Intercepting a Stealthy Network

Mai Ben Adar Bessos[1(✉)] and Amir Herzberg[1,2]

[1] Bar-Ilan University, 5290002 Ramat Gan, Israel
mai.bessos@gmail.com, amir.herzberg@gmail.com
[2] University of Connecticut, Storrs, CT 06269-2157, USA

Abstract. We investigate an understudied threat: *networks of stealthy routers (S-Routers)*, communicating across a restricted area. S-Routers use short-range, low-energy communication, detectable only by nearby devices.

We examine algorithms to intercept S-Routers, using one or more mobile devices, called *Interceptors*. We focus on *Destination-Search* scenarios, in which the goal of the Interceptors is to find a (single) destination S-Router, by detecting transmissions along one or more paths from a given (single) source S-Router. We evaluate the algorithms analytically and experimentally (simulations), including against a parametric, optimized S-Routers algorithm.

Our main result is an Interceptors algorithm which bounds the expected time until the destination is found to $O\left(\frac{N}{\hat{B}}\log^2(N)\right)$, where N is the number of S-Routers and $\hat{B}$ is the average rate of transmission.

1 Introduction

Stealthy wireless communication channels have been widely deployed and studied, already since World War I, and mainly for (human) intelligence. Stealthy channels involve a stealthy *source*, communicating to a remote *destination*. Advanced stealthy transmission and encoding methods were developed to hide the transmissions and location of the source, while ensuring reliable communication to the remote destination. Counter-intelligence efforts involved deployment of intercepting-devices (*Interceptors*), deploying advanced techniques to detect the communication and locate the stealthy source. See details in [5–7,10,15,20] (additional related topics are surveyed in the draft of the full version of this work [3]).

Recent advances in Wireless Sensor Networks (WSNs) introduce a new variant of stealthy communication: a *stealthy network*. In a stealthy network, communication is relayed along a *path* consisting of stealthy devices, which we refer to as S-Routers. The S-Routers are covert devices, 'hidden' within a restricted-access area; the source and destination are simply (special) S-Routers. Since adjacent S-Routers are physically near, they can use low-energy, short-range, communication. Energy savings are important; however, it is even more beneficial that such low-energy communication can be hard to detect and localize by

I. Dinur et al. (Eds.): CSCML 2018, LNCS 10879, pp. 188–205, 2018.
https://doi.org/10.1007/978-3-319-94147-9_16

remote Interceptors. On the other hand, S-Router may not be able to deploy the most stealthy techniques, due to size, cost and energy considerations. As a result, an Interceptor would often succeed in detecting and locating a nearby transmitting S-Router.

Prevention of stealthy communication, and interception of S-Routers, is important for many scenarios, including commercial (prevention of industrial espionage) and personal/political, as well as the 'classical' military, counter-intelligence and counter-terrorism. Stealthy networks have been studied and deployed since roughly 2006 [17–19].

The proliferation of Wireless Sensor Networks (WSN), based on low-cost, miniature networked devices [2,8,12,14], may facilitate extensive use of stealthy networks, including commercial and private privacy-intrusive applications. Examples include outdoor or indoor eavesdropping [4,16], industrial espionage, and a command-and-control channel for physical attacks e.g. against communication or energy infrastructure.

In spite of the wide-ranging implications of stealthy networks, this is the first work which presents defense mechanisms to efficiently intercept the stealthy network, using a set of mobile Interceptors, which can detect an S-Router transmitting nearby. We present, analyze and experimentally evaluate algorithms for ensuring efficient search by a set of Interceptors, to intercept and locate the destination of a stealthy communication network (e.g. a base station of a WSN).

Model. To study the problem analytically, we introduce a model for evaluating stealthy routing algorithms, as well as stealthy network interception algorithms.

The model is a round-based process between two parties: S-Routers and Interceptors, both operating on the plane (two-dimensions space). Our focus is on a protocol for the Interceptors party, which defines the operation of all Interceptors, and whose goal is to find the destination S-Router.

We focus on *Destination-Search* scenarios, where the S-Routers route all information from one special S-Router, called the *source*, to another special S-Router, called the *destination*, and denoted by D; the goal of the Interceptors is to find D. A single round of the process is illustrated in Fig. 1.

We assume a single, known *source* of the stealthy communication, which, WLOG, we fix at the center of coordinates, i.e., $(0, 0)$. There are N S-Routers, one of which is the destination D; the other S-Routers relay information from the source $(0, 0)$ to D. S-Routers may transmit up to one data unit at each round. Out model assumes that an Interceptor exposes an S-Router if the S-Router transmits within the interception range of the Interceptor.

S-Routers transmit data at a certain average rate, denoted $\hat{B}$. Intuitively, it seems easier to find D when the rate is high; indeed, we show that when the rate $\hat{B}$ is 1, the Interceptors can use a more optimized algorithm which improves their performance. Note that $\hat{B} = 1$ means that the S-Routers have to transmit data at every round.

Algorithms are measured by their impact on the *lifetime*, which is the number of rounds that pass before the destination is exposed by Interceptors; S-Routers

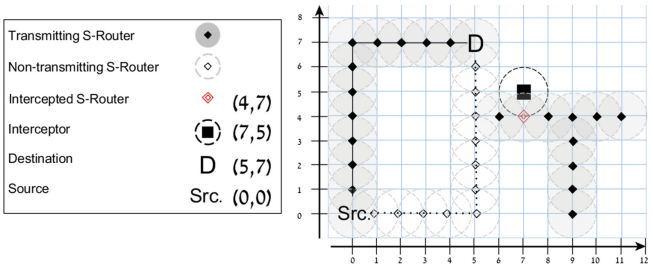

Fig. 1. Illustration of one round, with a single Interceptor searching for destination D, located at $(7, 5)$ (in 2D, i.e., $\mathbb{R}^2$). Two routes of S-Routers connect the source $(0, 0)$ to D (at $(7, 5)$); the first (via $(0, 7)$) transmits at this round (black diamonds), the other not (white diamonds). Several other S-Routers (from $(6, 4)$ to $(11, 4)$ and to $(9, 0)$) transmit 'dummy' messages, to divert the Interceptor from the real routes, and lead it to two 'dead ends': $(11, 4)$ and $(9, 0)$.

attempt to maximize the lifetime, while Interceptors attempt to minimize it. The Interceptors may utilize up to M separate agents simultaneously.

In the particular scenario we study, Interceptors do not 'disable' non-destination S-Routers, even if they 'know' their location. Our model may be extended to consider scenarios/models where the Interceptors can disable an S-Router, e.g. by installing a nearby jamming device or by physical elimination. However, we prefer, in this work, to focus on the scenario where S-Routers are not disabled, both for simplicity and since such scenarios may be important. In particular, in the expected case where there is a large network of multiple S-Router paths, sources and destinations, the Interceptors may prefer not to disable an S-Router, since this may alert the S-Routers network and foil detection of the rest of the stealthy network. Disabling S-Routers before exposing end-points may trigger defensive and evasive reactions by S-Routers e.g. temporarily shutting down nearby communication, alternating communication routes and even activating nearby disconnected S-Routers in order to mislead Interceptors.

Video illustration. A video illustration of the algorithms presented in this work is available online in [1].

Contributions.

- We introduce the threat of stealthy networks, with a flexible model facilitating analysis of algorithms.
- We study two approaches for Interceptors: network graph search and spatial search, and explore advantages and disadvantages of each approach.
- We present a Divide and Conquer (D&C) Algorithm: a spatial search algorithm for Interceptors which ensures that the expected lifetime is in $O(\frac{N}{M\hat{B}} \cdot \log^2(N))$.
- We introduce a parametric algorithm for S-Routers, and use it to experimentally evaluate the Interceptors algorithms presented in this work, via simulations.

2 The 2D Stealthy Network Model

In this section, we present a model for studying problems involving S-Routers, whose goal is to route data without discovery of D, and Interceptors, whose goal is to find D.

The model is round-based; i.e., it operates in consecutive, discrete rounds $t \in \mathbb{N}$ of equal duration, starting from $t = 1$. At each round, the model invokes two algorithms, Π_I for the Interceptors and Π_S for the S-Routers. We model both as centralized algorithms; see [3] for discussion on this simplification. To end a round, the model invokes a third algorithm, Π_E, which models the *environment*.

The environment algorithm Π_E determines the results of the actions of the Interceptors and S-Routers, including the inputs for next round. We believe that the modeling of the environment by an algorithm Π_E gives significant flexibility to the model. For example, in the scope of this work we study two environments, one of which enforces that S-Routers transmit continuously throughout the process, i.e., where $\hat{B} = 1$, and another which does not enforce $\hat{B} = 1$. Other variants of the problem may be modeled by other environments. For example, our model does not restrict the movement of Interceptors from one round to the next; but only a minor change in the environment is required to limit the movement of each Interceptor.

The initial input to all three algorithms are the number of Interceptors, denoted M, and the number of S-Routers, denoted N.

We next define the Interceptors algorithm Π_I. The input for Π_I combines the observations of all Interceptors into a list of points from which transmissions were detected at the previous round (up to one point per Interceptor), and the output is the joint list of locations at the following round. That is, the output for each Interceptor is an encoding of the location, in $\mathbb{R}^2$, for the corresponding Interceptor. The algorithm also has a state as input and output, modeled as binary string.

Next, we define the S-Routers algorithm Π_S. In this work, S-Routers do not move; we denote the location of S-Router $i \in \{0, ...N\}$ by $S_L(i) \in \mathbb{R}$, where 0 is the destination and $1 \dots N$ are the (other) S-Routers. Furthermore, in this work, the model does not include any input to the S-Routers, in particular, S-Routers do not have any way to detect Interceptors, and S-Routers are never impacted in any way; hence, their entire behavior can be determined initially. Their 'behavior' only consists of a *transmission schedule*, $S_T(t) \subset \{0, ..., N\}$, identifying the S-Routers that transmit at round $t \in \mathbb{N}$, and by the (fixed) location of the destination D. The S-Routers algorithm Π_S implements S_T; details omitted.

In most of the paper, we refer directly to S_L, S_T and D, instead of to Π_S. One of the S-Routers serves as the source, which is always at $(0, 0)$; i.e. $(0, 0) \in \cup S_L$. The set of S-Routers must ensure connectivity from source $(0, 0)$ to destination; see definition of connectivity below.

We next model the environment, Π_E. Π_E models the behavior of the environment, as a (probabilistic) algorithm, allowing analysis of different stealthy-network scenarios and goals. The inputs to Π_E are the outputs of Π_I, and

the locations of the S-Routers that transmit at the current round. The outputs are the interceptions to be provided, in the next step, to Π_I. In addition, the environment has a state of its own as input and output. Upon termination, the environment also outputs the average rate of transmission; $\hat{B}$, as defined in Definition 2.

The values returned at the end of the execution of the process are $\hat{B}$, as returned by Π_E, and the *lifetime* of the process, i.e., the number of rounds until the process terminates. If the process never ends, then lifetime is ∞.

2.1 Destination-Search Environments

In this subsection define two environments Π_E used in this work. In the destination-search scenario, the goal of the S-Routers is to maintain a connection between $(0,0)$ and D using the available S-Routers. Therefore, we begin with the notion of connectivity.

Definition 1 (Connectivity). *Two points $p_1, p_2 \in \mathbb{R}^2$ are* connected *iff their Euclidean distance is at most one, i.e., $||p_1 - p_2|| \leq 1$. Let $Connected(p_1, p_2) = 1$ if p_1, p_2 are connected, and 0 otherwise. A list of points is* connected *if every pair of two consecutive points is connected. Two points $p_1, p_2 \in \mathbb{R}^2$ are* connected via *a list of points P if there is a list of points in P, say $(l_1, ...l_k)|(\forall i)(l_i \in P)$, s.t. the list $(p_1, l_1, \ldots, l_k, p_2)$ is connected.*

The value of lifetime represents the success/reward of the S-Routers. This value is also the cost/penalty for the Interceptors, whose goal is to minimize the time needed for intercepting D. However, measuring performance using lifetime alone may be misleading, as S-Routers are often able to increase it simply by decreasing the number of transmissions.

Therefore, we define an additional criteria, the transmission rate $\hat{B}$, which indicates how often S-Routers transmit. For denoting whether a data unit was transmitted at a specific round $t \in \mathbb{N}$, we use $b(t)$.

Definition 2 (Transmission rate measurement $b(), \hat{B}$). *Let $b(t) = 1$ if data is transmitted from $(0,0)$ to D at round t. That is, $b(t) = 1$ if exists is a list of S-Routers $S = (r_1, r_2 \ldots)$ s.t. the source $(0,0)$ and D are connected via their locations $\{S_L(r_i) : r_i \in S\}$, and all S-Routers in the list transmit at round t i.e. $S \subseteq S_T(t)$. Otherwise, let $b(t) = 0$.*

The average transmission rate of S-Routers, denoted with $\hat{B}$, is $\sum_t b(t)$ divided by the lifetime.

Note that in this work, S-Routers can not buffer messages and deliver them later. Future work may remove this restriction, to allow for delay-tolerant networking by S-Routers. One reason for this (simplifying) restriction, is that each round represents a (potential) physical movement by the Interceptors, and movements are normally much slower than communication.

Initialization. Upon initialization, Π_E is provided with an indication of whether the *continuous transmission constraint* has to be enforced (the constraint is defined in Definition 3).

Termination. The environment Π_E will terminate the execution if either party acts in a way forbidden for that execution or if one of the Interceptors is directly connected to the destination D.

The process terminates when Interceptors visit a point near D, due to the assumption that D is a long-range transmitter, which is expected to be larger cf. to S-Routers and therefore easier to localize.

For simplicity, we also assume that Interceptors are able to expose D even while S-Routers do not transmit. Note that for the algorithms presented in this work, this assumption does not affect their performance asymptotic complexity (with an exception for the Naive Disc Search algorithm).

As previously mentioned, if the process never ends, the lifetime is ∞; however, in this work, Interceptors may always avoid this, since D necessarily may be reached after a finite number of rounds.

Transmission rate constraints. As presented so far, the model allows rounds t in which there no transmission-path from source $(0,0)$ to D (i.e., $b(t) = 0$). However, it seems that in many scenarios, Interceptors will transmit continuously to D. We refer to this as the *continuous transmission constraint/assumption.*

Definition 3 (Continuous Transmission Constraint). *We say that the* Continuous transmission constraint *holds for an execution, if for every round t in the execution, $b(t) = 1$ holds, i.e., $(0,0)$ and D are connected via some set of S-Routers at that round. We say that Π_E* Enforces Continuous Transmission *if Π_E terminates the execution, with $\hat{B} = 0$, upon a round in which the constraint does not hold.*

3 Introducing Interceptors Algorithms

We found that the design of efficient Interceptors algorithm is more challenging than appears initially, with resulting algorithm being somewhat counter-intuitive. Obviously, we cannot repeat here all the variations we experimented with; however, we present few basic algorithms, which we believe will help the reader understand the problem better, preparing the ground for the more efficient - but less intuitive - algorithms presented in the following sections.

In this section we present three Interceptors algorithms. We begin with an observation that the Interceptors may limit their search to a bounded area, specifically, a *disc*. We then present an algorithm which essentially searches this disc. Afterwards, we present two naive attempts to find D by 'following the path' from the source $(0,0)$ to D, which are reminiscent of graph-search algorithms.

To our disappointment, we did not yet find an efficient graph-search algorithm, that works in the general case. However, it is possible that future work would find better ways to use the graph-search approach. For a more thorough

discussion on this topic, and for proofs of the Propositions and Lemma presented in this section, see the draft of the full version of this work [3].

Naive Disc Search Algorithm. We begin with a very simple algorithm that we call *Naive Disc Search*. Basically, the Naive Disc Search algorithm exhaustively searches for D in a bounded disc. We first show in Lemma 1 that it suffices to search for D within a disc, specifically, the disc of radius N whose center is the source $(0,0)$; this simple bound may also used by the more advanced algorithms. The lemma uses the following notation.

Notation: *Disc*. Given a point $c \in \mathbb{R}^2$ and a distance $r \in \mathbb{R}$, let $\boldsymbol{Disc}(r,c) = \{p \in \mathbb{R}^2 | \ ||p-c|| \leq r\}$ denote the region of a disc whose center is c and whose radius is r.

Lemma 1. *If D is connected to the source $(0,0)$ via any list of S-Routers $S \subset \cup S_L$, then $D \in Disc(N,(0,0))$.*

Since it suffices to search for D within $Disc(N,(0,0))$, a simple, naive approach is to exhaustively search this disc. More precisely, such algorithm will visit different points within the disc, where each point results in covering a disc of radius 1 centered in that point, until the entire disc was covered - or D found.

The order of visitations may affect the performance of the algorithm. For example, if all S-Routers are located 'densely' around $(0,0)$, as illustrated in Fig. 2(a), Interceptors may sort all points by (increasing) distance from $(0,0)$ then visit them in that order in order to find D efficiently in $O(N)$ rounds. However, if the search is deterministic and known in advance, D may be placed so it is found only by the very last searched points. For example, if Interceptors keep visiting points with increasing distance from $(0,0)$ but S-Routers are located as illustrated in Fig. 2(b), roughly the entire disc $Disc(\frac{N}{2},(0,0))$ will be covered before D is found. Hence, a random order is slightly preferable for Interceptors.

The Naive Disc Search Algorithm uses a predefined set of points to search, i.e., the covering of a disc of radius N by discs of radius 1. Let $\boldsymbol{DiscCoverage(N)}$ denote the set of points that cover $Disc(N,(0,0))$. It is difficult to minimize the size of $DiscCoverage(N)$ [11], but its complexity is necessarily $O(N^2)$, and efficient implementations for $DiscCoverage(N)$ can achieve it [9].

In the The Naive Disc Search Algorithm, the Interceptors search for D by visiting every point in $DiscCoverage(N)$ in random order. At each invocation, the algorithm keeps all previously visited points, and outputs a list of M previously unvisited points to visit next.

Proposition 2. *The expected lifetime of the Naive Disc Search algorithm is in $O(N^2/M)$.*

Naive Graph Search Algorithm. The Naive Disc Search Algorithm does not use the interceptions (detections), which seems wasteful. Surely, we can use interceptions to find D more efficiently. One natural idea is to exploit the fact that D must receive transmissions from the source $(0,0)$; we can try to 'follow' these transmissions, by always searching in the vicinity of one of the points where

we intercepted a transmission, plus the source $(0, 0)$. This Naive Graph Search Algorithm keeps a set of locations from which a transmission was intercepted (initialized to the source $\{(0, 0)\}$), then visits at each round at points that are within distance ≤ 3 from one of the points in the vicinity of previously successful search locations, chosen at random with uniform probability.

Proposition 3. *The expected lifetime of the Naive Graph Search algorithm is in* $O(\frac{N^2}{\hat{B} \cdot M})$.

Uniform Graph Search Algorithm. Since the Naive Graph Search algorithm selects points with uniform probability at each step, points in the vicinity of earlier interceptions have more opportunities for being selected. Intuitively, if newly discovered interceptions will be visited more frequently, the performance of the algorithm may be significantly improved. In order to examine this approach, we have designed the *Uniform Graph Search* algorithm. The algorithm assumes that only a single Interceptor is available, i.e., $M = 1$. The algorithm is defined similarly to the Naive Graph Search Algorithm, with the following modifications:

1. For each point in $DiscCoverage(N)$, initialize a counter to 0.
2. Each time a point is visited by the algorithm, increase its counter by 1.
3. When selecting a point to visit, select points with minimal counter value.

Namely, all points that the algorithm may visit will be roughly visited an equal number of times, and for each interception, the new points and its vicinity will be visited repeatedly, until their associated counter value is no longer minimal. For example, if S-Routers use the network illustrated in Fig. 2(b), and the continuous transmission constraint, as defined in Definition 3, holds, then the Uniform Graph Search will frequently intercept new S-Routers (and eventually D) at the 'front' of the few routes, since data is transmitted through at least one of the routes at each step, and the algorithm will repeatedly visit points near the 'front' after each interception. This scenario is handled far less efficiently by the Naive Disc Search and the Naive Graph Search algorithms.

Unfortunately, in the worst case, the performance of this algorithm is not significantly better (compared to the naive algorithm). Even if the continuous transmission constraint holds, if the network graph includes many separate alternate routes that connect $(0, 0)$ and D, the transmission rate in each route may be reduced proportionally (as illustrated in Fig. 2(c)), and interception of new S-Routers will be infrequent.

Proposition 4. *If the continuous transmission constraint holds, the expected lifetime of the Uniform Graph Search is in* $\Omega(N^2/log(N))$.

Note that in the above result, the term used for bounding on the lifetime excludes $\hat{B}$, since the continuous transmission constraint ensures that $\hat{B} = 1$.

In the following sections we present Interceptors algorithms and prove their expected performance is significantly better (assymptocially) compared to the Uniform Graph Search algorithm. However, for many cases, where S-Routers

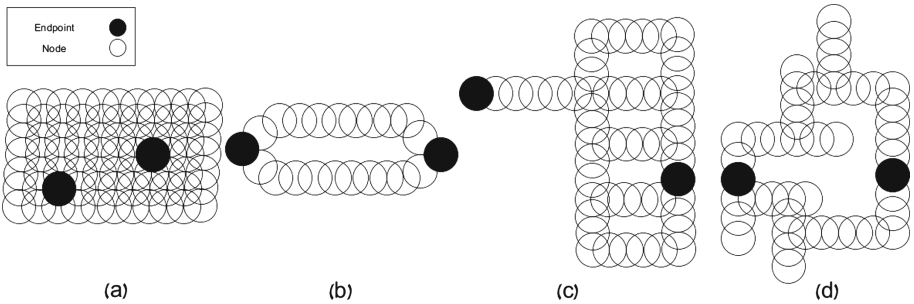

Fig. 2. Examples of different S-Router networks: (a) All S-Routers are located 'densely' around the source $(0,0)$. An exhaustive search around $(0,0)$ may reach D efficiently. (b) Only few separate 'long' alternate routes connect $(0,0)$ and D. If Continuous Transmission Assumption holds, the rate of transmission in at least one of these routes is relatively high, which allows the Uniform Graph Search algorithm to expose S-Routers efficiently. (c) A network which uses numerous separate alternate routes. The transmission rates in different points may vary significantly; in particular, this prevents Uniform Graph Search algorithm from exposing new S-Routers efficiently even if the Continuous Transmission Assumption holds. (d) A network with few paths but numerous 'dead ends'. Even if a graph-search algorithm can efficiently cope with routes which transmit slowly, it is difficult to discern such routes from actual 'dead ends'. Since most 'walks' in the network lead to a 'dead end' and S-Routers may make 'dead end' routes appear exactly like other routes, it is also difficult to avoid them

is small enough, the Uniform Graph Search algorithm outperforms all other algorithms, as illustrated in Fig. 6. A more detailed performance comparison is given in Sect. 5.

4 Divide and Conquer Interceptors Algorithm

In this section we present an algorithm for the Interceptors, the *Divide And Conquer Algorithm*, which bounds the expected lifetime to $O(\frac{N}{\bar{B} \cdot M} \log^2(N))$ for M Interceptors. Counter-intuitively, and in contrast to the less efficient graph search algorithms of the previous section, this algorithm does *not* try to 'search' the graph of S-Routers from $(0,0)$ to D. Instead, this algorithm takes a 'divide and conquer' method, to find the destination D 'directly' - without exposing the entire path to it.

We begin with few a preliminaries in Sect. 4.1, then describe the algorithm in Sect. 4.2. For a thorough analysis of the algorithm, proofs, and additional topics, see the draft of the full version of this work [3].

4.1 Preliminaries: Ranges and Walls

We begin this section with few additional topological concepts which are used in this section.

First, given a location $l \in \mathbb{R}^2$, let $Range(l)$ denote its *range*, i.e., the set of points whose communication would be intercepted by an Interceptor located at location l. Formally, $\boldsymbol{Range}(l) = \{x \in \mathbb{R}^2 | Connected(l, x)\}$. The range notation extends to a set of points L, namely we denote $Range(L) = \bigcup_{l \in L} Range(l)$.

The Divide And Conquer Algorithm uses the fact that D must be within $Disc(N, (0, 0))$, as shown in Lemma 1. The algorithm partitions $Disc(N, (0, 0))$ into smaller regions, then examines these regions by visiting points on their boundaries. If the boundaries 'separate' between $(0, 0)$ and D, and considering the S-Routers transmit from $(0, 0)$ to D, it follows that these transmissions must 'cross' one or more of the boundaries which are visited by the algorithm. If the points of a boundary are sufficiently-close, then the algorithm may intercept transmissions from at least one of these points.

We define two topological notions which are important in this algorithm: a $\sqrt{3}$-spaced *wall* and *closed wall*.

Definition 4 (Wall, closed wall, and In/Out regions). *An $\sqrt{3}$-spaced wall is a list of points $L = \{l_1, l_2, \ldots, l_k\} \in \left(\mathbb{R}^2\right)^k$ such that the distance between every two consecutive points l_i, l_{i+1} is at most $\sqrt{3}$. A $\sqrt{3}$-spaced closed wall L (abbreviated to* closed wall*), is a wall where the distance between l_1 and l_k is at most $\sqrt{3}$. We denote the outer region by $\boldsymbol{Out}(L)$, and the internal region, excluding $Range(L)$ itself, by $\boldsymbol{In}(L)$.*

In the definition above, to define the inner and outer region, we use basic topological notions such as boundary and region, which are quite intuitive and standard; precise definitions can be found, e.g., in [13].

We focus on $\sqrt{3}$-spaced walls and closed walls, since a $\sqrt{3}$-spaced closed wall separates between (points in) its *internal region*, $In(L)$, and (points in) its *outer region*, $Out(L)$. The formal statement is given in the next Lemma, and illustrated in Fig. 3. Therefore, we write *walls* and *closed walls*, always referring to $\sqrt{3}$-spaced walls and closed walls (an x-spaced closed wall with $x > \sqrt{3}$ may fail to provide the separation property referred to in the Lemma).

Lemma 5. *Given a ($\sqrt{3}$-spaced) closed wall L, no pair of points $p_{In} \in In(L)$ and $p_{Out} \in Out(L)$ are connected. Namely, for all $p_{In} \in In(L)$ and $p_{Out} \in Out(L)$ holds $||p_{In} - p_{Out}|| > 1$.*

The Divide And Conquer Algorithm generates closed walls, then instructs the Interceptors to visit them in a random order. In order to calculate the probability of interception when visiting a point in a closed wall that 'separates' $(0, 0)$ and D (such as the closed wall illustrated in Fig. 3).

Definition 5 (Separating closed walls). *We use* separating closed wall *to refer to a closed wall that contains D but excludes $(0, 0)$, namely a closed wall L for which $D \in In(L) \cup Range(L)$ and $(0, 0) \in Out(L) \cup Range(L)$ hold.*

Proposition 6. *Let L be a separating closed wall, and let $t \in \mathbb{N}$ be a round s.t. $b(t) = 1$. There exist $v \in L$ and $x \in S_T(t)$ s.t. $Connected(S_L(x), v)$.*

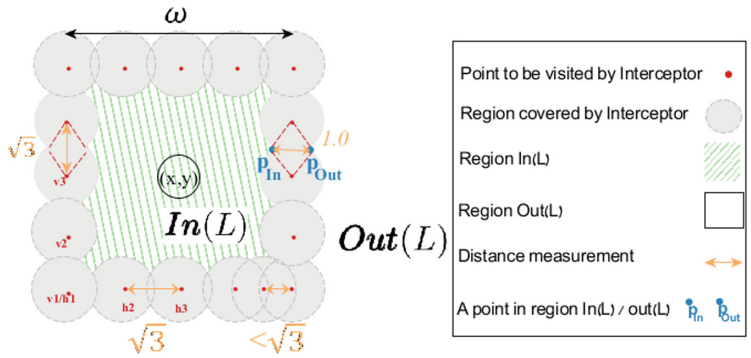

Fig. 3. The closed wall $L = \{v_1, v_2, \ldots\} \cup \{h_1, h_2, \ldots\} \cup \ldots$ separates the plane into the the inner and outer regions $In(L)$, $Out(L)$ and its range $Range(L)$. Note that all points in L rest on the boundary of the same square. Any point from region $In(L)$ is not connected to any point in region $Out(L)$. This property is used by our algorithms: Interceptors inspect, randomly, the points in closed walls separating D (inside) from the source $(0, 0)$.

Finally, we define *leading square walls*, which are the 'smallest' separating square walls; by dividing these walls, the algorithm 'zooms in' on D.

Definition 6 (Leading square walls). *Let $\mathcal{L} = \{L_1, L_2, \ldots\}$, where L_i is a subset of the 'watched' points, be the set of all separating square walls in the 'watched' points. We refer to a separating square wall $L \in \mathcal{L}$ as a* leading square wall *if no other separating square wall is contained in L i.e. $\forall L' \in \mathcal{L} : L' \not\subseteq (In(L) \cup Range(L))$.*

4.2 Divide And Conquer Algorithm

We now present the Divide And Conquer Algorithm. The algorithm visits at each round M distinct points out of a set of 'watched' points. These 'watched' points are placed as a closed wall around *squares* containing D; we begin with very large squares and repeatedly divide them into smaller squares, until we find D. Let us first present our notation for a square.

Notation. (Square). Given a point $(x, y) \in \mathbb{R}^2$, and a length $w \in \mathbb{R}$, let $\textbf{\textit{Square}}(w, (x, y)) = [-\frac{w}{2} + x, \frac{w}{2} + x] \times [-\frac{w}{2} + y, \frac{w}{2} + y]$ denote the region of a square whose center is (x, y) and each of its edges are of length w. For example, Fig. 3 gives a visualization of the closed wall L, where all points in L rest on the boundary of the square $Square(w, (x, y))$.

The algorithm searches for D in $Square(2N, (0, 0))$ (which contains $Disc(N, (0, 0))$ and D in particular). The algorithm partitions $Square(2N, (0, 0))$ into smaller squares and places Interceptors at several random points along walls on their boundaries. That is, a closed wall is kept per square, s.t. one of these 'watched' closed walls contains the destination D; we refer to these square-shaped closed walls as *square wall*.

When a square wall shows signs of possibly containing D, namely when a transmission was intercepted from one of it's points, the algorithm further divides the corresponding square into four quarters, and repeats the process for these smaller squares, until finding D. For efficiency, the total size of walls of 'watched' squares should be small; to find D, the regions must contain it. The algorithm carefully ensures both properties.

It is crucial to randomize the location of the squares, to foil S-Router placements that exploit predictable locations of square walls. S-Routers may lead the algorithm into 'watching' many additional regions that do not contain D, due to S-Routers that deliberately expose themselves at specific locations. Hence, we first select a random point o from $Square(2N, (0,0))$. From the beginning, we 'watch' the four $2N \times 2N$ squares shown in Fig. 4(a), s.t. o is a shared corner. From Lemma 1, we can assume that D is within one of these four squares.

At each round, we put the Interceptors at distinct random points in the walls of 'watched' squares. We try to detect the transmissions by S-Routers crossing these walls, from source $(0,0)$ to D; this identifies 'suspect' squares, worthy of further decomposing into four sub-squares. From Lemma 1, it suffices, however, to put Interceptors at points which are within $Disc(N, (0,0))$; see our 'focus' on $Disc(N, (0,0))$ in Fig. 4(b).

If we use only large squares, e.g., the four large, $2N \times 2N$ squares shown in Fig. 4(a), then it is quite possible that no path from $(0,0)$ to D will cross their walls at all - since D will be within the same large square. Indeed, in Fig. 4(a), we see that D and $(0,0)$ are both in the lower-right large square (shown more closely in Fig. 4(b)). To ensure that each path of S-Routers from $(0,0)$ to D will cross one of the 'watched walls', we divide, from the very beginning, each of the 'watched squares' containing $(0,0)$ into its four sub-squares, until reaching squares small enough to ensure localization of D. This is not *that* wasteful: the additional length of all these initial sub-squares is less than the length of the initial $2N \times 2N$ squares.

Assuming D is in $Disc(N, (0,0))$, at least one of the initial square walls includes D and excludes $(0,0)$ i.e. the algorithm begins with at least one leading square wall; this ensures Proposition 6 holds, and at least one point in one of the initial square walls will allow the Interceptor to detect a transmission. Points to search are selected at random with uniform probability from all square walls, such that the probability of selecting a point from any certain square wall is proportional to its size. For each successful search, on top of the square walls associated with that search point, four smaller square walls that encircle the same region are added, in an attempt to decrease the size of the leading square wall. Sufficiently small squares are covered by a single visit.

Theorem 7. *The expected lifetime of Divide And Conquer Algorithm is* $O(\frac{N}{\widehat{B} \cdot M} \cdot \log^2(N))$.

Theorem 7 holds since the algorithm has to divide a leading square wall at most $\lceil log_2(N) - 1 \rceil$, until D is found, and since the expected number of transmitted data units until the algorithm divides a leading square wall is $O(\frac{N}{M} log(N))$.

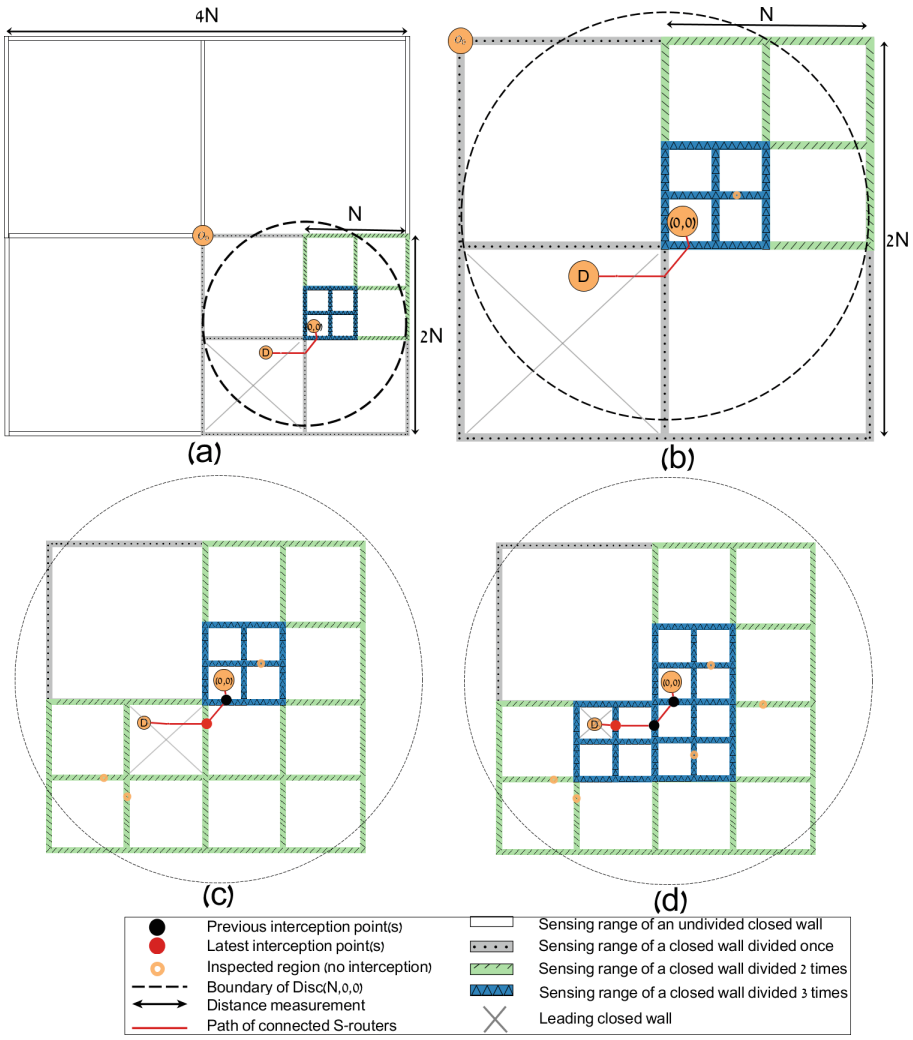

Fig. 4. Illustration of the operation Divide And Conquer Algorithm. The algorithm begins by visiting points on square walls generated upon initialization, and with each interception it generates additional square walls which potentially include D. (a) Illustrates the initialization of the algorithm, where four square walls of size $2N \times 2N$ are generated with random offset such that the entire searched region $Disc(N,(0,0))$ is contained by them. (b) Illustrates the next initialization step. The square wall that includes $(0,0)$ is repeatedly divided for $log(N)$ times. Note that one of the square walls includes D, but excludes $(0,0)$ (c) Illustrates the first detection of a transmission while visiting a point in one of the square walls (the red circle). The two square walls adjacent to the point of detection(s) are divided into four. After this division, the smallest square wall which includes the destination and excludes $(0,0)$ (the leading square wall) is smaller. (d) Illustrates the second and third transmission detection. The second detection occurs at the same point, which leads to additional division. The third transmission detection (the red circle) will lead to another division (excluded from this illustration), and then to the detection of D. (Color figure online)

As previously mentioned, a more thorough discussion and analysis of the algorithm is given in the draft of the full version of this work [3]. In particular, we present a modified version of the Divide And Conquer Algorithm, referred to as D&CCTA, which achieves better performance if the continuous assumption holds (the difference in performance is illustrated in Fig. 6).

5 Evaluation and Results

There is always a challenge in evaluating the practical performance of a defensive mechanism, whose results depend completely on the behaviour of the adversary - in our case, the S-Routers. Our approach was to use a set of S-Routers algorithms, each one 'optimized' for per each Interceptors algorithm. Of course, we do not really know how to produce the 'best' S-Router algorithm (in general, or for a particular Interceptors algorithm). Instead, we tried our best to develop good S-Routers algorithms, in two steps.

In the first step, we developed a parametric heuristic S-Routers algorithm, the *Parametric Segmented Network* algorithm, based on our analysis of different Interceptors algorithms, and on 'trial and error', using a simulation and visualization environment we developed for this purpose. We will make this tool freely available in [1], to allow further research and reproducibility.

In the second step, we used genetic programming to optimize the parameters of Parametric Segmented Network algorithm for each of the Interceptors algorithms, and then compared the results of the different Interceptors algorithms - each running against the 'best' parameters (of the parametrized S-Routers algorithm). We begin with a description of the Parametric Segmented Network algorithm, then present simulation results.

The Parametric Segmented Network Algorithm. The algorithm receives as input parameters for selecting the number of segments, the number of S-Routers that compose each segment, the number of parallel paths in each segment, and what portion of these paths lead to a 'dead end'.

Figure 5 illustrates a network composed of $N = 128$ S-Routers which are separated into three segments. The segments are composed of $0.35N, 0.3N, 0.3N$ S-Routers (from left to right), where $N = 128$. D is the leftmost S-Router, while $(0, 0)$ is the rightmost S-Router. Parallel paths of the same segment begin and end with a joint path (perpendicular to the parallel paths), and adjacent segments share one S-Router that connects the joint paths. The remaining $0.05N$ S-Routers that are disconnected transmit continuously, in order to mislead non-graph search algorithms (such as the Divide And Conquer Algorithm). The leftmost segment transmits into one additional 'dead end' parallel route.

The Parametric Segmented Network algorithm satisfies the continuous-transmission constraint. At each round, every segment transmits data through one of its paths, and S-Routers of joint (perpendicular) paths transmit continuously. In Fig. 5, the top path in each segment transmits.

More specific details on the implementation of the Parametric Segmented Network algorithm are given in the full version of this paper in [3].

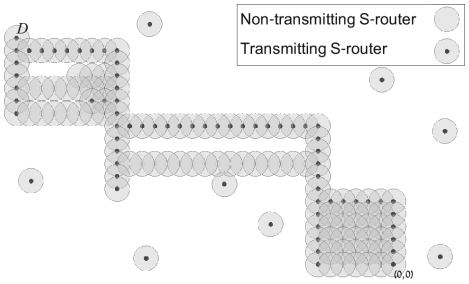

Fig. 5. The Parametric Segmented Network algorithm as visualized by the simulation, for 128 S-Routers that are separated to three segments, at a specific round. At each step, one of the parallel paths in each segment transmits. The S-Routers connecting adjacent segments transmit continuously. Hence, data may flow from the rightmost S-Router in the network, $(0, 0)$ to the leftmost S-Router, D.

Results. We now compare results obtained by simulations. In Fig. 6, S-Routers use the Parametric Segmented Network algorithm. In order to limit the dimension of optimization, the number of segments was limited to four. The parameters for the algorithm were optimized separately for each scenario i.e. for each Interceptor algorithm and each N pairing. After the selection of the best parameters for each scenario, we ran the process enough times to ensure the evaluation of the parameter set is accurate. The results are displayed in Fig. 6. A confidence interval of 99% is used. All Interceptors algorithms utilize a single Interceptor. Due to the high computational costs, we used the genetic algorithm only for $N \leq 128$. For larger N values, the parameter sets were selected according to the conclusions found from smaller values.

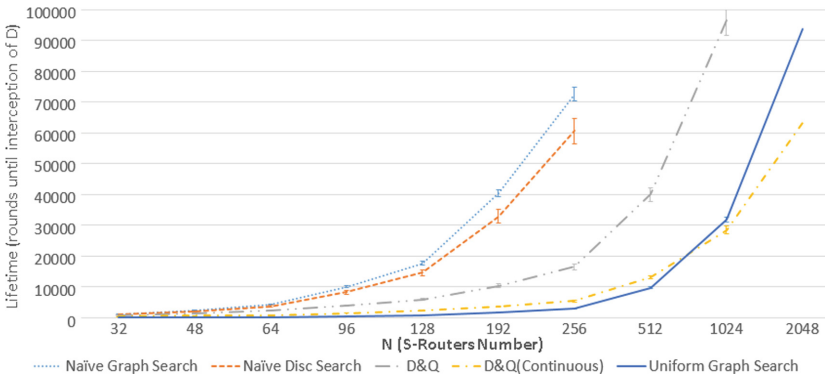

Fig. 6. Performance comparison of different Interceptors algorithms. Performance is estimated against Parametric Segmented Network S-Routers algorithm - optimized for each Interceptor algorithm (using a single Interceptor) and each N value separately. Confidence interval is displayed as vertical lines.

The genetic algorithm consistently keeps a population size of 500. At each epoch, the algorithm applies standard roulette selection. Crossovers are done until for 75% of the population is replaced, and then mutation is applied to 20% of the population where on average two elements of each mutated parameter set change. Changes are done by adding a Gaussian distributed random value. Since the S-Routers algorithms are making extensive use of randomization, the value of a given parameter set was the average lifetime when running the process repeatedly for 15 times. We allocated 24 h of CPU time for each scenario. Running a larger scenario takes significantly more time; therefore, the number of generations varied from several hundreds (for $N \leq 64$) to 48 (for $N = 128$).

Discussion. After examining the parameter sets that resulted from optimization, we discovered that certain properties consistently maximize the outcome for S-Routers. S-Routers that were disconnected from $(0, 0)$ and 'dead end' paths were not useful, and optimized solutions did not express them.

When S-Routers face the Uniform Graph Search algorithm, only two segments are needed. The segment adjacent to $(0, 0)$ is composed of single route, with $0.1N$ to $0.4N$ of the S-Routers, which is always intercepted in its entirety by the algorithm. The segment adjacent to D uses the remaining S-Routers for composing parallel paths. The number of parallel paths increases with N. Additionally, these routes are spread with distance of three from each other, in order to minimize the number of distinct 'watched' points.

A result we did not expect was that when S-Routers face the Divide and Conquer algorithm, the lifetime is maximized when only a single, long route was used, i.e., all segments express a single transmission route. In hindsight, we understood this; if using multiple routes than the Interceptors algorithms will eventually generate square walls that are too small to intersect all parallel paths of the same segment. As a consequence, these square walls have a lower probability for being divided, the total number of points that are 'watched' by the Interceptors algorithms decreases, and their performance improves (lower outcome). This may motivate an S-Routers algorithm that combines the graph-search approach with the divide-and-conquer approaches.

6 Conclusions and Extensions

Stealthy networks, comprised of hard-to-locate devices, are becoming a part of reality; we use the term S-Routers for such devices, who can relay information, to form large networks. Stealthy networks will be used for different applications; many of the applications may represent threats to privacy of individuals and organizations. Hence, it is important to develop efficient countermeasures. Due to the small size of the devices and their use of short-range communication, we envision the use of mobile devices, Interceptors, to localize the S-Routers.

In this work, we investigated algorithmic issues related the interception of stealthy networks. Our focus was on developing efficient algorithms for Interceptors, to expose the destination of stealthy network; we believe that such

algorithms may be deployed as part of the design of countermeasures to stealthy networks.

There are many directions for improvements, extensions/variations, and further research. For example, if Interceptors may predict the S-Routers' transmission schedule, they may be able to accelerate their search significantly.

Improvements may also be possible for the analysis. The current results provide an upper bound for the expected lifetime in the studied environment, but a lower bound is yet to be found. While it is relatively simple to prove that S-Routers may ensure the expected lifetime is bounded from by $O(N)$, developing additional algorithms for S-Routers may be required in order to find the exact bounds, or at least for narrowing the gap between $O(N)$ and $O(\frac{N}{M\hat{B}}log^2(N))$.

The presented model is general enough, to allow investigation of several related problems, including (1) multiple sources and/or destinations, (2) allowing S-Routers to buffer data, (3) introducing mobility, and much more.

Finally, note that the current model does not support decentralized algorithms. We expect that in some practical scenarios, S-Routers may have to risk exposure in order to coordinate. An extension to the model is necessary for studying such scenarios.

Acknowledgments. This work is supported by the Israeli Ministry of Science and Technology.

References

1. Herzberg, A., Ben Adar Bessos, M.: Intercepting a stealthy network - simulation demonstration (2018). https://sites.google.com/view/stealthynetinterception/home
2. Baisch, A.T., Ozcan, O., Goldberg, B., Wood, R.J.: High speed locomotion for a quadrupedal microrobot. Int. J. Robot. Res. **33**(8), 1063–1082 (2014)
3. Herzberg, A., Ben Adar Bessos, M.: Intercepting a stealthy network. vixra.org/abs/1712.0510
4. Bobic, I.: Ted cruz wants police to 'patrol and secure' U.S. Muslim communities after brussels, March 2016. www.huffingtonpost.com. Accessed 21 Nov 2017
5. Bash, B.A., Goeckel, D., Towsley, D.: Hiding information in noise: fundamental limits of covert wireless communication. IEEE Commun. Mag. **53**(12), 26–31 (2015)
6. Che, P.H., Bakshi, M., Jaggi, S.: Reliable deniable communication: hiding messages in noise. In: 2013 IEEE International Symposium on Information Theory Proceedings (ISIT), pp. 2945–2949. IEEE (2013)
7. Chen, O., Meadows, C., Trivedi, G.: Stealthy protocols: metrics and open problems. In: Gibson-Robinson, T., Hopcroft, P., Lazić, R. (eds.) Concurrency, Security, and Puzzles. LNCS, vol. 10160, pp. 1–17. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-51046-0_1
8. Chen, X., Purohit, A., Pan, S., Ruiz, C., Han, J., Sun, Z., Mokaya, F., Tague, P., Zhang, P.: Design experiences in minimalistic flying sensor node platform through sensorfly. ACM Trans. Sensor Netw. (TOSN) **13**(4), 33 (2017)
9. Das, G.K., Das, S., Nandy, S.C., Sinha, B.P.: Efficient algorithm for placing a given number of base stations to cover a convex region. J. Parallel Distrib. Comput. **66**(11), 1353–1358 (2006)

10. Hu, J., Yan, S., Zhou, X., Shu, F., Wang, J.: Covert communication in wireless relay networks (2017). CoRR abs/1704.04946
11. Kershner, R.: The number of circles covering a set. Am. J. Math. **61**(3), 665–671 (1939)
12. MacGregor, A.: Russian scientists create cockroach spy robot. thestack.com. Accessed 2 May 2018
13. Munkres, J.R.: Topology. Prentice Hall, Upper Saddle River (2000)
14. Rubenstein, M., Ahler, C., Nagpal, R.: Kilobot: a low cost scalable robot system for collective behaviors. In: 2012 IEEE International Conference on Robotics and Automation (ICRA), pp. 3293–3298. IEEE (2012)
15. Shabsigh, G.: Covert Communications in the RF Band of Primary Wireless Networks. Ph.D. thesis, University of Kansas (2017)
16. Sidahmed, M.: NYPD's muslim surveillance violated regulations as recently as 2015: report, August 2016. www.theguardian.com. Accessed 21 Nov 2017
17. Bokareva, T., Hu, W., Kanhere, S.S., Jha, S.: Wireless sensor networks for battlefield surveillance. In: Proceedings of the Land Warfare Conference, pp. 1–8 (2006)
18. He, T., Krishnamurthy, S., Luo, L., Yan, T., Gu, L., Stoleru, R., Zhou, G., Cao, Q., Vicaire, P., Stankovic, J.A., Abdelzaher, T.F., Hui, J., Krogh, B.: VigilNet: an integrated sensor network system for energy-efficient surveillance. ACM Trans. Sensor Netw. (TOSN) **2**(1), 1–38 (2006)
19. He, T., Vicaire, P., Cao, Q., Yan, T., Zhou, G., Gu, L., Luo, L., Stoleru, R., Stankovic, J.A., Abdelzaher, T.F.: Achieving long-term surveillance in vigilnet. Technical report. Department of Computer Science, Virginia Univ Charlottesville (2006)
20. Wang, L., Wornell, G.W., Zheng, L.: Fundamental limits of communication with low probability of detection. IEEE Trans. Inf. Theory **62**(6), 3493–3503 (2016)