Placement optimization of IoT security solutions for edge computing based on graph theory

Tanguy Godquin, Morgan Barbier, Chrystel Gaber, Jean-Luc Grimault, Jean-Marie Le Bars

Placement optimization of IoT security solutions for edge computing based on graph theory

Tanguy Godquin[1,2], Morgan Barbier[1], Chrystel Gaber[2], Jean-Luc Grimault[2], and Jean-Marie Le Bars[1]

[1]Normandie Univ, UNICAEN, ENSICAEN, CNRS, GREYC, 14000 Caen
[2]Orange Labs, France

October 11, 2019

Abstract

In this paper, we propose a new method for optimizing the deployment of security solutions within an IoT network. Our approach uses dominating sets and centrality metrics to propose an IoT security framework where security functions are optimally deployed among devices. An example of such a solution is presented based on EndToEnd like encryption. The results reveal overall increased security within the network with minimal impact on the traffic.

1 Introduction

Since 2009, the estimated birth of the IoT by Cisco ISBG, the number of connected objects has been growing steadily, to reach in 2020, 25 (Gartner [15]), 50 (Cisco [11]) and even 200 billion devices according to Intel [2].

Besides a large amount of devices, the wide variety of devices and the multiplicity of communication protocols make the IoT a complex area to model. Different approaches consist in decreasing the variety of objects by regrouping them under taxa using taxonomies [10]. While these approaches provide an attractive level of abstraction, they do not allow for the different network topologies available in the IoT to be addressed. Because of the coexistence of numerous protocols, the IoT combines distinct network topologies that are important to model.

The communication aspect of connected objects raises important security concerns. Often constrained in terms of hardware, software, and energy, a large majority of devices do not offer sufficient security to their communications. HP reports in 2014 that 70% of devices did not perform encryption during their communications [17]. This problem is exacerbated as manufacturers who use and design connected devices are not aware of the security issues of their products [12].

The rapid increase of connected objects leads to high data production. With over 70% of all wirelessly connected objects [2], factories, businesses and health services account for the majority of uses of IoT devices. Exploiting these data usually involves significant computational capabilities, including transferring the processing and analyzing operations to dedicated servers into the cloud.

Using cloud technologies over the IoT raises several anxieties. One is related to data confidentiality, which is addressed when sensitive or identifiable data are used (e. g. video stream feedback for trajectory analysis in malls [16]). A second preoccupation is the observed latency when communicating with cloud servers

while using IoT applications. Although more than tolerable in most situations, reducing this duration is crucial in vital applications such as healthcare or autonomous driving. Perhaps the most significant issue hoisted by the use of cloud technologies is the amount of information being transmitted to servers and the lack of sufficient capacity to secure said information by the devices themselves. Overall, the IoT applications reliance on the cloud raises major security considerations, magnified by the overwhelming volume of information being produced and consumed in the area.

Concerns expressed regarding the cloud are partially mitigated with edge computing where cloud functionalities are decentralized to the network's edge. However, migrating from cloud to edge raises major concerns about service placement.

To the best of our knowledge, this article is the first one which addresses placement of security functions using graph theory. Several studies have targeted IoT, edge or fog networks but they either did not focus on security ([4, 9, 18, 23, 25]) or did not use graph-based solutions and problems [9, 18, 23, 25]. It is worth noting that Banerjee et al. [4] addresses network coverage using graph theory and centrality metrics. The authors define network coverage optimization as the minimum vertex cover problem where each edge of a graph has to have one extremity in the vertex cover set. While this problem is appropriate for network coverage, it does not cover our specific requirements (discussed in section 3).

In the present paper, we propose a framework where an object without security capabilities could entrust security measures to network edge security entities as security nodes. We model an IoT network in the form of a graph, in which security nodes are defined based on the identification of a dominating set. Two major constraints are expressed, one regarding the financial costs to adjust security nodes, the second involves the location of nodes to fit networks information flow. We therefore propose a methodology to optimize the placement of security nodes while minimizing their number. The methodology can be adapted to various security needs.

This paper is organized as follows. Section 2 presents important contextual information regarding security constraints and attack threats on the network. A first approach is discussed consisting to apply edge computing concepts to IoT security. Section 3 provides an abstract methodology for optimal positioning of edge computing security functions on IoT networks. Section 4 outlines the proposed methodology behind a given security function and observes the benefits of this approach. Finally, we conclude this study in section 5 with a summary of our contributions supplemented by perspectives for further development of the model.

2 Security model

The limited hardware capacities of IoT devices reduce access to advanced security functions such as cryptography, identity and authorization management or antivirus protection for example. A solution would be to substitute all concerned objects by equipments with suited capacities but it appears neither cost-effective nor realistic. Cloud Computing addressed this by centralizing services in powerful datacenters and servers. However, in the past few years, there has been a tendency to decentralize the functions present in the cloud, a shift that is consistent with edge computing.

In this new paradigm, processing activities are moved closer to the information producers or consumers. Some of the most well-known examples of these mechanisms are the provision of video resources by streaming platforms on servers close to users who are most likely to consume such resources [5] or processing tasks closer to information producers such as providing face recognition at the edge of the network [1].

2.1 Threat model

It is acknowledged that no absolute security exists and the security of computer systems consists often in protecting a system against a specific threat. To propose a securing method, it is, therefore, necessary to define a model that reflects the capabilities

and objectives of an attacker to protect a device effectively against that threat.

Stellios et al. [24] discuss an attacker model suitable for the IoT composed of 3 categories: device accessibility, attacker capabilities and motivations. The attacker we consider is based on the *outsider* model defined by Stellios et al. and is depicted Figure 1. He has no physical nor proximal access to the object (in range of targeted device wireless protocols). Resources of the addressed model are those of a regular person with moderate technical capacities (between neophyte and expert). Motivations of the attacker model depends on the targeted IoT network. We consider an occasional hacker or a cybercriminal whose ressources remain moderate as per the definitions of motivations and attacker profiles provided by Mosenia and Jah [19] and Onik et al. [20].

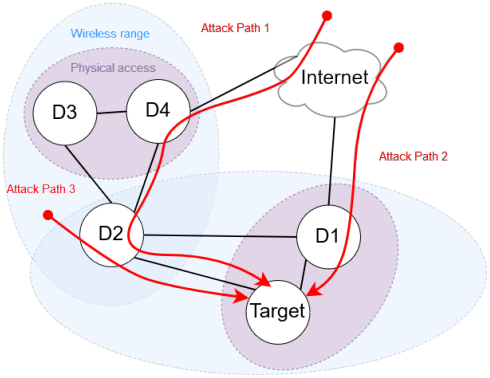

Figure 1: Positioning of the attacker on considered model.

Given a target device invisible from the Internet and the capacities of the attacker model, the considered threats use one or multiple intermediate devices to reach the target device, and the source is either on the internet or in a local network. The presence of the attacker within the wireless range or his ability to physically access the target is not considered due to the higher cost of performing such attacks.

Choosing this configuration is a trade-off on the costs of security in accordance with the proportion of attackers evicted.

2.2 Security as a Service Model for Edge Computing

To address this threat model, we propose to deploy security services at the edge of the network, thus benefitting from latency reduction, control over data processing and physical proximity of the objects. These security services can be deployed either as dedicated devices with sufficient capacities added to the considered network or as a virtualized services in existing devices or gateways. Examples of such services are cryptographic services which encrypt/decrypt messages or proxy services controlling inputs and outputs.

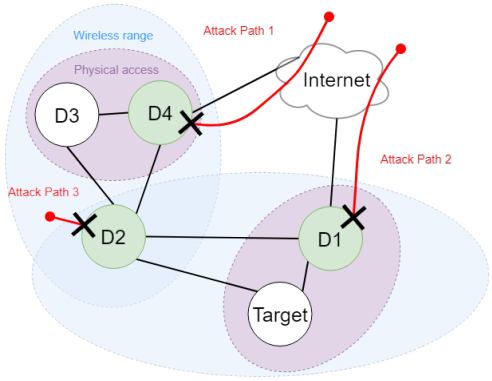

Figure 2: Example of applied security solutions (green) over considered model.

Figure 2 illustrates the deployment of security solutions to prevent threats identified in Figure 1. It is noticeable that all three attack paths (red arrows in fig. 1) have been blocked as a result of the insertion of security solutions (green circles in fig. 2) in specific devices on the network. There is no need to deploy security solutions on every device in the network, however, it must be located with regards to identified threats.

2.3 Model relevancy

To achieve the objective described in 2.2, the way security solutions are placed is of paramount concern. Indeed, it is necessary to maximize efficiency and coverage to minimize costs. Section 3 proposes an optimization method to address the positionning of security solutions in an IoT network.

Studying the most suitable locations for deploying security measures requires modelling an IoT network. Therefore, a simplified modeling is proposed, even if not all the characteristics of the IoT are covered, it is sufficient for a first study.

In the pursuit of the approach envisaged, two important elements must be considered: the bandwidth of the different communication interfaces and the inherent capabilities of IoT devices. In our study, the bandwidth of all interfaces is considered equally. However, a difference in bandwidth between interfaces could impact the way communications transit over a network.

Similarly, to further improve the positioning of security solutions, it would be advantageous to be aware of the inherent capabilities of devices. This model assumes that no single device has sufficient security capabilities by itself. It requires that the devices be improved to accommodate new security solutions. Introducing device capabilities into the security solution deployment process could thus help to favor devices that have sufficient capabilities to perform required tasks. In addition to device capabilities, device vulnerability could be considered in the deployment process to avoid deploying security solutions on objects that have a significant risk.

3 Optimization methodology

In order to propose better security at the edge of the network, it is important to define the best positioning of this security. Due to the multitude of connected devices, we first propose to adopt a more abstract representation of IoT networks by converting these networks into graphs with objects as vertices and communications as edges. This representation enables a higher degree of abstraction while at the same time providing the opportunity to introduce graph theory concepts to IoT security.

In the following paragraph, d refers to the distance (measured in number of hops) between an IoT device and its security method. Not being able to always provide security measures on objects ($d = 0$), a compromise must be made. While the usage of the security capabilities of a direct neighbor ($d = 1$) may be acceptable assuming that an attacker does not interfere with this communication, proposing security methods at a higher distance ($d \geq 2$) introduces security issues regarding the entities that are relaying the information.

Therefore, we seek to ensure that each object in our network is in direct contact with at least one device that offers security services (security node). Representing this demand on a graph would be equivalent to the identification of a dominating set.

A dominating set is a subset S of vertices of a graph G for which each vertex of G is either in S or adjacent to a vertex of S. To reduce the expense of converting devices into security nodes, one would like to identify a minimum dominating set. Finding a minimum dominating set is proved NP-Hard where even the estimation of the minimum size remains difficult [14]. Aware of this situation, we do not attempt to identify the minimum dominating set; a small set corresponding to a local minimum is sufficient.

Finding a small dominating set is not our only objective, we also aim to ensure that the positioning of the vertices making the dominating set is consistent with the transit of information in the graph. Figure 3 illustrates a bad scenario where the vertices of a dominating set (in red) are not positioned along the information path.

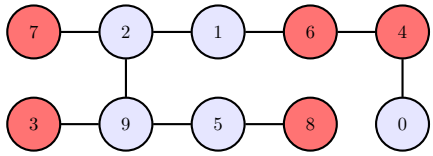

Figure 3: Dominating set (red vertices) localized off the dataflow.

We propose to improve the positioning of the vertices of the dominating set by valuing preliminarily the importance of each vertex of the graph with concepts of graph centrality.

Indeed, the importance of a vertex in a graph can be measured using centrality metrics. Depending on our security needs, some centrality metrics are more appropriate.

The dominating set vertex selection process relies on the previous evaluation to promote adding vertices of high importance (high centrality values). Our methodology could be evaluated by analyzing the appropriateness of the location of the vertices in the dominating set according to our security needs. A suggested evaluation consists of comparing the size of the assemblies obtained and examining whether the positioning of the vertices is in accordance with our safety constraints.

Once a good dominating set has been identified, IoT devices corresponding to security nodes can either be updated to deliver desired security features or be replaced if their security capabilities are not sufficient. Two devices interacting with each other are able to use those security nodes as gateways whenever it is necessary to benefit from the security functions offered by the latter ones. To ensure that communication has been carried out through one (or more if necessary) of these security gateways, the Proof Of Transit [7] could be applied. For this procedure, one or more secrets are shared via Shamir Secret Sharing to nodes where one wishes to attest the passage of the communication. The security nodes would use their secrets (or sub-secrets) to sign communication packets flowing through them. Recipients could then verify that all previously imposed signatures had been applied to communications and then attest the application of asked security methods.

4 Experiments over a specific use case

4.1 Dataset

Acquiring data related to communications between IoT objects is a difficult task. A large part of IoT datasets only contains values from various sensors. Researchers working on the concept of Social IoT have made available a few datasets containing information on devices interactions [3]. In our work, we use a graph generated from the OOR (Ownership Object Relationship) adjacency matrix containing information about public and private objects that we have filtered to keep only public static devices. The graph thus obtained is a non-oriented graph composed of 1458 vertices and 35657 edges. The vertices correspond to IoT objects while the edges are potential communications between these devices determined according to the available communication protocols of vertices (Bluetooth: 40 meters, Wifi: 400 meters and LoRa: 1,500 meters).

4.2 Centrality measures

The approach we suggest involves identifying the importance of each vertex in our network using graph centrality. Many graph centrality metrics are available, it is, therefore, important to identify which one best matches our security constraints. We concentrate our investigations on four of the most popular centrality measures, namely degree centrality, eigenvector centrality, closeness centrality, and betweenness centrality.

Degree centrality The degree centrality is a measure of the importance of a vertex according it's degree.

Eigenvector centrality The idea behind eigenvector centrality is that the importance of a vertex depends on the importance of its neighbors [6, 21]. Given a graph G, the centrality value of a vertex $i \in G$ is calculated using the following equation where n is the number of vertices

of G, A is the adjacency matrix of G and λ_{max} is the largest eigenvalue (non-negative due to the Perron-Frobenius theorem):

$$C(i) = \frac{1}{\lambda_{max}} \sum_{j=1}^{n} A_{ij} C(j).$$

Closeness centrality The closeness value corresponds to the proximity of a vertex with all vertices of the graph [22]. To compute this measure, all the shortest paths between the analyzed vertex and the rest of the vertices of the graph are studied. The centrality measurement of a vertex thus correspond to the average distance between the vertex and the other vertices of the graph. The normalized mathematical representation of this measurement is the following where n is the number of vertices in the graph and g_{ij} the geodesic distance between vertices i and j:

$$C'(i) = \frac{n-1}{\sum_{\substack{j=1 \\ i \neq j}}^{n} g_{ij}}.$$

Closeness centrality is a metric of centrality adapted when one wishes to favor the vertices from which it is faster to reach all the other vertices of the graphs. It is used when positioning fire stations in cities [8] and could also be used for the positioning of update servers in IoT networks when required.

Betweenness centrality This measurement corresponds to the number of times a vertex acts as a bridge on the shortest path between two other vertices in the graph [13]. The centrality value of a vertex i is expressed using the following equation where g_{jk} is the number of shortest paths (or geodesics) connecting the vertices j and k, $j \neq k$, and $g_{jk}(i)$ is the number of these that pass through i:

$$C(i) = \sum_{\substack{i \neq j \\ i \neq k}} \frac{g_{jk}(i)}{g_{jk}}.$$

Assuming the information in the graph flows along the shortest paths, the betweenness centrality measures the contribution of a node on a network's traffic.

Depending on the centrality measure that is used, the importance of a vertex in a graph may differ. Figure 4 corroborates this statement by displaying the vertex with the highest centrality value for the four defined measures on a given graph.

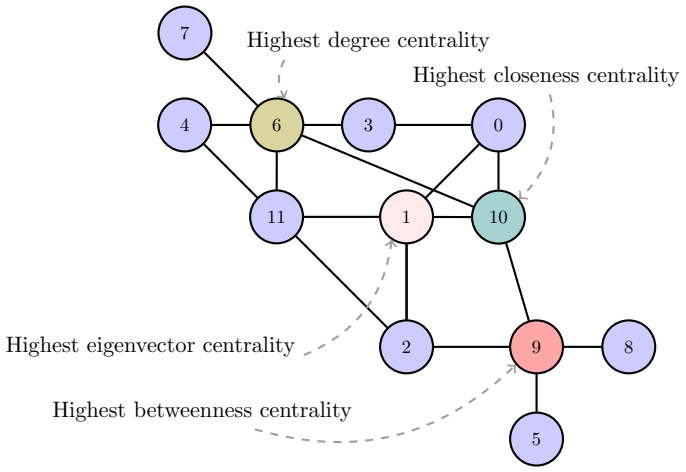

Figure 4: Difference of centrality values on a given graph.

EndToEnd encryption involves encrypting all traffic that passes from a sender to a receiver. Aiming to replicate this process with communications relayed by our security nodes, the centrality metric that theoretically seems the most appropriate is the one that takes into account the flow of information in the graph, namely the betweenness centrality.

4.3 Dominating set selection

Once the centrality values for all vertices in the graph have been calculated, our methodology suggests selecting a dominating set by favoring the addition of vertices with the highest centrality values. More precisely, the selection operates as follows: the weakest centrality value vertices are first put apart from the dominating set. A vertex is placed in the dominating set when it has no other options.

We propose to use the algorithm 2 for dominating set selection dependent on the sequence in which the

vertices of a graph are browsed. Supposing a graph $G(V, E)$ where V and E correspond respectively to the vertices and edges of G, the neighborhood of $x \in$V is defined as $N[x]$.

The execution of the algorithm involves defining several sets of vertices. One is called F and corresponds to free vertices that have not yet been browsed by the algorithm. Once the vertex is scanned, it is either transferred to the dominating set (DS), to covered vertices set (C, vertices adjacent to vertices in DS) or in the set B corresponding to browsed vertex that will later be in C but require an adjacent vertex in DS. An adjacency list, labeled $N_F(x)$ tracks the neighborhood of vertex x within F at any time. This suggested algorithm requires the following two functions (algorithm 1) to be specified: $ForcedDominant$ and $Propagation$.

Algorithm 1 Functions used on algorithm 2

```
1: function ForcedDominant(node)
2:     if N_F(node) = ∅ then
3:         Return True
4:     for all n ∈ N[node] ∩ B do
5:         if |N_F(n)| = 1 then
6:             Return True
7:     Return False
8: function Propagation(node)
9:     for all n ∈ N[node] ∩ B do
10:        Move n from B to C
```

The $ForcedDominant$ function specifies whether a vertex provided as a parameter should be added to a dominating set. To do so, it checks whether the vertex no longer has the possibility of being covered or whether one of its neighbors depends on it for its coverage. In this case, the addition in the DS package is required. The $Propagation$ function is performed when a vertex is added to the dominating set (DS). It consists of covering all the neighbors of this vertex that have already been traversed by the algorithm and that were on hold (B set).

The proposed algorithm (2) browses a given set of vertices, if the current vertex analyzed is not covered yet but has at least one neighbor to be analyzed (meaning it could be covered later by this neighbor), the latter is put on hold. When a vertex no longer has neighbors not analyzed to provide coverage, the current vertex is added to the dominating set. By providing a list of vertices sorted by ascending value of centrality as input to the algorithm ($vList$), the dominating set discovered (DS) favors vertices of high importance. Vertices with lowest centrality values are first placed in C or B providing a possibility to integrate vertices with higher centrality values latter in DS.

Algorithm 2 Central Dominating Set Algorithm.

Input:
$vList$: ordered list of V
Output:
DS: dominating set vertices

```
1: F ← V
2: DS, B, C, N_F ← ∅
3: while vList ≠ ∅ do
4:     a ← vList[0]
5:     if a ∈ F then
6:         if ForcedDominant(a) then
7:             Move a from F to DS
8:             Propagation(a)
9:         else if |N_F(a)| = 1 then
10:            b ← N_F(a)[0]
11:            Move a from F to C
12:            Move b from F to DS
13:            Propagation(b)
14:        else
15:            Move a from F to B
16:    else
17:        vList ← vList − a
18: Return DS
```

4.4 Evaluation

The algorithm introduced allows the analysis of the performance offered by applying various centrality metrics. To reflect our requirements regarding the number of security nodes and vertices positioning, we use the following rating methods: the number of vertices in the dominating set, the protection

provided through their positioning as well as a normalized value of protection independent of the graph size.

The protection provided by vertex placement on dominating set is obtained by computing the percentage of vertex pairs from the graph that have at least one shortest path secured as follows:

$$protection = \frac{\sum_{a,b} S(a,b)}{\binom{n}{2}}$$

where:

$S(a,b) = 1$ if there is a secured shortest path between vertices a and b
$S(a,b) = 0$ otherwise.

We define a shortest path secured if the vertices (sender and receiver) are adjacents or if each vertex of that pair is either member of the dominating set or adjacent to one. The illustration Figure 5 visually depicts this definition. Red vertices refer to dominating set vertices, while blue vertices are free to belong either to the set or not.

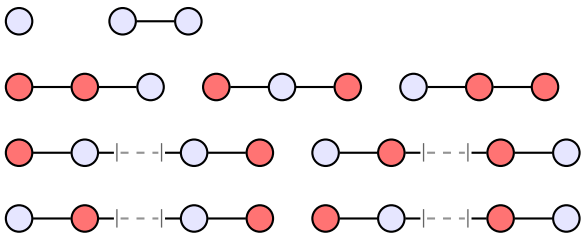

Figure 5: Illustration of secured paths

The normalized value of protection corresponds to the protection value normalized by the size of the graph over the size of it's dominating set. It is computed using the following equation:

$$normalized\ value = protection * \frac{n}{|DS|}.$$

This process is essential to consider the size of a dominating set when comparing the contributions of different methods of selection. Otherwise, a method returning a dominating set containing all vertices of a graph (100% protection ratio according to our evaluation) could be considered better than any dominating set.

Consider Figure 3, where we described the positioning of vertices not consistent with the information flow. According to our evaluation methodology, the protection provided by the dominating set placement on Figure 3 is 53%. Using the proposed approach for placement of dominating set using the betweenness centrality on the same graph, we obtain the graph in Figure 6 with a protection index of 100%. The dominating set discovered using our methodology is positioned across the information flow unlike in the one on Figure 3.

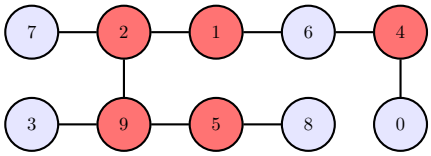

Figure 6: Dominating set (red vertices) identified using our methodology on Figure 3 graph.

4.5 Results

According to the assumption that information flows along an optimal path, using betweenness centrality appears more suitable for our EndToEnd encryption emulation security examples. Algorithm 2 was used several times with lists of vertices ordered differently, the resulting dominating sets were then examined. We focused our experiments on the contribution of centrality usage by comparing its four most popular metrics, namely closeness centrality (CC), eigenvector centrality (EC), degree centrality (DC) and betweenness centrality (BC). The proposed algorithm favors selection of vertices located at the end of the list provided, therefore these listings were sorted by increasing order of centrality. The dominating sets obtained are analyzed according to our evaluation criteria (size, protection provided by the set and protection brought by one vertex) to be compared with those identified from randomly ordered lists (R) of vertices. Comparison to randomly

ordered lists of vertices highlights the importance of vertex order in the selection of dominating sets. The results of this comparison are summarized in Table 1 below.

Table 1: Comparison of central dominating set results between centrality ordered and randomly ordered vertex lists.

Graph	Metric	Centralities				Random order	
		BC	*CC*	*DC*	*EC*	*R* seed=82	*R* seed=45
IoT dataset (1458 vertices)	DS size	**107 (7.33%)**	**107 (7.33%)**	155 (10.63%)	176 (12.07%)	191 (13.10%)	212 (14.54%)
	Protection (%)	**100**	99.92	99.95	99.91	90.87	99.16
	Normalized protection	**13.63**	13.62	9.402	8.277	6.937	6.820

4.6 Analysis on IoT dataset

Results from this study indicate that the betweenness and closeness centralities provide smaller dominating sets. It is noticeable that using centrality values in general offers better performance overall. Table 1 presents two random tests by choice of readability of the data. The dominating sets identified using random lists of vertices benefit from similar performance to Table 1 ranging from 180 to 210 vertices to a proportion of protection from 81% to 99.91%.

Dominating set sizes are about twice as small using betweenness and closeness values as when vertices are sorted in random order. The gain observed is not limited to the number of vertices of the dominating sets but also their positioning. While the majority of the results in Table 1 reveals protection level approaching 100% (only reached with betweenness centrality), it should be remembered that this value depends on the number of vertices in the dominating set. A larger dominating set is more likely to have a high protection value. Normalizing the measure according to the number of vertices of sets is therefore important for an accurate representation of gains.

Protection index provided by a vertex from our IoT reference graph using betweenness centrality is almost twice as high as for random lists. The metrics of betweenness and closeness centralities stand out particularly from this analysis by offering a gain of one-third superior to the degree centrality (highest index of all other results).

The dominating sets identified by betweenness and closeness, although highly similar from a performance point of view, are not identical. There are 18 vertices that are not shared between them, this similitude lies in the calculation methods of these metrics which both use shortest paths within a graph.

Closeness centrality attaches increasing importance to vertices that can spread information the fastest in a graph while betweenness centrality reveals vertices that are located across the information flow. We are more likely to select betweenness centrality when using EndToEnd encryption as described above. Closeness centrality cannot be excluded, however, it could very well correspond to a use case requiring the propagation of information in an IoT network.

We have decided to distinguish the centrality measurement choice from the proposed algorithm and our methodology in order to allow the user to decide which option is best suited to his security needs.

5 Conclusion and future work

We propose a strategy to effectively deploy security solutions within an IoT network while minimizing costs. Based on graph theory, we reformulate the security constraints as the identification of minimum dominating set relying on centrality notions. Our solution has a minimum impact on the information transit while respecting the assumption that it flows optimally in a graph.

Interesting perspectives could be to improve the proposed model by including other IoT features (device capabilities, risks or even bandwidth) to achieve a more realistic model. Future studies will have to investigate the role of those features when looking for optimal positioning of security solutions on the edge of IoT networks. Based on the usage and restrictions issued, additional limitations may be

applied when selecting security nodes.

Acknowledgment

The authors would like to thank Paul Dorbec and Nancy Perrot for their help regarding dominating sets and their constructive remarks.

References

[1] Face recognition: Moving more and more to the edge - asmag.com.

[2] A Guide to the Internet of Things Infographic.

[3] Social Internet of Things.

[4] P. S. Banerjee and B. Maiti. Optimality criterion for the insertion of multi-interface nodes to improve connectivity in heterogeneous IoT framework. In *2017 Devices for Integrated Circuit (DevIC)*, pages 556–560, Kalyani, India, Mar. 2017. IEEE.

[5] K. Bilal and A. Erbad. Edge computing for interactive media and video streaming. In *2017 Second International Conference on Fog and Mobile Edge Computing (FMEC)*, pages 68–73, May 2017.

[6] P. Bonacich. Power and centrality: A family of measures. *American journal of sociology*, 92(5):1170–1182, 1987.

[7] F. Brockners, S. Bhandari, S. Dara, C. Pignataro, H. Gedler, S. Youell, J. Leddy, D. Mozes, and T. Mizrahi. Proof of Transit. *Network Working Group Internet-Draft, draft-brockners-proof-of-transit-05*, May 2018.

[8] M. Dehmer and F. Emmert-Streib. *Quantitative Graph Theory: Mathematical Foundations and Applications*. Discrete Mathematics and Its Applications. CRC Press, 2014.

[9] B. Donassolo, I. Fajjari, A. Legrand, and P. Mertikopoulos. Fog Based Framework for IoT Service Provisioning. In *2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, pages 1–6, Las Vegas, NV, USA, Jan. 2019. IEEE.

[10] B. Dorsemaine, J.-P. Gaulier, J.-P. Wary, N. Kheir, and P. Urien. Internet of Things: A definition & taxonomy. In *Next Generation Mobile Applications, Services and Technologies, 2015 9th International Conference On*, pages 72–77. IEEE, 2015.

[11] D. Evans. The internet of things: How the next evolution of the internet is changing everything. *CISCO white paper*, 1(2011):1–11, 2011.

[12] S. Faux. La sécurité à l'ère des objets connectés: Comment s'y prendre ? In *Workshop "Sécurité Des Objets Connectés"*, 2017.

[13] L. C. Freeman. A Set of Measures of Centrality Based on Betweenness. *Sociometry*, 40(1):35–41, 1977.

[14] M. R. Garey and D. S. Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness. 1978.

[15] Gartner. *Gartner Says 4.9 Billion Connected "Things" Will Be in Use in 2015*. 2014.

[16] A. Ghose, B. Li, and S. Liu. Mobile Targeting Using Customer Trajectory Patterns. page 52.

[17] HP. *HP Study Reveals 70 Percent of Internet of Things Devices Vulnerable to Attack*. 2014.

[18] F. B. Jemaa, G. Pujolle, and M. Pariente. QoS-Aware VNF Placement Optimization in Edge-Central Carrier Cloud Architecture. In *2016 IEEE Global Communications Conference (GLOBECOM)*, pages 1–7, Dec. 2016.

[19] A. Mosenia and N. K. Jha. A Comprehensive Study of Security of Internet-of-Things. *IEEE Transactions on Emerging Topics in Computing*, 5(4):586–602, Oct. 2017.

[20] M. H. Onik, N. Al-Zaben, H. P. Hoo, and C.-S. Kim. A Novel Approach for Network Attack Classification Based on Sequential Questions. 2(2):14, 2018.

[21] B. Ruhnau. Eigenvector-centrality—a node-centrality? *Social networks*, 22(4):357–365, 2000.

[22] G. Sabidussi. The centrality index of a graph. *Psychometrika*, 31(4):581–603, 1966.

[23] O. Skarlat, M. Nardelli, S. Schulte, M. Borkowski, and P. Leitner. Optimized IoT service placement in the fog. *Service Oriented Computing and Applications*, 11(4):427–443, Dec. 2017.

[24] I. Stellios, P. Kotzanikolaou, M. Psarakis, C. Alcaraz, and J. Lopez. A Survey of IoT-enabled Cyberattacks: Assessing Attack Paths to Critical Infrastructures and Services. *IEEE Communications Surveys Tutorials*, pages 1–1, 2018.

[25] Y. Xia, X. Etchevers, L. Letondeur, T. Coupaye, and F. Desprez. Combining hardware nodes and software components ordering-based heuristics for optimizing the placement of distributed IoT applications in the fog. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing - SAC '18*, pages 751–760, Pau, France, 2018. ACM Press.