EUSIPCO 2013 1569744129

ALLOCATING ADVERSARIAL RESOURCES IN WIRELESS NETWORKS

Stylianos Gisdakis, Dimitrios Katselis and Panos Papadimitratos

KTH Royal Institute of Technology
Stokcholm, Sweden
Emails: gisdakis@kth.se, dimitrik@kth.se, papadim@kth.se

ABSTRACT

A plethora of security schemes for wireless sensor networks (WSNs) has been proposed and their resilience to various attacks analyzed; including situations the adversary compromises a subset of the WSN nodes and/or deploys own misbehaving devices. The higher the degree of such intrusion is, the more effective an attack will be. Consider, however, an adversary that is far from omnipotent: How should she attack, how should she deploy her resources to maximally affect the attacked WSN operation? This basic question has received little attention, with one approach considering genetic algorithms for devising an attack strategy [5]. In this work, we recast the problem towards a more systematic treatment and more computationally efficient solutions: a combination of a genetic algorithm with a convex relaxation, and an ℓ_1-constraint formulation. The devising of near-optimal attack strategies efficiently strengthens the adversary, allowing her to adapt and mount effective and thus harmful attacks even in complex and dynamically changing settings.

***Index Terms*—** Attack, cryptographic key, genetic algorithm (GA), security.

1. INTRODUCTION

Wireless sensor networks (WSNs) cover a broad range of applications that are vulnerable to attacks. Security and resilience are therefore highly desirable and a broad gamut of security schemes for WSNs have been proposed for, e.g., cryptographic key management [4], secure communication [9], secure data aggregation [1], data confidentiality [7], and sybil attack detection [8].

In spite of the protection such schemes provide, an adversary can work her way into the network: compromise nodes and cryptographic keys, and then control those nodes and/or deploy her own devices. In principle, the more security schemes are in place, the lesser the effect of the adversarial nodes on the WSN operation. But the stronger the presence of the adversary, simply put the more nodes she controls, the more severe the harm done: the more links compromised, the more nodes masqueraded, the more bogus measurements injected.

Nonetheless, the adversary cannot be omnipotent, that is, have an overwhelming number of devices deployed or compromise the vast majority of the WSN cryptographic keys. If she did, it would be trivial to take over the WSN (or essentially deploy her own). What is interesting are powerful yet bounded adversaries. For those, the pertinent question is: How can they best deploy their "forces"? In other words, how to utilize their physical presence and cryptographic key knowledge, in order to affect the more the attacked WSN operation?

This question is important for the WSN security designers, to assess the ability of the adversary. The only investigation on this question, to the best of our knowledge, was done recently in [5], in an application- and security-protocol-agnostic manner. A solution to determine a close-to-optimal attack was proposed, combining a *genetic algorithm* (GA) with a *dynamic programming* (DP) stage. The resultant computational complexity is not, perhaps, a severe limitation if the WSN is not a large scale one or it is not mobile, or if it is not part of a dynamically changing cyber-physical system.

But, here, we want to see if the adversary can use efficient algorithms that would enable optimized attacks even under such challenging conditions. Moreover, we want to further systematize the treatment of the problem. We first approach the problem by combining a GA and a *convex relaxation* (CR) stage and then by formulating it in an ℓ_1-constraint framework [2, 3], with convex solvers to obtain solutions. We assume the problem is noiseless, in the sense that no unknown random process affects its parameters, i.e., a *deterministic* setup. In the ℓ_1-constraint case, sparse solution vectors of deterministic optimization problems have only to be derived, with solution entries known beforehand and only the support being unknown. To the best of our knowledge, this is the first time such an approach is taken on an adversarial resource allocation problem.

Next, we define the system (Sec. 2). The adversary model and the problem formulation at hand are presented in the sequel (Sec. 3). Then, we develop the (adversarial) solution to the problem, our GA-CR and the ℓ_1-constraint approaches (Sec. 4), and evaluate them through simulations (Sec. 5).

Notation. For a set $\mathcal{M}$, $|\mathcal{M}|$ denotes its cardinality. $\cup$ is the set union and $\varnothing$ the empty set. $\mathbb{N}$ is the set of nonnegative integers and $\mathbb{R}_+$ is the set of positive real numbers. Vectors

are denoted by bold lower case letters, while for a vector $\boldsymbol{a}$ its ith entry is denoted by $\boldsymbol{a}_i$ or by $[\boldsymbol{a}]_i$. For a vector $\boldsymbol{x}$ we define the index function $I(\boldsymbol{x}_i) = i$ if $\boldsymbol{x}_i \neq 0$ and 0 otherwise. Moreover, $\|\boldsymbol{x}\|_0$ is the zero "norm" of $\boldsymbol{x}$ which equals the number of nonzero entries in $\boldsymbol{x}$ and $\|\boldsymbol{x}\|_1$ is its ℓ_1-norm. $\mathbf{1}_m$ denotes the $m \times 1$ all one vector. Finally, ^{T} denotes the transposition operator.

2. SYSTEM MODEL

We model the WSN as a set of n nodes divided in N clusters, $S = \{C_1, C_2, \ldots, C_N\}$. We assume that the clusters are formed based on the requirements of the supported application and the monitored area and phenomena. The number of *benign* nodes within a cluster C_i is given by the function $F_{ben}(C_i) : S \to \mathbb{N}$, with $F_{ben}(C_i) = |C_i|$. Fig. 1 illustrates a WSN with four clusters.

Each cluster has a *utility* or *a value* for the WSN operation, with $F_{val} : S \to \mathbb{R}_+$ mapping clusters to utility values, $F_{val}(C_i) = V_i$. The total WSN utility, U_{total}, is the sum of the cluster utilities: $\sum_{i=1}^{N} V_i = U_{total}$.

The cluster value assignment is application-, deployment- and conditions-specific and we do not dwell further on it. For example, for a facility monitoring or disaster relief enabling WSN: a cluster with 3 nodes close to the point of interest (where an incident occurs) is greater than that of a 100-node cluster further away; while for an environment measuring WSN, more populated clusters could be more valuable.

We remain protocol- and security-mechanism agnostic, as in [5], and assume only that each benign node has a unique identity corroborated by a single cryptographic key. In a simple data collection setup, this can be a symmetric key shared between each WSN node and the sink. Clearly, security mechanisms leveraging these and other cryptographic keys can very well be present. But for the problem at hand, we make no further assumptions focusing on this simple setup.

3. ADVERSARY MODEL

The adversary controls R_{phy} physical devices and R_{crypto} cryptographic keys. The former can be either compromised nodes among the initially benign ones or devices the adversary possesses and deploys. The latter can be compromised keys in possession of benign nodes (e.g., read out of their memory) or keys the adversary obtained in any way but are legitimate for the WSN.

We assume that $R_{phy} \leq R_{crypto}$, that is, the adversary can equip each of her nodes with more than one keys. This, in practice, means that such a device, e.g., the one labeled "2" in C3 of Fig. 1, can masquerade two distinct nodes to the sink (and thus provide seemingly distinct measurements). We assume the adversary is not re-using any of her R_{crypto} keys, i.e., she is not assigning one of those keys to more than one of her nodes simultaneously; this ensures the adversarial nodes do not trigger a *sybil detection* scheme.[1]

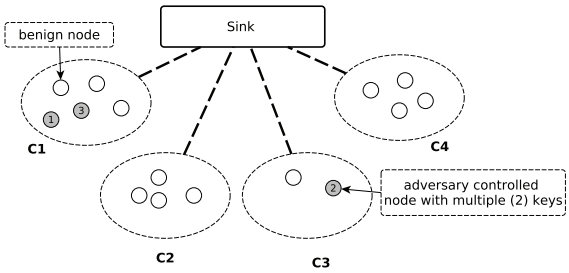

Fig. 1. A WSN under attack

We equip the adversary with the knowledge of both F_{ben} and F_{val}, that is, knowledge on the network and what is to be gained by controlling each cluster. The larger part of U_{total} the adversary controls, the more successful the attack. To achieve this, the adversary needs to control a subset of the clusters. To do so, she has to deploy across the WSN clusters her R_{phy} physical devices and distribute to them her R_{crypto} cryptographic keys to maximize her utility U_{mal}. Mathematically formulated, the **problem statement** is:

$$\max_{\pi_{phy}, \pi_{crypto}} \quad U_{mal}, \tag{1}$$

where π_{phy} and π_{crypto} denote the distribution of adversarial nodes across clusters and the distribution of keys among adversarial nodes, respectively. The adversary must first select a subset, M, of clusters to attack, $|M| \leq R_{phy}$, by allocating malicious nodes to them. Then, she must distribute the available cryptographic keys among these nodes. Both choices critically affect the achievable utility value U_{mal}.

The U_{mal} is the aggregate of the utilities of the clusters under adversarial control. "Taking over" a cluster depends on the WSN protocols and its security mechanisms in place. We abstract these away with the help of an $F_{cost} : S \to \mathbb{N}$ function, the cost in terms of physical and cryptographic resources) of controlling a cluster. Given the protocol under attack and the protection mechanism(s), one has to define an appropriate F_{cost}. Here, for simplicity in presentation and due to space limitations, we consider $R_{crypto} \geq F_{cost}(C_i) \geq F_{ben}(C_i)/2+1$, that is, an adversary is deploying in each cluster she chooses to attack a number of keys capable to become the *majority* in the cluster. In C1 of Fig. 1, she deploys two physical nodes with one and three keys respectively, thus controlling four versus three benign nodes (and keys; recall the assumed one-to-one correspondence). With such a majority, the data can be heavily influenced and even distributed protocol executions (e.g., consensus) could be under the control of the adversary.

[1] We plan to relax this condition, i.e., strengthen the adversary in partially re-using her keys across the WSN in future work.

4. ADVERSARIAL METHODS

For the cluster selection, the authors in [5] have proposed the use of genetic algorithms. The main algorithmic structure is a *chromosome* with a number of genes equal to R_{phy}. The appropriateness of a chromosome is evaluated on the basis of a *fitness function*, while evolutionary techniques of probabilistic nature are applied to force the GA to converge towards better solutions dictated by the fitness function. The mechanism driving the evolution process is based on two fundamental operators usually defined in the context of GAs, namely, the *mutation* and the *cross-over* operators. The mutation operator takes in a candidate chromosome solution and probabilistically changes one or more of its genes. The probability of changing a gene, as well as, the number of genes to be changed probabilistically are operator parameters to be decided by the adversary. The cross-over operator receives two candidate chromosome solutions and produces a new offspring candidate chromosome by combining the two parental chromosomes. The way that the offspring is produced can again be parameterized in the context of the cross-over operator by the adversary with the corresponding parameter being in this case the cross-over point. In essence, the two parental chromosomes generate a new chromosome by combining the first m genes of the one chromosome with the last $K-m$ genes of the other chromosome. Here, K is the number of genes in a chromosome and m is the cross-over point. Illustrations of these operators are given in [5].

4.1. A GA-CR Approach

Problem (1) can be approached by imitating the treatment is [5]. Defining a chromosome of size R_{phy} with each gene associated to a specific cluster, a GA is applied to a population of chromosomes to select a candidate chromosome solution, i.e., a candidate subset of clusters to attack to. The fitness function corresponds to the achieved U_{mal} by each chromosome. This value is dictated by the assignment of the cryptographic keys among the malicious nodes. The GA will be executed for an empirically selected number of evolutions or as long as U_{mal} is not improving for a predefined number of evolutions. Furthermore, to allow the possibility that an optimal cluster selection has less than R_{phy} genes, we allow the chromosomes to have *empty* genes.

The key distribution problem can be relaxed to a convex optimization problem as follows. Given a candidate chromosome solution $C' = \{C'_1, C'_2, \ldots, C'_{R_{phy}}\}$ with $C'_i \in S \cup \varnothing$ and $C'_i \neq C'_j$ when $i \neq j$ unless $C'_i = C'_j = C'_0 = \varnothing$, we define the vectors

$$\begin{aligned}\boldsymbol{f}_{val}(C') &= \left[F_{val}(C'_1), F_{val}(C'_2), \ldots, F_{val}(C'_{R_{phy}})\right]^T \\ \boldsymbol{f}_{cost}(C') &= \left[F_{cost}(C'_1), F_{cost}(C'_2), \ldots, F_{cost}(C'_{R_{phy}})\right]^T\end{aligned} \tag{2}$$

Here, we assume that $F_{val}(\varnothing) = F_{cost}(\varnothing) = 0$.

The optimal key distribution problem can be set as

$$\begin{aligned}\max_{\boldsymbol{x}} \quad & U_{mal} = \boldsymbol{f}_{val}^T(C')\boldsymbol{x} \\ \text{s.t.} \quad & \boldsymbol{f}_{cost}^T(C')\boldsymbol{x} \leq R_{crypto} \\ & \boldsymbol{x}_i \in \{0,1\}, i = 1, 2, \ldots, R_{phy},\end{aligned} \tag{3}$$

which is a $0-1$ integer linear program and it is among the Karp's 21 NP-complete problems [6]. The problem can be relaxed by replacing the last line with box constraints leading to

$$\begin{aligned}\max_{\boldsymbol{x}} \quad & \boldsymbol{f}_{val}^T(C')\boldsymbol{x} \\ \text{s.t.} \quad & \boldsymbol{f}_{cost}^T(C')\boldsymbol{x} \leq R_{crypto} \\ & 0 \leq \boldsymbol{x}_i \leq 1, i = 1, 2, \ldots, R_{phy}\end{aligned} \tag{4}$$

The last problem is convex [10] and can be efficiently solved.

Remarks:

1. The GA-CR approach is essentially the same as the one proposed in [5]. The difference is that in [5], (3) is solved by applying dynamic programming leading to a great computational burden on top of the GA algorithm, as the size of the problem increases. Formulation (4) greatly alleviates the burden of dynamic programming at the cost of losing some optimality in the achievable U_{mal}.

2. We need some sort of thresholding to obtain a final $\boldsymbol{x}$ containing binary entries. This can be achieved, e.g., by simply setting to 1 all the entries of $\boldsymbol{x}$ such that $\boldsymbol{x}_i \geq \varepsilon$, for some $\varepsilon \in [0, 1)$.

3. Due to the aforementioned thresholding operation, the obtained binary $\boldsymbol{x}$ may fail to satisfy the inequality constraint in (3) and (4). We can resolve this problem by following the principle of *least restriction*: Initially, (4) is solved with R_{crypto} and we gradually decrease R_{crypto}'s value by 1 till we obtain the first binary $\boldsymbol{x}$ satisfying $\boldsymbol{f}_{cost}^T(C')\boldsymbol{x} \leq R_{crypto}$. In this last inequality test, R_{crypto} should assume its initial (maximum) value.

4. The implication of problem (4) is that there is no need to allocate physical devices in a gene that no keys will be distributed. These devices may be reserved for future attacks by the adversary.

4.2. An ℓ_1-Constraint Approach

Instead of using the GA approach to identify candidate chromosome and key distribution solutions, we will treat both the cluster selection and the key distribution subproblems simultaneously. We define two $N \times 1$ binary vectors $\boldsymbol{c}$ and $\boldsymbol{y}$ with entries in $\{0, 1\}$. Furthermore, we can set the $N \times 1$ vectors $\boldsymbol{f}_{val}(\boldsymbol{c})$ and $\boldsymbol{f}_{cost}(\boldsymbol{c})$ such that

$$[\boldsymbol{f}_{val}(\boldsymbol{c})]_i = F_{val}(C_{I(\boldsymbol{c}_i)}), \;\; [\boldsymbol{f}_{cost}(\boldsymbol{c})]_i = F_{cost}(C_{I(\boldsymbol{c}_i)})$$

Here, we assume again that $C_0 = \varnothing$ and that $F_{val}(\varnothing) = F_{cost}(\varnothing) = 0$.

The vector $\boldsymbol{c}$ contains at most R_{phy} 1's. These 1's designate a selected cluster with index the corresponding index of the $\boldsymbol{c}_i$ entry. Similarly, the vector $\boldsymbol{y}$ contains at most R_{phy} 1's, each one designating if the adversary will or not assign cryptographic keys to the nodes of a selected cluster given our predescribed attack setup. The problem of optimal adversarial resource allocation can be posed as follows:

$$\begin{aligned} \max_{\boldsymbol{c},\boldsymbol{y}} \quad & \boldsymbol{f}_{val}^T(\boldsymbol{c})\boldsymbol{y} \\ \text{s.t.} \quad & \boldsymbol{f}_{cost}^T(\boldsymbol{c})\boldsymbol{y} \leq R_{crypto} \\ & \|\boldsymbol{c}\|_0 \leq R_{phy}, \|\boldsymbol{y}\|_0 \leq R_{phy} \\ & \boldsymbol{c}_i \in \{0,1\}, \boldsymbol{y}_i \in \{0,1\}, i = 1,2,\ldots,N \end{aligned} \quad (5)$$

The last problem is clearly hard. Nevertheless, since $\boldsymbol{c}_i \in \{0,1\}$ and $\boldsymbol{y}_i \in \{0,1\}$ for all i, it follows that

$$\|\boldsymbol{c}\|_0 = \|\boldsymbol{c}\|_1, \;\; \|\boldsymbol{y}\|_0 = \|\boldsymbol{y}\|_1 \quad (6)$$

Using (6) and removing the binary constraints, problem (5) can be relaxed to the following formulation:

$$\begin{aligned} \max_{\boldsymbol{c},\boldsymbol{y}} \quad & \boldsymbol{f}_{val}^T(\boldsymbol{c})\boldsymbol{y} \\ \text{s.t.} \quad & \boldsymbol{f}_{cost}^T(\boldsymbol{c})\boldsymbol{y} \leq R_{crypto} \\ & \|\boldsymbol{c}\|_1 \leq R_{phy}, \|\boldsymbol{y}\|_1 \leq R_{phy} \\ & 0 \leq \boldsymbol{c}_i \leq 1, 0 \leq \boldsymbol{y}_i \leq 1, i = 1,2,\ldots,N \end{aligned} \quad (7)$$

The last problem is still nonconvex due to the bilinear nature of the objective and the first constraint as well as the discrete value set of $\boldsymbol{f}_{val}(\boldsymbol{c})$ and $\boldsymbol{f}_{cost}(\boldsymbol{c})$. Nevertheless, one may observe that in our context $\boldsymbol{y}$ depends on $\boldsymbol{c}$ in the sense that it can only have 1's in positions that $\boldsymbol{c}$ has 1's. I.e., we can only assign keys to malicious nodes allocated in certain clusters. Furthermore, in our context placing nodes to clusters without assigning keys to them is useless. With this in mind, the selection of an optimal $\boldsymbol{y}$ can indicate the selection of an optimal $\boldsymbol{c}$. Therefore, a convex relaxation of (7) can be obtained as follows:

$$\begin{aligned} \max_{\boldsymbol{y}} \quad & \boldsymbol{f}_{val}^T(\boldsymbol{1}_N)\boldsymbol{y} \\ \text{s.t.} \quad & \boldsymbol{f}_{cost}^T(\boldsymbol{1}_N)\boldsymbol{y} \leq R_{crypto} \\ & \|\boldsymbol{y}\|_1 \leq R_{phy} \\ & 0 \leq \boldsymbol{y}_i \leq 1, i = 1,2,\ldots,N \end{aligned} \quad (8)$$

This problem is convex. The corresponding $\boldsymbol{c}$ can be selected to coincide with $\boldsymbol{y}$ after applying the necessary thresholding to the R_{phy} largest entries of the obtained $\boldsymbol{y}$, while zeroing the remaining $N - R_{phy}$ entries. Furthermore, the principle of *least restriction* may again be employed.

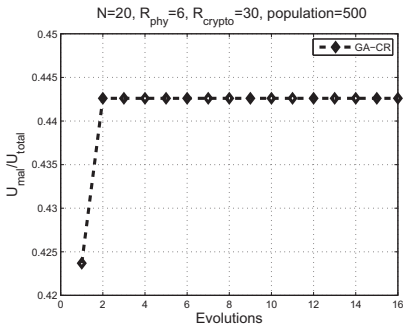

Fig. 2. Trajectory of the GA-CR approach for $N = 20, R_{phy} = 6, R_{crypto} = 30$. Furthermore, the mutation parameter is 6 and the corresponding cross-over point is 3. Thresholding with $\varepsilon = 0.1$.

5. SIMULATIONS

In this section, numerical results will be presented to illustrate the performance of the proposed approaches. We select $F_{cost}(C_i) = F_{ben}(C_i)/2 + 1$. For the GA-CR approach, we initially select a population of 500 chromosomes possibly containing empty genes. We maintain only the best 10% of them in terms of their U_{mal}. This 10% is augmented to the initial population size by using either mutation or cross-over. Mutation or cross-over are selected with probability 0.5.

In Fig. 2, a trajectory of the GA-CR algorithm with respect to the number of evolutions is presented. We assume a network with $N = 20$ clusters, each one containing from 1 to 30 nodes. The parameters R_{phy} and R_{crypto} are selected to be 6 and 30, respectively. Mutation is performed for 6 genes, while in the case of cross-over, the first 3 genes belong to the initial chromosome and the rest $R_{phy} - 3$ to a randomly selected chromosome among the current top 10%. Binary solution vectors are obtained by thresholding the obtained solution vectors with $\varepsilon = 0.1$ and by applying the least restriction principle if necessary. The final U_{mal}/U_{total} in this plot corresponds to 0.4426. This value was obtained after a long execution time due to the GA algorithm. For the same problem parameters, the ℓ_1-constraint approach was terminated in 2 CVX executions (due to least restriction) with achieved U_{mal}/U_{total} equal to 0.4426. Clearly, this example illustrates the fact that the complexity-achieved U_{mal} tradeoff is in favor of the ℓ_1-constraint approach. This would be the case even if we knew beforehand the number of evolutions needed to reach the performance saturation point, since even a single evolution of the GA algorithm has high complexity for a reasonable population size.

Focusing now on the ℓ_1-constraint approach, Fig. 3 demonstrates a very interesting aspect of this algorithm[2]. We plot the trajectory of the ℓ_1-constraint algorithm for fixed $N = 50$ and $R_{crypto} = 100$ versus R_{phy}. R_{phy} is allowed

[2]Clearly, this behavior characterizes the GA-CR algorithm, as well.

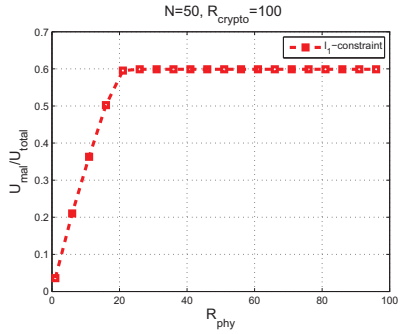

Fig. 3. ℓ_1-constraint approach for $N = 50$ and $R_{crypto} = 100$ with respect to R_{phy}. Thresholding with $\varepsilon = 0.1$.

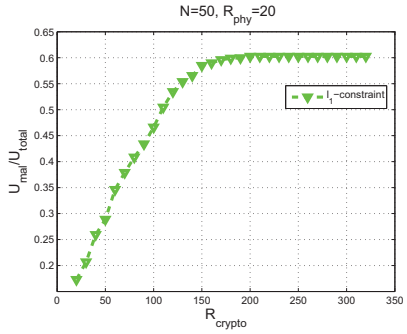

Fig. 4. ℓ_1-constraint approach for $N = 50$ and $R_{phy} = 20$ with respect to R_{crypto}. Thresholding with $\varepsilon = 0.1$

to take values in the range of 1 to R_{crypto}. We observe that as R_{phy} increases, the achievable U_{mal} tends to saturate. The saturation is quickly reached as compared with R_{crypto}. This plot essentially shows that a smart adversary should not distribute her physical devices in many different clusters when she can do it. Instead, she may use a small fraction of her devices while attacking to the network and preserve the rest of them for future use. The saturation point is dictated by R_{crypto} meaning that after placing devices to the 'best' possible clusters with respect to maximizing U_{mal} while using all the available keys, placing new devices in other clusters is meaningless if no keys are available.

Finally, the same saturation behavior seems to hold with an increasing R_{crypto} when N and R_{phy} are fixed. This is demonstrated in Fig. 4 for $N = 50$ and $R_{phy} = 20$. In this plot, the total number of nodes in the network is 806. This figure demonstrates the fact that a smart adversary should not waste a lot of resources in trying to obtain as many cryptographic keys as possible with respect to the total number of nodes in the network, since the number of nodes in the R_{phy} 'best' clusters from an attack point of view poses a limitation.

6. CONCLUSIONS

In this paper, a general setup of allocating adversarial resources to maximize the attack effect in a WSN was investigated. Two different approaches were presented to tackle the problem. The first was based on the combination of a GA with a CR stage. The second was based on an ℓ_1-constraint formulation of the problem. The complexity-attack effect tradeoff was shown to be in favor of the second approach.

7. REFERENCES

[1] J. M. Bohli, P. Papadimitratos, D. Verardi and D. Westhoff, "Resilient Data Aggregation for Unattended WSNs", *Proc. IEEE SenseApp 2011*, Bonn, Germany 4–7 Oct, 2011.

[2] D. L. Donoho, "Compressed Sensing", *IEEE Trans. on Inormation Theory*, vol. 52, no. 4, pp. 1289–1306, Apr. 2006.

[3] D. L. Donoho and M. Elad, "Optimally Sparse Representation in general (nonorthogonal) dictionaries via ℓ_1 minimization", *Proc. Natl. Acad. Sci.*, vol. 100, pp. 2197–2202, Mar. 2003.

[4] L. Eschenauer and V. D. Gligor, "A Key-Management Scheme for Distributed Sensor Networks", *Proc. ACM CCS 2002*, NY, USA, 2006.

[5] S. Gisdakis and P. Papadimitratos, "On the Optimal Allocation of Adversarial Resources", *Proc. ACM MiSeNet 2012*, Instanbul, Turkey, Aug. 2012.

[6] R. M. Karp, "Reducibility among Combinatorial Problems", in R. E. Miller and J. W. Thatcher (editors), *Complexity of Computer Computations*, New York: Plenum. pp. 85-103.

[7] J. Luo, P. Papadimitratos and J. -P. Hubaux, "GossiCrypt: Wireless Sensor Network Data Confidentiality Against Parasitic Adversaries", *Proc. SECON 2008*, San Fransisco, USA, June 2008.

[8] B. Parno, A. Perrig adn V. Gligor, "Distributed Detection of Node Replication attacks in Sensor Networks", *Proc. IEEE Symposium on Security and Privacy 2005*, Washington DC, USA, 2005.

[9] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen and D. E. Culler, "Spins: Security Protocols for Sensor Networks", *Wir. Net.*, 8, Sept. 2002.

[10] Y. Yesterov, *Introductory Lectures on Convex Optimization: A Basic Course*, Kluwer Academic Publishers, 2004.

[11] P. Papadimitratos, V. Gligor, J.-P. Hubaux, "Securing Vehicular Communications - Assumptions, Requirements, and Principles", *Proceedings of 4th Workshop on Embedded Security in Cars (ESCAR)*,2006, Berlin, Germany.