Using Non-adaptive Group Testing to Construct Spy Agent Routes

Georgios Kalogridis
Toshiba Research Europe Limited
Telecommunications Research Laboratory
32 Queen Square, Bristol, BS1 4ND, UK
georgios.kalogridis@toshiba-trel.com

Chris J. Mitchell
Information Security Group
Royal Holloway, University of London
Egham, Surrey, TW20 0EX, UK
c.mitchell@rhul.ac.uk

Abstract

We consider a network of remote agent platforms that are tested by roaming spy agents in order to identify those that are malicious, based on the outcome of each agent. It is shown that, given a set of spying requirements, the task of choosing the sets of platforms which spy agents visit can be abstracted as a group testing problem. Non-adaptive group testing, in particular, is considered in greater detail, and a simple combinatorial construction for a set of agent routes is presented which combines known results from the prior art. Although existing techniques enable us to construct efficient sets of agent routes, optimality remains an open problem.

1. Introduction

Spy agents, introduced in [1], are mobile software agents that are concurrently dispatched from one (or more) secure platform(s) and migrate through a series of potentially insecure remote hosts, before returning to a secure location. Their purpose is to gather information to help evaluate the trustworthiness of the visited platforms. Unlike malicious agents such as viruses, spy agents are legitimate mobile agents in the sense that they interact with visited hosts in the way expected by the hosts. As such they are analogous to software decoys [2] and honeypots [3].

The main objective of a spy agent system is to evaluate trust in remote environments, where the remote hosts should be unaware that their trustworthiness is being tested. The structure of a spy agent should be no different to that of a generic mobile agent (e.g. an agent designed to obtain the prices of goods from multiple e-commerce sites). A spy agent will include decoy private data and a pre-coded routing scheme, which hosts are usually able to access to facilitate migration. The evaluation of the remote platforms is based on the collective outcomes of the spy agents (i.e. the detectable impacts on the agents after visiting the defined set of platforms), which could either be negative (if there is no sign of security violation), or positive (otherwise).

The main focus of this paper is the choice of sets of routes for spy agents which efficiently identify which remote hosts misbehave. For the purposes of this paper, a route is defined as an (unordered) set of hosts which an agent visits. This contrasts with the more usual definition of route, in which the order of destinations is of significance.

A simple solution to this problem would be to send each spy agent to a single target host. The evaluation process for such a design is trivial, since a positive result for a spy agent implies a misbehaving host. However, improved solutions are sought for the following reasons:

- *Security*: Larger test groups yield more credible results. As discussed in [1], the longer the migrating route, the less a malign host is likely to suspect a spying scenario, and the greater the chance that it can cheat without being detected. As a result, it is more likely to misbehave, revealing its true character.
- *Economy*: Less agents and less time are needed if hosts are tested in groups instead of one by one. This is of particular importance when routinely performing large scale tests.

The problem of choosing a set of agent routes is an application of *group testing* theory [4], in which a (large) population of items containing a small set

of *defectives* are to be tested in order to identify the defectives.

2. Previous work

A good overview of mobile agent security can be found in [5], [6]. The use of spy agents to assess the credibility of remote hosts can be classified as a pre-emptive security evaluation technique for protecting agents against malicious hosts. A similar concept is discussed in [7], [8], but this previous work only applies in trusted environments. In [9] it is suggested that a security assessment can be made as a result of agent migration to unknown hosts by assuming that target hosts provide the agents with the security information they request. However, as argued in [1], a corrupted remote host could detect incoming security agents and selectively behave well in order to escape detection. The same host could behave inappropriately when it has the opportunity to cheat without being detected. Spy agents address this issue with the use of combinatorial group testing methods to try to assess the genuine character of remote hosts.

Cryptographic tracing can be used to detect unauthorised modifications to an agent's code in a multi-host route [10]. Traces are logs of the operations performed by an agent during its lifetime. In a tracing scenario, platforms are required to create, maintain and exchange authenticated logs for all incoming agents. After an agent's termination, its owner can acquire all the secure logs and compare them against the logs produced locally to "reconstruct the crime" and detect discrepancies. However, this approach has the following limitations:

- If the language adopted for agents allows for the processing of very complicated data structures, the modifications to the agent's internal state could be difficult to represent and require a lot of space, rendering the mechanism practically infeasible [10].
- Privacy issues are likely to arise for secure log management.
- Only attacks involving illegal modification of code, state and execution flow of a mobile agent can be detected. Indirect attacks, such as misuse of a personal email address, cannot be detected.

The spying approach allows unaware malicious hosts to abuse "normal" agents (e.g. without requesting proofs), and infers their trustworthiness from the somewhat limited information available after agent termination.

3. Detecting attacks on spy agents

A taxonomy of attacks on mobile agents is given in [5]. The impact of an attack depends on the scenario. For example, unauthorised manipulation of the agent's code [11] might be detectable, or the delayed return of an agent could imply delibarate retention of an agent for malicious processing [12]. However, it is not always possible to detect an attack. Spy agents are used in scenarios where attacks result in detectable impacts, and the detection of the origin of such attacks (i.e. one or more malicious host(s)) is inferred from examining the results of all the spy agents.

Note that it is not necessary for all the agents to arrive back in a safe environment in order to complete the analysis. A lost agent could be considered as a positive outcome, in an appropriate scenario.

One possible means of detecting host misbehaviour is the observation of privacy violations of Personally Identifiable Information (PII) included in an agent, such as email addresses. In [13] the authors implement a honeypot email system, in which decoy email addresses are uniquely registered with providers, in accordance with a mutual privacy agreement. A provider is deemed to be responsible for the reception of any unsolicited email via the associated email address, provided that the "secrecy" of this unique email address is not compromised. A host is held responsible for both attacks in which it abuses the email address itself, and indirect attacks, involving the disclosure of the decoy email address to an unauthorised third party, knowingly or unknowingly (e.g. due to poor storage security). However, the use of this honeypot email system is limited by the fact that the registation process requires manual input.

A spy agent containing such a decoy email address can both automate the process of revealing it (to more than one host) and can give a malicious host an incentive to misbehave, as described in section 1. The outcome of a spy agent test is determined by whether or not a host in the agent's route violates the confidentiality of the spy agent's email address. In this case, a positive outcome is equivalent to the reception of unsolicited email via the unique email address associated with the route.

4. Problem formulation

4.1. Assumptions and objectives

We make the following basic assumptions when designing our sets of agent routes.

- The order in which a spy agent visits a subset of target platforms is of no significance. Thus, the subset of target hosts to be visited defines a route.
- A target platform is aware of the routes of all the spy agents that visit it.
- Potentially malicious target hosts do not share any information about visiting agents, and will not collaborate to attempt to avoid detection.
- An untrustworthy spy agent will always perform an act which will enable the sender of the agent to detect that it has visited a malicious platform. That is, a spy agent visiting a malicious host will always yield a positive outcome. (This is rather a strong assumption, which we revisit in Section 6).

These assumptions enable us to consider spy agent routes as unordered sets of hosts that spy agents visit, where each set returns "positive" when it contains at least one malicious host, and "negative" otherwise. Clearly the problem of efficiently identifying the malicious (defective) hosts is identical to the classic group testing problem [14].

Optimal sets of spy agent routes can be designed using efficiency criteria, security criteria or a combination of both. In terms of efficiency, the following criteria are adapted from the group testing literature [15]:

(Eff-1) A design is deemed optimal if it can be used to test the maximum possible number of target hosts using a given number of spy agents.

(Eff-2) Alternatively, we can define a design to be optimal if it can reliably detect the maximum number of malicious hosts from amongst a given number of target hosts.

The following spying security requirements are relevant here [1]:

- Target hosts should be incapable of deciding whether or not they are dealing with spy agents;
- Potentially malicious target hosts should be given a motive to misbehave by using spy agents as "bait".

From these general requirements, the following security criteria can be derived [1].

(Sec-1) The number of hosts that each spy agent visits should be maximised. The rationale for this is that the more target hosts that a spy agent visits, the greater is the likelihood that the target host is unable to distinguish it from a "regular" agent and, hence, the more reliable the tests are.

(Sec-2) The number of hosts in common between any pair of spy agent routes should be minimised. This is based on the assumption that the target hosts will be less likely to be able to distinguish spy agents from "regular" agents if spy agents appear to be as different from one another as possible. Ideally, any two spies should visit no more than one common target host.

From the above it is clear that the spy agent route optimisation problem is a version of the classic group testing problem, modified by the inclusion of some additional security optimization criteria.

4.2. Group testing scenario hypotheses

There are two types of group testing algorithms: sequential and nonadaptive. Sequential algorithms allow later tests to use the outcomes of all previous tests. In a nonadaptive scheme, the set of tests is completely predetermined. Sequential algorithms require, in general, fewer tests, since the extra information allows for more efficient designs. On the other hand, nonadaptive algorithms can be conducted in parallel, which, in general, reduces the overall test time if not the number of tests.

The history of sequential algorithms goes back over 60 years [14]. Non-adaptive algorithms have been studied for a shorter time, including work on the hypergeometric *non-adaptive group testing* (NGT) problem [16] and d-complete designs [17]. Much recent NGT research has been driven by applications in DNA library screening in molecular biology, where the items are DNA subsequences (clones) and tests are done on pools of clones to determine which clones contain a particular DNA sequence [15]. For that reason NGT algorithms are also known as *pooling designs* [18].

The choice of the most suitable algorithm depends on the application. In this paper the following scenario hypotheses are made:

- Completing a single test may take a long time.
- A test outcome that is initially negative may, after some unidentified time, change to positive. In such cases, adaptive tests may perform poorly from both an efficiency and a security point of view.
- The assumption that malicious hosts always process agents in a detectably malicious way is more valid within a narrower time frame.

These hypotheses may characterise, for example, a scenario where each spy agent contains a unique decoy email address and tests a destination set against a

violation of PII privacy, as described in section 3. In this case, the impact of the misuse of a set of email addresses could be realised at any time after the related set of spy agents have commenced their migration.

It is clear that in scenarios such as the one described above, NGT designs are preferable. There is a rich literature on such schemes and their applications [19], including equivalent objects such as *cover-free families* [20]. Constructing optimal examples of the various classes of design is an ongoing research topic, and comparison of different schemes is a non-trivial subject [15].

In this paper we propose the use of a rather simple NGT construction, which is based on well-known combinatorial block designs, for reasons given in Section 5.2. Next, some fundamental notation and properties for such designs are given.

4.3. Notation

Let S be the set of n items (target hosts), i.e. $|S| = n$. We define a spy agent *route design*, $\mathcal{D}$, as a set system $\mathcal{D} = (S, \mathcal{R})$, where $\mathcal{R}$ is a set of subsets of S. The route design is said to be *uniform* if all routes contain the same number of hosts, i.e. $|R_i| = |R_j|$ for all $R_i, R_j \in \mathcal{R}$. The cardinality of $\mathcal{R}$, $|\mathcal{R}|$, equals the total number of spy agents.

A route design may be represented by an incidence matrix $M = (m_{ij})$, where rows are labelled by routes (tests) and columns by target hosts (items). Then, $m_{ij} = 1$ if route i contains host j, and $m_{ij} = 0$ otherwise. Each row is an incidence vector, $\mathbf{r}_i$, of the subset $\{j | m_{ij} = 1\}$, and each column is an incidence vector, $\mathbf{c}_j$, of the subset $\{i | m_{ij} = 1\}$. We then identify these row and column vectors ($\mathbf{r}_i$ and $\mathbf{c}_j$) with the routes (R_i) and hosts, so that we refer to a route $\mathbf{r}_i$ and a host $\mathbf{c}_j$. We also abuse our notation further and apply set operations to these binary vectors. For example, the union (intersection) of two (or more) incidence vectors corresponds to applying the boolean OR (AND) operation to their corresponding entries. In the same fashion, the size of an incidence vector (i.e. its Hamming weight) is equal to the cardinality of its corresponding subset, i.e. $|\mathbf{r}_i| = |R_i|$.

The set of defective items can be represented as a binary vector $\boldsymbol{\delta} = (\delta_1, \delta_2, \ldots, \delta_n)$, where $\delta_j = 1$ if host $\mathbf{c}_j$ is defective, and $\delta_j = 0$ otherwise. The outcome of all tests can be represented by a vector $\mathbf{a} = (a_1, a_2, \ldots, a_{|\mathcal{R}|})$, where $a_i = 1$ if route $\mathbf{r}_i$ contains a defective host, and $a_i = 0$ otherwise, i.e. $a_i = \{i : |\mathbf{r}_i \cap \boldsymbol{\delta}| > 0\}$. The union of all the columns corresponding to defective hosts is thus equal to the outcome vector. This follows from the fourth assumption in section 4.1, i.e. that if a host is defective then all routes containing it will yield a positive outcome.

Definition 1: Suppose that there are at most d malicious nodes in a network of n nodes ($n \geq d$). If the outcome vector, $\mathbf{a}$, of the incidence matrix, M, of a route design, $\mathcal{D}$, can be used to successfully identify all the malicious and honest nodes in the network, regardless of how they are distributed, then we call the route design a *d-classifier*.

In group testing terminology, a d-classifier $\mathcal{D}$ is a (d, n) NGT design [19]. If we assume the use of a uniform design $\mathcal{D}$, the problem of designing an optimal spy agent route design (as defined in Section 4.1) can be summarised as follows:

Problem 1: For any given d and n find a d-classifier route design $\mathcal{D}$ satisfying one or more of the following criteria (where the choice of criteria depends on the scenario):

1) For given other parameters, maximize $|R_i|$ (derived from (Sec-1)).
2) $|R_i \cap R_j| \leq 1$ for every pair of distinct routes $R_i, R_j \in \mathcal{R}$ (derived from (Sec-2)).
3) For given other parameters, maximize n (derived from (Eff-1)).
4) For given other parameters, maximize d (derived from (Eff-2)).

5. A spying classifier design

5.1. Properties of d-classifiers

In order to show how to construct a d-classifier, we first give some key properties of the incidence matrices, M, of route designs, $\mathcal{D}$. All the results in this section are taken from Kautz and Singleton [21] (who use a somewhat different terminology).

Definition 2: An incidence matrix M is said to be d-separable if the unions of the subsets of at most d columns are all distinct.

Theorem 1: A route design D is a d-classifier if and only if its incidence matrix M is d-separable.

One problem with the use of d-separable designs (i.e. route designs with a d-separable incidence matrix) is that no efficient general decoding algorithm is known for such designs. The trivial decoding algorithm (involving considering every possible subset of defective hosts) has complexity exponential in n. As a result we consider a slightly more restrictive class of designs for which we do have an simple and efficient decoding algorithm.

Definition 3: An incidence matrix M is said to be d-disjunct if the union of any set of d columns does not contain any other column as a subset.

Lemma 1: An incidence matrix M is d-disjunct if and only if, given any column $\mathbf{c}_j$ and any set of d other columns, there is at least one row in which $\mathbf{c}_j$ has a one and all the d columns have a zero.

Theorem 2: If the incidence matrix M is *d-disjunct* then M is *d'-separable* and *d'-disjunct* for all $d' \leq d$.

Theorem 3: An incidence matrix M is d-disjunct if and only if the union of all negative rows (i.e. rows that yield a negative outcome) contains all the negative columns (i.e. columns that correspond to negative items).

Theorem 3 implies that the decoding algorithm of a d-disjunct design has complexity linear in n, since items (columns) not appearing in negative rows are defective (positive). In other words, in a d-disjunct design, if a column vector is contained in the outcome vector, then this column vector is positive.

Corollary 1: The route design of a d-disjunct matrix is a d-classifier with the following decoding:

$$\boldsymbol{\delta} = \{j : \mathbf{c}_j \subseteq \mathbf{a}\}$$

5.2. A simple block design construction

There are numerous constructions for d-disjunct matrices [19], [20]. In this paper we consider constructions based on block designs.

A block design is an ordered pair $(V, \mathcal{B})$, where the set $\mathcal{B}$ is a collection of $|\mathcal{B}| = b$ subsets (blocks) of the set V, and each element of V is contained in r blocks. A t-(v, k, λ) block design (or simply a t-design) is a block design $(V, \mathcal{B})$, where $|V| = v$, $|B| = k$ for every $B \in \mathcal{B}$, and any t subset of V occurs in exactly λ blocks. For a 2-design the following equations apply [22]:

$$bk = vr \tag{1a}$$

$$\lambda(v-1) = r(k-1) \tag{1b}$$

We specifically focus on 2-designs for the following reasons:

- If we use a block design as a route design, taking blocks as hosts and elements of V as routes, then the security requirement $|R_i \cap R_j| \leq 1$ (Problem 1) is automatically satisfied by 2-$(v, k, 1)$ designs.
- It can be shown [23], [24] that for a small number of defectives, more precisely if $d \leq 2$, there are cases where 2-designs are optimal pooling designs.
- There are many known construction methods for 2-designs (see, for example, [22], [25]).

Theorem 4: An incidence matrix of a t-$(v, k, 1)$ design, where the blocks correspond to columns and the elements to rows, is $(q-1)$-disjunct, where:

$$q = \lceil \frac{k}{t-1} \rceil \tag{2}$$

Proof: By definition of a t-design and since $\lambda = 1$, two columns intersect in at most $t-1$ rows. The weight of each column is $|B| = k$. Hence, it takes the union of at least $q = \lceil \frac{k}{t-1} \rceil$ columns to cover another column [21, Theorem 6]. The result follows. □

From Theorem 4 it follows immediately that a 2-$(v, k, 1)$ design may be used as a d-disjunct route design, where:

$$d = k - 1 \tag{3}$$

This gives a simple means of constructing sets of visited hosts for spy agents. If we assume the use of a 2-$(v, k, 1)$ design, then the optimisation problem given in Problem 1 becomes rather simpler. This is because there is much less freedom to choose the parameters of the route design (essentially there are only two "degrees of freedom"). Note, though, that the reduced problem may exclude some optimal solutions.

Problem 2: Find a 2-$(v, k, 1)$ design in the following scenarios:

1) (Sec): Given the number of target hosts n $[= b]$, find a design that maximises the cardinality of each route r. This is achieved when k is minimised, since, from $(1a)$ and $(1b)$, $(k-1)r^2 + r - bk = 0$.
2) (Eff): Given the number of spy agents v, find the maximum number of target hosts b that can be tested. This is also achieved when k is minimised, since, from $(1a)$ and $(1b)$, $b = \frac{v(v-1)}{k(k-1)}$.

In Problem 2 the choice of a minimum d $[= k-1]$, gives a 2-design which is optimal both from a security and an efficiency point of view. This follows since a larger number of blocks (i.e. target platforms) implies a larger number of blocks containing an element (i.e. larger route cardinality). However, the choice of d is critical. An underestimate for the maximum number of malicious hosts means that malicious hosts will not be identified, whereas an overestimate for the maximum number of malicious hosts will compromise the efficiency and security of the route design.

5.3. Examples

The finite projective plane of order two is known as the Fano plane, and is a 2-design with $v = b = 7$ and $k = r = 3$. The corresponding spy agent system thus has seven hosts, seven tests, three hosts per test

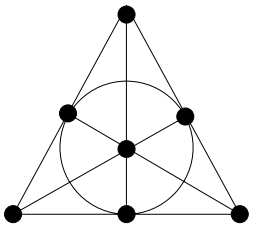

Figure 1. Fano plane

Table 1. 2-$(7, 3, 1)$ incidence matrix

	B_1	B_2	B_3	B_4	B_5	B_6	B_7
v_1	1	1	1	0	0	0	0
v_2	1	0	0	1	1	0	0
v_3	1	0	0	0	0	1	1
v_4	0	1	0	1	0	1	0
v_5	0	1	0	0	1	0	1
v_6	0	0	1	1	0	0	1
v_7	0	0	1	0	1	1	0

(and three tests per host). The Fano plane is shown diagrammatically in Figure 1. Note that each point has three lines on it and each line contains three points.

An incidence matrix for the Fano Plane is given in Table 1, which has the following seven blocks: $(V, \mathcal{B})$: $\mathcal{B} = \{B_1, B_2, B_3, B_4, B_5, B_6, B_7\}$, $B_1 = \{v_1, v_2, v_3\}$, $B_2 = \{v_1, v_4, v_5\}$, $B_3 = \{v_1, v_6, v_7\}$, $B_4 = \{v_2, v_4, v_6\}$, $B_5 = \{v_2, v_5, v_7\}$, $B_6 = \{v_3, v_4, v_7\}$ and $B_7 = \{v_3, v_5, v_6\}$.

It is trivial to verify that this matrix is 2-disjunct and is not 3-disjunct. Hence, this design will successfully identify two defectives but not three.

If, for example, the first two hosts are malicious, then the matrix in Table 1 produces an outcome vector $\mathbf{a} = \{1, 1, 1, 1, 1, 0, 0\}$. From Corollary 1, the defectives (i.e. the hosts not appearing in all negative tests, $\mathbf{r}_6$ and $\mathbf{r}_7$) are $\mathbf{c}_1$ and $\mathbf{c}_2$, which is correct. However, if the first three hosts are defectives, then the outcome vector is $\mathbf{a} = \{1, 1, 1, 1, 1, 1, 1\}$. The decoding algorithm given in Corollary 1 suggests that all hosts are defective, contradicting the implicit assumption that there are most two defectives. It is thus impossible to decide which hosts are defective.

Consider another scenario in which we require each spy route to contain at least five hosts and $d = 2$. Hence:

$$r \geq 5 \overset{(1b)}{\Rightarrow} \frac{(v-1)}{k-1} \geq 5 \Rightarrow v \geq 5k - 4 \overset{(3)}{\Rightarrow} v \geq 11$$

One set of 2-design parameters satisfying this inequality has $(v, b, r, k, \lambda) = (15, 35, 7, 3, 1)$. There are known to be at least 80 pair-wise nonisomorphic 2-designs with these parameters [25], which could, for example, be utilised for multiple re-evaluations of a set S.

6. Conclusions and future work

Remote security evaluations are unreliable by nature. Malicious hosts may selectively misbehave depending on whether or not they believe they are being evaluated. This paper defines a way of identifying the malicious hosts in a hostile network, while maximising the chance that they will misbehave. This is done by applying combinatorial group testing theory to the spy agent paradigm. A simple block design is proposed for use in constructing spy agent routes with the desired properties.

A number of possible directions for further research present themselves, including the following.

- How might the rather strong assumptions made in this paper be relaxed? More specifically:
  - What if malicious nodes behave badly in a probabilistic fashion, when the incentive for them to misbehave is maximised? What if the spying design is not even optimum?
  - Can the criteria of maximum size of routes and minimum route intersection sizes be justified in a practical scenario? What other criteria may be considered?
- What practical scenarios can be devised, and what aspects of the host trustworthiness can be evaluated?
- How might spy agents be constructed, what data could they carry, and what operations may they run remotely, e.g. in an e-commerce scenario?
- How could spy agent network scenarios be compromised by colluding malicious hosts, and how might this be addressed?

Future research will explore these areas. For example:

- A honeypot email system could be used to detect the abuse of the agent's email address privacy (section 3).
- Inconsistent behaviour could be addressed by applying error-tolerant group testing designs [23].

In general, practical scenarios may require a range of different testing mechanisms for determining outcomes, different criteria for maximising the incentive to misbehave, and different optimum route set constructions (which may not necessarily be non-adaptive). Work on the application of spy agents to specific scenarios is ongoing.

References

[1] G. Kalogridis, C. J. Mitchell, and G. Clemo, "Spy agents: Evaluating trust in remote environments," in *Proceedings of the 2005 International Conference on Security and Management*, H. R. Arabnia, Ed. Las Vegas, Nevada, USA: CSREA Press, June 20-23 2005, pp. 405–411.

[2] B. Michael, N. Rowe, M. Auguston, D. Drusinsky, R. Riehle, H. Rothstein, L. Montiero, D. Julian, G. Fragkos, E. Uzuncaova, and T. Wingfield, "Intelligent software decoys," in *Naval Postgraduate School Research*, February (special ed.) 2003, vol. 13, no. 1SE, pp. 42–44.

[3] N. Provos, "A virtual honeypot framework," in *Proceedings of the 13th USENIX Security Symposium*, August 2004, pp. 1–14.

[4] D. Du and F. Hwang, *Combinatorial Group Testing and Its Applications*, 2nd ed. World Scientific, 2000.

[5] N. Borselius, "Security for agent systems and mobile agents," in *Security for Mobility*, C. J. Mitchell, Ed. IEE, London, 2004, ch. 12, pp. 287–303.

[6] W. Jansen and T. Karygiannis, "Mobile agent security," National Institute of Standards and Technology, NIST Special Publication 800-19, August 1999.

[7] P. C. Chan and V. K. Wei, "Preemptive distributed intrusion detection using mobile agents," in *11th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*. IEEE Computer Society, June 2002, pp. 103–108.

[8] D. Dasgupta and H. Brian, "Mobile security agents for network traffic analysis," in *DARPA Information Survivability Conference and Exposition II*. Anaheim, California: IEEE Computer Society Press, June 2001, pp. 12–14.

[9] N. Borselius, C. J. Mitchell, and A. T. Wilson, "On mobile agent based transactions in moderately hostile environments," in *Advances in Network and Distributed Systems Security, Proceedings of IFIP TC11 WG11.4 First Annual Working Conference on Network Security*, B. D. Decker, F. Piessens, J. Smits, and E. V. Herreweghen, Eds. Kluwer Academic Publishers, Boston, November 2001, pp. 173–186.

[10] G. Vigna, "Cryptographic traces for mobile agents," in *Mobile Agents and Security*, ser. Lecture Notes in Computer Science, G. Vigna, Ed., vol. 1419. Springer-Verlag, Berlin, 1998, pp. 137–153.

[11] A. Corradi, M. Cremonini, R. Montanari, and C. Stefanelli, "Mobile agents integrity for electronic commerce applications," *Information Systems*, vol. 24, no. 6, pp. 519–533, 1999.

[12] O. Esparza, M. Soriano, J. L. Muñoz, and J. Forné, "A protocol for detecting malicious hosts based on limiting the execution time of mobile agents," in *Proceedings of the Eighth IEEE International Symposium on Computers and Communications*. IEEE Computer Society, 2003, pp. 251–256.

[13] J. Seigneur, A. Lambert, P. Argyroudis, and C. Jensen, "PR3 email honeypot," Department of Computer Science, University of Dublin, Tech. Rep. TCD-CS-2003-39, 2003, available at https://www.cs.tcd.ie/publications/tech-reports/reports.03/TCD-CS-2003-39.pdf.

[14] R. Dorfman, "The detection of defective members of large populations," *Annals of Mathematical Statistics*, vol. 14, no. 4, pp. 436–440, 1943.

[15] H. Q. Ngo and D. Z. Du, "A survey on combinatorial group testing algorithms with applications to DNA library screening," in *Discrete Mathematical Problems with Medical Applications*, ser. DIMACS Discrete Math. Theoret. Comput. Sci., vol. 55, 2000, pp. 171–182.

[16] F. K. Hwang and V. T. Sòs, "Non-adaptive hypergeometric group testing," *Studia Sci. Math. Hungar.*, vol. 22, pp. 257–263, 1987.

[17] K. A. Bush, W. T. Federer, H. Pesotan, and D. Raghavarao, "New combinatorial designs and their application to group testing," *Journal of Statistical Planning and Inference*, vol. 10, pp. 335–343, 1984.

[18] D. J. Balding, W. J. Bruno, E. Knill, and D. C. Torney, "A comparative survey of non-adaptive pooling designs," *Genetic Mapping and DNA Sequencing*, vol. 81, pp. 133–154, 1996.

[19] D. Z. Du and F. K. Hwang, *Pooling Designs and Nonadaptive Group Testing*. World Scientific, 2006.

[20] R. Wei, "On cover-free families," 2006, preprint available at http://ccc.cs.lakeheadu.ca/psfiles/CFF.ps.

[21] W. Kautz and R. Singleton, "Nonrandom binary superimposed codes," *IEEE Transactions on Information Theory*, vol. 10, no. 4, pp. 363–377, 1964.

[22] T. Beth, D. Jungnickel, and H. Lenz, *Design Theory*, 2nd ed. Cambridge University Press, 1999, vol. I.

[23] D. J. Balding and D. C. Torney, "Optimal pooling designs with error detection," *Journal of Combinatorial Theory, Series A*, vol. 74, no. 1, pp. 131–140, 1996.

[24] P. Erdös, P. Frankl, and Z. Füredi, "Families of finite sets in which no set is covered by the union of two others," *Journal of Combinatorial Theory, Series A*, vol. 33, pp. 158–166, 1982.

[25] C. J. Colbourn and J. H. Dinitz, *The Handbook of Combinatorial Designs*, 2nd ed. CRC Press, 2006.