Robust Authentication in Trusted Sensing Networks with Physically Uncloneable Functions

Kristján Valur Jónsson[*] and Ýmir Vigfússon

Reykjavik University, School of Computer Science, Reykjavik, Iceland

Abstract. We consider the problem of secure distributed aggregation in arbitrary networked systems. In previous work, we proposed a general approach based on the principles of trusted systems theory, in which the functionality of a generic aggregation system was secured by the application of dedicated trusted devices. Here, we propose a symmetric trusted third party authentication protocol, applicable to any distributed aggregation system. We also describe a version incorporating physically unclonable functions, able to resist simulation attacks in the event of an adversary being able to breach a trusted device to the extent of extracting embedded secret keys.

Keywords: Distributed systems, secure in-network aggregation, trusted systems

1 Introduction

While new networking paradigms, such as wireless sensor networks [28], shared and participatory sensing [14] and the Internet-of-things [7], enable novel applications, they also bring unprecedented scalability challenges. One approach to cope with the scale of future networked systems and the sheer volume of information produced is to use distributed aggregation algorithms. Unfortunately, the security of this important class of algorithms is currently unsatisfactory for applications where the integrity of information is at stake. Of particular concern are stealthy insiders, those who can arbitrarily bias the aggregate computed for whatever purpose by corrupting even a single data contributor.

We have previously outlined an approach towards ensuring the integrity of the sensing and computation process in a distributed aggregation network [12]. Based on the principles of trusted systems theory, we proposed dedicated trusted sensor and aggregator modules, physically hardened to prevent tampering. We showed that correctness guarantees can be given for a distributed aggregation system by presuming that the devices are physically invulnerable and always provide verifiably correct functionality.

In this paper, we investigate these underlying assumptions more carefully by focusing on the task of entity authentication in such a system. Since a trusted device is accepted as such based on valid credentials, proper authentication is critical to enable secure aggregation in a system of the kind we described. In particular, we describe a trusted third party authentication protocol that only uses symmetric cryptographic primitives, suitable for use in any distributed aggregation system. Mutual authentication is

[*] Supported by Graduate Studies Grant #080520008 by Rannís (Icelandic Centre for Research), the Icelandic Center of Excellence in Theoretical Computer Science (ICE-TCS) and Rannís grant #090032021, and funding by Reykjavik University.

based on a symmetric private key that is permanently “burned” into a trusted device as a stamp of authenticity by a trusted third party.

A pivotal objective is to overcome a weakness whereby an adversary, given sufficient time and resources, breaches a trusted device and retrieves the secret key. Possession of the key enables the adversary to *exactly* simulate the actions of a device for the corresponding identity, including all cryptographic functions. Naturally, the unmasking of secret communication constitutes a catastrophic fault for security and integrity of the system. To counter simulation attacks of this sort, we propose to integrate the tamper-resistance measures with a physically unclonable function (PUF), thus preventing extraction of the permanent secret key. We amend the authentication protocol with a phase in which a trusted third party issues a challenge which can only be answered by a PUF, guaranteed to be altered by any physical breaches extensive enough to compromise the key. The new protocol gives resilience to cloning attacks proportional to the probability of the adversary guessing correct responses to challenges.

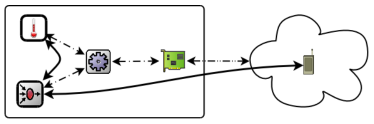
2 System Model

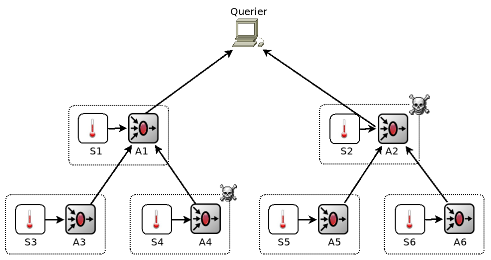
Network and Aggregation Model. We consider an arbitrary connected network graph $G = (\boldsymbol{V}, \boldsymbol{E})$. An aggregation overlay $G' = (\boldsymbol{V}', \boldsymbol{E}')$ is imposed on top of G; $\boldsymbol{V}' \subseteq \boldsymbol{V}$, $\boldsymbol{E}' \subseteq \boldsymbol{E}$. A querier q initiates a distributed aggregation protocol over G', which we assume to be composed of sensors $s \in \boldsymbol{S}$, which deliver local observations, and aggregators $a \in \boldsymbol{A}$, which merge and process information in-network; $\boldsymbol{V}' = \{q, \boldsymbol{S}, \boldsymbol{A}\}$.

We will limit our discussion to a tree-based aggregation network [15, 1, 3], although the techniques which will be described alter on are equally applicable to other families of aggregation protocols. Hence, the aggregation model is classic convergecast [20, pp. 34], in which the querier q is the root. Further, for simplicity of analysis, we assume a synchronous network for the present discussion. Sensors $\boldsymbol{S}' \subseteq \boldsymbol{S}$ deliver a set of local inputs (their observations) $\boldsymbol{I}^0$ at time zero. An aggregator a_i receives a set of inputs $\boldsymbol{I}^0_a \subset \boldsymbol{I}^0$, computes an aggregation function $y^1 = f(\boldsymbol{I}^{\boldsymbol{0}}_{\boldsymbol{a}})$ and releases as a partial aggregate update $m^1_a = \langle y^1 \rangle$ in the next round to the parent in the aggregation tree. The wave of partial aggregate updates progresses upwards until a final aggregate is computed by the root q in the last round.

Adversarial Model. Let us define *observation nodes* as generic platforms which fulfil both the functionalities of sensing and aggregation. We may assume the observation nodes are capable of establishing secure channels, thereby excluding outsiders from eavesdropping on or tampering with partial aggregate updates. However, we must assume observation nodes are inherently corruptible, a common and prudent assumption in many types of distributed systems. An observation node corrupted by the adversary is an insider, having the potential to manipulate the aggregate computation, despite pre-established secure relations. Further, we assume a *stealthy adversary* [21], one whose goal is to influence the aggregate computation long term, without risking significant probability of detecting corrupt nodes.

The querier q is inherently trusted by virtue of being the originator of queries and consumer of data. We focus on stealthy attacks whose goal is to induce the querier to


(a) *Observation node schematic.*



(b) *Secure aggregation network.*

Fig. 1: ***(a)*** *Trusted modules and channels are shown in bold. Untrusted services utilized by the aggregation process are indicated by the cogwheel and network interface card symbols.* ***(b)*** *Trusted devices, hosted by vulnerable platforms (observation nodes), form a trusted aggregation overlay.*

produce a biased aggregate. Integrity (correctness) is the primary objective. We neglect other important classes of attacks, such as denial-of-service and other availability attacks, as these run contrary to our stealthiness assumption. Attacks against the sensing process itself, or process-of-measurement attacks [23], are also neglected; we focus on attacks in the electronic signal chain from transducer head to querier.

3 Secure Aggregation with Trusted Devices

The system model described above is applicable to many families of distributed aggregation networks [15, 10, 22], which must be considered inherently vulnerable to being influenced via node compromise. Various proposals have been made towards the goal of reducing the adversaries potential [9, 2, 21, 17, 29], but none to date succeed in provably guaranteeing correctness of the aggregate computation.

Our position is that firm and tight bounds can only be placed on the adversary by establishing trust *a-priori*. In order to do so, we have proposed a system of trusted sensors and aggregators based on the principles of trusted systems theory [26], which together form a secure aggregation network [12, 13]. An observation node in such a system is shown schematically in Figure 1a. A trusted sensor and aggregator pair have established secure intra-node relations, excluding the untrusted node services indicated by the cogwheel symbol. The aggregator has established (on behalf of the observation node) secure inter-node relations with a remote peer. Figure 1b shows a schematic overview of a tree-based aggregation network. A trusted aggregation overlay (tree) is superimposed on the untrusted aggregation graph G', in which trusted sensors are leafs and aggregators are interior vertices.

The functionality of (i) trusted data production (sensing) and (ii) trusted function evaluation (aggregation) on generic untrusted observation nodes is secured by the introduction of trusted modules, sensors and aggregators. The devices are small and limited in terms of their functionality as well as circuitry and code size, implying that rigorous specification and verification of their operation is tractable. A device, whose functionality has been certified as correct by a trusted party, enables the systems designer to place trust in the correct function of the overall aggregation system, given some prerequisites are met; we refer to [12, 13] for details.

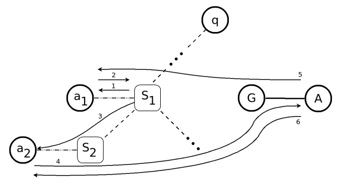


Fig. 2: *Authentication messages in a partial aggregation tree. q is the querier, while S_1 and S_2 are two untrusted observation nodes, hosting trusted aggregators a_1 and a_2.*

Assuming correct and invulnerable devices gives overall correctness guarantees: an adversary in control of an observation node is unable to influence local inputs (trusted sensors) or function evaluation (trusted aggregators). Given trusted relations between aggregators, we can take on faith that the aggregate received by q is composed entirely of correct observations, and hence, correct. We focus on correctness, as defined by Narasimha and Tsudik [16] in the context of outsourced databases, adapted to a distributed aggregation network. The complimentary objective of completeness, that is, ensuring that all observations released by sensors are included in the aggregate, is excluded from this paper for sake of brevity.

4 Entity Authentication

Let us now proceed with the task of entity authentication in a distributed aggregation network. We require a protocol which gives us mutual entity authentication as well as establishes the shared keys required for trusted relations. In order to mutually authenticate, trusted devices must make sure that their potential communicating peers are members of the same (or overlapping) group of trust, that is, verified and certified by a party which is mutually trusted or traceable to a single root of trust.

Asymmetric protocols, for instance the Station-to-Station protocol [5], enable fully distributed authentication, given per-node certificates and the ability to verify such certificates as being issued by a mutually trusted party. However, given the fact that distributed aggregation systems are by definition a single connected component enables us to consider the simpler (in terms of burdens placed on the trusted devices) solution of authenticating with the assistance of a trusted third party. This enables a relatively simple symmetric protocol to be constructed, in effect an extension of a protocol used in our prototype implementation of a secure single (trusted) aggregator system [13]. The protocol, shown schematically in Figure 2, is inspired by the classic Needham-Schroeder symmetric TTP protocol adapted to our architecture, but incorporates nonces to prevent replay attacks [18, 4, 19]

The protocol is initiated by an untrusted agent S_1 whose trusted aggregator has already joined the trusted aggregation overlay. We can say that S_1 acts as an authentication proxy for a_2 (the unassociated aggregator of S_2) for the initiation of the protocol. G is an authentication gateway, which can in practice be co-located with q.

$$S_1 \to a_1 : \langle S_2 \rangle \tag{1}$$
$$a_1 \to S_1 : \langle \mathcal{A}_{a_1 a_2} \rangle \tag{2}$$
$$S_1 \to S_2 \to a_2 : \langle \text{INVITE}, S_1, a_1, \mathcal{A}_{a_1 a_2} \rangle \tag{3}$$
$$a_2 \to S_2 \to \cdots \to G \Rightarrow A : \langle \text{AUTH}_1, a_2, \mathcal{E}_{Aa_2}(a_2, a_1, S_2, S_1, N_{a_2 a_1}, \mathcal{A}_{a_1 a_2}) \rangle \tag{4}$$
$$A \Rightarrow G \to \cdots \to S_1 \to a_1 : \langle \text{AUTH}_2, \mathcal{E}_{Aa_1}(a_1, a_2, N_{a_1 a_2}, R) \rangle \tag{5}$$
$$A \Rightarrow G \to \cdots \to S_2 \to a_2 : \langle \text{AUTH}_3, \mathcal{E}_{Aa_2}(a_2, a_1, N_{a_2 a_1}, R) \rangle \tag{6}$$

1. *Initiating event:* S_1 discovers S_2. If S_1 hosts an aggregator which is already part of the trusted overlay, then it initiates the protocol. Otherwise, it caches S_2 and waits. S_1 begins by notifying its hosted trusted module a_1 of the existence of peer S_2.
2. a_1 responds with an one-time authentication token for the potential (a_1, a_2) association, $\mathcal{A}_{a_1a_2} = \mathcal{E}_{Aa_1}(S_2, N_{a_1a_2})$; $N_{a_1a_2}$ is a nonce. The token in encrypted under the permanent secret device key K_{Aa_1} shared pairwise with A.
3. S_1 sends a INVITE message to its potential peer S_2, including the public identity of its trusted aggregator and its one-time authentication packet.
4. a_2 responds with a message intended for A, composed of the identities of the trusted and untrusted devices participating in the potential association, a nonce $N_{a_2a_1}$ and the authentication token from a_1. The message is routed end-to-end $a_2 \to A$, treating all intermediaries as outsiders. A retrieves an encryption key K for the identity a_1, decrypts and verifies the authentication request; proper verification provides implicit entity authentication based on the knowledge of a shared key. Similarly, A verifies the identity a_1 by decryption of its authentication token. Replays are prevented by verification of nonces.
5. A composes a message intended for a_1 which contains the identities of the trusted devices, the nonce $N_{a_1a_2}$ and a random number R used as key material.
6. A sends a similar message to a_2.

Both a_1 and a_2 verify nonces and derive a session key set $K_{a_1a_2} = \text{KDF}(R)$.

Key confirmation and transport key establishment. The session key $K_{a_1a_2}$ is used exclusively to set up shared keys for data transfer. The following protocol is executed right after the derivation of $K_{a_1a_2}$ to (i) confirm the key and (ii) exchange key material for generation of a transport key K_T for the trusted channel $a_1 \Leftrightarrow a_2$. Either node can in principle initiate the key confirmation, but logical choice is for the inviter a_1 to begin as described below.

$$a_1 \to S_1 \to S_2 \to a_2 : \langle \text{REKEY}, a_1, \mathcal{E}_{a_1a_2}(a_1, a_2, N'_{a_1a_2})\rangle \tag{7}$$

$$a_2 \to S_2 \to S_1 \to a_1 : \langle \text{NEWKEY}, a_2, \mathcal{E}_{a_1a_2}(a_2, a_1, N'_{a_1a_2} - 1, R')\rangle \tag{8}$$

7. a_1 sends a re-key message to a_2 (routed through untrusted hosts), containing the identities of the trusted peers and a fresh nonce. The message is encrypted by the new session key $K_{a_1a_2}$, and hence, only readable by a_1 and a_2.
   a_2 looks up the session key for a_1, decrypts and verifies the message. a_2 generates key material R' and derives transport key set $K_T = \text{KDF}(R')$ for the a_1, a_2 association. K_T is stored in the state for a_1, along with an expiration timestamp.
8. a_2 sends an encryption under the session key of the identities of the nodes (reversed), the decremented nonce (to prove a live node holds the key) and the key material R'.
   a_1 looks up the session key for the (a_1, a_2) pairing, decrypts and verifies the message, and derives a transport key set $K_T = \text{KDF}(R')$.

Re-keying protocol can be executed as often as needed. The initiating node deletes the transport key prior to initiating the protocol to signal that it is in key renewal phase.

5 PUF Challenge/response

The protocol sketched in the previous section establishes secure relations between mutually trusted devices, based on the presumed exclusivity of the per-device secret key. However, we must consider the eventuality that a crafty adversary can breach a tamper-resistant trusted device, given sufficient time and resources. It is reasonable to assume that such a breach leaves the trusted device in question inoperable, buying the adversary little in terms of his goal of biasing the aggregate. However, we must account for the eventuality of a crafty adversary being able to extract the secrets embedded in the device. Consider for example the cold-boot attack described by Halderman [8].

Extraction of secret keys enables a devastating attack against the correctness of the aggregation network. Given an extracted secret key, the adversary is able to simulate a trusted device; in keeping with proper cryptographic design principles, we assume that only the key is secret. As shown by Wagner [27], even a single data source under adversarial control is able to introduce arbitrary bias in the case of many common scalar aggregation functions, such as MIN, MAX, SUM, AVERAGE; we extend the result to a top-k aggregate in [11]. Hence, the ability to simulate even a single device represents the potential for a catastrophic failure in terms of our primary objective of guaranteed correct aggregation. We propose to counter simulation attacks by incorporating a physically unclonable function (PUF) [6, 24], assumed to be an integral part of the tamper-resistant packaging of a trusted device. A PUF is a hardware feature sufficiently unique to be used as a physical one-way function. Combined with challenge-response authentication protocols, PUFs provide strong guarantees of device integrity. Several classes of PUFs have been proposed. One class is coating-based PUFs, which are based on random doping of a dedicated layer on top of the sensor processor with dielectric particles [25], which can be incorporated into tamper-resistant shielding.

The authentication protocol is amended as follows: A includes a challenge/response pair $\mathcal{CR}_{a_1a_2} = (\mathcal{E}_{Aa_2}(C_2, N''), R_2)$ in message 5 and $\mathcal{CR}_{a_2a_1} = (\mathcal{E}_{Aa_1}(C_1, N''), R_1)$ in message 6. Note that the challenges are encrypted along with a nonce using the key of the peer in order to prevent the challenger from ever learning a challenge/response pair. The key confirmation protocol is amended as follows:

$$a_1 \to S_1 \to S_2 \to a_2 : \langle \text{REKEY-C}, a_1, \mathcal{E}_{a_1a_2}(a_1, a_2, N'_{a_1a_2}, \mathcal{E}_{Aa_2}(C_2, N''))\rangle \quad (7)$$

$$a_2 \to S_2 \to S_1 \to a_1 : \langle \text{REKEY-CR}, a_1, \mathcal{E}_{a_1a_2}(a_2, a_1, N'_{a_1a_2} - 1, N'_{a_2a_1}, \mathcal{E}_{Aa_1}(C_1, N''))\rangle \quad (8)$$

$$a_1 \to S_1 \to S_2 \to a_2 : \langle \text{REKEY-R}, a_2, \mathcal{E}_{a_1a_2}(a_1, a_2, N'_{a_2a_1} - 1, R_1)\rangle \quad (9)$$

$$a_2 \to S_2 \to S_1 \to a_1 : \langle \text{NEWKEY}, a_2, \mathcal{E}_{a_1a_2}(a_2, a_1, N'_{a_1a_2}, R_T)\rangle \quad (10)$$

7. a_1 initiates the key confirmation and includes the pre-packed challenge received from A. Note that a_1 is unable to read the challenge which is encrypted by the private device key of a_2.
8. a_2 extracts the challenge, computes $R_2 = \text{PUF}(C_2)$ and returns to a_1, along with the encrypted challenge received from A.
9. a_1 compares the received R'_2 against the R_2 received from a_2, computes $R_1 = \text{PUF}(C_1)$ and returns to a_2.

10. a_2 validates the response and finally generates key material R_T and returns to a_1.

Both trusted modules derive transport keys $K_T = \text{KDF}(R_T)$.

The protocol enables both nodes to ascertain that their peers are authentic and intact trusted modules; the probability that a simulated entity is able to produce a correct response is $p = 1/2^{l_R}$, where l_R is the length of the expected response in bits. The encryption of challenges along with a nonce ensures that the challenger (or any other party) never learns the challenge for a particular response and is hence unable to map the peers PUF.

A simulated node $\bar{a_2}$ does not have access to the PUF of the original device a_2. Hence, $\bar{a_2}$ is unable (with probability $p = 1/2^{l_R}$) to produce the correct response to a challenge served by A to a_1. The successful insertion of a simulated device must be considered an improbable event.

We may consider an alternative protocol in which A challenges each trusted device between messages 4 and 5 in the protocol. However, the number of messages remains the same, while the path length is unknown. In the proposed form, the challenge/response exchange always involves exactly three hops either way, two being internal to observation nodes.

6 Conclusions

We described a robust symmetric trusted third party authentication protocol, suitable for use in large distributed aggregation networks. Symmetric, rather than asymmetric, cryptographic primitives were chosen to reduce the burden on the resource constrained nodes expected to execute the protocol. Our assumption of the existence of a trusted third party is reasonable in a distributed aggregation network, which by definition is a connected component. Hence, we may assume some trusted infrastructure to be reachable by any node participating in such a network.

We considered the resiliency of the system to node compromise and found a critical flaw in terms of the adversary potentially being able to simulate a trusted device in the event of secret key extraction from a compromised device. This weakness was countered by the use of a challenge/response phase based on physically unclonable functions. This step allows us to claim graceful degradation in terms of our primary goal of correct aggregation.

References

1. C. Adam, K. Lim, and R. Stadler. Decentralizing network management. Technical report, KTH, December 2005.
2. H. Chan, A. Perrig, and D. Song. Secure hierarchical in-network aggregation in sensor networks. In *CCS*, pages 278–287, New York, NY, USA, 2006. ACM.
3. M. Dam and R. Stadler. A generic protocol for network state aggregation. In *RVK 05*, Linköping, Sweden, June 2005.
4. D. E. Denning and G. M. Sacco. Timestamps in key distribution protocols. *Commun. ACM*, 24(8):533–536, 1981.
5. W. Diffie, P. C. V. Oorschot, and M. J. Wiener. Authentication and authenticated key exchanges, 1992.

6. B. Gassend, D. Clarke, M. van Dijk, and S. Devadas. Silicon physical random functions. In *CCS*, pages 148–160, New York, NY, USA, 2002. ACM.
7. N. Gershenfeld, R. Krikorian, and D. Cohen. The Internet-of-Things. *Scientific American*, October 2004.
8. J. A. Halderman, S. D. Schoen, N. Heninger, W. Clarkson, W. Paul, J. A. Calandrino, A. J. Feldman, J. Appelbaum, and E. W. Felten. Lest we remember: Cold boot attacks on encryption keys. In *USENIX Security Symposium*, 2008.
9. L. Hu and D. Evans. Secure aggregation for wireless networks. *Symposium on Applications and the Internet Workshops*, pages 384–391, Jan. 2003.
10. M. Jelasity, A. Montresor, and O. Babaoglu. Gossip-based aggregation in large dynamic networks. *ACM Trans. Comput. Syst.*, 23(1):219–252, 2005.
11. K. V. Jónsson, K. Palmskog, and Ý. Vigfússon. Secure distributed top-k aggregation. In *IEEE International Conference on Communications (ICC)*, Ottawa, Canada, 2012.
12. K. V. Jónsson and Ý. Vigfússon. Securing distributed aggregation with trusted devices. In *NORDSEC*, Tallinn, Estonia, Oct. 2011.
13. K. V. Jónsson and Ý. Vigfússon. Bootstrapping trust in networked measurement systems with secure sensors. In *Sensor Applications Symposium SAS*, Brescia, Italy, Feb. 2012.
14. A. Kansal, S. Nath, J. Liu, and F. Zhao. SenseWeb: An infrastructure for shared sensing. *IEEE Multimedia*, 14(4):8–13, 2007.
15. S. Madden, M. Franklin, J. Hellerstein, and W. Hong. TAG: A Tiny AGgregation service for ad-hoc sensor networks. In *5th Symposium on Operating Systems Design and Implementation*, pages 131–146, 2002.
16. M. Narasimha and G. Tsudik. Authentication of outsourced databases using signature aggregation and chaining. In *DASFAA*, pages 420–436, 2006.
17. S. Nath, P. B. Gibbons, S. Seshan, and Z. Anderson. Synopsis diffusion for robust aggregation in sensor networks. *ACM Trans. Sen. Netw.*, 4(2):1–40, 2008.
18. R. M. Needham and M. D. Schroeder. Using encryption for authentication in large networks of computers. *Commun. ACM*, 21(12):993–999, 1978.
19. R. M. Needham and M. D. Schroeder. Authentication revisited. *SIGOPS Oper. Syst. Rev.*, 21(1):7–7, 1987.
20. D. Peleg. *Distributed Computing: A Locality-Sensitive Approach (SIAM Monographs on Discrete Mathematics and Applications 5).* SIAM, 2000.
21. A. Perrig, H. Chan, B. Przydatek, and D. Song. SIA: Secure information aggregation in sensor networks. *Journal of Computer Security*, 15(1):69–102, 2007.
22. R. Stadler, M. Dam, A. Gonzalez, and F. Wuhib. Decentralized real-time monitoring of network-wide aggregates. In *LADIS '08: Proceedings of the 2nd Workshop on Large-Scale Distributed Systems and Middleware*, pages 1–6, New York, NY, USA, 2008. ACM.
23. W. Trappe, Y. Zhang, and B. Nath. MIAMI: methods and infrastructure for the assurance of measurement information. In *DMSN*, pages 11–17, New York, NY, USA, 2005. ACM.
24. P. Tuyls and L. Batina. RFID-tags for anti-counterfeiting. In *Topics in Cryptology*, volume 3860, pages 115–131. Springer Verlag, 2006.
25. P. Tuyls, G.-J. Schrijen, B. Škorić, J. van Geloven, N. Verhaegh, and R. Wolters. Read-proof hardware from protective coatings. In *CHES*, pages 369–383, Berlin, Heidelberg, 2006. Springer-Verlag.
26. U.S. Department of Defense. Trusted computer system evaluation criteria (Orange Book), December 1985.
27. D. Wagner. Resilient aggregation in sensor networks. In *SASN*, pages 78–87, New York, NY, USA, 2004. ACM.
28. J. Yick, B. Mukherjee, and D. Ghosal. Wireless sensor network survey. *Computer Networks*, 52(12):2292 – 2330, 2008.
29. H. Yu. Secure and highly-available aggregation queries in large-scale sensor networks via set sampling. In *IPSN*, pages 1–12, Washington, DC, USA, 2009. IEEE Comp. Soc.