Session Privacy Enhancement by Traffic Dispersion

Haim Zlatokrilov
School of computer science
Tel-Aviv University
Tel-Aviv, Israel
zlatokri@post.tau.ac.il

Hanoch Levy
School of computer science
Tel-Aviv University
Tel-Aviv, Israel
hanoch@cs.tau.ac.il

Abstract— **Traditional network routing uses the single (shortest) path paradigm. This paradigm leaves the session vulnerable to a variety of security threats, such as eavesdropping. We propose to overcome this via dispersive routing, conducted over multiple paths. This increases significantly the costs inflicted on an attacker who wishes to eavesdrop sessions by hijacking network links (or routers). We formulate the Security Traffic Manager (STM) problem (route session fragments[1], over multiple paths, so that protection against an attacker, with a known hijacking budget, is guaranteed) and the attacker problem (find the cheapest hijacking strategy). The problems are analyzed for cases in which the attacker must eavesdrop all the fragments as well for cases in which it must eavesdrop only a fraction of them. We analyze the theoretical complexity of these problems and offer algorithms for finding dispersive routes that guarantee security. Though some theoretical cases of the problem are shown to be NP-Hard, typical practical cases can be solved by polynomial time algorithms. We extend the STM problem to more practical situations where the goal of the STM is to guarantee privacy, using minimal number of limited-length paths. The algorithms are tested through simulation and shown to be efficient in many scenarios. The model and algorithms offered in this study can be used for deploying a "session enhanced security" service in packet networks[2].**

Keywords-component; traffic dipersion, security, eavesdrop, multi-path routing

I. INTRODUCTION

Security is becoming a major concern in today's networks. The traditional single, usually shortest, path routing paradigm leaves sessions vulnerable to attacks along the route. Attackers can eavesdrop sessions, drop and modify their packets and perform denial-of-service (DoS) attacks by attacking nodes or links along the path.

Protection against eavesdropping attacks is essential for providing privacy. Eavesdropping can be considered as the "passive" version of the DoS "active" attack. While there are many methods for protection against DoS attacks (e.g. rerouting, blocking etc.), eavesdropping attack cannot be usually detected by the parties and thus left without any active defense. The common techniques for providing privacy, such as end-to-end encryption and authentication, are considered to provide good privacy, but once the adversary cracks the encryption or obtains the keys in some other ways, its road for obtaining sensitive information is clear. A lot of effort is invested these days on security against internal-attackers, such as organization's employees. These attackers may have inside information and access to network's physical infrastructure and network elements. Thus, encryption and password-protected techniques might not provide sufficient protection against such attackers. For some applications encryption may have implementation difficulties due to increased delay, fragments size, etc. Also, encryption does not always protect against man-in-the-middle attacks, in which a device along the route eavesdrops all fragments and can pretend to be the other party for both participants. In such cases, the attacker will be able to obtain and modify the information without the awareness of session participants.

This work explores and studies a new paradigm for enhancing traffic privacy. The approach exploits the natural structure of the network and uses traffic dispersion to enhance privacy. This can be used as the only means of protection or as an additional protection layer (added to encryption). The idea is that while encryption is a good defense against attackers that managed to eavesdrop the entire session, dispersing session fragments along multiple paths aims at preventing the attacker from conducting a meaningful eavesdrop in the first place.

Our objective in this paper is to study the problem of the Security Traffic Manager (STM) whose goal is to make it hard on the attacker to eavesdrop a session. We use a worst-case approach in which the STM assumes that the attacker knows the path taken by each fragment, either by some implicit knowledge or by a lucky guess.

We model the problem by associating each network link with some arbitrary adversary hijacking cost. This cost may vary across links and reflects the differences between them and is a function of the physical-geographical conditions (e.g., enterprise-internal link vs. enterprise-external WAN link), physical link properties (e.g. wire vs. wireless), network element properties (e.g. the operating system), etc. We assume that the attacker is capable of collecting session packets crossing several links and reconstructing the session from them. Though in practice the coordination of the hijacked information may pose additional costs, for the sake of simplicity we assume that the hijacking costs are additive.

Using disjoint paths, as proposed in [3] and [9], can be used as one method for providing security. Our model makes neither this requirement nor other requirements on the paths taken in the dispersive routing. This reflects many practical setups in

[1] We use "fragments" as a general term to denote pieces of a session which can be packets in IP network, cells in ATM, etc.

[2] This work was partially supported by Israeli Science Foundation grant 235/03 and by Euro NGI network of excellence.

which the hijacking costs of links near the source and the destination are likely to be high due to physical protection provided by the organization. However, the number of these links is expected to be low. In contrast, the hijacking cost in the middle of the network may be low while the number of links one can use is relatively high.

The model and algorithms presented in this work can be used as a basis for offering an enhanced security service. Such service can provide additional layer of security for many applications. It can be offered in private managed networks, by ISPs in the Internet or even in overlay network architecture. Throughout the paper we will present some practical considerations for implementation of such service. The practical implementation of routing session packets over multiple (not necessarily the shortest) paths can be carried out in variety of methods such as source routing, tunneling, static route configuration in MPLS/ATM, and by overlay peer-to-peer networks above IP networks (including the Internet). We assume that the STM can plan and execute the transmission of session fragments over the multiple paths, without relying on specific machinery.

Successful eavesdropping attack may require the interception of all fragments in some cases or only a fraction of them in other cases (e.g. V^2oIP applications). Our model and study cover both cases explicitly by accounting for the fraction of fragments that need be captured as a parameter. For the sake of simplicity we neglect coding issues and assume that all fragments carry equally important information.

Our model does not account for the capacity of the network links. The reason is that in most cases only a small fraction of the sessions routed over the network requires enhanced security service. Moreover, such service should be given upon demand. Thus, we neglect bandwidth constraints, assuming that session's bandwidth requirements are small in comparison to network capacity.

Our study focuses first on the (pure) STM problem. The problem is how to route the session fragments over multiple paths as to assure that the capturing cost of the session (that is, capturing a fraction q of its fragments) exceeds a certain (attacker) budget B. To this end we first pose a "dual" problem, named the *attacker problem:* Given the routing of a session fragments over the network, the link capturing costs and the fraction q of fragments that need be captured, which links should the attacker capture to minimize the session capturing cost. The attacker problem is addressed in Section III where we prove that the problem is in NPC and provide approximations to several cases of the problem. This result will be used in section IV in the analysis of the STM problem.

Having addressed the attacker problem, we then (Section IV) address the (pure) STM problem. We provide polynomial time algorithms for the STM problem for the case in which the attacker must eavesdrop all session fragments, and for the case in which the attacker must eavesdrop only fraction q of the fragments but the network has uniform link costs. We show that the general STM problem, in which the attacker must eavesdrop only a fraction of the fragments and the network has non-uniform link costs, is NP-Hard. Moreover, checking the correctness of the output generated by the STM algorithm (the attacker problem) is NPC. However, in many practical cases the attacker's budget will be small in comparison to the maximal cost the STM can pose on the attacker. In this case a polynomial time algorithm, proposed in this study, can provide a solution. Our set of algorithms, called *CASH_FLOW*, uses a concept of transforming link costs to link capacities and uses this to find expensive-to-hijack routes and packet flows.

Dispersing session fragments over multiple paths might cause degradation of QoS due to jitter and out-of-order effects (but can also increase performance in some scenarios [25]). When security is of high priority these drawbacks can be fixed by mechanisms such as increased buffer sizes at the source and the destination nodes. To address QoS and practical aspects of the problem we extend the STM problem. In addition to guaranteeing security we add the requirement of using minimal number of length-limited paths. Adding this requirement is proper in many real-life settings in which the attacker's budget is limited and thus the STM does not need to use many paths. In such cases we seek dispersion strategies that guarantee protection, while in addition use minimal number of paths (to reduce operational costs and fragment disordering) and length limited paths (to limit maximal delay as to provide good QoS). For this problem we propose two algorithms, the first has low complexity in comparison to the second algorithm, while the second may provide a better solution (smaller number of paths). This analysis is provided in Section V. Simulation results examining the quality of the algorithms developed are then presented in Section VI. Summary and concluding remarks are given in Section VII.

It is important to note that throughout the paper we address the STM problem in the context of eavesdropping attacks. Nonetheless, the model holds for protection against malicious dropping attacks [17] as well. The main difference between the problems would be the fraction of fragments the STM is required to protect. For example, if eavesdropping attack requires capturing fraction q=0.7 of the fragments (a reasonable number for VoIP applications), then a DoS attack can be considered to require the dropping of roughly a fraction (1-q)=0.3 of the fragments. Though theoretically the problem may improve reliability, practically there are other better active ways to deal with DoS (such as rerouting the fragments as discussed in [16], etc).

A. Related Work

Multi-path routing has many implications on network efficiency and Quality of Service (QoS). Multi-path routing was studied mostly in the context of load-balancing [8][23], reliability [19], network efficiency [13][8][21] and QoS routing [11][22]. In contrast to these studies, we focus on the context of dispersing the packets of a *single session* for enhancing privacy.

The idea of dispersion for security purposes was presented in [3][4][9][12][17] and was used in other technologies for security purposes, e.g. frequency hopping in radio networks. In [3] routing over disjoint paths was proposed as a security mechanism for perfectly secure communication in the absence of secure infrastructure.

In [17] multi-path routing was proposed as a technique for tackling DoS attacks. The problem addressed in [17] was to minimize the maximal damage incurred by a single-link attack (or failure). For that purpose it was suggested to use a *secure multi-path approach* and address the problem through distributed routing decisions with bandwidth-constraints adaptation. In [12] *secure stochastic routing* is presented. This approach explores the existence of multiple paths by flooding and forces packets to take alternate paths probabilistically. In

[4] the problem of finding routing policies that are robust to interception attacks are formulated as a zero-sum game between the designer and an adversary. A set of games and strategies was presented in that study.

Many studies were conducted on tackling DoS attacks. Some of the approaches focus on specific scenarios, network architecture and filtering such as Secure Overlay Services (SOS) [16]. Overlay network, such as in Resilient Overlay Networks (RON) [2], were suggested as a mechanisms for recovering from link outage and other network failures by taking advantage of the dense overlay connectivity. The problem of engineering attack-resistant networks was presented in [6]. However, DoS attacks are usually tackled by active response when the attack occurs. Eavesdropping attacks cannot be addressed by these mechanisms since they cannot be detected.

At first glance, the problem addressed in this paper may look similar to some versions of load-balancing, QoS routing and other problems. However, since the objective of the problem we address differs from that of those problems, it requires special theoretical and practical analysis.

II. A Brief Description of the Problem

Given are a network and the hijacking cost of each link. The STM's objective is to find a set of paths and assignment of the session's N fragments so that the attacker will have to invest at least budget B in order to capture a fraction q of the fragments. The modeling assumptions that lead to the model are described in the introduction.

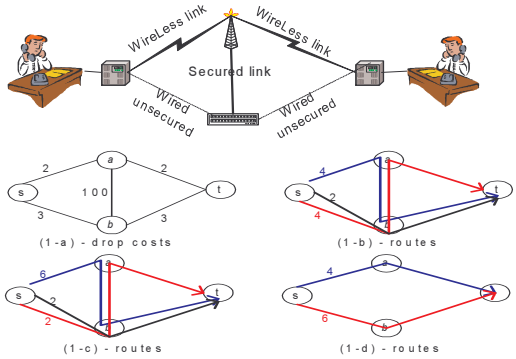


Figure 1. An example of the STM problem.

To demonstrate the problem, consider the network depicted in Figure 1 over which two parties desire to conduct a VoIP session. Figure 1-a, depicts the network where the numbers represent link hijacking costs. The STM's objective is to transfer a session consisting of 10 fragments from s to t, such that the attacker will incur a cost of at least 5 units in order to drop 8 fragments. If the fragments are directed as shown in Figure 1-b or Figure 1-c, the attacker will be able to drop 8 fragments at the cost of 4 units by hijacking links $s \rightarrow a$ and $a \rightarrow t$ (or link $b \rightarrow t$ in Figure 1-c). In contrast, using the dispersive routing strategy depicted in Figure 1-d will keep the session safe. Note that N can be generalized to any number of fragments, and the amount of fragments shipped over the paths, in the example, can be translated to fractions of N.

One should note that we are dealing with an offline, centralized[3] problem where the STM can deterministically dictate the path taken by each fragment. Also, throughout the paper we study the model in which only the links are associated with hijacking costs; the model and the algorithms presented can be easily extended to incorporate node costs.

III. The Attacker Problem

In this section we formalize and analyze the attacker problem. The understanding of this problem is essential for the fundamental understanding of the STM problem, which is the center of this work. The analysis of the attacker problem will be used as a basis for the analysis of the STM problem.

The attacker's objective is to eavesdrop a session, for which purpose he/she must hijack network links, each characterized by its hijacking cost. To succeed in an eavesdrop attack, the attacker must intercept a fraction $0 < q \leq 1$ of the session fragments. The fraction depends on the application being eavesdropped.

When session fragments are dispersed over several paths, the attacker may be required to hijack several links, collect the fragments and reconstruct the session. Assuming that he/she has full knowledge regarding the path taken by each fragment, the attacker's objective is to hijack a minimal-cost set of links whose links jointly carry qN unique fragments[4].

The assumption that the attacker has the knowledge regarding the path taken by each fragment is derived from the STM's requirement to guarantee session protection in the worst-case scenarios. In practice the attacker may not have this knowledge. However, the STM must protect the session against "lucky" attackers as well.

In this paper we focus on the STM problem. We use the attacker's problem mainly to demonstrate the complexity of the problem faced by the STM. The intuition is that if the attacker problem is hard, the STM problem is much harder, since its objective is to produce inputs for the attacker problem. Moreover, if the STM wishes to validate that the session is secured it will face the attacker problem.

We will now provide the formal description of the attacker problem and prove it to be NPC, even for q=1 and uniform links costs. We show that there is no fixed constant approximation algorithm to the attacker problem. Then we will present the approximation ratios to the attacker problem, which we directly derive from the approximation ratios to the *Partial-Weighted-Set-Cover* (*PWST*) problem. Due to the direct mapping between the problems, these approximations can be considered as the best available approximations for the attacker problem. These approximations imply that there is no polynomial time algorithm, which can be used by the STM to evaluate whether a dispersion strategy guarantees privacy (find the minimal hijacking budget imposed on the attacker). For that reason and for a better understanding of the problem, we have formulated the attacker problem as a linear programming problem. Due to lack of space we omit the details of the formulation and present them in [27]. This formulation can be used to solve the attacker problem by a Linear Programming (LP) solver. The latter technique will be used in an algorithm solving the STM problem.

The formal description of the attacker problem is as follows:

[3] The STM is familiar with full network topology

[4] By the term "unique fragments" we mean that a fragment is counted only once even if the attacker obtained it several times.

INSTANCE: Graph $G(V,L)$, cost c_j for each link $l_j \in L$, a set P of paths from s to t ($P = \{p_1..p_K\}$), the path taken by each of the N fragments and a parameter q.

QUESTION: Find a set of links $L' \subseteq L$, such that the number of unique fragments on L' is greater than or equal to qN and such that the cost of L', $\sum_{l_j \in L'} c_j$, is minimal.

Theorem 1:For any $0 < q \le 1$ the attacker problem is NPC. This holds even when link costs are uniform.

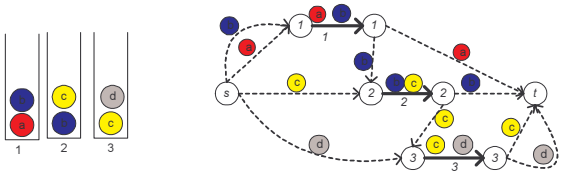
Proof: It is easy to see that the attacker problem is NP. To show that it is NPC, we will show a polynomial reduction from the *PWST* problem, known to be NPC [20], to the attacker problem. The *PWST* problem is:

INSTANCE: Finite set A, finite cover of subsets $F = \{F_1,..,F_n\}$ of A with positive costs $\{c_1,..,c_n\}$, and $0 < q \le 1$.

QUESTION: Find a q-partial cover (a cover of $q|A|$ elements in A) $F' \subseteq F$ of A whose cost (sum of costs of the subsets in F') is minimal.

Our reduction will show that given an instance of a *PWST* problem (where $|A|=N$), we can construct a graph $G'(V,L)$ and an assignment of N fragments to paths from s to t, such that one can solve the attacker problem in G', if and only if, he can solve the *PWST* problem.

The construction of G' is as follows: Construct a set of $|F|$ vertex disjoint links where, for each $F_j \in F$, construct a link l_j, whose cost is c_j. For each $a \in A$, let $\{F_{a_1},..,F_{a_M}\} \subseteq \{F_1,..,F_n\}$ be the subsets to which a belongs. Then, for each a construct in G' a path p_a, from s to t, consisting of the M links $l_{a_1},..,l_{a_M}$ and of additional M+1 new links (denoted by $l_{b_1},..,l_{b_{M+1}}$) whose cost is $c_{max} = max\{c_1,..,c_n\}$, which are used to connect s, $l_{a_1},..,l_{a_M}$ and t. Clearly, the construction is done in polynomial time. The construction is illustrated in Figure 2 where (a) depicts the initial *PWST* problem and (b) depicts the reduction to the attacker problem.



(a) (b)

Figure 2. Reduction from *PWST* to the attacker problem.

We will show that for any solution in cost B of the *PWST* there is a solution in cost B for the attacker problem in G'. Then we show that any optimal solution of the attacker problem, with cost B, in G', can be mapped to a solution of the *PWST* problem with the same cost B. These two properties together imply that the latter mapped solution of the *PWST* is an optimal solution to that problem which will complete the proof of the reduction.

i) First we show that for any solution, in cost B, of the *PWST* there is a solution cost B for the attacker problem in G'. Let $F' \subset F$ denote the subsets in the solution for the *PWST* problem. To obtain a solution to the attacker problem in G', one shall use the links L', corresponding to the subsets in F'. The subsets in F' cover $q|A|$ elements. Let $A' \in A$ denote the elements covered by F'. Clearly, $|A'|=q|A|=qN$. The construction of G' guarantees that the number of paths using the links in L' is equal to the number of elements covered by F'. Thus, qN paths cross the links in L'. Note that each path carries exactly one fragment. Therefore, qN fragments cross L', as required. The cost of the *PWST* problem is $B = \sum_{l \in F'} c_l$. Since $\sum_{l \in F'} c_l = \sum_{l \in L'} c_l$, B is also the attacker's cost in G'.

ii) Second we will show that optimal solution of the attacker problem, with cost B, in G', can be mapped to a solution of the *PWST* problem with the same cost B.

Let L' denote the set of links in the optimal attacker problem. The attacker's cost is then $B = \sum_{l \in L'} c_l$. If L' contains links that do not correspond to subsets in F (that is, these are the links is the set $\{l_{b_1},..,l_{b_{M+1}}\}$), each such link l_{b_i} carries at most one fragment and its cost is the maximal link cost. We replace each such link in L' by one of its adjacent links, l_{a_j}, that contains the fragment sent over l_{b_i}, to construct a new link set L''. Clearly L'' eavesdrops at least qN fragments and its cost is $B = \sum_{l \in L'} c_l$ (since links $l_{b_1},..,l_{b_{M+1}}$ have maximal cost and replacing them cannot increase B, further it cannot decrease B, since B is the optimal solution). A solution to the *PWST* is then the subset F'' that corresponds to the links in L''. Since qN paths cross the links in L'', F'' contains the corresponding $q|A|=qN$ elements, and $B = \sum_{l \in F'} c_l = \sum_{l \in L'} c_l$.

iii) Lastly, from *i*) and *ii*) it is implied that the mapped solution (F'') found in *ii*) must be an optimal solution of the *PWST* problem.

This completes the proof of the reduction and thus the attacker problem is NPC.

Remark: Note that the problem remains NPC even for the case of identical link costs, q=1 (the basic *unweighted set cover* problem [20]) and N is polynomial in the size of the network.■

Now we will show that not only is the attacker problem NPC, the attacker's minimal cost cannot be approximated within a constant approximation ratio to the optimal cost.

Theorem 2: Let B_{opt} be the cost of the optimal solution to the attacker problem. Then for any fixed constant $k > 1$, it is NP-hard to approximate the attacker's cost. That is, no polynomial time algorithm A can guarantee finding a solution whose cost B_A obeys: $B_A < k\, B_{opt}$.

Proof: It is enough to show that a polynomial-time approximation algorithm A to the attacker problem, that guarantees approximation ratio in constant factor $k > 1$, will lead to a polynomial-time algorithm for approximating *PWST* with the same constant factor k.

Base on the reduction in **Theorem 1**: it is clear that if there exists algorithm A that solves the attacker problem in cost B_A and guarantees an approximation ratio that obeys $B_A < k\, B_{opt}$, this algorithm can be used to solve the *PWST* in cost $B'_A \le B_A$, by using the method presented in **Theorem 1** part *ii*). Since we

showed that the minimal cost in the attacker problem is similar to the cost in the *PWST*, by using this method, algorithm A leads to an approximation ratio $B'_A < k\ B_{opt}$ of the *PWST* problem.■

Theorem 3: A greedy algorithm can approximate the attacker problem with the following performance ratios (where B_{opt} denotes the optimal solution and B_A denotes the solution found by the greedy algorithm and $H()$ is the harmonic function ($H(x) \approx ln(x)+1$)) :

i) $B_A / B_{opt} \le H(3+2|N|)$ for q<1 and non-uniform link costs

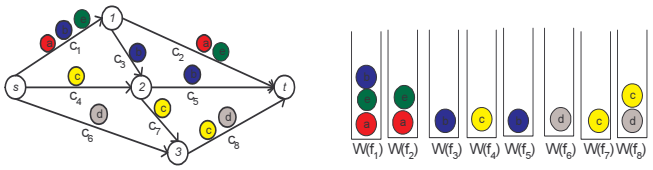
ii) $B_A / B_{opt} \le H(q|N|)$ for q<1 and uniform link costs

iii) $B_A / B_{opt} \le H(|P|)$ for q=1, where P is the number of paths in the attacker problem

Proof: We will prove **Theorem 3** by showing that any approximation ratio to the *PWST* can be used to approximate the attacker problem. We prove this by translating the attacker problem to an instance of *PWST* in polynomial time. The solution to the attacker problem is simply deduced from the translation.

The attacker problem can be mapped to the *PWST* problem by the following translation:

Let A be a finite set of N elements. Set q in the *PWST*, to the fraction of fragments, q, in the attacker problem. For each link l_j that carries at least one fragment, construct a set F_j. The cost of F_j is set to the cost of link l_j in the G, c_j. Each fragment i is translated to an element $a_i \in A$. Element $a_i \subseteq F_j$ if fragment i is shipped over link l_j. The construction is illustrated in Figure 3 where (a) depicts an instance of the attacker problem, which is mapped to the *PWST* problem in Figure 3(b).



(a) (b)

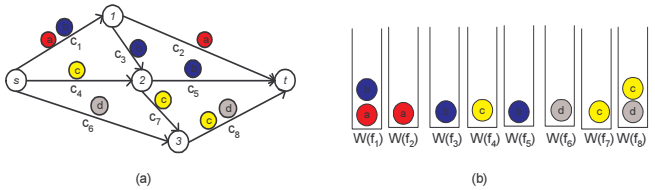
Figure 3. Translation of attacker problem (in (a)) to fragment cover problem (in (b)); circles represent fragments

To this end note that the links l_j, their costs c_j and their fragment contents are mapped one-to-one to the sets F_j, the costs c_j and their element contents. The same solution of the first problem maps, one-to-one to a solution of the second. Finding the minimal *PWST* and using the links corresponding to the chosen sets, provides a solution to the attacker problem.

The approximation ratio to the *PWST* problem by a greedy algorithm [20] is known to be $H(3+2|N|)$. The attacker problem with q<1 and non-uniform link costs is mapped to this problem and thus *i*) stands. The approximation ratio to the (*non-weighted*) *partial-set-cover* problem is known to be $H(q|N|)$. The attacker problem with q<1 and uniform link costs is mapped to this problem and thus *ii*) stands.

To deal with *iii*) we note that the attacker problem maps to the (*non-partial*) *weighted-set-cover*, which is known to be have $H(|N|)$ approximation ratio. We now show a better approximation of $H(|P|)$,where P is the number of paths in the attacker problem, to this case.

For q=1, the attacker must hijack at least one link on every path used by the fragments. Thus, we can aggregate fragments shipped on the same path to a single fragment. Clearly, any solution to the initial problem is a solution to the aggregated fragment problem, and vice-versa. Now, each distinct path will carry a single fragment and the number of fragment N is reduced to P. Since the approximation ratio of the corresponding *weighted-set-cover* is now $H(|P|)$, it is also an approximation ratio to the attacker problem with q=1, which proves *iii*).



(a) (b)

Figure 4. : Mapping of attacker problem, with q=1 to *weighted-set-cover* for finding optimal solution: (a) attacker problem, (b) mapping to the *weighted-set-cover* problem (circles represent paths)

The mapping of *iii*) is depicted in Figure 4 for the same problem given in the example in Figure 3. In the example, the attacker problem contains 6 elements shipped over 5 paths. Using the first mapping, results in $|A|$=6 in the *weighted-set-cover* problem. Aggregating the elements results in $|A|$=5 elements. In this particular example, the number of paths is in the order of the number of fragments, however in most cases $P<<N$ and thus the approximation improves considerably.■

IV. THE STM PROBLEM

The pure STM problem is to ship session fragments from source to destination in a way that will force the attacker to invest at least cost B in order to successfully eavesdrop the session. We assume the worst-case scenario in which the attacker has the full knowledge about the path taken by each fragment. The reason to this assumption is that the STM aims at guaranteeing privacy even for the most pessimistic cases.

In this section we formalize and analyze the STM problem. We show that the general case of the STM problem is at least NP-Hard. Nonetheless, we prove that polynomial time algorithms can solve some particular cases. To deal with the NP-Hard cases we present a heuristic algorithm. The main idea of the algorithms is to use a flow algorithm to ship "*cost units*" through the network whose link capacities represent the link costs; the total "value" of these units is set to the attacker budget. The paths where the *cost units* go through provide an indication where one would like to ship the fragments. The fraction of *cost units* on each path in the flow indicates the fraction of fragments to be shipped on each path. Thus, a second phase of the algorithm translates the costs units to fragments units, which leads to a legal flow of fragments.

Remark: For the sake of convenience throughout the discussion we assume that N is large enough such that any fraction α of N encountered in the analysis, αN, is an integer. In practice this is the case for most problems, especially if N reflects the number of bits in the session and the STM can place a variable number of bits on a packet. Nonetheless, the analysis can be repeated for smaller values of N at the expense of more detailed analysis with marginal change of the results.

A. *Formulation of the STM problem*

The formal description of the STM problem is as follows:

INSTANCE: A directed or undirected graph $G(V,L)$, cost c_j for all links $l_j \in L$, source and destination nodes (s and t respectively), a parameter $0<q\leq 1$ and a constant $B>0$.

QUESTION: Find a set of paths P from s to t and an assignment of the N fragments to the paths such that: There exists no set of links L', obeying $\sum_{l_j \in L'} c_j < B$, that captures qN fragments.

We classify the problem into three categories:

i) q=1: We propose a polynomial time algorithm, *CASH_FLOW_1* and prove that is solves the problem.

ii) q<1 and the network links have identical costs: We propose a polynomial time algorithm, *CASH_FLOW_2* and prove that is solves the problem.

iii) q<1 and the network links have non-identical costs (the general STM problem): we prove that the problem is at least NP-Hard. We propose *CASH_FLOW_3*, which is based on a heuristics, to solve the problem.

B. The STM problem with q=1

In this special case, the attacker must eavesdrop all session fragments. This can model a flow of a data application, in which all packets must be eavesdropped by the attacker in order to reconstruct the information.

We propose a simple polynomial time algorithm that guarantees to solve the STM problem if a solution exists and otherwise return negative answer.

CASH_FLOW_1

1: **INPUT**: graph $G(V,L)$ with cost c_j for every link l_j
2: Construct a new graph $G'(V,L)$, set the capacity c'_j on link l_j, to be the value $c'_j=c_j$
3: F = Find max-flow/min-cut in G'
4: **if** $F \leq B$
5: **return** negative result and stop
6: Let f_p denote the flow on path p, in F
7: Assign N'_p fragments on path p, where $N'_p = Nf_p / F$.
8: **return** success

The idea of the algorithm is to translate the costs to capacities and ship "cost units", instead of fragments. If B cost units can be shipped from s to t a solution to the STM problem is constructed based on the flow F. Otherwise there is no solution since the attacker will always be able to hijack all links in the min-cut, and the STM cannot guarantee protection for the session. The complexity of this algorithm is identical to that of the max-flow algorithm [1].

Theorem 4: *CASH_FLOW_1* finds a solution for the STM problem with q=1, if such a solution exists. The algorithm returns a negative result if no solution exist.

Proof: In Step 2 the link costs are translated to link capacities. The Max-Flow/Min-Cut algorithm finds the cut with minimal capacity. Since capacity is now a synonym to cost, in Step 3 the algorithm finds the Minimal Cost Cut (MCC). If the attacker's budget B>MCC, the attacker will always be able to hijack all links in the MCC and thus conduct a successful attack regardless of how the STM will transfer the session. Thus, if B>MCC the algorithm returns with a negative answer (Step 5). Otherwise (if $F>B$), flow through every cut (including MCC) is greater than B. Then, Step 7 provides that any non-empty link l_j of G' is also non-empty on G and that guarantees that the cost of capturing all traffic is greater than B.

Note that the algorithm does not deal with specific fragments but only with fractions sent on each path in the flow. Thus, N can be any number as long as $\forall p \quad N \geq F / f_p$ (each path must carry at least one fragment to maintain the solution correctness).■

One should note that if the condition $\forall p \quad N \geq F / f_p$ does not hold (the sessions are very short), the problem becomes NPC. The proof is based on reduction from the 3SAT problem (similar construction to the one in [24], the details can be found in [27]).

Since we use the basic max-flow algorithm, the *CASH_FLOW_1* algorithm works both for directed and undirected network models (where the latter is transformed to a directed graph via standard techniques). The rest of the algorithms in this paper are based on a similar concept. Thus, we will refer to networks as directed or undirected graphs and the standard transformation can be used when needed.

Also note that the attacker can use *CASH_FLOW_1* to find the MCC. If the attacker's budget $B \geq$ MCC, he/she can capture the MCC and prevail. If MCC>B, then either the attacker has no ability of capturing the session (if the STM used *CASH_FLOW_1*) or he/she is facing an NPC problem as presented in Section II.

C. The STM problem with q<1 and uniform link costs

In this case the attacker needs to eavesdrop a fraction q<1 and all network links have identical hijacking cost $c_j=c$. This case can model the defense against eavesdropping attack on voice or video applications (it can also apply in many cases for data transfer applications such as FTP) in networks with no diversity with respect to link security properties.

We propose a simple polynomial time algorithm for solving this problem:

CASH_FLOW_2

1: INPUT: graph $G(V,L)$ with cost $c_j=c$ for every link l_j
2: Construct a new graph $G'(V,L)$, set the capacity on link l_j, to be the value $c'=qc$
3: F = Find max-flow/min-cut in G'
4: **if** $F \leq B^*$ $(B^* = B - (B \bmod c))$
5: **return** negative result
6: Assign N'_j fragments on link l_j, where $N'_j = Nf_j / F$ and f_j denotes the flow on link l_j, in F.
7: **return** success

The idea behind *CASH_FLOW_2* is to deal with cost units and then translate the fraction of costs units to the fraction of fragments to be shipped on the paths, similarly to *CASH_FLOW_1*. In order to get a valid solution we limit the link capacity, in cost units, to qc. This will guarantee the correctness of the solution. The complexity of this algorithm is identical to that of the max-flow algorithm.

Theorem 5: *CASH_FLOW_2* finds a solution for the STM problem with $q \leq 1$ and uniform link costs, if such solution exists. If no solution exists the algorithm returns a negative answer.

Proof: First we prove that if the algorithm finds a solution it is a valid solution: The capturing cost per fragment on each link l_j in the solution is $c/(f_j N/F) = cF/f_j N$. If B/c is not an integer, the leftover budget $B \bmod c$ cannot be used to hijack links. Thus, $B^*=B-(B \bmod c)$ is the effective budget for the attacker. The maximal number of fragments that can be eavesdropped using hijacking budget B is then:

$B^*/(cF/f_jN) = B^* f_j N/cF < qN$ (where the inequality is due to $f_j \le qc$ and $F>B^*$, which result from the construction of G') and the STM prevails.

Second, we show that if the algorithm fails (Step 5) there exists no solution for the STM. Consider the min cut and let L' denote the set of links in the min cut. If the algorithm fails, we have $B \ge q \sum_{l_j \in L'} c = qc|L'|$. Thus the attacker can hijack $B^*/c \ge q|L'|$ links. Now, for any way that the STM chooses to direct the flow, the attacker can sort the min cut links by their flow level and select the B^*/c highest flow links. This provides that he/she captures at least $(B^*/c)\cdot(N/|L'|)$ of the traffic, which is greater than $(q|L'|)\cdot(N/|L'|)=qN$ of the flow.■

D. The STM problem with q<1 and non-uniform link costs

In this section we discuss the general STM problem in which the attacker must drop fraction $q<1$ of session fragments and link hijacking costs are non-uniform. This case models the defense for a variety of applications shipped over a network with links having diverse security properties. We show that the problem is at least NP-hard and propose a heuristic for solving the problem.

Theorem 6: The STM problem with $q<1$ and non-uniform link costs is at least NP-hard.

Proof: In section III we showed a mapping from the attacker problem to the *PWST* problem. That is, we can transform the attacker problem, by a polynomial algorithm, to the *PWST* problem. In the theory of complexity, the set of problems that generate hard inputs (an input that makes it hard for any algorithm to solve the problem) to NPC problems, are known to be at least NP-Hard (see [15]). We now can look at the STM problem as the problem of producing hard inputs to the *PWST*. Since this problem is NPC, the STM problem is at least NP-Hard.■

For this case of the STM problem we propose the following algorithm:

CASH_FLOW_3

1: Examine feasibility: *run CASH_FLOW_1* with q=1, if it fails in Step 5 **return** negative result.
2: Construct a new graph $G'(V,L)$, set the capacity c'_j on link l_j, to be $c'_j=qc_j$, where c_j is the cost of l_j in G.
3: F = Find max-flow/min-cut in G'
4: Assign fragments such that the number of fragments on link l N'_j, is $N'_j = Nf_j/F$ (f_j is the flow on link l_j in F)
5: **if** $F<B$
6: The fragment assignment found in (4) might not guarantee against an attack
7: **if** $F \ge B$ the fragment assignment found in (4) guarantees against an attack.

The idea behind *CASH_FLOW_3* is similar to the idea behind *CASH_FLOW_2* extended to networks with non-uniform link costs. The complexity of this algorithm is identical to that of the max-flow algorithm.

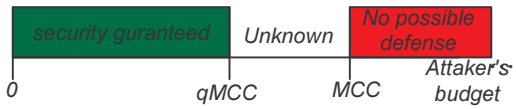


Figure 5. STM problem as a function of Minimal-cost-cut (MCC) and hijacking budget B

Theorem 7: *CASH_FLOW_3* possesses the following properties (as summarized in Figure 5) for the STM problem with $q \le 1$ and non-uniform link costs:

i) If there exists no valid solution (if $B>$MCC), the algorithm returns negative result and no possible solution exists.

ii) If $B<q$MCC, the algorithm finds guaranteed solution to the STM problem.

iii) If qMCC$<B<$MCC, the algorithm produces a valid solution (fragment assigment) that may or may not guarantee against an attack. Checking whether this solution is good is NPC (the attacker problem).

Proof: If the attacker's budget obeys $B>$MCC, the attacker can always hijack the links of the MCC (computed by *CASH_FLOW_1*, in Step 1) to eavesdrop all fragments. This proves property *i*).

If $F \ge B$ in Step 4, the flow F found in Step 3 guarantees that the attacker's minimal eavesdropping cost is greater than B. Note that in this case $B<q$MCC (because in the construction of G' in Step 2, we limit the capacity on each link to qc_j). The cost per fragment on each link l_j in the solution, after Step 4, is $c_j/(f_j N/F) = Fc_j/f_j N$. Thus, the maximal number of fragments that can be eavesdropped using hijacking budget B is: $B/(c_j F/f_j N) = BNf_j/Fc_j < qN$ (where the inequality is due to $f_j \le qc_j$ and $F>B$, which result from the construction of G). This proves property *ii*).

If qMCC$<B<$MCC, it is not clear whether the assignment of fragments (in Step (4)) guarantees against an attack. Moreover, checking the correctness of the solution found (the attacker problem) is an NPC problem by itself and the approximations to the attacker problem are not tight.■

As discussed above, the cases in which the attacker's hijacking budget is qMCC$<B<$MCC are very hard to solve for the STM, mainly because it leaves little room for solutions for the STM problem. Nonetheless, we believe that in practice the STM should protect itself, in most cases, against hijacking budget B, which is less than qMCC, especially in large networks (where qMCC can be very high). This brings us to the practical problem analyzed in the next section.

Remark: One should note that the model deals with link hijacking. However the model and the algorithms presented above can be extended to deal with nodes by the standard method of splitting a node into two nodes and setting the node's hijacking cost as the cost of the link connecting the two nodes.

V. THE STM PROBLEM WITH MINIMAL NUMBER OF LENGTH LIMITED PATHS (PL STM)

In the previous sections we analyzed the *pure* STM problem. We believe that in many practical setups, the

attacker's budget against which the STM wishes to guarantee security will be significantly low in comparison to the cost of the minimal-cost-cut. In these cases the STM can guarantee session privacy (as shown in Section IV) and can further improve dispersion strategies by accounting for additional quality aspects in addition to guaranteeing security. Such factors are: 1) The operating costs which can be reduced by decreasing the number of paths and 2) The application QoS that can improve by limiting the paths lengths. We thus propose to use a minimal set of length-limited paths. We will refer to the STM problem with minimal number of length-limited paths as the PL (Path Limited) STM problem.

We will show that the PL STM problem is NPC regardless of the value of q and the values of the link costs. We will then provide two heuristic algorithms. The complexity of the first algorithm (*CASH_FLOW_AUGMENT*) is lower than that of the second (*PL_VALIDATE*), while the second may find a better solution (smaller number of paths).

A. Formulation of the PL STM problem

The formal description of the PL STM problem is as follows:

INSTANCE: A directed or undirected graph $G(V,L)$, cost c_j for all links $l_j \in L$, source and destination nodes (s and t respectively), a parameter $0 < q \le 1$, a constant $B>0$, and constant a D.

QUESTION: Find a minimal cardinality set of paths P from s to t, the length of each $p_i \in P$ is limited to D, and an assignment of the N fragments to the paths such that: There exists no set of links L', obeying $\sum_{l_j \in L'} c_j < B$, that captures qN fragments.

Theorem 7: The PL STM problem is NPC (even for q=1 and uniform link costs).

Proof: We will carry out the proof by showing that the particular case of the PL STM with uniform link costs is NPC. In particular, we prove that by solving the PL STM problem in a network with uniform link costs in polynomial time, one can solve the K^{th} Shortest Path problem (are there K or more link disjoint paths from s to t, each having total length of D' or less) on a graph with uniform link costs, by a polynomial time algorithm. The proof then follows from the fact that when the link costs are uniform the K^{th} Shortest Path problem is NPC.

Claim 1: If there is a solution to the PL STM problem in a network with uniform link costs, where the link costs are c, it **must** use exactly $\lceil \lfloor B/c \rfloor / q \rceil + 1$ length limited, link disjoint paths.

Proof of claim 1: First we show that given $\lceil \lfloor B/c \rfloor / q \rceil + 1$ link disjoint length limited paths, the STM can guarantee session security: Given $\lceil \lfloor B/c \rfloor / q \rceil + 1$ link disjoint paths the STM should spread the fragments equally over the paths. That is, ship exactly $N/(\lceil \lfloor B/c \rfloor / q \rceil + 1)$ fragments on each path. In this case, the attacker, using the budget B, can hijack at most $\lfloor B/c \rfloor$ links. The number of fragments on these links is at most:

$$\lfloor B/c \rfloor \cdot N/(\lceil \lfloor B/c \rfloor / q \rceil + 1) \le \lceil \lfloor B/c \rfloor \cdot N/((\lfloor B/c \rfloor / q) + 1) \rceil$$
$$\le \lceil \lfloor B/c \rfloor \cdot N/(\lfloor B/c \rfloor / q) \rceil \le qN.$$

That is, the maximal number of eavesdropped fragments in budget B, is less than qN as required.

Now we will show that there is no solution that uses less than $\lceil \lfloor B/c \rfloor / q \rceil + 1$ paths and that a solution that uses exactly this amount of paths, must use paths that are link disjoint: Let us assume that there is a solution that uses less than $\lceil \lfloor B/c \rfloor / q \rceil + 1$ paths. That is, at most $\lceil \lfloor B/c \rfloor / q \rceil$ paths are used. The attacker can now sort the paths by the number of fragments shipped over them and hijack the $\lfloor B/c \rfloor$ links with the highest amount of fragments (one link in each path). The number of eavesdropped fragments is at least $\lfloor B/c \rfloor \cdot \lceil N/(\lfloor B/c \rfloor / q) \rceil = qN$ (since on each of these paths at least $\lceil N/(\lfloor B/c \rfloor / q) \rceil$ fragments are shipped). As we see, using less than $\lceil \lfloor B/c \rfloor / q \rceil + 1$ paths, does not guarantee security for an attacker with budget B, that wishes to eavesdrop qN unique fragments.

Second, in case the $\lceil \lfloor B/c \rfloor / q \rceil + 1$ paths are not link disjoint, there exists some link l_j and there are at least two paths p_i and p_j that use link l_j. Now the attacker can account for p_i and p_j as if they were a single path (since it can capture both paths via l_j). Thus the attacker practically faces $\lceil \lfloor B/c \rfloor / q \rceil$ paths and as we saw above, in this case the attacker can eavesdrop qN fragments with budget B. That is, there must be $\lceil \lfloor B/c \rfloor / q \rceil + 1$ link disjoint paths in the optimal solution (in case there are less length limited, link disjoint paths in the network, there is no solution to the PL STM that guarantees session security). By this we conclude the proof of *claim 1*.■

Now we will conduct a polynomial reduction from the K^{th} Shortest Path problem to the PL STM problem. Given an instance of the K^{th} Shortest Path on graph G with uniform link lengths, consider the PL STM problem as follows:

1. PL STM problem with c=1 for all links, q=1, $D=D'$ and $B=K-1$ (Let K' denote the number of paths in the solution).
2. **if** no solution is found return a negative answer; otherwise return a positive answer.

Clearly, if the algorithm for solving PL STM is polynomial, then the above algorithm is polynomial as well.

First we show that if there is a solution to the PL STM it must contain K length limited disjoint paths. According to *Claim 1*, the optimal solution to the PL STM problem will result in $\lceil \lfloor B/c \rfloor / q \rceil + 1 = \lfloor B/1 \rfloor + 1 = B + 1$ length limited (to parameter $D=D'$) link disjoint paths. Since $B=K-1$, we have that the number of paths in the solution must be exactly K. In this case the answer to the K^{th} Shortest Path problem is positive.

Second, we show that if there is no solution to the PL STM problem, there are no K length-limited disjoint paths. This result derives immediately from *Claim 1*. If there is no solution to the PL STM problem there are less than $K=B+1$ length limited link disjoint paths and the result to the K^{th} shortest path is then negative.

This proves that if a polynomial algorithm for solving the PL STM problem exists, there is a polynomial time algorithm for solving the K^{th} Shortest Path. But since K^{th} Shortest Path is NPC, so is the PL STM problem.

Since we proved that the particular case of the PL STM is NPC, the general PL STM problem is NPC as well.■

Though we have proved that the PL STM problem is NPC, we will now show in the case of uniform link costs and $0 < q \le 1$, any algorithm for solving the k^{th} Shortest Path problem can be used to solve the PL STM problem. Further,

the PL STM problem can be solved in pseudo-polynomial time.

Theorem 8: Any algorithm that solves the K^{th} Shortest Path problem, can be used to solve the PL STM problem on networks with uniform link costs and $0 < q \leq 1$.

Proof: To find a solution for the PL STM problem we can use the algorithm for solving the K^{th} Shortest Path problem, to find K link disjoint paths, where $K=\lceil \lfloor B/c \rfloor / q \rceil + 1$. The correctness of this result follows directly from *Claim 1*. In case that the K^{th} Shortest Path returns negative results, there are less than K link disjoint length limited paths connecting s and t. In that case, according to *Claim 1*,there is no solution to the PL STM problem with budget B. In case that the algorithm returns positive result, the K disjoint paths can be used by the STM to ship the fragments, by equally spreading the fragments along the paths. As shown in *Claim 1* this proves that the attacker will be able to eavesdrop less than qN fragments using budget B.■

Corollary 1: The PL STM problem with uniform link costs and $0 < q \leq 1$ can be solved in pseudo-polynomial time in |V|, K and log(D), where D is the path length limit.

The corollary results from **Theorem 8** and from the fact that the K^{th} Shortest Path problem is known to be solved in pseudo-polynomial time in |V|, K and log(D). These algorithms can be used for solving the PL STM problem [10].

B. PL STM with non uniform link costs

This case of the PL STM problem is at least NP-hard because it is a generalization the *pure* STM problem with q<1 and non uniform link costs (which was proved to be NP-Hard in Section IV). It is, of course, also the generalization of the PL STM problem in networks with uniform link costs, which is also NPC as proved above.

Though the problem in NPC, in many practical cases a valid solution (dispersion strategy that guarantees security) can be found. The goal will be to find the optimal dispersion strategy in terms of minimal number of length-limited paths. We propose two algorithms for solving this problem. The algorithms are based on the idea of the *CASH_FLOW_3* algorithm, in which we attempt to ship a flow of B cost units from s to t. After finding a solution in terms of cost units we translate the relative number of cost units shipped over a path to the amount of fragments to be sent on it.

WidestLenLimitPath (G, D)

```
1: let set caps denote the sorted set (from high to low),
   without repetitions, of links residual capacities, cap[j] .
   Also assume the link are sorted by cap[j]
2: for all c in caps
3:     for all links l_j in G
4:         if cap[j] ≥ c then metric[j] = 1
6:         else metric [l] = D
8:     path = DIJKSTRA[5](s, t, metric)
9:     if( length(path) < D) then
10:        flow(path)= c
11:        return path
12: return no paths exists
```

[5] Run Dijkstara algorithm from s to t, over G where the weight of link lj is *metric*[j].

The first algorithm, *CASH_FLOW_AUGMENT* (for details see Appendix *A*), seeks a flow of B *cost units* using augmentation paths and then assigns the fragments over the paths that were used for the flow. For augmentation paths we select length-limited path whose residual capacity is the largest. The idea is that "wide" augmentation paths reduce the number of paths required to carry the flow. The procedure that computes the augmentation path, *WidestLenLimitPath* (see above), starts from an empty network and iteratively adds links according to their residual capacity until a path (that is also length-limited) is found. In case that the algorithm does not find any length limited augmentation path, the algorithm will stop and return negative answer. However, one should note that this is unlikely to occur since the initial assumption in the PL STM problem is that there should be a solution. The algorithm does not guarantee finding a valid solution even if one exists. However, when it finds a solution, security is guaranteed. In order to guarantee security, the algorithm limits the initial capacity of link l_j to qc_j, similarly to *CASH_FLOW_3* (see Section IV). The correctness of the algorithm, when a solution is found, can be easily proved similarly to the correctness of the *CASH_FLOW_3* algorithm.

Complexity analysis: The *CASH_FLOW_AUGMENT* first assigns link initial capacities (O(L)). Then, until finding a flow $F>B$, the algorithm may repeat calling *WidestLenLimitPath* at most B/q iterations. This is because in each iteration the minimal path capacity can be q since link capacities are a multiple of q (In case that the procedure *WidestLenLimitPath* will no find a path, *CASH_FLOW_AUGMENT* will stop and return negative result). To reach flow B, there will be at most B/q calls for *WidestLenLimitPath*. In each application of *WidestLenLimitPath* we first sort the links (O(Llog(L))) in Step 1. Then we add the links by residual capacities *cap*[j], from low to high, in O(L). For each iteration we might use DIJKSTRA algorithm (O(L^2)). That is, the complexity of *WidestLenLimitPath* procedure is then: O(Llog(L))+O(L^3)=O(L^3). Finally the total complexity of the *CASH_FLOW_AUGMENT* algorithm is O(L^3B/q).

Though the worst case complexity of the *CASH_FLOW_AUGMENT* algorithm is O(L^3B/q), one should note that since the algorithm is expected to find a result by using a small number of paths, the average complexity is expected to be significantly lower.

Remark: One should note that in our description of the model we consider equal-length links. However, the model can be extended to links with non-uniform lengths. This can be easily also translated to delay limited paths by replacing the link length by delay metrics.

In many cases, as will be shown in the simulation results (see Section VI), the *CASH_FLOW_AUGMENT* algorithm will increase unnecessarily the number of paths used. The reason is that in order to guarantee security, the algorithm limits the number of fragments that can be sent on each link, similarly to algorithm *CASH_FLOW_3* (the initial capacity of cost units of link l_j is qc_j). Reduction in number of fragments that can be sent on each link may, in many cases, increase the number of paths used for carrying the flow and also may limit the guaranteed minimal attacker's cost, i.e. the algorithm return negative result even though security can be guaranteed.

The PL STM problem is NP hard and there is no polynomial time algorithm that can calculate the optimal

solution[6]. We offer the second algorithm, *PL_VALIDATE,* that will provide better results (less paths) in practical cases, even at the expense of execution time. This algorithm can be used to find a solution for many practical cases and further will be used in this paper as a reference to which we compare the performance of the *CASH_FLOW_AUGMENT* algorithm.

To enhance the *CASH_FLOW_AUGMENT* algorithm one should not limit, or in some cases even artificially increase the link capacities. Of course, the problem with such an algorithm would be to calculate the attacker's cost in order to validate security is guaranteed, and that problem cannot be solved by a polynomial time algorithm (as discussed in Section III). To overcome this limitation, we propose using a linear programming solver (such as CPLEX [14]), to solve the attacker LP problem (see section III) and validate security.

The second algorithm we propose, *PL_VALIDATE* (for details see Appendix B), is based on iteratively executing *CASH_FLOW_AUGMENT* with varying values of q. The minimal attacker's cost is validated by a linear programming solver (not before we discrete the output of *CASH_FLOW_AUGMENT* to reduce computation time). We will show through simulations that in many practical scenarios this algorithm finds a solution with smaller number of paths in comparison to *CASH_FLOW_AUGMENT*, and its execution time is reasonable.

Observations about the *PL_VALIDATE* algorithm:

1. If the algorithm returns a positive result, the solution guarantees session security because the minimal attacker's cost is validated via linear programming. Also, the paths are length limited since the algorithm uses *CASH_FLOW_AUGMENT*, which uses only length limited paths.
2. The execution time of the linear programming validation can be enhanced by the a simple procedure of removing links that will not be used in the attacker's solution (remove link l_j if there exists a link l_i such that the same set of paths $p_i..p_j$ uses these links and $c_j \le c_i$). For details see Appendix B, procedure *RemoveUnusedLinks*.
3. There is a tradeoff between the quality of the solution (the number of paths being used) and execution time. This can be determined by setting the number of iterations made by the algorithm.
4. In *PL_VALIDATE*, we use a simple linear strategy for decreasing the parameter q. Other strategies can alternatively be used, e.g. non-linear decrease of q, setting q as a function of link cost c_j, etc.

VI. SIMULATION RESULTS

In this section we examine the quality of the proposed algorithms via simulation. The simulation is conducted on random networks. A network is constructed by providing the number of nodes |V| and the number of links |L| (the average degree is |L|/|V|). The links randomly connect between pairs of nodes. Source node s is randomly selected. Destination node t is selected by sorting the nodes by the order of shortest paths length from the source, and then picking the |V|/4 node. This selection of destination is used to analyze average cases and eliminate trivial scenarios of adjacent (or close) nodes and very distant nodes. Links are provided with integer hijacking costs in the range of 1 to 5 randomly. We believe that in practice there is no need for more than a countable number of different costs, since the number of parameters that affect security is limited. Links connected to source and destination are assigned with integer random hijacking costs in the range of 5 to 10, reflecting the belief that links near the source and destination can be better protected. This also eliminates trivial solutions for the attacker to hijack all links connecting to the source and/or the destination (this looks like a straightforward good selection for an attacker in case that all links are of equal or roughly equal costs).

Our network model differs from the hierarchical or self-similar network models proposed for the Internet, such as in BRITE [5]. This is because we believe that the privacy enhancement service presented in this paper is not limited to networks such as the Internet. On the contrary, private or overlay networks seem to be better candidates. In these networks we assume that the number of nodes is limited to 50 nodes and the topology is more random than hierarchical.

As mentioned, the model and the algorithms developed in this work are applicable both for directed and undirected network models. In the simulation we assume that the network is undirected and used the algorithms accordingly.

We first study the protection value of dispersive routing as a function of the node degree. We evaluate this via the maximal hijacking cost against which the STM can protect itself, and conduct the evaluation for $q=1$ and $q=0.7$. For $q=1$ this value is given by the MCC. For $q=0.7$ finding the exact minimal attacker's cost is NPC, and thus we use the highest budget value against which *CASH_FLOW_3* can guarantee protection, namely qMCC (recall from Section IV that in the latter case *CASH_FLOW_3* may be able to pose higher costs on the attacker, while guaranteeing, without validation, qMCC).

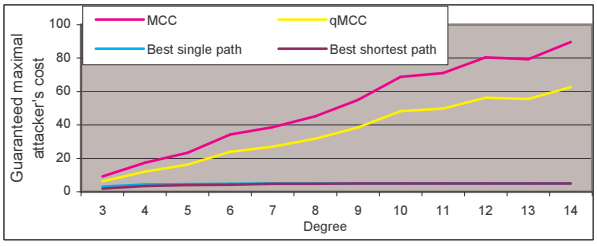


Figure 6. Protection Value (Attacker's Minimal guaranteed cost) as a function of node degree.

Figure 6 depicts the average (computed over 50 independent experiments for each degree) of these values as a function of network degree for networks with |V|=50 nodes. For the sake of comparison we also plot the max hijacking cost if a single, not necessarily shortest, path is used by STM (denoted "Best Single Path" in the figure), and the max hijacking cost if the shortest path is used (select the maximal cost path among all the shortest paths; denoted "best Shortest path" in the figure). As expected, the minimal cost inflicted on the attacker increases as a function of graph degree, since more paths can potentially be used by the STM for shipping the traffic. The increase is roughly linear. One may conclude from this experiment that traffic dispersion can highly increase the guaranteed attacker's cost for conducting an eavesdrop attack.

[6] The problem cannot be defined as a linear programming problem since it is not convex.

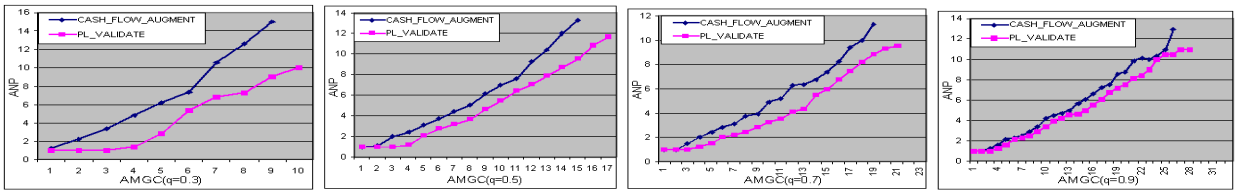


Figure 7. Average Number of Paths (ANP) as a function of Attacker's Miminal Guarenteed cost (AMGC) and parameter q (degree=4 and |V|=50)

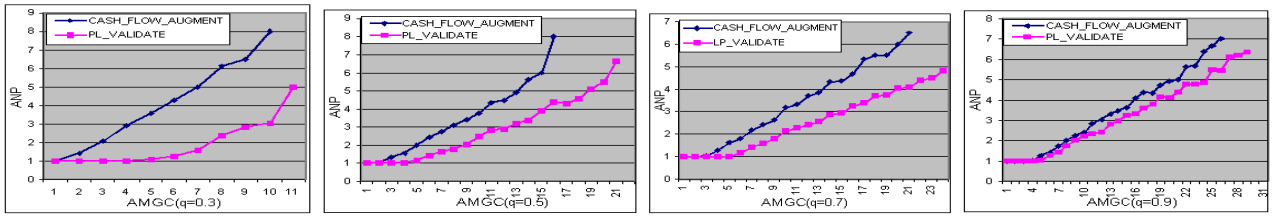


Figure 8. Average Number of Paths (ANP) as a function of Attacker's Miminal Guarenteed cost (AMGC) and parameter q (degree=4 and |V|=10)

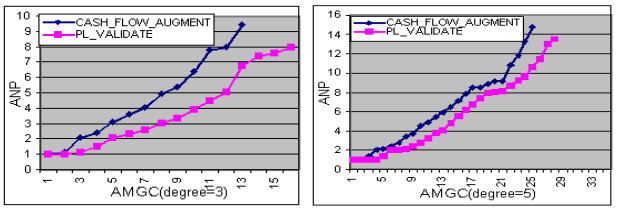


Figure 9. Average Number of Paths (ANP) as a function of Attacker's Miminal Guarenteed cost (AMGC) and degree (q=0.7 and $|V|$=50)

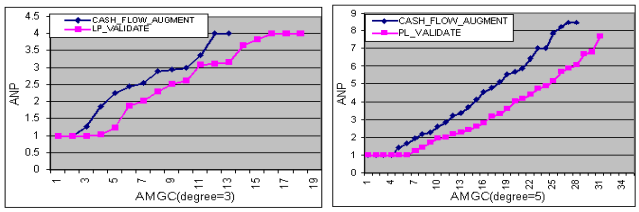


Figure 10. Average Number of Paths (ANP) as a function of Attacker's Miminal Guarenteed cost (AMGC) and degree (q=0.7 and $|V|$=10)

Second, we examine the quality of the solution produced by the algorithms for the PL STM problem proposed in Section V. We consider 50 independent networks and evaluate the number of paths used by the algorithms in order to guarantee protection, as a function of the attacker's budget[7]. The path length in each network was limited to be twice the length of the shortest path between s and t in that network. Figures 7 to 10 depict the simulation results. Figures 7 and 8 depict the simulation results using varying values of the parameter q (0.3, 0.5, 0.7 and 0.9), on networks with |V|=50 and |V|=10 respectively and degree=4. Figures 9 and 10 depict the simulation results under varying node degrees (3, and 5, the case of degree=4 is depicted in Figures 7 and 8 where q=0.7) for networks with |V|=50 and |V|=10 respectively, and q=0.7. Note that parameter q=0.7 can be a logical value for the case of guaranteeing security against eavesdropping on applications such and V^{2}oIP.

As expected, the average number of paths increases with the increase in the minimal guaranteed attacker's cost, in both algorithms. Note that the curves are monotonic non-decreasing. This results from the monotonicity that holds for each individual graph (increasing the number of paths will monotonically increase the attacker's minimal guaranteed cost).

From figures 7 to 10 we can see that *PL_VALIDATE* algorithm produces in most cases better results. The algorithm not only reduces the average number of paths but can also guarantee security against higher attacker's budget. However the gain in some cases is low and one should note that the execution time is much higher. Comparing the two algorithms we can roughly say that the reduction in the number paths decreases as parameter q is higher. The reason is that *CASH_FLOW_AUGMENT* utilizes better the links (the limitation of link capacity in the algorithm to qc_l has less effect). Also in case that the attacker's budget allows him to hijack small number of paths, *LP_VALIDATE* results in most cases in significantly higher guaranteed cost in comparison to *CASH_FLOW_AUGMENT*.

Lastly, we evaluate the execution time of *PL_VALIDATE*. Recall that the algorithm uses linear programming for checking the solution iteratively and this is the critical part of the algorithm consuming the largest amount of computation. To show that the algorithm can be executed in reasonable time, we present in Figure 11 the average execution time of a single execution of the LP procedure. In this test we used the AMPL CPLEX solver on 1.7Gh CPU processor on windows OS. The figure depicts the average execution time (taken over 100 randomly generated networks) of a single call to procedure *ValidateSolutionWithLP* as a function of the number of paths included in the validation procedure. The Figure shows the execution time for networks 50 nodes degree of 4 and 5. One should note that executing *PL_VALIDATE* requires several iterations before finding a solution. As can be seen, execution time increases in the number of paths in use; it however remains in the order of one second in the setups we simulated. This shows that though the problem is very complicated, many practical setups can be solved in reasonable amount of time using linear programming solvers.

[7] In order to conduct averaging in a consistent way, for each attacker cost the average number of paths is computed only over the networks for which a solution to the problem exists. The number of such networks was 50 for the low values of attacker cost, and decreased for higher values.

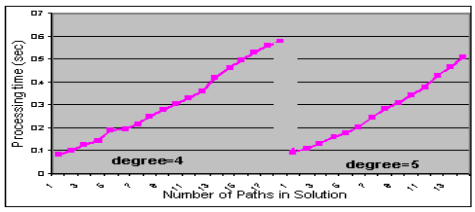


Figure 11. Average execution time of *ValidateSolutionWithLP*, as a function of the number of paths (|V|=50, degree=4, 5).

One should note that the STM could do its calculations in an off-line manner. This fact, combined with the fact that in many cases the applicability of traffic dispersion for security enhancement is for relatively small networks, imply that in most cases the problem has a practical solution.

VII. SUMMARY AND CONCLUSIONS

We proposed dispersive routing as a machinery for privacy enhancement by providing protection against eavesdropping attacks. We formulated the problem as a graph model where links or nodes are associated with hijacking costs. In this environment, we analyzed the STM problem, which is to guarantee session security against an attacker with a known hijacking budget. We analyzed the problem under the worst-case scenario in which we assumed that the attacker knows the dispersing strategy conducted by the STM.

We formulated the attacker problem and proved that it is in NPC and does not have algorithms that can guarantee good approximation. Though the attacker problem is NPC, and the STM problem seems to be harder, we proved that some cases of the STM problem can be solved by polynomial time algorithms. For the general STM problem (in a network with non-uniform link costs and $q<1$) we proved it to be at least NP-Hard. An algorithm to solve this problem was proposed. This algorithm guarantees security for any attacker's budget that is lower than qMCC (Minimal-Cost-Cut) but does not guarantee solution to costs higher than this.

We believe that in many practical cases the attacker's budget against which the STM must guarantee security will be less than qMCC. Thus, we extended the STM problem to account for QoS properties, which translates to the use of minimal number of length-limited paths (LP STM problem). For that problem we proposed two heuristics-based algorithms, one is relatively fast but may result in high number of paths, while the second can reduce the number of paths but requires significantly higher computation. We demonstrated the properties of heuristic-based algorithms using simulation results.

VIII. REFERENCES

[1] R. K Ahuja, T. L. agnanti, J. B Orlin "Network Flows - theory, algorithms and applications".

[2] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris. Resilient overlay networks. In Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP), October 2001.

[3] A. Bagchi, A. Chaudhary, M. T. Goodrich, S. Xu, "Constructing Disjoint Paths for Secure Communication", DISC 2003: 181-195.

[4] S. Bohacek, J. P. Hespanha, K. Obraczka, J. Lee, C. Lim, "Enhancing Security Via Stochastic Routing", In Proc. of the 11th IEEE ICCCN, May 2002.

[5] BRITE: Boston university Representative Internet Topology gEnerator, www.cs.bu.edu/brite/

[6] T. Bu, S. Norden, T. Woo, "Trading Resiliency for Security:Model and Algorithms", Proceedings of the 12th IEEE International Conference on Network Protocols (ICNP'04).

[7] J. Chen, P. Druschel, and D. Subramanian "A Simple Practical Distributed Multipath Routing Algorithm." Department of Computer Science, Rice University, TR98-320 (1998).

[8] R.Cohen, G. Nakibli, "On the Computational Complexity and Effectiveness of N-hub Shortest-Path Routing", IEEE Infocom 2004.

[9] D. Dolev, C. Dwork, O. Waarts, M. Yung, "Perfectly Secure Message Transmission", Journal of the Association for Computing Machinery, Vol. 40, No. 1, January 1993, pp. 17-47.

[10] M. R. Garey, D. S. Johnson, "Computers and intractability – A guide to the Theory of NP-Completness".

[11] Y. Guo, F. Kuipers, P. Van Mieghem "Link Disjoint Paths for reliable QoS", International Journal of Communication Systems 779-798 NOV 2003.

[12] J. P. Hespanha and S. Bohacek, "Preliminary results in routing games," in Proc. Of the 2001 American Control Conference, June, 2001.

[13] C. Hoops "Analyzis of an equal-cost, multi-path algorithm", RFC 2992, Nov. 2000.

[14] ilog, AMPL CPLEX System: http://www.ilog.com/products/ampl/

[15] R. Impagliazzo. "A personal view of average-case complexity", In *Proceedings of the 10th Conference on Structure in Complexity Theory*, IEEE Computer Society Press, pages 134-147, 1995

[16] A. Keromytis, V. Misra, and D. Rubenstein. SOS: An Architecture for Mitigating DDoS Attacks. IEEE JSAC, Special Issue on Service Overlay Networks, 22(1), January 2004.

[17] P. P. C. Lee, Vishal Misra and D. Rubenstein, "Distributed Algorithms for Secure Multipath Routing", IEEE Infocom 2005.

[18] S. Mao, S. Lin, S. Bushmitch, S. Narayanan, S. S. Panwar, Y. Wang, R. Izmailov "Real Time Transport With Path Diversity", The 2nd Annual New York Metro Area Networking Workshop, Sep 2002, NY.

[19] A. Orda, A. Spronston, "Efficient Algorithms for Computing Disjoint QoS Paths", Infocom 2004 Hong-Kong.

[20] P. Slavik. A tight analysis of the greedy algorithm for set cover. Journal of Algorithms, 25(2):237–254, 1997.

[21] Villamizar, "OSPF optimized multi-path (OSPF-OMP)", draft-ietf-ospf-omp-03.

[22] Z. Wang and J. Crowcroft. Quality-of-service routing for supporting multi-media applications. IEEE JSAC, pages1288–1234, September 1996.

[23] J. Wang, "Load Balancing in Hop-by-Hop Routing with and without traffic splitting", Ph.D. Thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, October, 2003.

[24] D. Xu, Y. Chen, Y. Xiong, C. Qiao , X. He, "On Finding Disjoint Paths in Single and Dual Link Cost Network", IEEE INFOCOM, Mar. 2004, Hong-Kong.

[25] H. Zlatokrilov H.Levy "Packet Dispersion and the Quality of Voice over IP Applications in IP networks", IEEE Infocom 2004, Hong Kong,

[26] H. Zlatokrilov, H. Levy, "Privacy and Reliability by Dispersive Routing", IWQoS 2005: 362-365 (short paper).

[27] H. Zlatokrilov, H. Levy, "Privacy and Reliability by Dispersive Routing", Technical report, http://www.cs.tau.ac.il/~zlatokri/.

IX. APPNDIX A: ALGORITHM PL_CASH_FLOW

Input parameters: G – directed graph, B – attacker's budget, q – faraction of fragments to the eavesdropped by the attacker, D – path length limit

Variables:

- *cap*[*j*] – capacity on link l_j
- *flow*(path) – the number of cost units sent on path

CASH_FLOW_AUGMENT (*G, B, q, D*)

```
1: F=0
2: PathSet = empty set
3: for each link l_j
4:       cap[j] = q'c_j
5: while ( F<B and flow(path)>0 )
6:       path = WidestLenLimitPath (G, D)
7:       F += flow(path)
8:       for each link l_j in path
9:             cap[j] = cap[j] – flow(path)
10: if (F > B)
11:      for each path in PathSet
12:            send N·flow(path)/F   fragments on paths
13:      return success
14: else return negative result
```

Note, that the algorithm assumes that the input graph is directed. For undirected graphs the following modifications should be done: Each undirected link *v*--*u* will be duplicated to two directed links $v \rightarrow u$ and $u \leftarrow v$, both with the original cap[*l*]. And in Step 9 in *CASH_FLOW_AUGMENT* should be modified to : cap[$v \rightarrow u$]= *cap*[$u \leftarrow v$] = *cap*[$u \leftarrow v$] – *flow*(path).

X. APPENDIX B: ALGORITHM *PL_VALIDATE*

Variables:

- Δ - the reduction factor, which determines the number of iterations and the accuracy of the solution.
- *x* - the least common devisor of link costs $c_1 \ldots c_j$.

PL_VALIDATE (*G,B,q,D*)

```
1: Normalize link costs c_1...c_j by dividing them by the least
   common devisor x, and also the attacker's budget: B'=⌈B/x⌉
2: q'=2
3: while (q'>q)
4:    call CASH_FLOW_AUGMENT [8] (G, B',q',D) and if the
      answer is positive, return result in PathSet
5:    DiscretizeSolution(PathSet)
6:    LP model = prepare input to LP solver
7:    RemoveUnusedLinks(LP model)
7:    if ValidateSolutionWithLP(LP model) is positive then
8:       send N·flow(path) fragments on each path in PathSet
9:       return positive answer
10:   q'= q'- Δ
11: return negative answer
```

The following procedure discretizes the solution found by *CASH_FLOW_AUGMENT* in order to decrease the execution time of the linear programming solver. The procedure makes sure that each path will carry a flow which is a multiple of the common divisor *x*:

DiscretizeSolution (PathSet)

```
1: leftover = 0
2: Sort PathSet by flow(path) (from low to high)
3: forall path in PathSet
4:      Y = modolu(flow(path), x)
5:      if (Y>0) then
6:          flow(path)= flow(path)- Y
7:          leftover = leftover + Y
8:      if (leftover >=1 ) then
9:          flow(path)= flow(path)+ x
10:         leftover -=1
11: if (leftover >= 0) then
12:         flow(path)= flow(path)+x
```

The following procedure removes links, which will not be used by the attacker, and thus can be removed from the model to accelerate the execution of the LP solver.

RemoveUnusedLinks (LP model)

```
1. For each link l_i in links in LP model
2.    For each link l_j in links in LP model
3.       if the set of paths in which l_i participates is equal to
            the set of paths in which l_j participates then
4.             if c_i<c_j then remove l_j from LP model
5.             else remove l_i from LP model
```

Procedure *ValidateSolutionWithLP* solves the attacker problem linear programming model [27] by a linear programming solver such as CPLEX [14]. The procedure returns the minimal guaranteed attacker's cost.

[8] Steps 11–12 in CASH_FLOW_AUGMENT should be skipped and PathSet should be returned.