Appears in: Computer Networks, Elsevier, 2013 (DOI: 10.1016/j.comnet.2013.02.013)
Article first published online: 28 Feb 2013. Available at: http://dx.doi.org/10.1016/j.comnet.2013.02.013

Outbound SPIT Filter with Optimal Performance Guarantees

Montefiore Institute
Technical Report ULg 13-02
The University of Liège

Tobias Jung[1]
tjung@ulg.ac.be

Sylvain Martin[1]
sylvain.martin@ulg.ac.be

Mohamed Nassar[2]
mohamed.nassar@inria.fr

Damien Ernst[1]
dernst@ulg.ac.be

Guy Leduc[1]
guy.leduc@ulg.ac.be

[1]Montefiore Institute
Department of Electrical Engineering and Computer Science
University of Liège
Belgium

[2] INRIA Grand Est - LORIA Research Center
France

Outbound SPIT Filters with Optimal Performance Guarantees

Technical Report ULg 13-02

Draft January 29, 2013

Abstract

This paper[1] presents a formal framework for identifying and filtering SPIT calls (SPam in Internet Telephony) in an outbound scenario with provable optimal performance. In so doing, our work is largely different from related previous work: our goal is to rigorously formalize the problem in terms of mathematical decision theory, find the optimal solution to the problem, and derive concrete bounds for its expected loss (number of mistakes the SPIT filter will make in the worst case). This goal is achieved by considering an abstracted scenario amenable to theoretical analysis, namely SPIT detection in an outbound scenario with pure sources. Our methodology is to first define the cost of making an error (false positive and false negative), apply Wald's sequential probability ratio test to the individual sources, and then determine analytically error probabilities such that the resulting expected loss is minimized. The benefits of our approach are: (1) the method is optimal (in a sense defined in the paper); (2) the method does not rely on manual tuning and tweaking of parameters but is completely self-contained and mathematically justified; (3) the method is computationally simple and scalable. These are desirable features that would make our method a component of choice in larger, autonomic frameworks.

Keywords: security, internet telephony, SPAM, sequential probability ratio test

Short title: Outbound SPIT Filters with Optimal Performance Guarantees

[1] Note that this article is an expanded journal version of the earlier conference paper [5] which appeared in the proceedings of the *International Conference on Autonomous Infrastructure, Management and Security* (AIMS'12).

1 Introduction

Over the last years, Voice over IP (VoIP) has gained momentum as the natural complementary to emails, although its adoption is still young. The technologies employed in VoIP are widely similar to those used for emails and a large portion is actually identical. As a result, one can easily produce hundreds of concurrent calls per second from a single machine, replaying a pre-encoded message as soon as the callee accepts the call. This application of SPAM over Internet Telephony – also known as SPIT – is considered by many experts of VoIP as a severe potential threat to the usability of the technology [15]. More concerning, many of the defensive measures that are effective against email SPAM do not directly translate in SPIT mitigation: unlike with SPAM in emails, where the content of a message is text and is available to be analyzed *before* the decision is made of whether to deliver it or flag it as SPAM, the content of a phone call is a voice stream and is only available when the call is actually answered.

The simplest guard against SPIT would be to enforce strongly authenticated identities (maintaining caller identities on a secure and central server) together with personalized white lists (allowing only friends to call) and a consent framework (having unknown users first ask for permission to get added to the list). However, this is not supported by the current communication protocols and also seems to be infeasible in practice. Thus a number of different approaches have been previously suggested to address SPIT prevention, which mostly derive from experience in e-mail or web SPAM defense. They range from reputation-based [6, 1] and call-frequency based [14] dynamic black-listing, fingerprinting [22], to challenging suspicious calls by captchas [13, 16, 11], or the use of more sophisticated machine learning. For example, [9, 7] suggests SVM for anomaly detection in call histories, [8] decision trees for classification, and [21] semi-supervised learning, a variant of k-means clustering with features optimized from partially labeled data, to cluster and discriminate SPIT from regular calls.

These methods provide interesting building blocks, but, in our opinion, suffer from two main shortcomings. First, they do not provide performance guarantees in the sense that it is difficult to get an estimate of the number of SPIT calls that will go through and the number of regular calls they will erroneously stop. Second, they require a lot of hand-tuning for working well, which cannot be sustained in future's networks.

The initial motivation for this paper was to explore whether there would be ways to design SPIT filters that would not suffer from these two shortcomings. For this, we start by considering an abstracted scenario amenable to theoretical analysis where we make essentially two simplifying assumptions:

1. we are dealing with *pure source* SPIT detection in an *outbound* scenario,
2. we can extract features from calls (such as, for example, call duration) whose distribution for SPIT and regular calls is *known in advance*.

Here, "outbound scenario" means that our SPIT detector will be located in, or at the edge of, the network where the source resides, and will check all outgoing calls originating from within the network. Technically, this means that we are able to easily map calls to sources and that we can observe multiple calls from each source. By "pure source" we mean that a source either produces only SPIT or only NON-SPIT calls for a certain observation horizon. By "known in advance" we mean that the filter requires knowledge about the world in form of a generative model for the features of both SPIT and NON-SPIT calls. As in practice the "true" generative model will, of course, never be available and needs to be estimated from data, this requirement means that we need to have labeled[2] instances of both SPIT and NON-SPIT calls. Assuming that these requirements can all be fulfilled, we have been able to design a SPIT filter which requires no tuning and no user feedback and which is optimal in a sense that will be defined later in this paper.

This paper reports on this SPIT filter and is organized in two parts: one theoretical in Section 3 and one practical in Section 4. The theoretical part starts with Section 3.1 describing precisely and in mathematical terms the context in which we will design the SPIT filter. Section 3.2 shows how it

[2]As it turns out and will be explained later, it is at the time of this writing unfortunately quite difficult to acquire actual samples of SPIT calls.

is then possible to derive from a simple statistical test a SPIT filter with the desired properties and Section 3.3 provides analytical expressions to compute its performance. Monte Carlo simulations in Section 3.4 and 3.5 then examine the theoretical performance of the SPIT filter. The practical part starts with Section 4.1 describing how the SPIT filter could be integrated as one module into a larger hierarchical SPIT prevention framework. The primary purpose of this section is to demonstrate that the assumptions we make in Section 3 are well justified and can be easily dealt with in a real world application. (Note that a detailed description of the system architecture is not the goal of this paper.) For example, Section 4.2 describes how the assumption that the distributions for SPIT and NON-SPIT must be known in advance can be dealt with using maximum likelihood estimation from labeled prior data (which in addition allows us to elegantly address the problem of nonstationary attackers). Then, using learned distributions, Section 4.3 demonstrates for data extracted from a large database of real-world voice call data that the performance of our SPIT filter remains in accordance with the theoretical performance bounds derived analytically in Section 3 and degrades gracefully as the learned distribution departs from the model.

2 Related Work

To systematically place our work in the context of related prior work, we will have to consider it along two axes. The first axis deals with (low-level) *detection* algorithms: here we have to deal with the question on what abstract object we want to work with (e.g., SIP headers, stream data, call histories), how to represent this object such that computational algorithms can be applied (e.g., what features), and what precise algorithm is applied to arrive at a decision, which can be a binary classification (NON-SPIT/SPIT), a score (interpreted as how likely it is to be NON-SPIT/SPIT), or something else. The second axis deals with larger SPIT detection *frameworks* in which the (low-level) detection algorithm is only a small piece. The framework manages and controls the complete flow and encompasses detection, countermeasures, and self-healing. The formal framework for SPIT filtering we propose in this paper clearly belongs to the first category and only addresses low-level detection.

The Progressive Multi Gray-leveling (PMGL) proposed in [14] is a low-level detector that monitors call patterns and assigns a gray level to each caller based on the number of calls placed on a short and long term. If the gray level is higher than a certain threshold, the call is deemed SPIT and the caller blocked. The PMGL is similar to what we are doing in that it attempts to identify sources that are compromised by bot nets in an outbound scenario. The major weakness of the PMGL is: (1) that it relies on a weak feature, as spitters can easily adapt and reduce their calling rate to remain below the detection threshold, and high calling rates can also have other valid causes such as a natural disaster; (2) that it relies on "carefully" hand-tuned detection thresholds to work, which makes good performance in the real world questionable and – in our opinion – is not a very desirable property as it does not come with any worst-case bounds or performance guarantees. Our approach is exactly the opposite as it starts from a mathematically justified scenario and explicitly provides performance guarantees and worst-case bounds. Our approach is also more generic because it can work with *any* feature representation: while we suggest call duration is a better choice than call rate, our framework will work with whatever feature representation a network operator might think is a good choice (given their data).

In [21] a low-level detector based on semi-supervised clustering is proposed. They use a large number of call features, and because most of the features become available only after a call is accepted, is also primarily meant to classify pure sources (as we do). The algorithm they propose is more complex and computationally more demanding than what we propose. In addition, their algorithm also relies on hand-tuned parameters and is hard to study analytically; thus again it is impossible to have performance guarantees and derive worst-case bounds for it. Performance-wise it is hard to precisely compare the results due to a different experimental setup, but our algorithm compares favorably and achieves a very high accuracy.

The authors of [9, 7] propose to use support vector machines for identifying a varied set of VoIP misbehaviors, including SPIT. Their approach works on a different representation of the problem (call histories) with a different goal in mind and thus is not directly comparable with what we do. While it also cannot offer performance guarantees and worst-case bounds, in some respect it is more general than

what we propose; it also describes both detection and remediation mechanisms in a larger framework.

In SEAL [13], the authors propose a complete framework for SPIT prevention which is organized in two stages (passive and active). The passive stage performs low-level detection and consists of simple, unintrusive and computationally cheap tests, which, however, will only be successful in some cases and can be easily fooled otherwise. The purpose of the passive stage is to screen incoming calls and flag those that could be SPIT. The active stage then performs the more complex, intrusive and computationally expensive tests, which with very high probability can identify SPIT (these tests more actively interact with the caller). SEAL is very similar to what we sketch in Section 4 of this paper. On the other hand, the low-level detection performed in SEAL is rather basic: while it is more widely applicable than what we do, it essentially consists only in comparing a weighted sum of features against a threshold. As with all the other related work, weights and thresholds again need to be carefully determined by hand; and again, since the problem is not modeled mathematically, performance guarantees and worst-case bounds are impossible to derive.

Finally, it should be noted that, while the problem of SPIT detection can, in some sense, be related to the problem of anomaly detection and prevention of DoS attacks in VoIP networks, for example see the work in [12, 23, 4, 24], it is not the same. The reason why this is not the same is that these security threats are typically specific attacks aimed at disrupting the normal operation of the network. SPIT on the other hand operates on the social level and may consist of unwanted advertisements or phishing attempts – but not *per se* malicious code. As a consequence, techniques from anomaly detection and prevention of DoS attacks cannot be directly applied to SPIT detection.

3 A SPIT Filter with Theoretically Optimal Performance

This section describes a formal framework for an outbound SPIT filter for which it is possible to prove optimality and provide performance guarantees. Note that this section is stated from a theoretical point of view. In Section 4 we outline how one could implement it in a real world scenario.

3.1 Problem Statement

As shown in Figure 1, we assume the following situation: the SPIT filter receives and monitors incoming calls from a number of different sources. The sources are independent from each other and will each place numerous calls over time. We assume that, over a given observation horizon, each source will either *only produce regular calls* or will *only produce SPIT calls*. As the sources are independent, we can run a separate instance of the SPIT filter for each and thus in the following will only deal with the case of a single source.

Every time a call arrives at the filter, the filter can essentially do one of two things: accept the call and pass it on to the recipient or block[3] the call. Each call is associated with certain features and we assume that *only if a call is accepted*, we can observe the corresponding features (which would be for example the case with call duration as feature). The fundamental assumption we make is that the distribution over the features will be different depending on whether a call is a SPIT call or a regular call, and, in this Section 3, that these distributions are known in advance. To quantify the performance of the filter, we consider three types of costs: (1) the cost for erroneously accepting a SPIT call; (2) the cost for erroneously blocking a NON-SPIT call; and (3) the basic cost for running the filter and extracting the features regardless of whether the call is SPIT or NON-SPIT.

Our goal is to decide, after observing a few calls, whether or not the source sends out SPIT. More precisely, we look for a decision policy that initially accepts all calls, thus refining the belief about whether or not the source is SPIT, and then at some point decides to either block or accept all future calls from the source. Seen as a sequential decision problem, the SPIT filter has three[4] possible actions:

[3] "Block" may seem like a rather severe option which is here justified because of our focus on abstract modeling. In practice, as we will describe in Section 4, the SPIT filter would ideally run in parallel with other detection algorithms and thus only recommend a course of action.

[4] It should be noted that, alternatively, one could imagine modeling this scenario as an optimal stopping problem with just two actions: (1) accept next call; (2) block all future calls. Doing this, however, would require a different mathematical

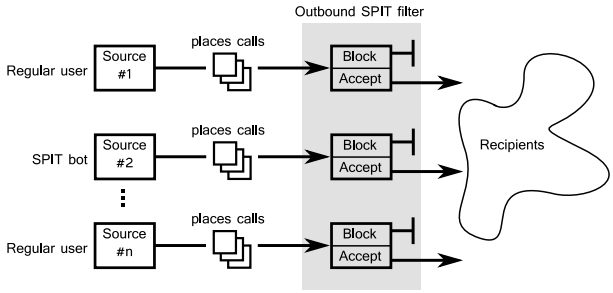


Figure 1: Sketch of the simplified problem. The SPIT filter operates in an outbound fashion and checks all outgoing calls originating from within the network. The filter treats the source of a call as being pure over a certain observation horizon. In this scenario, each source (which will correspond to a registered user) will try to sequentially place calls over time which are either all SPIT (if the source is compromised by a SPIT bot) or all NON-SPIT (if the source is a regular user). The SPIT filter then has to decide for each source individually if, given the calls that originated from that source in the past, it is a regular user or a SPIT bot.

(1) accept the next call, which reveals its features and thus refines the belief about the type of the source, (2) block all future calls, and (3) accept all future calls.

In doing so, we arrive at a well defined concept of *loss*, which we define as the total costs accumulated over the observation horizon. Within this framework, every conceivable SPIT filter algorithm will then have a performance number: its expected loss. The particular SPIT filter that we are going to describe below will be one that minimizes this expected loss.

Note that the expected loss is an example for the typical exploration vs exploitation dilemma. On the one side, since the decision to accept or block all future calls is terminal, we want to be very certain about its correctness to avoid making an expensive error. On the other side, as long as we are observing we are automatically accepting all calls and thus will increase our loss both because of the basic cost of running the filter plus the potential cost of having accepted a SPIT call. To minimize our expected loss, we therefore also want to observe as few samples as possible.

To address the problem mathematically, we employ Wald's sequential probability ratio test for simple hypotheses introduced in [18]. The sequential probability ratio test (SPRT) has the remarkable property that among all sequential tests procedures it minimizes the expected number of samples for a given level of certainty and regardless of which hypothesis is true (the optimality of SPRT was proved in [19]). In addition, the SPRT comes with bounds for the expected stopping time and thus allows us to derive concrete expressions for the expected loss as a function of the characteristics of the particular problem (meaning we can express the loss as a function of the parameters of the distribution for SPIT or NON-SPIT). Finally, SPRT requires only simple algebraic operations to carry out and thus is easy to implement and computationally cheap to run.

3.2 SPIT Detection via the SPRT

The SPRT is a test of one simple statistical hypothesis against another which operates in an online fashion and processes the data sequentially. At every time step a new observation is processed and one of the following three decisions is made: (1) the hypothesis being tested is accepted, (2) the hypothesis being tested is rejected and the alternative hypothesis is accepted, (3) the test statistic is not yet conclusive and further observations are necessary. If the first or the second decision is made, the test procedure is terminated. Otherwise the process continues until at some later time the first or second decision is made.

Two kind of misclassification errors may arise: decide to accept calls when source is SPIT, or decide to block all future calls when source is NON-SPIT. Different costs may be assigned to each kind, upon

approach, namely dynamic programming over belief states. While technically there is no problem in doing it, it is not the scenario we consider in this paper.

which the performance optimization process described in Section 3.3 is built.

To model the SPIT detection problem with the SPRT, we now proceed as follows: Assume we can make sequential observations from one source of *a priori* unknown type SPIT or NON-SPIT. Let x_t denote the features of the t-th call we observe, modeled by random variable X_t. The X_t are i.i.d. with common distribution (or density) p_X. The calls all originate from one source which can either be of type SPIT with distribution $p_{\text{SPIT}}(x) = p(x|\text{source=SPIT})$ or of type NON-SPIT with distribution $p_{\text{NON-SPIT}}(x) = p(x|\text{source=NON-SPIT})$. Initially, the type of the source we are receiving calls from is not known; we write $p(\text{SPIT})$ for the prior probability of a source being SPIT and $p(\text{NON-SPIT})$ for the prior probability of being NON-SPIT (note $p(\text{NON-SPIT}) = 1 - p(\text{SPIT})$).

In order to learn the type of the source, we observe calls $x_1, x_2, \ldots$ and test the hypothesis

$$H_0 : p_X = p_{\text{SPIT}} \qquad \text{versus} \qquad H_1 : p_X = p_{\text{NON-SPIT}}. \tag{1}$$

(Note again that in this formulation we assume that the densities p_{SPIT} and $p_{\text{NON-SPIT}}$ are both known so that we can readily evaluate the expression $p(x|\text{source=SPIT})$ and $p(x|\text{source=NON-SPIT})$ for any given x.)

At time t we observe x_t. Let

$$\lambda_t := \frac{p(x_1, \ldots, x_t|\text{NON-SPIT})}{p(x_1, \ldots, x_t|\text{SPIT})} = \prod_{i=1}^{t} \frac{p(x_i|\text{NON-SPIT})}{p(x_i|\text{SPIT})} \tag{2}$$

be the ratio of the likelihoods of each hypothesis after t observations $x_1, \ldots, x_t$. Since the X_i are independent, we can factor the joint distribution on the left side to obtain the right side. In practice, it will be more convenient for numerical reasons to work with the log-likelihoods. Doing this allows us to write a particular simple recursive update for the log-likelihood ratio $\Lambda_t := \log \lambda_t$, that is

$$\Lambda_t := \Lambda_{t-1} + \log \frac{p(x_t|\text{NON-SPIT})}{p(x_t|\text{SPIT})}. \tag{3}$$

After each update we examine which of the following three cases applies and act accordingly:

$$A < \lambda_t < B \implies \text{continue monitoring} \tag{4}$$
$$\lambda_t \geq B \implies \text{accept } H_1 \text{ (decide NON-SPIT)} \tag{5}$$
$$\lambda_t \leq A \implies \text{accept } H_0 \text{ (decide SPIT)} \tag{6}$$

Thresholds A and B with $0 < A < 1 < B < \infty$ depend on the desired accuracy or error probabilities of the test:

$$\alpha := P\{\text{accept } H_1 \,|\, H_0 \text{ true}\} = P\{\text{decide NON-SPIT} \,|\, \text{source=SPIT}\} \tag{7}$$
$$\beta := P\{\text{reject } H_1 \,|\, H_1 \text{ true}\} = P\{\text{decide SPIT} \,|\, \text{source=NON-SPIT}\}. \tag{8}$$

Note that α and β need to be specified in advance such that certain accuracy requirements are met (see next section where we consider the expected loss of the procedure). The threshold values A and B and error probabilities α and β are connected in the following way (cf. [18], Eqs. (3.18)-(3.19))

$$\beta \leq A(1 - \alpha) \qquad \text{and} \qquad \alpha \leq (1 - \beta)/B. \tag{9}$$

Note that the inequalities arise because of the discrete nature of making observations (i.e., at $t = 1, 2, \ldots$) which results in λ_t not being able to hit the boundaries exactly. In practice we will neglect this excess and treat the inequalities as equalities (cf. [18], p.131ff):

$$A = \beta/(1 - \alpha) \qquad \text{and} \qquad B = (1 - \beta)/\alpha. \tag{10}$$

Let T be the random time at which the sequence of the λ_t leaves the open interval (A, B) and a decision is made that terminates the sampling process. (Note that stopping time T is a random quantity

due to the randomness of the sample generation.) The SPRT provides the following pair of inequalities for the expected stopping time in both cases (cf. [18], Eqs. (4.80)-(4.81))

$$\mathbb{E}_{X_i \sim p_{\text{SPIT}}}[T] \geq \frac{1}{\kappa_0}\left(\alpha \log \frac{1-\beta}{\alpha} + (1-\alpha)\log\frac{\beta}{1-\alpha}\right) \tag{11}$$

$$\mathbb{E}_{X_i \sim p_{\text{NON-SPIT}}}[T] \geq \frac{1}{\kappa_1}\left(\beta \log \frac{\beta}{1-\alpha} + (1-\beta)\log\frac{1-\beta}{\alpha}\right). \tag{12}$$

(which we can treat as equalities when we use Eq. (10)). The constants κ_i with $\kappa_0 < 0 < \kappa_1$ are the Kullback-Leibler information numbers defined by

$$\kappa_0 = \mathbb{E}_{x \sim p_{\text{SPIT}}}\left[\log \frac{p(x|\text{NON-SPIT})}{p(x|\text{SPIT})}\right] \tag{13}$$

$$\kappa_1 = \mathbb{E}_{x \sim p_{\text{NON-SPIT}}}\left[\log \frac{p(x|\text{NON-SPIT})}{p(x|\text{SPIT})}\right]. \tag{14}$$

The constants κ_i can be interpreted as a measure of how difficult it is to distinguish between p_{SPIT} and $p_{\text{NON-SPIT}}$. The smaller they are the more difficult is the problem.

3.3 Theoretical Performance of the SPIT Filter

We will now look at the performance of our SPIT filter and derive expressions for its expected loss. Let us assume we are going to receive a finite number N of calls and that N is sufficiently large such that the test will always stop before the calls are exhausted. We consider the following scalar costs per call:

c_0	basic cost of running the filter
c_1	cost of erroneously accepting SPIT
c_2	cost of erroneously blocking NON-SPIT

Let us recall the decision-making policy the filter implements: at the beginning, all calls are accepted (observing samples x_i) until the test becomes sufficiently certain about its prediction. Once the test has become sufficiently certain, the filter executes the following rule: if the test returns that the source is SPIT, then all future calls from it will be blocked. If the test says that the source is NON-SPIT, then all future calls from it will be accepted. Let L denote the loss incurred by this policy (note that L is a random quantity). To compute the expected loss $\mathbb{E}[L]$, we have to divide N into two parts: the first part from 1 to T corresponds to the running time of the test where all calls are automatically accepted ($T < N$ being the random stopping time with expectation given in Eqs. (11)-(12)), the second part from $T+1$ to N corresponds to the time after the test.

If H_0 is true, that is, the source is SPIT, the loss L will be the random quantity

$$L|\text{source=SPIT} \;=\; (c_0 + c_1)T + \alpha c_1 (N - T) \tag{15}$$

where $(c_0 + c_1)T$ is the cost of the test, α the probability of making the wrong decision, and $c_1(N-T)$ the cost of making the wrong decision for the remaining calls. Taking expectations gives

$$\mathbb{E}_{X_i \sim p_{\text{SPIT}}}[L] = \alpha c_1 N + [c_0 + c_1(1-\alpha)]\mathbb{E}_{X_i \sim p_{\text{SPIT}}}[T]. \tag{16}$$

Likewise, if H_1 is true, that is, the source is NON-SPIT, our loss will be the random quantity

$$L|\text{source=NON-SPIT} \;=\; c_0 T + \beta c_2 (N - T) \tag{17}$$

where $c_0 T$ is the cost of the test, β the probability of making the wrong decision, and $c_2(N-T)$ the cost of making the wrong decision for the remaining calls. Taking expectation gives

$$\mathbb{E}_{X_i \sim p_{\text{NON-SPIT}}}[L] = \beta c_2 N + [c_0 - \beta c_2]\mathbb{E}_{X_i \sim p_{\text{NON-SPIT}}}[T]. \tag{18}$$

The total expected loss takes into consideration both cases and is simply

$$\mathbb{E}[L] = p(\text{SPIT}) \cdot \mathbb{E}_{X_i \sim p_{\text{SPIT}}}[L] + p(\text{NON-SPIT}) \cdot \mathbb{E}_{X_i \sim p_{\text{NON-SPIT}}}[L] \tag{19}$$

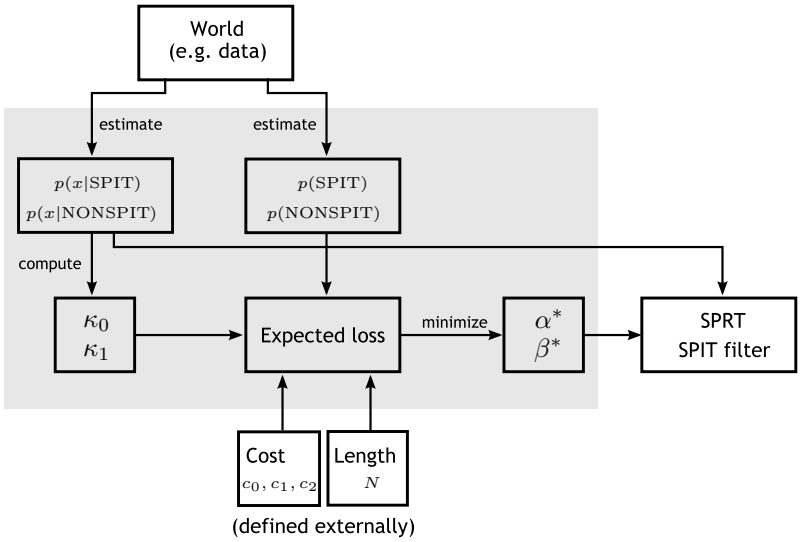


Figure 2: A dependency graph for the SPIT filter. The sketch shows how the various objects are related and which quantities are required to perform what computational step. Note that the computational steps inside the grey box are fully automated.

where p(SPIT) and p(NON-SPIT) is the prior probability for a source sending out SPIT or regular calls.

To summarize this part, let us consider the situation shown in Figure 2 where we want to apply the filter in practice. Recall that in order to run the filter (see Eqs. (3)-(6)), we need four objects: p_{SPIT}, $p_{\text{NON-SPIT}}$, α, and β. The first two are obvious: any specific problem is fully characterized by the joint distribution of features and class which we can factor into p_{SPIT}, $p_{\text{NON-SPIT}}$, and the priors p(SPIT) and p(NONSPIT). These distributions can be estimated from data as is described in Section 4. Knowing these distributions, we can compute κ_0, κ_1. We assume that the cost c_0, c_1, c_2 and the number of calls N are defined externally. Looking at Eq. (19), we see that, given all this information, the expected loss will be a function of α, β. To make this visually more clear, we will write the expected loss as a function of two variables, $\bar{L}(\alpha, \beta)$. One natural way of choosing α, β for the SPIT filter now is to look for that setting α^*, β^* that will minimize the expected loss $\bar{L}$. Doing this results in the generally nonconvex optimization problem

$$\min_{\alpha,\beta} \bar{L}(\alpha, \beta) \qquad s.t. \quad \alpha \geq 0, \beta \geq 0 \tag{20}$$

which has to be solved by iterative techniques. In practice, to ensure reasonable results one will also bound the variables using box constraints, such as $\alpha, \beta \in [10^{-6}, 10^{-1}]$.

3.4 Example: Exponential Duration Distribution

In this section we assume that the distributions p_{SPIT} and $p_{\text{NON-SPIT}}$ are of a certain parametric form for which it is possible to compute the derived quantities analytically and in closed form. Specifically, we assume that both are exponential distributions with parameters $\lambda_0, \lambda_1 > 0$, that is, are given by

$$p(x|\text{SPIT}) = \lambda_0 \exp(-\lambda_0 x), \quad p(x|\text{NONSPIT}) = \lambda_1 \exp(-\lambda_1 x) \tag{21}$$

for $x > 0$. While this section is primarily meant as a numerical example to illustrate the behavior of a SPRT-based SPIT filter theoretically, it is not an altogether unreasonable scenario to assume for a real world SPIT filter. For example, we could assume that the only observable feature x of (accepted) calls is their duration (see Section 4). In this case SPIT calls will have a shorter duration than regular calls because after a callee answers the call, they will hang up as soon as they realize it is SPIT. The majority of regular calls on the other hand will tend to have a longer duration. While this certainly simplifies the situation from the real world (e.g., it is generally assumed that call duration follows a more complex and heavy-tailed distribution [2]), we can imagine that both durations can be modeled

by an exponential distribution with an average (expected) length of SPIT calls of $1/\lambda_0$ minutes and an average length of NON-SPIT calls of $1/\lambda_1$ minutes ($\frac{1}{\lambda_1} > \frac{1}{\lambda_0}$).

First, let us consider the expected stopping time from Eqs. (11)-(12). From Eqs. (13)-(14) we have that the Kullback-Leibler information number for Eq. (21) is given by

$$\begin{aligned}\kappa_0 &= \int_0^\infty \log\left[\frac{\lambda_1 \exp(-\lambda_1 x)}{\lambda_0 \exp(-\lambda_0 x)}\right] \cdot \lambda_0 \exp(-\lambda_0 x) dx \\ &= \log\frac{\lambda_1}{\lambda_0} \int_0^\infty \lambda_0 \exp(-\lambda_0 x) dx + (\lambda_0 - \lambda_1) \int_0^\infty x \cdot \lambda_0 \exp(-\lambda_0 x) dx \\ &= \log\frac{\lambda_1}{\lambda_0} + 1 - \frac{\lambda_1}{\lambda_0} \end{aligned} \tag{22}$$

(On the second line, the first integral is an integral over a density and thus is equal to one; the second integral is the expectation of p_{SPIT} and thus is equal to $1/\lambda_0$.) Similarly we obtain for κ_1 the expression

$$\kappa_1 = \log\frac{\lambda_1}{\lambda_0} - 1 + \frac{\lambda_0}{\lambda_1}. \tag{23}$$

Note that for more complex forms of distributions we may no longer be able to evaluate κ_i in closed form.

As we can see, κ_i only depends on the ratio λ_1/λ_0. Thus for fixed accuracy parameters α, β the expected stopping time in Eqs. (11)-(12) will also only depend on the ratio λ_1/λ_0. The closer the ratio is to zero, the fewer samples will be needed (the problem becomes easier); the closer the ratio is to one, the more samples will be needed (the problem becomes harder). Of course this result is intuitively clear: the ratio λ_1/λ_0 determines how similar the distributions are.

In Table 1 we examine numerically the impact of the difficulty of the problem, in terms of the ratio λ_1/λ_0, on the expected number of samples until stopping for different settings of accuracy α, β. For instance, an average NON-SPIT call duration of 2 minutes as opposed to an average duration of SPIT calls of 12 s leads to $\lambda_1/\lambda_0 = 0.1$, and distributions that are sufficiently dissimilar to arrive with high accuracy at the correct decision within a very short observation horizon: with accuracy $\alpha, \beta = 0.001$ the filter has to observe on the average 1.0 calls if the source is NON-SPIT and 4.9 calls if the source is SPIT to make the correct decision in at least 99.9% of all cases. (Notice that the stopping time is not symmetric.) On the other hand, with $\lambda_1/\lambda_0 > 0.5$ the similarity between SPIT and NON-SPIT becomes too strong, which is an indication that another feature or collection of features should be chosen to discriminate the two (see Section 4). In Table 2 we examine numerically how the ratio λ_1/λ_0, the setting of the cost $c_2 = kc_1$ (and $c_0 = 0$), and the prior probabilities affect what choice of accuracy thresholds α^*, β^* is optimal and how this affects the combined expected stopping time. Note that in this table we consider two different "worlds": one where 50% of all sources are SPIT bots, and one where only 1% are SPIT bots.

Next we will compute the log-likelihood ratio Λ_t. From Eq. (3) we have

$$\Lambda_t = \sum_{i=1}^{t} \log\frac{p(x_i|\text{NON-SPIT})}{p(x_i|\text{SPIT})} = \sum_{i=1}^{t} \left[\log\frac{\lambda_1}{\lambda_0} + (\lambda_0 - \lambda_1) x_i\right]. \tag{24}$$

The decision regions for the SPRT from Eq. (4) are thus

$$\log\frac{\beta}{1-\alpha} < t \cdot \log\frac{\lambda_1}{\lambda_0} + (\lambda_0 - \lambda_1) \sum_{i=1}^{t} x_i < \log\frac{1-\beta}{\alpha} \tag{25}$$

or, equivalently,

$$\log\frac{\beta}{1-\alpha} + t \cdot \left(\log\frac{\lambda_0}{\lambda_1}\right) < (\lambda_0 - \lambda_1) \sum_{i=1}^{t} x_i < \log\frac{1-\beta}{\alpha} + t \cdot \left(\log\frac{\lambda_0}{\lambda_1}\right). \tag{26}$$

λ_1/λ_0	κ_0	κ_1	$\alpha,\beta = 0.05$ $\mathbb{E}_{\text{SPIT}}[T]$	$\alpha,\beta = 0.05$ $\mathbb{E}_{\text{NON}}[T]$	$\alpha,\beta = 0.01$ $\mathbb{E}_{\text{SPIT}}[T]$	$\alpha,\beta = 0.01$ $\mathbb{E}_{\text{NON}}[T]$	$\alpha,\beta = 0.001$ $\mathbb{E}_{\text{SPIT}}[T]$	$\alpha,\beta = 0.001$ $\mathbb{E}_{\text{NON}}[T]$
0.99	-0.00005	0.00005	52646.2	52294.7	89463.4	88865.9	136938.9	136024.5
0.95	-0.00129	0.00133	2049.0	1980.1	3481.9	3364.9	5329.7	5150.5
0.90	-0.00536	0.00575	494.3	460.8	840.0	783.0	1285.8	1198.6
0.70	-0.05667	0.07189	46.7	36.8	79.4	62.6	121.6	95.8
0.50	-0.19314	0.30685	13.7	8.6	23.3	14.6	35.6	22.4
0.30	-0.50397	1.12936	5.2	2.3	8.9	3.9	13.6	6.1
0.10	-1.40258	6.69741	1.8	0.3	3.2	0.6	4.9	1.0
0.01	-3.61517	94.39486	0.7	<0.1	1.2	<0.1	1.9	0.1

Table 1: How does the difficulty of the problem, expressed in terms of the ratio λ_1/λ_0, affect the expected number of samples until stopping $\mathbb{E}_{\text{SPIT}}[T]$ and $\mathbb{E}_{\text{NON-SPIT}}[T]$ for different settings of the accuracy parameters α, β.

$(N = 20)$ λ_1/λ_0		$p(\text{SPIT}) = 0.5$ $c_2 = c_1$	$c_2 = 10c_1$	$c_2 = 1000c_1$	$p(\text{SPIT}) = 0.01$ $c_2 = c_1$	$c_2 = 10c_1$	$c_2 = 1000c_1$
	α^*	1.01e-06	1.00e-06	1.00e-06	1.15e-06	1.01e-06	5.85e-02
0.1	β^*	3.92e-02	3.96e-03	3.97e-05	4.08e-04	4.01e-05	1.00e-06
	$\mathbb{E}[T]$	2.13	2.99	4.64	2.07	2.11	0.51
	α^*	1.00e-06	1.00e-06	9.98e-02	5.37e-02	9.99e-02	9.75e-02
0.2	β^*	8.04e-02	8.60e-03	5.25e-05	5.94e-04	5.31e-05	1.00e-06
	$\mathbb{E}[T]$	4.15	5.79	5.75	1.29	1.06	1.10
	α^*	1.00e-06	7.66e-02	9.99e-02	9.99e-02	9.98e-02	6.80e-04
0.3	β^*	9.40e-02	9.46e-03	8.94e-05	9.03e-04	9.04e-05	4.09e-05
	$\mathbb{E}[T]$	7.74	5.10	9.10	2.12	2.17	6.59
	α^*	1.62e-06	2.37e-06	7.11e-06	7.37e-03	8.04e-03	1.03e-02
0.4	β^*	9.78e-02	7.74e-02	1.02e-02	1.58e-03	1.55e-03	1.57e-03
	$\mathbb{E}[T]$	13.63	14.04	17.23	8.49	8.34	7.91

Table 2: How do the characteristics of the problem, expressed as the ratio λ_1/λ_0 and prior $p(\text{SPIT})$, and the choice of the cost terms $c_2 = kc_1$ (and $c_0 = 0$) affect the optimal thresholds α^*, β^* and combined expected stopping time $\mathbb{E}[T] = p(\text{SPIT})\mathbb{E}_{\text{SPIT}}[T] + (1 - p(\text{SPIT}))\mathbb{E}_{\text{NON-SPIT}}[T]$.

From the latter we can see that the boundaries of the decision regions are straight and parallel lines (as a function t of samples). Running the SPRT can now be graphically visualized as shown in Figure 3: the log-likelihood ratio Λ_T starts for $t = 1$ in the middle region between the decision boundaries and, with each new sample it observes from the unknown source, does a random walk over time. Eventually it will cross over one of the lines after which the corresponding decision is made. For a fixed value of α, β, changing the ratio λ_0/λ_1 changes the slope of the decision boundaries. For a fixed value of λ_0, λ_1, changing the accuracy α, β shifts the decision boundaries upward and downward.

3.5 Experiment: perfectly known distribution

To examine the (theoretical) performance of SPRT, we ran a large number of Monte Carlo simulations for various settings of problem difficulty λ_1/λ_0 (λ_0 was set to 1, λ_1 was varied between 0.9 and 0.1) and accuracy α, β ($\alpha, \beta = 0.05$, $\alpha, \beta = 0.01$, $\alpha, \beta = 0.001$). For each setting we performed 50,000 independent runs and recorded, for each run, how many samples were necessary for SPRT to reach a decision and whether or not that decision was wrong. The experiment examines separately the case where the source is SPIT and NON-SPIT. The result of the simulation is shown in Figure 4 and confirms the expected stopping time computed analytically in Table 1. In addition, the results show that the actual number of mistakes made (the height of the bars in the figure) is in many cases notably smaller than the corresponding error probabilities α, β (the dashed horizontal lines in the figure), which are merely upper bounds.

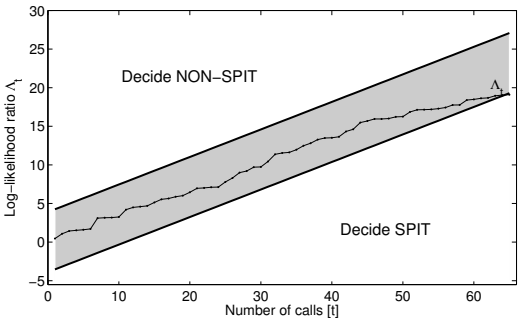


Figure 3: An example run of SPRT. The log-likelihood ratio Λ_t starts at $t = 1$ in the region between the decision boundaries and, with each new sample observed from the unknown source, performs a random walk over time. Eventually it will cross over one of the decision boundaries and either enter the region marked "decide SPIT" or enter the region marked "decide NON-SPIT".

4 Network Operator's Perspective

Having so far described our SPIT filter from a purely theoretical point of view, we now discuss the steps necessary to deploy it in the real world. Note that in what follows it is neither our intent nor within the scope of the paper to describe in detail the architecture of a fully functional SPIT prevention system.

The section is structured as follows. First we will sketch how the SPIT filter, which should more appropriately be seen as a SPIT detector, could be integrated into a larger SPIT prevention system as one building block among many others. We will make suggestions on how the problem-dependent parts of the SPRT can be instantiated by specifying:

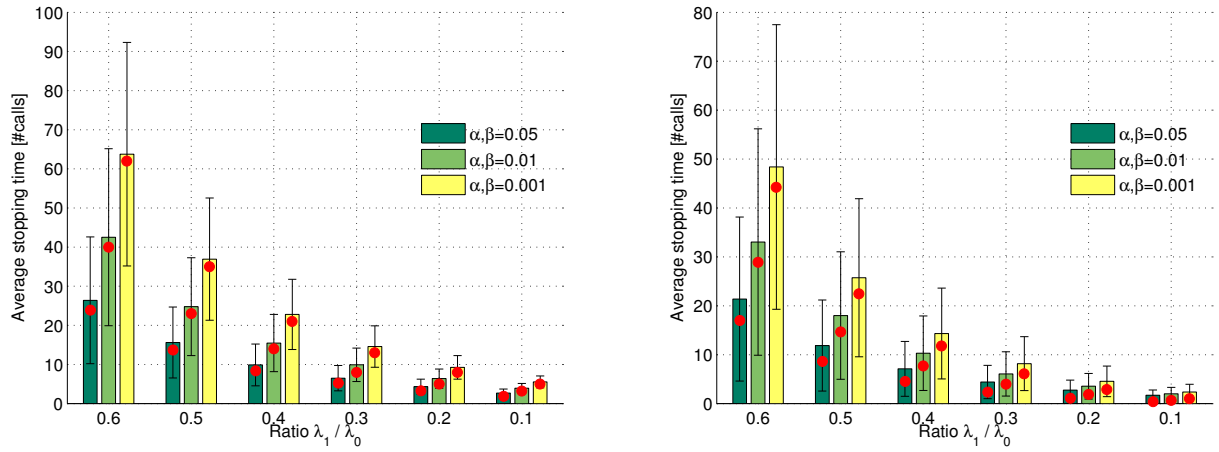
- *sources:* how to map calls to sources such that the pure source assumption is fulfilled
- *features:* what call features to use such that SPIT and NON-SPIT calls are well presented
- *actions:* what action to take if the SPRT indicates a source is likely to send out SPIT

We will then explain how the distribution of the features that discriminate SPIT from NON-SPIT can be learned from labeled data by first assuming that the distribution is of a certain parameterized form and then estimating these parameters from the data via maximum likelihood. In the second part of the section we use data extracted from a large database of real-world voice calls and demonstrate empirically that the performance of the SPIT filter under real-world conditions with *a priori* unknown distribution is also very good.

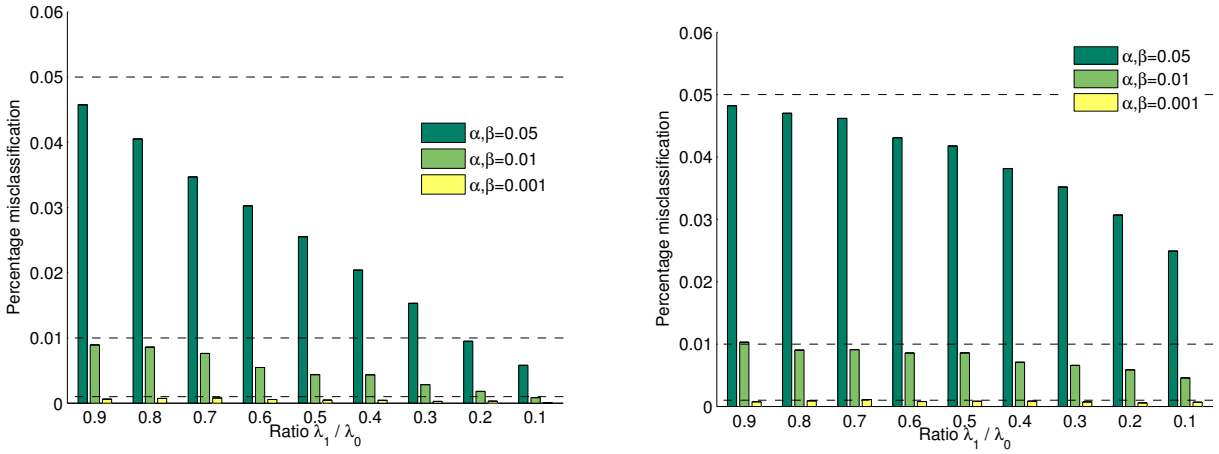
4.1 Integration into a SPIT Prevention System

For a network operator, a SPIT prevention system such as the one we propose and sketch in Figure 5 must allow both to maintain and guarantee an acceptable level of service under adverse operating conditions and have a low maintenance cost. To achieve this goal, we adopt the overall strategy presented in [17] which proposes a hierarchical system consisting of two modular layers: a basic service layer and a diagnostic layer. The basic service layer manages and processes call requests and as a whole serves to protect against attacks in VoIP networks – among which SPIT is just regarded as one particular threat. For the prevention of SPIT the basic layer is made up of two subcomponents (a conceptually similar setup was also proposed for SEAL [13]): *always-on detection* and *on-demand protection*.

Always-on detection consists of passive modules which essentially extract and make use of information which is "already there" and thus have zero or very low computational costs. On the other hand, these modules are only weak detectors in that they are successful only under restrictive conditions. If the always-on detection component cannot establish with high certainty that a call is NON-SPIT, in which case it would be allowed to pass through unharmed, the call is internally forwarded to the on-demand protection component.



Average stopping time if source is SPIT (left) and NON-SPIT (right)



Error rate if source is SPIT (left) and NON-SPIT (right)

Figure 4: Monte Carlo simulation of SPRT with the results averaged over 50,000 independent runs and error bars denoting one standard deviation. The top row shows the average number of calls necessary before SPRT stops for various settings of accuracy α, β. The red dots indicate the respective expected stopping time. The bottom row shows the proportion of SPRT ending up making the wrong decision (that is, accepting SPIT or blocking NON-SPIT) for various settings of accuracy α, β (shown as horizontal lines). As can be seen, in many instances the height of the bars is notably below the corresponding horizontal line, meaning that the actual error rate can be much smaller than what the accuracy parameters α, β would suggest, which are merely an upper bound.

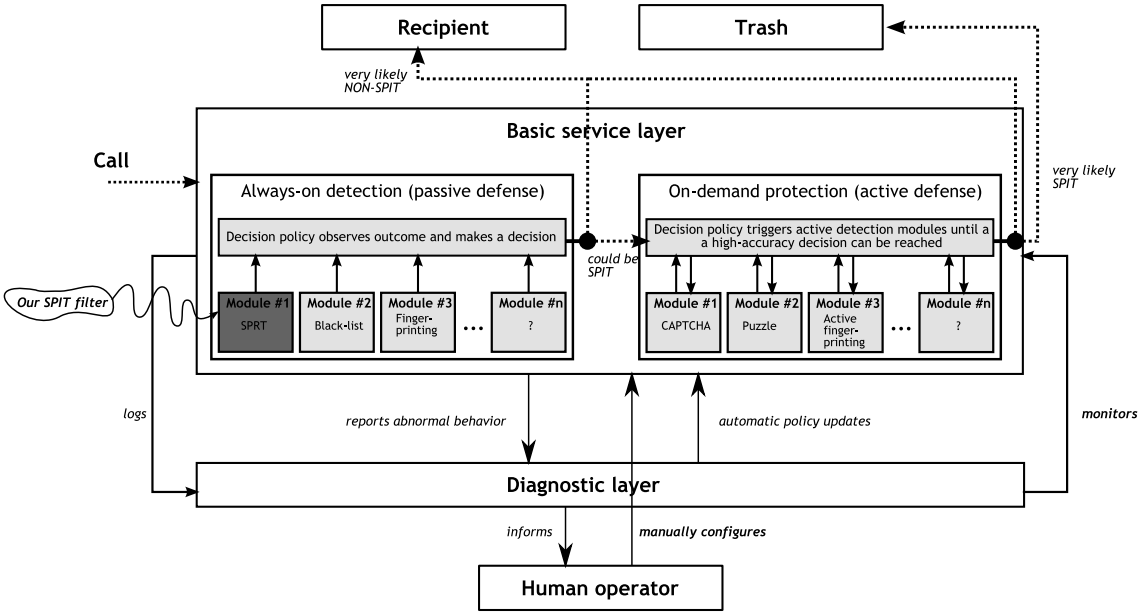


Figure 5: A hierarchical and modular SPIT prevention system. The SPRT-based SPIT filter we propose in this paper is one particular module in the always-on component of the basic service layer.

The on-demand protection component consists of active modules which require additional processing and can have medium to high computational costs; e.g., digit-based audio CAPTCHAs [16] or hidden Turing tests and computational puzzles [11]. These modules are meant to protect the network with high accuracy. However, because of the cost involved (they interfere with natural communication, consume and block resources, and may to some degree annoy human callers), they are triggered only individually and on demand. An intelligent decision policy controls the precise way a call gets probed by the various security tests, such that resource consumption is minimized, until a final decision SPIT or NON-SPIT can be made with high certainty.

4.1.1 Deployment as outbound SPIT filter

Logically, the SPIT filter we propose would be located within the always-on component. Physically, the SPIT filter would be located at the proxy servers which form the gateway between one's own network and the outside world. The SPIT filter will then act as an *outbound* filter: it will perform self-moderation of outgoing calls and unveil the presence of a SPIT botnet within one's own network before other networks take countermeasures. Previous experience with e-mail has shown that outbound filters are critical to keep control over one's traffic. They ensure that the whole network's address space will not end up on a black list as soon as a single of its child systems becomes enrolled into a botnet.

4.1.2 Defining sources

For an outbound filter, the definition of *sources* – that is, the mapping of individual call requests to the appropriate state slot in our filter – is straightforward since sources correspond to registered users and/or customers. The amount of information required per source (one additional number) and the number of potential sources itself is low enough to accommodate most operator's need without having to rely on aggregation.

4.1.3 Defining features

The choice of which call features to use in our SPIT filter is crucial for its performance. While the SPIT filter is theoretically guaranteed to work with any choice of single feature or combination of features

(under which the distributions for SPIT and NON-SPIT are non-identical), for practical reasons features with the following properties are highly desired:

- good separation of the distributions p_{SPIT} and $p_{\text{NON-SPIT}}$ as quantified by the Kullback-Leibler information numbers κ_1, κ_0 from Eqs. (13)-(14). This ensures that the filter will be able to stop a source from sending out further SPIT as quickly as possible.
- hard to manipulate for spitters
- availability of data, e.g., from old logfiles
- easy access to the feature during runtime, meaning that the feature should be easily observable during normal operation without requiring extra machinery.

We believe that in this regard a good choice of features in SPIT detection are features which capture the *reaction of users* to SPIT rather than features that capture the technical properties of SPIT bots. Indeed, a SPIT call is: (1) undesirable and has likely shorter duration, as the call would be hanged up by the callee with higher probability; (2) likely to be playing back a pre-recorded message such that double-talk[5] may occur; (3) unexpected with possibly longer ringing time and a higher rate of unanswered/refused calls; (4) automated with likely shorter time-to-speech and fewer pause during the call. Although these features are more likely to be affected by cultural or social habits, they are much harder to manipulate for a spitter than features such as inter-arrival time or port number. They are also less likely to be affected by the technical characteristics of one specific botnet, and thus could more easily take the moving nature of SPIT attacks into account.

In this paper, we argue that *call duration* might be a good feature (also because it simplifies calculation).

Of course, other choices of features are also possible. In fact, the theoretical framework in Section 3 allows one to do feature selection. In practice, one would thus start by identifying a set of all possible candidate features. Given data, one would then compute the κ_0 and κ_1 Kullback-Leibler information number either from parametric density estimation (as shown below in Section 4.2), or in more complicated cases from non-parametric density estimation such as, e.g., histograms. Knowing the respective κ_1, κ_0 allows one to rank the subsets and ultimately to select the features that achieve minimum expected regret, as the worst-case false positive and false negative rates can be explicitly computed using the equations presented in Section 3.3.

4.1.4 Defining decisions

Finally we have to talk about the actions the SPIT filter can take. In Section 3 we have assumed that once a source has been identified as a SPIT bot, all subsequent calls are to be blocked. And conversely, once a source has been identified as a regular user, all subsequent calls are to be allowed through. It is clear that in practice this decision rule alone will not be sufficient. However, recall from the beginning of this section that our SPIT filter is meant to be only one particular module in a larger SPIT prevention system (see Figure 5). Thus the outcome of the SPRT should be seen as another feature by itself, based on which a higher-level decision-making policy would then act. (The specific details of this high-level decision-making policy are outside the scope of the paper.)

4.2 Example: learning the distribution from labeled data

Maximum likelihood estimation is one standard tool from statistics to learn distributions from labeled data. Assume we are given n calls either all labeled as SPIT or all labeled as NON-SPIT (without loss of generality we assume they are all SPIT). To estimate the distribution p_{SPIT} necessary to perform the SPRT, we proceed as follows. First, we extract the feature representation from each call, yielding $x_1, \ldots, x_n$. Next, we make an assumption about the form of p_{SPIT}; for example, in this paper we

[5] Double-talk means that caller and callee talk at the same time. As is described in [20], this can be computed directly from the packet header information and does not require expensive processing of the voice stream.

assume that x_i is the call duration and that we believe that an exponential distribution with (unknown) parameter λ would describe the data well. To find the parameter λ that best explains the data (under the assumption that the data is i.i.d. drawn from an exponential distribution) we then consider the likelihood of the data as function of λ and maximize it (or equivalently, its logarithm):

$$\max_{\lambda>0} \mathcal{L}(\lambda) := \log p(x_1, \ldots, x_n|\lambda) \tag{27}$$

$$= \log\Big(\prod_{i=1}^{n} p(x_i|\lambda)\Big) = \sum_{i=1}^{n} \log\big(\lambda \exp(-\lambda x_i)\big)$$

$$= n\log(\lambda) - \lambda \sum_{i=1}^{n} x_i. \tag{28}$$

The best parameter λ_{ML} is then found by equating the derivative of $\mathcal{L}(\lambda)$ with zero, yielding $\frac{n}{\lambda} - \sum x_i = 0$, or

$$\lambda_{\mathrm{ML}} = \frac{n}{\sum_{i=1}^{n} x_i}. \tag{29}$$

To run the SPIT filter we would thus take $p(x|\mathrm{SPIT}) := \lambda_{\mathrm{ML}} \exp(-\lambda_{\mathrm{ML}} x)$ in Eq. (21), with $p(x|\mathrm{NONSPIT})$ learned analogously.

It should be noted that while the above procedure leading to Eq. (29) is very basic and is only applicable for the exponential distribution we consider in the paper, the maximum likelihood procedure itself is more widely applicable and can be used to also fit more accurate but also more complex density models to the data (e.g., mixture models). For further information on this subject, we refer the interested to the vast literature on density estimation in statistics and machine learning.

4.3 Evaluation with learned distributions

As said above, in the real world we cannot assume that we know the generating distributions p_{SPIT} and $p_{\text{NON-SPIT}}$. Instead we have to build a reasonable estimate for the distributions from labeled data. The natural question we have to answer then is: what happens if the learned distribution used in the SPIT filter does not exactly match the true but unknown distribution generating the data we observe (remember, the case where they do match was examined in Section 3.5).

To examine this point with real-world data, we used call data from 106 subjects collected from mobile phones over several months by the MIT Media Lab and made publicly available[6] in [3]. The dataset gives detailed information for each call and comprises about 100,000 regular voice calls. Ideally we would have liked to perform the evaluation based on real-world data for both SPIT and NON-SPIT. Unfortunately, this dataset only contains information about regular calls and not SPIT—and at the time of writing, no other such dataset for SPIT is publicly available.[7] In the following we will again consider call duration as feature for our filter to discriminate SPIT from NON-SPIT. To obtain SPIT calls from the dataset, we artificially divide it into two smaller datasets: one that is designated as SPIT and one that is designated as NON-SPIT. The set of SPIT calls is obtained by taking 20% of all calls whose call duration is <80 seconds, the remaining calls are assigned to the set of NON-SPIT calls. Having thus prepared the data, our general experimental procedure is as follows (also refer to Figure 2):

1. We first need to estimate the generating distributions p_{SPIT} and $p_{\text{NON-SPIT}}$. To do this, we assume that the estimates, $\hat{p}_{\mathrm{SPIT}}$ and $\hat{p}_{\mathrm{NONSPIT}}$, are exponential distributions the parameters of which can be estimated via maximum likelihood as in Eq. (29). In the remainder, we then use the estimate $\hat{p}_{\mathrm{SPIT}}$ and $\hat{p}_{\mathrm{NONSPIT}}$ as a proxy for the true but unknown distribution p_{SPIT} and $p_{\text{NON-SPIT}}$.

[6] `http://reality.media.mit.edu/download.php`

[7] The earlier work described in [10] set out to precisely change that. In it the authors describe a methodology for creating SPIT traffic and also provide a common data set for the use in benchmark comparisons. However, the data set they provide is generated from "emulated users based on a social model"; in essence, the authors use common tools to generate the SPIT traffic, where the relevant features, such as call duration, inter-arrival time, behavior upon receiving a call, etc. are all modeled by sampling from distributions. For example, the call duration was generated from an exponential distribution the parameter of which was specified by hand (which amounts to the same as what we do here).

Note that because the true distribution generating the data is unlikely to be exactly an exponential, we will introduce an estimation error which can negatively affect the performance of the SPIT filter (the theoretical bounds we derived in Section 3 only apply if the data is generated from the true distribution). However, if the true distribution is "close" to an exponential, then we can expect that the result obtained from using only the learned distribution will also be "close". In Figure 6 we give a visual comparison of the data and the fitted model; the figure shows a histogram plot of the actual distribution of call duration in the data and the idealized distribution from the fitted model. As can be glanced from the figure, the fit is good but not perfect: in particular for the NON-SPIT case the model underestimates calls with a short duration (which will negatively affect the performance of the SPRT filter by making optimization select too optimistic error thresholds).

2. Performing this step, we obtain the exponential distributions $\hat{p}_{\text{SPIT}}$ with mean $\lambda_0^{-1} = 30.23$ seconds and $\hat{p}_{\text{NONSPIT}}$ with mean $\lambda_1^{-1} = 129.64$ seconds so that the ratio becomes $\lambda_1/\lambda_0 = 0.23$.

3. From this $\hat{p}_{\text{SPIT}}$ and $\hat{p}_{\text{NONSPIT}}$ we then compute κ_0, κ_1 from Eqs. (22)-(23).

4. We then systematically examine the behavior of the SPIT filter when we vary the remaining design-specific cost parameters c_1, c_2, N ($c_0 = 0$) and prior probability $p(\text{SPIT})$. For each setting of these parameters, the following steps are repeated:

   (a) We first compute the α^*, β^* that is optimal for each particular setting by solving numerically Eq. (20) using MATLAB's inbuilt interior point solver.

   (b) We then simulate the SPIT filter by running 1,000,000 independent trials for this setting. Each such trial consists of first randomly determining the type of the source (a Bernoulli event generated from the prior probability $p(\text{SPIT})$) and then drawing calls uniformly at random from the corresponding dataset we prepared above. The average success rate obtained over all these trials is then reported in Table 3.

The results show that overall the performance depends on two factors: the prior probability of a source being SPIT and the choice of the cost terms c_1, c_2 together with the number of calls N. If the prior probability for SPIT is very small (i.e., we expect that the majority of sources is NON-SPIT), the optimization procedure automatically selects a higher and less accurate threshold α for SPIT and, similarly, a lower and thus more accurate threshold β for NON-SPIT. This in turn increases the error rate for SPIT but decreases the error rate for NON-SPIT. For example, for $N = 20$ and $c_2 = 100c_1$, in the case $p(\text{SPIT}) = 0.5$ (i.e., 50% of all sources are SPIT), the average relative error of the filter for SPIT is 0.16% and the average relative error for NON-SPIT is 1.94%. The same situation for a world where $p(\text{SPIT}) = 0.01$ (i.e., 1% of all sources are SPIT), the average relative error for SPIT increases to 6.29% and the average relative error for NON-SPIT decreases to 0.32%. The second factor affecting performance is the choice of the cost terms which dictate if, all else being equal, it is more important to avoid erroneously accepting SPIT or erroneously blocking NON-SPIT.

Finally, we can see from these empirical results on real data that the empirical error rate and stopping time is (in some cases notable) higher than what we would have expected from the theoretical analysis in Section 3. This, however, should not come as a surprise. The reason for this discrepancy is of course that in our experiments the true distribution generating the data is unknown and that the estimated distribution we use as its surrogate does not perfectly agree with it (see Figure 6). Still, we can see that the performance degrades rather gracefully in the estimation error and is still quite accurate. In practice, one would also use more sophisticated (and more accurate) methods to estimate the underlying distributions from the data.

5 Summary

In this paper, we presented the first theoretical approach to SPIT filtering that is based on a rigorous mathematical formulation of the underlying problem and, in consequence, allows one to derive performance guarantees in terms of worst case cumulative misclassification cost (the expected loss) and thus,

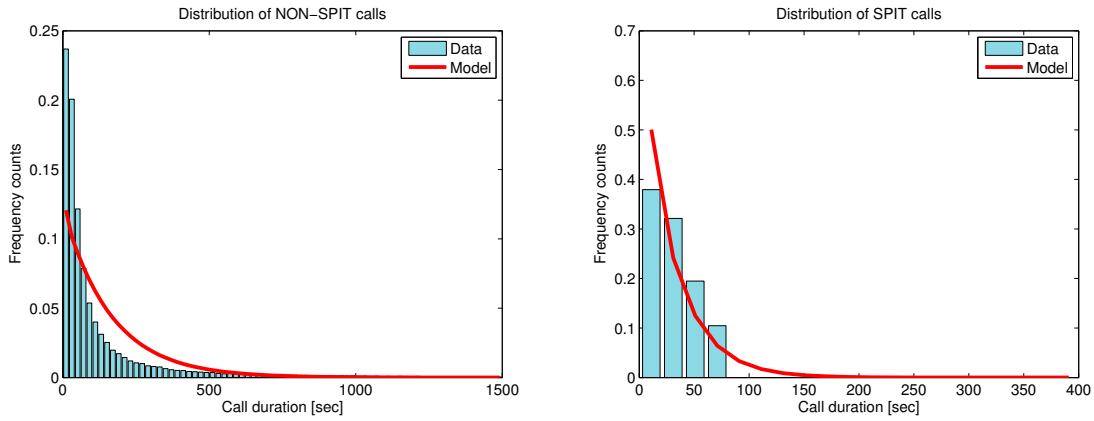


Figure 6: Histogram of the distribution of the feature "call duration" in the data set for NON-SPIT calls (left panel) and SPIT calls (right panel). The red curve shows the frequency count that we would have obtained if the data were truly generated by an exponential distribution with parameter λ_{ML} learned via maximum likelihood (see Section 4.2). The figure shows to what extent the real-world data and the learned distribution model agree; for example, it can be seen that in the NON-SPIT case the model fits the data not perfectly and underestimates calls with short duration (which in turn affects the SPRT filter and makes optimization select too optimistic error thresholds).

		$p(\mathrm{SPIT}) = 0.5$				$p(\mathrm{SPIT}) = 0.01$			
		$c_2 = c_1$	$c_2 = 2c_1$	$c_2 = 5c_1$	$c_2 = 100c_1$	$c_2 = c_1$	$c_2 = 2c_1$	$c_2 = 5c_1$	$c_2 = 100c_1$
N=5	α^*	1.00e-06	1.00e-06	1.00e-06	4.61e-02	4.56e-02	4.31e-02	4.40e-02	4.38e-02
	β^*	1.00e-01	9.09e-02	5.43e-02	6.49e-03	6.61e-03	6.65e-03	6.54e-03	6.69e-03
	$T/\mathbb{E}[T]$	3.4/3.3	3.4/3.3	3.8/3.7	4.1/3.1	3.3/1.1	3.3/1.1	3.3/1.1	3.3/1.1
	RErr (SPIT)	11.247%	12.315%	23.752%	87.108%	87.280%	89.652%	86.753%	88.650%
	RErr (NON)	26.259%	25.157%	18.221%	1.844%	1.927%	1.965%	1.897%	1.959%
N=10	α^*	1.00e-06	1.00e-06	1.00e-06	1.00e-01	1.00e-01	1.00e-01	1.00e-01	1.74e-03
	β^*	1.00e-01	3.21e-02	1.65e-02	8.60e-05	8.68e-04	1.74e-04	8.69e-05	9.56e-05
	$T/\mathbb{E}[T]$	4.1/3.3	5.1/4.1	5.6/4.	6.9/4.	4.1/0.8	4.2/0.8	4.3/0.9	5.5/2.3
	RErr (SPIT)	0.030%	0.272%	0.788%	70.421%	20.134%	53.386%	70.780%	68.744%
	RErr (NON)	33.439%	20.856%	15.883%	0.671%	3.539%	1.149%	0.624%	0.624%
N=15	α^*	1.00e-06	1.00e-06	1.00e-06	1.00e-01	9.98e-02	1.00e-01	1.00e-01	1.00e-01
	β^*	8.79e-02	1.87e-02	9.45e-03	5.78e-05	5.84e-04	1.17e-04	5.84e-05	1.00e-06
	$T/\mathbb{E}[T]$	4.5/3.4	6.0/4.4	6.5/4.8	7.9/5.0	4.5/0.8	4.6/0.9	4.7/0.9	4.8/0.9
	RErr (SPIT)	0.000%	0.000%	0.006%	6.593%	1.034%	3.448%	6.442%	63.821%
	RErr (NON)	33.480%	18.918%	14.569%	1.642%	4.638%	2.211%	1.600%	0.096%
N=20	α^*	1.00e-06	1.00e-06	1.00e-06	1.00e-01	1.00e-06	6.50e-02	9.99e-02	1.00e-01
	β^*	6.35e-02	1.31e-02	6.60e-03	4.35e-05	6.70e-04	9.55e-05	4.39e-05	1.00e-06
	$T/\mathbb{E}[T]$	5.0/3.6	6.5/4.6	7.1/5.0	8.1/5.1	8.78/4.7	5.0/1.0	4.8/0.9	4.9/0.9
	RErr (SPIT)	0.000%	0.000%	0.000%	0.163%	0.000%	0.000%	0.209%	6.293%
	RErr (NON)	30.660%	17.132%	13.134%	1.940%	5.679%	2.516%	1.857%	0.320%
N=100	α^*	1.00e-06	1.00e-06	1.00e-06	1.00e-06	1.01e-06	1.01e-06	1.01e-06	1.00e-06
	β^*	1.13e-02	2.25e-03	1.13e-03	1.13e-05	1.14e-04	2.28e-05	1.14e-05	1.00e-06
	$T/\mathbb{E}[T]$	6.9/4.7	8.2/5.6	8.6/6.0	11.5/8.5	9.7/4.7	10.0/4.7	10.1/4.7	10.2/4.7
	RErr (SPIT)	0.000%	0.000%	0.000%	0.000%	0.000%	0.000%	0.000%	0.000%
	RErr (NON)	16.886%	9.432%	7.659%	1.458%	3.166%	1.923%	1.476%	0.610%

Table 3: Examining the empirical performance of the SPRT-based SPIT filter on the data set when systematically varying the design parameters $c_2 = k \cdot c_1$ and N. Two scenarios are considered: one where the prior p_{SPIT} is set to 0.5 (50% of all sources are SPIT), and one where it is set to 0.01 (1% of all sources are SPIT). Each entry in the table consists of five values: α^*, β^* are the parameters minimizing the loss; T is the empirical stopping time (followed by the expected stopping time); and RErr is the relative error for SPIT and NON-SPIT (i.e., what percentage of SPIT calls was erroneously accepted and what percentage of NON-SPIT calls was erroneously blocked).

on the number of samples that are required to establish with the required level of confidence that a source is indeed a spitter. The method is optimal under the assumption of knowing the generating distributions, does not rely on manual tuning and tweaking of parameters, and is computationally simple and scalable. These are desirable features that make it a component of choice in a larger, autonomic framework.

Moreover, we have outlined the procedure that needs to be followed to apply this SPIT filter as an *outbound* filter in a realistic SPIT prevention system, including which potential call features to use and how the best feature could be found from the candidates via automated feature selection. In particular, we have sketched how the generating distributions can be learned from data. The difficulty of the problem of successfully detecting SPIT is then only related to how similar/dissimilar the generating distributions are. This difficulty can be quantitatively expressed in terms of the Kullback-Leibler information numbers κ_1, κ_0—which in turn can be calculated analytically or approximately from the learned distributions. Taken together this means that the worst case performance of the SPIT filter can be computed in real-world operation (and can thus be potentially used to tune the other hyperparameters of the whole system).

Our experimental evaluation verifies that our approach is feasible, efficient ("efficient" meaning that only very few calls need to be observed from a source to identify SPIT), and able to produce highly accurate results even when the generating distribution is not *a priori* specified but inferred from data.

Acknowledgements

Sylvain Martin acknowledges the financial support of the Belgian National Fund of Scientific Research (FNRS). Tobias Jung acknowledges financial support from a research fellowship of ULg. This work is also partially funded by EU project ResumeNet, FP7–224619.

References

[1] N. Chaisamran, T. Okuda, G. Blanc, and S. Yamaguchi. Trust-based voip spam detection based on call duration and human relationships. In *Proc. of the 11th Int. Symp. on Applications and the Internet (SAINT)*, 2011.

[2] D. E. Duffy, A. A. Mcintosh, M. Rosenstein, and W. Willinger. Statistical analysis of ccsn/ss7 traffic data from working ccs subnetworks. *IEEE JSAC*, 1994.

[3] N. Eagle, A. Pentland, and D. Lazer. Inferring social network structure using mobile phone data. *Proceedings of the National Academy of Sciences (PNAS)*, 106(36):15274–15278, 2009.

[4] D. Geneiatakis and C. Lambrinoudakis. A lightweight protection mechanism against signaling attacks in a sip-based voip environment. *Telecommunication Systems*, 36(4):153–159, 2008.

[5] T. Jung, S. Martin, D. Ernst, and G. Leduc. SPRT for SPIT: Using the sequential probability ratio test for spam in VoIP prevention. In *Proc. of 6th Int. Conf. on Autonomous Infrastructure, Management and Security (AIMS)*, Lecture Notes in Computer Science. Springer, 2012.

[6] P. Kolan and R. Dantu. Socio-technical defense against voice spamming. In *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 2007.

[7] M. Nassar, O. Dabbebi, R. Badonnel, and O. Festor. Risk management in voip infrastructure using support vector machines. In *International conference on Network and Service Management (CNSM'10)*, pages 48–55, 2010.

[8] M. Nassar, S. Martin, G. Leduc, and O. Festor. Using decision trees for generating adaptive spit signatures. In *Proc. of the 4th International Conference on Security of Information and Networks (SIN 2011)*, 2011.

[9] M. Nassar, R. State, and O. Festor. Monitoring sip traffic using support vector machines. In *Proceedings of the 11th international symposium on Recent Advances in Intrusion Detection*, RAID '08, pages 311–330, Berlin, Heidelberg, 2008. Springer-Verlag.

[10] M. Nassar, R. State, and O. Festor. Labeled VoIP data-set for intrusion detection evaluation. In *In: Proceedings of the 16th EUNICE/IFIP WG 6.6*, 2010.

[11] J. Quittek, S. Niccolini, S. Tartarelli, M. Stiemerling, M. Brunner, and T. Ewald. Detecting SPIT calls by checking human communication patterns. In *IEEE International Conference on Communications (ICC 2007)*, June 2007.

[12] K. Rieck, S. Wahl, P. Laskov, P. Domschitz, and K.-R. Müller. Self-learning system for detection of anomalous sip messages. In *Principles, Systems and Applications of IP Telecommunications, 2nd International Conference, IPTComm (2008)*, pages 90–106, 2008.

[13] R. Schlegel, S. Niccolini, S. Tartarelli, and M. Brunner. SPIT prevention framework. In *IEEE GLOBECOM'06*, pages 1–6, 2006.

[14] D. Shin, J. Ahn, and C. Shim. Progressive multi gray-leveling: a voice spam protection algorithm. *IEEE Network*, 20:18–24, 2006.

[15] Y. Soupionis, G. Marias, S. Ehlert, Y. Rebahi, S. Dritsas, M. Theoharidou, G. Tountas, D. Gritzalis, A. Bergmann, T. Golubenco, and M. Hoffmann. SPAM over Internet telephony Detection sERvice final report. `http://projectspider.org/documents/Spider_D4.2_public.pdf`, Sep 2008.

[16] Y. Soupionis, G. Tountas, and D. Gritzalis. Audio CAPTCHA for SIP-based VoIP. In *Emerging Challenges for Security, Privacy and Trust*, volume 297 of *IFIP Advances in Information and Communication Technology*, pages 25–38, 2009.

[17] J. Sterbenz, D. Hutchison, E. K. Çetinkaya, A. Jabbar, J. P. Rohrer, M. Schöller, and P. Smith. Resilience and survivability in communication networks: Strategies, principles, and survey of disciplines. *Computer Networks*, 54:1245–1265, June 2010.

[18] A. Wald. Sequential tests of statistical hypotheses. *Annals of Mathematical Statistics*, 16:117–186, 1945.

[19] A. Wald and J. Wolfowitz. Optimum character of the sequential probability test. *Annals of Mathematical Statistics*, 19:326–339, 1948.

[20] C.-C. Wu, K.-T. Chen, Y.-C. Chang, and C.-L. Lei. Detecting voip traffic based on human conversation patterns. In Henning Schulzrinne, Radu State, and Saverio Niccolini, editors, *Principles, Systems and Applications of IP Telecommunications. Services and Security for Next Generation Networks*, volume 5310 of *Lecture Notes in Computer Science*, pages 280–295. Springer Berlin / Heidelberg, 2008.

[21] Y.-S. Wu, S. Bagchi, N. Singh, and R. Wita. Spam detection in voice-over-ip calls through semi-supervised clustering. In *Proceedings of the 2009 Dependable Systems Networks*, pages 307–316, 2009.

[22] H. Yan, K. Sripanidkulchai, H. Zhang, Z.-Y. Shae, and D. Saha. Incorporating active fingerprinting into spit prevention systems. In *Third annual security workshop (VSW'06)*, 2006.

[23] G. Zhang, S. Ehlert, T. Magedanz, and D. Sisalem. Denial of service attack and prevention on sip voip infrastructures using dns flooding. In *Principles, Systems and Applications of IP Telecommunications, 1st International Conference, IPTComm (2007)*, 2007.

[24] G. Zhang, S. Fischer-Hübner, and S. Ehlert. Blocking attacks on sip voip proxies caused by external processing. *Telecommunication Systems*, 45(1):61–76, 2010.