(In)Corruptibility of Routing Protocols*

Raphaël Jamet[1] and Pascal Lafourcade[2]

[1] Univ. Grenoble Alpes, VERIMAG, F-38000 Grenoble, France
CNRS, VERIMAG, F-38000 Grenoble, France
raphael.jamet@imag.fr,
[2] Université d'Auvergne, LIMOS, France
pascal.lafourcade@udamail.fr

Abstract. Analyses of routing protocols security are nearly always supported by simulations, which often evaluate the ability to deliver messages to a given destination. Several competing definitions for secure routing exist, but to our knowledge, they only address source routing protocols. In this paper, we propose the notion of *corruptibility*, a quantitative computational definition for routing security based on the attacker's ability to alter the routes used by messages. We first define *incorruptibility*, and we follow with the definition of *bounded corruptibility*, which uses two routing protocols as bounds for the evaluated protocol. These definitions are then illustrated with several routing algorithms.

1 Introduction

Internet is made out of several independent entities controlling their own networks. To be routed, packets need to get through several networks until they reach their destination, and so the Internet can be seen as a large ad hoc network. In this context, routing relies on several protocols, including the Border Gateway Protocol (BGP, [10]). This protocol ensures the dissemination of routing information between autonomous systems (AS), and is notoriously insecure [12,13,9].

For instance, the AS 7007 incident [3] caused an internet-wide outage in 1997 because this AS declared itself able to route to the whole Internet. This declaration was made in a way that ensured most networks would choose the AS as the preferred gateway to the rest of the Internet. This misconfiguration then propagated through the Internet, overloading the faulty AS, and causing huge packet losses.

Another example, still related to BGP, has been seen more recently in the wild [7]. In this incident, China Telecom's subnetwork declared itself preferential for the routing to more than 50,000 IPs, including some strategic subnetworks for the USA. Unlike the previous example, the infrastructure of the problematic subnetwork still managed to route packets to their destination.

* This research was conducted with the support of the "Digital trust" Chair from the Foundation of the University of Auvergne.

In the context of wireless ad hoc networks (WANET), attacks and misconfigurations are not as well studied as in traditional networks. For instance, in [17], the authors present some usual routing protocol vulnerabilities. In [8], the analysis is more specific to the security of routing protocols on wireless sensor networks, which are a subset of the WANET family with more limited resources and specific protocols.

Our intuition is that a secure routing protocol should guarantee that malicious parties cannot influence the routes a message will take, or at least that this influence is limited in a clearly stated way. We call such protocols *incorruptible*, and using an incorruptible routing protocol would have prevented both previously mentioned incidents. We do not consider confidentiality or integrity of the data, as they are properties which are not necessarily tied to the routing layer. Furthermore, this study is centered on wireless networks, but we believe the notion can easily be transposed to the context of wired networks.

Contribution: We propose the first steps towards a computational notion for the security of routing protocols, based on the ability for an attacker to influence how messages are routed. We provide three measures. The first one quantifies the difference between routing protocols in a safe context. The second one, denoted *routing protocol corruption*, quantifies how much an attacker is able to change how messages are routed. The last definition allows one to prove that an attacker can only corrupt a protocol within some limits. We only consider protocols where the nodes memories do not evolve once the attack begins. Finally, we illustrate these definitions with the analysis of a simple protocol.

Related Work: In [14], the authors proposed the Source Routing Protocol (SRP), which is an on-demand route discovery protocol. In on-demand routing protocols, route discovery is the process of building a valid path for a given data message. Using BAN logic [4], they claimed that routes generated by this protocol are correct and their integrity is respected. An attack has been found later on that protocol by [11], who argued that the results of the analysis are flawed because such an analysis is "*a misuse of BAN*", as this logic has been designed to study trust relationships, and not security notions.

The authors of [1] provide a definition of provably secure on-demand route discovery. They assume the adversary has compromised a few nodes in the network which gives him full control of their actions and memories. To prove security of protocols, they use the *simulation paradigm*, which uses two models: the *real-world model*, and an *ideal-world model* where the protocol is idealized. This way, they can detect specific problems in the protocol, while avoiding the inherent problems of such routing protocols. In their model, a protocol is con-

sidered secure if the executions set in the real-world model are indistinguishable from those set in the ideal-world version. This model was expanded in [5], where the authors provided an automated way to check protocols in this model.

The main differences between our model and theirs lie in the limitations on the evaluated protocols, and the property being evaluated. Regarding protocols, their model is able to track of the evolution of the internal states on nodes and to model broadcasts, while we do not consider these situations. Regarding the properties, the one we verify is universal to routing protocols, and could be applied to any, while their property of secure route discovery only makes sense on source routing protocols.

Outline: In Section 2, we model networks and protocols in our formalism, and in Section 3 we provide some routing protocols. We present our definitions of an incorruptible routing protocol in Section 4, along with the analysis of one of the protocols given in Section 3. Finally, we conclude and present the perspectives of this work in Section 5.

2 Definitions

To represent the network *topology*, we use a vertex-labeled directed graph named the *topology*, and denoted by $T = \{V, E, f\}$, where vertices V represent network nodes, and edges E represent their connectivity. We consider only static networks, and we suppose that nodes cannot send messages to themselves. The function f associates labels to nodes, which are used to model pre-existing distinctions between nodes, such as sinks and sensors in the case of a wireless sensor network. We denote by $\mathcal{N}eig_v$ the set of neighbors of a node v (that is, the nodes at one hop of v). We notice that $v \notin \mathcal{N}eig_v$.

2.1 Routing Protocols

To forward messages, all the nodes follow a routing protocol $\mathcal{P}$. This protocol must verify that all messages are routed independently at the time of the analysis. The path that a message will take should not be influenced by what other messages have been routed before. Note that acknowledgments can be modeled by considering they are the continuation of the route of the initial message.

We define K as the array of individual node memories, denoted by $K[v]$ for any node $v \in V$. Once initialized, a node's memory is never modified again. This is a strong restriction, and it reduces the range of protocols that can be modeled. On the other hand, for a given message, these protocols generate routes that

do not depend on the past messages, which is an important property for the following security proofs, and we embrace that assumption in the rest of this paper. We discuss ways to lift this restriction in the conclusion.

We denote by η_d the data size, and by η the security parameter for cryptographic functions. $a \xleftarrow{\$} X$ denotes that a is a random value obtained according the distribution represented by X. If X is a set, a is randomly drawn using the uniform law on X. Similarly, if X is a probabilistic algorithm, a is drawn at random using the algorithm.

We define a routing protocol $\mathcal{P}$ as the set of four oracles $\{\mathcal{P}^I, \mathcal{P}^G, \mathcal{P}^R, \mathcal{P}^D\}$ which respectively model the initialization, message generation, routing and the depacketing phases. They are defined in Definitions 1 through 4.

Definition 1 (Initialization oracle). *Let $T = \{V, E, f\}$ be a topology, which contains nodes $v_1 \dots v_n \in V$. The* initialization oracle $\mathcal{P}^I(T, \eta)$ *models the setup phase of $\mathcal{P}$ on the topology represented by T, with security parameter η, which initializes the memories of the nodes. This oracle call returns K, an array associating to each node its memories.*

Definition 2 (Message generation oracle). *Let $T = \{V, E, f\}$ be a topology with $o, d \in V$. Once memories K have been initialized, the* message generation oracle $\mathcal{P}^G_K(o, d, \eta_d)$ *models the generation of a new random data of length η_d to route by o, for d. This oracle call returns a message m, and does not modify K.*

Definition 3 (Routing oracle). *Let $T = \{V, E, f\}$ be a topology with $v \in V$. Once memories K have been initialized,* the routing oracle $\mathcal{P}^R_K(v, m)$ *models how v would route m given the initialization K. This oracle call returns either $\perp$ if no message is forwarded, or (w, m') if a message m' is forwarded to w (with $w \in \mathcal{N}eig_v$). That call does not modify K.*

Definition 4 (Depacketing oracle). *Let $T = \{V, E, f\}$ be a topology with $d \in V$. Once memories K have been initialized,* the depacketing oracle $\mathcal{P}^D_K(d, m)$ *models how d would unpack the message m given the initialization K. This oracle call returns either the $Data$ contained in the message if extractable, or $\perp$ if that operation is not possible. That call does not modify K.*

2.2 Message Lifecycle and Routes

We now present how to model the natural lifecycle of a message. First, the network needs to be initialized by calling $K \xleftarrow{\$} \mathcal{P}^I(T, \eta)$. Then, a new message containing a random data is generated by a node o with destination d using the generation oracle $m_0 = \mathcal{P}^G_K(o, d, \eta_d)$. That message is first routed

by the node o with the routing oracle $\mathcal{P}_K^R(o, m_0) = (h_1, m_1)$, assuming that message is not dropped. We then continue with its first hop, h_1, who reacts accordingly: $\mathcal{P}_K^R(h_1, m_1) = (h_2, m_2)$. This process continues until the message finally reaches a node h_n such that $\mathcal{P}_K^R(h_n, m_n) = \perp$: at this point, the message is stopped. We refer to such a sequence $[h_0, \ldots, h_n] = \mathbf{R}$ as a *route*, which can be empty, in which case it is denoted by $[\,]$.

Definition 5 (GENROUTE$(m_0, h_0, \mathcal{P}_K^R)$). *Given a message m_0, a node identifier h_0, and a routing oracle $\mathcal{P}_K^R$ initialized with node memories K, we define* GENROUTE$(m_0, h_0, \mathcal{P}_K^R)$ *the function that generates a route for m_0 starting at h_0. This function calls $\mathcal{P}_K^R$, first with arguments (h_0, m_0), and then with the pair (h_i, m_i) returned by the previous call, until the oracle returns $\perp$. The function returns the route $[h_1, \ldots, h_n]$.*

As $\mathcal{P}$ can be probabilistic, making several calls to GENROUTE with the same arguments can result in different routes. However, since we require message routing independence, the probabilistic distribution of routes should stay the same, no matter what messages have been routed before. Finally, we define a predicate on routes which we denote by Φ.

Definition 6 ($\Phi(\mathbf{R}, a, b)$). *Given a route $\mathbf{R}$ and two nodes a and b, $\Phi(\mathbf{R}, a, b)$ is a function that returns true if and only if a route $\mathbf{R}$ contains a and that a appears before any occurrence of b.*

3 Examples of Routing Protocols

We now provide some routing protocols in our formalism. Given a message m and the initialized memories array K, the $\mathcal{P}^D$ oracles return the $Data$ that is contained in m.

Let $\mathcal{S}$ be a signature scheme with three functions: GENASYMKEYPAIR(η) generates asymmetric key pairs given the security parameter η, SIGN(x, sk) generates a signature of x using the key sk, VERIFY(x, pk, S) verifies a signature S against the input x with key pk. The function SHORTROUTE(o, d, T) takes as input an origin, a destination, and a topology, and returns uniformly at random one of the shortest routes between the origin and destination. Finally, FINDNEXT$(\mathbf{R}, v)$ is the function that returns the node identifier coming right after v in the route $\mathbf{R}$, or $\perp$ otherwise.

The null protocol $\mathcal{P}\emptyset$ is defined in Figure 1. It drops all messages, and adds no information in the packets it generates. The uniform random walk $\mathcal{RW}$ is defined in Figure 2. The following protocol is called $\mathcal{SI}$ (for Shortest-Insecure) and it is described in Figure 3. That protocol stores routes in messages without protecting them.

We define two other protocols, which stem from $\mathcal{SI}$, and use the signature scheme $\mathcal{S}$. First, $\mathcal{S}\emptyset$ (Figure 4) is a secured version of $\mathcal{SI}$, which prevents any alteration to the route stored in a message by using signatures. The route is signed by the message sender, and if that signature does not verify, then the message is discarded. The next protocol, $\mathcal{SR}$ (Figure 5), works in a similar way, but instead of discarding the message, it routes it randomly until the message reaches the destination.

4 Incorruptibility

We now present how our notion of incorruptibility is formalized, and how it can be used to show the security of routing protocols. We begin by defining what is an attacker in our context, and follow with the measures of distance, incorruptibility, and bounded corruptibility.

4.1 Attacker

Our model deals with an attacker external to the network, who did not compromise any honest node. This entity controls the network links, and is able to intercept, create and manipulate messages. Its goal is to alter how a challenge message is routed. Notice that manipulating communications implies the possibility of forging messages to observe how nodes would react, and to observe which messages are generated in the network. However, since we suppose the attacker is external to the network, it does not have any direct access to the node's memories.

This adversary is modeled as a probabilistic polynomial-time Turing machine. It can query the oracles $\mathcal{P}_K^G$, $\mathcal{P}_K^R$ and $\mathcal{P}_K^D$ a polynomial number of times, but does not have direct access to the array of node memories K. None of its actions can modify K by hypothesis, but the adversary has its own memory. Note that we provide access to $\mathcal{P}_K^D$ to the adversary, as we are not concerned by confidentiality.

We denote the trivial attacker that returns its given input message by $\mathcal{A}_{safe}$. We name it *"safe"* as it is a placeholder attacker that effectively does nothing.

4.2 Measuring How Routing Protocols Operate

We define an experiment named $\mathbf{Expt}_{\mathcal{P}}^{Rt}$, which allows us to reason on an adversary's ability to influence how a message is routed.

Initialization $\mathcal{P}\emptyset^I(T,\eta)$:

1: **Return** $\emptyset$

Message generation $\mathcal{P}\emptyset^G_K(o,d,\eta_d)$:

1: $Data \xleftarrow{\$} \{0,1\}^{\eta_d}$
2: **Return** $Data$

Message routing $\mathcal{P}\emptyset^R_K(m,v)$:

1: **Return** $\perp$

Fig. 1: Protocol $\mathcal{P}\emptyset$

Initialization $\mathcal{RW}^I(T,\eta)$:

1: **Return** $\emptyset$

Message generation $\mathcal{RW}^G_K(o,d,\eta_d)$:

1: $Data \xleftarrow{\$} \{0,1\}^{\eta_d}$
2: **Return** $next, (Data, d)$

Message routing $\mathcal{RW}^R_K(m,v)$:

1: $(Data, d) \leftarrow m$
2: **if** $v \neq d$ **then**
3: $\quad next \xleftarrow{\$} \mathcal{N}eig_v$
4: $\quad$ **Return** $next, (Data, d)$
5: **else**
6: $\quad$ **Return** $\perp$
7: **end if**

Fig. 2: Protocol $\mathcal{RW}$

Initialization $\mathcal{SI}^I(T,\eta)$:

1: **for** all nodes v in T **do**
2: $\quad K[v] \leftarrow T$
3: **end for**
4: **Return** K

Message generation $\mathcal{SI}^G_K(o,d,\eta_d)$:

1: $Data \xleftarrow{\$} \{0,1\}^{\eta_d}$
2: $\mathbf{R} \leftarrow$ SHORTROUTE$(o,d,K[o])$
3: **Return** $(Data, \mathbf{R}, d)$

Message routing $\mathcal{SI}^R_K(m,v)$:

1: **if** $v \neq d$ **then**
2: $\quad next \leftarrow$ FINDNEXT$(\mathbf{R},v)$
3: $\quad$ **Return** $(next, (Data, \mathbf{R}, d))$
4: **end if**
5: **Return** $\perp$

Fig. 3: Protocol $\mathcal{SI}$

Initialization $\mathcal{S}\emptyset^I(T,\eta)$:

1: **for** all nodes v in T **do**
2: $\quad (pk[v], sk[v]) \leftarrow$ GENASYMKEYPAIR(η)
3: $\quad K[v] \leftarrow (T, pk, sk[v])$
4: **end for**
5: **Return** K

Message generation $\mathcal{S}\emptyset^G_K(o,d,\eta_d)$:

1: $Data \xleftarrow{\$} \{0,1\}^{\eta_d}$
2: $\mathbf{R} \leftarrow$ SHORTROUTE$(o,d,K[o])$
3: $S \leftarrow$ SIGN$((Data, \mathbf{R}, o, d), sk[o])$
4: **Return** $(Data, \mathbf{R}, o, d, S)$

Message routing $\mathcal{S}\emptyset^R_K(m,v)$:

1: $(Data, \mathbf{R}, o, d, S) \leftarrow m$
2: **if** $v \neq d$ **then**
3: $\quad$ **if** VERIFY$((Data, \mathbf{R}, o, d), pk[o], S)$ **then**
4: $\quad\quad next \leftarrow$ FINDNEXT$(\mathbf{R},v)$
5: $\quad\quad$ **Return** $(next, (Data, \mathbf{R}, o, d, S))$
6: $\quad$ **end if**
7: **end if**
8: **Return** $\perp$

Fig. 4: Protocol $\mathcal{S}\emptyset$

Initialization $\mathcal{SR}^I(T,\eta)$:

1: **for** all nodes v in T **do**
2: $\quad (pk[v], sk[v]) \leftarrow$ GENASYMKEYPAIR(η)
3: $\quad K[v] \leftarrow (T, pk, sk[v])$
4: **end for**
5: **Return** K

Message generation $\mathcal{SR}^G_K(o,d,\eta_d)$:

1: $Data \xleftarrow{\$} \{0,1\}^{\eta_d}$
2: $\mathbf{R} \leftarrow$ SHORTROUTE$(o,d,K[o])$
3: $S \leftarrow$ SIGN$((Data, \mathbf{R}, o, d), sk[o])$
4: **Return** $(Data, \mathbf{R}, o, d, S)$

Message routing $\mathcal{SR}^R_K(m,v)$:

1: $(Data, \mathbf{R}, o, d, S) \leftarrow m$
2: **if** $v \neq d$ **then**
3: $\quad$ **if** VERIFY$((Data, \mathbf{R}, o, d), pk[o], S)$ **then**
4: $\quad\quad next \leftarrow$ FINDNEXT$(\mathbf{R},v)$
5: $\quad$ **else**
6: $\quad\quad next \xleftarrow{\$} \mathcal{N}eig_v$
7: $\quad$ **end if**
8: $\quad$ **Return** $(next, (Data, \mathbf{R}, o, d, S))$
9: **end if**
10: **Return** $\perp$

Fig. 5: Protocol $\mathcal{SR}$

Definition 7 ($\mathbf{Expt}^{Rt}_{\mathcal{P}}(\mathcal{A}, T, o, d, s, a, b, \eta, \eta_d)$). *Let $\mathcal{P}$ be a routing protocol. Let $\mathcal{A}$ be an adversary and $T = \{V, E, f\}$ a topology with $o, d, s, a, b \in V$. We define:*

$$
\begin{aligned}
&\mathbf{Expt}^{Rt}_{\mathcal{P}}(\mathcal{A}, T, o, d, s, a, b, \eta, \eta_d) : \\
&\quad K \xleftarrow{\$} \mathcal{P}^I(T, \eta) \\
&\quad m \xleftarrow{\$} \mathcal{P}^G_K(o, d, \eta_d) \\
&\quad m' \xleftarrow{\$} \mathcal{A}^{\mathcal{P}^R_K, \mathcal{P}^G_K, \mathcal{P}^I, \mathcal{P}^D_K}(m, o, d, s, a, b, T) \\
&\quad \textbf{If } \mathcal{P}^D_K(d, m') \neq \mathcal{P}^D_K(d, m) \\
&\quad\quad m' \leftarrow m \\
&\quad \mathbf{R} \xleftarrow{\$} \textsc{GenRoute}(m', s, \mathcal{P}^R_K) \\
&\quad \textbf{Return } \Phi(\mathbf{R}, a, b)
\end{aligned}
$$

First, the initialization is done by calling $\mathcal{P}^I(T, \eta)$, which returns the array of node memories K that is used through the experiment. A challenge message m is generated using K, and given to the adversary. The adversary should then change m in a new message m', containing the same data as m. The experiment returns a value $\Phi(\mathbf{R}, a, b)$ with $\mathbf{R}$ a route generated for m' from the node s. The predicate $\Phi(\mathbf{R}, a, b)$ is true when the route passes through a before b. We use the equality of depacketed messages as a way to prevent replay attacks: an attacker has no incentive in returning new messages containing random data, as their answer would get replaced by the challenge message, which they could have output in the first place. Note that we do not take into account the messages formats or contents because we focus only on their route.

For instance, the return value of $\mathbf{Expt}^{Rt}_{\mathcal{P}}(\mathcal{A}_{safe}, T, o, d, s, a, b, \eta, \eta_d)$ models whether a random message m generated by o in destination of d gets routed by a before b when sent first from s, when all those nodes follow the routing protocol $\mathcal{P}$. When we use an arbitrary adversary $\mathcal{A}$, then the experiment $\mathbf{Expt}^{Rt}_{\mathcal{P}}(\mathcal{A}, T, o, d, s, a, b, \eta, \eta_d)$ represents the same observation, except that m has been tampered with by $\mathcal{A}$ before being routed by s. We remark that $\mathcal{P}\emptyset$ has an interesting property here: for any $\mathcal{A}, T$ and o, d, s, a, b, the probability $Pr[\mathbf{Expt}^{Rt}_{\mathcal{P}\emptyset}(\mathcal{A}, T, o, d, s, a, b, \eta, \eta_d)] = 0$.

We then compare two such measures in order to define the *distance* between a tuple protocol, attacker and another.

Definition 8 (Distance). *For a topology $T = \{V, E, f\}$ with nodes $o, d, s, a, b \in V$, two adversaries $\mathcal{A}_1$ and $\mathcal{A}_2$, two protocols $\mathcal{P}_1$ and $\mathcal{P}_2$, we define the distance $Dist((\mathcal{P}_1, \mathcal{A}_1), (\mathcal{P}_2, \mathcal{A}_2), T, o, d, s, a, b, \eta, \eta_d)$ as*

$$\left| Pr[\mathbf{Expt}^{Rt}_{\mathcal{P}_1}(\mathcal{A}_1, T, o, d, s, a, b, \eta, \eta_d)] - Pr[\mathbf{Expt}^{Rt}_{\mathcal{P}_2}(\mathcal{A}_2, T, o, d, s, a, b, \eta, \eta_d)] \right|$$

The notion of distance is a way to compare the routes being generated by $(\mathcal{P}_1, \mathcal{A}_1)$ and $(\mathcal{P}_2, \mathcal{A}_2)$, given T and o, d, s, a, b. We now present how to measure observable differences between routing protocols using this experiment.

4.3 Routing Similarity

We begin by expressing the similarity of routing protocols using $Dist$. We recall that stating that a function $\mu(x) : \mathbb{N} \to \mathbb{R}$ is negligible in x means that for every positive polynomial P there exists an integer I such that for all $x > I$, $\mu(x) < |\frac{1}{P(x)}|$, as given in [2].

Definition 9 (Routing protocols similarity). *For a topology $T = \{V, E, f\}$, we say that two protocols $\mathcal{P}_1$ and $\mathcal{P}_2$* route messages similarly *on the topology T if $\forall o, d, s, a, b \in V$, $Dist((\mathcal{P}_1, \mathcal{A}_{safe}), (\mathcal{P}_2, \mathcal{A}_{safe}), T, o, d, s, a, b, \eta, \eta_d)$ is negligible in η and in η_d.*

Intuitively, two protocols route messages similarly on a topology if the routes generated for random messages are computationally indistinguishable for all origins o, destinations d, and senders s. For instance, $\mathcal{RW}$ and $\mathcal{SI}$ are not similar for all topologies, as the latter generates distinguishably shorter routes. However, on a topology T_2 consisting of two connected nodes, they are similar, as messages are either routed to the neighbor if $o \neq d$ and not routed at all otherwise.

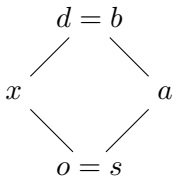

Fig. 6: Topology T_s

Consider for instance $\mathcal{S}\emptyset$ and $\mathcal{P}\emptyset$ on a topology T_s as described in Figure 6. We observe that on this topology, the probability $Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A}_{safe}, T_s, o, d, s, a, b, \eta, \eta_d)] = 0.5$, as there are two shortest routes from o to d and only one reaches a before b. We know that $Pr[\mathbf{Expt}^{Rt}_{\mathcal{P}\emptyset}(\mathcal{A}_{safe}, T_s, o, d, s, a, b, \eta, \eta_d)] = 0$, since null routes will never reach any of the two nodes. So, we can deduce that $Dist((\mathcal{S}\emptyset, \mathcal{A}_{safe}), (\mathcal{P}\emptyset, \mathcal{A}_{safe}), T_s, o, d, s, a, b, \eta, \eta_d) = 0.5$, which means that those two protocols are not similar on T_s, and we can conclude that $\mathcal{P}\emptyset$ and $\mathcal{S}\emptyset$ are not *similar on every topology*, as they differ on at least T_s.

Notice that this definition does not include attackers: two protocols routing messages similarly may not behave in the same way in presence of an *active* adversary. This allows us to show that a secure version of a protocol is similar to its original counterpart. For instance, $\mathcal{SR}$, $\mathcal{S}\emptyset$ and $\mathcal{SI}$ are all similar.

4.4 Incorruptibility of a Protocol

We propose a measure which evaluates whether an attacker can alter a message m into another message m' in order to make its routing distinguishably different. We call this measure the *incorruptibility* of a routing protocol, the related advantage is denoted by $\mathbf{Adv}^{INC}_{\mathcal{P}}$, and we define it using $Dist$.

Definition 10 ($\mathbf{Adv}^{INC}_{\mathcal{P}}(\mathcal{A}, T, o, d, s, a, b, \eta, \eta_d)$)**.** *For an adversary $\mathcal{A}$, a topology $T = \{V, E, f\}$ with five nodes $o, d, s, a, b \in V$, we define*
$\mathbf{Adv}^{INC}_{\mathcal{P}}(\mathcal{A}, T, o, d, s, a, b, \eta, \eta_d) = Dist((\mathcal{P}, \mathcal{A}), (\mathcal{P}, \mathcal{A}_{safe}), T, o, d, s, a, b, \eta, \eta_d)$

Definition 11 (Incorruptible protocol). *If for any adversary $\mathcal{A}$, any topology $T = \{V, E, f\}$, and any five nodes $o, d, s, a, b \in V$, the value of the advantage $\mathbf{Adv}^{INC}_{\mathcal{P}}(\mathcal{A}, T, o, d, s, a, b, \eta, \eta_d)$ is negligible in η and in η_d, we say that $\mathcal{P}$ is* incorruptible*.*

Informally, a protocol is corruptible if an adversary's alterations of a message can result in distinguishably different routes. For instance, the $\mathcal{P}\emptyset$ protocol is incorruptible, as it always generates null routes. Similarly, $\mathcal{RW}$ is incorruptible: it is not influenced by any information contained in the messages, and so intuitively an attacker which can only alter the content of a message is not able to influence in any way how a message is routed.

However, this definition is too restrictive for some protocols that intuitively cannot be attacked, such as $\mathcal{S}\emptyset$. Most protocols whose behavior depends on the message contents can be influenced, as an attacker can use that dependency in order to differ from the safe behavior of the protocol. We now provide an attacker for $\mathcal{S}\emptyset$ to illustrate this reasoning, and in the next subsection, we provide a generalization of the incorruptibility advantage to answer those concerns.

In order to show the corruptibility of $\mathcal{S}\emptyset$ (which is described in Figure 4), we use the adversary $\mathcal{A}_{zero}$ that takes as input the message $m = (Data, \mathbf{R}, o, d, S)$, and returns the altered $m' = (Data, \mathbf{R}, o, d, 0)$. Intuitively, this attacker destroys the signature S of the message m, which ensures the protocol drops it at the next hop. This behavior differs significantly from how the original m would have been routed.

We recall the definition of $\mathbf{Adv}^{INC}_{\mathcal{S}\emptyset}(\mathcal{A}_{zero}, T, o, d, s, a, b, \eta, \eta_d) =$

$$\left|Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A}_{zero}, o, d, s, a, b, \eta, \eta_d)] - Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A}_{safe}, o, d, s, a, b, \eta, \eta_d)]\right|$$

We omit T, o, d, s, a, b, and η, η_d from the parameters list when it is clear from the context. We are first interested in the left part of this subtraction. $\mathcal{A}_{zero}$ changes the signatures of messages it is given. Let us consider what happens with an altered message m'_{zero} (containing S_{zero}) and its corresponding $\mathbf{R}_{zero}$,

generated in the $\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A}_{zero}, o, d, s, a, b, \eta, \eta_d)$ experiment. We separate the case where S is valid and where it is not. We have:

$$\begin{aligned}
Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A}_{zero}, o, d, s, a, b, \eta, \eta_d)] = \\
&Pr[\Phi(\mathbf{R}_{zero}, a, b) | \textsc{Verify}((Data, \mathbf{R}_{zero}, o, d), pk[o], S_{zero})] \times \\
&Pr[\textsc{Verify}((Data, \mathbf{R}_{zero}, o, d), pk[o], S_{zero})] + \\
&Pr[\Phi(\mathbf{R}_{zero}, a, b) | \neg\textsc{Verify}((Data, \mathbf{R}_{zero}, o, d), pk[o], S_{zero})] \times \\
&Pr[\neg\textsc{Verify}((Data, \mathbf{R}_{zero}, o, d), pk[o], S_{zero})]
\end{aligned}$$

If we assume that $\mathcal{S}\emptyset$ uses a secure (UF-CMA in the sense of [6]) signature scheme $\mathcal{S}$ of security parameter η, then we know that the probability ϵ of the signature being forged by an intruder (*i.e.* $\textsc{Verify}((Data, \mathbf{R}, o, d), pk[o], S)$ returns true) becomes *negligible* in η. Therefore:

$$\begin{aligned}
Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A}_{zero}, o, d, s, a, b, \eta, \eta_d)] = \\
&Pr[\Phi(\mathbf{R}_{zero}, a, b) | \textsc{Verify}((Data, \mathbf{R}_{zero}, o, d), pk[o], S_{zero})] \times \epsilon + \\
&Pr[\Phi(\mathbf{R}_{zero}, a, b) | \neg\textsc{Verify}((Data, \mathbf{R}_{zero}, o, d), pk[o], S_{zero})] \times (1 - \epsilon)
\end{aligned}$$

We first consider the case where the signature is invalid. Consider the oracle $\mathcal{S}\emptyset^R_K$ described in Figure 4. If the signature of the message is not valid, then the message is dropped. Therefore, all the routes generated for m' in this context are equal to the empty route $[\ ]$. We know that $\Phi([\ ], a, b)$ is always false for any a and b. We can therefore conclude that the experiment returns 0 with a probability $(1 - \epsilon)$, and remove it from the equation.

$$\begin{aligned}
Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A}_{zero}, o, d, s, a, b, \eta, \eta_d)] = \\
&Pr[\Phi(\mathbf{R}_{zero}, a, b) | \textsc{Verify}((Data, \mathbf{R}_{zero}, o, d), pk[o], S_{zero})] \times \epsilon
\end{aligned}$$

We denote by p the probability of the experiment returning 1 when the signature is valid. Going back to the advantage, we have:

$$\mathbf{Adv}^{INC}_{\mathcal{S}\emptyset}(\mathcal{A}_{zero}) = \left|(0 + \epsilon \times p) - Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A}_{safe})]\right|$$

The first part of the subtraction is negligible in η, as p is a probability and ϵ is negligible in η. However, $Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A}_{safe}, T, o, d, s, a, b)]$ may not be negligible, depending on T and o, d, s, a, b (as we have shown using the topology in Figure 6). Intuitively, without attacker interference, $\mathcal{S}\emptyset$ actually routes messages to their destination, which ensures the existence of such nodes. Therefore, there exist some topologies T (T_s being one of them) where the advantage $\mathbf{Adv}^{INC}_{\mathcal{S}\emptyset}(\mathcal{A}_{zero}, T, o, d, s, a, b, \eta, \eta_d)$ is not negligible in η and in η_d, and so $\mathcal{S}\emptyset$ is not *incorruptible on all topologies*.

4.5 Bounded Corruptibility

We generalize the notion of corruptibility to a definition using two reference protocols. We define another advantage, called $\mathbf{Adv}^{BINC}_{\mathcal{P},\mathcal{B}_1,\mathcal{B}_2}$. It follows the same principle as $\mathbf{Adv}^{INC}_{\mathcal{P}}$, but instead of considering how an attacker can force $\mathcal{P}$ to behave differently, we consider how it can be corrupted to the outside of a reference routing *interval*, defined by the safe execution of two protocols $\mathcal{B}_1$ and $\mathcal{B}_2$ on T, measured for the parameters o, d, s, a, b.

Definition 12 ($\mathbf{Adv}^{BINC}_{\mathcal{P},\mathcal{B}_1,\mathcal{B}_2}(\mathcal{A}, T, o, d, s, a, b, \eta, \eta_d)$). *Let $\mathcal{A}$ be an attacker, and let $T = \{V, E, f\}$ be a topology with nodes $o, d, s, a, b \in V$. We consider a protocol $\mathcal{P}$, which is compared with two protocols $\mathcal{B}_1$ and $\mathcal{B}_2$. We define $\mathbf{Adv}^{BINC}_{\mathcal{P},\mathcal{B}_1,\mathcal{B}_2}(\mathcal{A}, T, o, d, s, a, b, \eta, \eta_d)$ as:*

$$\begin{aligned} max(\ & Dist((\mathcal{B}_1, \mathcal{A}_{safe}), (\mathcal{B}_2, \mathcal{A}_{safe}), T, o, d, s, a, b, \eta, \eta_d), \\ & Dist((\mathcal{P}, \mathcal{A}), (\mathcal{B}_1, \mathcal{A}_{safe}), T, o, d, s, a, b, \eta, \eta_d), \\ & Dist((\mathcal{P}, \mathcal{A}), (\mathcal{B}_2, \mathcal{A}_{safe}), T, o, d, s, a, b, \eta, \eta_d)\ ) \\ )\ & - Dist((\mathcal{B}_1, \mathcal{A}_{safe}), (\mathcal{B}_2, \mathcal{A}_{safe}), T, o, d, s, a, b, \eta, \eta_d) \end{aligned}$$

Informally, $\mathbf{Adv}^{BINC}_{\mathcal{P},\mathcal{B}_1,\mathcal{B}_2}(\mathcal{A}, T, o, d, s, a, b, \eta, \eta_d)$ is a measure of the maximal distance the behavior of $\mathcal{P}$ attacked by $\mathcal{A}$ can get from the outside of the interval determined by the safe behavior of $\mathcal{B}_1$ and $\mathcal{B}_2$. Remark that if the attacked protocol's behavior is in the interval, then the advantage is 0.

Definition 13 (Bounded corruptibility). *If for any adversary $\mathcal{A}$, for any topology T, and for all o, d, s, a, b, $\mathbf{Adv}^{BINC}_{\mathcal{P},\mathcal{B}_1,\mathcal{B}_2}(\mathcal{A}, T, o, d, s, a, b, \eta, \eta_d)$ is negligible in η and in η_d, we say that $\mathcal{P}$'s corruptibility is bounded between $\mathcal{B}_1$ and $\mathcal{B}_2$.*

Remark that this definition has some interesting properties:

- $\mathbf{Adv}^{BINC}_{\mathcal{P},\mathcal{B}_1,\mathcal{B}_2}(\mathcal{A}, T, o, d, s, a, b, \eta, \eta_d) = \mathbf{Adv}^{BINC}_{\mathcal{P},\mathcal{B}_2,\mathcal{B}_1}(\mathcal{A}, T, o, d, s, a, b, \eta, \eta_d)$: The bounds for bounded corruptibility are commutative.
- $\mathbf{Adv}^{BINC}_{\mathcal{P},\mathcal{P},\mathcal{P}}(\mathcal{A}, T, o, d, s, a, b, \eta, \eta_d) = \mathbf{Adv}^{INC}_{\mathcal{P}}(\mathcal{A}, T, o, d, s, a, b, \eta, \eta_d)$: Stating that a protocol is bounded between itself and itself is the same as stating its incorruptibility.
- $Dist(\mathcal{B}_1, \mathcal{B}_2, T, o, d, s, a, b, \eta, \eta_d) = 0 \Rightarrow \forall \mathcal{B}_3$, $\mathbf{Adv}^{BINC}_{\mathcal{P},\mathcal{B}_1,\mathcal{B}_3}(\mathcal{A}, T, o, d, s, a, b, \eta, \eta_d) = \mathbf{Adv}^{BINC}_{\mathcal{P},\mathcal{B}_2,\mathcal{B}_3}(\mathcal{A}, T, o, d, s, a, b, \eta, \eta_d)$: If two protocols route messages identically, then those protocols are equivalent for bounding purposes.

Example: Bounded Corruptibility of $\mathcal{S}\emptyset$: We try to bound $\mathcal{S}\emptyset$ using itself and $\mathcal{P}\emptyset$. We assume that $\mathcal{S}\emptyset$ uses a secure UF-CMA signature scheme [6] $\mathcal{S}$ of security parameter η. By definition, $\mathbf{Adv}^{BINC}_{\mathcal{S}\emptyset,\mathcal{S}\emptyset,\mathcal{P}\emptyset}(\mathcal{A}, T, o, d, s, a, b)$ equals:

$$
\begin{aligned}
max(&Dist((\mathcal{S}\emptyset, \mathcal{A}_{safe}), (\mathcal{P}\emptyset, \mathcal{A}_{safe})),\\
&Dist((\mathcal{S}\emptyset, \mathcal{A}), (\mathcal{S}\emptyset, \mathcal{A}_{safe})),\\
&Dist((\mathcal{S}\emptyset, \mathcal{A}), (\mathcal{P}\emptyset, \mathcal{A}_{safe})),\\
)-&Dist((\mathcal{S}\emptyset, \mathcal{A}_{safe}), (\mathcal{P}\emptyset, \mathcal{A}_{safe}))
\end{aligned}
$$

By using the fact that $Pr[\mathbf{Expt}^{Rt}_{\mathcal{P}\emptyset}(\mathcal{A}_{safe})] = 0$, we get the following:

$$
\begin{aligned}
max(&|Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A}_{safe})]|,\\
&|Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A}_{safe})] - Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A})]|,\\
&|Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A})]|\\
)-&|Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A}_{safe})]|
\end{aligned}
$$

Probabilities are positive, which allows us to remove some of the absolute values. We also rewrite $|a|$ as $max(a, -a)$ in the last case, to remove all absolute values:

$$
\begin{aligned}
max(&Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A}_{safe})],\\
&Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A}_{safe})] - Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A})],\\
&Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A})] - Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A}_{safe})],\\
&Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A})]\\
)-&Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A}_{safe})]
\end{aligned}
$$

We include the subtraction in the maximum and simplify further:

$$
\begin{aligned}
max(0,\ &0 - Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A})],\\
&Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A})] - 2Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A}_{safe})],\\
&Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A})] - Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A}_{safe})])
\end{aligned}
$$

We know $Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A})] - 2Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A}_{safe})] \leq Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A})] - Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A}_{safe})]$, and therefore we can remove the right part of the inequation from the maximum. Similarly, $0 - Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A})] \leq 0$. We therefore simplify the maximum to:

$$
max(0, Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A})] - Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A}_{safe})])
$$

We want to prove that this is negligible in η and in η_d for all $T = \{V, E, f\}$ and $o, d, s, a, b \in V$. We reformulate this as $Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A})] - Pr[\mathbf{Expt}^{Rt}_{\mathcal{S}\emptyset}(\mathcal{A}_{safe})] \leq \epsilon$, with ϵ negligible in η and in η_d. Looking at the experiment, this means that

$$
Pr[\Phi(\textsc{GenRoute}(m', s, \mathcal{S}\emptyset^R_K), a, b))] - Pr[\Phi(\textsc{GenRoute}(m, s, \mathcal{S}\emptyset^R_K), a, b))] \leq \epsilon
$$

The difference between these probabilities is null unless the routes generated from attacked messages m' and the routes generated from m are different. Looking at the route generation process, the only factors influencing the oracle $\mathcal{S}\emptyset_K^R$ (defined in Figure 4) are the validity of the signature, the contents of $\mathbf{R}$, and the identity of the receiver (which cannot be modified by the attacker). Furthermore, all those solutions require $\mathcal{S}\emptyset_K^D(m') = \mathcal{S}\emptyset_K^D(m)$, as otherwise the experiment would have run as if the attacker output m.

We therefore know that the advantage is null unless the attacker either made the signature invalid, or it altered the route stored in the message and the signature is still valid. In that first case, the invalid signature forces the message to be dropped. Consequently, the generated route is equal to the empty route $[\,]$. Since $\Phi([\,], a, b) = 0$, then the probability of the experiment returning true is null, and so this strategy does not provide an higher advantage. In the other case, the attacker managed to alter the route stored in the message, while keeping the same data, and keeping the signature valid. To have a valid signature for an altered message, the attacker has either forged it, or recovered it from $\mathcal{S}\emptyset_K^G(o, d)$. Note that it cannot create a valid signature for a key it created, as that key would not be present in any node's K.

We first consider the case of the attacker trying to forge the signature. We proceed by assuming it manages to forge or guess the signature with a probability p_F when running on T', o', d', s', a', b'. This adversary $\mathcal{A}$ could also be used to build an adversary $\mathcal{A}_S$ who breaks the UF-CMA experiment with probability p_F. $\mathcal{A}_S$ needs to emulate $\mathcal{S}\emptyset$ on T', creating its own initialization on the network except for the node o', who uses the challenge's key. As $\mathcal{A}_S$ does not know the keys of o', it should use the chosen-plaintext oracle and verification oracle provided in the UF-CMA experiment to simulate its knowledge. $\mathcal{A}$ will therefore be in the right simulated context for $\mathbf{Expt}_{\mathcal{S}\emptyset}^{Rt}(\mathcal{A}, T', o', d', s', a', b')$, and will be provided a message m originating from o' (which costs $\mathcal{A}_S$ one call to its chosen-plaintext oracle). By assumption, this adversary will therefore return a message m' containing a valid forged signature with probability p_F. In the end, given an adversary $\mathcal{A}$ for $\mathbf{Expt}_{\mathcal{S}\emptyset}^{Rt}$ who makes q_G queries to $\mathcal{S}\emptyset_K^G$ and q_R queries to $\mathcal{S}\emptyset_K^R$ to forge signatures with probability p_F, we built an adversary $\mathcal{A}_S$ making $q_G + 1$ queries ($\mathcal{A}$'s queries, plus one to create the setting) to the chosen-plaintext oracle, and q_R queries to the verification oracle such that $\mathbf{Adv}_S^{UFCMA}(\mathcal{A}_S, \eta) = p_F$. As we supposed the signature scheme $\mathcal{S}$ secure, we know that p_F is negligible in η.

The attacker can also try to obtain the signature without forging it. The only source of valid signatures for $\mathcal{A}$ is $\mathcal{S}\emptyset_K^G(o, d)$, but this oracle provides packets containing random data. Therefore, given the attacker does q_G queries, the probability of obtaining a valid packet m' verifying $\mathcal{S}\emptyset_K^D(m') = \mathcal{S}\emptyset_K^D(m)$ is

$\left(\frac{2^{\eta_d}-1}{2^{\eta_d}}\right)^{q_G}$, which is negligible in η_d. Summing all the possibilities before:

$$Pr[\Phi(\text{GENROUTE}(m', s, \mathcal{S}\emptyset_K^R), a, b))] - Pr[\Phi(\text{GENROUTE}(m, s, \mathcal{S}\emptyset_K^R), a, b))]$$
$$\leq \mathbf{Adv}_S^{UFCMA}(\mathcal{A}_S, \eta) + \left(\frac{2^{\eta_d}-1}{2^{\eta_d}}\right)^{q_G}$$

Therefore, we can say that for all adversaries $\mathcal{A}$ making a polynomial number of queries to $\mathcal{S}\emptyset_K^G(o, d)$ and $\mathcal{S}\emptyset_K^R(m, v)$, $\mathbf{Adv}^{BINC}_{\mathcal{S}\emptyset,\mathcal{S}\emptyset,\mathcal{P}\emptyset}(\mathcal{A}, T, o, d, s, a, b)$ is negligible in η and in η_d.

5 Conclusion

In this paper, we have presented a notion of routing security, named *incorruptibility*. Incorruptibility is a quantitative measure, based on the ability of an attacker to influence how messages are routed. We provide a few example protocols in our modelization, and proved that some of them are indeed incorruptible. However, some protocols require a broader notion: after showing why one of them is corruptible, we propose the notion of *bounded corruptibility*, a generalization of the previous measure. This more accommodating notion allows us to prove that some routing protocols can only be influenced between given limits, which are exprimed using routing protocols. We finally provide a proof of the bounded corruptibility of one of our example routing protocols.

Perspectives: There are several ways this work could be expanded. Modeling node state changes is possible, but this would require more complex proof techniques to obtain results given messages influence how the next ones are routed. This would be an important step towards more complex protocols.

Insider attacks on routing protocols suppose one or more nodes in the network are controlled by the attacker. To model this, our first intuition was to allow the attacker some degree of access to K, for instance $K[s]$ (as s represents the last hop before attacker alteration of m). However, this simple modification of the game does not work because of the $\mathcal{P}_K^D(d, m') \neq \mathcal{P}_K^D(d, m)$ check: if the attacker has enough access to K, most protocols get trivially broken in this model as the attacker can create a completely new message containing the data of its choice. However, this attack, based on building from scratch another packet, is not meaningful as all protocols relying on cryptography are vulnerable to it. Our goal is to ensure the attacker's m' is based on m, without blocking any legitimate alteration of the message that would change its route.

Finally, it may be interesting to consider dynamic topologies, which correspond better to what may be actually found in a network, either because of

varying wireless transmission quality, or because of an intruder actively disrupting the connections. To do this, the attacker could be able to actively choose the topology used by the network during initialization and evaluation of the challenge message.

References

1. G. Ács, L. Buttyán, and I. Vajda. Provably secure on-demand source routing in mobile ad hoc networks. *Transactions on Mobile Computing*, 5(11):1533–1546, 2006.
2. M. Bellare. A note on negligible functions. *Journal of Cryptology*, 15(4):271–284, 2002.
3. V. J. Bono. 7007 explanation and apology. *Appears in NANOG mailing list*, 1997.
4. M. Burrows, M. Abadi, and R. M. Needham. A logic of authentication. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 426(1871):233–271, 1989.
5. L. Buttyán and T. V. Thong. Formal verification of secure ad-hoc network routing protocols using deductive model-checking. In *Wireless and Mobile Networking Conference (WMNC'10)*, pages 1–6. IEEE, 2010.
6. S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988.
7. R. Hiran, N. Carlsson, and P. Gill. Characterizing large-scale routing anomalies: A case study of the China Telecom incident. In *Passive and Active Measurement*, pages 229–238. Springer, 2013.
8. C. Karlof and D. Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. *Ad hoc networks*, 1(2-3):293–315, 2003.
9. S. T. Kent. Securing the border gateway protocol: A status update. In *Communications and Multimedia Security. Advanced Techniques for Network and Data Protection*, pages 40–53. Springer, 2003.
10. K. Lougheed and Y. Rekhter. Border Gateway Protocol (BGP). RFC 1105 (Experimental), June 1989. Obsoleted by RFC 1163.
11. J. Marshall. An analysis of SRP for mobile ad hoc networks. In *International Multiconference in Computer Science (IMECS'02)*, pages 18–21, 2002.
12. S. Murphy. BGP Security Vulnerabilities Analysis. RFC 4272, January 2006.
13. O. Nordström and C. Dovrolis. Beware of BGP attacks. *ACM SIGCOMM Computer Communication Review*, 34(2):1–8, 2004.
14. P. Papadimitratos and Z. J. Haas. Secure routing for mobile ad hoc networks. In *Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS'02)*, pages 193–204, 2002.
15. C. Tseng, P. Balasubramanyam, C. Ko, R. Limprasittiporn, J. Rowe, and K. Levitt. A specification-based intrusion detection system for AODV. In *Workshop on Security of Ad hoc and Sensor Networks (SASN'03)*, pages 125–134. ACM, 2003.
16. C. Tseng, T. Song, P. Balasubramanyam, C. Ko, and K. Levitt. A specification-based intrusion detection model for OLSR. In *Recent Advances in Intrusion Detection (RAID'06)*, pages 330–350. Springer, 2006.
17. B. Wu, J. Chen, J. Wu, and M. Cardei. A survey of attacks and countermeasures in mobile ad hoc networks. In *Wireless Network Security*, pages 103–135. Springer, 2007.