cryptography

Article

ICRP: Internet-Friendly Cryptographic Relay-Detection Protocol

Ghada Arfaoui [1], Gildas Avoine [2], Olivier Gimenez [2,3,*] and Jacques Traoré [3]

[1] Orange Labs, 4 Rue du Clos Courtel, 35510 Cesson-Sévigné, France
[2] INSA Rennes, Univiversity Rennes, IRISA, CNRS, 35042 Rennes, France
[3] Orange Labs, 42 Rue des Coutures, 14000 Caen, France
* Correspondence: olivier.gimenez35@gmail.com

Abstract: Traffic hijacking over the Internet has become a frequent incident over the last two decades. From minor inconveniences for casual users to massive leaks of governmental data, these events cover an unprecedently large spectrum. Many hijackings are presumed to be caused by unfortunate routing mistakes, but a well-organized attacker could set up a long-term stealthy relay, accessing critical traffic metadata, despite suitable encryption schemes. While many studies focus on the mitigation of known attacks, we choose to design a complete detection method regardless of the attacker's strategy. We propose a two-party cryptographic protocol for detecting traffic hijacking over the Internet. Our proposal relies on a distance-bounding mechanism that performs interactive authentication with a "Challenge–Response" exchange, and measures the round-trip time of packets to decide whether an attack is ongoing. Our construction is supported by worldwide experiments on communication time between multiple nodes, allowing us to both demonstrate its applicability and evaluate its performance. Over the course of this paper, we demonstrate our protocol to be *efficient*—itrequires only two cryptographic operations per execution inducing negligible workload for users and very few loss of throughput, *scalable*—no software updates are required for intermediate network nodes, *routing protocol independent*—this means that any future update of the route selection process will not induce changes on our scheme, and *network friendly*—the added volume of transiting data is only about 1.5%.

Keywords: hijacking; distance-bounding; protocol

check for updates

Citation: Arfaoui, G.; Avoine G.; Gimenez O.; Traoré J. ICRP: Internet-Friendly Cryptographic Relay-Detection Protocol. *Cryptography* **2022**, *6*, 52. https://doi.org/10.3390/cryptography6040052

Academic Editor: Josef Pieprzyk

Received: 6 September 2022
Accepted: 3 October 2022
Published: 17 October 2022

1. Introduction

The Internet is structured as an interconnection of smaller networks owned by different entities (academic, governmental, commercial, or else). These networks are called Autonomous Systems (AS) and can be identified by their peers thanks to a unique AS number (ASN). An AS owns a set of IP addresses that can be assigned to users or network equipment. Each AS handles its internal communications and has one or more gateways linked to adjacent AS. According to the report produced by the Number Resource Organization (NRO), the number of allocated ASN in December 2021 was around 116,900 [1]. To handle such an amount of potential communication, two main protocols are in use: the Internet Protocol (IP) for the data plane and the Border Gateway Protocol (BGP) for the control plane. BGP's task is to make sure every router knows how to forward incoming packets. In a nutshell, BGP allows AS to construct routing tables by advertising the set of IP addresses they own and spreading those advertisements along with the ordered list of traversed AS so far. Hence, every AS can keep its routing table up to date. The issue with that procedure is that it fully relies on trust. There are no security features preventing an AS to advertise a set of IP addresses that it does not truly own. This kind of misleading advertisement (whether intentional or accidental) can lead to modified routing tables and is known when performed on purpose, as a prefix hijacking attack. This kind of incident has occurred frequently during the last two decades. For instance, on February 2008, Youtube became unreachable for two hours after Pakistan Telecom falsely claimed to be the better route for joining it [2]. Another striking breakdown happened on April 2010, when China Telecom advertised the wrong traffic routes: for approximately 20 minutes, no less than 15% of the Internet's traffic adopted those routes, including some traffic of the US government, military sites and commercial sites such as Yahoo! and IBM [3]. More recently, in June 2019, an AS owned by the company "Safe Host" mishandled an update on routers and started to advertise wrong routes. China Telecom echoed those claims, which ultimately led to an important amount of traffic going through China Telecom before getting to their initial destination. This situation lasted about 2 h [4].

Those incidents had an important and very noticeable impact, hence they were quickly detected. However, a smaller-scale relay may be established by a stealthy attacker over the long term, which could allow man-in-the-middle attacks and metadata gathering, or could globally impact the network in terms of latency or private data leaks. A noticeable example of stealthier kinds of attacks is presented in [5], where authors show that hijacking bitcoin messages in order to delay block propagation can entail financial losses. Aside from those practical issues, a successful international relay attack would also damage the geopolitical reputation of the victim's country.

Since July 1994, when BGP-4 was first described [6], many proposals tried to enhance its security [7–10]. All these contributions aimed to strengthen BGP by working on the possibility to authenticate and authorize BGP updates between AS. The Internet Engineering Task Force (IETF) initiated the BGPSec standardization project [11] based on Secure-BGP [7]. The key idea is to use the RPKI public-key infrastructure to certify AS signatures. Hence, a BGPSec update will contain the reachable set of IP addresses along with the list of all the AS that received the update where all participating AS signed their own pre-path. Other attempts using techniques for anomaly detection, localization, and mitigation are to be found with different levels of efficiency [12–14], searching for strategies such as alternative routes creation, or hijacked BGP routes announcement analyzes. According to [15], these attempts target only specific subproblems and do not provide a complete detection.

On a more global scale, Path Aware Networking (PAN) is emerging as a novel way of thinking routing architecture allowing more accurate knowledge on the path traveled by data [16–21]. An important goal for those architectures is to achieve precise and, above all, trustworthy path tracking of the traversed routers during the sending of a packet. While a complete redesign of the Internet routing architecture might just be the perfect long-term solution, there is still a long way before a worldwide suitable design makes it to standardization.

Our Contribution.

1. We approach the detection of relay attacks by using time measurement. To the best of our knowledge, this has never been achieved before in the context of Internet communications.
2. We analyze the time stability between two communicating nodes by running intercontinental experiments over 5 months.
3. We propose ICRP: a two-party cryptographic protocol performing simultaneously the sending of messages, the measurements of the timings, the authentication of the receiver and the decision about the legitimacy of the route. The decision process uses a so-called decision function, taking as inputs a sample of measures captured on the fly. The function checks if the sample matches the "expected behavior" between the nodes, and outputs a Boolean (1 if the sample is suspicious, i.e., the traffic might be hijacked, 0 otherwise).
4. We implement a prototype to test the performances of our protocol for a large amount of data exchanges (Up to 200 Mo).

The remainder of the paper is structured as follows.

In Section 2, we analyze time stability over Internet communication and show that this stability is achieved for UDP communications between terminals from different locations, even for intercontinental exchanges.

In Section 3, we introduce the function deciding whether a given exchange is suspicious. This decision function outputs a Boolean (1 for suspicious and 0 otherwise). It takes as inputs a freshly collected sample and a so-called "reference sample". The reference sample represents the "expected behavior" between the nodes and is constructed during a learning phase prior to the first execution of the protocol. We then test the decision function efficiency by observing the false positive and false negative rates over a large group of both genuine communications and relay simulations.

In Section 4, we first describe the Distance-Bounding Protocols. They are used in short range contactless communications for two-party authentication [22–26]. Those protocols achieve authentication while ensuring an upper bound on the distance separating the parties, which highly complexifies the possibilities for relay attacks. Secondly, we present our protocol ICRP which translates the idea of Distance-Bounding in the context of Internet communications.

In Section 5, we describe our prototype implementation and evaluate the overhead induced by our solution in terms of latency, computational complexity, and packet size.

We believe our approach to be innovative and realistic for practical applications, and so, in section 6, we present an illustrative example.

2. Internet Latency

The protocol we introduce in this paper strongly relies on Internet latency and its stability over time. We consequently describe, in this section, the experiments we performed to measure this latency and evaluate how much it is impacted by a traffic hijacking attack.

2.1. Time Measurement

We distinguish two methods for measuring the transit time between two machines. The One Way Transit Time (OWTT) represents the time measured between the sending of a packet and the arrival to the destination. This approach attempts to capture the real-time separating two endpoints but demands a precise clock synchronization of those points and sending the timestamp along with the packet.

The Round Trip Time (RTT) is measuring the time between the sending and the reception of a response. As this is a one-sided measure, there is no need for clock synchronization. The approximation $OWTT = \frac{RTT}{2}$ is often made, but there is no insurance that the transit times in both directions are comparable. It is then preferable to consider RTT as a stand-alone metric rather than a way to measure OWTT.

In this paper, we adopt the RTT metric. Using the OWTT metric constrains one of the party members of our protocol to send its timestamp data to the other party for a travel time to be computed. This has at least two clear downsides: (1) it raises the overall quantity of data to be sent, and (2) it may become a breach for an attack aiming to falsify the measures.

2.2. Experimental Setup

We measured RTTs for UDP traffic between two parties, $\mathcal{S}$ the Sender and $\mathcal{R}$ the Receiver, sometimes relayed by $\mathcal{A}$ the Attacker. We define below the key points of our experiments.

- *Locations.* We use four nodes located in different countries for our experiments:
  - France
  - Germany
  - Poland
  - USA, Oregon
- *Hijacked traffic.* We ran our experiments on the Internet, hence we had no control over the route between $\mathcal{S}$ and $\mathcal{R}$. For this reason, we simulated the presence of a relay by sending directly the packets from $\mathcal{S}$ to $\mathcal{A}$ and then from $\mathcal{A}$ to $\mathcal{R}$.
- *Packet size.* The impact of packet length on RTT is very weak for realistic variations [27]. Hence, we arbitrarily chose to use 512-bytes packets across all our experiments.

2.3. *RTT Measurements without Adversary*

2.3.1. Stability Over a Short Period

We present the result of short-period (i.e., a few minutes) experiments in Figures 1–3. Figure 1 shows 6 graphs, in which each "+" represents the value of one RTT in milliseconds (readable in the *y*-axis). Each graph is a plot of 7000 RTTs between two end-points collected in a row. The dates and times of the start and end of the measurements are given on each individual graph. Figures 2 and 3 display the same samples on a more zoomed-in scale along with their statistic distribution.

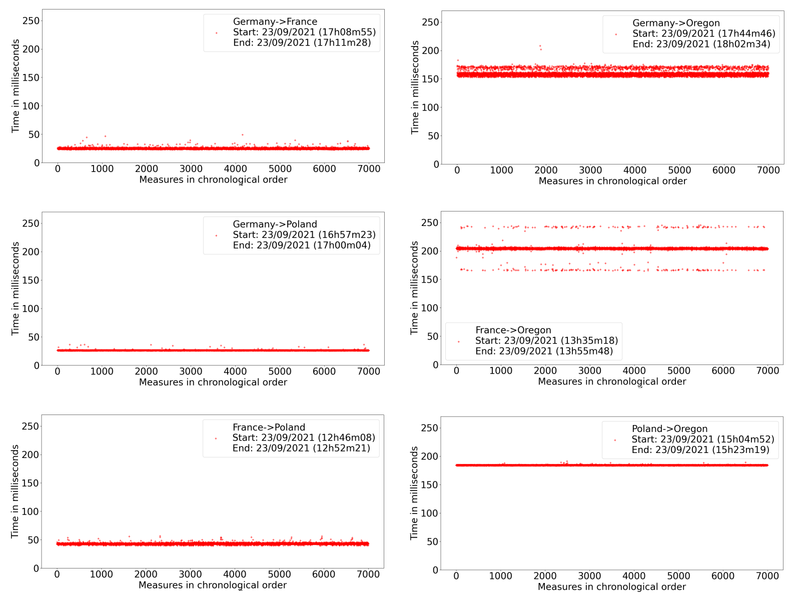

Figure 1. RTTs for 7000 512-bytes packets between various locations.

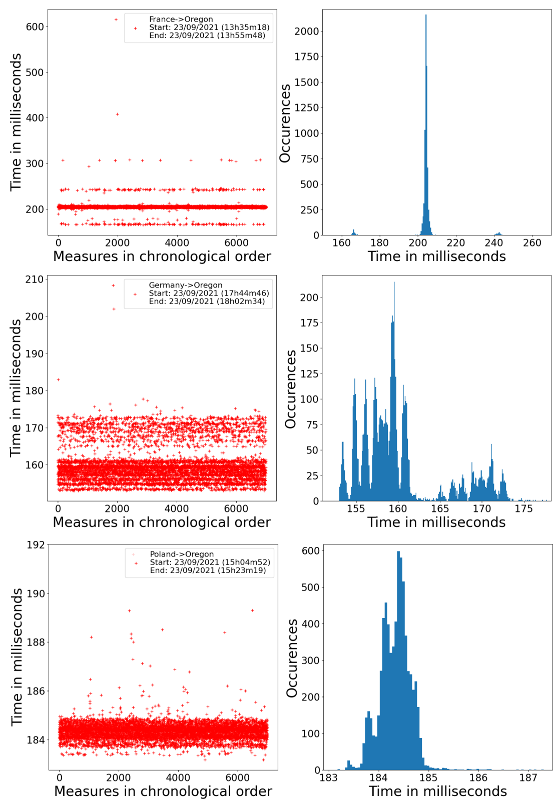

Figure 2. Focus on RTTs from Europe to the USA with their distributions.

Regarding the stability of the measurements, the distributions show that the majority of the measures are concentrated in one dense interval. Depending on the sources and destinations, the samples appear in different shapes. Noticeably, samples from Germany-Oregon or France-Oregon seem to be formed of several layers. When this is the case, it seems that one layer always outstands the others. Indeed, for the sample gathered from France to Oregon, 93% of its measures are in $[200, 210]$ and for Germany to Oregon, 81% are in $[150, 164]$.

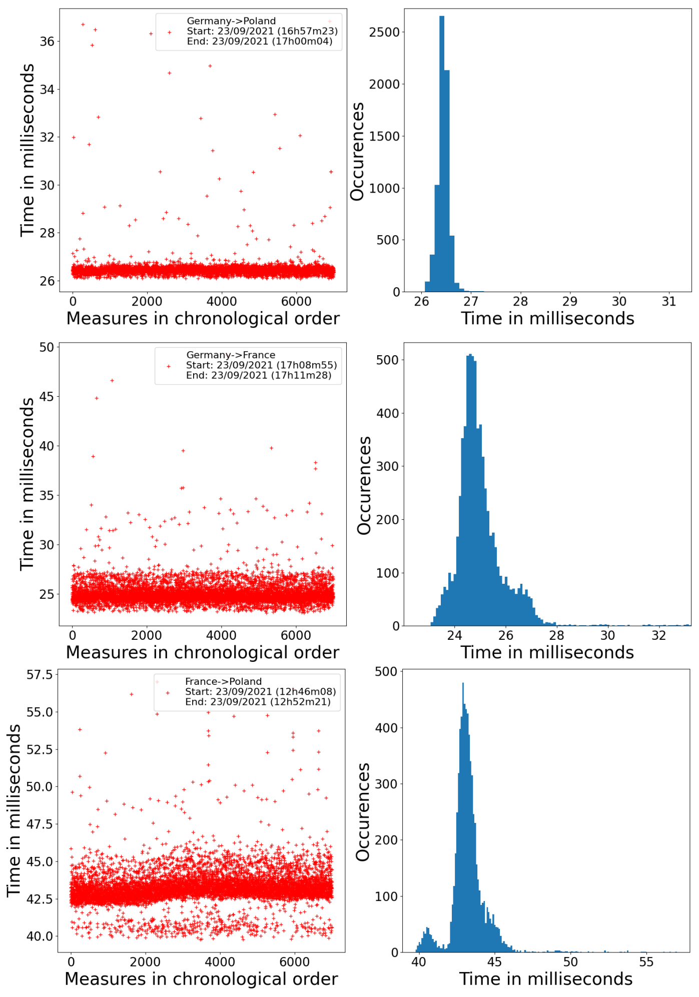

Figure 3. Focus on RTTs from Europe to Europe with their distributions.

2.3.2. Stability over a Long Period

In this paper, we decided to check the RTT stability over long periods (a few months) for two main reasons. We firstly want to validate our time-based approach, given that this kind of method has never been used for Internet relay detection before. Secondly, as our decision function uses a reference sample to test fresh samples, we need to know if this reference remains representative over time or if it should be regularly updated.

In this section, we display long-term measurements. During a full month, we gathered 1000 RTTs per hour between two nodes (Poland and Oregon) to observe the overall evolution. Figure 4 shows that long-term stability is achieved over this period. However, comparing this very large sample with older measures on Figure 5 also shows a slight modification of about 3ms. Going further on this analysis, we observed several samples collected between early September and mid-January. We see in Figure 6 two graphs, the top one showing the means in milliseconds of those samples, the days on which they were collected are readable in the x-axis with their respective sizes (between parenthesis). The bottom one shows the same sample on a more zoomed-in scale. Figure 6 shows that the stability of the measures is susceptible to evolve for the order of magnitude of the milliseconds. This same result is noticeable as well for samples between Germany and Oregon. Those variations remain small in comparison with the impact caused by a relay on the path (see Section 2.4).

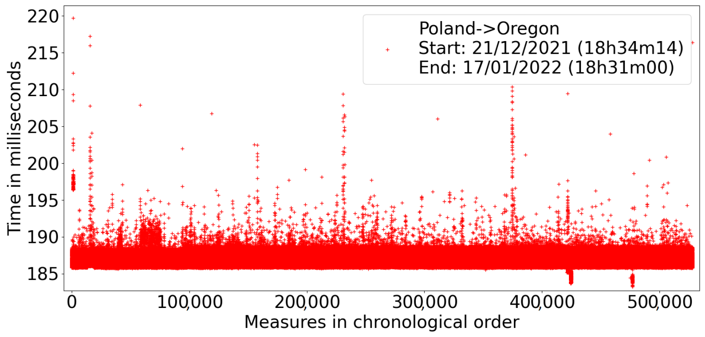

Figure 4. Long term continuous experiments between Poland and Oregon.

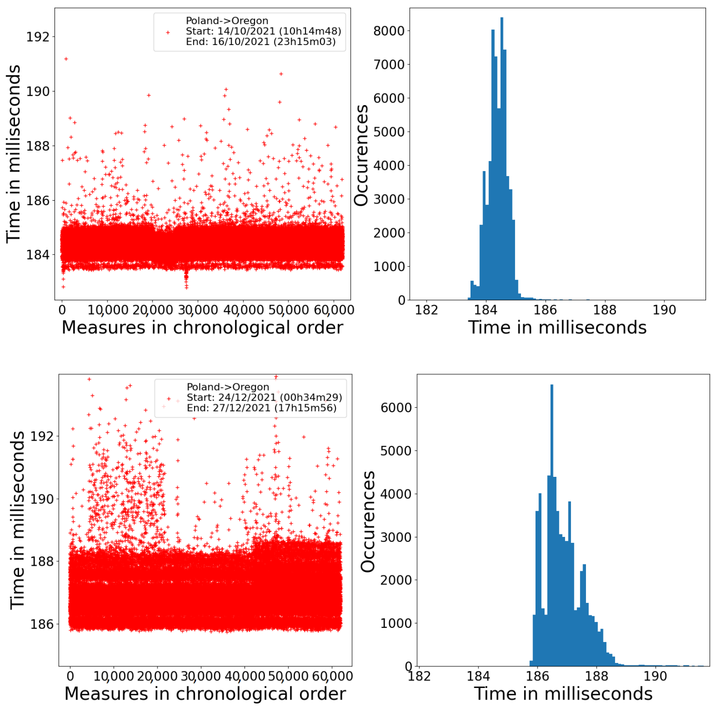

Figure 5. Two samples and their distribution between Poland and Oregon, 2 months apart.

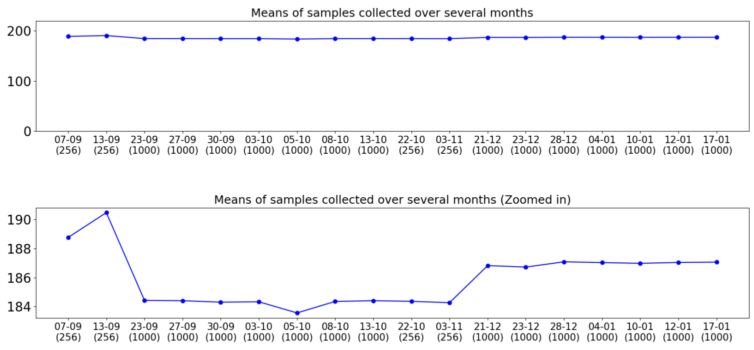

Figure 6. Experiments between Poland and Oregon over 4 months.

2.4. RTT Measurements with Adversary

Figure 7 shows the impact of a relay over the Round Trip Time for exchanges between the node in Poland and the one in Oregon. We display an alternance of standard communications and relayed communications going through the node in France.

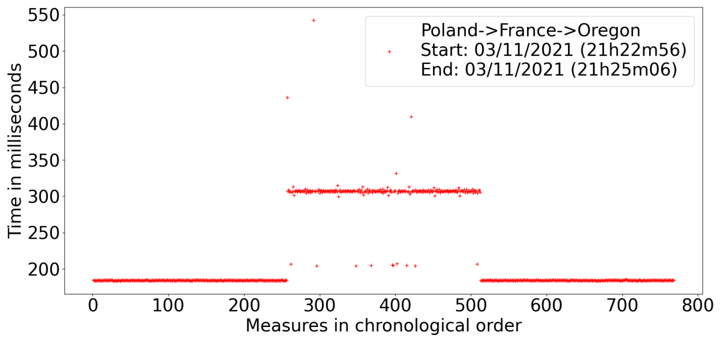

Figure 7. RTTs for 7000 packets of size 512 bytes between Germany and Oregon through Poland.

It appears that the relay creates a drastic impact on the measured time. Indeed, the RTTs get increased by more than 150 ms.

For this specific route, the impact on the time caused by the relay is more than enough to efficiently distinguish between a genuine route and a relayed one.

The impact of a relay may be caused by many factors, such as: the number of traversed routers, the location of the attacker, his proximity to a genuine route, his control over some network equipment, and so forth. This means that there exists one or multiple optimum setups, lowering the impact of a relay to a minimum. With that information in mind, we choose to define our decision process so that it can modify its detection sensibility. By doing that, we provide users a dynamic capacity to face adversaries even in very efficient setups, see Section 3.

3. Decision Function

We define a decision function noted Verify_Time.

$$\boxed{\mathsf{Verify\_Time}_{ref}: \begin{array}{ccc} \mathbb{R}_+^n & \to & \{0,1\} \\ samp & \mapsto & b \end{array}}$$

This function takes as input a fresh sample ($samp$) of size n and returns a bit $b = 0$ (i.e., accepted) or $b = 1$ (i.e., rejected). Verify_Time uses a parameter called ref which is a trusted sample of RTT. The sample ref can be seen as a fingerprint of the expected behavior of time between two nodes and, as seen in Section 2, this reference sample is not subject to great changes over long periods of time.

We tested Verify_Time on numerous samples, some of which are genuine communication between a sender $\mathcal{S}$ and a receiver $\mathcal{R}$, and the rest issued from a relay simulation where $\mathcal{S}$ sends its packets to an intermediary node $\mathcal{A}$ which relay them to $\mathcal{R}$.

3.1. Definitions

In this section, we define the keywords, concepts and ideas that will be used throughout this paper.

3.1.1. Reference Sample

The reference sample ref consists of a large set of measures gathered in advance during a learning phase performed between $\mathcal{R}$ and $\mathcal{S}$. It represents the standard values we can expect when measuring RTTs between $\mathcal{R}$ and $\mathcal{S}$. It is worth noting that the learning phase should take place when there is no ongoing attack, that is, when the route taken by the packets during the measurements has not been altered by a malicious party.

The reference sample should be updated when the genuine RTTs deviate from their reference due, for example, to modifications in the network topology. The experiments presented in Section 2, Figure 5, show that such a modification may occur, but does not cause a drastic change in the measures in comparison with the impact of a relay.

In environments where RTTs are not stable, one can consider performing dynamic updates of the reference sample to improve the reliability of the protocol. For example, any new valid execution of the protocol provides 256 fresh RTTs that can be concatenated to ref while the 256 oldest ones can be removed from ref. Automatic updates should be monitored, though, as they may allow poisoning attacks on the reference sample.

3.1.2. Terminology

Verify_Time outputs a binary response: 0 if the tested sample is considered genuine, 1 otherwise. Throughout Section 3.2, we challenge Verify_Time with genuine and relayed samples and analyze its efficiency using the following terminology:

- False positive: Verify_Time outputs 1 to a genuine sample
- False negative: Verify_Time outputs 0 to a relayed sample

3.2. Description and Efficiency

Given the stability of the samples, we choose to use a positional decision process. Our decision function selects a threshold t depending on the reference sample ref it uses. This threshold is a time limit allowing at most a given proportion p of the fresh sample $samp$ to be above. So typically, the threshold should be around the $(1-p)$th percentile of ref. The decision function accepts $samp$ if this upper bound on the proportion is fulfilled, and rejects it otherwise (see Algorithm 1).

Algorithm 1 Verify_Time pseudocode

```
  **Input** $t, samp, p$
  **Output** 0 or 1
$counter = 0$
**for** $i$ in $samp$ **do**
    **if** $i \geq t$ **then**
        $counter = counter + 1$
    **end if**
**end for**
$prop = \frac{counter}{len(samp)}$
**if** $prop \geq p$ **then**
    **return** 1
**else**
    **return** 0
**end if**
```

3.2.1. False Negatives and False Positives

We show in Figure 8 (in Figure 9, respectively) the false positive and negative rates we obtained in relation to the threshold value for communications between Germany and Oregon (Poland and Oregon, respectively). Those tests have been performed on more than 500 samples gathered over several months. As we saw in Section 2.4, the measurements of our hijacking simulation created such a time gap that this decision function is strong enough to achieve absolute detection, Figures 8 and 9 both highlight this by having a very large interval of possible threshold values leading to no false positive nor false negative.

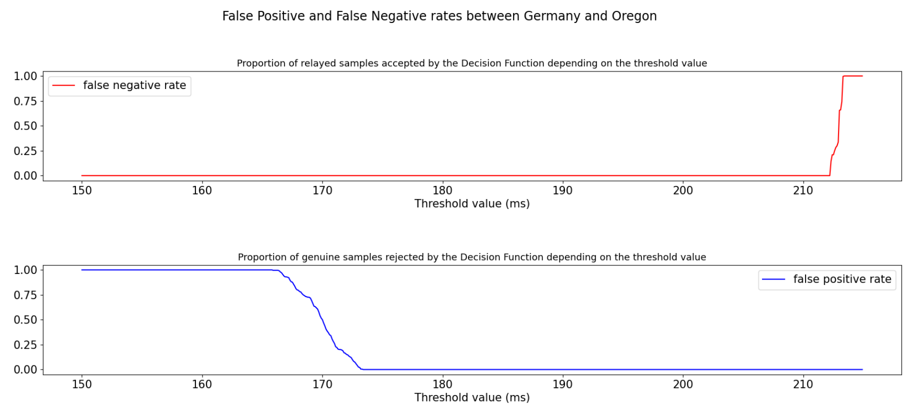

Figure 8. Evolution of false positives and negatives for exchanges between Germany and Oregon.

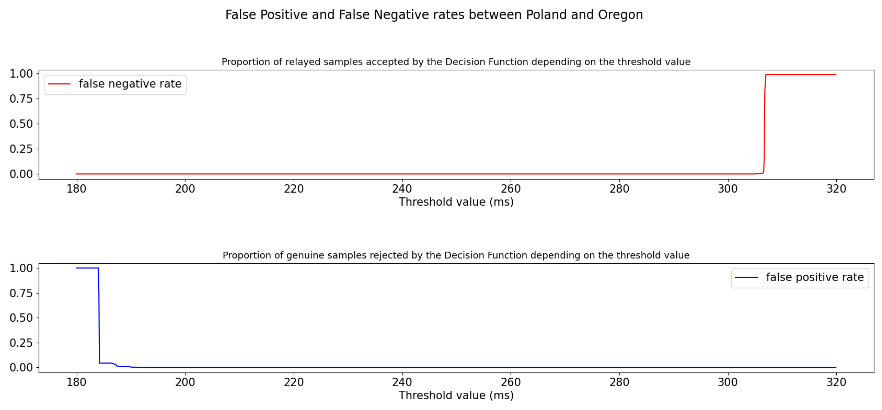

Figure 9. Evolution of false positives and negatives for exchanges between Poland and Oregon.

Note that those graphs may change depending on the capacity and positional setup of the intermediary. Indeed, it is expected that the number of additional routing equipment visited during the relay is highly related to its efficiency. This would be represented by a false negative rate growing closer to 1 for lesser threshold values. Choosing a suited threshold then becomes a matter of appreciation of how efficient an attacker can get.

3.2.2. Choosing the Threshold

As we stated in Section 2.4, the efficiency of an attack may depend on many factors such as the attacker's connection speed, the current network topology and probably other factors. To detect an attacker disposing of an optimal relay setup, we should set the decision function to the highest sensitivity that can be supported. This is achieved by letting the threshold be as low as possible while keeping some breathing room to avoid the maximum of false positives as well. From Figure 9, we see that the minimum threshold value getting 0% of false positive on our tests lies around 192 ms. Nevertheless, we observed on Section 2.3.2 that small variations on the samples might emerge over long periods of times (Figures 5 and 6). Hence, users can choose to slightly loosen the sensibility of the decision function with a higher threshold.

Allowing the samples to live about 5 to 10 ms higher than normally expected trades off the insurance of very few false positives against the possibility of an attack, assuming that such an efficient relay is achievable between those nodes.

4. Protocol

In this section, we elaborate on our overall protocol. Section 4.1 describes the concept of Distance-Bounding protocols for Radio Frequency communications. Our idea of using time measurements is highly inspired by this context, even though we will discuss the strong differences to take into account when transposing from RFID to the Internet. Then, Section 4.2 describes our proposal in which the decision function (previously describe) is considered as a black box ideal primitive.

4.1. Distance-Bounding Protocols

To the best of our knowledge, the strongest example of countermeasure to relay attacks using time measurements are the distance-bounding protocols, also known as proximity checks. They have been massively studied [26] in the context of Radio Frequency IDentification (RFID), and they are implemented in some contactless smartcards, e.g., Mifare Plus [28] and Mifare DESfire [29]. Given that the signal propagation cannot be faster than the speed of light, a verifier considers that there is no relaying adversary if the RTTs between the verifier and the prover are below a given upper bound. A well-known relay attack is the mafia fraud introduced by Desmedt, Goutier and Bengio in 1987 [30], and applied to the Fiat–Shamir Zero-Knowledge Argument of Knowledge [31]. This protocol is based on the complexity of the quadratic residuosity problem and allows an interactive authentication of a prover $\mathcal{P}$ to a verifier $\mathcal{V}$.

The attack name comes from Shamir's claim that the Fiat–Shamir protocol remains secure even in a scenario where the prover is a mafia-owned store, which is contradicted by [30]. The mafia fraud actually allows the attacker to get authenticated by simply relaying the exchange between the genuine prover and the verifying device. Such an attack especially makes sense in contactless authentication that needs the prover (card, transit pass, or else) to be in the proximity of the verifying device.

Brands and Chaum introduced in 1994 [22] a countermeasure to this fraud. They indeed added to the Fiat–Shamir protocol a feature to bound the distance from which is standing the genuine prover and to dismiss the authentication if it concludes that the prover is standing further than a given distance. This countermeasure is a so-called distance-bounding protocol, and it uses a series of rapid bit-exchanges to measure the round trip time between the prover and the verifier, and so the distance, using the speed of light as an upper bound. We illustrate the protocol of Brands and Chaum in Figure 10. In this protocol, $\mathcal{P}$ proves to $\mathcal{V}$ that he knows x such that $x^2 = X \bmod n$ in three steps:

- *Initialization*: $\mathcal{P}$ picks k nonces r_i, computes their squares $R_i = r_i^2 \mod n$, then picks k random bits $c_i^{\mathcal{P}}$. He then sends the R_i's and a commitment (typically a hash) of the $c_i^{\mathcal{P}}$s. The prover $\mathcal{V}$ then also computes k random bits $c_i^{\mathcal{V}}$.
- *Fast bit-exchange*: for $i = 1 \ldots k$, $\mathcal{V}$ creates a timestamp t_i, sends $c_i^{\mathcal{V}}$, receives the responses $c_i^{\mathcal{P}}$ and immediately creates another timestamp t_i', and stores $(t_i' - t_i)$.
- *Verification*: $\mathcal{P}$ computes all the $c_i = c_i^{\mathcal{P}} \oplus c_i^{\mathcal{V}}$, $z_i = r_i x^{c_i} \mod n$ and sends z_i to $\mathcal{V}$. The latter checks (i) if the committed $c_i^{\mathcal{P}}s$ in the initialization phase are the same as those he received in the fast bit-exchange phase (typically by recomputing the commit$(c^{\mathcal{P}})$, (ii) computes the c_i similarly, (iii) checks if z_i^2 is equal to $R_i X^{c_i}$ and (iv) checks if $\max(\{t_i' - t_i\})$ is below a given upper bound.

Brands and Chaum's seminal work paved the way for many other distance-bounding protocols. One could for example cite Hancke and Kuhn's protocol [23] that uses only symmetric-key cryptography. Although describing the body of literature related to distance-bounding protocols is out of the scope of this article, interested readers will find a complete analysis of distance-bounding protocols in [26]. It is worth noting that these protocols are well suited for RFID authentication because communications are end-to-end (from the physical layer perspective) and the computations performed by the RFID tag are lightweight, which implies that the RTTs are very stable. However, it is important to raise that distance-bounding protocols do not actually detect relays: they detect abnormally long RTTs, and that there is a risk of a relay attack.

The key difference between Internet communications and RF communications is that the former involves physical relays, namely routers, and routes dynamically evolve over time. In spite of that, RTTs are pretty stable, as shown previously in this article.

Consequently, instead of comparing RTTs with a reference time bound as performed with RF communications, our protocol compares RTTs with a reference profile defined during a learning phase.

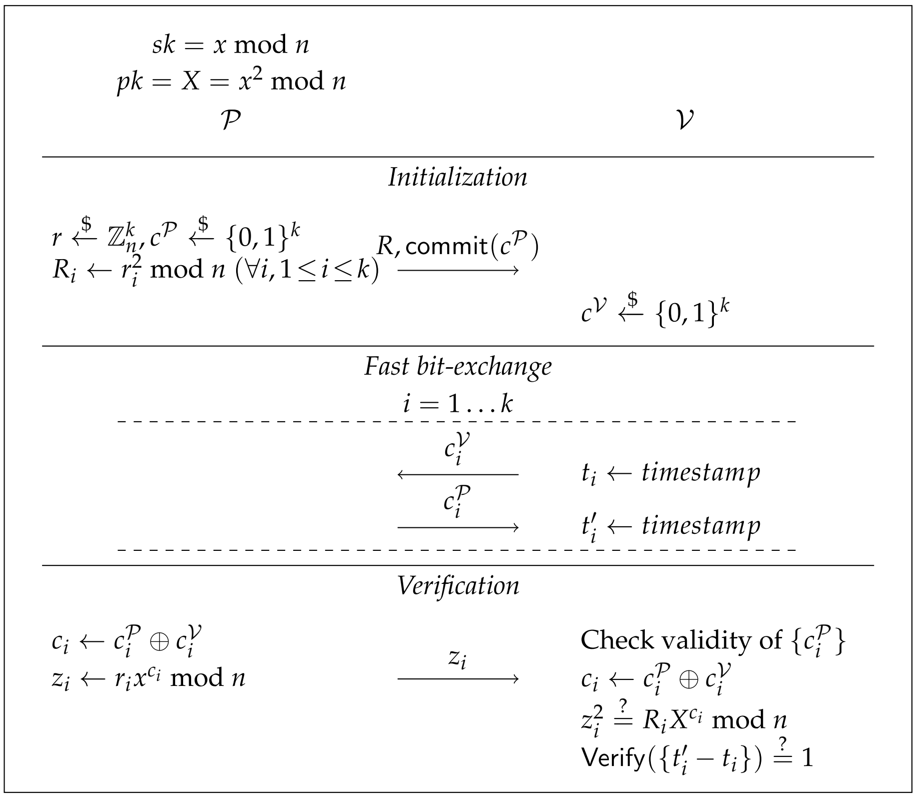
$sk = x \bmod n$
$pk = X = x^2 \bmod n$
$\mathcal{P}$
$\mathcal{V}$
Initialization
$r \xleftarrow{\$} \mathbb{Z}_n^k, c^{\mathcal{P}} \xleftarrow{\$} \{0,1\}^k$
$R_i \leftarrow r_i^2 \bmod n \ (\forall i, 1 \le i \le k)$
$R, \mathsf{commit}(c^{\mathcal{P}})$
$c^{\mathcal{V}} \xleftarrow{\$} \{0,1\}^k$
Fast bit-exchange
$i = 1 \ldots k$
$c_i^{\mathcal{V}}$
$t_i \leftarrow timestamp$
$c_i^{\mathcal{P}}$
$t_i' \leftarrow timestamp$
Verification
$c_i \leftarrow c_i^{\mathcal{P}} \oplus c_i^{\mathcal{V}}$
$z_i \leftarrow r_i x^{c_i} \bmod n$
z_i
Check validity of $\{c_i^{\mathcal{P}}\}$
$c_i \leftarrow c_i^{\mathcal{P}} \oplus c_i^{\mathcal{V}}$
$z_i^2 \overset{?}{=} R_i X^{c_i} \bmod n$
$\mathsf{Verify}(\{t_i' - t_i\}) \overset{?}{=} 1$

Figure 10. Brands and Chaum's distance-bounding protocol based on the Fiat–Shamir zero-knowledge authentication.

4.2. ICRP (Internet-friendly Cryptographic Relay-detection Protocol)

4.2.1. Description

Our protocol ICRP, described in Figure 11, runs between a sender $\mathcal{S}$ and a receiver $\mathcal{R}$. A full run of a protocol consists of n rounds and allows $\mathcal{S}$ to gather one sample of n RTTs concurrently to the sending of n packets $\{p_i\}_{i=1\ldots n}$ containing data or messages (unlike in Distance-Bounding protocols where the exchange only serves an authentication purpose). If the amount of data to be sent exceeds n packets, multiple runs of the protocol will be performed, we call each run a session of the protocol. Both parties must know the number of rounds n and, if defined, the number of sessions k prior to an execution, which makes them public parameters. When a session is initiated, $\mathcal{S}$ marks n upcoming packets p_i by a random bit s_i, creates a timestamp t_i and sends $p_i || s_i$ to $\mathcal{R}$. In each marked round, $\mathcal{R}$ responds with a random bit r_i. Upon reception of r_i, $\mathcal{S}$ creates a new timestamp t_i'. The RTT of the current round is actually the time difference $t_i' - t_i$.

Once the n rounds have been performed, $\mathcal{R}$ signs the hash of the p_i's along with the s_i's and the r_i's; this hash is denoted $H_{\mathcal{R}}$ and the signature $\sigma_{\mathcal{R}}$. Finally, $\mathcal{S}$ verifies that $\sigma_{\mathcal{R}}$ is a valid signature on $H_{\mathcal{S}}$ and let the collected sample of RTTs be analyzed by a decision function called Verify_Time (see Section 3).

Note that the scheme displayed in Figure 11 describes the case where the party performing the verification is also the party willing to send the messages. In some cases, though, it is $\mathcal{R}$ who needs to make sure that no relay was ongoing. This is achievable by swapping around the time measurements and by making $\mathcal{S}$ send to $\mathcal{R}$ some init message indicating to $\mathcal{R}$ that the protocol may start, see Figure 12.

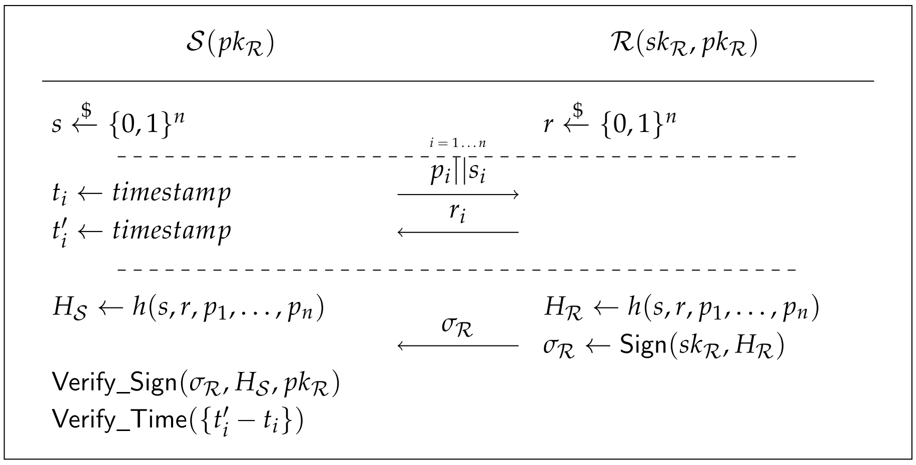
$\mathcal{S}(pk_{\mathcal{R}})$
$\mathcal{R}(sk_{\mathcal{R}}, pk_{\mathcal{R}})$
$s \xleftarrow{\$} \{0,1\}^n$
$r \xleftarrow{\$} \{0,1\}^n$
$i = 1 \dots n$
$t_i \leftarrow timestamp$
$p_i || s_i$
r_i
$t'_i \leftarrow timestamp$
$H_{\mathcal{S}} \leftarrow h(s, r, p_1, \dots, p_n)$
$H_{\mathcal{R}} \leftarrow h(s, r, p_1, \dots, p_n)$
$\sigma_{\mathcal{R}}$
$\sigma_{\mathcal{R}} \leftarrow \mathsf{Sign}(sk_{\mathcal{R}}, H_{\mathcal{R}})$
$\mathsf{Verify\_Sign}(\sigma_{\mathcal{R}}, H_{\mathcal{S}}, pk_{\mathcal{R}})$
$\mathsf{Verify\_Time}(\{t'_i - t_i\})$

Figure 11. Our protocol ICRP for traffic hijacking detection with verification performed by $\mathcal{S}$.

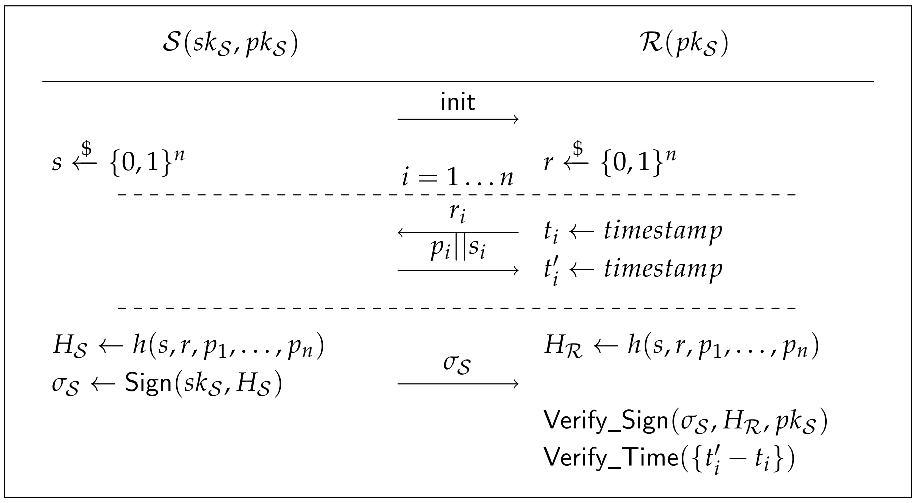
$\mathcal{S}(sk_{\mathcal{S}}, pk_{\mathcal{S}})$
$\mathcal{R}(pk_{\mathcal{S}})$
init
$s \xleftarrow{\$} \{0,1\}^n$
$i = 1 \dots n$
$r \xleftarrow{\$} \{0,1\}^n$
r_i
$t_i \leftarrow timestamp$
$p_i || s_i$
$t'_i \leftarrow timestamp$
$H_{\mathcal{S}} \leftarrow h(s, r, p_1, \dots, p_n)$
$\sigma_{\mathcal{S}}$
$H_{\mathcal{R}} \leftarrow h(s, r, p_1, \dots, p_n)$
$\sigma_{\mathcal{S}} \leftarrow \mathsf{Sign}(sk_{\mathcal{S}}, H_{\mathcal{S}})$
$\mathsf{Verify\_Sign}(\sigma_{\mathcal{S}}, H_{\mathcal{R}}, pk_{\mathcal{S}})$
$\mathsf{Verify\_Time}(\{t'_i - t_i\})$

Figure 12. Our protocol ICRP for traffic hijacking detection with verification performed by $\mathcal{R}$.

4.2.2. Active and Passive Modes

Depending on the type and amount of data to be sent, we can distinguish two ways to use this protocol:

1. Passive mode: we call Passive mode a long-term background use of the protocol. This mode is relevant in cases where $\mathcal{S}$ and $\mathcal{R}$ frequently exchange small amounts of data. In this case, a complete session of the protocol might be achieved over multiple exchanges. The protocol then passively keeps track of the overall number of sent packets and performs the verification when n packets have been sent.
2. Active mode: we call Active mode the execution of the protocol used specifically over the sending of one or several large files. This case is relevant when $\mathcal{S}$ and $\mathcal{R}$ exchange larger amounts of data on a more spread-out frequency. In this case, multiple sessions of the protocol might be achieved over a single fast-stream flow.

Figure 13 illustrates how and when ICRP runs in both modes. It shows that, in Active mode, the verification phase must be performed concurrently with the measuring phase for performance's sake. This is not specifically required for Passive mode.

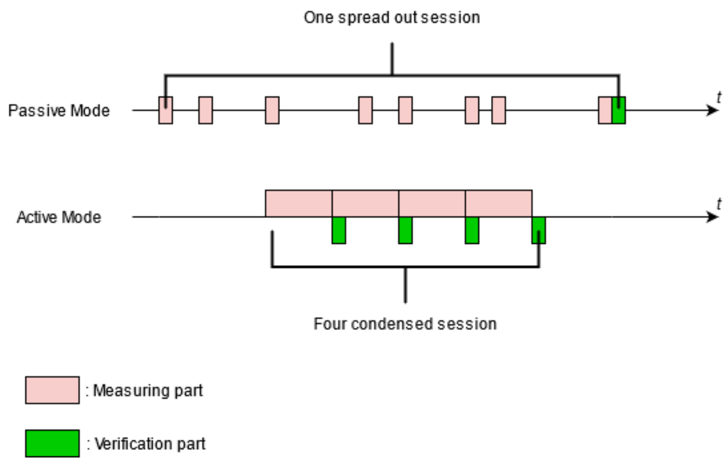

Figure 13. Active and Passive mode overview.

5. Performances

In this section, we evaluate ICRP performances. Firstly, we supply an overview of our prototype implementation by explaining the problem with the sequential representation given in Figures 11 and 12. Then, we analyze the results based on this implementation regarding 3 main points of attention: (1) the computational capabilities for the cryptographic operations (hashes, signature, verification and decision function); (2) the throughput capabilities in comparison with direct sending of UDP packets; and (3) the data overhead added from a classical sending.

5.1. Prototype Implementation

Our proposal as displayed by Figures 11 and 12 has a very clear downside, which is its sequentiality. Indeed, with this representation, each time $\mathcal{S}$ sends a packet, he has to wait for a response before sending the next one. This is especially problematic when ICRP runs in active mode and aims to send many consecutive packets. Hence, $\mathcal{S}$ needs to concurrently perform the sending and the reception of $\mathcal{R}$'s responses.

Similarly, if $\mathcal{S}$ and $\mathcal{R}$ need to run multiple consecutive sessions in active mode, the verification and authentication part of the protocol must not be realized sequentially as it would force $\mathcal{S}$ to wait until the end of the session to start a new one. Figure 14 schematically shows the differences in terms of efficiency between 3 simplified models of implementation for $\mathcal{S}$'s side: the top one is the sequential implementation, the middle one displays 2 concurrent threads for the sending and the reception of acknowledgments, the bottom one displays 3 concurrent threads for sending, receiving and verifying. The dotted lines represent a repeated operation, while the solid lines represent inactive periods of time for the current thread.

Our prototype is implemented according to the bottom model of Figure 14. The third part handling verification is separated into three threads for synchronization purposes. The other party $\mathcal{R}$ is also implemented concurrently, with one thread handling the reception of packets and the sending of the response, and a second thread performing the cryptographic computations. We provide below a description of each thread's actions.

On $\mathcal{S}$'s side:

1. Thread send: this thread is in charge of sending all the packets to $\mathcal{R}$ and generating a timestamp when it does. It then stores the timestamp in a structure shared by all threads.
2. Thread recv: this thread is in charge of receiving every response from $\mathcal{R}$, generating a timestamp when it does, and computing the RTTs from the timestamps placed in the shared structure.
3. Thread pre_Hash: this thread is in charge of updating the Hash context with the values known beforehand by $\mathcal{S}$. That is the content of packets $\{p_i\}$ and the bits s_i.

4. Thread final_Hash: this thread is in charge of updating the Hash context with the values received from $\mathcal{R}$. That is the bits r_i.
5. Thread verif: this thread is in charge of receiving $\mathcal{R}$'s signature and waits for all the data it needs to be available from other Threads. It then proceeds to check the signature and applies the decision function on the RTTs for the current session.

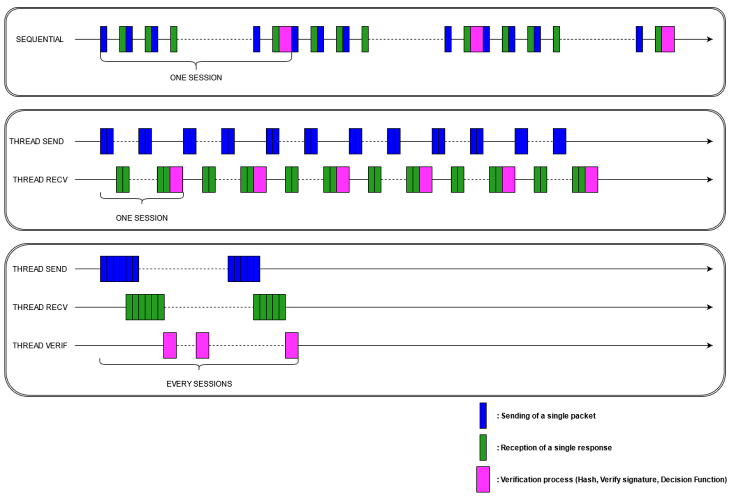
SEQUENTIAL
ONE SESSION
THREAD SEND
THREAD RECV
ONE SESSION
THREAD SEND
THREAD RECV
THREAD VERIF
EVERY SESSIONS
: Sending of a single packet
: Reception of a single response
: Verification process (Hash, Verify signature, Decision Function)

Figure 14. Execution in active mode using 3 different implementations ($\mathcal{S}$'s side).

On $\mathcal{R}$'s side:

1. Thread recv: this thread is in charge of receiving the packets and sending the responses to $\mathcal{S}$.
2. Thread auth: this thread is in charge of updating the Hash context with the values known beforehand by $\mathcal{R}$. That is the bits r_i. It then waits for all the data it needs to be available from other threads (the content of packets $\{p_i\}$ and the bits s_i), proceeds to Hash&Sign, and finally, sends the signature to $\mathcal{S}$.

Using multiple threads to boost up the performances forces the parties to tag each packet with a sequence number in order to link each message and response to the correct round and session and not get confused with the time measurements. Consequently, we added a 4 bytes header to indicate the sequence number. These 4 bytes are formed by 2 bytes indicating the current session number followed by 2 bytes indicating the current round number. Writing the round and session indexes on 2 bytes each is good enough to fit our experimental needs while being easily implemented in our prototype. In a real case scenario though, the number of rounds n for an ICRP session should not be greater than 512 because a relay detection can only occur once the n rounds are over. If n is at most 512, its associated field in the 4 bytes header can be limited to only 9 bits, leaving 23 bits for the session index. We choose to use a 4 bytes header as it is also the size used for the field SQN of the TCP header which serves the same purpose of keeping the sessions synchronized between the nodes. This TCP field is reset once reached the maximum value of 2^{32}, which is high enough to ensure not having two packets with the same SQN transiting at the same time.

5.2. Analysis

Our protocol has three main parameters:

- k: the number of sessions to execute.
- n: the number of rounds per session. It also defines the size of a collected sample and the number of packets sent during one session. We assume n to be constant from one session to the next.
- p: the size of each packet in bytes. We also assume that p remains constant over rounds and sessions.

We call a (n, p, k)-sending, the sending with our prototype of $n * p * k$ bytes of data through k sessions of n rounds with constant p-bytes packet size.

5.2.1. Complexity of the Computations

We leave to users the choice of the cryptographic primitives, as we believe that they are interchangeable in our protocol. For our experiments, we have arbitrarily chosen to use $SHA256$ as the hash function, and $RSA2048$ as the Signature algorithm. Those choices are voluntarily poor performance-wise. However, they allow us to give an upper complexity bound.

For each session, $\mathcal{R}$ ($\mathcal{S}$, respectively) performs Hash&Sign (Hash&Verify, respectively) over the packets $\{p_i\}_{0\leq i<n}$ and the random bits $\{r_i\}_{0\leq i<n}$ and $\{s_i\}_{0\leq i<n}$. This is $n \cdot (8p + 2)$ bits of data to be hashed. $SHA256$ is based on the Merkel–Damgård construction, this means that the message to hash is separated into blocks of identical size which are processed by a compression function. Hence, its complexity is linear in the number of blocks involved. $SHA256$ uses 512-bits blocks, so for each session, the Hash complexity will be $\mathcal{O}(np)$ in the number of compression function, given by the following computation: $\frac{n(8p+2)}{512} = \frac{n(4p+1)}{256}$.

Table 1 shows the number of applied compression functions and the corresponding hashing time depending on n with a fixed value of $p = 512$.

Table 1. ICRP final Hash complexity (using $SHA256$) with $p = 512$ bytes.

n	**256**	**512**	**768**	**1024**	**1280**	**1536**
n° of compression	2049	4098	6147	8196	10,245	12,294
Hashing time (ms)	0.731	1.479	2.253	2.834	3.813	4.434

Regarding the signature and the verification, the input value is always a 256 bits string, and so the time taken for this operation remains constant for both operations.

Finally, $\mathcal{S}$ runs the decision function on the sample. This process is linear in the size n of the sample as it goes through the table of RTTs and increments a counter every time the treated value is higher than the chosen threshold. Note that n should not be too large because the verification is performed for each n packet sent. Hence, a high n leaves a wider amount of data to be relayed before the detection. We believe $n = 256$ or $n = 512$ to be the most suitable choices. These values being very small, we can consider the decision complexity to be negligible.

Note that, the slower the overall verification process is, the later a suspicious sample will be detected. However, as the authentication and verification are performed concurrently with the other processes, those times do not impact the throughput performances.

5.2.2. Throughput

In this section, we test the impact of parameters n, p, and k over the sending time of large blocks of data. We then compare those times with the throughput given by the sending of raw UDP packets unsupervised by ICRP.

Impact of Parameters n and k

We see on Table 2 the times in seconds involved in an $(n, 500, k)$-sending of 20 Megabytes (resp. 100 Megabytes) of data using 3 possible values of (n, k) which are $(250, 160)$, $(320, 125)$ and $(400, 100)$ (resp. $(250, 800)$, $(320, 625)$ and $(400, 500)$). The parameter p remains constant to 500 bytes for those tests. For this configuration, we see that the number of sessions k and rounds n creates no visible impact on the overall sending time.

Table 2. ICRP sending times in seconds for 20 Mb and 100 Mb depending on n (number of rounds) and k (number of sessions).

Size (Mb)	n	k	n	k	n	k
	250	**160**	**320**	**125**	**400**	**100**
20	3.65		3.56		3.53	
Size (Mb)	n	k	n	k	n	k
	250	**800**	**320**	**625**	**400**	**500**
100	16.71		16.81		16.62	

Impact of Packet Size p

As it was stated in [27], the packet size has a low impact on the sending time of a single packet. This means that the more data contained within every single packet, the faster the sending of the overall message (containing multiple packets) will be. Figure 15 shows the evolution of the time to send 100 Megabytes of data depending on the size of the individual packets. We can observe that the time decreases for realistic values of p. The maximum size of UDP packets is implicitly specified in the official IETF documentation RFC768 [32], as the UDP header contains a field called "Length". This field is 16 bits in size and represents the length in bytes of the packet (header included), which means the theoretical maximum size for a UDP packet would be 65,535 bytes. In practice, however, most services (for instance DNS) restrict the largest packet length to 512 bytes in order to respect the Maximum Transmission Unit (MTU) on the Internet and avoid frequent packet loss.

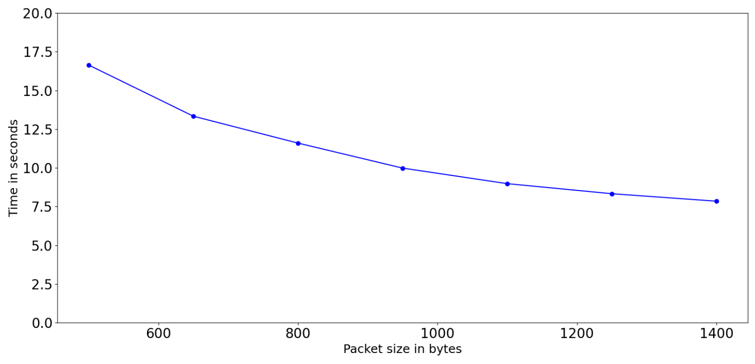

Figure 15. Time to send 100 Mb depending on the individual packet size.

Comparison with a Direct Sending of UDP Packets

To see how ICRP performs, we compared the times involved in the sending of a given amount of data between two fixed nodes, using our prototype implementation and a direct sending including no time measurements, authentication, or acknowledgments.

As Figure 15 has demonstrated, the size of individual packets has an impact on the global sending time of a file. Hence, those 2 methods should send packets of comparable sizes. The following Table 3 displays the measures of the overall sending of 10, 40, 100 and 200 Megabytes of data using the two methods with a constant packet size of 500 bytes and compares the obtained Throughput.

The average throughput is about 2.15 Mb/s for direct sending and about 1.79 Mb/s using ICRP. The measures were performed between a personal computer based in Caen, France, and a Server supplied by AWS (Amazon Web Services). The slight loss in performance is due to the fact that ICRP has to handle multiple threads concurrently on both the Sender's and Receiver's side, which is obviously not the case for direct sending. It induces that the processing capabilities of the endpoints have an impact on throughput performances. This impact remains low though, as, in this experiment, the sending machine was a personal laptop with few processing capabilities, and still limited the throughput loss to 16.7%.

Table 3. Overall sending times in seconds for large files.

Size and Parameters (n, k, p) \ Sending Method	Direct	ICRP
10 Mb $(250, 80, 500)$	4.668	6.138
40 Mb $(250, 320, 500)$	20.010	21.936
100 Mb $(250, 800, 500)$	44.384	54.323
200 Mb $(250, 1600, 500)$	89.916	106.508

5.2.3. Volume

We quantify the volume of overhead data added through a (n, p, k)-sending in Active Mode in comparison with the amount of raw information transmitted ($n \cdot p \cdot k$ bytes).

1. Each message (resp. response) is marked with a bit s_i (resp. r_i). This gives an additional $2 \cdot n \cdot k$ bits of information traveling through the network.
2. Each message (resp. response) is complemented with a sequence number encoded onto a 4 bytes header. This adds another $64 \cdot n \cdot k)$ bits of additional data traveling.
3. During the verification part of our prototype, a $RSA2048$ signature is sent for each session, with an additional 2 bytes tag indicating the current session number. This adds another $(2048 + 16) \cdot k$ additional bits.

Overall, the total overhead of our protocol is $(66n + 2064)k$ bits. The proportion of additional data traveling through the network is :

$$\frac{(66n + 2064)k}{8npk} = \frac{66n + 2064}{8np}$$

This proportion is unrelated to the number of sessions k. Table 4 displays the overhead proportion for a few practical values of n and p.

Table 4. Proportion of additional data induced by ICRP depending on (n, p).

n \ p	500	1000	1500
256	1.85%	0.93%	0.62%
512	1.75%	0.88%	0.58%

It appears that increasing the values of n and p lowers the overhead proportion, but, as stated before, it is preferable to keep the number of round n below 512. Otherwise, the decision process would be too rarely applied. It is also advised to restrain the size p of individual packets to avoid too frequent packet loss.

6. Applicative Example

In this section, we present an example of a practical application for our protocol.

6.1. Situation

6.1.1. Context

In a healthcare service, we can have a central database securely storing the health information of the citizens. Every citizen has access to his personal health data and can securely share it with the medical staff when needed. This will help to operate appropriate treatments on time. For instance, when a doctor, specializing in emergency medical assistance, arrives at a car crash, he usually has to apply some urgent and precise treatments and operations to save the lives of the victims. Any mistake (e.g., potential allergies, health issues. . .) or delay may cause a human life loss. To help the doctor, he may have mobile equipment enabling him to remotely access (through the mobile network) the victim's health information that is securely stored in the central health servers and eventually add notes and remarks for the nursing staff who will take care of the patient once in the hospital.

In this situation, sensitive information will regularly travel from the doctor's device to the e-health/hospital database and have to be protected in rest as well as in transit. Protecting data at rest means securing all the infrastructure storing sensitive data at both hardware and software levels while protecting data in transit means providing assurance that the data has not been compromised in any sense during its travel from one point to another. Indeed, 5G tends to be a flexible and dynamic network, namely using network virtualization. This allows the running of virtualized network functions over any hardware machine located in any area (i.e., inside the same country or in different countries) and even those not owned by the network operator. This arises new challenges, especially in sensitive contexts such as e-heath use cases where stringent requirements about e-health data are imposed. E-health data are considered private and sensible and so, must be protected in rest and in transit.

The protocol we have presented in this article can precisely serve last requirements by quickly detecting a stealthy relay attack or even an accidental route alteration.

6.1.2. Data Path

Using our protocol in this situation means that we have to define where to place the two end points $\mathcal{S}$ and $\mathcal{R}$. Concretely, whenever a piece of information is sent from a doctor's device to a hospital database throughout the mobile network, there are three main networks involved in its forwarding:

1. Radio Access Network (5G-RAN) provides wireless connectivity for devices through 5G antenna (gNB).
2. Core Network (5G-CN) namely enables the authentication of devices and the transfer of user data.
3. Data network (Internet) enables the transfer of user data from the Core Network to its final destination.

Figure 16 schematizes this process with the 3 involved areas (RAN, CN, and Internet).

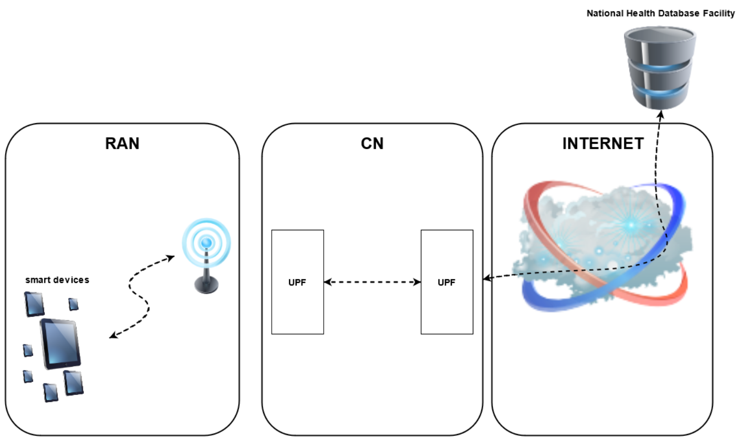

Figure 16. The 3 Networks traveled by data between hospitals and central database.

6.2. Our Protocol

In the next sections, we will describe and analyze four different setup options of our protocol in this architecture to protect health data from any intentional or accidental relay attack.

6.2.1. Over the Complete Path

The first obvious option would be to place the sender $\mathcal{S}$ and the receiver $\mathcal{R}$ at the very end-points of the communications (e.g., $\mathcal{S}$ being set on the doctor's device and $\mathcal{R}$ on the hospital servers). It is easy to observe that this option is not suitable to achieve the time stability needed by our protocol. As previously mentioned, the information transits through three types of networks, which would lead to frequently changing reference tables or even to a completely chaotic time behavior.

6.2.2. Over the RAN

Although it is possible for an attacker to listen to radio frequency communications between smart devices and the gNB, such an eavesdrop is not strictly speaking a relay because the information is actually broadcasted through the air and the attacker would need to recognize specific messages coming from the hospital or the doctor to perform metadata analysis. There are already many proposals allowing untraceable communications, some of which are already integrated into the Authentication Key Agreement for a 5G handshake. Moreover, applying our protocol over the RAN is actually quite meaningless because $\mathcal{S}$ and $\mathcal{R}$ would frequently move (geographically speaking), inducing a constantly changing RTT between the two.

6.2.3. Over the Core Network

As it was mentioned earlier, 5G technologies will bring a lot of virtualization, implying more uncertainty on the location of the involved physical equipment. In cases where the Mobile Network Operator (MNO) has not full control over this physical equipment (i.e. if other companies are providing them to the MNO), our protocol can help to detect non-advertised changes if those changes create an impact on RTT, which is probably the case if the infrastructure provider gets compromised. The Core Network is an IP private network, hence, it should be a highly stable area time-wise, which makes our protocol easily applicable between two nodes of the Core Network such as the User Plane Function (UPF) entry node and the UPF exit node.

6.2.4. Over the Internet

This application directly follows the guiding thread of this paper, which is applying our protocol between two fixed nodes communicating through the Internet. Here, $\mathcal{S}$ is running on the CN exit gateway and $\mathcal{R}$ is running on the e-health server.
In this situation, $\mathcal{S}$ belongs to the network operator and $\mathcal{R}$ belongs to the e-health service. Obviously, the e-health service is the one that would be in charge of verifying that no relay has been running during data exchange. To do so, we can apply ICRP as in Figure 12 (on Section 4.2).

7. Conclusions

In this paper, we introduced an innovative approach to detect relay attacks on time-stable networks. Our protocol is based on a distance-bounding mechanism that detects abnormally long round-trip times. To the best of our knowledge, our solution is the first time-based hijack detection over the internet. We experimentally show that our protocol is able to detect all deviated traffic up to realistic attacker capabilities, and we offer users the possibility to control the sensitivity of the decision process. This sensitivity control allows an absolute 0% of false positives while leaving realistically low chances to perform a successful relay in the long run. We studied the long-term behavior of RTT over the Internet for UDP packets and observed satisfying stability, allowing the reference sample not to be updated too frequently. We showed that our protocol adds absolutely no workload for the forwarding intermediary equipment and demands very low computational capabilities for the endpoints while keeping very satisfying throughput performances. Finally, we described what we believe to be two realistic applications of our protocol where both parties can benefit from its use. It is worth noting that our protocol can be easily deployed:

application-oriented, it does not require any update of the routers, it can be used by partners without following a long standardization process and does not require to be updated when the routing process changes as long as the RTT conserves a satisfying stability.

Author Contributions: Conceptualization, O.G. and G.A. (Gildas Avoine) and G.A. (Ghada Arfaoui) and J.T.; investigation, O.G.; data curation, O.G.; methodology, O.G. and G.A. (Gildas Avoine) and G.A. (Ghada Arfaoui) and J.T.; software, O.G.; writing—original draft preparation, O.G.; writing—review and editing, G.A. (Gildas Avoine) and G.A. (Ghada Arfaoui) and J.T.; supervision, G.A. (Gildas Avoine) and G.A. (Ghada Arfaoui) and J.T. All authors have read and agreed to the submitted version of the manuscript.

Funding: The research leading to these results partly received funding from the European Union's Horizon 2020 research and innovation program under grant agreement no. 871808 (5GPPP project INSPIRE-5Gplus). The article reflects only the authors' views. The Commission is not responsible for any use that may be made of the information it contains.

Data Availability Statement: All the measures supporting this work can be found at https://github.com/OlivierGim/ICRP.git, accessed on 29 August 2022.

Acknowledgments: Authors want to thank the support of Orange Labs Services, Poland which supplied virtual nodes to extend the experiments to a larger scale.

Conflicts of Interest: The authors declare the following conflict of interest: Gildas Avoine is a member of the editorial board of the journal Cryptography.

Abbreviations

The following abbreviations are used in this manuscript:

CN	Core Network
ISP	Internet Service Provider
MDPI	Multidisciplinary Digital Publishing Institute
MNO	Mobile Network Operator
PAN	Path Aware Networking
RAN	Radio Access Network
RFID	Radio Frequency IDentification
RTT	Round Trip Time
UPF	User Plane Function

References

1. NRO-Statistics-2021-Q4. Available online: https://www.nro.net/wp-content/uploads/NRO-Statistics-2021-Q4-FINAL.pdf (accessed on 31 December 2021).
2. RIPE—Youtube Hijacking, a RIPE NCC RIS Case Study. Available online: https://www.ripe.net/publications/news/industry-developments/youtube-hijacking-a-ripe-ncc-ris-case-study (accessed on 17 March 2008).
3. Arstechnica—How China Swallowed 15% of Net Traffic for 18 Minutes. Available online: https://arstechnica.com/information-technology/2010/11/how-china-swallowed-15-of-net-traffic-for-18-minutes/ (accessed on 17 November 2010).
4. Arstechnica—BGP Event Sends European Mobile Traffic through China Telecom for 2 Hours. Available online: https://arstechnica.com/information-technology/2019/06/bgp-mishap-sends-european-mobile-traffic-through-china-telecom-for-2-hours/ (accessed on 8 June 2019).
5. Apostolaki, M.; Zohar, A.; Vanbever, L. Hijacking bitcoin: Routing attacks on cryptocurrencies. In Proceedings of the 2017 IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, 22–26 May 2017; IEEE: Washington DC, USA, 2017; pp. 375–392.
6. Rekhter, Y.; Li, T.; Hares, S. A Border Gateway Protocol 4 (BGP-4). Available online: https://www.rfc-editor.org//rfc4271 (accessed on 29 August 2022).
7. Kent, S.; Lynn, C.; Seo, K. Secure border gateway protocol (S-BGP). *IEEE J. Sel. Areas Commun.* **2000**, *18*, 582–592. [CrossRef]
8. White, R. Securing BGP through secure origin BGP (soBGP). *Bus. Commun. Rev.* **2003**, *33*, 47.
9. Wan, T.; Kranakis, E.; van Oorschot, P.C. Pretty Secure BGP, psBGP. In Proceedings of the NDSS, San Diego, CA, USA, 3 February 2005; Citeseer: State College, PA, USA, 2005.
10. Karlin, J.; Forrest, S.; Rexford, J. Pretty good BGP: Improving BGP by cautiously adopting routes. In Proceedings of the ICNP, Washington, DC, USA, 12–15 November 2006; pp. 290–299. [CrossRef]
11. Lepinski, M.; Sriram, K. BGPsec Protocol Specification. Available online: https://www.rfc-editor.org/rfc/rfc8205 (accessed on 29 August 2022)

12. Sermpezis, P.; Kotronis, V.; Gigis, P.; Dimitropoulos, X.; Cicalese, D.; King, A.; Dainotti, A. ARTEMIS: Neutralizing BGP hijacking within a minute. *IEEE/ACM Trans. Netw.* **2018**, *26*, 2471–2486. [CrossRef]
13. Qiu, T.; Ji, L.; Pei, D.; Wang, J.; Xu, J. TowerDefense: Deployment strategies for battling against IP prefix hijacking. In Proceedings of the The 18th IEEE International Conference on Network Protocols, Kyoto, Japan, 5–8 October 2010; pp. 134–143. [CrossRef]
14. Holterbach, T.; Vissicchio, S.; Dainotti, A.; Vanbever, L. SWIFT: Predictive Fast Reroute. In Proceedings of the SIGCOMM; Association for Computing Machinery, SIGCOMM '17, New York, NY, USA, 3–4 April 2017; pp. 460–473. [CrossRef]
15. Mitseva, A.; Panchenko, A.; Engel, T. The state of affairs in BGP security: A survey of attacks and defenses. *Comput. Commun.* **2018**, *124*, 45–60. [CrossRef]
16. Calvert, K.L.; Griffioen, J.; Poutievski, L. Separating routing and forwarding: A clean-slate network layer design. In Proceedings of the 2007 Fourth International Conference on Broadband Communications, Networks and Systems (BROADNETS'07), Raleigh, NC, USA, 10–14 September 2007; IEEE: Washington DC, USA, 2007, pp. 261–270.
17. Yang, X.; Clark, D.; Berger, A.W. NIRA: A New Inter-Domain Routing Architecture. *IEEE/ACM Trans. Netw.* **2007**, *15*, 775–788. [CrossRef]
18. Raghavan, B.; Verkaik, P.; Snoeren, A.C. Secure and policy-compliant source routing. *IEEE/ACM Trans. Netw.* **2008**, *17*, 764–777. [CrossRef]
19. Godfrey, P.B.; Ganichev, I.; Shenker, S.; Stoica, I. Pathlet routing. *ACM SIGCOMM Comput. Commun. Rev.* **2009**, *39*, 111–122. [CrossRef]
20. Anderson, T.; Birman, K.; Broberg, R.; Caesar, M.; Comer, D.; Cotton, C.; Freedman, M.J.; Haeberlen, A.; Ives, Z.G.; Krishnamurthy, A.; et al. The nebula future internet architecture. In *Future Internet Assembly*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 16–26.
21. Perrig, A.; Szalachowski, P.; Reischuk, R.M.; Chuat, L. *SCION: A Secure Internet Architecture*; Springer: Berlin/Heidelberg, Germany, 2017.
22. Brands, S.; Chaum, D. Distance-Bounding Protocols. In Proceedings of the Advances in Cryptology—EUROCRYPT '93, Santa Barbara, CA, USA, 22–26 August 1993; Helleseth, T., Ed.; Springer: Berlin/Heidelberg, Germany, 1994; pp. 344–359.
23. Hancke, G.P.; Kuhn, M.G. An RFID Distance Bounding Protocol. In Proceedings of the First International Conference on Security and Privacy for Emerging Areas in Communications Networks (SECURECOMM'05), Washington, DC, USA, 5–9 September 2005; pp. 67–73.
24. Fischlin, M.; Onete, C. Subtle Kinks in Distance-Bounding: An Analysis of Prominent Protocols. In Proceedings of the Sixth ACM Conference on Security and Privacy in Wireless and Mobile Networks, WiSec '13, Budapest, Hungary, 17–19 April 2013; Association for Computing Machinery: New York, NY, USA, 2013; pp. 195–206. [CrossRef]
25. Boureanu, I.; Vaudenay, S. Challenges in distance bounding. *IEEE Secur. Priv.* **2015**, *13*, 41–48. [CrossRef]
26. Avoine, G.; Munilla, J.; Peinado, A.; Rasmussen, K.; Singelée, D.; Tchamkerten, A.; Trujillo-Rasua, R.; Vaudenay, S.; Bingöl, M.; Boureanu, I.; et al. Security of Distance-Bounding: A Survey. *ACM Comput. Surv.* **2018**, *51*, 1–33. [CrossRef]
27. Arfaoui, G.; Avoine, G.; Gimenez, O.; Traoré, J. How Distance-Bounding Can Detect Internet Traffic Hijacking. In Proceedings of the Cryptology and Network Security, Vienna, Austria, 13–15 December 2021; Conti, M., Stevens, M., Krenn, S., Eds.; Springer International Publishing: Cham, Switzerland, 2021; pp. 355–371.
28. MIFARE Plus EV2. Available online: https://www.nxp.com/docs/en/data-sheet/MF1P(H)x2_SDS.pdf (accessed on 29 August 2022).
29. MF3D(H)x3, MIFARE DESFire EV. Available online: https://www.nxp.com/docs/en/data-sheet/MF3DHx3_SDS.pdf (accessed on 29 August 2022).
30. Desmedt, Y.; Goutier, C.; Bengio, S. Special uses and abuses of the Fiat-Shamir passport protocol. In Proceedings of the Conference on the Theory and Application of Cryptographic Techniques, Santa Barbara, CA, USA, 16–20 August 1987; Springer: Berlin/Heidelberg, Germany, 1987; pp. 21–39.
31. Fiat, A.; Shamir, A. How To Prove Yourself: Practical Solutions to Identification and Signature Problems. In Proceedings of the Advances in Cryptology—CRYPTO' 86, Santa Barbara, CA, USA, 11–15 August 1986; Odlyzko, A.M., Ed.; Springer: Berlin/Heidelberg, Germany, 1987; pp. 186–194.
32. Postel, J. User Datagram Protocol. Available online: https://www.rfc-editor.org/rfc/rfc768 (accessed on 29 August 2022).