechanisms are separated from the data plane, as described in the following sections.

3.2. Control Plane

The control plane is responsible for discovering paths, making those paths available to end hosts and route the traffic accordingly. For routing SCION differentiates between intra-ISD and inter-ISD, which both use **path construction beacons (PCB)** to determine valid routes. A core AS constructs a PCB

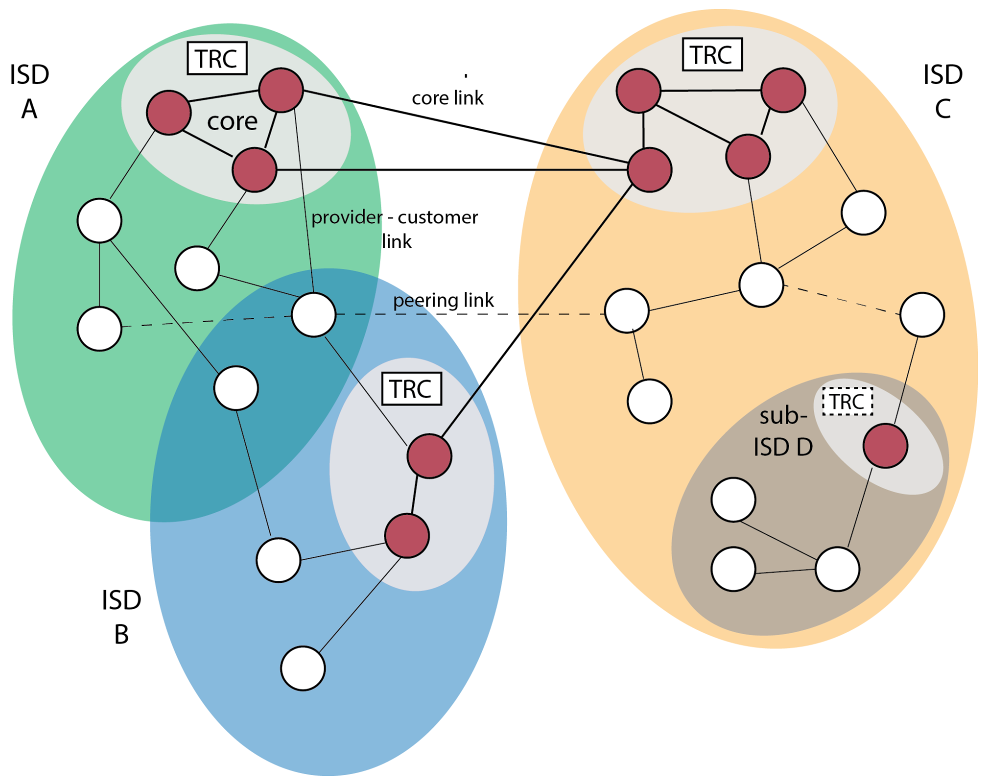


Figure 3.1.: Illustration of SCION ISDs with their core section (grey). Thick lines represent core links connecting pairs of core ASes. Dashed lines represent peering links between ASes. ISD A and ISD B intersect, while ISD D is a sub-ISD of ISD C.

and propagates it either among all other core ASes or within the ISD. By traversing ASes the PCBs accumulate cryptographically protected path information. These contents (further referenced as *opaque fields*) are chained together and create a data transmission path segment traversing several ASes. This mechanism allows ASes to learn path segments to their corresponding core ASes. Each core AS in turn acquaint information on how to reach other core ASes. Border routers do not need to maintain inter-domain routing tables because all AS-level path information is carried by the packet. The authors call this concept **Packet-Carried Forwarding State (PCFS)**. Compared to regular routing where a packet gets forwarded according to a routing table, the packet gets forwarded along a fixed path which contains all routing information.

The fundamental elements of the control plane are:

- **Beacon servers (BS)** are responsible for path information discovery and disseminate path construction beacons. Beacon servers in a core AS periodically generate intra-ISD PCBs that are sent to all other ASes within the ISD. Non-Core AS beacon servers receive such PCBs and propagate them to their customer ASes. In inter-ISD propagation, the beacons are flooded multi-path over policy-compliant paths to discover multiple paths between any pair of ASes.
  To ensure consistency between all local servers, they run a fault tolerant protocol (*Paxos* [57]) and agree on a master beacon server. Periodically, this master beacon server generates a set of PCBs (compliant with a policy) announcing the paths through which it wants the AS to be reached and forwards it to its customer ASes.

- **Path servers (PS)** store mappings from AS identifiers to sets of such announced path segments, and are organized as a hierarchical caching system. This is similar to DNS as used in the Internet. Each AS selects a set of segments through which it wants to be reachable using the master beacon server and then uploads the set to a path server in the ISD core. Path servers learn path segments by extracting content from PCBs they obtain from the local beacon servers. In case of link failure, segment expiration, or better segments becoming available, the path server keeps updating the registered segments.
- **Certificate server (CS)** assist with validating received information by caching copies of TRC files and other ASes certificates retrieved from the ISD Core. They manage keys and certificates for securing intra-AS communication. Eventually a beacon server queries a certificate server in case it does not have a required certificate available to validate the authenticity of a PCB.

Path construction. A SCION path is a combination of multiple path segments (up to 3), that are bi-directional and can be inverted. Path segments between ASes and core ISD are either called **up-segment** (*towards* core ISD) or **down-segment** (*from* core ISD to an AS). In case the destination AS is outside of the current ISD, a **core-segment** between the two ISD cores is additionally required to achieve end-to-end communication. First, the source host learns how to reach its corresponding core AS by querying the local path server. The local path server forwards the resolution request to the core path server. If the target host is outside the current ISD, the core path server queries all down-segments from the core path server in the destination ISD. In both cases the source host receives up to k up- and down-segments and if needed core-segments. Special cases apply if the selected up- and down-segment intersect at a non-core AS or a peering link exists between the two segments. In the former case a shorter path is possible by removing the extraneous part of the path. In the latter a shortcut via the peering link is possible.
After selecting a valid path, it is encoded in the SCION packet header. The destination host can respond by either inverting the path or performing a path resolution as described.

Link failures are not automatically resolved but require active handling by the end hosts. The approach chosen by SCION typically masks link failures and quickly establish new available paths. PCB dissemination is an asynchronous process and occurs every few seconds. This constantly establishes new paths (if available). Availability of every link is checked using ICMP-like messages. In case a link breaks, additional PCB disseminations are triggered. Furthermore, end-hosts use multipath communication and can switch to another available path.

Path revocation. Unlike in DNS, where revocation solely relies on TTL-based expiration, SCION also offers an explicit revocation mechanism. Removing a path should only be possible with proper authorization (apart from expiration), otherwise an adversary could simply disconnect an entire AS from the Internet by repeatedly revoking all paths to that AS. Revocation in SCION is based on AS interface revocation and works as follows:

1. In case an interface needs to be revoked by an AS, the corresponding BS informs all ERs, leading them to be aware of revoked interfaces within an AS.
2. If a packet containing the revoked interface arrives at an ER, the ER issues a revocations message which gets propagated along the reversed path contained in the packet header. Additionally, ingress ER along the path forward the message to their local PS, so that the affected paths get removed.
3. BS in downstream-ASes need to be informed about the revocation, because they should not be able to register new paths containing the revoked interface and they also need to deregister affected paths.

A revocation message needs to be authentic and only issued by an authorized party, while being ef-

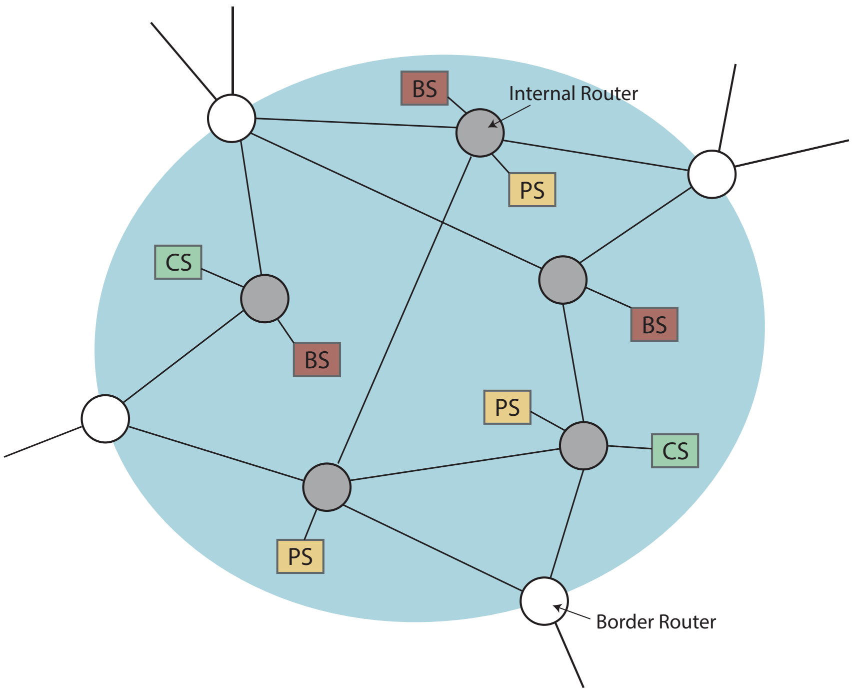


Figure 3.2.: Magnified view of an AS with its routers and servers. All servers are replicated to ensure high availability. BS = Beacon Server, CS = Certificate Server, PS = Path Server

ficiently verifiable. Thus, the SCION revocation mechanism builds on cryptographic hash-functions (e.g., *SHA-256*). Precisely: $y = \mathcal{H}(x)$, where $\mathcal{H}$ is a publicly known cryptographic hash function $\mathcal{H} : \{0,1\}^* \to \{0,1\}^k$, x is a preimage of y and a so called *revocation token*, and y the interface ID. To revoke an interface, an AS can simply reveal x. Every affected party can efficiently check whether x is indeed a valid preimage of y. Due to link versioning, each new revocation message for an interface needs to also revoke all previous versions of that interface. To solve this issue, the system makes use of hash chains [43].

The revocation messages disseminate quickly through the network. Interestingly, the dissemination performance is largely affected by the request frequency of the ASes. This means that as soon as an AS wants to use a revoked path it will in-turn receive the revocation message. Following, the dissemination performance is lower bounded by the propagation speed of the network and upper bounded by the lifespan of a path [43].

3.3. Data Plane

After discovering end-to-end paths, the data plane is responsible for packet forwarding using the given paths. Border routers forward packets according to the path specified in the header. Only the destination border router needs to inspect the packet header to forward the packet to the right AS. During the forwarding process, the router first checks that the packet arrived through the correct ingress interface. The next hop is defined through the egress interface.

An important aspect of this design is the separation of network locator (path to target) and the identifier (destination address). A network domain can choose an arbitrary addressing format for its hosts as long

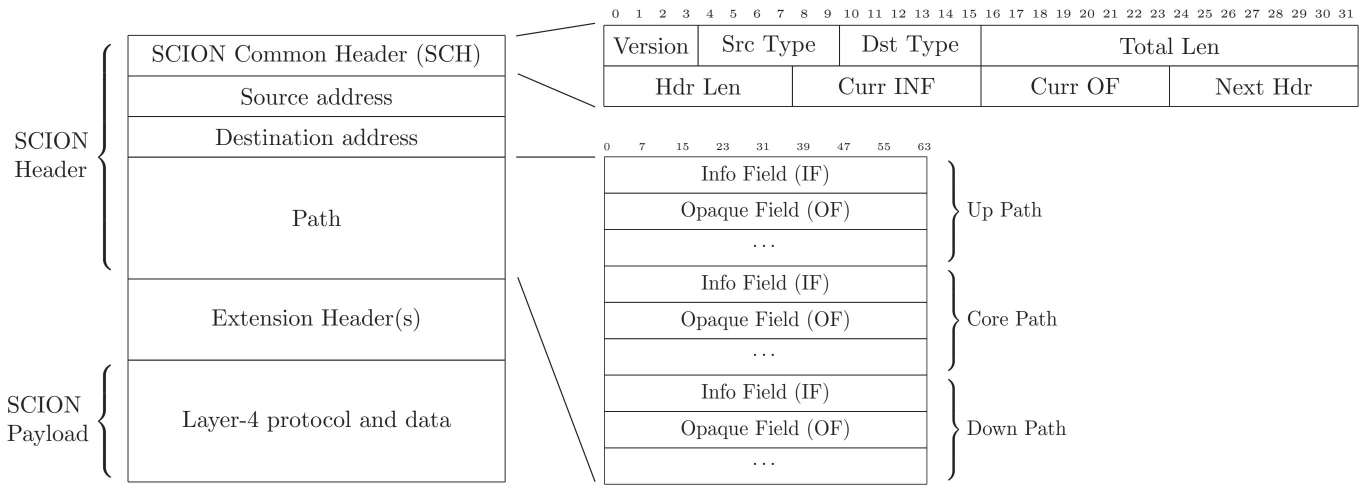


Figure 3.3.: SCION packet details. On the left side, there is high level layout of SCION Packet. The SCION Common Header (SCH) and the SCION Path are shown in detail.

as their border routers understand where to forward the packets [11].

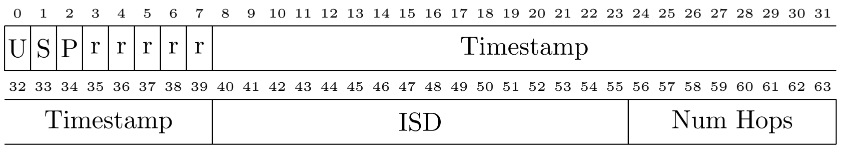
SCION packets (see figure 3.3) are structured as follows:
SCION usually encapsulates the payload using any transport layer protocol (such as TCP or UDP) and a *SCION header*. This header consists of a mandatory *SCION common header (SCH)* which encodes the most important information such as packet length, type of addresses and the current position in the path (pointers to current field in path). Immediately after the SCH, the source and destination addresses are placed. If the packet's context is unambiguous without address (e.g., forward packets on loopback), source and destination addresses are optional. Next, a packet contains a path consisting of maximally three parts (Up-, Core- and Down-Path). None of the segments is mandatory and a packet with an empty path is sent locally within an AS. Each path segment is lead by an Info Field (IF, see figure 3.4a). It identifies the type of the path and contains information required for opaque field validation such as the length of the corresponding path segment and the identifier as well as the timestamp of the ISD which initiated propagation of the path. Then, the path segment contains a sequence of Opaque Fields (OF, see figure 3.4b). There exists an OF for each AS traversed on the path and it contains information required by border routers for packet forwarding such as the ingress and egress interface, an expiration time and a 3-byte message authentication code (MAC). The expiration time expressed by the 1-byte value is relative and an absolute expiration time is computed as follows (in *seconds*):

$$\text{Timestamp} + ((1 + \text{ExpTime}) \cdot \lfloor \frac{24 \cdot 60 \cdot 60}{256} \rfloor) \qquad (3.1)$$

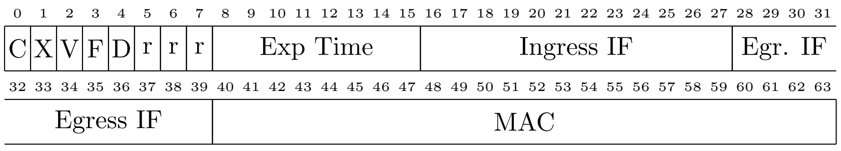
Thus, the minimal lifetime of an OF is around 5 minutes, whereas an OF could be maximally valid for 24 hours.

3.4. Security Aspects

Authentication mechanisms. To mitigate malicious behaviour SCION uses several authentication and encryption mechanisms. Many of them are based on the assumption that modern block ciphers



(a) Info Field (IF)



(b) Opaque Field (OF)

Figure 3.4.: SCION Path Fields in detail.

(such as *AES*) enable faster computation than a memory lookup from DRAM. All mechanisms support algorithm agility and can be replaced independently within an AS.

The entity authentication is based on traditional certificates. Certificates bind identifiers to public keys and contain digital signatures which can be verified by a root of trust. In the current internet architecture participants have to mutually agree on either a single (*monopoly*) or several roots of trust which are all equally and fully trusted (*oligarchy*). Using isolation domains the trust environment is split up into multiple sections. Each section can define its own root of trusts using TRC. Each of these configurations contains a version number which allows efficient certificate revocation. Entities can also specify how many signatures are needed for each operation.

To authenticate the routing layer of SCION, each AS signs a traversing PCB before they forward it. This allows validation by all entities. By cryptographically protecting the path information stored in PCFS path correctness is guaranteed. However, signature verification prohibits efficient forwarding, so a symmetric secret key is shared among beacon servers and border routers to compute a Message Authentication Code (MAC) over the forwarding information. The information for each AS consists of the ingress and egress interface, expiration time and the MAC and is stored in an 8-byte structure called **Opaque Field** (OF). The OF's are chained together and a bit flag stores the verification direction. For service authentication additional entities are taken into account to ensure higher security. Log servers keep a public log of all certificates to monitor CA's operations. To verify the consistency of this log servers efficient gossip protocols are used. Clients randomly exchange short information about the logs so that misbehaviour eventually will be detected [11].

Availability. Availability in the presence of adversaries is an exceedingly challenging property to achieve. As SCION ASes must deploy certificate, beacon, and path servers, these servers are a critical factor for availability, as otherwise path construction, verification and dissemination cannot be guaranteed. The SCION infrastructure should therefore be highly distributed and *replicated* to guarantee high availability even when single parts of the system fail or are under attack, e.g., during a DDoS attack. All server instances in an AS connect to the consistency service of that AS and thus become group members. If an instance becomes unavailable for any reason, this affiliation vanishes. When an instance joins the AS, it tries to become a leader (master). When the leader terminates, a new election process takes place. SCION's current codebase uses a distributed consistency service based on Apache Zookeeper [4].

Another feature of SCION to increase the availability of ASes is *multipath communication*. SCION allows ASes to make use of multiple paths simultaneously and requires an adversary to flood all upstream links of a victim AS in order to successfully affect availability.

3.5. Summary

This chapter presented a brief summary of the SCION design and elements of its infrastructure. SCION is a clean slate future Internet architecture that focuses on security, availability and transparency. It is still in its infancy in terms of development and deployment, but offers significant advantages over the current Internet. The control- and data-plane are separated in order to allow path control and make sure that packet forwarding cannot be influenced. Isolation Domains (ISDs) allows logical grouping and isolation of multiple ASes based on the control plane. It also enables scoped trust for entity authentication and provides transparency for path selection and packet forwarding. As a consequence of transparency, entities know who is accountable for incorrect path construction and message propagation, which might improve the recovery from attacks. More details about SCION can be found in [11], and on the SCION website[1].

Although SCION is the result of multiple years of research, attacks against the architecture might still exist. An implementation of SCION is in progress. Vulnerabilities as in any other software project are likely to exist also in this implementation. We analyse and discuss the security of SCION in the subsequent chapters.

[1]SCION website: `https://www.scion-architecture.net/`

4 Security of the SCION Infrastructure

In this chapter we analyse the security of the SCION infrastructure, specifically the end hosts and infrastructure services. First, we define a threat model as a basis to formalize the capabilities of an attacker. The architecture hence needs to provide mechanisms to circumvent these malicious elements. Then, we discuss software verification as a tool to detect vulnerabilities in the SCION codebase, and based on our findings suggest the testing process could be integrated into the SCION infrastructure software development process. We follow up with analysing the use of cryptographic primitives in SCION, as well as availability of SCION infrastructure services.

4.1. Threat Model

In the threat model we assume the following capabilities of an attacker targeting the SCION infrastructure:

The design of the SCION architecture is publicly available and described in detail in various papers [99, 11]. The implementation of SCION is soon to be released as open-source and freely available. This follows Kerckhoffs' second principle, meaning that a system should still be secure even if the enemy has an exact copy of it [54]. We assume that the attacker has full access to the source code, and can therefore identify possible flaws in the implementation. He can run own SCION nodes and infrastructure services (e.g., a beacon server), incorporate entire ASes and participate in a network based on SCION. The adversary is also aware of other publicly available exploits to compromise end hosts (e.g., in exploit kits like *Metasploit* [69] or *Angler* [44]). By exploiting an end host, the adversary gets full control over that host and learns all private keys stored on it.

When discussing cryptographic primitives, we assume that the adversary is computationally bounded (i.e., has limited computing power) and has no efficient way of breaking asymmetric cryptography (e.g., using quantum computers). For cryptographic hash-functions, it is hard for an attacker to find a preimage given a hashed value nor is he able to find a collision, due to the collision-resistance property [68].

A passive attacker is able to eavesdrop and intercept messages that have been sent over the network and bypassed one of his observation points. A more powerful, active attacker has control over the network, can modify or replay messages, and actively perform man-in-the-middle attacks.

4.2. Software Verification

The goal of software verification is that software satisfies all expected requirements. To perform verification two complementary techniques exist, *static* and *dynamic* analysis. The former is useful for proving the correctness of a program, while latter is often used to find implementation flaws and errors that have been overlooked by human testers.

4.2.1. Static Code Analysis

Static code analysis tools are designed to analyse source code and/or compiled versions of code in order to help find security flaws. Ideally, such tools would automatically find security flaws with low false positive rate. However, this is beyond the state of the art for many types of application security flaws. Thus, such tools frequently serve as aids for an analyst to help them zero in on security relevant portions of code so they can find flaws more efficiently, rather than a tool that just automatically finds flaws.

As of writing, Fortify Static Code Analyzer (SCA) by Hewlett Packard (HP) is the most advanced and used software in the area of static code analysis for security purposes [31]. To test the SCION implementation for flaws, we used HP Fortify SCA and Rough Auditing Tool for Security (RATS) as an aid. The former classifies the vulnerabilities according to a taxonomy for security defects in source code[1] (e.g., API abuse, Input validation, Security features, etc.). Unfortunately, our license did not allow access to the python module, and we focused the analysis on the C++ code (SSP, dispatcher, HSR). The latter focuses (among others) on calls to vulnerable library functions and bad practices that can lead to buffer overflows.

In the following, we list the most important vulnerabilities found in the SCION codebase.

Memory leaks. Our analysis found incorrect management of memory allocations that can lead to a resource leak (e.g., missing “free” after a “malloc” call in C++). Other possibilities include a failure during a resize of an allocated block, which leaks the original block. The code testing revealed small memory leaks in the SCION implementation (e.g., dispatcher). For a long-running applications such as a router firmware, even small leaks eventually lead to resource exhaustion. A long-term run of the default SCION infrastructure on a single machine lead to a crash after 48 hours runtime.

Buffer Overflow using a Format String. If a program uses an improperly bounded format string, it allows an adversary to write outside the bounds of allocated memory, a so called buffer overflow. This behavior could corrupt data, crash the program, or lead to the execution of malicious code. Buffer overflows are well-known and still occur commonly.
A simple example (not SCION-specific) is:

```
snprintf(buf, sizeof buf, arg);
```

which prints into a string with length checking. The third argument is a format string which we are able to control. Through supplying the format string the attacker is able to control the behaviour of the format function. A simple attack using format string vulnerabilities is to make the process crash. This can be useful for some things, for example to crash a daemon that dumps core and there may be some useful data within the coredump. In some network attacks it is useful to have a service not responding, for example when DNS spoofing. If we can see the output string, we can gather useful information about what our format string does and how the process layout looks like. All buffer overflows using format

[1]see https://vulncat.fortify.com/en/vulncat/index.html

strings that we identified in our analysis, have turned out to be false positives and did not affect code security.

False Positives. A false positive is a result that indicates a given condition has been fulfilled, but turned out that it actually has not been fulfilled. This happens frequently in static code analysis and is the main reason behind human involvement in an analysis. For example, a possible memory leak due to a missing "free"-statement could also be intentional as it is needed by design (e.g., pointers that get reused very frequently), or a reported buffer overflow that cannot be exploited by an adversary.

Other vulnerabilities. Static analysis tools are capable of detecting other flaws in a program than the aforementioned. It can detect the use of insecure function such as `strcpy`, which cannot be used safely, dead code that never will be executed, or insecure use of cryptographic functions (hardcoded salt/seed, insufficient key sizes, etc.). These have not been observed in the SCION codebase.

Finding 1

Static code analysis contributes to a secure codebase, but is not enough to claim that code is secure. Using these tools, apart from the overhead of false positives, we were able to detect several issues of medium and low severity. These have been fixed upstream.

4.2.2. "Franken"-Packets

Dynamic testing is an often automated technique using invalid, unexpected, or random data to reveal unexpected behaviour such as failing assertions, crashes, or memory leaks in software. This technique is often called *fuzz testing* and commonly used to test for security problems. The approaches are categorized into *mutation-based* and *generation-based* forms. The former fuzzers mutate existing data samples to create test data, while generation-based fuzzers define new test data based on models of the input [91]. Both variants can only test a subset of all possible inputs, as the underlying space is huge, i.e. 2^l where l is the length of the packet. Dynamic analysis often detects only simple bugs. However, security testing benefits from fuzzing because it can find odd oversights and defects which human testers would fail to find.

In "Franken"-certs [15], the authors present a mutation-based technique to generate valid test inputs in a huge underlying input space consisting only a tiny fraction of valid test cases. They test available SSL/TLS implementations using synthetic certificates that are randomly mutated from parts of real certificates and thus include unusual combinations of extensions and constraint. To address the problem of interpreting the test they use differential testing. Differential testing discovers discrepancies by comparing different independent implementations with each other. As there only exists a single SCION implementation, differential testing cannot be used. However, the process of random mutation can be applied to the SCION packet structure, resulting in "Franken"-packets. Apart from *random mutation*, we also used the following modifications for fuzz testing:

- bit flips: flip a single or multiple bits of a packet without taking the packet structure into account.
- small modifications of values: each packets is structured in multiple fields of different type. This fields can be modified (e.g., by incrementing an integer).
- negative values: are often unexpected as input and trigger unexpected behaviour in the implementation.

- boundary values: represent the maximal and the minimal value a type can have (e.g., the maximal value for an 32-bit integer is 2’147’483’647).
- crossing boundary values: are tested by taking a boundary value as before and incrementing respectively decrementing it to create an invalid value for the particular type. It is expected to result in errors, but often also reveals interesting behaviour.
- swap/replace entire OFs: to test the chaining of OFs, we tried to replace OFs and monitored its outcome.

For the testing, we modified an existing end-to-end integration test and recorded packets on multiple end hosts. The packets are then mutated and re-injected onto the network link. In case we modify integrity protected fields, we expect the packets to get discarded after the verification takes place. For non-integrity protected data, we expect the SCION implementation to remain stable, even in presence of an adversary, misconfiguration, or hardware failure. However, as the implementation is still in an early development phase, crashes are likely to be observed.

Simple modifications in some cases resulted in exceptions thrown at the target infrastructure element, which in turn made them stop working. The problem is that error handling is not yet implemented SCION, but will soon be deployed. We confirmed that our franken-packets should not permanently crash the infrastructure.
Modifications of OFs, stored in the path section of the packet, resulted the packet being discarded. Swapping entire OFs also made verification at the affected hop fail. This behaviour is as expected. By removing OFs, we could reconstruct the path truncation attack (as described in Section 5.4.5), where the packet did not reach its intended end-AS.
Unexpected values, such as negative or boundary crossing values, made packet parsing fail and always resulted in a crash. In part due to our findings, packet serialization code will be replaced by Cap’n proto (capnp) [18]. These measures will significantly reduce the number of crashes by elements of the SCION infrastructure and improve the stability of SCION implementation.

We noticed that dynamic testing only leads to meaningful and interesting results for software that is expected to run stable. Otherwise almost every change eventually results in an exception or crash and makes the interesting results hard to discover. The proposed improvements will make the implementation more stable, and reveal more bugs in a re-run of dynamic testing.

Finding 2

As of writing, the SCION codebase is not sufficiently stable to perform an in-depth dynamic analysis. Once the code matures, these dynamic tests should continue to be performed.

4.3. Use of Cryptographic Primitives in SCION

4.3.1. Key Derivation and Cryptographic Hash Functions

Key Derivation Functions (KDFs) are used to derive cryptographic keys from a shared random string or an entropy source. The KDF takes an imperfect input, sourced of semi-secret randomness, and outputs a pseudo-random key. Naive algorithms such as `SHA1(String)` are not resistant against brute-force attacks (e.g., using *hashcat* [40]).

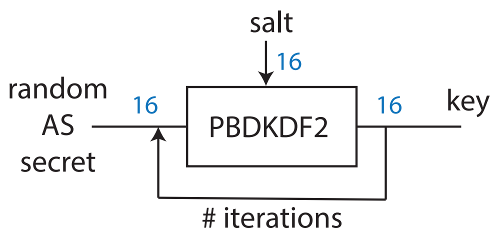


Figure 4.1.: Password-Based Key Derivation using a pseudo random function and a salt.

Passwords are still a main mechanism for humans to interact with cryptography and often the only available secret (with poor randomness and low entropy). A good password hashing function must be tunable, slow, and include a salt to mitigate brute-force attack. For this purpose, NIST and ISO standardized *Password-Based Key Derivation functions (PBKDF)* (see NIST SP 800-132 [92] and ISO/IEC 1770-6 [51]). PBKDFs are based on any secure pseudo random function (PRF). The PRF is iterated a specified number of times, which is used to increase the workload of dictionary attacks. It should be as large as possible whilst ensuring the compute time is not unnecessarily long [28].

Current State. As of writing, the SCION code randomly generates a 16-byte password string (using pycrypto.Random), encodes it using base64 and stores it in the per-AS configuration file. This operation is executed when a topology is generated and stored statically (no roll-over). The key derivation function uses default settings with a hard-coded salt and returns a 16-byte key. As a PRF `HMAC-SHA1` with 1000 iterations is used (see Figure 4.1).

Analysis. Using the PBKDF2 wrapper is a secure way to generate a key from a low entropy password, but it depends on its configuration (salt, PRF, number of iterations).

As a second input, the function takes a salt, which should be generated with a secure random number generator to allow the generation of large key sets from a single password. The chosen salt could be stored in the configuration file along with the password. As a benefit, this makes it configurable. The configuration file is assumed to be secure (otherwise the key gets compromised anyway). Furthermore, the keys are reconstructible based on this file if the it needs to be replicated among multiple servers (e.g. every border router in SCION).

Per default, `HMAC-SHA1` is used as a PRF in PBKDF2. `SHA-1` is considered as deprecated and theoretically attackable [96]. All major Internet browsers scheduled its disappearance by the end of 2016. As a successor, NIST recommends using a hash functions of the family `SHA-2`, which contains two similar hash functions with different block length (32-bit words versus 64-bit words). They are considered to be secured and implemented in most libraries.
As a result of a competition in 2012, `SHA-3` (formerly known as Keccak) has evolved. It supports the same hash lengths as `SHA-2`, but its internal structure differs significantly from the rest of the SHA family. A future Internet architecture should use state-of-the-art hash functions; ideally this is `SHA-3`. `SHA-3` still keep good software performance. According to the official released data, `SHA-3` achieves around 12.5 cycles per input byte on Intel Core 2 CPUs (SHA-3 hash output is 512-bits).

Despite the ability to adjust the number of iterations it is still possibly to implement dictionary attacks relatively cheaply on GPUs (compression functions as used in HMAC can be nicely implemented in parallel on a GPU). An optimization allows reduction of PBKDF2-HMAC calculations by a factor of two. PBKDF2 calls HMAC for each iteration. Each run of HMAC normally involves two hashing operations. A hashing operation involves two compressions. For the same data, two of those compressions are

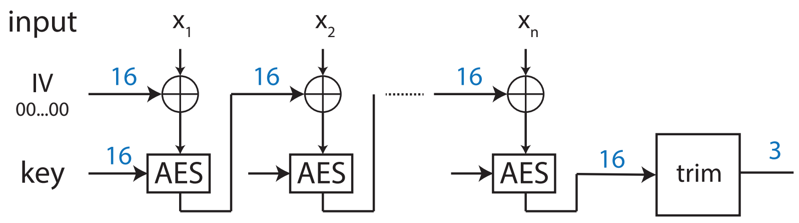


Figure 4.2.: CBC-MAC with IV and trim to 3 byte output.

going to result in exactly the same. To reduce the computation needed, an adversary can compute the compression only once and store the result. A defender needs to take this speed-up into account when selecting a certain number of iterations [34].

However, the attacker has no possibility to observe the output of a PBKDF2 as it is assumed to be secret. He would need to compromise a host, or try to reconstruct the key by reversing a hash functions, e.g. using a rainbow table as described in Section 2.4.2.

Other functions such as *bcrypt* [77] (based on the blockcipher Blowfish) and *scrypt* [76] also provide greater resistance to respective attacks. They are memory-bounded and need additional memory to perform an extensive attack. However they are less commonly used than PBKDF2, especially when a certification is needed.

Finding 3

To derive a key from a password, PBKDF2 with a random salt, `HMAC-SHA3` as a PRF and a large number of iterations should be used. As of 2016, a minimum 50'000 iterations (increasing every year) are suggested for new keychains.

As of writing SCION is lacking a key rollover mechanism. Keys are stored statically and generated every start-up. A simple scheme would include a long-term master key which is used to derive multiple short-term keys (including reserve keys).

4.3.2. Security of Message Authentication Codes

Current State. SCION uses CBC-MAC with underlying AES encryption and a key-size of 128 bit. The Initialization Vector (IV) is chosen as 16 null-bytes. The output (last 16-byte block) is trimmed to a 3-byte MAC, which is used to provide message unforgeability for an Opaque Field. Formally, the keyed hash function $\mathcal{F}_k$, where k is a fixed key of size n, maps a variable length input $\{0,1\}^*$ to a fixed length output $\{0,1\}^m$, where m is the output size. For the entire function (including the trimming) the size of the output is 24 bits.

Existential forgery. Given a secure block cipher, CBC-MAC is considered to be secure for fixed-length messages. For variable length messages, an attacker that knows the correct message-tag pairs (m,t) and (m',t') can generate a third message m'' whose tag will be t' by XORing the first block of t and then concatenating m with this part, i.e. $m'' = m \mid \left[m'_1 \oplus t \mid ... \mid m'_n\right]$. This is often called *existential forgery* [68]. Appending a message size block in the end does not help with in the mentioned case.

The SCION architecture uses two different sizes of Opaque Fields, i.e. 8 byte and 16 byte. This makes the current implementation vulnerable to length extension attacks as described above. There exist simple solutions to make CBC-MAC secure against variable-length attacks. The most standard way is to prepend the message length [13]. A more SCION-specific solution would include a reserved bit to represent whether the Opaque Field was of regular or double size. Obviously this does not scale (e.g., if in a later version another OF size is available). Apart from that, an AS could also use a different key for each of the OF sizes (input-length key separation). This scales better as multiple keys can be derived from a single master key. The National Institute of Standards and Technology (NIST) recommends using other modes such as CMAC or OMAC that are by design not vulnerable to length-extension attacks [27].

Finding collisions. Unlike cryptographic hash functions, MAC functions in general don't provide collision resistance (e.g., if the key gets leaked, it is trivial to find a collision), but instead (typically) provide message unforgeability. For high security levels, the output size of the function should be equivalent to the block length of the underlying block cipher (i.e., 128-bit). Shortened MAC outputs may be secure if the length that the MAC key is relatively short-lived. Thus the MAC can only be verified for a short length of time.

The *birthday attack* as described by [94] is worth noting when discussing the security of a PRF. If E is a block cipher with output-length l then there is an inherent limit to its quality as a PRF, namely that security vanishes as the adversary asks about $2^{l/2}$ queries. This is regardless of key-size of E [13].

Mathematically speaking: given a function f, the goal of the attacker is to find two different inputs x_1, x_2 such that $f(x_1) = f(x_2)$, which is called a collision. Assume that function f leads to H different outputs with equal probability. From a set of H values, we choose n values uniformly at random (including repeated values). Let $p(n, H)$ be the probability that at least one value is chosen more than once:

$$p(n; H) \approx 1 - e^{-n(n-1)/2H} \approx 1 - e^{n^2/2H} \tag{4.1}$$

Now, let $n(p; H)$ be the minimal number of values, such that probability of finding a collision is at least p. By inverting equation 4.1, we get:

$$n(p; H) \approx \sqrt{2H \ln \frac{1}{1-p}} \tag{4.2}$$

For $p = 0.5$, we get $n(0.5; H) \approx 1.18\sqrt{H}$. Therefore an attacker needs about $\sqrt{n}$ attempts to find a collision with probability $p = 0.5$ for a MAC of length n. This is called the *birthday bound* [82].

In case of SCION, there exist about $H \approx 1.7 \cdot 10^7$ combinations. Therefore the attacker will find a collision after evaluating the function for about 5120 different arguments on average. In practice, SCION offers no oracle to evaluate the function f for different arguments, because an external attacker cannot influence the content of an OF before it gets MACed. Furthermore, there should exist about 2^{40} different plaintexts which are distributed uniformly at random and ideally map to different MACs. This does not hold in practice, because the input is relatively static for a given path (finite number of interfaces and fixed expiration time). However, an on-path attacker can record bypassing message/tag pairs, and will find a collision after recording $\sqrt{n}$ different pairs with the same timestamp. Following, an attacker is not able to arbitrarily generate a valid OF, but replaying a colliding OFs should still be possible.

Other aspects. The IV for CBC-MAC must be chosen fixed (unlike in CBC encryption). Otherwise the random IV must known to the verifier, and possibly read from an unsafe location which can be influenced by an attacker. This allows him to modify the entire first block, because the first step of CBC-MAC is to XOR the IV with the message. The SCION codebase uses a hard-coded, fixed IV and meets this requirement.

As CBC-MAC uses an underlying block cipher, the input must be padded to a multiple of the block length. To satisfy the resilient API constraint (as mentioned in Section 2.5.2), the library function should check the input size. As of writing, padding and checking is done outside of the library function.

Finding 4

Using CBC-MAC with underlying AES encryption is only secure for fixed-length messages. SCION needs two different keys to support the two available OF sizes. As SCION wants to include variable length OFs in a future release, the use of **CMAC** would be a long term solution.

24-bit MACs are too short to prevent forgery attacks. Even if the attack can only be done online, the visibility of the attack is dependent on the number of packets process by the victim. Due to the birthday bound, an attacker is likely to observe collisions for different inputs resulting in the same MAC.

4.3.3. Symmetric Ciphers

Current State. The current implementation of SCION uses AES as a symmetric cipher. As of today, AES is widely adopted and supported in both hardware and software. Intel CPU processors support the AES-NI instruction set (such as AESENC, AESDEC, etc.) for one round AES encryption and decryption. This increases the performance.

Analysis. Until today, no practical cryptanalytic attacks against AES has been discovered [23]. However, attacks not only target the encryption algorithm, but also focus on the implementation and ways to bypass the algorithm such as side-channel attacks or attacks against the key generation system (e.g., bad random generators). To allow a degree of "future-proofing" against progress in the ability to perform exhaustive key searches, AES has built-in flexibility of key length. NIST recommends a minimal key length of 128-bits to be secure against cryptanalysis attacks until 2030. For high-security data, a 256-bit key is recommended [10]. As keys in SCION are relatively short-lived, the former choice is sufficient. Further, AES is a good choice for the mentioned reasons above.

4.4. Service Availability

To affect service availability, an adversary might not only target the processing power of a server respectively bandwidth towards server, but also exploit programming flaws in an application. This allows him to crash processes, or perform a resource exhaustion attack on the server itself (e.g., redirecting all outgoing to the listening interface).

Further, the SCION infrastructure services might get overwhelmed by an adversary that issues queries very frequently. Running multiple, coordinated instances does not help with this issue, but rather makes it worse as synchronization also consumes processing power of other nodes. To avoid such misbehaviour, the number of paths that are available on a path server could be limited and the rate of new paths that can be registered must be proportional to the steady available throughput, so that each client can only register a certain amount of paths within a given timespan. Queries to beacon servers could be rate-limited by the number of calls to expensive operations they issue. Using this technique, slow loris attacks, that try to open and main as many connections to a service as possible, can be mitigated.

To identify expensive operations, one could use profiling and monitor how much time is spent in each part of the code. However, the simulation must be run on hardware that offers hardware support for cryptographic ciphers. Otherwise these results will get distorted. By combining the previous results with the number of evaluations, costly code parts can effectively be identified. They should be rate limited to mitigate overwhelming by a malicious entity.

4.5. Summary

In this chapter, we have shown how static and dynamic code analysis can be used to improve the security of the SCION codebase. We were able to identify medium severity flaws, that have been fixed upstream. Further, we analysed the use of cryptographic primitives in SCION and based on our findings made suggestions to future-proof SCION.

Finally, we discussed the service availability of the SCION infrastructure services. Rate-limiting expensive function calls can mitigate overrunning attacks by a malicious entity.

5 Attacks on the SCION Architecture

In this chapter we describe the effect of attacks that are possible on today's Internet, or become possible in SCION's architecture. As a methodology, we assume an attacker's point-of-view and emulate a real world adversary to analyse the design specification and source code of SCION, similar to an adversary that has access to an exact copy of SCION. For each attack, we describe how SCION can either defend against it, or can be modified to defend against it. The focus of the chapter mainly lies on the SCION control plane, that is responsible for routing and packet forwarding. From a security point of view, an adversary should be prevented from redirecting traffic. For a full security treatment, we also consider attacks on the data plane, availability, and misconfiguration, as it might lead to traffic misdirection through an unauthorized ASes.

First, we define a general threat model that describes the attackers capabilities. We follow up with a proposal of criteria on how to classify attacks on SCION and evaluate known attacks on the current Internet architecture. Finally, we discuss attacks on the SCION control plane as well as the data plane.

5.1. Threat Model

As a basis, we assume the **Dolev Yao** model [26]. The model formalizes the capabilities of an attacker. In the underlying environment model we assume that there exist hostile participants. The adversary can not only passively eavesdrop messages, but also actively tamper with the communication, i.e. drop, delay, or alter packets that it should forward, or inject packets into the network. All hostile nodes are consolidated as a single attacker which implies that they share a channel for information exchange outside of the current network. However, the adversary is computationally bounded, which means that he cannot break asymmetric cryptography nor gain access to private keys of any other non-malicious node.

We assume the adversary is able to register as an AS with the ISD core and perform regular operations. However, we expect core registration operations to be throttled. There should be a mechanism to prevent large number of ASes from joining the ISD rapidly or automatically. We expect that for each new AS registration, some amount of due diligence and verification takes place by the core.

If an AS acts maliciously, we assume that the adversary can eavesdrop on all control and data messages traversing the AS. By compromising an AS, the adversary learns all keys and settings, observes all traffic that traverses the compromised node, and is able to control how the AS behaves including redirecting traffic, fabricating, replaying, and modifying packets. Data signed by the AS will successfully verify until the certificate for that AS is revoked.

Assuming the previous capabilities, the goal of the adversary is to successfully introduce a modification to the control plane, such as traffic attraction to flow through network elements under its control, passively wiretap traffic, or actively interfere with the delivery of packets to prevent paths from being formed. If he targets availability and controls a large number of hosts (e.g., botnet), he could also perform DoS attacks on end hosts or selectively congest network links. As SCION is also used as a secure interdomain routing protocol it must display Byzantine robustness.

5.2. Attack Taxonomy

To aid in identifying and defending attacks in SCION, we introduce an attack taxonomy. To be successful the taxonomy should be comprehensible, exhaustive and repeatable. Our taxonomy builds on previous work [83] that is well accepted, but we extend it for the purpose of SCION. We use three major classifiers as a foundation: target of the attack, the attack vector used to execute the attack, and the impact the attack had on the architecture, which is illustrated in Figure 5.1. The taxonomy is organized as a tree structure to classify attacks in a mutually exclusive way by using multiple associations and allow a classification of blended attacks, where an adversary uses a series of attack vectors to make him succeed.

There exists also minor classifiers such as *visibility* of an attack, the *location* the attack takes place (core versus non-core), the *position* of an attacker (on-path, off-path; within ISD, outside of ISD), or the *purpose* behind the attack (malicious, benign), but they are omitted for the sake of simplicity. Successfully using an attack taxonomy contributes to the root cause of the attack by structuring essential information and helps to speed up recovery. Furthermore, a taxonomy that is constructed using simple terms can be used to point out an attack to a person who is not an expert in the area (e.g., management of a company).

Target. The first classifier specifies the element of the SCION architecture that has been targeted by the adversary. Internally, it is split up into five sections: end hosts, routers, SCION infrastructure servers (servers, router, end hosts), network plane (control plane, data plane, network links), and trust infrastructure (public key infrastructure, entity validation infrastructure). Targeting one or a combination of these might lead to a successful attack. The provided list of targets is specifically suited for SCION.

Attack vector. When an attack takes place, there is a possibility that it uses multiple vectors as a path to a full-blown attack. An attack vector is defined as a path the adversary utilizes to execute the attack. Considering that most attacks are not isolated events, a combination of attack vectors may be used to depict the complete path of an attack. The attack vectors range from common exploitation of vulnerabilities on the end host to social engineering. As SCION represents an alternative redesign, we also include design flaws (e.g., in the routing protocol) as a possible attack vector. Misconfiguration, when used to gain access to a system, can cause a variety of attacks. If the integrity of an element is affected, often integrity verification at an endpoint has been insufficient (e.g. an attacker actively forges a valid MAC for a foreign OF before propagating the packet).

Impact. An attack on a system has potential to impact *confidentiality* of sensitive information, the *integrity* of bypassing packets, or the *availability* of the system in various ways. The three main categories can be further partitioned using sub-categories. In our illustration we show a non-exclusive list of examples, which can be extended as needed. For example, disclosure, as a sub-classifier of confidentiality, typically provides the attacker a view of information he would normally not have access to. Discovery of information not previously known, for example, can be obtained using a scanning tool to probe for information, which then is used to launch an attack on a particular target.

Examples. We will use our attack taxonomy with attacks on SCION (see Table 5.1). This will depict how the taxonomy successfully classifies malicious events. We use examples from different sections of the SCION architecture to show its versatility and completeness. For completeness, we also added attacks that are not focused in this thesis.

Attack description	Target	Attack vector	Impact
DoS attack on beacon server	beacon server	denial of service	availability
Man-in-the-middle attack	data plane	insufficient verification of communication partner	confidentiality
Vulnerability exploitation at end host	end host	buffer overflow	confidentiality
Benign beacon announcement	control plane	misconfiguration	integrity
Physically cutting a network cable	data plane	physical attack	availability
Persuade key-holder to leak a private key	data plane	social engineering	confidentiality

Table 5.1.: Example use of SCION attack taxonomy.

5.3. Path Hijacking

An off-path malicious AS may attempt to attract traffic for inspection, as done during BGP hijacking attacks [17]. In SCION, these types of attacks require an adversary to manipulate the path selection process to become on-path. For example, he convinces a sender that a path through the adversary-controlled AS has higher fidelity than those traversing other ASes. However, existing paths cannot be manipulated, as their metadata is cryptographically protected and verified for correctness at each hop. After becoming on-path, the attacker might follow up with black- or grey-holing to affect the availability of a remote AS.

5.3.1. Beacon Theft

Beacons that are transmitted downstream may be recorded by on-path adversaries (e.g., eavesdrop on a link between ASes). By re-injecting that beacon onto another link, the adversary can extend paths as long as the beacon is correctly forwarded. This attack is possible because SCION does not include the next-hop information in the path construction beacon, allowing any next hop to extend the path arbitrarily using the previous beacon.

For example, an AS wants to share its peering link only with one of its downstream neighbors and therefore decides to selectively announce the peering link using beacons received from upstream. As shown in Figure 5.2, the monitoring adversary misuses this beacon to get access to the peering link by pre-pending it to his own path. The presence of such a link can be discovered by querying a path server.

This attack can be mitigated by including the next hop information in the PCB before disseminating it further downstream. Therefore, the beacon can only be used by the intended AS and beacon theft is

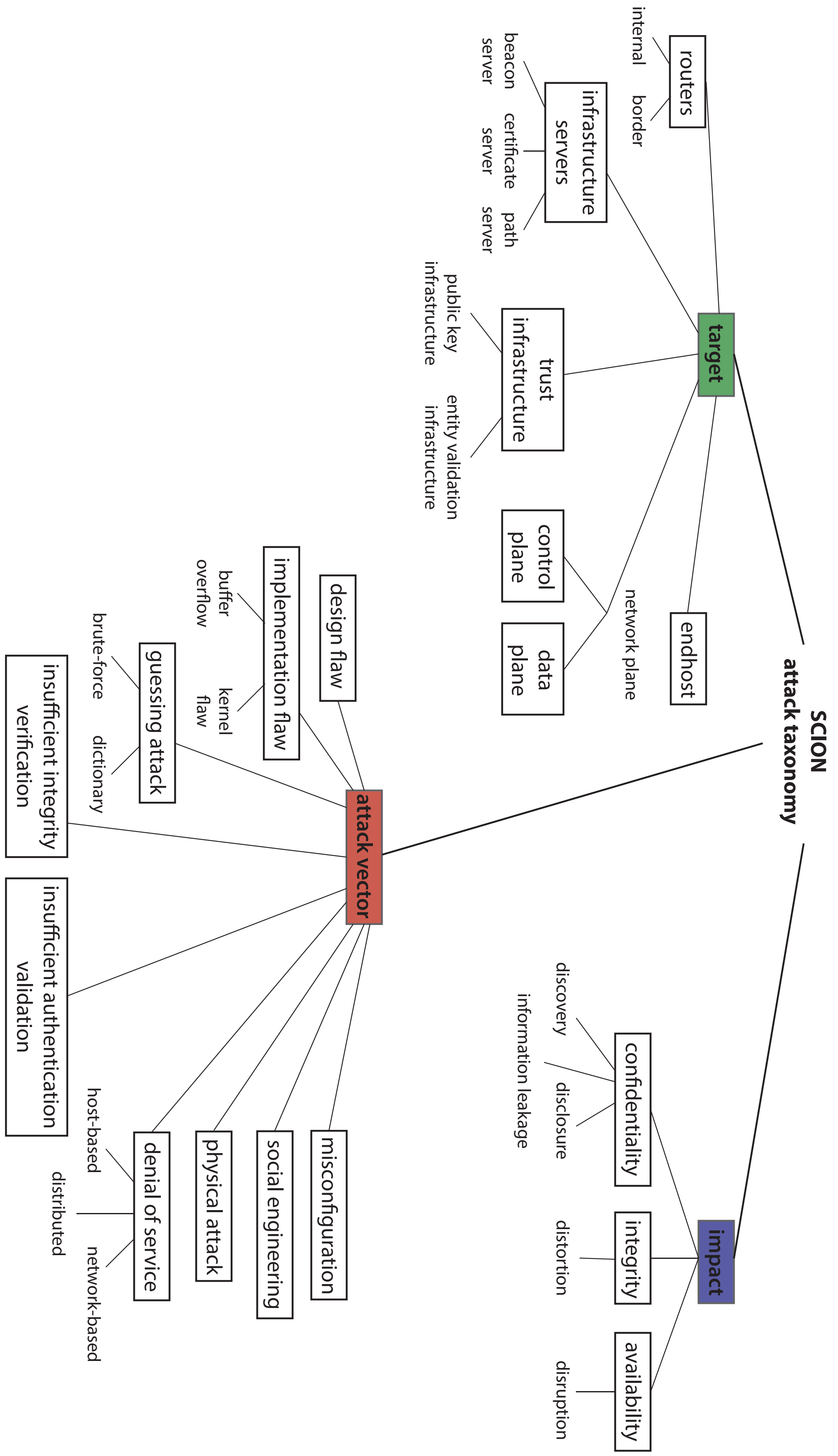


Figure 5.1.: SCION attack taxonomy

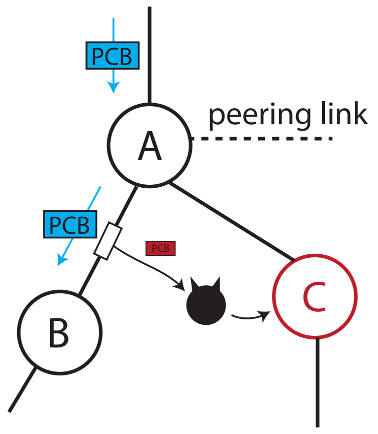


Figure 5.2.: Beacon theft. AS A wants to selectively share access to the peering link with B, but not AS C. Eavesdropping adversary reads beacon intended for AS B and re-injects it at his own AS C to get access to the peering link.

impossible. In the current version of SCION, this problem has already been addressed using our proposed solution.

According to our taxonomy, this attack would be classified as: 1) target: control plane, 2) attack vector: design flaw, 3) impact: integrity.

5.3.2. Interposition Attack

A malicious AS who can observe links between a customer and provider AS can also inject its own opaque fields into the PCB toward downstream ASes. When used in conjunction with blocking downstream PCBs from the benign AS, the malicious AS can position itself as an upstream path to the core, as illustrated in Figure 5.3.

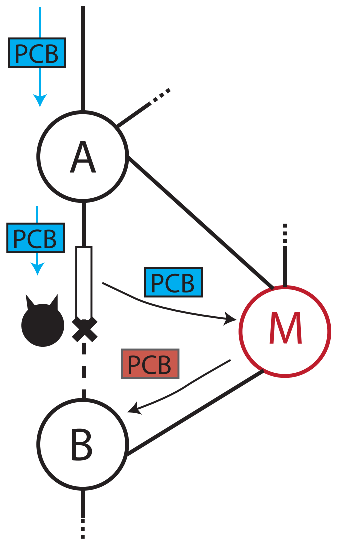


Figure 5.3.: Interposition attack. A malicious AS who can observe a link between benign ASes can block downstream beacons and insert its own to position itself upstream of the victim.

The attack is detectable by downstream ASes who perform verification of inbound PCBs; the adversary's PCBs are not signed with the expected key. Additionally, it is unclear why an adversary would attempt to redirect all traffic through it since this is a visible and thus detectable attack. Cautious on-path adversaries are more likely to eavesdrop passively on the link. By introducing a next-hop information for beacons as described in the previous section, this attack can also be mitigated.

The SCION attack taxonomy classifies this attack as: 1) target: control plane, 2) attack vector: design flaw, 3) impact: integrity.

Finding 5

Adding a specific destination to a beacon before sending it to the next hop solves the problem of beacon theft and interposition attacks, as the beacon can only be used by the intended host.

5.4. Path Manipulation

Path manipulation attacks try to alter the path chosen by a sender. The goal of the adversary is to redirect traffic at will (possibly without the sender noticing). SCION paths are integrity protected, but can be manipulated, as shown in the following sections.

5.4.1. Fake AS/ISD Creation

Each AS appends information to PCBs as they are disseminated downstream. Malicious ASes in SCION can spoof entities by crafting new PCBs and OFs and signing them using a valid certificate. These entities can represent anything from a fake AS to an entire ISD. If traffic is sent upstream with the spoofed entity as a source entry, it will appear to originate downstream from beyond the malicious AS. This allows the adversary to plausibly deny the misbehaviour and makes detection of this attack difficult.

However, the attack above is difficult to execute, because spoofing a new AS requires a registration of that AS with the Core. ISD Cores should enforce a minimum level of entity verification. Similarly, if someone wants to introduce a new ISD (possibly spoofed), all cross-verifying ISDs need to verify its legitimacy. This verification process needs to be specified in detail as it probably needs an out-of-band process.

According to our taxonomy, this attack would be classified as: 1) target: control plane, 2) attack vector: insufficient authentication validation (during registration process), 3) impact: integrity.

5.4.2. Source Address Spoofing

A recent study has shown that more than 40% of all global ASes allow some level of IP address spoofing [14]. These ASes don't perform appropriate levels of ingress filtering, which means that they forward traffic that claims to have originated in another network or AS. Source address spoofing is often used in conjunction with DoS attacks (e.g., reflection attacks as described in Section 2.3.1).

SCION addresses can be replaced by an adversary as they are not integrity protected. However, as SCION does not route based on source or destination address, the effect of address spoofing is limited. If an adversary replaces the address in the header of a SCION packet, he would just influence the destination of the packet within the target AS. This can be used to mask the origin of a packet within an AS. For an end domain to spoof traffic through an upstream AS, they would need to have a valid OF for traversing the upstream AS. The only two ways of obtaining a valid OF when the victim AS is not upstream is to craft it, and guess the MAC secret key. Additionally, an AS could ingress filter packet addresses which origin outside from the current AS. Another solution could include a logical binding of the source address, so that the destination AS can verify the origin of the packet.

The SCION attack taxonomy classifies this attack as: 1) target: control plane, 2) attack vector: insufficient integrity verification (of source address), 3) impact: integrity.

Finding 6

The almost complete absence of source address spoofing makes the SCION architecture less vulnerable to attacks based on IP spoofing.

5.4.3. Modification of Path Metadata

Some fields in the header and in the path segment metadata (as described in Section 3.3) are not authenticated or integrity protected and thus vulnerable to unauthorized modifications. Specifically, the SCION common header (SCH) contains the total packet length, source and destination types, the header length, pointer to the current Info Field (IF), to the current Opaque Field (OF), and to the next header (either SCION extension or header of a layer-4 protocol). The pointers are adjusted as the packet traverses the network (e.g., whenever a new path segment has started). If an unauthorized entity changes these pointers, path verification might fail. An adversary can also extend the packet with arbitrary content, set the packet length accordingly, and adjust the pointers to a location of his choice. This leads to a *path extension attack*, where an adversary adds arbitrary OFs of his choice, modifies the pointer, and sends the packet further downstream. Similarly, modification of the destination type will make the border router in the victim's AS incapable to deliver the packet to the correct end host.

The metadata for each path segment is stored in an Info Field (IF). The three least significant bits represent flags to express the direction of verification and whether the path contains a short-cut or peering link. Subsequently a timestamp, set by the initiator of the PCB, follows. This value cannot be modified as it is included in the calculation of the MAC. Otherwise, the timestamp could be adjusted and make a path appear invalid by backdating the timestamp, or make it valid in the future by setting in to a later date. The last byte indicates the number of OFs the given path segment contains. To add or remove OFs to/from a path segment, this value must be adjusted.

Modification of path metadata can be mitigated using integrity protection. In the future, SCION will offer a end-to-end security extension which protects these fields using asymmetric cryptography so that the entire SCH will be integrity protected.

Using the SCION taxonomy, this attack would be classified as: 1) target: control plane, 2) attack vector: insufficient integrity validation, 3) impact: integrity.

5.4.4. Opaque Field Manipulation

Modifying opaque fields is designed to be difficult without knowledge of the MAC secret key. If the key is not known, the attacker must attempt to brute-force the key to generate a valid MAC for the modified path. This can be done offline, but as SCION uses 128-bit keys, a brute-force attacks is computationally infeasible for our adversary model. Furthermore, intermediate keys for the signing process are likely to

be short-lived (they will be rolled after a specified amount of time).
The adversary might also try to brute-force a MAC. With 3 bytes (i.e., 24 bits), the length of these MACs is relatively short. Given an OF with arbitrary values, the attacker would need try $\approx$ 17 million different plaintext/MAC combinations (given plaintext, different MAC) to successfully cheat the integrity verification at the next hop. Note that brute-forcing the MAC can only be done online. These attempts would be highly visible to a monitoring entity. Monitoring these failures without keeping state would be research question on its own and is out of scope for this thesis. Generally speaking, the attacker need to generate 2^l packets in the worst case, where l is the length of the MAC. With sufficient length the visibility and feasibility of these attacks increases respectively decreases. A more detailed analysis of the MAC length can be found in Section 4.3.2.

The attack taxonomy classifies OF manipulation as: 1) target: control plane, 2) attack vector: design flaw, 3) impact: integrity.

5.4.5. Path Truncation

Each a AS extends a SCION path with its own information and signs it as it forwards beacons. The signature also takes the information of the previous OF into account. Therefore, it is possible to shorten a path by removing OFs from the either the beginning or the end of the path (see Figure 5.4. As long as the previous OF is included (if it exists), verification in the current AS will succeed. Path metadata includes a hop count, but this metadata is not cryptographically protected. This allows traffic destined for an AS to be stopped on the path upstream of the AS without evidence that misbehavior is taking place. However, an on-path adversary as described can always drop bypassing traffic and thus blackhole ASes on the path downstream.

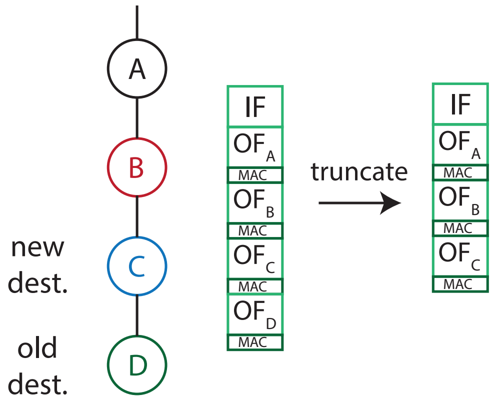


Figure 5.4.: Path truncation attack. A malicious AS B strips off an OF at the end of a path.

Blackhole attacks by an on-path adversary are hard to defend against. To give evidence that a path shortening attack has taken place, SCION could make use of a different structure for OFs at the head and the tail of a path. They could include a cryptographically protected flag, which marks the end of a path. Apart from that, the OFs at the head and tail of a path have no requirement for an egress interface (just ingress). Therefore the structure of these OFs can be different compared to the others.

According to our taxonomy, this attack would be classified as: 1) target: control plane, 2) attack vector: design flaw, 3) impact: integrity.

Finding 7

OFs at the head and the tail of the path must have a different structure, so that traversal OFs cannot be misused as endpoint OFs.

5.4.6. Path Splicing

Path splicing attacks describe attacks where an adversary takes valid path segments of different paths and splices them together to get a new valid path. We call these attacks "Franken"-path attacks.

Assume there exist two valid paths p1 and p2, each containing an IF and a sequence of OFs (see Figure 5.5). All OFs are chained by appending the previous OF to the current OF and then calculating the MAC. The integrity of the previous OF is not checked, but cannot be checked since the MAC verification key is not available to entities other than the AS that generated the OF. AS C is acting maliciously (active and on-path) and replaces OF_A with OF_B (and the source AS if necessary) for an incoming message and forwards it to AS D. From D's point-of-view, the MAC validation is successful. D can't determine that the path has been modified by C and thinks that the traffic is coming from B. On the reversed path, C reverts its path changes and forwards the packet to A. MAC verification is still successful, because after reversing the path (indicated by the 'up' flag) the integrity is verified with the 'next' OF instead of the 'previous' one.

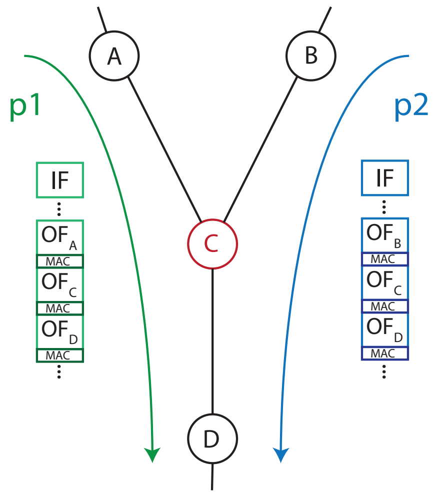


Figure 5.5.: Path splicing attack. A malicious AS C combines multiple paths to trick an entity downstream relative to him.

Weak path integrity. Weak path integrity property describes the chaining of OFs by taking the previous OF into account when calculating the MAC. This property is in general sufficient as it guards against errors and misconfigurations, as well as simple path alterations. Without chaining, any end host could do a very simple "Franken"-path attack by combining OFs that traverse ASes. The chaining prevents such attacks by end hosts, but does not prevent active path alterations by malicious routers on the path. This property is still considered to be strong, since an off-path entity cannot attract/influence the flow.

To defend against path modification SCION needs to make sure that the traversed path matches the path described in the packet header. This could be achieved using signatures. The origin AS would sign a statement that includes the set of hops traversed. The destination AS can then verify that the path traversed through those hops upon receiving a packet using the signed statement. However, asymmetric cryptography needs a lot processing power, which will decrease the throughput of the SCION architecture.

The SCION attack taxonomy classifies this attack as: 1) target: control plane, 2) attack vector: design flaw, 3) impact: integrity.

Finding 8

SCION only offers weak path integrity to defend against malicious entities. This property does not defend against on-path attackers, but is sufficient against simple splicing attacks, as well as off-path adversaries. In the future, SCION can provide strong path integrity using a path validation extension.

5.5. Path Preference Attacks

Path preference attack aim to influence the path selection process of senders. The goal of such an attack is to make paths controlled by the attacker more attractive than other available paths. A simple example is a low or even negative price in a pricing system. The price of a path is typically derived from an edge weight and an “attractiveness” value. If such a system is deployed in SCION, path diversity attacks must be taken into account, especially for powerful adversaries.

5.5.1. Fake Link Flooding

In a fake link flooding attack, the attacker receives PCBs from its provider (or peer) and announce this path downstream. If this adversary-announced path complies with the policy of downstream ASes, it may be added as one of the k paths available to the end host. At this point, the end host may select it for future communication. The attacker controls the full domain and can also advertise fake upstream links downstream. The advertised paths may selectively crafted with good properties such as high bandwidth, long lifetimes, cheap price, or any other path fidelity parameter. This increases the chance of selection by senders. A concrete example is shown in Figure 5.6, where an attacker floods $m \times n$ paths which become available to the sender.

However, this attack is only successful if the attacker is located within the same ISD and upstream relative to the victim AS. It is not possible to attract traffic away from the core as traffic travels upstream towards the core. Furthermore, the attack may be discovered downstream (e.g., by seeing large numbers of paths become available) but also during path registrations. After detection, paths traversing the adversary AS can be identified and avoided by regular ASes.

According to our taxonomy, this attack would be classified as: 1) target: control plane, 2) attack vector: insufficient authentication validation (during registration), 3) impact: integrity.

5.6. Information Leakage

Information leakage happens whenever a system reveals some information to unauthorized parties. While often being neglected, information leakage can subtly or completely destroy the security of an otherwise

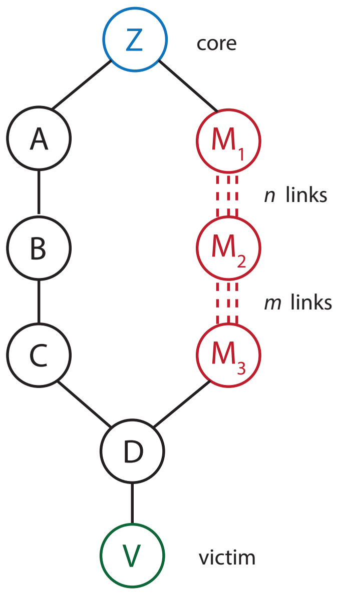


Figure 5.6.: Fake link flooding attack. Malicious ASes M_1 - M_3 advertise $m \times n \gg 1$ to increase chance of selection by victim V.

secure system (e.g., key leakage). The SCION infrastructure can unintentionally leak information about its configuration, internal workings, or violate privacy through a variety of problems. Furthermore, it can also leak internal state (e.g., utilization of infrastructure element) via how long they take to process certain operations. Often, this information can be leveraged to launch powerful attacks.

5.6.1. Leaking Internal Topology

As a downside of transparency, ASes respectively ISDs leak information about their internal topology. An on-path adversary obtains this information by header inspection of bypassing traffic. Even if OFs are only meaningful to the AS itself, other information such as the hop count or the beacon creation time (stored in IF field) is useful to the adversary to recreate the internal topology of an ISD. Also PCBs contain topology information which is useful for malicious activity (e.g., path flooding attacks). The adversary can also query the infrastructure services within an ISD to obtain information.

Completely mitigating information leakage is technically challenging, as also side-channels such as timing information exist. An improvement to mitigate leakage of internal topology would be path encryption using a per-path shared key or asymmetric cryptography. This way only entities knowing the key are able to obtain this information. The infrastructure services could implement authorization to answer queries only to authorized entities. To mitigate information leakage of source and destination address, address translation could be used.

The attack taxonomy classifies leaking internal topology as: 1) target: control plane, 2) attack vector: design flaw, 3) impact: confidentiality.

5.6.2. Path Enumeration

We assume that a malicious AS wants to enumerate possible paths to discover hidden or previously unknown paths. There exists no simple scanning possibility in SCION, as packets with an invalid OF will get dropped. The only way to enumerate paths is to craft a valid packet including valid OFs for each hop. Assuming that there exists only a small number of interface at the next hop, the adversary can reduce the number of different possibilities by some degree. Specifically, there exists 2^{64} different OFs. Using the previous assumption, the adversary can reduce the space to $\approx 2^{47}$ (7 bits from each interface field, plus the 3 reserved bits). This is still a huge address space which makes it infeasible for guessing attacks. Following, in SCION path enumeration is difficult to achieve.

According to our taxonomy, this attack would be classified as: 1) target: control plane, 2) attack vector: guessing attack, 3) impact: confidentiality.

5.6.3. Leaking Non-registered Paths

Non-registered paths in SCION are defined as paths that are used in an out-of-band (OOB) manner. They are not registered at path server, but rather distributed to only authorized senders.
Due to the transparency in SCION, once one of these paths is used, an on-path adversary can inspect the down-path segment. Using the information gained, he can misuse it to send traffic to the destination without being intended to use this particular path. Therefore, the “hidden” path gets revealed.

In this situation, on-path adversaries, including compromised end hosts, are difficult to defend against. As a receiver, one could inspect the full path and observe which AS has been the sender. By keeping track of all parties given access to the hidden path, the receiver can determine if an unauthorized party has used it. However, this requires additional effort on the receiver side.

Another solution to selectively allow access to specific paths is to form a separate sub-ISD with all other parties that are allowed to send traffic. The separate sub-ISD propagates its own paths, which can’t be used outside of that ISD. As an AS can be part of multiple ISD (overlap), In this case, isolation solves the problem of misusing a OOB-path by an external party.

A more heavyweight approach would be path encryption, where the path indeed remains “hidden”. However, this would require a shared key amongst every AS on the path.

The SCION attack taxonomy classifies this attack as: 1) target: control plane, 2) attack vector: insufficient authentication validation, 3) impact: confidentiality.

Finding 9

Transparency as proposed in SCION is a delicate topic relating to information leakage. Isolation using a separate sub-ISD solves path misuse by an unauthorized party.

5.7. Attacks on the Data Plane

In this section we discuss attacks on the data plane of SCION. Attacks like this typically target the integrity and confidentiality of a packet payload. If a malicious AS exists on the path between source and destination, it is able to inspect the traffic between the communicating endpoints. On-path adversarial traffic inspection is inevitable in any network infrastructure, including today’s Internet. Therefore, endpoints need to use transport- or application-layer encryption to hide sensitive data. Otherwise, a cautious on-path adversary can easily eavesdrop passively on the link. An active adversary can take a step further and alter data of bypassing packets, or just drop them (blackhole).

5.7.1. Surveillance

On-path traffic inspection is often referred to as surveillance. The scope of such an attack can be scaled to target a large number of victims. A powerful attacker can either passively monitor traffic at multiple observation points or actively try to redirect traffic for inspection.

In SCION, ISDs prevent entities in remote domains from manipulating other's control plane, precluding adversaries from tricking other networks into sending traffic through them. Inside ISDs, the core may appear to be a good vantage point for doing surveillance. Oppressive states may collude with or compel their local ASes to perform surveillance. However, peering links offer an alternative path out of the ISD without traversing the core. Due to source selected paths and path transparency, senders can select paths through ISPs they trust, optionally taking peering links to leave the ISD.

According to our taxonomy, this attack would be classified as: 1) target: data plane, 2) attack vector: insufficient authentication validation, 3) impact: confidentiality.

5.8. Availability

In this section, we discuss the effect of attacks and misuse targeting availability of the SCION control plane using standard attacks also known in the current Internet, as well as SCION-specific attacks.

5.8.1. Denial of Service

The current Internet lacks a mechanism to defend against Denial of Service (DoS) attacks; some properties even facilitate the execution of such attacks (see Section 2.3.1). Most defense services are based on content distribution and cloud-based mitigation. DoS attacks as such can target end hosts, essential infrastructure services, specific links or even entire domains.

SCION's core defense against DoS is to enable differentiated inter-domain traffic resource allocation to individual flows (using SIBRA) and high availability through multi-path communication. Other features such as non-registered paths also make an effective DoS attack harder. Victim ASes have control over inbound paths through which they want to be reached. This enables them to end an attack along with inbound connectivity by removing a specific path. The attacker can continue to send traffic until the path expires. The attack time can therefore be controlled by the path lifetime.

DoS on end hosts. SCION does not provide a specific defense for DoS attacks on end hosts, because as an inter-domain routing architecture, it has no control over end hosts. If a path to the destination exists and is publicly available, a remote host will be able to send traffic to it, enabling host-specific attacks such as server flooding or exploitation of a vulnerability.
Attacks on the end host increasingly target the application layer [65], and thus appropriate measures must be ensured. SCION's path transparency property helps identifying the sources of traffic if they are located within an ISD. This enables it to filter or prioritize traffic (e.g., inside-ISD traffic over outside-ISD traffic) based on its path. If only specific paths are affected by the attack, withdrawing them and using other paths might help to partially evade the attack.

DoS on essential infrastructure services. SCION depends on its infrastructure services for regular network operations. If any of the services is not available, path creation, dissemination and authentication cannot be achieved.

The simplest way to provide high availability to these services is replication. For each service, a pool of servers is available which share state amongst them. As additional capacity is needed, new servers can be added to the pool without disruption. Once a service pool is operational, a master server is elected. All other servers automatically replicate the state of the master server. If the master becomes inactive (for any reason) and does not respond within a specific amount of time, a new master will be elected.
For lookup services (query a server), load balancing helps to distribute the load among all available servers. For an update operation (as done by a beacon server) only the master needs to be available and eventually propagate the current state.

In addition to replication, ASes can rate limit inbound requests coming from outside the AS. Rate limitation could be based on the internal server capacity or the available bandwidth.

DoS on links. DoS can not only target end hosts but also specific links. Such attacks as described in Coremelt and Crossfire generate seemingly regular traffic that traverses specific inter-AS links. These links are often high-capacity, and over-provisioning them can impact the Internet traffic.

In SCION attackers can target specific links more effectively than in today's Internet, because source selected paths give the sender more control over the end-to-end path. However, link bottlenecks are also present in today's architecture and solved using re-routing to a backup path if a link fails due to overload.

DoS attacks are classified using the SCION attack taxonomy as follows: 1) target: all target, 2) attack vector: denial of service, 3) impact: availability

5.8.2. Link Flapping

A malicious AS could attempt to overwhelm path servers by repeatedly enabling an immediately disabling interfaces. Events like these are called link flapping [66]. When a link changes to a disabled state, all paths using those links must be revoked from path servers (see Section 3.2). As soon as path revocation is ongoing, the interface may come back online allowing new paths to be created, which in-turn get registered at a path server. Events described above don't only apply to malicious ASes, but also occur in software- or hardware-defects which cause link flapping (e.g., bad cable, duplex mismatch, or bad interface converter).

The solution for the attacks above is for end domains to avoid unstable links. For example, an interface could be marked as erroneous if it flaps more than 5 times within 10 seconds, and avoided in subsequent communication. To help prevent overloading of path servers, briefly holding off on path registrations might help (but can also interfere with availability). The hold off could be synchronized with the beacon dissemination process. For example, if the PCB starts to get distributed, new path registrations are not possible, because it first needs to be propagated through the network. After the hold off phase, all ASes can register their newly discovered paths at the path server. This way, the path server is less likely to be overloaded as he can process queuing registration request during the hold off phase.

According to our taxonomy, link flapping would be classified as: 1) target: control plane, 2) attack vector: design flaw, 3) impact: availability.

5.8.3. Packet Replication

In packet replication attacks the adversary resends packets previously received from other nodes or considered "stale" (packets that are delayed past a certain time). With large amount of packets replayed, both the bandwidth of the network and the computing power of nodes are consumed in vain. This leads to the early termination of network operations and can be used to perform a DoS attack on victims.

This is *not* a SCION-specific attack as it also occurs in today's Internet. Typically the attacker belongs to the same network as the victims and attacks the resources inside the network by broadcasting them. In SCION, an attacker must be on-path to successfully spam a victim with packet replication. The path transparency helps identifying the malicious entity.

Regular actions to prevent replay attack such as sequence numbers, expiration times and storing hashes to filter with bloom filters seem to insufficient for the architecture, because they cause overhead or state at the end host.

The SCION attack taxonomy classifies this attack as: 1) target: data plane, 2) attack vector: design flaw, 3) impact: availability.

5.8.4. Forwarding Loops

Forwarding loops occur if packets traverse the same subset of ASes on the up-path as well as on the down-path (see Figure 5.7). Each AS is in this subset is traversed twice. Instead the optimal path would not include the loop part but instead use a short-cut. The impact of this attack might be limited considering a single attacker, but if a collaboration of malicious entities exploit this possibility, availability of the intermediate ASes (e.g., on network link between ASes) can be affected. The amplification factor of this attack is 2, which makes it comparatively harmless.

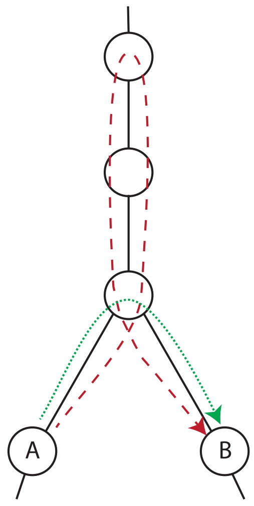


Figure 5.7.: Forwarding loop occurs if a packets is sent along the red path to get from source A to destination B. Using a short cut as represented by the green path avoids the loop.

Apart from being an attack, forwarding loops are a design flaw and should be avoided in a clean-slate design. However, due to source-selected path principle of SCION the end-to-end path cannot be influenced by intermediate nodes. Regular operating nodes can easily avoid routing loops by using a short-cut. If this flaw gets misused or an intermediate ASes suspects to be part of a route loop, it could inspect the up- and down-path segment and check them for similarities.

According to our taxonomy, this attack would be classified as: 1) target: data plane, 2) attack vector: design flaw, 3) impact: availability.

5.8.5. Resource Exhaustion

To achieve scalability SCION avoids keeping router state wherever possible. This should prevent state-exhaustion attacks and state inconsistency. However, resource exhaustion at any point in the infrastructure can not be excluded. Hardware such as memory and computational power are limited and will be exhausted at some point, if the queuing rate of jobs is higher than the processing rate.

SCION extensions that use asymmetric cryptographic operations during session setup are susceptible to resource exhaustion if too many sessions are initiated through a single node in a short period of time. An adversary controlling multiple endpoints may attempt to simultaneously initiate SCION sessions through a particular AS and effectively exhaust memory and computation resources on the victim. Asymmetric cryptographic operations are much slower than symmetric ones. Simultaneous initiation of SCION extension sessions through a particular AS thus might eventually lead to resource exhaustion, which prevents new sessions from being created.

Asymmetric cryptography not only has a high overhead, but also requires the public keys of the AS or end host to be available. SCION uses the Dynamically Recreatable Key (DRKey) [55] protocol that enables routers to derive symmetric cryptographic keys on-the-fly from a single local secret. The protocol runs before the session starts. For each session created along a path, a session key is derived from an initial asymmetric key pair. Each intermediate router then encrypts its derived key using the source's public key. This keys are used to set up shared keys between the source, destination and the intermediate routers to enable subsequent path validation even after communication has begun. An adversary that tries to establish new sessions in high frequency might eventually reach resource exhaustion at any point on the path. Each node has to process the pre-session overhead as well as store the session key for each other node on the path.

Resource exhaustion is classified as: 1) target: router, 2) attack vector: implementation flaw, 3) impact: availability.

5.9. Misconfiguration

Misconfiguration that affects security can happen at any level of an application stack. SCION offers various tools to disclose misconfiguration making them easily detectable. In the following we discuss possible misconfiguration and their influence on the SCION architecture, in particular routing decisions.

5.9.1. Benign Packet Forwarding

A misconfigured AS can accidentally forward packets across the wrong interface. The interface numbers are stored for each packet in the per-AS OF, but internally not statically defined. In case an erroneous assignment between logical interface numbers and physical links happen, a misconfiguration as above can happen. Upon a packet arriving at the next AS, the next hop will try to verify its own OF and fail. At this point, it is likely that the packet will get dropped, which will eventually lead to the path being marked as unstable. This type of misconfiguration leads to a blackhole, but will get resolved by the architecture, as unstable paths are avoided by a path-selecting source.

Benign packet forwarding is classified as: 1) target: control plane, 2) attack vector: misconfiguration, 3) impact: availability.

5.9.2. Benign Beacon Announcement

Through misconfiguration, ASes can accidentally announce beacons they did not intend to announce. These announcements may result in paths temporarily being created. For a third party (possibly an adversary), this misconfiguration is hard to distinguish from a regular attack. The OFs that were propagated did not leak any internal topology information about the misconfigured AS, as they are meaningful only to the AS that created that OF. The erroneous announcement can be detected by inspecting the traffic traversing the AS at a later time. Misconfiguration in the beaconing process does not impact routing decisions in a remote ISD.

The SCION taxonomy classifies benign beacon announcement as: 1) target: control plane, 2) attack vector: misconfiguration, 3) impact: confidentiality

5.10. Summary

In this chapter, we performed an in-depth analysis of various aspects of SCION's security. We analyse security defenses against existing network attacks, examine attack vectors on the new architecture, and describe attacks found. The attacks range from attacks on the forwarding process of SCION up to attacks on availability of the infrastructure. For each new attack, we discuss its impact and a possible solution to prevent them (if they don't have minimal impact). Our fixes will be integrated where possible in future versions of SCION.

The proposed taxonomy offers a way to classify attacks in SCION, but also may be of interest for other FIAs. Even a series or combination of attack vectors to a powerful attack can be illustrated. We summarize the classification of attacks found in Table 5.2.

	Target	**Attack vector**	**Impact**
Path Hijacking			
Beacon theft	control plane	design flaw	integrity (of path)
Interposition attack	control plane	design flaw	integrity (of path)
Path manipulation			
Fake AS/ISD creation	control plane	insufficient authentication validation	integrity
Source address spoofing	control plane	insufficient integrity verification	integrity
Modification of path metadata	control plane	insufficient integrity verification	integrity
Opaque field manipulation	control plane	design flaw	integrity
Path truncation	control plane	design flaw	integrity
Path splicing	control plane	design flaw	integrity
Path preference attacks			
Fake link flooding	control plane	insufficient authentication validation	integrity
Information leakage			
Leaking internal topology	control plane	design flaw	confidentiality
Path enumeration	control plane	guessing attack	confidentiality
Leaking non-registered paths	control plane	insufficient authentication validation	confidentiality
Attacks on the data plane			
Surveillance	data plane	insufficient authentication validation	confidentiality
Availability			
Denial of service	all targets	denial of service	availability
Link flapping	control plane	physical attack	availability
Packet replication	data plane	design flaw	availability
Forwarding loops	data plane	design flaw	availability
Resource exhaustion	router	implementation flaw	availability
Misconfiguration			
benign packet forwarding	control plane	misconfiguration	availability
benign beacon announcement	control plane	misconfiguration	confidentiality

Table 5.2.: Classification of presented attacks using SCION attack taxonomy.

6 Attacks on Trust Infrastructure

Remark: *Parts of this chapter will be published in the paper "Internet Kill Switches Demystified" and contain contributions from Dr. David Barrera and Daniele Asoni.*

SCION makes heavy use of cryptographic keys to enable verification and trust infrastructure with multiple roots. In this chapter, we analyze the impact of trust infrastructure-related security issues. The scope of this chapter aims on powerful adversaries. First, we focus on currently partly deployed and emerging trust infrastructures DNSSEC and BGPsec, and analyse whether the same attacks also exist for SCION. Further, we describe the necessary steps to recover from the described attacks and present ideas on how to quantify them.

6.1. Threat model

The scope of this chapter is on powerful adversaries (most likely governments conducting cyber-warfare) aiming to disable, either partially or fully, another party's access to the Internet. This can be done by injecting forged routing information into a BGP-enabled network, injecting forged DNS records or responses, or executing denial of service attacks on critical elements of the infrastructure (e.g., path servers in SCION). As recently reported [38], adversaries may coerce organizations (e.g., hardware or software vendors, network operators) to implant back doors, or independently collect data through mass surveillance techniques [29]. Our main focus lies on scenarios where adversaries use these methods to steal private signing keys, but also attacks where keys remain secret are investigated.

In section 6.2 and 6.3, we assume pervasive and sole deployment of DNSSEC and BGPsec. We do not consider dual DNS/DNSSEC and BGP/BGPsec environments where the victim could switch off the secure variant and fall back to the insecure protocol. Indeed, an adversary powerful enough to steal cryptographic keys could also execute prefix hijacking attacks or DNS poisoning attacks to implement a kill switch, and thus effectively execute a kill switch in BGP or DNS.

As of writing DNSSEC is deployed on $\sim$87.52% of all top-level domains (TLD) [48], and $\sim$12.63% of all DNS queries are validated [5]. For RPKI only $\sim$6.32% of advertised IPv4 prefixes have corresponding ROAs (both valid and invalid) [73].

In section 6.4, we assume advanced deployment of the SCION architecture. Instead of a strict hierarchical organization as in DNSSEC or BGPsec, the Internet is structured into multiple ISDs (possibly containing ISDs themselves). These ISDs can represent groups of various scales, such as companies, conglomerates or countries.

6.2. Triggering DNSSEC Kill Switches

This section discusses the necessary steps to adversarially trigger a kill switch on networks that have deployed DNSSEC. We describe operational considerations necessary for the attack. We categorize the kill switches in DNSSEC into two classes; those where a private signing key is compromised, and those where keys remain secret.

6.2.1. Kill Switches with Key Compromise

Key Compromise Vectors. There are several attack vectors for compromising a DNSSEC signing key. Attackers might exploit a software vulnerability (e.g., as in Heartbleed), assume control on network operator's computers or routers (e.g., using malware), or use social engineering as an attack vector. Employees motivated by personal profit may go rogue, or be threatened/extorted to leak keys. Governments may (possibly legally) compel network operators to either leak their private keys or behave according to their intentions. Depending on the country, compromised keys can enable control of a top level domain (e.g., *.ca* in Canada) or even control of the root zone (i.e., the United States).

Many registrars support hosted DNSSEC for registered domains. These services handle all further signing of the zones as well as key roll-over, but create another attack vector as they might allow control over all domains hosted there.

Under some circumstances, private keys corresponding to DNSSEC public keys can be brute-forced. The DNSSEC Operational Practices [56] recommends key sizes of 1024 bits and larger, but a recent study found that 512-bit keys are surprisingly persistent [93], which can be factored in a few hours [41].

Response Forgery. In order to refresh DNS entries with low time-to-live (TTL) and to support dynamic DNS changes, the ZSK is usually kept online. If an adversary obtains the ZSK of a particular zone, he would be able to modify the Delegation Signer (DS) entries linking to the child zone and validly sign them. DS entries can be crafted such that all clients trying to verify the chain of trust will fail, believing that the KSK of the child zone has been changed. As a consequence, for strictly validating clients, an entire sub-zone might become invisible [8].

An attacker in possession of the private key to a ZSK or KSK could mount an active attack to forge DNSSEC responses for any descendants below that location in the chain. These responses could be crafted to point to invalid descendants, making verification of records from that point on will make authentication of records fail and result in data being marked as bogus.

In the case that a KSK is compromised, the attacker is able to add its own ZSK and sign the zone's keyset using this new ZSK. From that point on the adversary proceeds as in the previous case. Both compromises are externally visible but are indistinguishable from a regular key roll-over (described below). Because KSKs are only used for rolling ZSKs, they should be kept offline most of the time. With the exception of the trust anchor in the root zone, establishing a new KSK also requires the interaction of the parent zone as parents need to update their DS records as well. Involving another party in the attack increases complexity and attack visibility.

Key Roll-over. The procedure to roll over a key in DNSSEC depends on the type of key. To establish a new KSK pair, trust in the current key must be ensured; either there exists a signed and validated DS record in the parent zone or a trust anchor has been explicitly configured. In the former case, the parent zone must add a new DS record and remove the obsolete one. In the latter case, there is no superior key to anchor the rolling key. To allow resolvers build trust into the incoming key, a "hold-down" phase is used in which the incoming key is published but not yet active. After this phase the key is effectively rolled over to the new one [87].

For a ZSK roll-over, DNSSEC-validating resolvers need access to a signature corresponding to a valid ZSK. The key roll conventionally involves introducing a new ZSK signed by the zone's KSK. The new record will eventually propagate as the old cached entries expire. At this point the new ZSK is ready for use and all entries in the zone can be re-signed. The old key is kept to allow old cached entries to be validated and will finally be removed after another TTL period [72].

Roll-over of the Root Zone KSK is particularly precarious. Every validating resolver maintains a local copy of this key, which is updated through the operating system's software update mechanism or by the resolver software itself by retrieving well-known URLs (e.g., `https://data.iana.org/root-anchors/root-anchors.xml`). If some resolvers are stranded with the old KSK, they will no longer operate as intended until being reloaded with the new KSK value. The Root Zone KSK uses a five year key roll schedule, but the roll-over has not been done since 2010 due to availability concerns and unpredictable outcome after the update procedure [47].

Finding 10

Generating, securing, and managing cryptographic keys is known to be challenging [14]. To fill this gap, hosted key management systems have emerged. Hosted PKIs centralize trust in yet another party; this creates high value targets and negates the benefits of independent key management.

6.2.2. Kill Switches without Key Compromise

Inducing Verification Failures. In addition to refusing to validate correct signatures from invalid keys, DNSSEC resolvers will also refuse to validate incorrect signatures. Thus an attacker on the path upstream from a validating resolver can cause signature validations to fail by modifying traversing DNS packets. Because DNSSEC does not provide confidentiality of queries and responses, the attacker can selectively cause responses to fail (e.g., for a specific domain). Of course, if a client can use a different path to a resolver (e.g., through a VPN), the attack will no longer succeed.

Root key injection. Resolvers themselves (including running on end-user machines) can be targeted through a root public key injection attack, which would compromise the chain of trust [1]. The attack surface for key injections may grow depending on the root key update mechanism used by the resolver and/or host operating system.

Even tough DNSSEC provides unforgeable authentication of RRsets, it does not protect against misconfiguration or bogus information on the authoritative name server. As a consequence, if a client repeatedly receives bad information for a particular zone and fails to verify the response, the zone will become invisible.

Denial of Service and Other Attacks. As with any network service, DNSSEC servers are vulnerable to Denial of Service (DoS) attacks. While servers in the root zone are highly replicated (anycast) and globally distributed, smaller TLDs or name servers for specific domains may be less resilient to attacks.

DNS responses for a DNSSEC-signed domain are typically larger than those of an unsigned domain, which makes the problem even worse. Checking signatures results in increased processing cost per DNS message and also an increased number of messages needed to answer a query for validating resolvers. An attacker can mount a large DoS attack on specific resolvers and take them offline. DNSSEC allows attackers to amplify their attack volume. A recent study [95] shows that the average amplification of DNSSEC exceeds that of regular DNS by a factor of 6-12. In extreme cases amplification in the hundreds is possible. Thus, components of the DNSSEC infrastructure are a valuable target.

6.3. Triggering Kill Switches in BGPsec

RPKI enables administrative control over delegated IP address ranges, but also brings disadvantages. The reliability and redundancy of the Internet is supported by its highly distributed nature. Using RPKI seriously constrains these properties, forcing reliance on a small number of authorities. As with DNSSEC, we categorize the possible kill switches into those where a key is compromised and those where it is not.

6.3.1. Kill Switches with Key Compromise

Key Compromise Vectors. The threats to key compromise for DNSSEC (see Section 6.2.1) also apply to BGPsec. In addition to those threats, major network equipment vendors have been unable to protect their codebase against backdoors [30, 52]. Some regional registries (e.g., RIPE and ARIN) offer their customers "hosted" versions of RPKI, where the registry handles all key management operations as well as generation of ROAs. Hosted RPKI adds yet another trusted party to BGPsec, and makes registrars who offer the service high value targets.

Certificate Forgery. An adversary might compromise a private key associated with a router or an AS. Using this key, the adversary can forge updates that appear to have passed through the compromised AS. The forged updates can be injected into neighboring ASes from any adversary-controlled AS. This would effectively transfer the affected address space to the adversary or allow redirection of traffic. If the forgery is used in a remote AS, the neighbors might not accept an incoming update directly from the victim's AS. In this case the adversary can take a legitimate route traversing through the compromised AS and position itself as the next hop to become on-path [53]. An on-path adversary can trivially drop traffic, black-holing the victim's AS.

Certificate Revocation. RPKI makes conventional use of Certificate Revocation Lists (CRLs) to revoke certificates that are no longer valid but have not yet expired. A revocation may happen for several reasons including key rollover or termination of the resource allocation. CRLs are always issued by the CA which issues the corresponding certificates, and updated at declared, regular intervals. Delegated parties must publish their own version of a CRL. All RIRs update their CRL every 24 hours for online and hosted member CAs, and every 3 months for offline CAs.

A certificate revocation triggered by a parent CA causes other BGPsec-enabled networks to reject routing updates from the targeted AS. This effectively shatter the visibility of the prefix.

An attack that results in the revocation and replacement of a key or certificate causes the affected subject to be unable to sign new objects using his private key. An adversary controlling intermediate routers could drop propagation of withdrawal messages and thus make its attack more persistent.

6.3.2. Kill Switches without Key Compromise

Certificates, manifests, CRLs, and ROA repositories are critical elements in RPKI. An attacker compromising those repositories or their publication points could remove signed objects, inject invalid objects, or replace existing objects with older but still valid objects. Because relying parties cache the data they acquire, they are able to partially mitigate such attacks by reverting to the cached data, or possibly by accepting stale data.

Another critical point is a compromise of management functions and tools used for RPKI. The adversary could modify the local routing policy to black-hole certain routes or trigger certificate revocations causing router certificates or ROAs to be revoked. By requesting new ROAs, the adversary could re-allocate prefixes allocated to the target. This results in other networks believing that these prefixes no longer originate from the affected network [53].

Furthermore, BGPsec is unable to achieve desired properties such as blackhole-resistance or loop-free routing due to wormhole and mole attacks [60]. These attacks could be used to blackhole traffic or overload network links.

Finding 11

Hierarchical, centralized trust infrastructures are not well-suited for the distributed nature of Internet constituents. These infrastructures give parent authorities unilateral control over delegated resources, enabling kill switches.

6.4. Triggering Kill Switches in SCION

As with any other network architecture, SCION exposes a set of possible attack vectors which allow to trigger kill switches. SCION's trust architecture is fundamentally different than opposed hierarchical infrastructures, SCION uses isolation domains to logically separate a compound of ASes, and each ISD manages their own trust roots instead of a single global entity providing those roots. All entities inside an ISD need to follow the ISD's policy. Depending on the set-up in a real world deployment, the largest ISDs at the top level could be compared with TLD zones in DNSSEC; whereas a sub-ISD would represent SLD zone. Despite not having a centrally controlled trust, kill switches in SCION are to some extent still possible. In this section, we discuss the necessary steps for an adversary to execute such an attack and categorize them by whether they target a local or a remote ISD/AS.

Similar to DNSSEC and BGPsec, the criticality of a kill switch (or rather "impact" as described in section 6.6) depends on the location where the attack takes place. A compromise of a core AS might allow an adversary to shut-down the entire ISD; whereas a compromise of a non-core AS affects only ASes downstream of the adversary or sub-ISDs rooted at the compromised AS. However a fundamental difference is that a core ISD is less likely to be switched off by a parent authority, as they manage their own trust roots. The trust hierarchy is only two levels per ISD deep. The core certifies all other non-core ISD members. In case of nested ISDs, a non-core AS acts as a core for a sub-ISD (which adds another level to the trust hierarchy).

6.4.1. Local ISD kill switches

An adversary local to an ISD is able to execute a kill switch in various ways. However, visibility and persistency of these are different from DNSSEC and BGPsec. All parts of the SCION infrastructure are crucial for operation. If an attacker is present at a network link or border router of a core AS, he could block incoming path requests and therefore trivially shut down communications traversing the core. Other parts such as Border routers, Certificate servers, or Beacon Servers represent also a valuable target to influence connectivity inside and even outside of an AS. In the following we discuss the consequences of such misbehaviour.

Compromise of Beacon servers. As Beacon servers are fundamental for path discovery, they represent a critical part of the SCION infrastructure. If a core BS is affected, a possible adversarial act is to stop generating (as a master) or propagating (as a slave) intra-ISD beacons, which eventually will lead to concealment of intra-ISD paths. In case of non-core BS, the attacker might stop propagating beacons to customer ASes. Both cases are externally visible to a victim as they observe the absence of beacons.

Compromise of Certificate servers. Certificate servers are queried by beacon servers when validating the authenticity of PCBs (i.e., when a beacon server does not have a corresponding certificate). Thus a compromised server could return fake certificates, which makes validation of the PCBs fail. As a victim this failure is hard to differentiate from invalid PCBs being disseminated. In doubt, he would need to query another, trusted CS. But if no trusted servers can be reached, this would effectively become a kill switch.

Compromise of Path servers. To shut down path servers in victim ASes, an adversarial entity controlling an ISD (e.g., a government) could compel ASes to stop replying to path requests. Further, ASes upload a set of segments through which they want to be reached to a path server in the ISD Core. An adversary could try to diminish this set and return only a subset of the available paths to increase the likelihood of selecting paths under his control (and follow up with black-holing). This can effectively hinder traffic from getting out of an ISD.

Key leakage. If an AS-specific key gets leaked, the consequences are catastrophic as authenticity of OFs and path revocation messages cannot be guaranteed. An adversary might simply generate own OFs, validly sign them using the leaked key. The attacker can inject the forged OF upstream relative to the victim and replace the original OF. As the packet traverses the victim AS, the AS would successfully verify the integrity of the packet. By doing this he affects the traffic flow, which could be used to lead it to a blackhole. The victim AS, where the leakage happened, can inspect the forged OF, but not distinguish it from a benign misconfiguration (e.g., invalid ingress-/egress interface, misconfigured path server that returns invalid paths). Even if the victim would have recorded the MAC for a PCB before sending it downstream, an adversary can trivially find a second preimage that results in the same message tag (see Section 4.3.2).
If the adversary is located downstream relative to the victim AS, he can still replace the OFs for all bypassing packets and affect the traffic flow, because the response packet is sent upstream along the reversed path.
In case the authentication mechanism for path revocation is also compromised (e.g., the adversary is present at a border router), the adversary can issue an authenticated path revocation for every incoming request, which hinders the affected ASes to select a valid path. Both aforementioned attacks can be combined to execute a route flapping attack. After the path discovery phase, the adversary will eventually revoke the discovered path, which renders the victim unable to establish a valid path. A leaked key could even be used remotely (e.g., inject a path revocation message into a remote AS that wants to use a path traversing the victim AS.

6.4.2. Remote ISD kill switches

Since SCION ISDs independently manage their own keys and namespace, it is not possible for an attacker to cause a kill switch in a different ISD through a network attack. We assume that all ISDs are highly connected; otherwise if an ISD is only reachable through a single other ISD core, which is possibly controlled by a malicious entity, a kill switch is still possible (e.g. drop intra-ISD beacon messages). In general the attacker is limited to data plane attacks. Even with access to private trust root keys in the remote ISD, he would need access to an AS to inject faulty information.

Denial of Service. As one of the main goals SCION focuses on availability even in the presence of adversaries, which is an exceedingly challenging property to achieve. Even tough SCION includes various mechanisms (e.g., traffic prioritization, multi-path communication, etc.) to mitigate these attacks, resource exhaustion at some point cannot be ruled out. Denial of service as such can be triggered locally or remotely, if we assume that paths into the remote ISD exist, and might also include collusion amongst several compromised ASes. For a detailed analysis of availability in SCION, we refer to section 4.4 and 5.8.

Finding 12

Triggering a kill switch in SCION is more difficult compared to other centralized infrastructures and there exists no global kill switch. In the worst case, an entire ISD is affected by a kill-switch event, however this requires the adversary to be local to the ISD and have control over a core AS. Furthermore, remote ISD kill switches are not possible.

6.5. Recovering from Kill Switches

6.5.1. Recovery in DNSSEC and BGPsec

In both DNSSEC and BGPsec, the general recovery process when a key is compromised involves creating a new key pair and notifying the parent of the key update. At low levels of the trust chain (e.g., for `cs.university.edu` or for a small ISP), this process can be as simple as submitting the hash of the new key through a web form and waiting for DNS caches to expire or routing tables to converge. However at higher levels, and especially at the DNSSEC/BGPsec root, generating a new key pair and disseminating its existence downstream can require coordination among numerous parties and endure an extended time period. Furthermore, the attacker may have taken down networks required for recovery such as payment processors or communication infrastructure (e-mail, support fora, other ISP communication channels).

Concretely, when a non-root DNSSEC KSK is compromised, the victim must generate a fresh key and transmit it to the parent zone for publication as a new DS entry. In the unlikely event of a Root Zone KSK compromise, as per ICANN's emergency KSK roll-over procedures [62] an interim trust anchor will be generated and published within 48 hours. We note that the success of the emergency key roll-over procedure depends on deployment of automatic key roll-over support, as specified in RFC 5011 [87]. Lack of support requires human involvement to update the Trust Anchor, causing further delays while administrators are unavailable. Due to the temporary nature of the interim Trust Anchor, a scheduled key ceremony needs to take place to establish another root KSK according to the root zone DNSSEC practice statement [62]. However, a key compromise at the root level is difficult to achieve even for a powerful adversary.

If a ZSK has been compromised (but the KSK is safe), recovery requires signing and publishing a new ZSK key using the unaffected KSK. As KSKs are kept mostly offline, this delays the recovery. In case of the Root ZSK, VeriSign has procedures for unscheduled roll-over in place, but does not state a specific recovery time [63].

An important factor that limits recovery for DNSSEC is caching: we find that root ZSK records have a

TTL of 48 hours, and DS records from the root to the TLDs all have a 24 hour TTL. We also analyzed popular second level domains (SLD)[1]. While less than 2% of domains had DS records, half of those had a TTL of 24 hours while the rest had TTLs of around 1 hour.

A successful recovery process needs to include an investigation on how the compromise happened in order to mitigate attacks similar attacks in the future.

Furthermore, the private component of a key pair might be permanently lost. If the Trust Anchor is permanently lost, this loss will be at latest be detected at a key ceremony. A new key is established either at the same ceremony or in another ceremony within 48 hours [62].

Recovery after a key compromise in BGPsec is accomplished by establishing a new key and revoking forged certificates through a CRL. After a new key-pair has been created, the affected AS makes the new certificate available to the RPKI global repository and is propagated to RPKI caches within one cache update cycle. The RPKI cache refresh frequency may be chosen by the operator, but typically lies between 1-24 hours [16]. In anticipation of possible key compromise, an operator could pre-provision each router's *next* key in the RPKI to eliminate the propagation delay. From this point in time, BGP updates are signed using the new key and it takes one update cycle to fully propagate. The affected AS also publishes a CRL including the serial number of the old certificate. After the CRL entry has been published (in general every 24 hours [79] or 1 business day [7]), the updated entry must be disseminated. In the worst case, victims would not fetch the updated CRL in a timely fashion, allowing an attack to persist for a full day.

For kill switches without key compromise, recovery is simpler. These events typically do not involve client action, but rather waiting for the attack to conclude or bypassing the affected network path.

6.5.2. Recovery in SCION

Recovery time in SCION is shorter compared to current other centralized architectures. Due to lack of real world deployment, it is hard to quantitatively describe recovery. Instead we make the assumptions to give reasonable bounds for recovery in SCION (see Table 6.1).

We focus on describing the necessary steps to recover and give an estimate wherever possible. Furthermore, an adversarial kill switch is less likely to be successful as SCION provides multipath communication and path transparency which ideally speeds up the recovery process.

In case a core AS is affected by an attack, downstream ASes will eventually realize that it is acting maliciously. As a first measure, the malfunctioning AS could be excluded from the core. Exclusion is lower bounded by the path revocation process, which minimally takes about 2 seconds to reach every AS (5 levels of nesting $\cdot$ 4 hops per AS $\cdot$ 100ms RTT $\approx$ 2 seconds) and upper bounded by the path lifetime (24 hours). Second, downstream ASes could self-organize and form a new core to recover. By now operating autonomously, the new ISD can begin path discovery and traffic forwarding.
If the key of a core AS gets compromised and following its certificate must get renewed, this implies reissuing the TRC. Since it is cross-certified by neighboring ISDs, this process might be time consuming. We assume cross-certification is completed within 24 hours (similar to BGPsec and DNSSEC). In the best case, where all partners are available for cross certification, this process might take only a few seconds.

In event of a non-core AS key compromise, the impacted AS needs to obtain a new certificate from the core. This process will vary depending on internal issuance protocols. In the meantime valid OFs can

[1]Alexa top 10 000, www.alexa.com/topsites.

Name	Boundary
Levels of ISD nesting	1 to 5 levels
Average AS path length	4 hops[1]
Average network latency	$\approx$ 100 ms[2]
Path lifetime	24 h
OF lifetime	5 min to 24 h
TRC cross certification time	< 24 h

Table 6.1.: Assumptions made to estimate recovery times in SCION.

[1] Average AS Hop length in todays Internet: https://labs.ripe.net/Members/mirjam/interesting-graph-as-path-lengths

[2] Average worldwide RTT to Google: https://www.igvita.com/2012/07/19/latency-the-new-web-performance-bottleneck/

still be used. An AS downstream related to the attacked AS could simply select unaffected paths or try to establish new links (e.g. a new peering link) in order to increase their connectivity and possibly mitigate such an attack. Recovery in this case is bound by the path discovery process.

If a specific instance of the SCION infrastructure (e.g., a BS) has been compromised by an attacker, the fastest way to recover is to rely on other, redundant parts. In case they are not available, the victim would need to set up other instances, which might delay recovery.

A DoS attack will not trigger a kill switch event once a unaffected path has been discovered or more resources have been made available. Ultimately, it will end along with inbound connectivity. Once paths to the victim are no longer available, the attacker can only continue to send traffic until the path expires, since an end to end path can no longer be built thereafter. As of writing, the default path lifetime in SCION is 24 hours.

Finding 13

Recovery from a kill-switch event takes less or equal amount of time compared to BGPsec/DNSSEC. A key compromise within an AS can be resolved within 24 hours in the worst case. In the best case, where all partners are available for cross certification, this process might take only a few seconds. A malfunctioning service is easily detectable and can be recovered in a short amount of time (i.e., within 2 seconds using revocation).

6.6. Towards Kill Switch Evaluation

Assessing the effectiveness of a kill switch is challenging since many of the aspects that need to be considered are difficult to quantify. In this section we propose a set of properties intended to enable reasoning about, evaluating, and comparing kill switches.[2] Table 6.2 provides a preliminary qualitative

[2]Despite its importance, we do not consider the financial cost (to an adversary) of a kill switch because obtaining a precise cost measurement is challenging. The cost of a kill switch depends on numerous random factors, for instance on the existence of software vulnerabilities, or on human error.

	Attacker capabilities	Impact	Visibility	Recoverability
DNSSEC	Root KSK/ZSK compromise	●	○	>48h
	TLD KSK/ZSK compromise	◑	◑	>24h
	SLD KSK/ZSK compromise	○	●	(<24h)
	Inject root key into resolvers	○	●	~
	DoS against resolvers/NSes	○	◑	~
BGPsec	Root key compromise	●	○	<48h
	RIR key compromise	◑	◑	<48h
	RPKI hosting compromise	○	◑	<48h
	Wormhole/mole attack [60]	◑	○	~
SCION	Core AS compromise	●	◑	<24h
	Non-core AS compromise	◑	◑	<24h
	Malfunctioning Core AS	◑	○	~
	Malfunctioning Non-core AS	○	○	~
	DoS aimed at infrastructure	◑	◑	<24h

Table 6.2.: Properties of different types of kill switches. The qualitative values range from best for security ○ (low impact/high visibility) to worst ● (high impact/low visibility). The indications are qualitative estimates, and they are specified in terms of how "good" they are for security, from ideal ○ (least convenient for the adversary), to worst case ● (best for the adversary). A "~" for recoverability indicates that the value cannot be estimated in an absolute way.

comparison of the kill switches discussed in this paper based on our proposed properties. Quantitative values are a rough estimate based on our findings in Section 6.5.

Impact. Intuitively, the *impact* of a killswitch is the amount of damage the killswitch causes. The ideal metric would be expressed in terms of financial loss, but because of the difficulty in concretely computing such a metric we suggest to use network disruption as a metric instead, i.e., what percentage of Internet communications is blocked by the killswitch. We can simplify the metric further by considering what percentage of *potential* communications are disrupted, so that the result does not depend on the actual amount of communications between hosts, and instead we consider only the fraction of pairs of end hosts which cannot communicate because of the killswitch (i.e., we assume that all pairs of end hosts are equally likely to communicate with each other). Further, because it is hard to quantitatively describe impact, we use qualitative values (high impact, medium impact, and low impact).

Visibility. The *visibility* of a kill switch is directly related to its impact. Visibility should be measured in relation to detection mechanisms designed to raise an alert in case of the activation of a kill switch. Dainotti et al. [24] suggest using Internet Background Radiation-based detection for BGP-based anomalies, while for DNSSEC we hypothesized a distributed monitoring network continuously checking DNS infrastructure. Because of the distributed nature of DNS it is hard to construct an infrastructure that can detect more targeted attacks [58]; so to attain more advanced detection capabilities a system similar to Certificate Transparency would be required [98].

Recoverability. This metric considers how easily or quickly the network can recover. As with impact, we propose the use of a simplified metric that does not consider financial costs of recovery, but solely considers the time to recovery. Depending on the analysis, it could be useful to consider the range between the best-case and worst-case recovery time. In cases where partial recovery can be performed quickly, but full recovery takes longer (e.g., client software updates are needed), a more useful metric would be the time to 90% recovery.

Precision and Predictability. While the previous properties concern a specific instance of the activation of a kill switch, properties relative to the goals of the adversary can be considered as well. These properties specify the degree of control that the adversary has over the kill switch prior to its execution. The first such property is the *precision* of a kill switch with respect to its target area, intended as the collateral damage that the kill switch causes. It could be quantified as the ratio between the impact outside the target area, and the impact within the target area. The second property is *predictability*, i.e., how well the adversary can predict the effects of triggering the kill switch.

6.6.1. Discussion

There appears to be an inherent relationship between attack feasibility, visibility, and recoverability. For example, the execution of a kill switch high up at the DNS root would require successfully bypassing several layers of physical and digital access control, or expending large amounts of resources on denial of service of root servers. However, if the root KSK were replaced (e.g., to execute a global DNSSEC kill switch), it is reasonable to expect many resolver operators would notice malfunction (e.g., through support calls or log messages). Recovery from such an attack could be time consuming, in particular if it requires coordination amongst several remote parties.

6.7. Comparison between current and next generation Trust Architectures

In SCION, kill switch events are harder to trigger compared to current monopolistic architectures; mainly due to the different trust and isolation design. Isolation domains reduce the scope of a successful attack and pursue the privilege reduction principle (see Section 2.5.2). Trust agility allows an ISD to select its own trust roots and makes them less vulnerable to a kill-switch event by a parent entity. Still, SCION can be compared to a trust model with multiple root of trusts (e.g., BGPsec without global root) where the biggest ISDs represent top-level domains (such as .ch,.com, etc.). Kill switches are possible to some extent and a core-compromise affects a domain similarly as one in a monopolistic architecture.

The transparency feature helps with visibility for external parties, especially if an AS is malfunctioning. Also the system is less prone to misconfiguration. As an optimization, redundant infrastructure parts in a reserve mode could be considered, so that if an entity is malfunctioning or was compromised, it can be entirely replaced by one of the reserve parts. The replaced entity could then be resolved without any effect on the infrastructure.

6.8. Summary

In this chapter, we have shown that hierarchical, centralized trust infrastructures are ill-suited for the distributed nature of Internet constituents. These infrastructures give parent authorities unilateral control over delegated resources, enabling kill switches. Not only are Internet kill switches feasible, they have potentially devastating impact due to the time-consuming processes needed for recovery. Moreover, attacks simultaneously targeting BGPsec and DNSSEC may create circular dependencies, further delaying restoration of service.

Using SCION does not fully mitigate such attacks, but generally makes it more difficult to execute them. Path transparency highly contributes to the visibility of attacks. As most services are considered to be executed redundantly, this decrease the recovery time. The high frequent path discovery process, makes the architecture more agile to attacks against a specific part of the infrastructure and possibly maintain availability as long as an attacker-free path between endpoints exists.

7 Conclusion

SCION is designed with security as a core feature. It achieves its goal of improving security using security principles that have proven their reliability in the past. Privilege separation is implemented using separation between control and data plane. Privilege reduction can be achieved using isolation domains and separation between core- and non-core ASes, and makes SCION more resilient against global attacks compared to today's Internet. Source-controlled paths and transparency over the chosen paths allow monitoring and enable accountability at the AS-level. Using trust agility and multiple trust roots, fixed and centralized trust architectures are avoided. Errors are inevitable in any design process, and maintainability is a key feature to future-proof SCION. Versioning, while allowing backward compability, is still difficult to achieve. Thus, errors must be eliminated earliest possible. Open-sourcing SCION's design and code will contribute to finding and fixing issues in a timely manner.

In this thesis, we preformed an in-depth analysis of the SCION future generation Internet architecture. We assumed an adversarial position, either off-path or on-path, in a SCION network and investigated possible attack vectors. Part of our analysis reviewed the routing protocols incorporated by the SCION control plane, where an adversary might try to interfere with path selection, perform traffic hijacking, or manipulate paths. We also evaluated how misconfiguration and attacks against availability affect a SCION network. For each attack found, we were able to classify it using our newly proposed SCION attack taxonomy and proposed solutions to mitigate the attack. To eliminate vulnerabilities in the current implementation, we performed a static and dynamic code analysis and examined all uses of cryptographic functions and key derivations.

Further, we analysed the MAC length that is used in SCION and formally argued that it should be increased. As an adversary might also thwart the availability of a specific service, we discussed issues and possible mitigations besides service replication. We compared trust-related issues, the so-called adversarial kill switches, between BGPsec respectively DNSSEC and the trust architecture chosen by SCION (using ISDs). Hierarchical, centralized trust infrastructures are indeed vulnerable to kill switches, as these infrastructures give parent authorities unilateral control over delegated resources. Our analysis shows that SCION does not have a global kill switch and partially mitigates this threat.

Even tough we have identified additional mechanisms to improve the security of SCION, these have proven to be out of scope of this thesis.

While the weak path integrity property using chaining of neighbouring OFs has shown to be secure against off-path adversaries and simple splicing attacks, we would like to have further guarantees on the security that this property achieves. In the future, we will work on formally verifying this aspect, either using automated theorem provers or a formal proof, which should help qualify the required properties.

Another line of research that appears promising, is finding a solution to decide whether a chosen path fulfils certain properties such as stability, guaranteed bandwidth, etc. We would like to identify where and how much information theoretic characteristics and topological knowledge can most effectively improve Internet availability, specifically in the case that multiple paths are available.

In conclusion, we have shown that a clean-slate redesign of an Internet architecture indeed improves security in multiple aspects compared to today's Internet. SCION could potentially become one of the key building blocks of tomorrow's Internet. We hope that this work contributes to a secure future Internet architecture.

A

Appendix

How to use HP Fortify SCA

C/C++

The C/C++ code must be first compiled using `sourceanalyzer` to be prepared for source code analysis. This can be done manually by simply invoking the binary as follows:

```
sourceanalyzer -b my_buildid gcc helloworld.c
```

The build command can also be integrated with a build command (e.g., `make`).

```
sourceanalyzer -b SCIONtest touchless make
```

Fortify runs the make command. When make invokes any command that Fortify SCA determines is a compiler, the command is processed by Fortify SCA. Note that the makefile is not modified.

Another option is to modify a makefile to invoke Fortify by replacing any calls to the compiler, archiver, or linker in the makefile with calls to Fortify.

```
CC=sourceanalyzer -b mybuild gcc
```

After the code has been build, we build the analysis model and scan it for vulnerabilities by referencing its build ID.

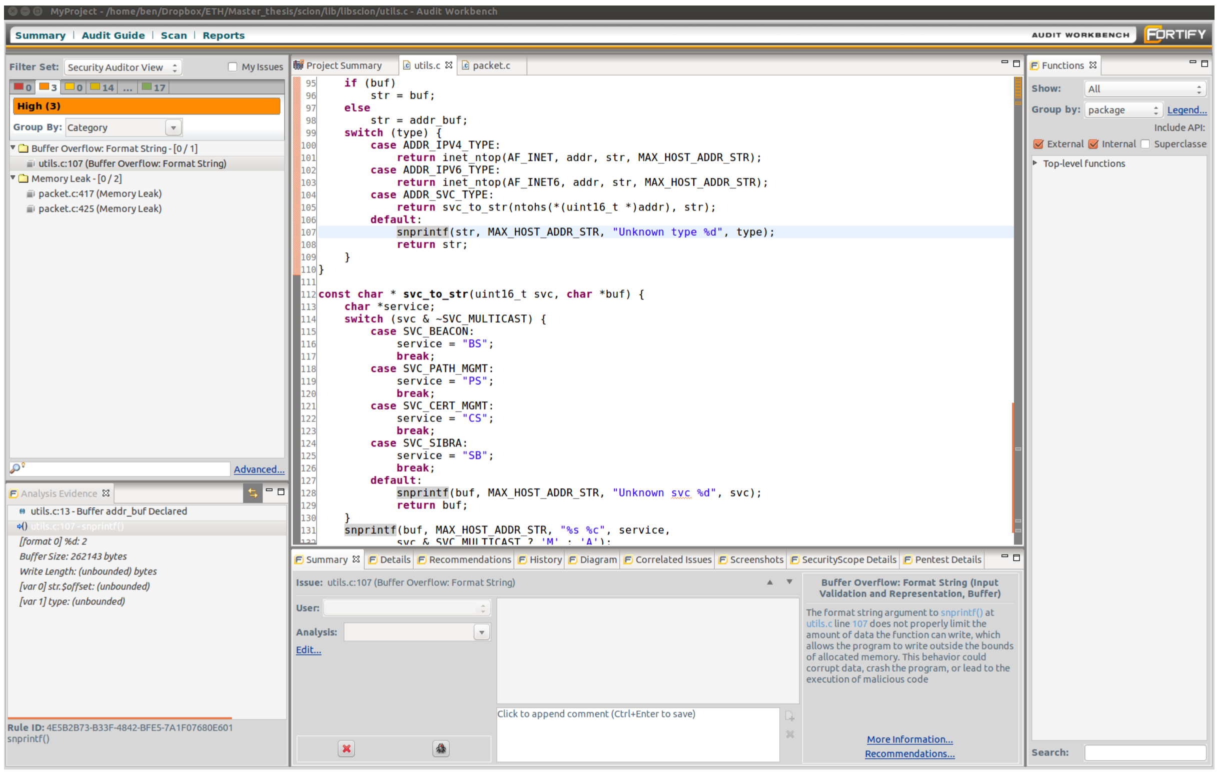
```
sourceanalyzer -b SCIONtest -scan -f SCIONtest.fpr
```

When the scan has been finished, the results are analysed using the `AuditWorkbench` (see Figure A.1) or summarized in a report using:

```
ReportGenerator -format pdf -f SCIONtest.pdf -source SCIONtest.fpr
```

Python

Either the code can be translated to a SCA model using similar commands as before (without specifying a compiler), or the Fortify `ScanWizard` can automatically generate a shell script which automates this process depending on the project root.



MyProject - /home/ben/Dropbox/ETH/Master_thesis/scion/lib/libscion/utils.c - Audit Workbench

Filter Set: Security Auditor View

High (3)

Group By: Category

- Buffer Overflow: Format String - [0 / 1]
  - utils.c:107 (Buffer Overflow: Format String)
- Memory Leak - [0 / 2]
  - packet.c:417 (Memory Leak)
  - packet.c:425 (Memory Leak)

```c
    if (buf)
        str = buf;
    else
        str = addr_buf;
    switch (type) {
        case ADDR_IPV4_TYPE:
            return inet_ntop(AF_INET, addr, str, MAX_HOST_ADDR_STR);
        case ADDR_IPV6_TYPE:
            return inet_ntop(AF_INET6, addr, str, MAX_HOST_ADDR_STR);
        case ADDR_SVC_TYPE:
            return svc_to_str(ntohs(*(uint16_t *)addr), str);
        default:
            snprintf(str, MAX_HOST_ADDR_STR, "Unknown type %d", type);
            return str;
    }
}

const char * svc_to_str(uint16_t svc, char *buf) {
    char *service;
    switch (svc & ~SVC_MULTICAST) {
        case SVC_BEACON:
            service = "BS";
            break;
        case SVC_PATH_MGMT:
            service = "PS";
            break;
        case SVC_CERT_MGMT:
            service = "CS";
            break;
        case SVC_SIBRA:
            service = "SB";
            break;
        default:
            snprintf(buf, MAX_HOST_ADDR_STR, "Unknown svc %d", svc);
            return buf;
    }
    snprintf(buf, MAX_HOST_ADDR_STR, "%s %c", service,
            svc & SVC_MULTICAST ? 'M' : 'A');
```

Analysis Evidence

- utils.c:13 - Buffer addr_buf Declared
- utils.c:107 - snprintf()

[format 0] %d: 2
Buffer Size: 262143 bytes
Write Length: (unbounded) bytes
[var 0] str.$offset: (unbounded)
[var 1] type: (unbounded)

Rule ID: 4E5B2B73-B33F-4842-BFE5-7A1F07680E601
snprintf()

Issue: utils.c:107 (Buffer Overflow: Format String)

User:

Analysis:

Click to append comment (Ctrl+Enter to save)

Buffer Overflow: Format String (Input Validation and Representation, Buffer)

The format string argument to snprintf() at utils.c line 107 does not properly limit the amount of data the function can write, which allows the program to write outside the bounds of allocated memory. This behavior could corrupt data, crash the program, or lead to the execution of malicious code

Show: All

Group by: package

Include API: ✓ External ✓ Internal ☐ Superclasse

Figure A.1.: Fortify AuditWorkbench Example Use.

Bibliography

[1] Adil Alsaid and Chris J Mitchell. *Revised Selected Papers of EuroPKI 2005*, pages 227–239. Springer, 2005.

[2] Nate Anderson. How China swallowed 15% of 'Net traffic for 18 minutes. http://arstechnica.com/security/2010/11/how-china-swallowed-15-of-net-traffic-for-18-minutes/, 2010.

[3] Tom Anderson, Ken Birman, Robert Broberg, Matthew Caesar, Douglas Comer, Chase Cotton, Michael J Freedman, Andreas Haeberlen, Zachary G Ives, and Arvind Krishnamurthy. The nebula future internet architecture. In *The Future Internet Assembly*, pages 16–26. Springer, 2013.

[4] Apache Zookeeper. https://zookeeper.apache.org/. Date accessed: 2016-06-07.

[5] APNIC. Use of DNSSEC Validation for World. http://stats.labs.apnic.net/dnssec/XA?c=XA&x=1&g=1&r=1&w=7&g=0.

[6] R. Arends et al. DNS Security Introduction and Requirements. RFC 4033 (Proposed Standard), 2005.

[7] ARIN. ARIN CPS for resource certification. https://www.arin.net/resources/rpki/cps.pdf, 2012.

[8] Suranjith Ariyapperuma and Chris J Mitchell. Security vulnerabilities in DNS and DNSSEC. *Proceedings - Second International Conference on Availability, Reliability and Security, ARES 2007*, pages 335–342, 2007.

[9] Hitesh Ballani, Paul Francis, and Xinyang Zhang. A study of prefix hijacking and interception in the internet. *ACM SIGCOMM Computer Communication Review*, 37(4):265, 2007.

[10] Elaine B. Barker and Allen L. Roginsky. SP 800-131Ar1. Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths. Technical report, NIST, Gaithersburg, MD, United States, 2015.

[11] David Barrera, Raphael M. Reischuk, Pawel Szalachowski, and Adrian Perrig. SCION Five Years Later: Revisiting Scalability, Control, and Isolation on Next-Generation Networks. *arXiv preprint arXiv:1508.01651*, pages 1–21, 2015.

[12] Cristina Basescu, Raphael M. Reischuk, Pawel Szalachowski, Adrian Perrig, Yao Zhang, Hsu-Chun Hsiao, Ayumu Kubota, and Jumpei Urakawa. SIBRA: Scalable Internet Bandwidth Reservation Architecture. *arXiv*, 2015.

[13] Mihir Bellare, Joe Kilian, and Phillip Rogaway. The security of the cipher block chaining message authentication code. *Journal of Computer and System Sciences*, 61(3):362–399, 2000.

[14] Robert Beverly, Ryan Koga, and KC Claffy. Initial longitudinal analysis of IP source spoofing capability on the Internet. *Calhoun Faculty and Researcher Publications*, 2013.

[15] Chad Brubaker, Suman Jana, Baishakhi Ray, Sarfraz Khurshid, and Vitaly Shmatikov. Using frankencerts for automated adversarial testing of certificate validation in SSL/TLS implementations. *Proceedings - IEEE Symposium on Security and Privacy*, pages 114–129, 2014.

[16] R. Bush. Origin Validation Operation Based on the Resource Public Key Infrastructure (RPKI). RFC 7115 (Best Current Practice), 2014.

[17] By Kevin Butler, Student Member Ieee, Toni R Farley, Patrick Mcdaniel, Senior Member Ieee, Jennifer Rexford, and Senior Member Ieee. A Survey of BGP Security Issues and Solutions. *Proceedings of the IEEE*, 98(1), 2010.

[18] The capnp Tool. `https://capnproto.org/capnp-tool.html`, 2016. Date accessed: 2016-03-03.

[19] Rocky K C Chang. Defending against flooding-based distributed denial-of-service attacks: A tutorial. *IEEE Communications Magazine*, 40(10):42–51, 2002.

[20] David Cheriton. The Internet Architecture: Its Future and Why it Matters. `http://www.sigcomm.org/sites/default/files/award-talks/cheriton-sigcomm03.pdf`, 2013.

[21] R. Cohen, K. Erez, D. Ben-Avraham, and S. Havlin. Breakdown of the internet under intentional attack. *Physical Review Letters*, 86(16):3682–3685, 2001.

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

[23] Nicolas T. Courtois and Gregory V. Bard. Algebraic Cryptanalysis of the Data Encryption Standard. Cryptology ePrint Archive, Report 2006/402, 2006. `http://eprint.iacr.org/2006/402`.

[24] Alberto Dainotti et al. Analysis of Country-Wide Internet Outages Caused by Censorship. *IEEE/ACM Transactions on Networking*, 22(6):1964–1977, 2014.

[25] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. Technical report, DTIC Document, 2004.

[26] D. Dolev and a. C. Yao. On the security of public key protocols. *22nd Annual Symposium on Foundations of Computer Science (sfcs 1981)*, pages 198–208, 1981.

[27] Morris J. Dworkin. SP 800-38B. Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication. Technical report, NIST, Gaithersburg, MD, United States, 2005.

[28] ENISA. Algorithms, key size and parameters report. Technical report, ENISA, 2014. https://www.enisa.europa.eu/publications/algorithms-key-size-and-parameters-report-2014.

[29] S. Farrell and H. Tschofenig. Pervasive Monitoring Is an Attack. RFC 7258 (Best Current Practice), 2014.

[30] FireEye Security. SYNful Knock - A Cisco router implant. `https://perma.cc/X3TX-ZJNM`, 2015. Date accessed: 2016-05-12.

[31] Gartner: Magic Quadrant in Application Security Testing. http://techbeacon.com/highlights-2015-gartner-magic-quadrant-application-security-testing, 2015. Date accessed: 2016-03-03.

[32] V. D. Gligor. The Crossfire Attack. *2013 IEEE Symposium on Security and Privacy*, pages 127–141, 2013.

[33] P Godfrey, Igor Ganichev, Scott Shenker, and Ion Stoica. Pathlet routing. *ACM SIGCOMM Computer Communication Review*, 39(4):111–122, 2009.

[34] Jeffrey Goldberg. On hashcat and strong Master Passwords as your best protection. https://blog.agilebits.com/2013/04/16/1password-hashcat-strong-master-passwords/, 2013. Date accessed: 2016-05-15.

[35] Sharon Goldberg. Why is it taking so long to secure internet routing? *Communications of the ACM*, 57(10):56–63, 2014.

[36] Dieter Gollmann. Computer security. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(5):544–554, 2010.

[37] Glenn Greenwald. Edward Snowden: NSA whistleblower answers reader questions. https://www.theguardian.com/world/2013/jun/17/edward-snowden-nsa-files-whistleblower, 2013.

[38] Glenn Greenwald. *No place to hide: Edward Snowden, the NSA, and the US surveillance state*. Macmillan, 2014.

[39] Simon Hansman and Ray Hunt. A taxonomy of network and computer attacks. *Computers & Security*, 24(1):31–43, 2005.

[40] Hashcat. https://hashcat.net/, 2016. Date accessed: 2016-06-15.

[41] Nadia Heninger, Zakir Durumeric, Eric Wustrow, and J. Alex Halderman. Mining Your Ps and Qs: Detection of Widespread Weak Keys in Network Devices. In *USENIX Security*, 2012.

[42] Micheal Hicks. Type safety. http://www.pl-enthusiast.net/2014/08/05/type-safety/, 2014. Date accessed: 2016-05-14.

[43] Samuel Hitz, Pawel Szalachowski, Severin Meier, Steven Shirley, and Adrian Perrig. SPRING: Secure Path Revocation in Next Generation Internet Architectures. Technical report, ETH Zürich, 2016. (under submission).

[44] Fraser Howard. A closer look at the Angler exploit kit. https://blogs.sophos.com/2015/07/21/a-closer-look-at-the-angler-exploit-kit/, 2015.

[45] John D Howard and Thomas A Longstaff. A common language for computer security incidents. *Sandia National Laboratories*, 1998.

[46] Dorota Huizinga and Adam Kolawa. *Automated Defect Prevention: Best Practices in Software Management*. Wiley-IEEE Computer Society Pr, 2007.

[47] Geoff Huston. Rolling Roots. http://labs.apnic.net/?p=796, 2016.

[48] ICANN. TLD DNSSEC Report. http://stats.research.icann.org/dns/tld_report/.

[49] ICANN. Trusted Community Representatives - Proposed Approach to Root Key

Management. http://www.root-dnssec.org/wp-content/uploads/2010/04/ICANN-TCR-Proposal-20100408.pdf, 2010.

[50] Privacy International. Mass surveillance. https://www.privacyinternational.org/node/52, 2016.

[51] ISO. Key derivation. Standard, International Organization for Standardization, 2010.

[52] Juniper. Important Juniper Security Announcement. https://perma.cc/53LM-JES4, 2015. Date accessed: 2016-05-12.

[53] S. Kent and A. Chi. Threat Model for BGP Path Security. RFC 7132 (Informational), 2014.

[54] Auguste Kerckhoffs. *La cryptographie militaire*. University Microfilms, 1978.

[55] Tiffany Hyun-Jin Kim, Cristina Basescu, Limin Jia, Soo Bum Lee, Yih-Chun Hu, and Adrian Perrig. Lightweight source authentication and path validation. In *ACM SIGCOMM Computer Communication Review*, volume 44, pages 271–282. ACM, 2014.

[56] O. Kolkman, W. Mekking, and R. Gieben. DNSSEC Operational Practices. RFC 6781 (Informational), 2012.

[57] Leslie Lamport et al. Paxos made simple. *ACM Sigact News*, 32(4):18–25, 2001.

[58] Ben Laurie. Certificate Transparency. *ACM Queue*, 12(8), 2014.

[59] M. Lepinski and S. Kent. An Infrastructure to Support Secure Internet Routing. RFC 6480 (Informational), 2012.

[60] Qi Li, Yih-Chun Hu, and Xinwen Zhang. Even rockets cannot make pigs fly sustainably: Can BGP be secured with BGPsec? In *NDSS Workshop on Security of Emerging Networking Technologies (SENT)*, 2014.

[61] Wilson Lian, Eric Rescorla, Hovav Shacham, and Stefan Savage. Measuring the Practical Impact of DNSSEC Deployment. In *USENIX Security*, 2013.

[62] F. Ljunggren, T. Okubo, R. Lamb, and J. Schlyter. DNSSEC Practice Statement for the Root Zone KSK Operator. https://www.iana.org/dnssec/icann-dps.txt, 2010.

[63] F. Ljunggren, T. Okubo, R. Lamb, and J. Schlyter. DNSSEC Practice Statement for the Root Zone ZSK Operator. http://www.root-dnssec.org/wp-content/uploads/2010/06/vrsn-dps-00.txt, 2010.

[64] K. Lougheed and Y. Rekhter. Border Gateway Protocol (BGP). RFC 1105 (Experimental), 1989.

[65] Georgios Mantas, Natalia Stakhanova, Hugo Gonzalez, Hossein Hadian Jazi, and Ali A Ghorbani. Application-layer Denial of Service Attacks: Taxonomy and Survey. *International Journal of Information and Computer Security*, 7(2-4):216–239, 2015.

[66] Zhuoqing Morley Mao, Ramesh Govindan, George Varghese, and Randy H Katz. Route flap damping exacerbates Internet routing convergence. In *ACM SIGCOMM Computer Communication Review*, volume 32, pages 221–233. ACM, 2002.

[67] Moxie Marlinspike. SSL And The Future Of Authenticity. https://moxie.org/blog/ssl-and-the-future-of-authenticity/, 2011.

[68] Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot. *Handbook of Applied Cryptography*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 1996.

[69] Metasploit: Penetration testing software. https://www.metasploit.com/. Date accessed: 2016-06-15.

[70] Jelena Mirkovic and Peter Reiher. A taxonomy of DDoS attack and DDoS defense mechanisms. *ACM SIGCOMM Computer Communication Review*, 34(2):39–53, 2004.

[71] P.V. Mockapetris. Domain names - concepts and facilities. RFC 1034, 1987.

[72] S. Morris, J. Ihren, J. Dickinson, and W. Mekking. DNSSEC Key Rollover Timing Considerations. RFC 7583 (Informational), 2015.

[73] Global Prefix/Origin Validation using RPKI. http://rpki-monitor.antd.nist.gov/, 2016.

[74] Chris Palmer. Against Security Nihilism. https://noncombatant.org/2016/01/28/against-security-nihilism/, 2016. Date accessed: 2016-02-24.

[75] Jianli Pan, Subharthi Paul, and Raj Jain. A survey of the research on future internet architectures. *Communications Magazine, IEEE*, pages 26–36, July 2011.

[76] Colin Percival and Simon Josefsson. The scrypt password-based key derivation function. *IETF*, 2015.

[77] Niels Provos and David Mazieres. A Future-Adaptable Password Scheme. In *USENIX Annual Technical Conference, FREENIX Track*, pages 81–91, 1999.

[78] Dipankar Raychaudhuri, Kiran Nagaraja, and Arun Venkataramani. MobilityFirst: A Robust and Trustworthy Mobility- Centric Architecture for the Future Internet. *SIGMOBILE Mob. Comput. Commun. Rev.*, 16(3):2–13, 2012.

[79] RIPE. RIPE NCC RPKI CPS. https://www.ripe.net/publications/docs/ripe-549, 2012.

[80] J. H. Saltzer, D. P. Reed, and D. D. Clark. End-to-end Arguments in System Design. *ACM Trans. Comput. Syst.*, 2(4):277–288, November 1984.

[81] Jerome H Saltzer and Michael D Schroeder. The Protection of Information in Computer Systems. *Proceedings of the IEEE*, 63(9):1278–1308, 1975.

[82] Bruce Schneier. Applied cryptography: Protocols, algorithm, and source code in C. *Government Information Quarterly*, 13(3):336, 1996.

[83] Chris Simmons, Charles Ellis, Sajjan Shiva, Dipankar Dasgupta, and Qishi Wu. AVOIDIT: A cyber attack taxonomy, 2009.

[84] Sooel Son and Vitaly Shmatikov. The hitchhiker's guide to DNS cache poisoning. *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering*, 50 LNICST:466–483, 2010.

[85] Kotikalapudi Sriram and Matthew Lepinski. BGPsec Protocol Specification. https://tools.ietf.org/html/draft-ietf-sidr-bgpsec-protocol-15, 2016.

[86] William Stallings. *Cryptography and Network Security*, volume 139. Pearson Education India, 5 edition, 2011.

[87] M. St.Johns. Automated Updates of DNS Security (DNSSEC) Trust Anchors. RFC 5011 (Internet Standard), 2007.

[88] Ahren Studer and Adrian Perrig. The coremelt attack. In *European Symposium on Research in Computer Security*, pages 37–52. Springer, 2009.

[89] Nick Sullivan. DNSSEC: An Introduction. https://blog.cloudflare.com/dnssec-an-introduction, 2014. Date accessed: 2016-04-02.

[90] L. Szekeres, M. Payer, T. Wei, and D. Song. SoK: Eternal War in Memory. In *IEEE Security and Privacy (SP) Symposium*, pages 48–62, May 2013.

[91] Ari Takanen. *Fuzzing for Software Security Testing and Quality Assurance*. Artech House, 2008.

[92] M S Turan, E B Barker, W E Burr, and L Chen. Recommendation for password-based key derivation. Technical report, NIST, 2010.

[93] Luke Valenta et al. Factoring as a Service. Cryptology ePrint Archive, Report 2015/1000, 2015. http://ia.cr/2015/1000.

[94] Paul C Van Oorschot and Michael J Wiener. Parallel collision search with cryptanalytic applications. *Journal of cryptology*, 12(1):1–28, 1999.

[95] Roland van Rijswijk-Deij, Anna Sperotto, and Aiko Pras. DNSSEC and its potential for DDoS attacks: a comprehensive measurement study. In *Internet Measurement Conference*, pages 449–460. ACM, 2014.

[96] Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Collision search attacks on SHA1, 2005.

[97] Xiaowei Yang, David Clark, and Arthur W Berger. NIRA: a new inter-domain routing architecture. *IEEE/ACM transactions on networking*, 15(4):775–788, 2007.

[98] Dacheng Zhang, Daniel Kahn Gillmor, Danping He, Behcet Sarikaya, and Ning Kong. Certificate Transparency for Domain Name System Security Extensions. https://tools.ietf.org/html/draft-zhang-trans-ct-dnssec-03, 2016.

[99] Xin Zhang, Hsu-Chun Hsiao, Geoffrey Hasker, Haowen Chan, Adrian Perrig, and David G Andersen. Scion: Scalability, control, and isolation on next-generation networks. In *2011 IEEE Symposium on Security and Privacy*, pages 212–227. IEEE, 2011.

ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby con rm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

Security Analysis of a Future Internet Architecture

Authored by (in block letters):
For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
Rothenberger	Benjamin

With my signature I con rm that
- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were signi cant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**
Zürich, 01.08.2016	B. Rothenberger

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.