Enabling Individually Entrusted Routing Security for Open and Decentralized Community Networks

Axel Neumann[a], Leandro Navarro[a], Llorenç Cerdà-Alabern[a]

[a] *Universitat Politècnica de Catalunya*

Abstract

Routing in open and decentralized networks relies on cooperation. However, the participation of unknown nodes and node administrators pursuing heterogeneous trust and security goals is a challenge. Community-mesh networks are good examples of such environments due to their open structure, decentralized management, and ownership. As a result, existing community networks are vulnerable to various attacks and are seriously challenged by the obligation to find consensus on the trustability of participants within an increasing user size and diversity. We propose a practical and novel solution enabling a secured but decentralized trust management. This work presents the design and analysis of securely-entrusted multi-topology routing (SEMTOR), a set of routing-protocol mechanisms that enable the cryptographically secured negotiation and establishment of concurrent and individually trusted routing topologies for infrastructure-less networks without relying on any central management. The proposed mechanisms have been implemented, tested, and evaluated for their correctness and performance to exclude non-trusted nodes from the network. Respective safety and liveness properties that are guaranteed by our protocol have been identified and proven with formal reasoning. Benchmarking results, based on our implementation as part of the BMX7 routing protocol and tested on real and minimal (OpenWRT, 10 Euro) routers, qualify the behaviour, performance, and scalability of our approach, supporting networks with hundreds of nodes despite the use of strong asymmetric cryptography.

Keywords: routing, trust, decentralized security, multi-topology, cooperation, mesh networks, community networks

1. Introduction

Community mesh networks [1, 2, 3] are ideally open and decentralized structures, growing and evolving organically as network infrastructure is contributed, deployed, and configured by their participants. Typically, a deployment of such networks, such as Guifi.net [4] with more than 34,000 total active nodes, is structured in different network clouds [5], each consisting of up to hundreds of nodes, constituting autonomous systems (AS), operating its cloud-specific internal routing protocol (e.g. OSPF and, OLSR), and peering with neighbouring clouds via an exterior gateway protocol (e.g. BGP).

The operation of such networks is based on the principle of cooperation among the members. These communities usually have participation rules, such as a membership licence or peering agreement [6, 7, 8], that define their freedom, openness and neutrality. Nonetheless, current designs and implementations of mesh networks impose comprehensive technical definitions and restrictions to achieve functional data transit and end-to-end delivery among any pair of network nodes [2]. That includes the use of a specific routing protocol and routing metric so that nodes can consistently learn and inform about the state of the network and update their own routing tables. In practice, due to the lack of mature implementations, only a very few of the proposed routing protocols are used in real deployments or have been experimentally analyzed [3, 9, 10, 11]. Among them there are the AODV [12, 13], Babel [14, 15], BMX6 [16], the widely used OLSR [17, 18, 19], and batman-adv [20] protocol implementations.

One shortcoming of the current solutions is given by the lack of routing-security support that comes without introducing centralized dependencies (e.g. certificate authorities) [21], which would contradict with the open and the decentralized objectives of such networks. Another problem lies in the protocol requirements for unified parametrization of metrics and policies to determine Quality of Service (QoS), routing, trust and security decisions for all network nodes [22]. Such a strong level of unification prohibits the usage of individually defined policies and limits its openness. It also imposes a substantial effort, increasing with the number and diversity of community members, for finding consensus on related questions.

To dilute the limitations of a single and unified set of QoS parameters for routing, QoS multi-topology (MT) routing has been proposed [23, 24], allowing the concurrent support of multiple virtual topologies (on top of a single physical topology), each established based on a different definition of QoS parameters. In OLSRv2 [19] MT is used for routers to support more than one link metric type. In this paper we use the MT con-

Email addresses: axel@ac.upc.edu (Axel Neumann), leandro@ac.upc.edu (Leandro Navarro), llorenc@ac.upc.edu (Llorenç Cerdà-Alabern)

Please cite this article as: A. Neumann, L. Navarro, L. Cerdà-Alabern, Enabling Individually Entrusted Routing Security for Open and Decentralized Community Networks, Ad Hoc Networks (2018), https://doi.org/10.1016/j.adhoc.2018.06.014

cept to concurrently support different security and trust sets. A network could for example maintain one topology (a) for nodes trusted by organization A, and a second topology (b) usable only by nodes certified via organization B.

The security design of the protocol proposed in this work ensures that each node is the only authority able to define and publish its set of individually trusted nodes based on which forwarding rules (routes) for delivering its traffic should be selected, propagated, and maintained. This way, our protocol pursues the MT approach as it establishes dedicated virtual topologies for each participating node. It further supports the cooperative, open, and decentralized philosophy that enables community networking, as deployed network infrastructures remain open for other nodes to join and be used while being independent from any central entity.

This paper extends [25] the initial design and analysis of the securely-entrusted multi-topology routing (SEMTOR) protocol. In summary, the main contributions of this work are:

- Propose novel, secured, and decentralized routing protocol mechanisms called SEMTOR. With SEMTOR users can define individual trust sets of nodes, which are the only ones allowed to route their traffic.
- Summarize the assumptions, and respectively achieved safety and liveness properties and prove their correctness with formal reasoning.
- Describe how SEMTOR is implemented in BMX7 by extending BMX6 (BMX6 + SEMTOR = BMX7), a routing protocol currently used in production community wireless mesh networks [5].
- Experimental validation of the resistance of the SEMTOR implementation to various attack vectors and challenging network scenarios.
- Analysis of the performance and resource requirements of SEMTOR by measuring traffic, CPU, and memory overhead. The results demonstrate the scalability of SEMTOR to support existing mesh-network deployments and using inexpensive off-the-shelf WiFi hardware.

The remainder of this paper is organized as follows. After reviewing related work in Section 2, we identify the addressed problems and design objectives in Section 3 and detail the system model with further assumptions and definitions in Section 4. Section 5 describes the design and mechanisms of our protocol to solve these objectives which is then validated from an formal and an experimental perspective in Section 6.3 and 7. This includes the presentation of our prototypical implementation and its functional and performance evaluation using embedded router hardware in a virtualized network environment. We discuss the contributions, open issues, and adjacent security solution in Section 8 and conclude in Section 9.

2. Background and Related Work

Existing work on secure routing for ad hoc and mesh networks has been reviewed in [26, 27]. Authenticated routing for ad hoc networks (ARAN) [28] as proposed by Sanzgiri et al. and admittance-control enabling extensions for OLSRv2 proposed by Herberg et al. [29] use digital signatures to verify the authenticity and integrity of control messages. Both rely on the existence of a central certificate server trusted by all participating nodes. Babel hash-based message authentication code (HMAC) cryptographic authentication [30] relies on one or several pre-deployed shared keys to validate messages via attached message authentication codes (MAC). However, the requirement for preserving shared keys as a private secret within an open network community disqualifies related approaches for any open Community Network (CN).

In addition, SEAD [31] and SAODV [32] encounter the dependency on a central trust authority with a self-securing control plane. Using Anchored Hash Chains (AHCs) to protect the mutable hop counter field of routing- update messages they ensure that a malicious node cannot claim better distances to any remote node than it really has. However, both remain vulnerable to data-plane attacks such as packet dropping or routing-table poisoning. Moreover, SAODV proposes the use of digital signatures to protect non-mutable data in routing messages. To avoid the dependency of a certification authority as a central root of trust that guarantees the binding between node public keys (nodePKs) and other node properties such as their IP address Zapata [33] proposed to bind the identity of nodes given by their public key to their allocated address by building it based on the hash of the public key.

Work in [34] and [35] addressed the problem of misbehaving nodes by punishing malicious nodes based on their forwarding behaviour as observed and assessed by neighbouring nodes. Adnane et al. [34] build on top of SOLSR and extend it with detection and reaction mechanisms. Mogre et al. [35] present another holistic approach combining self-securing routing, detection, reputation, and counter-measure mechanisms.

Introduced in [25], SEMTOR follows a different approach. In fact, guaranteeing in all aspects the correct operation of nodes is indeed hard and, as pointed out by Adnane et al. [34], cannot be guaranteed (e.g. data-plane attacks cannot be prevented) by securing the topological information exchanged between nodes. Therefore, instead of aiming to ensure or enforce correct operation, SEMTOR enables each node admin to freely define their individual subset (and resulting sub-topology) from the complete set of participating nodes that the admin considers sufficiently trustworthy to meet the security and data-delivery objectives and concerns. In addition, none of the presented work relying on asymmetric cryptography for verification of control messages has yet been analyzed in terms of performance or benchmarked based on embedded hardware and exposed to traffic and network characteristics that are typical for existing community mesh-network clouds. An overview of related work on routing security for IP-based mesh networks is given in Table 1.

An impressive amount of further related research about wireless mesh networks has been done in recent years. Selected publications are ordered thematically with respect to the importance for the objectives of this work. The case of community mesh networks is discussed in terms of legal implications,

Table 1: Related routing security work for IP-based Wireless Mesh Networks

	Admission & Control	Authentication	Control-Plane Security	Data-Plane Security	Routing Metric	Routing Basis	Usable	Evaluation
OLSRv2 ext. [29]	centralized	CA signed	CA trust	CA trust	*	proactive, link state	×	simulation
Babel HMAC [30]	orchestrated	shared key	group trust	group trust	*	proactive, DSDV	✓	×
ARAN [28]	centralized	CA signed	consistent	CA trust	RTT	reactive	×	simulation
SEAD [31]	open	AHC	consistent	insecure	HC	proactive, DSDV	×	simulation
SAODV [32, 33]	open	self-signed, AHC	consistent	insecure	HC	reactive, DSDV	✓	×
VRR [36]	open	self-signed	consistent	insecure	VRR,*	hybrid, SR, DHT	✓	sim., testbed
Trust OLSR [34]	open	signed reputation	punishment	detection	*	proactive, link state	×	simulation
AntSec [35]	centralized	CA-signed	punishment	detection	probabil.	proactive, stigmergic	×	simulation
CONFIDANT [37]	open	PGP reputation	punishment	detection	*	reactive, DSR	×	simulation
SEMTOR [25, 38]	individual, open	self-signed	individually trusted	individually trusted	*	proactive, DSDV	✓	CN realistic

motivation, design, and business models in [39, 40, 1, 7, 41, 42, 21]. In addition, scalability and performance aspects of routing protocols are handled in [43, 44, 45, 46, 47, 48, 9]. Trust and security related work is surveyed and discussed in [49, 27, 50, 51, 52], with solutions for particular routing functions in [53, 34, 54, 31, 37, 55, 56, 57], and presentations of holistic security frameworks in [36, 37, 58, 35, 59]. The last four also present measurement results based on simulation. Approaches towards supporting different or user-defined routing policies are handled in [22, 60]. Traffic validation, or how to recognize a misbehaving path, node or link and which information is needed, is addressed by sketches [61, 62, 63, 64], counters [65], fingerprinting [66] or sampling [62]. Distributed detection, the assessment of anomalous and faulty nodes based on sharing of distributed observations, considering the arbitrary behaviour of malicious nodes, is addressed by Π_2 and Π_{k+2} [67] in general, or by KDet [68], which is specifically for CNs.

3. Problem Statement

The goal of SEMTOR is to provide secure mechanisms to ensure that non-trusted nodes in an open network are effectively prevented from disrupting the routing between trusted nodes.

In multi-hop mesh networks, end-to-end routing is a distributed task that relies on the contribution of network resources by nodes and honest collaboration between them. However, in an open network, this allows new or unknown nodes to join without pre-conditions. Therefore, malicious or misconfigured nodes with byzantine behaviour can participate and interfere with the collaboration with faulty operations or adverse contributions. As such, the routing system is one of the most complex and fragile but, unfortunately, also one of the least protected components in a network infrastructure [51].

In such a hostile environment, our mechanism shall ensure that routing-related tasks such as path detection, (efficient and consistent) route establishment, and end-to-end packet forwarding are robust against any faulty operations from non-trusted nodes.

With reference to the work of Mizrak et al. [67], the problem of detecting and handling compromised nodes in a network can be split into the three sub-problems: (i) local traffic validation: traffic behaviour characterization based on local observation, (ii) distributed detection: assessment of anomalous and faulty nodes based on sharing of distributed observations, and (iii) response: enforcing the exclusion of a given set of identified faulty nodes.

In that sense, our protocol addresses the third sub-problem of a response. More precisely, given a known subset of faulty or distrusted nodes (from the overall set of existing nodes), how can we ensure that no subset of these nodes can negatively affect any routing-critical task? The other two sub-problems of traffic validation and distributed detection are out of the scope of this work, but there are complementary solutions suitable for this scenario with sketches [64] for the first problem, and KDet [68] for the second. Further, no assumptions are made on how a required subset of trusted nodes (or its relative complement from the set of overall nodes, the set of non-trusted nodes) is obtained. It is just taken for granted, as each community mesh-network can implement its own way to produce and maintain these lists.

3.1. Objectives

For the design of the protocol, the following four objectives have been identified and are briefly summarized as follows.

Regarding **ownership** of data traffic, forwarding routes, and node identities, the objective is that any CN-related routing-table entry and packet can be attributed to exactly one node of the network. Each node can be unambiguously identified with a secure identity and an identity-proving address. In addition, each node is the exclusive owner of exactly those routes pointing towards this address and those packets carrying this address as destination.

Regarding **security in terms of autonomy and robustness**, the general objective is that any contiguous group of nodes (node admins), trusting and willing to cooperate and support each other, can not be prevented by an external entity (e.g. an adversary) from doing so.

Regarding **openness and decentralization**, the objective is that each node admin can individually decide which of the other nodes to trust and rely on, that multiple cooperative groups of nodes (admins) can coexist, that group membership is not exclusive, and that there is no need for a central registry or authority. Nonetheless, similar to social networking or so-called

networks of trust, public-key servers or other decentralized coordination platforms may be used to facilitate the management of known and trustable nodes and corresponding node IDs.

Regarding **scalability**, an implementation of the proposed mechanisms should be feasible and scalable for the characteristics (e.g., number of nodes and links per node) of typical CN clouds and given the resource limitations (e.g., CPU, memory, and bandwidth) of low-budget but state-of-the-art embedded routers used in today's CN deployments.

4. System Model

4.1. Community Model

Figure 1 illustrates the key scenarios, assumptions, and characteristics that are consistent with real-world community-mesh-network clouds. Mesh clouds typically incorporate up to thousands of users, hundreds of mesh nodes, and tens of links per node [69, 70, 71, 72, 73] and can extend from a small neighbourhood to the coverage of a city or rural district. They interconnect via gateways with neighbouring community clouds and other private or public networks, such as the Internet.

The technical enablers are given by **nodes** based on wireless or wired router hardware [3] with usually limited storage, bandwidth, and computation capabilities. These nodes are contributed and controlled by individuals and mounted at low cost on rooftops or isolated rural locations powered by solar energy. The **links**, manually or automatically established between neighbouring nodes, create a **restricted-route network** without universal direct connectivity between all mesh nodes [74, 75]. As such, the overall infrastructure is owned, developed, and maintained by its **users**, who are united by the common idea to share their networking resources: They offer the service provided by their own routing hardware and links to others and benefit from those offered by others.

Users can be grouped into two stakeholder groups: **users** of the infrastructure and **administrators** that take the additional responsibility to **control** and maintain individual **nodes** (mesh routers) and other locally connected infrastructure such as **content servers** or **border gateways** to neighbouring networks. Due to the different roles, a **local trust relation** exists between the directly connected users of a node and its representative, the node administrator who implements the local user decisions. It is expected that such local node communities are rather small (typically at the scale of a house or organization) so that consensus on commons such as a licence or terms of usage (e.g., [6, 7, 8]) can be reached via direct communication. However, if consensus is impossible, an independent local node deployment, implementing a different usage policy, can be set up, such as the case for the upper right building in Figure 1, which hosts two nodes operated by different administrators.

Because of the implicit dependence on each network component being potentially involved in the cooperative task of routing and traffic forwarding, the usage of existing resources should not be imposed and the ability to select and disapprove among the overall set of existing cloud resources is important. However, for an open and decentralized CN that brings together individuals and organizations with different and even conflicting economic, political, and technical interests [76, 1, 77] and that can not prevent anybody from participating also with notoriously error-prone or even malicious configured equipment, the achievement of global consensus on the trustability of all other participating nodes and their administrators is actually impossible. Instead, it is expected that controversy exists and trustability is only given among various subsets of the overall user groups. Such **directed trust relations between admins** are exemplary and are illustrated as green arrows for users of admin E and F. The E users agreed on accepting the licence and trusting in the administrative setup of A, C, and F, while admin F and its connected users trust in B, D, and E.

4.2. Network Model and Terminology

This section introduces the terminology and assumptions used for specifying and justifying the SEMTOR approach for enabling node admins (as modelled in Section 4.1) to individually specify trusted nodes so that all non-trusted nodes can not interfere.

A network *node* is defined as a (routing) process owning an individual public/private key pair to support cryptographic operations for authentication, integrity-verification, and non-repudiation of data created by the node. The private key is assumed to be permanent, globally unique, and accessible exclusively by the key-owning node. A node is supposed to correctly perform the tasks of path detection, route establishment, and forwarding of IP data-packets as specified by algorithms and implemented by a specification-conforming *routing protocol*. A supposedly correct (benign) node may indeed not conform with the specification of correct behaviour. This may be due to accidental mis-configuration or *malicious* motivation of the person administering this node. The *public key* of a node X is denoted as K_X^{pub}. An identity function $V_X = v(K_X^{pub})$ is an injective function that assigns a label (identity) V_X to node X based on K_X^{pub}. A practical identity function $v(N)$ such as the secure SHA224 [78] hash function is assumed for this work. The set of all (correctly and incorrectly behaving) nodes is given as $\mathbb{V}$.

A *link* $E_{B,A}$ describes a directed connection that exists if a network interface of node V_B is in direct transmission range of a network interface of V_A and vice versa, providing a functioning bidirectional transmission opportunity between V_A and V_B. However, while the existence of $E_{B,A}$ implies the existence of $E_{A,B}$ it does not imply that these two links are symmetric or even identical: $E_{B,A} \neq E_{A,B}$. If more than one network interface per node is assumed then $E_{B,A}$ shall represent the best of all possible combination of links for transmitting from V_A to V_B. The set of all existing links between all nodes is given as $\mathbb{E}$.

A *path* $P_{T,S}$ is a sequence of nodes $V_h \in \mathbb{V}$ with $0 \leq h \leq s, s \geq 1$ so that with $V_T \equiv V_0$ and $V_S \equiv V_s$ the path $P_{T,S} = \{V_0, V_1, V_2, ...V_{s-1}, V_s\}$ explicitly identifies a particular sequence of nodes (connected via implicitly identified links $E_{0,1}, E_{1,2}, ..., E_{s-1,s}$ between consecutive nodes) from source node V_S to destination node V_T within G, providing a directed end-to-end transmission path with $h = \{0, 1, 2, ..., s-1, s\}$ representing the number of remaining hops (nodes) from V_h to

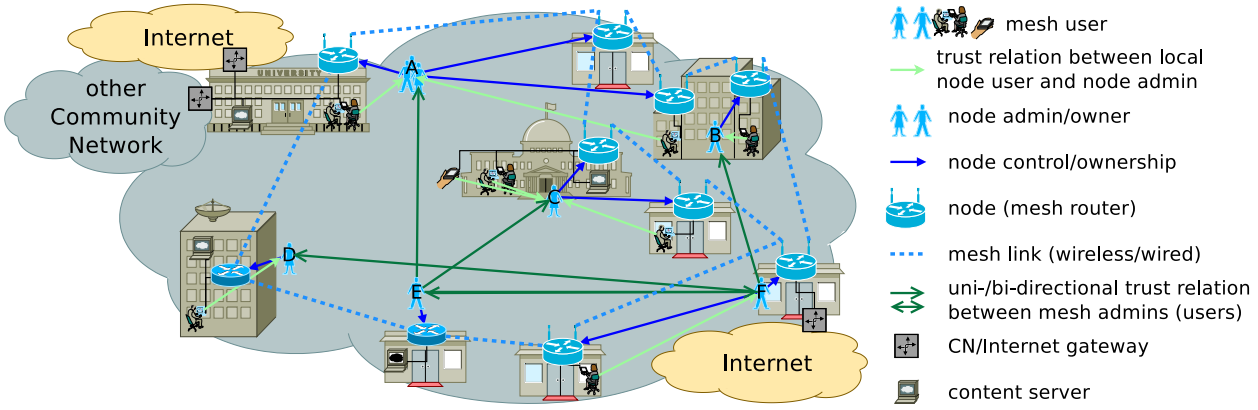


Figure 1: Model of a community mesh network

$V_T \equiv V_0$. The set of all possible paths for a given graph G and (V_T, V_S)-tuple is denoted as $\mathbb{P}_{T,S}$ with $P_{T,S} \in \mathbb{P}_{T,S}$.

Path detection denotes the task of identifying and maintaining, for a given network topology G and V_T, V_S-tuple, existing paths and path metrics so that identified (V_T, V_S)-paths could be differentiated and ranked.

Route establishment denotes the task of leveraging information from the path detection for setting up IP routing tables so that continuous and consistent forwarding paths are established between any tuple of source and destination nodes.

Forwarding denotes the task of forwarding and eventually delivering IP-data packets according to previously established routes from a source to a destination node.

Routing processes exchange routing update messages with neighboring nodes and therefore rely on the following assumptions and service primitives provided by the data-link layer:

- The network connectivity over time is given by the topology graph G during interval I.
- Links of G either exist (are up) or not (are down) and their states do not change within one particular interval I_n.
- $tx(d)$: The transmit function broadcasts data d to all neighbouring nodes to which a link from the broadcasting node exists.
- $rx(d)$: The receive function receives data d broadcasted by any node from which a link to the receiving node exists.
- The maximum transmission delay υ between any $tx(d)$ function call and corresponding $rx(d)$ events is much smaller than interval I: $\upsilon \ll I$

4.3. Threat Model and Assumptions

A threat is defined as a potential for disruption of route establishment or forwarding services towards a destination. Threat sources are given by adversaries motivated to disrupt this service and possessing the capabilities to violate the attack surface of routing processes. This attack surface is given by the exchange of protocol messages used to interact with their environment which could be created, modified, or treated in an unexpected manner.

In the following, adversaries are modelled as a class of network nodes. As such, the behaviour of a network node V_Y regarding another particular node V_X is either *correct* (benign) and complies with the protocol rules regarding V_X or it is *adverse* (malicious) and performs arbitrary behaviour (byzantine) regarding V_X that, detectable or not, deviates from the protocol rules. The set of correct nodes regarding V_X is given as $\mathbb{V}_X^c$. The set of adverse nodes regarding V_X is given as $\overline{\mathbb{V}_X^c}$ with $\overline{\mathbb{V}_X^c} = \mathbb{V} \setminus \mathbb{V}_X^c$.

Unless otherwise precluded, no assumptions are made on the methodologies and behaviour applied by adversaries to achieve their objectives. In particular, no assumptions are made on the point of time (which may be any time in the future) and the extent of deviation (which may happen at the control or data plane), collaboration (allowing coordinated and distributed attack), or communication capabilities (using extraordinary and dedicated channels) of adversaries. The following assumptions apply to all nodes (including adversaries).

Assumption 1. *Nodes cannot tamper with cryptographic primitives.*

This means nodes cannot attack fundamental cryptographic primitives and their key properties such as those given by a secure hash function, the asymmetric encryption and decryption, signing and signature verification based on a public-private key pair, the Diffie-Hellmann (DH) secure key-exchange, and a the identity and ownership proving properties provided by a Cryptographically Generated Addresss (CGAs). Nodes cannot control other nodes beyond the specified protocol rules. In particular private key material is inaccessible to other nodes, and other node services (e.g. ssh) are sufficiently secured.

Assumption 2. *Nodes cannot launch a wormhole attack [79] without being detected and without corresponding packets being discarded by the receiving node.*

This implies that it is impossible to replay entire packets without changing them so that they appear to the receiving nodes as being received directly from the node that originated the repeated packet in the first place. Such attack can be addressed by lower protocol layers, Phy or medium access-control (MAC), via packet leashes [80] or NPAs [81], or by higher level distributed detection [82].

Table 2: Summary of used symbols

Symbol	Description	Properties
$\mathbb{V}$	set of all nodes (correct or not)	
K_X^{pub}	public key of node X	
K_X^{priv}	private key of node X for K_X^{pub}	
$V_X = v(K_X^{pub})$	particular node X labeled as V_X	$V_X \in \mathbb{V}$
$A_X = a(V_X)$	IPv6 crypto-address (CGA) of V_X	
$tx(d)$	tx function broadcasting data d to all neighbours	
$rx(d)$	rx function receiving broadcasted data d from neighbour	
c^t, c^r	arbitrary transmitted or received message code, e.g. $c \in \{ping, pong, ...\}$	
I	interval between two points of time	
$S_{X,d} = s(K_X^{priv}, d)$	signature created by V_X with K_X^{priv} over data d	
$\overline{s}(K_X^{pub}, S, d)$	signature verification function	$\overline{s} : \{K, S, d\} \rightarrow \{false, true\}$
$D^c_{X,Q} = \{V_X, K_X^{pub}, Q^d_X, Q^a_{X,Q}, A_X, \mathbb{V}^t_X, \overline{M}_X, f_X\}$	unsigned description components of node V_X with sequence number $Q = Q^d_X$	
$D_{X,Q} = \{D^c_{X,Q}, s(K_X^{priv}, D^c_{X,Q})\}$	signed description of $D^c_{X,Q}$	
Q^d_X	description sequence number of D_{X,Q^d}	$Q^d \in \mathbb{N}^+$
$Q^a_{X,Q}$	AHC anchor of $D_{X,Q}$	
Q^b_X	AHC heartbeat matching $Q^a_{X,Q}$	
$Q^h_X = h(Q^a_{X,Q}, Q^b_X, V_X, Q^d_X)$	heartbeat sequence number of V_X	$Q^h \in \mathbb{N}^+$
$\mathbb{V}^c_X$	set of all correct nodes regarding V_X	
$\overline{\mathbb{V}^c_X}$	set of all adverse nodes regarding V_X	$\overline{\mathbb{V}^c_X} = \mathbb{V} \setminus \mathbb{V}^c_X$
$\mathbb{V}^t_X$	set of all nodes trusted by V_X	
$\overline{\mathbb{V}^t_X}$	set of all nodes not trusted by V_X	$\overline{\mathbb{V}^t_X} = \mathbb{V} \setminus \mathbb{V}^t_X$
$\mathbb{E}$	set of all links between all nodes	
$E_{B,A}$	particular directed link from V_A to V_B	$E_{B,A} \neq E_{A,B}$, $E_{B,A} \in \mathbb{E}$
$\mathbb{E}^t_X$	set of all links trusted by V_X	(see eq. (6))
$L_{B,A} = l(E_{B,A})$	directional link-metric quality for transmitting from A to B	$L \in \mathbb{R}^+$
$M_{T,S} = m(V_T, V_S, f_T, \hat{M}_T)$	path-metric quality from V_S to V_T depending on V_T-defined customizer function f and upper quality bound $\hat{M}_T$	$M \in \mathbb{R}^+$, $M \leq \hat{M}$
$G = g(\mathbb{V}, \mathbb{E})$	overall network topology graph	
$G^t_X = g(\mathbb{V}^t_X, \mathbb{E}^t_X)$	trusted topology graph of V_X	
$P_{T,S} = \{V_0, V_1, ..., V_n\}$	particular explicit path from V_S to V_T as ordered list of connected nodes $V \in \mathbb{V}$ from G with $V_0 \equiv V_S, V_n \equiv V_S$	
$\mathbb{P}_{T,S}$	set of all directed paths from V_S to V_T	
$\mathbb{P}^t_{T,S}$	set of paths trusted by V_T from V_S to V_T	$\mathbb{P}^t_{T,S} \subseteq \mathbb{P}_{T,S}$
$\mathbb{P}^c_{T,S}$	set of paths consisting only of correct nodes regarding V_T from V_S to V_T	$\mathbb{P}^c_{T,S} \subseteq \mathbb{P}_{T,S}$
$\mathbb{P}^{t,c}_{T,S}$	set of paths consisting only of trusted and correct nodes regarding V_T from V_S to V_T	$\mathbb{P}^{t,c}_{T,S} \subseteq \mathbb{P}^t_{T,S} \cap \mathbb{P}^c_{T,S}$
$R_{T,S} = r(V_T, V_S)$	established route from V_S to destination V_T along implicitly defined nodes	

Assumption 3. *Nodes cannot perform anonymous denial of service (DoS) attacks.*

Nodes cannot perform DoS attacks by exhausting processing capacities (in terms of memory or CPU) of neighbouring nodes without being detected as the cause of the critical resource exhaustion and their packets being discarded. Network and traffic monitoring and detection can detect and mitigate these attacks.

Assumption 4. *Nodes cannot physically disturb other nodes.*

This means that wireless channels cannot be disturbed via jamming, and traffic incurred by concurrent transmissions sharing the same local channel environment have only a marginal effect on the remaining capacity of the affected links.

All these assumptions are applicable to community mesh networks and, to the best of our knowledge, actually to any distributed networking service aiming at some sort of end-to-end data forwarding and delivery while relying on a limited set of resources. Cryptographic primitives can (and have been) based on considered secure standard work such as SHA2 hashes, RSA and HMAC signatures, and Diffie-Hellman key exchange. If one of these primitives prove to be insecure in the future, they can be replaced. Vulnerability of nodes to physical (destruction or jamming), (distributed) denial of service, or wormhole attacks are kind of obvious issues and cannot be prevented entirely unless the whole network is placed in mechanically and frequency shielded environments or based on quantum communication which is currently unfeasible for any open community network. Related issues and solutions are discussed in section 8.

5. Protocol Design

The basic idea to achieve our objectives may be best illustrated with a simple example: If person x declared that he/she fully trusts persons a and b, then any person y, that knows about the declaration of x, could hand out a value for x to a or b while fully respecting x's assumptions of trust and knowing x as being in full charge of this action and any subsequent consequences. Further, these trust declarations should not be weakened by composition (intransitive). In the above example, intransitive means that even if a or b assume (and declared) z to be trusted, they must not hand out any value for x to z unless also x explicitly declared z as trusted. Applying this idea to routing in IP networks, a person is represented by a node that uses a routing protocol to communicate the declaration of its administrator.

SEMTOR extends the concept of *receiver-driven routing* [22] (allowing nodes to express path-selection preferences via a descriptive profile) and the principles of table-driven, proactive, destination-sequenced distant-vector (DSDV) routing [83] (where sequenced routing updates, originated by each node of a network and updated and re-broadcasted by each hop, are used to propagate path cost and versioning information in the network). In contrast to the traditional DSDV protocol, containing IP address and sequence number to identify a particular node of the network and a specific version of its update, the routing update of SEMTOR contains a single and verifiable heartbeat value which unambiguously identifies a particular node of the network and a specific version of this node's self-defined description and routing-update version.

The concepts and relations between node identities, descriptions, and heartbeats are described in the following sections. A more detailed description is given in [38].

5.1. Global Node Identification and Description

A strong and permanent **node-public key** (nodePK) provides the root of trust for all operations performed on behalf of the node possessing the corresponding node-private key. The SHA hash of this public key (nodePKHash) is used as a **global ID**, an unambiguous and permanent reference for identifying the node.

The node **description** aggregates information about the node's current configuration and manifests at least the following information:

The nodePK and its **global ID** (nodePKHash).

A **description version** – sequence number (*descSqn*) that must be incremented with every **description update** or node reboot, always allowing differentiation between the latest and out-of-date versions of any node's description. The exhaustion of the 32-bit *descSqn* indicates the end-of-life of a used nodePK.

A **description signature** (descSignature) matching the permanent nodePK given in this description and used for verifying the authenticity and integrity (i.e. this description was indeed created by the node with the corresponding global ID and was not changed by anyone else) of the description.

Instead of aggregating all description data directly into a single and self-contained description message, parts of the overall description are defined from the **root description** by the SHA hash of the actual data. Such references-based definitions are called **content references** and are used for example to define public-keys or the trust sets of a node.

The approach comes with the implication that any received packet header or message, containing description or content references for which the referenced raw data is not known cannot be instantly processed and must be requested first. To resolve such situations, **description-** or **content-request** messages (containing only the unresolvable hash) are sent from the node not knowing the hash-matching raw data to the (neighbouring) node from which the packet containing this hash was directly received. The node receiving a description or content request and knowing the requested data then responds with a corresponding **description-** or **data-advertisements** messages, providing the requested data.

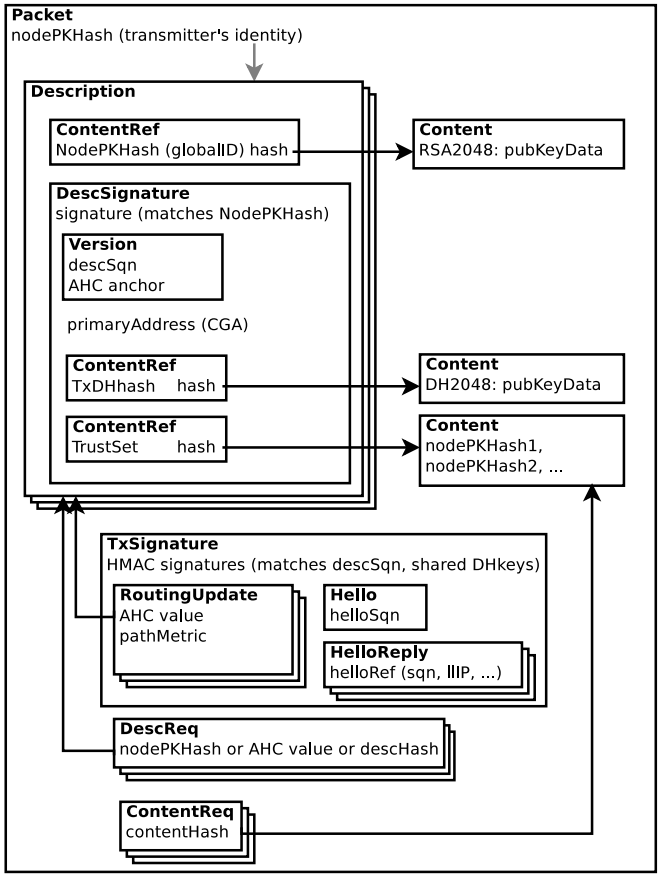


Figure 2: Protocol packet, description-, content-, and reference structure

5.2. Link Authentication

Neighbour and link authentication is used to (i) let neighbouring nodes, with a direct single-hop link between each other, detect and authenticate each other as the node they claim to be and (ii) to ensure authenticity, integrity, and non-repudiation of information exchanged between them (to avoid misunderstanding: link authentication can verify information X as being stated by neighbour A, but not that information X is necessarily true).

Since the authentication needs to be performed continuously and with a high frequency, the underneath cryptographic operation should be sufficiently lightweight to be accomplishable even by resource-constrained embedded routers.

Link authentication is achieved by authenticating exchanged messages with HMAC signatures. The to-be-authenticated message is concatenated with the hash of a shared secret (known only to the two specific link neighbours interested in authenticating the messages exchanged among them). The hash of this concatenation is concatenated a second time with a differently padded version of the shared secret. The resulting hash is then added to the list of MACs calculated for each link neighbor of a node. This list is transmitted with the original message. The link neighbour verifies the received message and MACs by matching the latter against its self-calculated MAC using the received message and the shared secret as inputs.

A variation of the DH key exchange mechanism, tailored to the concepts of SEMTOR, is used to let nodes establish the necessary shared secret over a public network. Therefore, each node (e.g. x) selects a private secret k_x (only known to itself) and publishes K_x, given as

$$K_x = g^{k_x} mod p \tag{1}$$

via its description, using pre-defined protocol constants for the modulus p and the base g. This way, any pair of nodes, given that each other's description is known, can calculate a mutual shared secret $S_{x,y}$ without additional handshake procedures as described by Garzia in [84]:

$$S_{x,y} = K_x^{k_y} mod p = K_y^{k_x} mod p \tag{2}$$

As with the transmission public keys, K values are used only temporary and can be replaced via description updates at any time. Further, a shared-key exchange is neither explicitly an-

nounced nor is its completion explicitly acknowledged. A successful shared-key exchange is only implicitly confirmed by receiving messages with a valid MAC based on a shared key calculated from the latest known descriptions of the two involved nodes.

5.3. Network Assets Authentication

5.3.1. Address, route, and traffic ownership

A **Cryptographically Generated Address (CGA)** provides the basis for identifying and proving the owner of addresses, routes, and data packets. The **primary IPv6 address** of a node provides such a CGA. It is computed by combining a 16-bit unique local address (ULA) [85] prefix with the most-significant 112 bits of the node's permanent global ID (the nodePKHash). The concept of CGAs has been proposed by [86], who also described mechanisms to calculate provable network ranges or further enhance the security of generated addresses and networks. Addresses within this 16-bit ULA prefix are owned by the node possessing the private key for the CGA mathing global ID.

CGA announcements (published via the owning nodes description) are used for declaring the ownership of CGAs to other nodes. Any description containing CGA announcements that are not matching the nodePKHash of the originating node are invalid and discarded upon reception.

Forwarding routes to a given CGA address, although installed in different routers, are owned by the node owning the destination address of the routing entry and must be maintained exclusively according to the routing updates originated by the owning node. Any received or self-created IP-data packets containing a verified CGA address as a destination address are owned by the node owning the CGA and must be forwarded according to the forwarding route configured to this CGA.

5.3.2. Sub-topology authentication

The **trust set** of a node is given as a list of global IDs of other nodes that constitute the **trusted nodes** of this node. This way, the trust set implicitly defines, with the links possible between any pair of trusted nodes, the **trusted virtual topology** of this trusting node. Only the good-listed nodes identified via this trust set are authorized for propagating routing updates originated by the trusting node. However, instead of expecting untrusted nodes to respect this demand, all trusted nodes are expected to ignore routing updates originated by the trusting node that were not received directly or securely authenticated via any of the trusted nodes. Consequently, only neighbouring nodes that are trusted nodes of a described trusting node are considered a next hop towards the trusting node and eventually, end-to-end established routing entries towards this trusting node are also only set along these trusted nodes.[1]

In addition, particular node IDs of a trusting primary node's trust set could be flagged as super-trusted nodes. This means that all nodes that are explicitly identified via the trust set of such super-trusted nodes shall also be considered trusted nodes of the primary node. More precisely, the current overall trust set of the primary node is given as the union of its own directly trusted nodes, defined via its current description, and the directly trusted nodes defined via the current description of the primary's super-trusted nodes.

This allows to **delegate**, in part or entirely, the assessment of trustworthiness to one or several other sources that are highly and individually trusted. This way administrators can be relieved from having to reconfigure all their nodes for each considered trustability change as only a single node needs to be reconfigured while all other nodes having this single node flagged as super-trusted would automatically adapt the trust set of the super-trusted single node. It also allows to delegate trust assessment to individuals (or even autonomous systems) that may be able to treat the topic of trustability with a much greater responsibility (for example due to a higher technical knowledge or better social connection with many CN participants) than the administrator of the actual delegating (primary) node.

5.3.3. Routing and heartbeat updates

An **Anchored Hash Chain (AHC) value** substitutes the destination network and sequence number of routing updates from traditional destination-sequenced distance-vector protocols such as [83]. The approach has been inspired by the SEAD protocol [31] which uses an AHC to protect the distance vector (metric) of routing updates in addition to the destination node sequence number. In SEMTOR, the IP addresses, towards which forwarding routes are maintained, are given by the CGA announcements published via the AHC-referenced description of each routing update

The creation, mapping, and validation of received AHC values has been designed as follows. Whenever a node creates a new description it picks two random 112-bit values s_0 and r_0, concatenates these with its node ID I and the upcoming *descSqn*, and calculates an initial SHA224 hash H_0 of these ingredients. This hashing is repeated n_{max} times[2] where for each iteration the first 112 bits of the output hash are used as input for s:

$$H_{n+1} = s_{n+1} \oplus r_{n+1} = SHA224(s_n \oplus r_0 \oplus I \oplus DescSqn) \quad (3)$$

The initial hash H_0 (and particularly its ingredient s_0) is kept as a private secret and is called the AHC root with which all

[1] In [38] Sect. 4.5.3.3 and 4.5.5 we also describe and discuss a mechanism to delegate the selection of trusted nodes to other individual nodes

[2] $n_{max} = 6000$ has been selected as the default to limit the needed hash power of used routing hardware, while allowing the sending of new routing updates every 6 seconds and avoiding description updates (due to exhausted AHC values) for $6sec * 6000 = 10hours$. Moreover, SHA224 has been selected as the most lightweight standard among current state-of-the-art secure hash functions. The truncation to 112 bits has been chosen considering available data from http://bitcoin.sipa.be/ (April 2017) that indicates the global cumulative number of yet calculated bitcoin double-sha256 hashes around $10^{26} \sim 2^{89}$ which is assumed to roughly correspond to that of 2^{90} single-sha224 hashes. Thus, we assume that, even given a doubling of hash power per year, a brute-force attack for finding the matching first 112 bits of a SHA224 hash remains unfeasible for the following $112 - 90 = 22$ years, especially if the attacked AHC root is invalidated every few hours with each description update.

follow-up hashes could be easily calculated. The final hash $H_{n_{max}}$ is published with r_0, I, *descSqn*, and n_{max} in the description of the node. It is called the anchor of the hash chain and allows easy verification of whether a given 112-bit value is part of this hash chain or not. Follow-up routing updates sent by an originating node start revealing the pre-calculated hash values starting with $n_{max} - 1$ towards n_1. The number of iterations i needed to match the hash-chain anchor $H_{n_{max}}$ of a received hash-chain value s_n (thus, $H_{n_{max}} = H_{n+i}$) is used to calculate the routing update sequence number (routeSqn) as:

$$routeSqn = (descSqn * n_{max}) + (i - n_{max}) \quad (4)$$

By performing a brute-force search for a hash-chain anchor in previously received and verified descriptions that matches a particular iteration output of Formula (3), receiving nodes can retain the identity of the node that originated the update and verify its authenticity and freshness in terms of the corresponding *descSqn* and *routeSqn*. This can be verified because node Id and *descSqn* are mandatory inputs for calculating the chain anchor, all of them being part of the description that has been signed with the node Id matching the public-private key pair. If a matching description can not be found for a received AHC value it is requested by sending a description request message containing the unresolvable AHC value to the node that send the update. To relieve updates-receiving nodes from the task of finding the hash-chain anchors for a given chain value, routing update messages are extended with an (16-bit) internal identifier field (IID) that works similar to IPsec's SPI and is described in more detail in [38] Ch.4.5.3.2.

Because the hash chain can not be calculated backwards, its testimony cannot be anticipated by any other node and only the originating node itself can create hash-chain values that yield a newer (or greater) *routeSqn* than any previously yielded *routeSqn* for that node. Therefore the provable *routeSqn* can always be used to identify the newest heartbeat of a node and disqualify old updates. It provides a secure heartbeat that makes so called "false destination sequence attacks" [87] impossible.

Routing updates exchanged between neighbouring nodes are covered by the link authentication mechanism described in Section 5.2, allowing verification of the authenticity, integrity, and non-repudiation of exchanged information. The signature of the sender covering the update allows the receiving node to verify whether the sender is authentic. The AHC-referenced description and contained trustSet allows the receiving node to verify if the sender is trusted by the originating node and whether the contained metric can be accepted for further path maintenance. This way, a routing update can be considered a signed promise by each propagating node, stating to the receiving nodes that (i) this particular update has been propagated exclusively via nodes trusted by the originator of the message, and (ii) that all intermediate nodes also stated their willingness and capability for routing packets (with destination addresses as expressed via the referenced and signed description) towards the originating node.

5.3.4. Optional description content

Without going into further detail, several optional node properties that are relevant in the context of computer networks can be propagated via node descriptions such as Routing metric definitions ([22] and [88]), hostname, tunnel, or tunnel-route announcements.

5.4. Bootstrapping Example

Figure 3 illustrates related bootstrapping procedures of the protocol in four phases: (1) the local node discovery, (2) link discovery, (3) global node discovery and trusted path establishment.

The local node discovery (phase 1) shows how nodes $n2$ and $n3$ react on the reception of a new (for $n2$ and $n3$ yet unknown) *nodePKHash* (I) or *descSqn* (given here by a received packet header) and resolve the description, identity (pubKey) and further referenced description content by sending corresponding **description or content request** messages (containing only the unresolvable hash) to node $n1$. Then $n1$ replies with the requested information. When all requested content has been resolved, $n2$ and $n3$ can assemble the full description and verify its authenticity with the signature and public key as defined by the initially received nodePKHash that triggered this process. The descHhash and signature are calculated over the raw description data and contained data hashes as they occur (not the referenced original data).

A transmission signature message (txSignature in Fig. 2 and 3) verifies the authenticity and integrity of all following **signature-demanding messages**. These include the locally exchanged link-discovery and routing-update messages but do not include the eventually globally exchanged descriptions (already signed explicitly via description signatures) or content advertisements (already signed implicitly via SHA reference hashes used in the signed descriptions).

To protect against link-local replay attacks, the txSignature also covers the transmission sequence number (txSqn) that again can only be reused after renewing the currently used txPK and a transmission IP (txIP) that must match the link-local src IPv6 address of the IP packet containing the protocol data.

The link discovery (phase 2) briefly indicates how the provided functionality can be used to authenticate the exchange of link-probing messages and subsequent deduced link qualities. Here, the hello message contains a sequence number that is unique during the lifetime of a node's description and the corresponding hello-reply message unambiguously references a previously received hello message and link through which this message has been transmitted (by specifying the hello-sequence number, the current description hash of the hello-sending node, and the txIP from which the hello message has been sent and received through its own interface). By receiving correctly signed hello and hello-reply messages (referencing the receiver's own hello messages via the included txSqn and txIP), the exchange of link-probing messages becomes a continuous and bidirectional challenge-response handshake that only the holder of the corresponding private keys can maintain.

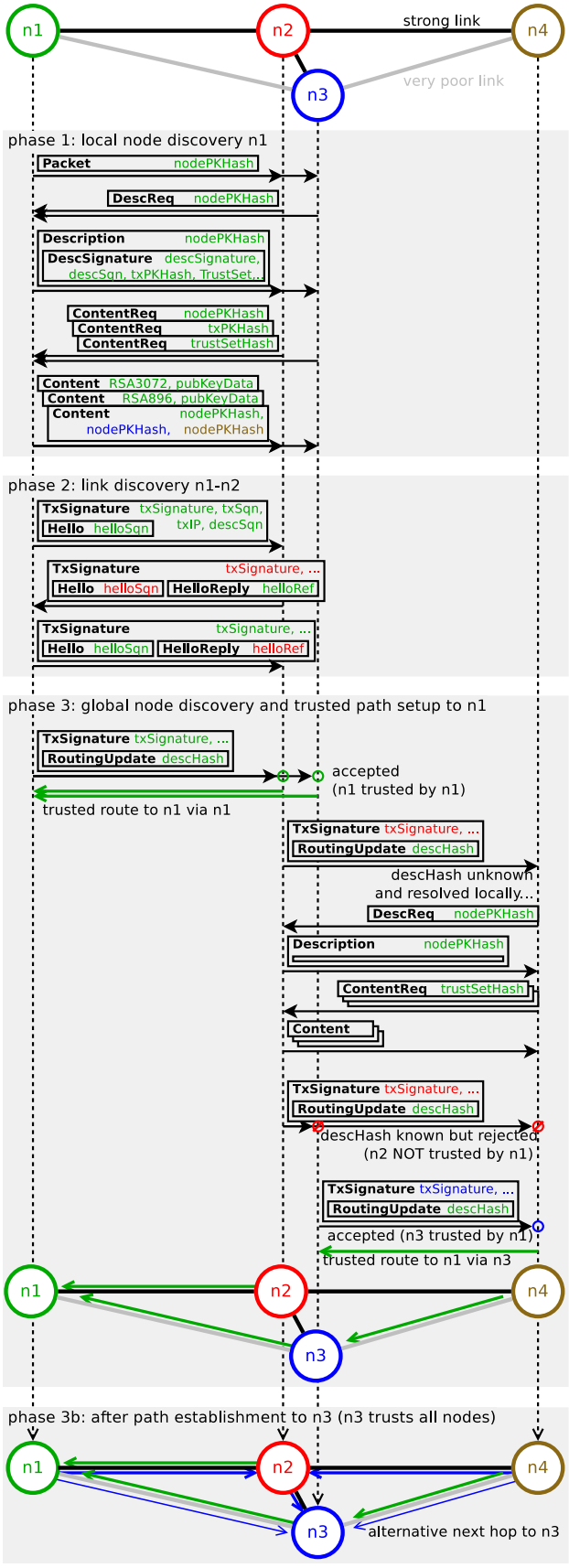


Figure 3: Network bootstrapping (node, link, and path discovery)

During the global node discovery (phase 3), nodes are discovered beyond the link neighbourhood. The discovery is triggered by the detection of an unresolvable AHC value contained in routing updates, which causes the receiving nodes to resolve and verify the full node description (similar to local node discovery in phase 1).

Phase 3 also illustrates the establishment and setup of forwarding paths along trusted nodes. First $n2$ and $n3$ receive the routing update from $n1$, whose description has already been fully resolved in phase 1. Because the update was received via an authenticated link (due to its txSignature) with known link quality (see phase 2) and because $n1$ itself is listed as a trusted node in the description, $n2$ and $n3$ know that $n1$ trusts the transmitter of the received message regarding correctly processing, updating, and re-broadcasting its routing-update messages. The end-to-end path cost to the originating node and via the trusted link peer can be calculated based on the path metric of the update and the current link quality. In case the calculated cost identifies this link as the best next hop towards the originating node, the routing update is re-broadcasted with a modified path metric value that reflects the cost of the additional link (see [89] for more details).

In the next illustrated step, node $n3$ and $n4$ receive the routing update originated by $n1$ and re-broadcasted by $n2$. This time, the update-conveying txSignature verifies $n2$ as the transmitter and last modifier of the routing update. However, the identity (nodePKHash) of $n2$ is not listed in the trust set described by the originating node $n1$; therefore, the update is discarded (by $n3$ and $n4$) for path establishment towards $n1$. Node $n4$ does not have a trusted route to $n1$ unless it receives the routing update via $n3$, which is listed within the trust set of $n1$ and can therefore be considered a next hop towards $n1$. The topology depicted at the bottom of phase 3 illustrates with arrows the resulting virtual topology for routing traffic towards $n1$. In this example, $n2$ is the only node that $n1$ does not trust. Consequently, its traffic is directed around $n2$. Given the assumption that $n3$ trusts all nodes, phase 3b, with blue arrows, illustrates the concurrent established topology for routing traffic towards node $n3$. Traffic from $n4$ to $n3$ will be routed via $n2$ (due to potentially strong involved links $n4{-}n2{-}n1$). However, traffic towards $n1$ remains routed via $n3$ (being the only trusted path option towards $n1$).

5.5. *Summary of Properties*

The fundamental properties of this approach are summarized as follows:

- Descriptions can be short while securely specifying large amounts of data (e.g., a single 224-bit SHA2 hash compared to a 2048-bit RSA signature or a trust set listing the global IDs of hundreds of nodes).
- Referenced data that does not change over a description update or that is used by several nodes (e.g., defining equal trust sets) does not need to be re-propagated since their reference (hash) would not change and can be reused.
- A significant processing advantage is achieved using a lightweight DH-HMAC-based verification mechanism for

the continuous and frequent packet signing and validation operations (typically occurring several times per second). In contrast, a reasonably strong and asymmetric RSA key can be used for signing the rather seldom-propagated description updates (typically once per hour) and thereby allowing a long-term protection of node IDs and descriptions.

- The mechanisms provide an infrastructure that allows (without requiring any pre-deployed central registry) the network-wide, dynamic, and secure (in terms of authenticity and integrity) distribution of public node keys that are further leveraged for integrity verification of node IDs, descriptions, link-discovery and routing-update messages between neighbouring nodes and ensures the propagation of routing updates and the establishment of forwarding path only along nodes trusted by the originator of these updates.

- With the ability to delegate the maintenance of trust sets to individually selected super-trusted nodes, it is possible to share and distribute the work of trustability assessment while conserving decentrality and self-determination and ensuring each actual node owner as the final root of trust that can withdraw or extend previously defined delegations at any time.

6. Specification and Correctness

6.1. Definitions

- A *node description* $D_{X,Q}$ denotes a particular version $Q \in \mathbb{N}^+$ of a statements' summary created and signed by the node with the identity V_X. A valid description $D_{X,Q} = \{D^c_{X,Q}, S_{X,D^c_{X,Q}}\}$ contains unsigned description components $D^c_{X,Q}$ and a signature $S_{X,d} = s(K^{priv}_X, d)$ with data $d = D^c_{X,Q}$ that, created with the private key K^{priv}_X of V_X. $D^c_{X,Q}$, covers non-ambiguously definitions for V_X, K_X, $Q = Q^d_X$, A_X, $\mathbb{V}^t_X$, $\hat{M}_X$, and $f_X(M, L)$ as defined in the following. Nodes are supposed to learn the latest description (with highest description sequence number Q^d) of all other nodes. Therefore each node creates and broadcasts a description for itself which is then flooded by receiving nodes across the network. All nodes are also supposed to resend any earlier flooded description when demanded by a neighbouring node so that new nodes can learn the descriptions of already existing nodes. Further background in section 5.1.

- The *primary IPv6 address* $A_X = a(V_X)$ of a node is a Cryptographically Generated Address (CGA) based on a truncated version of the nodes identity V_X. The existence of a sufficiently secure CGA function $a(V)$ (as described in section 5.3.1) that provides proof of identity and ownership is assumed for this work.

- A *link-metric function* $l(E_{B,A}) = L_{B,A}$ with $L \in \mathbb{R}^+$ is a heuristic function that quantifies the quality of a given link as a positive real value (including zero). For any $L_{B,A} \geq L_{Z,Y}$ it is defined that the link quality of link $E_{B,A}$ is better than (equal to) that of $E_{Z,Y}$. The existence of a sufficient link-metric function is assumed for this work. A simple but sufficient link-metric function for letting V_A assess the quality of link $E_{B,A}$ is described in [38]

- The overall *topology* of a network is given by the graph $G = g(\mathbb{V}, \mathbb{E})$ representing all nodes $V \in \mathbb{V}$ as vertices and all links $E \in \mathbb{E}$ as edges if $l(E) > 0$.

- A *path-quality value* $M_{T,S}$ and *path-metric function* $m(V_T, V_s, f_T, \hat{M}_T)$ with $V_T \equiv V_0, V_S \equiv V_s, s \geq 1$ is defined as:

  $M_{T,S} = M_{0,s} =$

  $$m(V_T, V_s, f_T, \hat{M}_T) = \begin{cases} \text{if } s == 1 : f_T(\hat{M}_T, L_{T,s-1,s}), \text{ else} \\ \text{if } f_T(m(V_T, V_{s-1}), L_{T,s-1,s}) == 0 : 0 \\ \text{else } : f_T(m(V_T, V_{s-1}), L_{T,s-1,s} \end{cases}$$

  $\hat{M}_T$ is a constant defined by V_T via $D_{T,Q}$ for the upper bound of any $M_{T,S}$. $f_T(M, L)$ is the customizer function specified via $D_{T,Q}$. It must ensure that

  $$0 \leq f(M, L) < M < \hat{M} \text{ for any } M \in \mathbb{R}^+ \, , L \in \mathbb{R}^+ \qquad (5)$$

  A function satisfying (5) would also ensure that:

  - Greater path quality values reflect better paths, thus for any path $P_{T,A} = \{V_T, ..., V_A\}$ with a better (equal) path quality than another path $P_{T,X} = \{V_T, ..., V_X\}$ that $M_{T,A} > (=) M_{T,X}$.
  - A sub path $P_{T,A} = \{V_T, ...V_A\}$ of another path $P_{T,X} = \{V_T, ..., V_A, ..., V_X\}$ with equal destination V_T and path components up to including V_A must lead a greater path metric $M_{T,A}$ than that of $M_{T,X}$. The existence of a sufficient path metric is assumed for this work.

- A *route* $R_{T,S} = \{V_T, V_{s-1}, V_s\}$ explicitly identifies the next hop V_{s-1} of a particular path $P_{T,S}$ from $V_s \equiv V_S$ via V_{s-1} and implicitly defines also all following nodes $V_{s-2}, ...V_1, V_0$ to $V_0 \equiv V_T$.

- The *set of all routes* from V_S to V_T with different next hops is limited by the number of neighbours of V_S and is given as $\mathbb{R}_{T,S}$.

  A *routing function* $r(V_T, V_a) = R_{T,A} = \{V_T, V_{a-1}, V_a\}$ implicitly also identifies all consecutive nodes by iterative resolving V_{a-2} from $r(V_T, V_{a-1}) = \{V_T, V_{a-2}, V_{a-1}\}$.

- The *trust set* $\mathbb{V}^t_X \subseteq \mathbb{V}$ of a node V_X lists the node identities (including itself) of its trusted nodes. E.g.: $\mathbb{V}^t_X = \{V_A, V_B, V_C, V_X\}$.

- A *trusting node* is a node (e.g. V_S) that defines a non-empty trust set $\mathbb{V}^t_S$ to secure itself against attacks from non-trusted nodes. Therefore, the cardinality of $\mathbb{V}^t_S$ must be greater or equal than one: $|\mathbb{V}^t_S| \geq 1$. Thus, if a particular node V_X does not trust any other node then it would define only itself as trustable: $\mathbb{V}^t_X = \{V_X\}$.

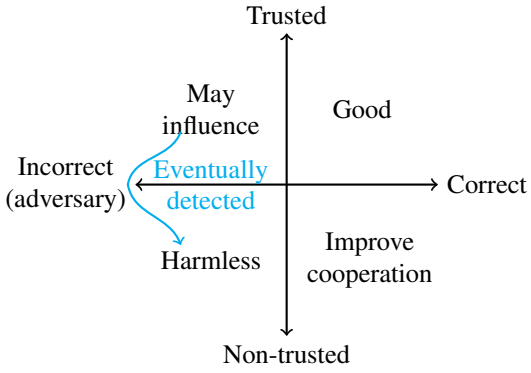


Figure 4: Cases in SEMTOR considering trust, correctness, and a detection mechanism

- A *trusted node* is a node (e.g. V_X) which is explicitly listed in the trust set $\mathbb{V}_A^t$ of a given node V_A. This is formalized as $V_X \in \mathbb{V}_A^t$. A *non-trusted node* V_Y is a node which is not listed in the trust set of a given other node V_A, formalized as $V_Y \notin \mathbb{V}_A^t$. Thus, trust is a tuple of the trusted node (V_X) and the *trusting node* (V_A). A particular node V_Z can be a trusted node of node V_A and at the same time be a non-trusted node of node V_B: $V_Z \in \mathbb{V}_A^t \bigwedge V_Z \notin \mathbb{V}_B^t$. Trust relations are not implicitly symmetric, e.g.: $V_Z \in \mathbb{V}_A^t \bigwedge V_A \notin \mathbb{V}_Z^t$. Direct trust relations are not transitive, e.g.: $V_C \in \mathbb{V}_B^t \bigwedge V_B \in \mathbb{V}_A^t \bigwedge V_C \notin \mathbb{V}_A^t$.
- A link $E_{B,A}$ is a *trusted link* of node V_X if the two neighbouring nodes V_A and V_B are trusted nodes of V_X. The *trusted link set* $\mathbb{E}_X^t$ of node V_X is defined as

$$\mathbb{E}_X^t = \{E_{B,A} | E_{B,A} \in \mathbb{E}, V_A \in \mathbb{T}_X, V_B \in \mathbb{T}_X\} \quad (6)$$

- The *trusted topology* G_X^t of node V_X is a sub graph of $G(\mathbb{T}, \mathbb{L})$ consisting of vertices $V \in \mathbb{T}_X^t$ and edges $E \in \mathbb{E}_X^t$ with $G_X^t = G(\mathbb{T}_X^t, \mathbb{E}_X^t)$.
- A *trusted path* is a connected end-to-end path between a given source-destination-node tuple that consists only of trusted nodes and links of the (trusting) destination node. It represents one particular combination of consecutive links from the trusted topology of the destination node. A *trusted route* denotes, according to a given path metric, the currently best existing trusted path.

6.2. Desired Properties

For any given node V_T two orthogonal properties of nodes are considered: correct ($V^c \in \mathbb{V}_T^c$) versus adverse ($\overline{V^c} \notin \mathbb{V}_T^c$), and trusted nodes ($V^t \in \mathbb{V}_T^t$) versus non-trusted nodes ($\overline{V^t} \notin \mathbb{V}_T^t$). This leads to four possible combinations as illustrated by the four sectors in Figure 4: Non-trusted correct nodes, trusted correct nodes, non-trusted adversaries, and trusted adversaries.

For any given (T, S)-path tuple and network graph G one of the following topological constellations can be considered:

a) Incorrect path: S is isolated from T either because a continuous (T, S)-path in G does not exist at all or because not a single consecutive path exists that consists only of T-correct nodes: $\mathbb{P}_{T,S}^c = \varnothing$

b) Correct path: At least one (T, S)-path of T-correct nodes exists, that may (or may not) consist of T-trusted nodes, but some T-trusted nodes of G may be T-adverse (incorrect) nodes: $\mathbb{P}_{T,S}^c \neq \varnothing \wedge \mathbb{V}_T^t \setminus \mathbb{V}_T^c \neq \varnothing$

c) Correct trust and path: At least one (T, S)-path of only T-correct nodes exists and T-trusted adversaries do not exist but a (T, S)-path of only T-trusted nodes may not exist: $\mathbb{P}_{T,S}^c \neq \varnothing \wedge \mathbb{V}_T^t \setminus \mathbb{V}_T^c = \varnothing \wedge \mathbb{P}_{T,S}^t = \varnothing$

d) Correctly trusted nodes and path: At least one (T, S)-path of only T-trusted and correct nodes exists and T-trusted adversaries do not exist: $\mathbb{P}_{T,S}^c \neq \varnothing \wedge \mathbb{V}_T^t \setminus \mathbb{V}_T^c = \varnothing \wedge \mathbb{P}_{T,S}^c \neq \varnothing$

We are interested in the correct and continuous provisioning of *awareness* (node and node state discovery) and *reachability* (path and route discovery and packet delivery) services.

These services can be characterized by a set of properties [90, 56] that can be differentiated between safety and liveness properties [91, 92].

Safety properties state the guarantee that bad things do not happen:

- *Information authenticity and integrity* are mandatory properties for an awareness service that, in our case, shall propagate node descriptions and heartbeats. We define this property as *update-safety* which guarantees that if a $Q_{T,S}$-sequenced update (supposedly created by V_T) can be verified by a receiving node V_S as correct then it must have been sent earlier by the supposed originating node V_T as $Q_{T,T}$ and has propagated unchanged to V_S. Because every $Q_{T,T,\tau_n} \in \mathbb{N}^+$ sent by V_T at time τ_n must be greater or equal to any previously sent $Q_{T,T,\tau_{n-1}}, \tau_n > \tau_{n-1}$ this guarantee is expressed as: $Q_{T,T,\tau} \geq Q_{T,S,\tau}$. This property shall apply to description and heartbeat updates, sequenced with Q^d and Q^h respectively.
- *Path-adversary-freedom* is an important property for guaranteeing the reachability of V_T from V_S along a path. A path $P_{T,S}$ is adversary-free if it has no adversary nodes.
- *Adversary-resilience:* Routing is adversary-resilient if its behaviour under the presence of adversary nodes is equivalent to that of a network that has no adversary nodes. This means that adversary nodes, if present, do not have any noticeable effect. An update-safe awareness service and a delivery service that routes and forwards data packets only along adversary free paths can be considered adversary resilient.
- *Loop-freedom:* A path is loop-free if it has no repetition of nodes.
- *Consistency:* packets only traverse loop- and adversary-free paths between source and destination. We define this property as *route-safety* for any established route $R_{T,S}$ that

consists only of non-repetitive nodes that behave correctly (non-adversary) regarding $V_T : R_{T,S} \subseteq \mathbb{V}^t_T$.

There are also liveness properties which state that good things [eventually] happen:

- *Update freshness:* An update is fresh with respect to an interval I and an incrementally increasing update-version Q (update-sequence number) if the last received update $Q_{T,S,\tau}$ from V_T by V_S at time τ is greater or equal than any update $Q_{T,T,\tau-I}$ sent by V_T since I. We define *update-liveness* as the property that guarantees update freshness so that: $Q_{T,S,\tau} \geq Q_{T,T,\tau-I}$.

- *Route freshness:* A route $R_{T,S,\tau}$ from V_S to V_T at time τ is fresh with respect to an interval $I = (\tau - I, \tau)$ and a given set of existing paths $\mathbb{P}_{T,S,I}$ if all implicitly via $R_{T,S,\tau}$ defined consecutive hops and respective links are up during I. We define *route-liveness* as the property that guarantees in addition to route freshness also consistency so that: $R_{T,S,\tau} \in \mathbb{P}^c_{T,S,I}$.

- *Route accuracy:* A route $R_{T,S}$ is accurate if it is consistent and fresh and, with respect to a path-metric function $m_T()$ defined via $D_{T,Q}$, if the path-metric value calculated and re-propagated by V_S conforms with the actual link-metrics $L_{B,A}$ between all successive path nodes $V_B, V_A \in R_{T,S}$.

- *Responsiveness:* Depending on the logic of the routing-update process, updates received by a node are validated (which, if necessary, may include the immediate request and resolution of depending information) and applied to its forwarding table and propagated to other routers, including those that potentially depend upon the outcome of the update. This affects how quickly the network reacts to changes and therefore it qualifies the timeliness of freshness.

  It is assumed that the transmission delay υ (from data-link layer assumptions in section 4.2) of up links and the responsiveness ρ of correct nodes for processing received updates and responding to resolution requests from neighbours is much smaller than I: $\upsilon + \tau \ll I$

- *Delivery:* data packets get eventually delivered to its destination though a path, with statistical guarantees as channels are usually lossy, typically in a small percentage. This property depends on the liveness of updates (freshness) and routes (freshness and accuracy).

- *Efficiency:* The expected resource overhead cost for each node amortized over all usage. Resources can be processing (CPU, memory in nodes) and communication (traffic in links), and the overhead cost, the additional resource consumption due to the protocol translated into a performance penalty (cost).

Safety and liveness properties about updates and routes are the basis to ensure overall properties: the (safety) consistency and resilience to adversaries; and the (liveness) responsiveness, delivery and efficiency of routing.

Table 3 summarizes the desired safety and liveness properties and identifies the previously described topology constellations a) - d) for which these properties shall be satisfied. Here the $*$-symbol is used to indicate that no assumptions are made on the requirement given in the respective header row (thus, may be true or false) and the ✓-symbol is used to indicate that a corresponding property can be satisfied. None of the four constellations makes assumptions on the presence of non-trusted adversaries.

It should be noted that the assumptions and objectives outlined in section 3.1 are actually reflected by the conditions and properties listed for constellation d) while constellation a) to c) list properties that shall be provided under even less restrictive conditions.

Regarding the security objective: Any group of nodes (e.g. subeset $\mathbb{V}_X \subseteq \mathbb{V}_Y$) that is trusting (so $\forall V_T \in \mathbb{V}_X : \mathbb{V}^t_T = \mathbb{V}_X$), willing to cooperate and support each other (correct, so $\mathbb{V}_X \subseteq \mathbb{V}^c_T$ with $\mathbb{V}^t_T = \mathbb{V}_X \Rightarrow \mathbb{V}^t_T \setminus \mathbb{V}^c_T = \varnothing$, so not trusted adversaries exist), and contiguous (connected, so with $\mathbb{V}^t_T = \mathbb{V}_X \Rightarrow \forall V_T, V_S \in \mathbb{V}_X : \mathbb{P}^t_{T,S} \neq \varnothing$) cannot be inhibited by an external $V_Z \notin \mathbb{V}_X$ on the objective of routing packets to each other (which is satisfied by the given route-safety and -liveness properties).

6.3. Proving Correctness

In this section we analyze the safety and liveness properties defined in section 6.2 for updates and routes as lemmas and theorems that we prove with formal reasoning.

Lemma 1. *Descriptions discovered by SEMTOR are authentic and can be differentiated between older and newer versions, independent of the presence of trusted or non-trusted adversaries.*

Proof: A description $D_{X,Q}$ received by a SEMTOR node is only accepted as valid if it contains the components $D^c_{X,Q} = \{V_X, K^{pub}_X, Q^d_X, A_X, \mathbb{V}^t_X, \overline{M}_X, f_X\}$ and a signature $S_{X,D^c_{X,Q}}$ for which V_X equals $v(K^{pub}_X)$ and $\overline{s}(K^{pub}_X, S, D^c_{X,Q})$ is *true* and Q^d_X is greater than any earlier received and accepted description from V_X. Description integrity is proven because, given Assumption 1, a $\overline{s}(K^{pub}_X, S, D^c_{X,Q})$-correct signature cannot be created by any other node than V_X, and therefore not by any adverse node. Identity and authenticity is implicitly proven because of the matching signature and V_X matching $v(K^{pub}_X)$. Exclusive ownership of A_X is proven based on Cryptographically Generated Address (CGA) respective rules. Possibly earlier received descriptions from V_X can be reliably identified as outdated if contained $Q^d_{earlier} < Q^d_{current}$. □

Lemma 2. *Heartbeats discovered by SEMTOR are authentic, can be related to a particular description of the originating node, and can be differentiated between older and newer versions, independent of the presence of trusted or non-trusted adversaries.*

Proof: A heartbeat $Q^b_{X,Q}$ received by a SEMTOR node is only accepted as valid and identified as originated by V_X if, among the latest accepted descriptions known from any node, one can be found that contains an AHC anchor $Q^a_{X,Q}$, which

Table 3: Topology constellations and respective protocol properties.

Constellation	correct path exists: $\mathbb{P}^c_{T,S} \neq \varnothing$	no trusted adversaries: $\mathbb{V}^t_T \setminus \mathbb{V}^c_T = \varnothing$	trusted path exists: $\mathbb{P}^t_{T,S} \neq \varnothing$	no adversaries: $\overline{\mathbb{P}^c_T} \neq \varnothing$	Update-Safety (au-thenticity): $Q_{T,T,\tau} \geq Q_{T,S,\tau}$	Update-Liveness (freshness): $Q_{T,S,\tau} \geq Q_{T,T,\tau-I}$	Route-Safety (consistency): $R_{T,S,Q^h} \subseteq \mathbb{V}^c_{T,Q^d}$	Route-Liveness (freshness): $R_{T,S,\tau} \in \mathbb{P}^c_{T,S,(\tau-I..\tau)}$
a)	*	*	*	*	✓	-	-	-
b)	true	*	*	*	✓	✓	-	-
c)	true	true	*	*	✓	✓	✓($R_{T,S} = \varnothing$)	-
d)	true	true	true	*	✓	✓	✓($R_{T,S} \neq \varnothing$)	✓

yields a positive heartbeat-sequence number calculated as $Q^h_X = h(Q^a_{X,Q}, Q^b_X, V_X, Q^d_X)$, and one of the following conditions is true: Either Q^h_X is equal to or greater than any previously accepted Q^h_X value from V_X that was based on the same $Q^a_{X,Q}$ and $Q = Q^d_X$, or the last accepted heartbeat sequence from V_X was based on an outdated description from V_X that has since then be replaced by the receiving node with an accepted updated description. The authenticity belonging to a particular description and the differentiability (in older and newer versions) of heartbeats is proven because of the cryptographic properties of AHCs (see also section 5.3.3 and Assumption 1). This assumption also covers V_X and Q^d_X so that only the originating node V_X that knows the secret AHC root can create heartbeats that yield Q^h_X values that are acceptable as newer (greater) than the heartbeat with the greatest corresponding Q^h_X value ever sent before by V_X. □

Theorem 1. *SEMTOR guarantees update-safety: the authenticity, integrity, and differentiability of discovered node descriptions and heartbeats.*

Proof: From Lemmas 1 and 2. □

Theorem 2. *SEMTOR guarantees update-liveness: Descriptions and heartbeats discovered by SEMTOR in the presence of correct paths are fresh so that any description $D_{X,Q}$ or heartbeat $Q^b_{X,Q}$ originated at time t_1 by V_X with $Q = Q^d_X$ is discovered by any V_Y before time $t_2 = t_1 + I$ if a consecutive path $P^c_{X,Y}$ of V_X-correct nodes exists between V_X and V_Y that is up during the entire interval $I = (t_1, t_2)$.*

Proof: Let $I^h = (t_2 - t_1)$ be the interval duration with which each node V_X broadcasts a new heartbeat (update) $Q^b_{X,Q}$ yielding a heartbeat-sequence number $Q^h_{X,Q}$ that is one greater than the previously broadcasted heartbeat sequence of V_X. If an up-link between the $Q^b_{X,Q}$-broadcasting node and a correct neighbouring node V_N of the broadcasting node exists during the broadcast of $Q^b_{X,Q}$ then V_N receives $Q^b_{X,Q}$ and one of the following three cases occur:

(i) V_N already knows and accepted $D_{X,Q}$ (see Lemma 1), on which $Q^b_{X,Q}$ is based and has also already accepted $Q^b_{X,Q}$ (see Lemma 2) due to an earlier reception from a neighbouring node of V_N. Then, V_N only responds to description requests received from its neighbouring nodes that demand a description known by V_N.

(ii) V_N already knows and accepted $D_{X,Q}$, on which $Q^b_{X,Q}$ is based and accepts $Q^b_{X,Q}$ as a new heartbeat update. Then, $Q^b_{X,Q}$ is re-broadcasted by V_N.

(iii) V_N does not know $D_{X,Q}$, on which $Q^b_{X,Q}$ is based. Then, the acceptance processing of $Q^b_{X,Q}$ is postponed by V_N for at most I^h, $Q^b_{X,Q}$ is cached for possible later processing for the duration of I^h, and a request is broadcasted to all neighbouring nodes of V_N demanding a description that allows V_N to accept $Q^b_{X,Q}$. Within I^h, if a description $D_{X,Q}$ is received by V_N that is acceptable and yet unknown by V_N, then V_N accepts $D_{X,Q}$ and searches its cached heartbeats for one that can be accepted based on $D_{X,Q}$. If this is the case, then V_N continues as described for case (ii).

As a consequence of this mechanism, and given that the transmission delay of up links and the responsiveness of correct nodes for processing received updates and responding to requests is much smaller than I, then $Q^b_{X,Q}$ and $D_{X,Q}$ will be received, accepted, and re-broadcasted by all correct neighbouring V_N nodes of the originating node V_X to which a link exists during I. In addition, they will be further propagated to all correct neighbouring nodes of V_N, which, if behaving correctly, will further propagate these until they are accepted by all nodes V_Y to which a consecutive path $P^c_{X,Y}$ of V_X-correct nodes exists that is up during the entire interval I. □

Theorem 3. *SEMTOR guarantees route-safety: Routes discovered by SEMTOR in the presence of non-trusted adversaries (implying that trusted adversaries do not exist) are loop free and adversary free.*

Proof: Any destination node V_T broadcasts a new and signed routing update message $U_{T,Q^b_T,T} = \{Q^b_T, M_{T,T}\}$ with each new heartbeat update Q^b_T and with $M_{T,T} = \hat{M}_T$.

Any route R_{T,S,Q^b_T} to destination V_T, propagated via routing update messages $U_{T,Q^b_T,N} = \{Q^b_T, M_{T,N}\}$, and broadcasted by node V_N (V_N may be equal to V_T) is only accepted by a correct receiving node V_S if all the following conditions are satisfied.

(i) The current description D_{N,Q^d_N} of V_N is known and accepted (from proof of Lemma 1).

(ii) The transmission signature that covers $U_{T,Q^b_T,N}$ proves the authenticity and integrity of the update as having been created and broadcasted by V_N (note also Assumptions 2 and 1 and related mechanism details described in Sections 5.2 and 5.3.3).

(iii) A description D_{T,Q^d_T} is known that allows accepting Q^b_T from $U_{T,Q^b_T,N}$ as a valid heartbeat update from V_T (from proof of Lemma 2)

(iv) Q^b_T is either new or Q^b_T is equal to the newest heartbeat update from V_T and $M_{T,N}$ from $U_{T,Q^b_T,N}$ is the greatest path-metric value ever accepted (via any neighbour) for V_T. Q^b_T.

(v) $M_{T,N}$ from $U_{T,Q_T^b,N}$ is greater than zero.

(vi) V_N is listed as a trusted node in $\mathbb{V}_T^t$ from D_{T,Q_N^d}.

Only if a routing-update messages received by the correct node V_S is accepted as valid then V_S configures a forwarding route to V_T via V_N and propagates the route further by creating and broadcasting a new and V_S-signed update as: $U_{T,Q_T^b,S} = \{Q_T^b, M_{T,S}\}$ with $M_{T,S} < M_{T,N}$.

Condition (vi) is only true if $V_N \in \mathbb{V}_T^t$, which, given that $\mathbb{V}_T^t \subseteq \mathbb{V}_T^c$, implies that $V_N \in \mathbb{V}_T^c$. Therefore, routes are only accepted and configured if propagated along correct and therefore adversary free, paths.

Condition (iv) is only true if the contained heartbeat and/or path-metric value represents an update that is newer or better than any previously received updates by the processing node. Therefore neither the processing node, nor any other correctly processing node, could yet be part of a route established by any updates with the same heartbeat sequence. Therefore, all accepted routes are loop free. □

Theorem 4. *SEMTOR guarantees route liveness: Routes discovered by SEMTOR, in the presence of non-trusted adversaries (implying that trusted adversaries do not exist) and at least one correct and trusted path, are fresh.*

Proof: The proof of route-liveness is provided by an equivalent reasoning as given for Theorem 2.

As a consequence of the routing-update propagation mechanism from the proof of Theorem 3, and given that the transmission delay of up links and the responsiveness of correct nodes for processing received updates and responding to requests is much smaller than I, then any routing update U_T originated by V_T will be received, accepted, and further propagated by all correct neighbouring V_N nodes of V_T to which a link exists during I. Additionally, they will be further propagated to all correct neighbouring nodes of V_N which, if behaving correct, will further propagate these till they are accepted by all nodes V_S to which a consecutive path $P_{T,S}^{c,t}$ of V_T-correct and trusted nodes exists that is up during the entire interval I. □

7. Experimental Validation

7.1. Implementation and Validation Framework

In this section the key security and performance achievements of SEMTOR are validated experimentally. Therefore, the proposed SEMTOR mechanisms have been implemented by extending the BMX6 routing protocol which eventually became BMX7 (BMX6 + SEMTOR = BMX7). This BMX7 implementation has then been observed when exposed to key attack scenarios and challenging network environments.

The BMX6 [16] routing-protocol has been selected as a basis because its message structure and handling already provides functional support for node-descriptive profiles, SHA hashes as referencing identifier, and a "hash-based profile-propagation mechanism" to disseminate node descriptions. For cryptographic operations such as hashing, asymmetric-key generation, signing, and verification the MbedTLS [93] library has been used. Our SEMTOR extensions currently supports RSA512 to RSA4096 public-key authentication for node and description verification, SHA224-based hashes for node-, description-, and routing-update references (the latter based on anchored-hash chains), and DH-key exchange for MAC-based neighbour verification. The complete source code is publicly available via the SEMTOR/BMX7 git-repositorys [16, 94]. Protocol implementation details are summarised in Table 4.

Table 4: Protocol implementation details

Topic	Details and used default
Protocol Source code	SEMTOR/BMX7 branch, git revision ee2f1d86
Crypto Library	MbedTLS version 2.4.0
Node-identity and description referenciation	SHA224 (RFC4634) hashes
Primary key foundation	RSA2048 (RFC8017)
Asymmetric txKey foundation	RSA896 (RFC8017)
Symmetric txKey foundation	112-bit truncated SHA224-MAC based on DHM2048 (RFC3526) negotiated shared key
Description-update interval	36000 s
Heartbeat authentication	112-bit truncated, SHA224-based, anchored hash chain
Routing updates interval	6 s
Link-probing interval	0.8 s
Aggregation (TX) interval	0.8 s

The scenarios to which our implementation was exposed were realised with an emulated network consisting of Linux nodes and using MLC [95], a suite of LXC-based scripts,[3] that allow virtualization of hundreds of Debian systems inside a single host and configuration of virtual topologies, essentially by linking virtual or real node interfaces via a virtual bridge and using ebtables[4] and tc[5] to control packet loss and delay between nodes on layer two.

The host system was an Intel i5-3230M 2.60 GHz CPU with four CPU cores and 8 GB RAM, allowing more than 200 virtual systems and protocol instances to run in parallel.

Further emulation parameters and ranges used for the functionality validation in Section 7.2 and the performance measurements in Section 7.3 are given in Table 5. The selected probing ranges comprise network sizes and densities that are typical for deployments in clouds and core-graph zones of real CNs [3, 4, 5, 69, 96, 97, 98, 99, 100, 71, 73].

7.2. Validation of Functionality

To validate our design and implementation (regarding the ownership, security, openness, and decentralization objectives as outlined in Section 3.1), several representative network attacks have been set up and analyzed. section 7.2.1 first details the implementation of different attack vectors and section 7.2.2 evaluates their effect on a node that naively (and falsely) trusts

[3] Linux containers `http://lxc.sourceforge.net/`.

[4] Linux ebtables, `http://ebtables.sourceforge.net`

[5] Linux traffic control: tc, `http://tldp.org/HOWTO/Traffic-Control-HOWTO/`.

Table 5: Emulation defaults and ranges

Parameter	Default [range]	Figures
Host hardware	Intel i5-3230M, 2.6 GHz, 4 cores, 8 GB RAM	
Host system	Linux, Debian	
Emulation system	MLC git rev. d63f726c	
Network virtualization	LXC, ebtables, qdisc, tc, brctl	
Emulated systems	Linux, Debian	
Measurement tools	tcpdump, top, ping, iperf	
Network structure (topology)	Grid 10x10 [10x2..20]	11
Link loss	0% loss [0%..60%]	9
Node interfaces	1	
Network size (# of nodes)	100 [10..200]	12a,12c
Density (# of links per node)	4 [4..20]	12d, 12f
Primary key strength	RSA2048 [512..4096]	
TxKey method	RSA512..1536 vs. DHM2048	[25], 12
Own description-upd. freq.	1/10h [0..0.5 Hz]	12b
Others description-upd. freq.	100/10h [0..5 Hz]	12e

in the correct behaviour of all nodes (including existing malicious and attacking nodes), which corresponds with constellation b) from Table 3.

Section 7.2.3 then validates our SEMTOR implementation and quantifies its performance to recover, when attacking nodes have been correctly identified (as given by constellation d) from Table 3), from the most powerful combination of previously introduced attacks. Eventually, Section 7.2.4 illustrates further protocol achievements for supporting openness, decentralization, and cooperativeness.

7.2.1. Attack vectors

To experimentally validate the resistance of SEMTOR when exposed to attacks, an implementation is needed that supports the execution of a corresponding attack. In this section, we describe and validate the implemented attacks by observing their effect on a node that is reachable via paths consisting of correct and adverse nodes and that considers all, thus including the attacking nodes, as trusted. That corresponds to the most adverse topology constellation b) from Section 6.2.

For the below detailed attacks, an attacking node behaves, unless explicitly noted, like a correct node (only the deviations from the correct behaviour are described). Functions implementing these attack-specific deviations from the correct behaviour are provided via a so-called 'evil' plugin, extending the default protocol implementation with malicious capabilities. The plugin allows (i) defining a set of attacked nodes based on a list of node IDs and (ii) defining the combination of attacks performed on these nodes.

We have considered attacks to manipulate or selectively suppress routing information about the identity and description of the nodes, the routing updates, and route announcements.

An **identity hijacking attack (IHA)** would be given by any successful attempt to propagate a manufactured or modified description of an attacked node, for example, by replacing the trust set or transmission key of the attacked node's original description. One variance of such description manufacturing has been implemented by transmitting descriptions of attacked node IDs with an incremented description SQN.

Tests confirmed that manipulated descriptions are discarded immediately upon reception by any correctly behaving node because of the description signature mismatch with the node ID's public key (remember that a node ID is defined as the SHA hash of its public key). To successfully execute this attack, an adversary would need to manufacture a matching signature, which is possible only by knowing the node ID's private key. Therefore, any IHA attack must fail as long as private keys are not compromised.

A **route (or primary IP) hijacking attack (RHA)** is implemented by letting an attacking node announce the primary (CGA) IP addresses of the attacked nodes via its own description and propagating this and the description-referencing routing updates to the network via its neighbouring nodes.

Tests confirmed that such composed descriptions and description-referencing routing updates are discarded immediately upon reception by any correctly behaving node because the contained CGA announcements do not match the description node ID.

A **heartbeat (or routing-update) manufacturing attack (HMA)** is implemented by transmitting all internally scheduled routing-update messages that reference the description of an attacked node ID with a hash-chain value replaced with random bits.

Tests confirmed that such way manipulated and received routing-update messages are not further processed or propagated because the contained hash-chain value does not match the hash-chain anchor of any known description.

A **description dropping attack (DDA)** is achieved by discarding all internally scheduled description transmissions that contain a node ID matching any of the attacked node IDs.

A **routing-update dropping attack (UDA)** is achieved by discarding all internally scheduled routing message transmissions that reference the description of an attacked node ID.

A **routing-update metric manipulation attack (MMA)** is achieved by transmitting all internally scheduled update messages that reference the description of an attacked node ID with a metric value modified to the maximum (best), suggesting to the update-receiving nodes a better path than actually exist.

In contrast to all former attacks, which can be classified as control-plane attacks, a **route or traffic dropping attack (RDA)** represents a typical data-plane attack. This is achieved by configuring a route towards the primary (CGA) address of an attacked node into a virtual '/dev/null' interface that drops all IP packets routed into it (instead of via the next hop of its collectively established path).

From the above attacks, three categories can be identified: (i) Attacks that every correctly behaving SEMTOR node always secures against: The first three (IHA, RHA, and HMA) attacks are of this kind. As they are prevented by design no further exploration of such attacks is needed.

(ii) Attacks that can be considered relatively harmless because the behaviour of the attackers is similar to the case in which the attackers do not exist: The DDA and UDA attacks are of this kind. However, it should be noted that the combination of the DDA attack with the RDA or MMA attack does make a difference that can be lucrative from an attackers per-

spective because not supporting the propagation of an attacked node's description updates may increase the latency with which these are eventually received by all other nodes. Especially, if such an update is received late, then it has the attacking node removed from its previous trust set.

(iii) Attacks that the protocol only secures against if performed by non-trusted nodes: These are given by the latter MMA and RDA attacks, which are analyzed (in combination with the DDA) in more detail in the following sections. To neutralize these attacks, it is necessary to identify the incorrect nodes and mark them as non-trusted. This identification could be achieved either by improving the assessment of trustability on a social layer among CN members, or it could be automated with a monitoring and distributed detection protocol. For the analysis of the re-coverage performance of SEMTOR in section 7.2.3, we take the availability of eventually correctly assessed trustability as given and focus on the performance of the protocol to enforce such updated assessments.

7.2.2. Network Attack Scenarios

To further explore the power of different attacks we first assume a network scenario without non-trusted nodes (thus, all nodes are trusted, whether correct or not) and therefore also without counter-measures. The used network scenario, as outlined in Figure 5, allows placing one or more adversaries (intermediate *B*-nodes) on differently "good" strategic positions for attacking the routing process between an attacked destination node *DA* and a given source node S. As an example, by activating, in addition to the permanent links (illustrated as solid lines), the optional $SB4$ link between node S and $B4$, any communication between nodes S and *DA* would fully rely on the only possible end-to-end path $S-B4-B3-B2-DA$, allowing any intermediate node $B2$, $B3$, or $B4$ to easily disrupt this communication, simply by dropping any packet (doing an RDA) sent via them to *DA*. However, if the optional $SA2$ link is also activated, then an alternative end-to-end path from S, via $A2$ to *DA* emerges which is also shorter than the previous path. In this case, the position of node $B3$ is less advantageous for attacking the communication between S and *DA* and additional attack measures (such as MMA) would be needed to convince node S to send packets to *DA* via the longer (RDA performing) $S-B4-B3-B2-DA$ path.

Further measurements are executed as follows. At time T_0, all nodes start bootstrapping a previously configured network topology and establishing end-to-end paths to all other nodes which takes less than 40 seconds to complete. At a randomly selected time T_x between 50 and 55 seconds, node S starts sending ICMPv6 packets with an interval of 10 ms to both destination nodes in parallel. At the same time all intermediate *B*- nodes start performing combined DDA and RDA attacks on destination node *DA*.

The graphs in Figures 6 and 7 illustrate the effect of the applied attacks in terms of success and latency as the duration (averaged over 5 measurement probes) between the attack activation at T_x and its successful completion. The completion is measured from two perspectives. The *path-collapse time* represents the duration between T_x and the last successfully delivered packet to the corresponding destination node. If none of the packets sent by node S since T_x were ever delivered, then this duration is assumed to be zero, meaning that the attack succeeded immediately.

On the other hand, if packets continued to be delivered for more than 40 seconds since T_x, then this attack is considered failed, the evaluated collapse time is considered infinite, and no result is printed. The *path-entrapment time* represents the duration between T_x and the first packet forwarded to any of the intermediate attacking nodes (which, due to their RDA configuration, drop the packets instead of further forwarding them). If no packets towards an attacked destination node could be received by any of the attacking intermediate nodes, then this duration is also considered infinite and no result is printed.

The *DA path-collapse time* and *DA path-entrapment time* lines in Figure 6 show, depending on the length of the attacking *B* path, the effect of an RDA attack (performed by all intermediate B-path nodes) on packets sent (by S) to destination node *DA*. The length of the alternative *A* path between S and *DA* was configured to three, six, or nine hops, respectively, for Figure 6a, 6b, or 6c. The results show that attack is only possible from nodes that are part of shortest path between a given source and an attacked destination node. If the attack is possible, then it succeeds immediately. However, whenever an existing non-attacked path is shorter (or generally better) than the attacking path, such as in 6a for a *B*-path length of three or more hops, in 6b for a *B*-path length of six or more hops, or in 6c for a *B*-path length of nine hops, then at least one of 5 observed attacks failed.

As a double check, the *DB path-collapse time* and *DB path-entrapment time* were measured to show the effect of all intermediate *A* nodes attacking the routing towards destination node *DB* (from the same source node S) with converse results. For the sake of brevity, these measurements are not shown here.

In summary, for DDA and RDA attacks, the most vulnerable scenario for the attacked destination nodes is given if the attacking nodes are in the advantageous position of a shorter or better path.

A different effect can be seen if an attacking node combines the DDA and RDA with the MMA attack as illustrated in Figure 7. For these measurements, only a single *B* node attacked the packet delivery to *DA*. For Figure 7a, apart from only a single attacking *B* node, the network scenario is equal to that of Figure 6c. The combined attack on *DA* also succeeds with a nine-hop *B* path that has the same length as the non-attacking *A* path. This is because the attacking node manipulates the distance-vector metric of the routing updates of *DA* to the best possible value so that other nodes, receiving these updates, perceive the path via the attacking node with a better path metric than it actually has. This effect proves even more drastic in the scenarios used for Figures 7b and 7c. In the former, *DA* is successfully attacked by $B8$ for a *B*-path length between five and nine hops. It should be noted that $B8$, in *B*-path lengths of five to seven hops, is not on any reasonable path between S and *DA*. In the case of the five-hop *B* path, it is actually three hops apart from the shortest and correctly behaving path given by S-$B5$-$B4$-$B3$-$B2$-*DA* and still succeeds in dropping routes redirected

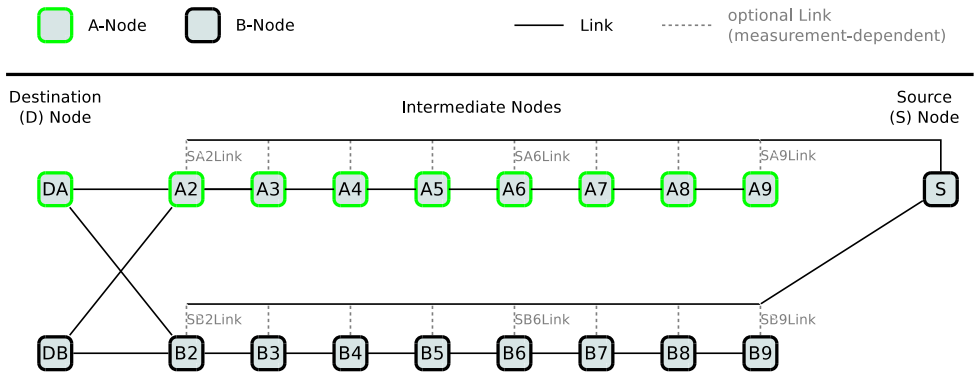
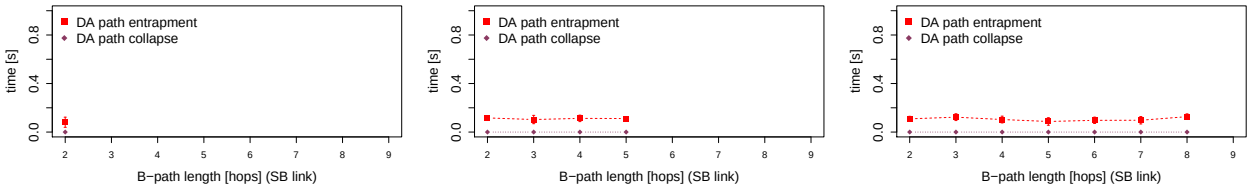
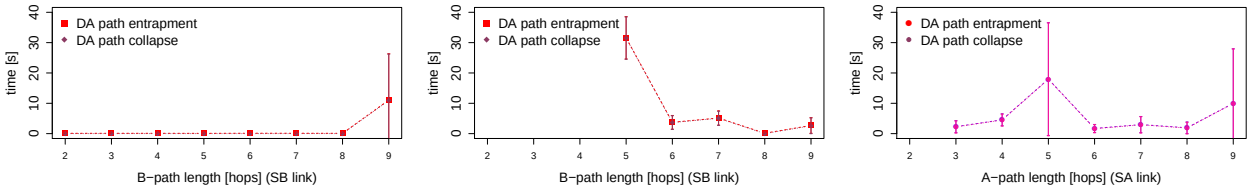


Figure 5: Base-network topology for validating attack vectors



(a) Attack impact with 3 hops A path (b) Attack impact with 6 hops A path (c) Attack impact with 9 hops A path

Figure 6: Performance of route (and packet) dropping attack (RDA) performed by *B* nodes on *DA* node depending on *A* and *B*-path length



(a) Attack impact of B2 (9-hops *A*-path) (b) Attack impact of B8 (9-hops *A*-path) (c) Attack impact of B8 (9-hops *B*-path)

Figure 7: Performance of DDA&RDA&MMA-combined attack on *DA* node depending on *A* or *B* path length and adversary position

towards itself. That is because *B*5, which resides on the shortest path, is closer to the MMA-attacking node *B*8 than to the destination node *DA* and is affected by the promising updates manipulated by *B*8. The same effect causes the successful attacks shown in Figure 7c where the actual shortest path is given by the varied *A*-path length, while the *B* path, where the attacking *B*8 node resides, has been fixed to nine hops. Still, *B* succeeds to attack *DA* even from a path that is up to six hops longer than the alternative *A* path.

7.2.3. Recovery Functionality and Performance

In this section, we evaluate our SEMTOR implementation regarding its performance to recover, given that attacking nodes have been identified, from the most critical combination of previously introduced attacks, by applying corrected trust sets that exclude all adverse nodes.[6]

This attack combination is given if the adverse nodes are placed in the advantageous position of a shorter path towards the attacked destination node, are as close as possible to the source node, and perform a combined DDA, RDA, and MMA attack. Such a scenario is given for the transmission from *S* to *DA* when attacked from *B*2 in the topology illustrated in Figure 8. The topology extends that of Figure 5 with a third group and path of neutral (correctly behaving) *C* nodes, which, in its default configuration, corresponds (apart from the new *C* nodes) with the scenario used for Figure 7a.

First, we deviate from this default configuration by varying the path length of correct and non-correct (attacking) *A* and *B* nodes. Later, the effect of an enabled neutral path length, as given by the intermediate *C* nodes, is discussed in more detail in section 7.2.4. To capture the recovery and convergence performance of our implementation at the presence of lossy and fading links, which must be expected for any realistic Wireless Mesh Network (WMN) deployment, we also vary the broadcast packet loss of configured links between 0% (corresponding to perfect links) and up to 60%, using the Linux traffic control

[6]Recovery performance could also be interpreted as the duration it takes to effectively exclude a set of identified adversaries from the routing towards a given destination node.

command mentioned earlier. However, the artificial loss is only applied to broadcast transmissions that the SEMTOR protocol uses for all its protocol data. All unicast transmissions along all configured links are kept as loss free. This allows us to trace the routing decisions of the protocol by capturing the path of forwarded unicast packets, such as the ICMPv6 ping request, and reply packets used for probing end-to-end packet delivery between a given source-destination pair.

All B nodes perform combined DDA, RDA, and MMA attacks to disrupt packet delivery to DA. In parallel, all A nodes perform the same combination of attacks to disrupt packet delivery to DB. Moreover, C nodes perform no attacks and are not attacked by any other node. The C nodes trust all other nodes but are not trusted by any A or B nodes. The S node is used as source node that sends packets to the destination nodes DA, DB, and DC in parallel with the same parametrization as given in Sections 7.2.1 and 7.2.2.

We measure path entrapment in the same way as performed for Figures 6 and 7. However, once all attacked destination routes are entrapped, we keep on running the emulation, wait a random period between 40 and 60 seconds to let protocol instances adapt to the given configuration, and then update the trust sets of DA and DB so that DA does not trust B nodes anymore (e.g., $V^t_{DA} = \{V_{DA}, V_{A2}, V_{A3}, V_{A4}, V_{A5}, V_{A6}, V_{A7}, V_{A8}, V_{A9}\}$) and DB does not trust A nodes anymore (e.g., $V^t_{DB} = \{V_{DB}, V_{B2}, V_{B3}, V_{B4}, V_{B5}, V_{B6}, V_{B7}, V_{B8}, V_{B9}\}$). Note that neither DA nor DB needs to trust S because, for this scenario, S is not needed to relay packets from S to DA or DB (S creates and sends packets but does not relay them). Furthermore, note that the trust set V^t_S of S is irrelevant for the delivery of packets sent by S. Such trust updates actually reflect a constellation switch from Table 3 from b) to d). We then measure the elapsed time from the moment of the updated trust sets until the last packet from S to DA was routed via any B node and until the last packet from S to DB was routed via any A node. Therefore, tcpdump captures from the master switch, that covers all existing links, were searched for respective packets from S to DA, DB, and DC that contained a MAC address from the deprecated path. In the following figures, the average of these measures are represented as *path avoidance* latency. We also measure the elapsed time from the moment of the updated trust sets until the first packet reaches the intended destination, which we represent as *path-recovery* latency.

For the measurements shown in Figure 9, the following attacks and links have been configured. From all optional SA links, only the $SA9$ link is activated, which enables a nine-hop A path between source node S and the destination nodes DA, DB, and DC. From all optional SB links, only the $SB2$ link is activated, which enables an alternative two-hop B path between S and DA, DB, and DC. None of the optional SC links is activated so that no third alternative C path exists.

Moreover, Figure 9 indicates a clearly increasing entrapment latency for the S-DB route, which is plausible given that routing updates from DB need to traverse several hops with increasing packet loss before an MMA-modified update by any A-path node could delude S to perceive the nine-hop A path as shorter than the two-hop B path. The figure also shows that the $S - DA$ route is entrapped from the beginning, simply because the shortest path already traversed B nodes even before these started to attack the $S - DA$ routing.

The recovery and avoidance latencies of the $S - DA$ and $S - DB$ routes increase with increased protocol-data loss. The avoidance of a deprecated path is achieved faster than the convergence to the new trusted path. In the case of the $S - DA$ route, the average recovery increases to up to 40 seconds, which is acceptable given that routing updates along the only trusted A path need to propagate along nine links, each having a remaining success rate for propagating broadcasted protocol messages of only 40%.

7.2.4. Openness, decentrality, and cooperativeness

The attainment of further protocol objectives and desired properties shall be illustrated with the measurements given in Figure 10. The measurements represent the effect of path-length variations based on the topology of Figure 8 which extends the default scenario used for Figure 9 by enabling a third path of neutral C nodes. However, as a reference, the grey vertical bar in each figure indicate measurements that can also be found in Figure 9.

Autonomy, decentrality was demonstrated indirectly via all performed experiments because none of the security enhancements provided by the gained capability for recovering functioning end-to-end routes required the existence of any central authority or entity. Instead, all path selection were orchestrated individually and autonomously via the trusted nodes definition of each destination node.

Support for openness, neutrality, and cooperativeness, in the sense that unknown nodes could join an existing network infrastructure and be neutrally supported without prior conditions, could be well observed in Figures 10a and 10b, where the reachability of the DC node from S is always provided, although no C node is ever trusted by any A or B nodes and an alternative S-DC path of intermediate C nodes does not exist in any of the captured scenarios.

Our experiments also demonstrate how nodes benefit from the existence of paths given by unknown (so non-trusted) but correctly behaving nodes as seen from the measurements shown in Figure 10c. Here, the existence of an alternative, although non-trusted but shorter C path between S and DA, clearly reduces the avoidance latency with which the non-trusted and DA attacking B-path nodes could be rendered as deprecated.

7.3. Validation of Resource Consumption and Scalability

To evaluate our design and implementation with respect to its performance implications and our scalability objectives outlined in Section 3.1, the effect of network and protocol parameters on total protocol overhead and performance has been measured by benchmarking our implementation on target given by a cheap embedded device with hardware characteristics as summarized in Table 6 and that can be considered the bottom end of devices typically used within CNs [3, 5]. This device, running the Lede OS (an OpenWrt-based [101] Linux system) and

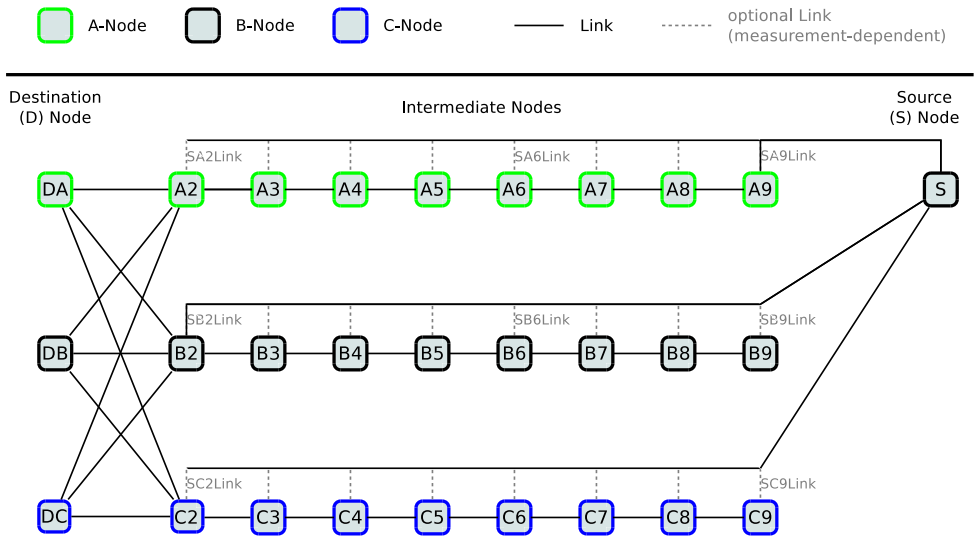


Figure 8: Network topology for measuring path-recovery performance

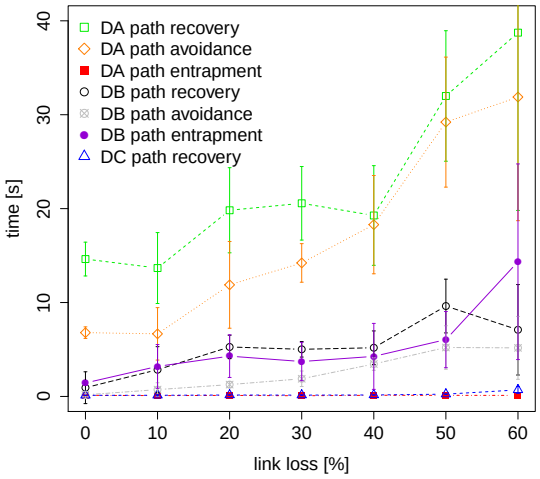


Figure 9: Path recovery latencies depending on average link loss

Table 6: HW and OS characteristics of the embedded target device

Characteristic	Details
Type / CPU	TP-Link TL-WR703N, Atheros AR7240@400 MHz
Wireless	AR9331, 802.11bgn 150 Mbps @100 mW
Flash / Memory	4 MB / 32 MB
Ports	100 MBit Ethernet, USB 2.0
Power supply	5 V, 100 mA, 0.5 W
Cost	approx 10 Euro (2013)
OS and distro	Linux OpenWrt/Lede (lede-17.01)
Further reading	http://wiki.openwrt.org/toh/tp-link/tl-wr703n

a cross-compiled version of our implementation, has been connected as a core node via Ethernet to the virtual network environment explained in section 7.1. Figure 11 illustrates the setup.

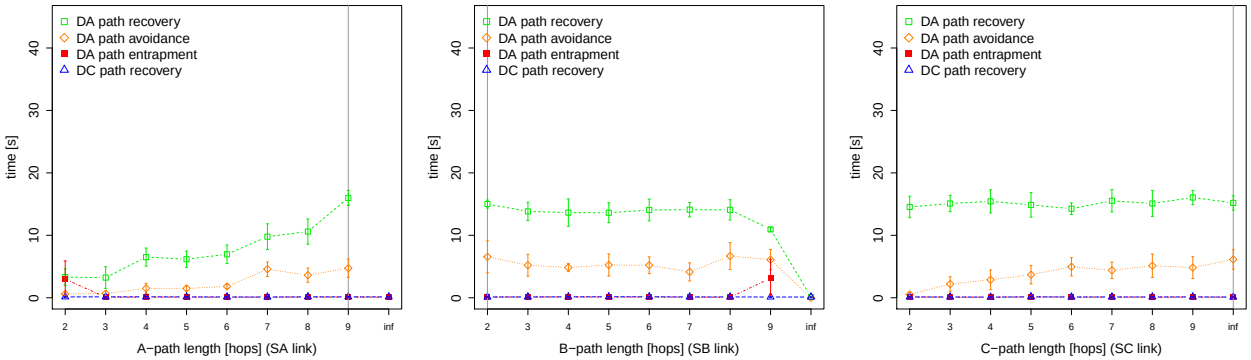
The overhead has been measured in terms of CPU usage (relative to the totally available in the node), absolute virtual memory usage (both using the Linux top command) and sent protocol-data (using tcpdump) . Graphs in Figure 12 represent the results of these measurements averaged over 30 seconds. To save space and ease comparability, CPU, memory, and protocol-traffic overhead (node TX) are represented in a single figure for each analyzed input parameter. All measurements represent the overall resources consumed by the BMX7 process on the embedded target device and not just the extra overhead added over the BMX6 implementation that was used as a basis for integrating the SEMTOR mechanisms.

The effect of nodes updating their description at higher (non-default) rates is illustrated in Figures 12b and 12e . The measurements show that protocol-data overhead and CPU usage are indeed significantly affected by such bootstrapping events and that update intervals occurring at rates higher than once per second have the potential to seriously harm the stability of the protocol, a finding that we will get back to later. To avoid capturing the effect of parallel bootstrapping phases, an unlikely scenario for real nodes under distributed administration, all other measurements have been delayed for a stabilization period of 120 seconds after each protocol-configuration change.

From Figures 12a and 12d it can be seen that network size[7] and number of links linearly affects CPU, memory, and traffic usage per node.

CPU depending on network size rises slower than the others which could be explained by the fact that the most CPU-intensive operation of continuous signing and verification of hello and routing-update messages could be significantly reduced with the symmetric DH2048-HMAC TX signatures (compared with asymmetric RSA-based TX signatures evaluated in [25]). Also, TX signatures are performed per packet which can efficiently aggregate an increasing amount of messages.

[7]Trust sets described per node were kept constant despite the varied number of participating nodes.



(a) Performance vers. trusted path length (b) Performance vers. attacked path length (c) Performance vers. neutral path length

Figure 10: Path recovery performance depending on different physical path-lengths at 0% link loss

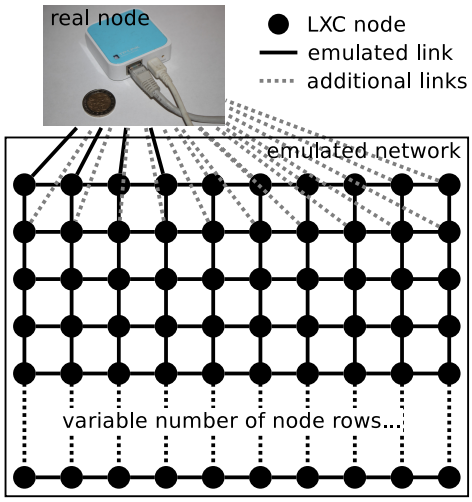


Figure 11: Experimentation setup consisting of benchmarked embedded router and MLC-emulated network environment

Figure 12d shows that CPU and traffic overhead heavily depend on the number of links to which the test node is exposed, while little additional memory requirements could be observed for maintaining an increasing number of links. This could be explained with the relatively small related memory requirements for maintaining these links compared to the allocations needed for maintaining node descriptions and thereby referenced data, which must be tracked anyway, also for non-neighbouring nodes.

In addition, the following system characteristics were observed with the objective to continuously validate the correct operation of the embedded target device itself and the emulation environment provided by the host (desktop) machine.

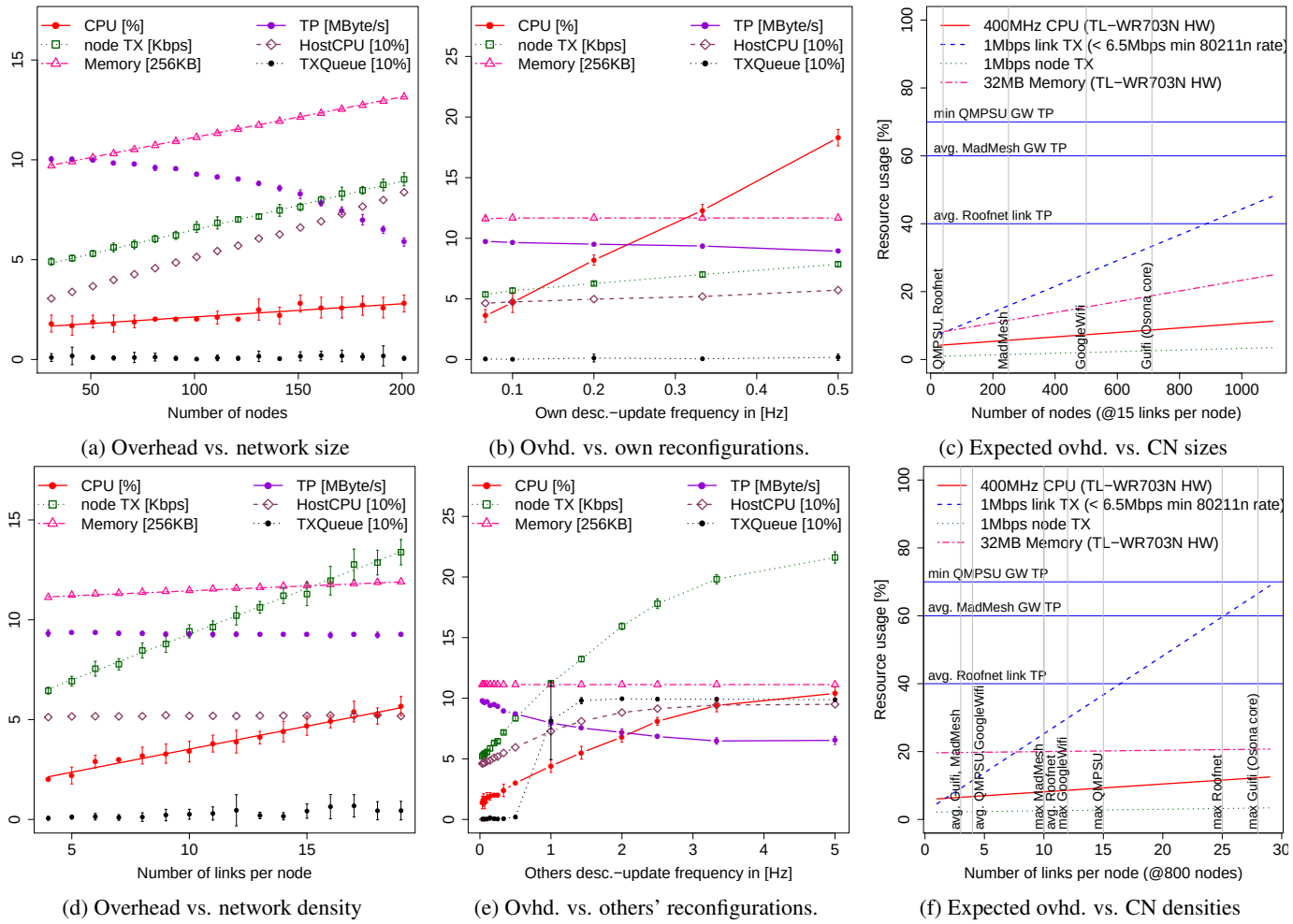
The fill rate (in 10%) of the TX-queue in the target device that is used by our implementation to throttle (via increased aggregation of scheduled messages) the transmission of protocol-data packets was captured as TXQueue. The sharp increase of the TX-queue fill rate in Figure 12e explains why the increase of the protocol-data overhead declines in the same figure. Once the fill rate reaches its maximum, scheduled messages are delayed for later aggregation, leading to less totally transmitted but more efficiently aggregated packets.

The CPU load of the host machine (given in % as HostCPU, relative to the total available processing capacity) was captured (again using top command) with all four host CPU cores fixed to their maximum frequency of 2.6 GHz.

The routing performance in terms of throughput (TP) for forwarding user-data traffic, along SEMTOR-configured routes, in and out of the target device was measured by conducting a TCP TP test (using iperf) between the $A1$ LXC node, the first node from the left in the top (A) row in Figure 11, to the $A4$ LXC node (the fourth in the same row). In all topological setups, since the target device was always linked to at least the nodes $A1$,$A2$,$A3$, and $A4$ (the first four top left nodes), the best (shortest) path between $A1$ and $A4$ always consisted of only the target device as the only intermediate hop. To ensure that only correctly routed (along this best path) TCP traffic is considered, all TCP traffic between $A1$ and $A4$ that was not routed back and forth via the target device was dropped using iptables.

The measurements confirm that (i) user traffic is continuously routed via the correct path and (ii) as long as the host CPU usage is below 50%, at the maximum speed that the 100 MBit ethernet HW of the target device is capable of processing. In fact, a clear relation can be seen between the host CPU load and the TP, which degrades when the host CPU load exceeds 50% (e.g., Figures 12a , 12e), while it appears uncorrelated with the CPU or protocol-traffic increase of the target device (e.g., Figures 12d , 12b). We explain this with the incapability of the Linux Kernel in the host machine to efficiently parallelize layer-2 packet-forwarding tasks over its four cores. Therefore, the observed degradation of routing performance can in fact be attributed to a tentative overload of the host system which, by emulating up to 200 nodes, is indeed challenged with an unrealistic setup.

The continuous lines for CPU, traffic, and memory overhead in Figures 12a and 12d represent their increase based on a linear fitted model, fed with the obtained measurement averages. This model has been used in Figures 12c and 12f for extrapolating the resource consumption for even larger and more dense



(a) Overhead vs. network size

(b) Ovhd. vs. own reconfigurations.

(c) Expected ovhd. vs. CN sizes

(d) Overhead vs. network density

(e) Ovhd. vs. others' reconfigurations.

(f) Expected ovhd. vs. CN densities

Figure 12: Measured and extrapolated overhead using DHM-packet signatures depending on network and protocol characteristics.

network scenarios and allows us to estimate the cost and limits of SEMTOR when exposed to network scenarios known from existing deployments. Therefore, the Y-axis represents in % a given assumed resource availability. Regarding CPU and memory, the used embedded target device has served as a reference for the available resources. Regarding resources for transmitting protocol-data overhead, a link capacity of 1Mbit per second has been assumed as a conservative but pragmatic upper bound (100%) for the available capacity of a (typically wireless) link. The blue horizontal lines relate this assumptions with link and path capacities known from existing wireless network deployments [69] such as the average link TP reported by Roofnet [96, 97] and the average and minimum end-to-end path TP to GW nodes reported by QMPSU [5] and MadMesh [98] which imply that each hop along these end-to-end pathes must have had an at least equal but typically greater link capacity. The gray vertical lines mark number of nodes and links per node reported from same QMPSU and MadMesh deployments, but also from GoogleWifi [99, 100] and from the Osona core network, the largest zone in Guifi.net [71, 73, 4]. From this comparison it can be seen that our SEMTOR/BMX7 implementation can be expected to well support existing deployment scenarios, generally without exhausting existing resources. Even in the case of the 40-nodes Roofnet deployment (based on rather outdated 80211b radio hardware combined with omnidirectinal antennas) the average reported link capacity of only 400kbps would not be exhausted unless the number of nodes grow to about 800 nodes (20 times of its actual reported size and assuming 15 instead of 10 average links per node).

In summary, since none of the observed performance degradations can be attributed to the overload of the target device itself, the measurements confirm the scalability of our implementation within the probed parameter space. Moreover, with regards to the observed linearly increasing CPU, memory, and protocol-data overhead depending on network size and density, the extrapolation of our measurements indicate that even several-times larger networks with up to thousand of nodes and tenths of links per node can be supported without exhausting the processing resources of the used embedded device nor the transmission capacities of wireless channels known from existing wireless network deployments.

8. Open Issues and Adjacent Solutions

In the following, routing-security aspects are discussed in the context of the objectives defined for this work and grouped into (i) those out of the scope or not considered an attack (although worth discussing), (ii) those addressed and explicitly handled, and (iii) those remaining a potential threat.

Confidentiality attacks via eavesdropping of disseminated protocol data fall into the first category. Such data includes topology, trust, identity, and if published, personal information. Particularly, the disclosure of trust sets has raised concerns, [8] as its transparent dissemination allows others to deduce trust relations between the users of a CN and exploit this knowledge for attacking individuals or groups of them in higher layers (e.g., the social layer). However, it should be noted that the provision of personal information via node descriptions or other parallel infrastructure services is left to the preferences of each node admin and could be omitted (thus revealing only trust relations between anonymous cryptographic identities) at the cost of complicating the task for identifying the nodes considered trustworthy.

Malign attacks also fall into this category, being the case where a malicious node aims to influence the reputation of another node by incorrectly reporting on its negative behaviour. However, SEMTOR does not facilitate such attacks simply because no functionality exists for detecting or reporting behaviour (good or bad) of other nodes.

This also means that **selfish, non-cooperative, and unfair behaviour** is not considered by the protocol and (as yet) is left to be solved independently. Here, approaches based on reputation such as [34, 35] or observed traffic validation [64] and distributed detection [68] mechanisms could be employed for detecting (groups of) faulty nodes and adding them to the list of non-trusted nodes.

However, with the support for verifiable and dynamically updatable node descriptions, self-bootstrapping public-key infrastructure, and individually definable trust topology, powerful tools are provided that can be used for arguing on the trustability of nodes and enforcing individual decisions without requiring consensus among network participants.

Several **data-plane attacks**, compromising confidentiality, authenticity, integrity, non-replication, and non-repudiation of user data are also considered out of the scope. Various existing and well-established end-to-end security solutions should be used instead, such as TLS-based protocols like HTTPS, VPN solutions like OpenVPN or tinc, and anonymization and privacy protocols like TOR. In fact, the security aspects covered by our protocol, operating on layer 3, complement these higher-layer solutions, as these build on the availability of a functioning (although best effort-based) routing and data-forwarding service in the first place, partially for creating encrypted overlay networks on top of it. What these higher-layer security protocols typically require are identity- and ownership-proven addresses and the availability of a functioning end-to-end packet delivery along trusted paths established between nodes owning these addresses. Related guarantees should be provided by the **control plane**, which can be attacked by disturbing the propagation of node and topology information or the forwarding of data packets. Attacks on this plane from non-trusted nodes fall into the second category, and preventing related attacks [27] (such as blackhole, partition, detour, routing-table poisoning, impersonation via replication, dropping, modification, or fabrication of protocol messages) is what the main contribution of this work is about. This is essentially achieved by excluding all non-trusted nodes from related attack-susceptible tasks, such as route-discovery, establishment, and forwarding. The key enablers for this approach include the strictly non-overlapping receiver-driven responsibility for these 'trusted tasks', the usage of a permanent strong asymmetric-key pair (used for authenticating node identity, cryptographically generated addresses ensuring conflict-free IPv6 addresses and route ownership, trusted nodes, and secondary-key replacements) and a secondary lightweight key pair of which security relies on its short lifetime and frequent replacement (used for frequent tasks of verifying communication between neighbouring nodes and to identify and discard data from non-trusted nodes).

Wormhole and denial of service (DoS) attacks fall into the third category that remains unsolved with yet-to-be defined mechanisms. While several approaches, such as TIK [80] using packet leashes or NPA [81] observing the standard deviation of round trip times, exist to defend against wormhole attacks, the threat of DoS attacks on the security of mesh-routing protocols has received much less attention.

The susceptibility to DoS attacks is actually a pillar stemming from the open and decentralization objectives pursued with our approach that provides no means for excluding malicious or selfish nodes from exploiting or exhausting other node resources. For example, an adversary can fake and propagate large numbers of physically non-existent virtual node IDs with frequently updated node descriptions that can hardly be distinguished from real nodes but whose processing may easily exceed existing memory and CPU resources in real nodes. The consequences of such attack could already be seen from the measurements performed in Section 7.3, showing overhead depending on the number of nodes or description update frequency. To defend against such attacks without sacrificing the decentralization and open preambles defined for this work, nodes may prioritize the processing of certain known nodes and throttle the support of the unknown, which is an approach we plan to research further in future work.

The design in section 5 and experimental validation in section 7 shows that it can safely separate trusted nodes from the influence of non-trusted nodes. As illustrated in Figure 4, when incorrect nodes are detected by any means, if these nodes are not considered trusted anymore and tagged as non-trusted, SEMTOR can render detected incorrect nodes harmless to the network.

In summary, with respect to the design objectives in section 3.1, SEMTOR can effectively ensure the ownership of data traffic, forwarding routes and node identities, so that they can be

[8] For example, during discussions and presentations of our approach at community events, such as the International Summit for Community Wireless Networks 2013 and the Wireless Battle Mesh in Germany 2014 and Slovenia 2015.

attributed to exactly one node (ownership in objective 1). Moreover, it can ensure the ability to communicate by a contiguous group of trusting nodes, never prevented by an external entity (security as autonomy and robustness in objective 2). In addition it can ensure following an open and decentralized model, where each node admin decides on the nodes to rely on (openness and decentralization in objective 3). Finally, it can ensure scalability (objective 4). Regarding the liveness properties, *responsiveness* is satisfied by the proactiveness of the protocol and the progress as soon as correct messages with correct information are delivered. Data *delivery* is satisfied according to the conditions in the above paragraph, across trusted nodes that are expected to be trustworthy (correct). The *freshness* and *accuracy* of the update and route information come by design of the information ownership and security mechanisms, and the *efficiency* of the protocol has been analyzed in detail in section 7.3.

Regarding the safety properties, the *consistency* and *loop-freedom* are provided by the baseline routing algorithm that ensures *route safety* combined with the security and trust mechanisms for node and path information (*update safety*). *Adversary freedom* and *adversary resilience* come from the separation between trusted and non-trusted nodes, particularly when trusted nodes are also trustworthy and incorrect nodes are or become non-trusted when detected.

9. Conclusions and Future Work

In this work, we pointed to new trust and security requirements arising in open and decentralized networks, such as CNs. We described SEMTOR, a novel routing protocol that can be used for satisfying these demands. Moreover, SEMTOR allows the verifiable and undeniable definition and distributed application of user-individual trust topologies for routing traffic towards each node. One particular advantage of SEMTOR is that it does not require a global consensus on the trustworthiness of any node. This gives each node admin the freedom to individually define the subset (and resulting sub topology) from the entire set of participating nodes that the admin considers sufficiently trustworthy to meet the security and data-delivery objectives and concerns.

Addressed security aspects have been discussed and put in context with related and orthogonal security solutions and how these can benefit by building on it. The proposed mechanisms have been implemented, tested, and evaluated for their correctness and performance to exclude non-trusted nodes from the network. Therefore, safety and liveness properties that are guaranteed by our protocol have been identified and proven with formal reasoning. The scalability of our implementation has been evaluated regarding network size, density, reconfiguration dynamics, and strength of used crypto parameters. Measurement results, obtained via benchmarking on low-end embedded hardware, show scalability limits and parameters with significant effects on performance and overhead. The results also show that the usage of strong asymmetric cryptography for building trusted routing topologies is possible, even given the scalability requirements imposed by the environment and characteristics of today's CNs. In the future, we plan to research the effects and challenges further due to the presence of large numbers of anonymous nodes and DoS attacks.

Acknowledgments

This work is partially supported by funded research projects: the European Community Framework Programme 7 (FP7) within the FIRE Initiative, Community Networks Testbed for the Future Internet (CONFINE) FP7-288535, the H2020 Programme netCommons H2020-688768, and the Spanish government TIN2013-47245-C2-1-R and TIN2016-77836-C2-2-R.

[1] C. Szabó, K. Farkas, Z. Horváth, Motivations, design and business models of wireless community networks, Mob. Netw. Appl. 13 (1-2) (2008) 147–159. doi:10.1007/s11036-008-0031-y.

[2] P. Pathak, R. Dutta, A survey of network design problems and joint design approaches in wireless mesh networks, Communications Surveys Tutorials, IEEE 13 (3) (2011) 396–428. doi:10.1109/SURV.2011.060710.00062.

[3] J. Avonts, B. Braem, C. Blondia, A questionnaire based examination of community networks, in: Wireless and Mobile Computing, Networking and Communications (WiMob), 2013 IEEE 9th International Conference on, 2013, pp. 8–15. doi:10.1109/WiMOB.2013.6673333.

[4] D. Vega, R. Baig, L. Cerdà-Alabern, E. Medina, R. Meseguer, L. Navarro, A technological overview of the guifi.net community network, Computer Networks 93, Part 2 (2015) 260 – 278, community Networks. doi:http://dx.doi.org/10.1016/j.comnet.2015.09.023.

[5] L. Cerdà-Alabern, A. Neumann, P. Escrich, Experimental evaluation of a wireless community mesh network, in: Proceedings of the 16th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems, MSWiM '13, ACM, New York, NY, USA, 2013, pp. 23–30. doi:10.1145/2507924.2507960.
URL http://doi.acm.org/10.1145/2507924.2507960

[6] Pico peering agreement, http://picopeer.net.

[7] M. Oliver, J. Zuidweg, M. Batikas, Wireless commons against the digital divide, in: 2010 IEEE International Symposium on Technology and Society, 2010, pp. 457–465. doi:10.1109/ISTAS.2010.5514608.

[8] R. Baig, R. Roca, F. Freitag, L. Navarro, Guifi.net, a crowdsourced network infrastructure held in common, Comput. Netw. 90 (C) (2015) 150–165. doi:10.1016/j.comnet.2015.07.009.

[9] D. Murray, M. Dixon, T. Koziniec, An experimental comparison of routing protocols in multi hop ad hoc networks, in: Telecommunication Networks and Applications Conference (ATNAC), 2010 Australasian, 2010, pp. 159–164. doi:10.1109/ATNAC.2010.5680190.

[10] D. Seither, A. Konig, M. Hollick, Routing performance of wireless mesh networks: A practical evaluation of batman advanced, in: Local Computer Networks (LCN), 2011 IEEE 36th Conference on, 2011, pp. 897–904. doi:10.1109/LCN.2011.6115569.

[11] A. Neumann, E. López, L. Navarro, Evaluation of mesh routing protocols for wireless community networks, Computer Networks 93, Part 2 (2015) 308 – 323, community Networks. doi:http://dx.doi.org/10.1016/j.comnet.2015.07.018.

[12] I. D. Chakeres, Aodv routing protocol implementation design, in: In ICDCSW '04: Proceedings of the 24th International Conference on Distributed Computing Systems Workshops - W7: EC (ICDCSW'04, IEEE Computer Society, 2004, pp. 698–703.

[13] C. Perkins, E. Belding-Royer, S. Das, Ad hoc On-Demand Distance Vector (AODV) Routing, RFC 3561 (Experimental) (Jul. 2003).

[14] Babel – a loop-avoiding distance-vector routing protocol, http://www.pps.univ-paris-diderot.fr/~jch/software/babel/.

[15] Juliusz Chroboczek, The Babel Routing Protocol, RFC 6126 (Experimental) (2011).

[16] BMX6 mesh networking protocol, http://bmx6.net (Jul. 2014).

[17] olsrd - an adhoc wireless mesh routing daemon, http://olsrd.org.

[18] T. Clausen, P. Jacquet, Optimized Link State Routing Protocol (OLSR), RFC 3626 (Experimental) (2003).

[19] C. Dearlove, T. Clausen, Multi-Topology Extension for the Optimized Link State Routing Protocol Version 2 (OLSRv2), RFC 7722 (Experimental) (Dec. 2015).

[20] batman advanced - a layer 2 implementation of the B.A.T.M.A.N. routing protocol, http://www.open-mesh.org/projects/batman-adv/wiki.

[21] I. F. Akyildiz, X. Wang, W. Wang, Wireless mesh networks: A survey, Comput. Netw. ISDN Syst. 47 (4) (2005) 445–487. doi:10.1016/j.comnet.2004.12.001.

[22] A. Neumann, L. Navarro, R. Baig, P. Escrich, Receiver-driven routing for community mesh networks, in: World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2013 IEEE 14th International Symposium and Workshops on a, 2013, pp. 1–7. doi:10.1109/WoWMoM.2013.6583481.

[23] P. Psenak, S. Mirtorabi, A. Roy, L. Nguyen, P. Pillay-Esnault, Multi-Topology (MT) Routing in OSPF, RFC 4915 (Proposed Standard) (Jun. 2007).

[24] M. Hauge, M. Brose, J. Sander, J. Andersson, Multi-topology routing for qos support in the consis convoy manet, in: Communications and Information Systems Conference (MCC), 2012 Military, 2012, pp. 1–8.

[25] A. Neumann, E. Lopez, L. Cerdà-Alabern, L. Navarro, Securely-entrusted multi-topology routing for community networks, in: 2016 12th Annual Conference on Wireless On-demand Network Systems and Services (WONS), 2016, pp. 1–8.

[26] S. Gupte, M. Singhal, Secure routing in mobile wireless ad hoc networks, Ad Hoc Networks 1 (1) (2003) 151 – 174. doi:http://dx.doi.org/10.1016/S1570-8705(03)00017-9.

[27] L. Abusalah, A. Khokhar, M. Guizani, A survey of secure mobile ad hoc routing protocols, Communications Surveys Tutorials, IEEE 10 (4) (2008) 78–93. doi:10.1109/SURV.2008.080407.

[28] K. Sanzgiri, D. LaFlamme, B. Dahill, B. Levine, C. Shields, E. Belding-Royer, Authenticated routing for ad hoc networks, Selected Areas in Communications, IEEE Journal on 23 (3) (2005) 598–610. doi:10.1109/JSAC.2004.842547.

[29] U. Herberg, T. Clausen, J. Milan, Digital signatures for admittance control in the optimized link state routing protocol version 2, in: Internet Technology and Applications, 2010 Int. Conf. on, 2010, pp. 1–4. doi:10.1109/ITAPP.2010.5566285.

[30] Denis Ovsienko, Babel Hashed Message Authentication Code (HMAC) Cryptographic Authentication, RFC 7298 (Experimental) (2014).

[31] Y.-C. Hu, D. B. Johnson, A. Perrig, Sead: Secure efficient distance vector routing for mobile wireless ad hoc networks, in: Proceedings of the Fourth IEEE Workshop on Mobile Computing Systems and Applications, WMCSA '02, IEEE Computer Society, Washington, DC, USA, 2002, pp. 3–.

[32] M. Guerrero-Zapata, Secure ad hoc on-demand distance vector (saodv) routing, draft-guerrero-manet-saodv-06.txt (Sep. 2006).

[33] M. G. Zapata, Key management and delayed verification for ad hoc networks, J. High Speed Netw. 15 (1) (2006) 93–109.

[34] A. Adnane, C. Bidan, R. T. de Sousa Júnior, Trust-based security for the {OLSR} routing protocol, Computer Communications 36 (10–11) (2013) 1159 – 1171. doi:http://dx.doi.org/10.1016/j.comcom.2013.04.003.

[35] P. S. Mogre, K. Graffi, M. Hollick, R. Steinmetz, A security framework for wireless mesh networks, Wirel. Commun. Mob. Comput. 11 (3) (2011) 371–391. doi:10.1002/wcm.984.

[36] M. Caesar, M. Castro, E. Nightingale, G. O'Shea, A. Rowstron, Virtual ring routing: Network routing inspired by dhts, Tech. rep., Pisa, Italy (August 2006).

[37] S. Buchegger, J.-Y. Le Boudec, Performance analysis of the confidant protocol, in: Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking &Amp; Computing, MobiHoc '02, ACM, New York, NY, USA, 2002, pp. 226–236. doi:10.1145/513800.513828.

[38] A. Neumann, Cooperation in open, decentralized, and heterogeneous computer networks, Doctoral thesis, Universitat Politècnica de Catalunya, Barcelona (Dec. 2017).
URL https://upcommons.upc.edu/handle/2117/114450

[39] P. A. Frangoudis, G. C. Polyzos, V. P. Kemerlis, Wireless community networks: an alternative approach for nomadic broadband network access., IEEE Communications Magazine 49 (5) (2011) 206–213.

[40] G. Camponovo, D. Cerutti, Wlan communities and internet access sharing: a regulatory overview, in: Mobile Business, 2005. ICMB 2005. International Conference on, IEEE, 2005, pp. 281–287.

[41] Y. Cao, M. Krebs, G. Toubekis, S. Makram, Mobile community information systems on wireless mesh networks-an opportunity for developing countries and rural areas, in: Fifth International Workshop on Ubiquitous Mobile Information and Collaboration Systems (UMICS'07), Citeseer, 2007, pp. 11–12.

[42] M. Bina, G. M. Giaglis, Unwired collective action: Motivations of wireless community participants, in: 2006 International Conference on Mobile Business, 2006, pp. 31–31. doi:10.1109/ICMB.2006.48.

[43] A. Neumann, E. Lopez, L. Navarro, An evaluation of bmx6 for community wireless networks, in: Wireless and Mobile Computing, Networking and Communications (WiMob), 2012 IEEE 8th International Conference on, 2012, pp. 651–658. doi:10.1109/WiMOB.2012.6379145.

[44] A. Zakrzewska, L. Koszalka, I. Pozniak-Koszalka, Performance study of routing protocols for wireless mesh networks, in: Proceedings of the 2008 19th International Conference on Systems Engineering, ICSENG '08, IEEE Computer Society, Washington, DC, USA, 2008, pp. 331–336. doi:10.1109/ICSEng.2008.49.

[45] D. Johnson, N. Ntlatlapa, C. Aichele, A simple pragmatic approach to mesh routing using BATMAN, in: 2nd IFIP International Symposium on Wireless Communications and Information Technology in Developing Countries, 2008, pp. 1–10.

[46] U. Ashraf, G. Juanole, S. Abdellatif, Evaluating routing protocols for the wireless mesh backbone, in: Proceedings of the Third IEEE International Conference on Wireless and Mobile Computing, Networking and Communications, WIMOB '07, IEEE Computer Society, Washington, DC, USA, 2007, pp. 40–48.

[47] M. Abolhasan, B. Hagelstein, J. C.-P. Wang, Real-world performance of current proactive multi-hop mesh protocols, in: Proceedings of the 15th Asia-Pacific conference on Communications, APCC'09, IEEE Press, Piscataway, NJ, USA, 2009, pp. 42–45.

[48] K. LaCurts, H. Balakrishnan, Measurement and analysis of real-world 802.11 mesh networks, in: Proceedings of the 10th Annual Conference on Internet Measurement (ACM), Melbourne, Australia, 2010, pp. 123–136.

[49] C. S. Hong, M. S. Siddiqui, Security issues in wireless mesh networks, in: 2007 International Conference on Multimedia and Ubiquitous Engineering (MUE'07), Vol. 00, IEEE Computer Society, Los Alamitos, CA, USA, 2007, pp. 717–722. doi:doi.ieeecomputersociety.org/10.1109/MUE.2007.187.

[50] A. Barbir, S. Murphy, Y. Yang, Generic Threats to Routing Protocols, RFC 4593 (Informational) (Oct. 2006).

[51] M. Hollick, C. Nita-Rotaru, P. Papadimitratos, A. Perrig, S. Schmid, Toward a taxonomy and attacker model for secure routing protocols, SIGCOMM Comput. Commun. Rev. 47 (1) (2017) 43–48. doi:10.1145/3041027.3041033.
URL http://doi.acm.org/10.1145/3041027.3041033

[52] P. Papadimitratos, Z. J. Haas, Securing mobile ad hoc networks, in: M. Ilyas, R. C. Dorf (Eds.), The Handbook of Ad Hoc Wireless Networks, CRC Press, Inc., Boca Raton, FL, USA, 2003, pp. 551–567.

[53] F. Hong, L. Hong, C. Fu, Secure olsr, in: Proceedings of the 19th International Conference on Advanced Information Networking and Applications - Volume 1, AINA '05, IEEE Computer Society, Washington, DC, USA, 2005, pp. 713–718. doi:10.1109/AINA.2005.305.

[54] B. R. Smith, S. Murthy, J. J. Garcia-Luna-Aceves, Securing distance-vector routing protocols, in: Proceedings of the 1997 Symposium on Network and Distributed System Security, SNDSS '97, IEEE Computer Society, Washington, DC, USA, 1997, pp. 85–.

[55] P. Papadimitratos, Z. J. Haas, Secure link state routing for mobile ad hoc networks, in: 2003 Symposium on Applications and the Internet Workshops, 2003. Proceedings., 2003, pp. 379–383. doi:10.1109/SAINTW.2003.1210190.

[56] P. Papadimitratos, Z. Haas, P. Samar, The secure routing protocol (spr) for ad hoc networks, drail-papadimitratos-secure-routing-protocol-0O.txt (2002) 12–1.

[57] P. Papadimitratos, Z. J. Haas, J. P. Hubaux, How to specify and how to prove correctness of secure routing protocols for manet, in: 2006 3rd International Conference on Broadband Communications, Networks and Systems, 2006, pp. 1–10. doi:10.1109/BROADNETS.2006.4374344.

[58] M. Wachs, A secure and resilient communication infrastructure for decentralized networking applications, Phd, Technische Universität München, München (Feb. 2015). doi:10.2313/NET-2015-02-1.

[59] G. C. Hadjichristofi, L. A. DaSilva, S. F. Midkiff, U. Lee, W. D. Sousa, Routing, security, resource management, and monitoring in ad hoc networks: Implementation and integration, Comput. Netw. 55 (1) (2011) 282–299. doi:10.1016/j.comnet.2010.09.001.

[60] S. Shen, Y. Huang, J. Ding, A cross-layer design for heterogeneous routing in wireless mesh networks, International Journal of Pervasive Computing and Communications 4 (1) (2008) 40–49. doi:10.1108/17427370810873093.

[61] F. Rusu, A. Dobra, Statistical analysis of sketch estimators, in: ACM SIGMOD International Conference on Management of Data, 2007, pp. 187–198.

[62] S. Goldberg, D. Xiao, E. Tromer, B. Barak, J. Rexford, Path-quality monitoring in the presence of adversaries, ACM SIGMETRICS Performance Evaluation Reviewdoi:10.1145/1384529.1375480.

[63] X. Zhang, C. Lan, A. Perrig, Secure and scalable fault localization under dynamic traffic patterns, in: IEEE Symposium on Security and Privacy, 2012, pp. 317–331. doi:10.1109/SP.2012.27.

[64] E. Lopez, L. Navarro, Tight bounds for sketches in traffic validation, 14th IEEE International Conference on Networking, Sensing and Control.

[65] K. Bradley, S. Cheung, N. Puketza, B. Mukherjee, R. Olsson, Detecting disruptive routers: a distributed network monitoring approach, IEEE Networkdoi:10.1109/65.730751.

[66] A. T. Mizrak, S. Savage, K. Marzullo, Detecting Malicious Packet Losses, IEEE Transactions on Parallel and Distributed Systemsdoi:10.1109/TPDS.2008.70.

[67] A. T. Mizrak, Y.-C. C. Cheng, K. Marzullo, S. Savage, Detecting and isolating malicious routers, IEEE Transactions on Dependable and Secure Computing 3 (3) (2006) 230–244.

[68] E. López, L. Navarro, Kdet: Coordinated detection of forwarding faults in wireless community networks, in: Proceedings of the 2015 IEEE Trustcom/BigDataSE/ISPA - Volume 01, TRUSTCOM '15, IEEE Computer Society, Washington, DC, USA, 2015, pp. 734–741. doi:10.1109/Trustcom.2015.441.

[69] L. Cerdà-Alabern, A. Neumann, P. Escrich, Experimental evaluation of wireless mesh networks: A case study and comparison, in: NICST'2013, New and smart Information Communication Science and Technology to support Sustainable Development, Clermont Ferrand, France, 2013.

[70] S. Vural, D. Wei, K. Moessner, Survey of experimental evaluation studies for wireless mesh network deployments in urban areas towards ubiquitous internet, Communications Surveys Tutorials, IEEE 15 (1) (2013) 223–239. doi:10.1109/SURV.2012.021312.00018.

[71] D. Vega, L. Cerdà-Alabern, L. Navarro, R. Meseguer, Topology patterns of a community network: Guifi.net, in: 1st International Workshop on Community Networks and Bottom-up-Broadband (CNBuB'2012), Barcelona, Spain, 2012, pp. 612–619.

[72] G. Castignani, L. Loiseau, N. Montavont, An evaluation of ieee 802.11 community networks deployments, in: The International Conference on Information Networking 2011 (ICOIN2011), 2011, pp. 498–503. doi:10.1109/ICOIN.2011.5723148.

[73] L. Cerdà-Alabern, On the topology characterization of guifi.net, in: 2012 IEEE 8th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob'2012), Barcelona, Spain, 2012, pp. 389–396.

[74] N. S. Evans, Methods for secure decentralized routing in open networks, Master's thesis, Technische Universität München, Garching bei München (Aug. 2011).

[75] L. Maccari, An analysis of the ninux wireless community network, in: 2013 IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), 2013, pp. 1–7. doi:10.1109/WiMOB.2013.6673332.

[76] G. Vasilakis, G. Perantinos, I. G. Askoxylakis, N. Mechin, V. Spitadakis, A. Traganitis, Business opportunities and considerations on wireless mesh networks, in: 2009 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks Workshops, 2009, pp. 1–8. doi:10.1109/WOWMOM.2009.5282419.

[77] M. Bina, Wireless community networks: A case of modern collective action, Ph.D. thesis, Athens University of Economics and Business (Jun. 2007).

[78] National Institute of Standards and Technology, FIPS PUB 180-4: Secure Hash Standard (SHS), Federal Information Processing Standards Publication, 2015.

[79] Y.-C. Hu, A. Perrig, D. B. Johnson, Wormhole attacks in wireless networks, IEEE Journal on Selected Areas in Communications 24 (2) (2006) 370–380. doi:10.1109/JSAC.2005.861394.

[80] Y.-C. Hu, A. Perrig, D. Johnson, Packet leashes: a defense against wormhole attacks in wireless networks, in: INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies, Vol. 3, 2003, pp. 1976–1986. doi:10.1109/INFCOM.2003.1209219.

[81] J. Zhou, J. Cao, J. Zhang, C. Zhang, Y. Yu, Analysis and countermeasure for wormhole attacks in wireless mesh networks on a real testbed, in: Advanced Information Networking and Applications (AINA), 2012 IEEE 26th International Conference on, 2012, pp. 59–66. doi:10.1109/AINA.2012.81.

[82] E. Lopez, L. Navarro, Coordinated Detection of Forwarding Faults in Wireless Community Networks, Journal of Network and Computer Applications (2016) 1–25.

[83] C. E. Perkins, P. Bhagwat, Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers, in: Proceedings of the Conference on Communications Architectures, Protocols and Applications, SIGCOMM '94, ACM, New York, NY, USA, 1994, pp. 234–244. doi:10.1145/190314.190336.

[84] F. Garzia, Handbook of Communications Security, WIT Press, 2013. URL https://books.google.de/books?id=uOhyAQAACAAJ

[85] R. Hinden, B. Haberman, Unique Local IPv6 Unicast Addresses, RFC 4193 (Proposed Standard) (Oct. 2005).

[86] T. Aura, Cryptographically Generated Addresses (CGA), RFC 3972 (Proposed Standard), updated by RFCs 4581, 4982 (Mar. 2005).

[87] W. Wang, Y. Lu, B. Bhargava, On security study of two distance vector routing protocols for mobile ad hoc networks, in: Proceedings of the First IEEE International Conference on Pervasive Computing and Communications, 2003. (PerCom 2003)., 2003, pp. 179–186. doi:10.1109/PERCOM.2003.1192740.

[88] R. P. Centelles, V. Oncins, A. Neumann, Enhancing reflection and self-determination in a real-life community mesh network, Computer Networks 93, Part 2 (2015) 297 – 307, community Networks. doi:http://dx.doi.org/10.1016/j.comnet.2015.09.043.

[89] L. Cerdà-Alabern, A. Neumann, L. Maccari, Experimental evaluation of bmx6 routing metrics in a 802.11an wireless-community mesh network, in: 2015 3rd International Conference on Future Internet of Things and Cloud, 2015, pp. 770–775. doi:10.1109/FiCloud.2015.28.

[90] J. P. John, E. Katz-Bassett, A. Krishnamurthy, T. Anderson, A. Venkataramani, Consensus routing: The internet as a distributed system, in: Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation, 2008, pp. 351–364.

[91] L. Lamport, Proving the correctness of multiprocess programs, IEEE transactions on software engineering 1 (2) (1977) 125–143.

[92] A. Datta, J. Franklin, D. Garg, L. Jia, D. Kaynar, On adversary models and compositional security, IEEE Security Privacy 9 (3) (2011) 26–32. doi:10.1109/MSP.2010.203.

[93] MbedTLS – SSL Library, https://tls.mbed.org/ssl-library.

[94] BMX/Semtor related public git repositories, https://github.com/axn/bmx6.

[95] Axel Neumann, Investigating Routing-Protocol Characteristics with Mesh Linux Containers (MLC), Workshop, UPC, Barcelona, Spain, https://raw.github.com/axn/mlc (Nov. 2011).

[96] D. Aguayo, J. Bicket, S. Biswas, G. Judd, R. Morris, Link-level measurements from an 802.11 b mesh network, ACM SIGCOMM Computer Communication Review 34 (4) (2004) 121–132.

[97] J. Bicket, D. Aguayo, S. Biswas, R. Morris, Architecture and evaluation of an unplanned 802.11 b mesh network, in: Proceedings of the 11th annual international conference on Mobile computing and networking, MobiCom '05, Cologne, Germany, 2005, pp. 31–42.

[98] V. Brik, S. Rayanchu, S. Saha, S. Sen, V. Shrivastava, S. Banerjee, A measurement study of a commercial-grade urban wifi mesh, in: Proceedings of the 8th ACM SIGCOMM conference on Internet measurement, 2008, pp. 111–124.

[99] M. Afanasyev, T. Chen, G. M. Voelker, A. C. Snoeren, Analysis of a mixed-use urban wifi network: When metropolitan becomes neapolitan, in: In ACM/USENIX IMC, 2008.

[100] M. Afanasyev, T. Chen, G. Voelker, A. Snoeren, Usage patterns in an urban wifi network, IEEE/ACM Transactions on Networking 18 (5) (2010) 1359–1372.

[101] Lede/OpenWrt Linux distribution for embedded devices, `http://lede-project.org` and `http://openwrt.org`.