CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. (2011)
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cpe.1767

Energy efficient authentication strategies for network coding

Anya Apavatjrut, Wassim Znaidi, Antoine Fraboulet, Claire Goursaud, Katia Jaffrès-Runser, Cédric Lauradoux*, † and Marine Minier

Université de Lyon, INRIA, INSA-Lyon, CITI, F-69621, Villeurbanne, France

SUMMARY

Recent advances in information theory and networking, e.g. aggregation, network coding or rateless codes, have significantly modified data dissemination in wireless networks. These new paradigms create new threats for security such as pollution attacks and denial of services (DoS). These attacks exploit the difficulty to authenticate data in such contexts. The particular case of xor network coding is considered herein. We investigate different strategies based on message authentication codes algorithms (MACs) to thwart these attacks. Yet, classical MAC designs are not compatible with the linear combination of network coding. Fortunately, MACs based on universal hash functions (UHFs) match nicely the needs of network coding: some of these functions are linear $h(x_1 \oplus x_2) = h(x_1) \oplus h(x_2)$. To demonstrate their efficiency, we consider the case of wireless sensor networks (WSNs). Although these functions can drastically reduce the energy consumption of authentication (up to 68% gain over the classical designs is observed), they increase the threat of DoS. Indeed, an adversary can disrupt all communications by polluting few messages. To overcome this problem, a group testing algorithm is introduced for authentication resulting in a complexity linear in the number of attacks. The energy consumption is analyzed for cross-point and butterfly network topologies with respect to the possible attack scenarios. The results highlight the trade-offs between energy efficiency, authentication and the effective throughput for the different MAC modes. Copyright © 2011 John Wiley & Sons, Ltd.

Received 7 December 2010; Revised 21 March 2011; Accepted 22 March 2011

KEY WORDS: xor network coding; pollution attack; denial of services; MAC; trees; linear universal hash functions

1. INTRODUCTION

New techniques have emerged in information theory to improve the dissemination of information over a network. Many of these new results, e.g. data aggregation [1] or network coding [2], can be used to increase the throughput or to preserve the energy. Other paradigms like rateless codes (fountain codes [3]) improve the resiliency of the transmission to packet losses. Many of these transformations have in common that the data exchanged by the nodes are some linear combinations of the data to be transmitted. The consequence is that it is more difficult for the relaying nodes to know whether a received data is legitimate. There is an opportunity for an adversary to inject his own data. Such an attack is called a *pollution attack*. The goals of the adversary can be multiple with a pollution attack: data corruption, denial of services (DoS) or energy exhaustion. The latter, which is nothing less than another form of DoS, exploits the fact that even with some security measures, illegitimate packets are carried through the network to be discarded by the destination. Thus, the energy used to transport such data is wasted.

*Correspondence to: Cédric Lauradoux, Université de Lyon, INRIA, INSA-Lyon, CITI, F-69621, Villeurbanne, France.
†E-mail: cedric.lauradoux@insa-lyon.fr

In this paper, the security issues implied by these new communication paradigms are analyzed in the case of wireless sensor networks (WSNs). Indeed, the node resources are limited in energy and in computational power: the full range of cryptographic primitives is not generally adapted to WSNs and it is very likely that an adversary attempts to exhaust node's energy to disrupt the communications. Such motivations and the goals of the adversary go beyond the classical properties provided by cryptographic techniques. Countermeasures to pollution attacks are investigated on a WSN taking advantage of xor network coding [4].

The core mechanism used here to defeat pollution attacks is message authentication codes (MACs) such as HMAC, CBC-MAC or the designs based on universal hash functions (UHFs). We establish that MACs based on the classical primitives that are block ciphers or cryptographic hash functions implying an energetic cost too important for the relaying nodes of the network. On the contrary, MACs based on UHFs offer more flexibility for the authentication. Indeed, MACs based on UHF can be made linear: a group of messages can be authenticated with a single verification rather than operating on each message independently. Authenticating groups of messages has however the disadvantage of losing the *accountability* [5], i.e. if a single message is false in the group all the messages are considered false. DoS attacks can be launched against such an algorithm. Recovering accountability is the key to defeat DoS. A basic form of group testing is applied to achieve this goal. The construction of the group is made using binary trees.

To support our analysis, simulations have been made for the cross-point network and butterfly topologies. Then, a more formal optimization problem is proposed for the butterfly network. The aim of this study is to analyze the energy efficiency of different MAC designs with respect to the pollution attack. If energy minimization is the main performance criterion targeted by the designer, it is shown which mode of operation should be used as a function of the probability of attack.

The context of the paper is introduced in Section 2. It contains the adversary model and a description of a pollution attack for a cross-point topology. The modes of operation for MAC algorithms are analyzed in Section 3 with respect to pollution attacks. It highlights all the advantages of linearity. Section 4 shows how to combine group testing and linearity to deal with DoS. The different MAC algorithms are compared in Section 5. Simulation results are provided in Section 6 for cross-point and butterfly topologies. Finally, the related work is discussed in Section 7.

2. CONTEXT

The security problems discussed in this paper are illustrated on a network which consists of three nodes, Alice, Bob and the relay Eve (Figure 1). Eve uses network coding to improve the communication between Alice and Bob. This example is known as the cross-point topology in the network coding literature [2, 4]. We have chosen to illustrate the data authentication problem because it is easier to follow than the butterfly.

2.1. Cross-point

Alice sends the message x_1 to Bob, and reciprocally Bob sends x_2 to Alice. At time t_1, Eve has received the messages x_1 and x_2, respectively, from Alice and Bob. At time t_2, Eve broadcasts $x_1 \oplus x_2$ to Alice and Bob (Figure 1). It corresponds to a WSN in which a node Alice sends data to the sink Bob. At the same time, Bob sends a new command to Alice through Eve. Alice, Eve and

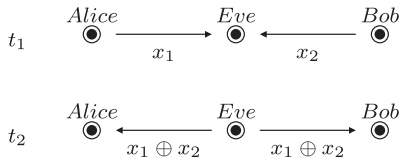

Figure 1. A cross-point in network coding.

DOI: 10.1002/cpe

Bob want to prevent an external adversary to inject corrupted packets. This type of communication exchange is often considered in WSN for data-centric routing protocols [6].

2.2. Model of the adversary

The literature on WSNs security [7] has identified two classes of threat: *outsider adversary* and *insider adversary*. The outsider adversary has the capabilities of a man-in-the-middle, i.e. he can eavesdrop and manipulate the messages exchanged between the nodes. However, he has no access to the internal state of the nodes.

The insider adversary has the opportunity to corrupt some nodes after their deployment. He has access to all the data contained or manipulated by these nodes. Such a situation occurs if the nodes are not *tamper resistant*: integrating physical protections such as those found in smart cards is too expensive.

The outsider threat can be defeated with cryptography. The insider threat is more difficult to deal with since the secrets contained in the corrupted nodes are exposed to the adversary. We focus solely on the case of the outsider adversary. We now define two particular attacks: *pollution attack* and DoS.

2.3. Pollution attacks and denial of services

Let us consider the communication scenario depicted by Figure 1 with an adversary Charlie and without security policies. Charlie can corrupt the message x_1 exchanged between Alice and Eve. Eve receives $\hat{x}_1 = x_1 \oplus \varepsilon$ instead of x_1. After receiving x_2, Eve broadcast $\hat{x}_1 \oplus x_2$. At the end of the communications, Alice and Bob obtain, respectively, $x_2 \oplus \varepsilon$ and $x_1 \oplus \varepsilon$. Such an attack is known as a *pollution attack* in the literature [8, 9]. Charlie exploits network coding to its advantage. The 'pollution' of a single message is amplified by Eve.

The role of Eve is critical to mitigate this attack. Two strategies can be devised for Eve. First, she can verify each received message. The problem of this strategy is that it does not fit well to the simplicity of network coding: Eve has to verify x_1 and x_2 which is a heavy burden. It corresponds to the AXMF mode of operation described in the following section. Second, she can sum (modulo two) all the received messages and verify the sum. It implies that the message authentication used by Eve is linear: there is only one verification. It corresponds to the XAF mode of operation which fits nicely to network coding. However, it introduces a new threat for the network: DoS attacks. Let us assume that Eve verifies the sum $\hat{x}_1 \oplus x_2$ in our pollution attack prior to any broadcasting. She detects that the sum is not correct and cancels broadcasting. The polluted message is not propagated as well as the valid one. By polluting one message, Charlie has prevented all valid messages to reach their destinations.

In the following section, different modes of message authentication are proposed for Eve to defeat pollution attacks. The problem of designing a scheme resilient to both pollution attacks and DoS is dealt with in Section 4.

3. MODES OF OPERATION FOR MESSAGE AUTHENTICATION

The primary goal of a message authentication scheme is to prevent an adversary to tamper with the messages (substitution) and to forge its own messages (impersonation). MACs, i.e. keyed cryptographic hash functions, can be used to thwart these attacks: a tag is appended to the message to check its integrity and its origin. Nowadays, three families of MAC algorithms can be found: (1) HMAC [10], and MDx-MAC [11] rely on cryptographic hash functions for which we are still waiting for the end of the SHA-3 competition[‡]; (2) CBC-MAC designs such as [12] depend on the implementation of block ciphers. In this latter case, the standard algorithm is the AES. (3) Finally, universal hash functions (UHFs) have been proposed for the design of MAC algorithms. A brief

[‡]http://csrc.nist.gov/groups/ST/hash/sha-3/.

DOI: 10.1002/cpe

introduction to UHFs is given in Appendix A as well as the design of secure MACs based on UHFs. This class of MAC algorithms offers an unconditional security, and at least the same level of performance than the two previously mentioned designs [13]. This design of MACs based on UHFs is well established in the network security community with UMAC [14] and the Galois/Counter mode (GCM) [15] of encryption.

A particular class of UHFs is considered here that is ε-almost XOR universal (ε-AXU) (see Definitions A.1 and A.2 of Appendix A). These MACs are also linear

$$h_k(x_1 \oplus x_2) = h_k(x_1) \oplus h_k(x_2) \tag{1}$$

with x_1 and x_2 two messages and k a secret authentication key shared by the network nodes. Such functions are also called homomorphic in the literature [16–18]. The design of an MAC from an ε-AXU hash function is given by the following equation:

$$h_k(m) = f_k(m) \oplus r \tag{2}$$

with f_k a universal hash function and r a random pad.

Remark 1
We consider linear MACs over $\mathbb{F}_2$ (Equation (1)). These functions are useful for xor network coding [4, 19] as shown below. Linear universal hash functions have been extended to other finite fields [16–18] in order to be used for random linear network coding.

Remark 2
Linear/homomorphic properties are well studied for encryption, i.e. $E_k(x_1 \vee x_2) = E_k(x_1) \circ E_k(x_2)$ (see [20] for instance). Stream ciphers are a well-known example of linear encryption. More details on such encryption schemes and their applications for in-network transformations can be found in [21]. For simplicity, the confidentiality problem is omitted in the remaining parts of the paper.

Three MAC designs are considered to prevent pollution attacks. Four modes of operation to protect the messages integrity are considered. They reflect the operations performed by the relay Eve: AUTHENTICATE-XOR-MAC-FORWARD (AXMF), AUTHENTICATE-XOR-FORWARD (AXF), XOR-AUTHENTICATE-FORWARD (XAF) and XOR-FORWARD (XF). The last three modes, i.e. AXF, XAF and XF, are only available for linear/homomorphic MACs.

3.1. AUTHENTICATE-XOR-MAC-FORWARD mode (AXMF)

In this mode of operation, Alice, Bob and Eve share a secret key k. Note that we do not discuss in this paper the key distribution (see [22] for instance). The AUTHENTICATE-XOR-MAC scheme can be implemented with HMAC, MDxMAC or CBC-MAC. We particularly consider the implementation of this strategy with HMAC-SHA-1 [10] used in IPV6 and AES-128-CBC-MAC. The communication consists of three steps:

1. *Data generation*. Alice computes $d_1 = h_k(x_1)$ the digest of her message, and reciprocally, Bob computes $d_2 = h_k(x_2)$. Then, the messages x_1, d_1 and x_2, d_2 are sent to Eve which receives x_1', d_1' and x_2', d_2'.
2. *Authenticate-Xor-MAC-Forward*. At time t_1, Eve has received x_1', d_1' and x_2', d_2'. Eve forwards $x_1' \oplus x_2'$ if and only if $h_k(x_1') \stackrel{?}{=} d_1'$ and $h_k(x_2') \stackrel{?}{=} d_2'$ (AUTHENTICATE). As we assume that the MAC algorithm is secured, the success of the verification step implies that $x_1' = x_1$ and $x_2' = x_2$. If these messages are successfully authenticated, she computes $m = x_1 \oplus x_2$ (XOR) and the digest of this value $d_3 = h_k(m)$ (MAC). Eve eventually broadcasts $m, h_k(m)$ to Alice and Bob (FORWARD).

DOI: 10.1002/cpe

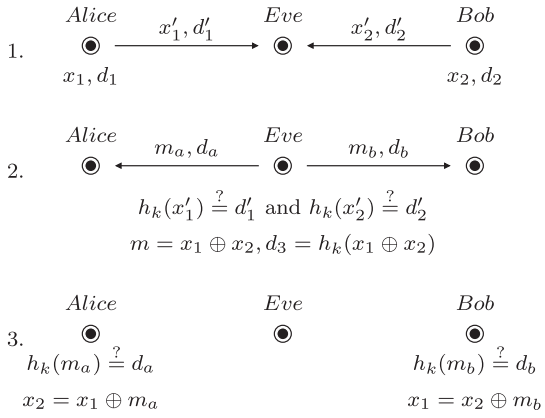

Figure 2. AUTHENTICATE-XOR-MAC-FORWARD.

3. *Delivery and verification*. Alice and Bob receive, respectively, m_a, d_a and m_b, d_b. They verify their respective values $h_k(m_a) \stackrel{?}{=} d_a$ and $h_k(m_b) \stackrel{?}{=} d_b$. When the corresponding verification succeeds, Alice obtains $x_2 = m_a \oplus x_1$ or/and Bob obtains $x_1 = m_b \oplus x_2$.

The AUTHENTICATE-XOR-MAC-FORWARD mode is described in Figure 2.

3.2. AUTHENTICATE-XOR-FORWARD mode (AXF)

Alice, Bob and Eve share a secret key k. The use of ε-AXU UHFs is assumed for the MAC and the random numbers required for the pad (Equation (2)) are produced by the AES in counter mode (AES-CTR). The Toeplitz hashing proposed by Krawczyk [23] (see Appendix A) is used in this paper for the function h. The AUTHENTICATE-XOR-FORWARD takes full advantage of the linear property (Equation (1)) of the underlying MAC. Eve does not evaluate explicitly an MAC but only 'xor' the tags of different messages. The mode is working as follows:

1. *Data generation*. This step is exactly the same as in the AUTHENTICATE-XOR-MAC-FORWARD mode.
2. *Authenticate-Xor-Forward*. At time t_1, Eve has received x'_1, d'_1 and x'_2, d'_2. Eve forwards the messages if and only if $h_k(x'_1) \stackrel{?}{=} d'_1$ and $h_k(x'_2) \stackrel{?}{=} d'_2$ (AUTHENTICATE). If the verification is successful, she computes $m = x'_1 \oplus x'_2$ (XOR). Eve needs to compute $h_k(m)$, it is obtained $d_3 = h_k(m) = d'_1 \oplus d'_2$. Eve broadcasts eventually m, d_3 to Alice and Bob (FORWARD).
3. *Delivery and verification*. This step is exactly the same as in the AUTHENTICATE-XOR-MAC-FORWARD mode.

AUTHENTICATE-XOR-FORWARD mode of operation is depicted in Figure 3.

3.3. XOR-AUTHENTICATE-FORWARD mode (XAF)

XOR-AUTHENTICATE-FORWARD mode of operation performs the verification prior to the data combination. This is possible because the combination operation is neutral for the data integrity. The idea of the XOR-AUTHENTICATE-FORWARD mode (Figure 4) is to verify the correctness of the data by checking the linear property. The mode proceeds as follows:

1. *Data generation*. This step is exactly the same as AUTHENTICATE-XOR-MAC-FORWARD. The messages $x_1, d_1 = h_k(x_1)$ and $x_2, d_2 = h_k(x_2)$ are sent to Eve.
2. *Xor-Authenticate-Forward*. At time t_1, Eve has received x'_1, d'_1 and x'_2, d'_2. Eve wants to verify the linear property between the messages and the digest: $h_k(x'_1 \oplus x'_2) = d'_1 \oplus d'_2$. She computes $m = x'_1 \oplus x'_2$ and $h_k(m)$ for the left side of the equality. Then, she computes the

DOI: 10.1002/cpe

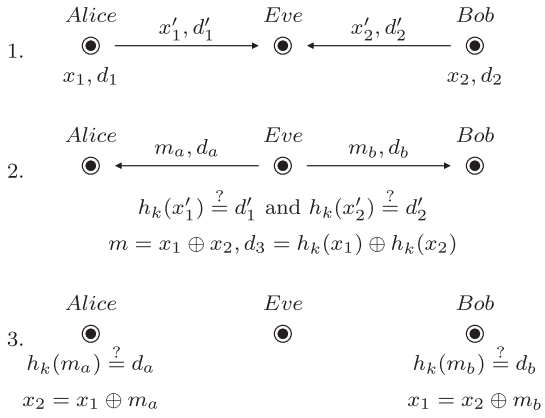

Figure 3. AUTHENTICATE-XOR-FORWARD.

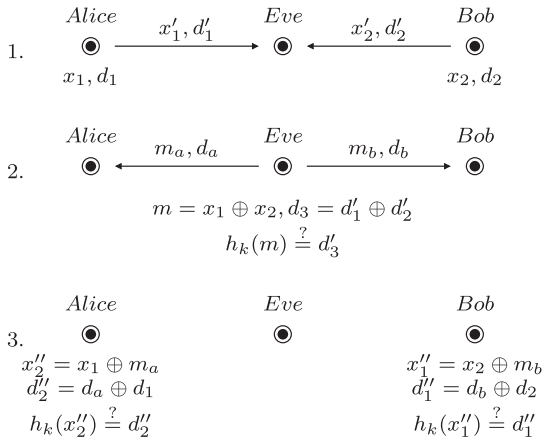

Figure 4. XOR-AUTHENTICATE-FORWARD.

sum $d_3 = d'_1 \oplus d'_2$ for the right side of the equality XOR). If $h_k(m) \stackrel{?}{=} d_3$ (AUTHENTICATE), the verification succeeds and Eve broadcasts m, d_3 to Alice and Bob (FORWARD).

3. *Delivery and verification*. Alice and Bob receive, respectively, m_a, d_a and m_b, d_b. Alice computes $x''_2 = m_a \oplus x_1$, $d''_2 = h_k(m_a) \oplus d_1$ and verifies that $h_k(x''_2) \stackrel{?}{=} d''_2$. The same computation and verification are performed by Bob: $x''_1 = m_b \oplus x_2$, $d''_1 = m_b \oplus d_2$ and verifies that $h_k(x''_1) \stackrel{?}{=} d''_1$.

3.4. XOR-FORWARD mode (XF)

The XOR-FORWARD mode postpones any verification to the legitimate receiver of the message: Eve does not authenticate the messages she receives. The generation step and the delivery/verification step are unchanged compared to the XAF mode. When Eve has received x'_1, d'_1 and x'_2, d'_2, she broadcasts $x'_1 \oplus x'_2$, $d'_1 \oplus d'_2$ to Alice and Bob without further verification. The XOR-FORWARD has two advantages for the relaying node: Eve does not need to know the secret key k which eventually simplifies the key distribution and she does not do any verification.

When no attack occurs, Eve has nothing to compute. The drawback is that the polluted packets are still forwarded in the case of an attack. This can waste the energy and the time of all the nodes. The XF and XAF mode could be combined together in the network to find an appropriate trade-off between the energetic cost when no attack occur and the energetic cost when an adversary injects corrupted packets.

DOI: 10.1002/cpe

3.5. Security remarks

Two security issues are to be discussed for our schemes: *replay attacks* and *malleability*. They were skipped in the previous section for clarity. The first attack is independent of the MAC algorithm. The second attack exploits the linearity property of UHFs to forge messages by eavesdropping several communications of a given user. Both attacks are defeated by the proper use of the random pad in Equation (2).

Replay attacks. The schemes described previously are secure if a single exchange needs to be supported, i.e. the defense against replay attacks is not detailed. Replay can be easily defeated by including *number used once* (nonce) [24]. These nonces can be implemented at a very low cost by a counter. The random pad r of Equation (2) can be computed using the counter mode (CTR) of a block cipher. The counter encrypted by the block cipher ensures that messages can not be replayed. It is worth mentioning that Alice, Bob and Eve must have their counters synchronized.

Malleability. This property of all linear/homomorphic encryption schemes such as RSA can be used by an adversary to forge messages. Let us consider that Alice sends at time t_1 the couple x_1, d_1 to Eve. At time t_2, Alice sends x_2, d_2. At time t_3, an adversary may substitute the message of Alice x_3, d_3 with $x_1 \oplus x_2$, $d_1 \oplus d_2$. The substitution of the adversary succeeds if:

$$f_k(x_1 \oplus x_2) \oplus r_3 = f_k(x_1) \oplus f_k(x_2) \oplus r_1 \oplus r_2,$$

$$r_3 = r_1 \oplus r_2.$$

In the same way, an adversary could withdraw a part of the sent messages. A classical and simple solution to discard this problem is to pad the message in a dependent way with a random number that must also be transmitted [25]. This method could also be used in our case to discard the malleability problem. Note also that if a mechanism is provided against replay attacks, an adversary cannot correctly forge a nonce or a counter and thus the attack induced by malleability is discarded.

4. XAF MODE WITH GROUP TESTING

Let us consider a network topology in which Eve has to combine n messages. The AXMF and AXF preserve the traffic, i.e. if α messages are invalid then $n-\alpha$ messages are forwarded. AXMF and AXF modes eliminate the invalid messages by testing them individually at the cost of a high complexity. The complexity given in this section are relative to the number of times the function f_k is computed (see Equation (2)).

The XAF mode operates differently. If the n messages are valid, they are combined and forwarded. In the XAF mode, a single attack $\alpha > 1$ is enough to prevent Eve from forwarding any messages (DoS). If Eve uses AXMF or AXF, Eve becomes silent if and only if n messages are polluted. Thus on the one hand, AXMF and AXF modes run in $\Theta(n)$ but the complexity of the DoS is in $\Theta(n)$. On the other hand, XAF runs in $O(1)$ but suffers from DoS attacks in $O(1)$. Fortunately, we do not have to choose between the lesser of the two evils. A trade-off between these modes can be found. Based on *group testing*, a modification of XAF has a running time which depends on α ($O(n)$) and the complexity of the DoS attack is in $O(n)$.

Group testing. It consists of finding efficiently α defective items amongst n. It has many applications, originating in World War II to detect diseases from blood samples. It has been well-studied in combinatorics [26] and many important results are summarized in [27]. The problem found in network coding is a special case of group testing in which α, i.e. the number of attacks, is unknown. The solution prescribed for this problem is known as *binary splitting* in the group testing literature. Group testing is particularly well-suited to our problem because XAF behaves like group signature algorithms based on homomorphic encryption [5, 28].

Let us assume that Eve has received four messages x_i and their corresponding authentication codes $d_i = f_k(x_i) \oplus r_i$. Binary splitting applied to our problem implies for Eve to compute two binary trees: the left tree (Ltree) is computed over the messages x_i (Figure 5) and the right tree

DOI: 10.1002/cpe

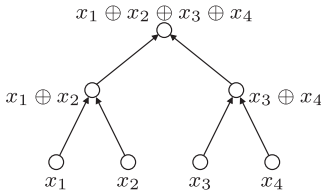

Figure 5. Ltree (messages).

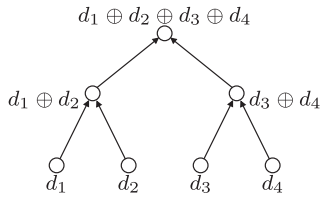

Figure 6. Rtree (authentication).

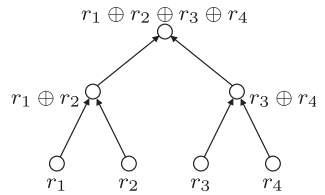

Figure 7. Ptree (random pad).

(Rtree) is computed over the authentication codes d_i (Figure 6). The detection of the invalid messages is done by comparing these two trees using h_k. For this purpose, an auxiliary tree (Ptree) containing the corresponding random pad is needed (Figure 7). The verification is an in-order traversal of the trees. After verifying the root node, the left subtree is visited and then the right subtree. The verification of the root consists of combining the root of the Ltree with the Ptree to compute $h_k(x_1 \oplus x_2 \oplus x_3 \oplus x_4)$. Then, Eve checks whether:

$$\underbrace{f_k(x_1 \oplus x_2 \oplus x_3 \oplus x_4) \oplus r_1 \oplus r_2 \oplus r_3 \oplus r_4}_{h_k(x_1 \oplus x_2 \oplus x_3 \oplus x_4)} \stackrel{?}{=} d_1 \oplus d_2 \oplus d_3 \oplus d_4.$$

The right side of the previous equation corresponds to the root of the Rtree. The left side is the sum modulo two of the root node of the Ltree and the Ptree. This verification pattern is applied recursively to the nodes of the tree. Each time a verification succeeds, the exploration of the considered subtree (or tree) is stopped.

Complexity. If no attack occurs $\alpha = 0$, Eve needs to apply only once the function f. If $\alpha > 1$, the complexity is given by the number of nodes of the smallest full binary tree that includes all the incorrect d_i's (Figure 8). In the worst case, all the nodes of the tree need to be checked. It requires $2n-1$ computations of f and it can be reached when $\alpha \geq n/2$. This worst-case complexity can be reduced to n by a wise use of the linearity as shown in the following remarks. In conclusion, the $\Theta(n)$ complexity of AXF and AXMF is changed to $O(n)$ with XAF associated to group testing. The best case needs a single computation.

Remark 3

Let us consider the case of an Ltree with two leaves (x_1 and x_2). During the verification, Eve computes first $h_k(x_1 \oplus x_2)$. Let us assume that this value does not match $d_1 \oplus d_2$. Eve explores the

DOI: 10.1002/cpe

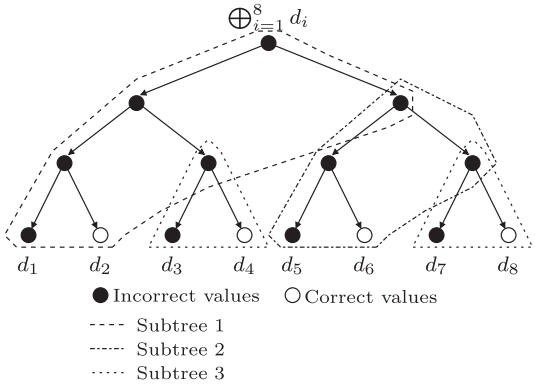

Figure 8. The figure details how the worst-case complexity is reached on an Rtree. The subtree 1 must be explored to detect that d_1 does not match x_1. If a second message is incorrect, the largest full binary tree that must be explored by Eve is the concatenation of the subtree 1 with the subtree 2. The exploration of subtree 2 corresponds to an incorrect value in the sum $\bigoplus_{i=5}^{8} d_i$, etc.

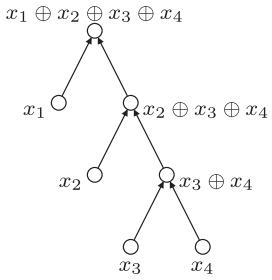

Figure 9. Ltree with priority level.

left leaves, computes $h_k(x_1)$ and checks its correctness. Whatever the result is, she needs also to check the right leaves. Eve does not need to compute $h_k(x_2)$ thanks once again to the linearity of the function f_k:

$$h_k(x_2) = h_k(x_1 \oplus x_2) \oplus h_k(x_1).$$

To verify this tree, Eve has only computed the function f_k two times. Generalizing this result to larger trees implies that the worst-case complexity is treated with only n computations of f_k.

Remark 4
The binary tree can be adapted to take into account messages that are more critical than others. For such messages, it may be important to know as fast as possible whether they are valid. The Ltree described in Figure 9 corresponds to a scenario in which x_1 has the highest priority, then x_2 and finally x_3, x_4 have the lowest priority level. This modification of the tree has no effect on the complexity.

Remark 5
The memory complexity of this method is $6n-3$.

To conclude, Eve can use the XAF mode with group testing to check the authentication code of each message received. The worst-case complexity is in n and it can be reached for $\frac{n}{2}$ polluted messages. To prevent an adversary to force the exploration of the whole tree, Eve must choose the positions in the tree of the messages x_i and their corresponding authentication codes d_i randomly.

DOI: 10.1002/cpe

5. ENERGY PERFORMANCE OF MAC

In this section, we consider the energy performance of the different modes of protection against pollution attack.

5.1. Complexity

Before describing simulation results, we give a theoretical comparison between CBC-MAC and XAF mode with and without group testing for the authentication of messages of arbitrary length b.

To authenticate a message of b blocks of 128 bits, b applications of the AES are required. Eve needs $(n+1)b$ computations of the AES to apply the AXMF mode on n messages: n verifications and one computation for the authentication code of sum modulo two of all the messages. Let us assume that the cost of computing the Toeplitz hashing is equivalent to δb computations of the AES with $\delta \in [0, 1]$. For the XAF mode without group testing, the cost is $\delta b + n$ since n applications of the AES are required for the random pad. With group testing, it is $n(1+\delta b)$ in the worst case. The complexity of the different modes is summarized in Table I.

5.2. Simulation setup

Performance results have been obtained using the WSim and WSNet open-source simulation tools [29]. Performance estimations have been performed on an MSP430-based platform similar to TelosB nodes (Senslab v1.4). These nodes use a Ti MSP430f1611 micro-controller and a Ti CC2420, 802.15.4 compliant, radio devices among others.

WSim[§] is a full system emulator that takes as input the full ELF firmware generated by cross-compilation tools. The toolchain used in the experiment is GCC 4.4.3 port for MSP430[¶]. Execution of the embedded software is cycle accurate. Time evaluation is also accurate thanks to the full emulation of the micro-controller clocking system. When coupled with WSNet[‖], an event driven network and physical layer simulator, WSim can estimate the correctness of applications and drivers for network devices at a byte precise level. These two simulation tools have been used to benchmark and generate non-intrusive execution traces of each MAC algorithm.

The eSimu energy estimation tool [30] can then estimate the energy profile of the execution trace. eSimu uses micro benchmarks to build platform energy profiles. eSimu achieves an error rate less than 5% for energy estimation and annotation at source level (C code) when used for full applications (including complex devices such as radio transmitter or flash memories). The error rate drops to less than 1% for computationally intensive algorithms for which only the energy spent by the micro-controller is computed.

Power consumption is mainly due to the MAC implementation and its associated complexity. Current micro controllers used in wireless sensor nodes and tiny devices do not contain complex hardware CPU features such as caches or bitwise operations. This lack of hardware support for complex cryptographic primitives must be accounted in the design of solutions for such devices.

5.3. Comparison of MACs

As a preliminary, we provide the performance of the studied MAC algorithms. The data to be authenticated is 20 bytes long. The key k used is 128 bits long. HMAC provides digests of 160 bits while AES-CBC-MAC and Toeplitz hashing provide digests of 128 bits due to their inherent designs. The main characteristics of the computation of the different MACs are summarized in Table II. The HMAC algorithm consists of two calls to a cryptographic hash function (SHA-1). For AES-CBC-MAC, two calls to a block cipher (AES-128) are done ($b=2$). For the Toeplitz hash, we need to compute the universal hash function h_k and to xor the result with a nonce produced by the AES-128 in the CTR mode (one call to AES-128).

[§] http://wsim.gforge.inria.fr/.
[¶] http://mspgcc4.sf.net/.
[‖] http://wsnet.gforge.inria.fr/.

DOI: 10.1002/cpe

Table I. Computational cost of the different modes in terms of calls to the AES and Xor.

Mode	AES (equivalent)	Xor
AXMF (CBC-MAC)	$(n+1)b$	$n-1$
AXF	$n(1+\delta b)$	$2n-2$
XAF (no group testing)	$\delta b+n$	$3n-3$
XAF (group testing)	$n(1+\delta b)$	$3n-3$
XF	0	$n-1$

Table II. Characteristics of the MACs for 20 bytes plaintexts.

		Calls to	
MAC	Output size (bits)	AES-128	SHA-1
HMAC	160	0	2
CBC-MAC	128	2	0
Toeplitz	128	1	0

Table III. Evaluation of MACs on TI MSP430 at 8 MHz.

Algorithm	Energy in 10^{-4} J	Delay in ms
SHA-1	0.623	4.64
AES-128	0.473	3.53
HMAC	2	14.84
CBC-MAC	1.476	11.045
Toeplitz	1.201	8.976

The processing time and the energy consumption of HMAC, AES-CBC and Toeplitz are given in Table III along with the performances of AES-128 and SHA-1. We also test a 512-bit exponentiation to give an order of the cost of homomorphic signatures based on exponentiation. This operation is performed in 15 052 ms and it consumes 1000 times the energy of HMAC.

The Toeplitz hashing reduces the energy consumption by 65% over HMAC and 23% over AES-CBC.

6. SIMULATION RESULTS

To compare the efficiency of the different modes, two network topologies are considered: the cross-point and the butterfly.

6.1. Cross-point

A detailed analysis of the energy consumption of the nodes is given in Figure 10(a)–(e). The case of a communication without attack is first considered. The energy consumed by a node is decomposed into four parts:

1. reception cost: all the radio operations performed by a node to receive a packet,
2. verification cost: all the computations to authenticate a packet,
3. transformation cost: it includes the combination of the packets and the generation of the tags associated to them,
4. emission cost: all the radio operations performed by a node to emit a packet.

DOI: 10.1002/cpe

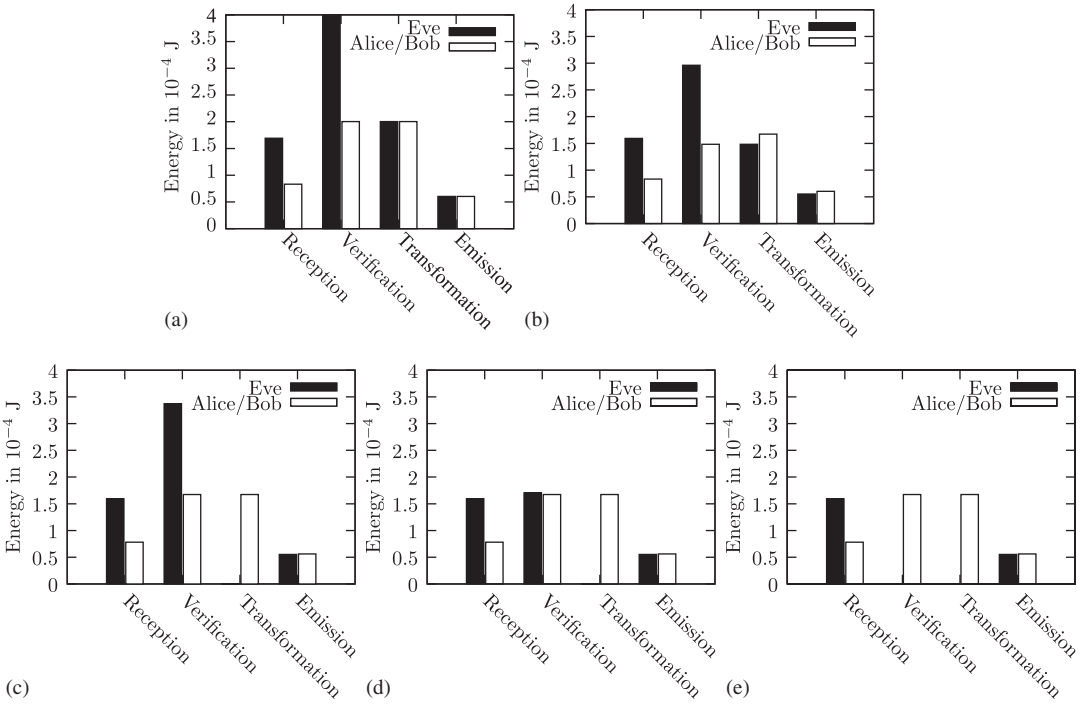

Figure 10. Energy cost for the different modes of operation in a cross-point topology: (a) AXMF with HMAC; (b) AXMF with CBC-MAC; (c) AXF with Toeplitz hash; (d) XAF with Toeplitz hash; and (e) XF with Toeplitz hash.

The AXMF mode with either HMAC (Figure 10(a)) or CBC-MAC (Figure 10(b)) are the most restricting ones for the relaying node. Eve consumes with HMAC more energy to relay (8.29×10^{-4} J) than Alice or Bob to generate and emit the data (5.53×10^{-4} J). The situation is almost the same with CBC-MAC. The situation is fairer with AXF (Figure 10(c)): 5.51×10^{-4} J are consumed by Eve and 4.68×10^{-4} J by Alice or Bob. XAF without group testing (Figure 10(d)) and XF (Figure 10(e)) provide, respectively, an improvement of 18 and 55% over AXF.

The reception and emission costs are almost the same for all the modes. They become very important when we distinguish the case of communication with or without attack. Indeed, Eve relays information depending on the success of her verification. This operation implies at least one emission for Eve and two receptions, one for Alice and one for Bob. The energy consumption of all the nodes of the network is given in Figure 11(a) for AXF, XAF and XF. The reader must keep in mind that the energy consumption when the communication is not attacked corresponds to a case in which Alice and Bob obtain data at the end. The energy consumed by the nodes ($E_{noattack}$) produces the expected results. When the network is under attack, Alice and Bob are not going to achieve the expected results and all the energy (E_{attack}) is wasted. XF is the most interesting mode when no attack occurs while XAF performs better under attack.

The choice of a mode of operation depends on the probability p of a pollution attack to occur. From the energetic cost of a mode $\mathcal{M}$, the average energy consumption can be defined as:

$$p\cdot E_{attack}(\mathcal{M})+(1-p)\cdot E_{noattack}(\mathcal{M}),$$

where $E_{attack}(\mathcal{M})$ and $E_{noattack}(\mathcal{M})$ denote, respectively, the energy spent by all the nodes of the network with the mode $\mathcal{M}$ with or without an attack. For the network of Figure 1, the probable energy spent for AXF, XAF and XF is given in Figure 11(b).

The best mode of operation for preserving the energy is XOR-FORWARD if $p<0.31$. Otherwise, XOR-AUTHENTICATE-FORWARD should be used.

DOI: 10.1002/cpe

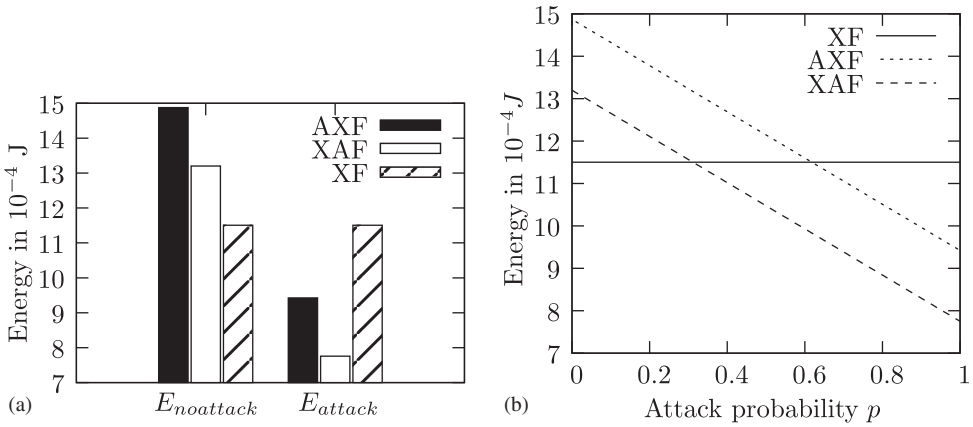

Figure 11. (a) Energy consumption of the network with/without an attack and (b) average energy consumed by the overall network for the different modes with different attack probabilities (cross-point).

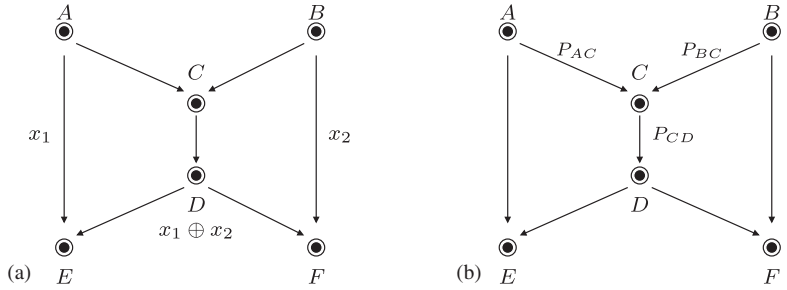

Figure 12. (a) Butterfly network and (b) attack probability for the critical links.

Table IV. Authentication strategies for C and D (butterfly).

Strategy	Mode for C	Mode for D
S_0	XF	F
S_1	XF	AF
S_2	AXF	F
S_3	AXF	AF
S_4	XAF	F
S_5	XAF	AF

6.2. *Butterfly network*

The cross-point topology has one node between the source and the destination. With a butterfly topology (Figure 12(a)), there are two relaying nodes C and D. These two nodes do not have the same role in the network and therefore not the same control on the authentication which is a major difference with the cross-point scenario.

The node C is in charge of combining the messages coming from A and B. C is called the *coding node*. The coding node has three authentication modes available (AXF, XAF and XF). In the same way, node D is a *relay node* and may or may not authenticate the messages. The strategies available to D are namely AUTHENTICATE-FORWARD (AF) and FORWARD(F).

There are six possible strategies for C and D. They are denoted S_i and are summarized in Table IV.

DOI: 10.1002/cpe

Remark 6
The number of strategies available to authenticate the messages is: $3^c \times 2^d$, where c and d are, respectively, the number of coding nodes and relay nodes. The exhaustive search approach used for the cross-point and for the butterfly is not scalable for larger networks unless proper optimization tools are derived.

Similar to the cross-point example, the security threat is modeled in the butterfly network by a *probability of attack* p_{ij} on the link (i, j). With the butterfly example, the significant attack probabilities are for the links (A, C), (B, C) and (C, D) (Figure 12). Indeed, attacks on the links (A, E), (B, F) (D, E) and (D, F) are systematically detected by the destinations E and F. Hence, our analysis concentrates on attacks occurring on links (A, C), (B, C) and (C, D).

Forwarding probabilities. Knowing the authentication strategies, the average amount of energy spent in the network can be derived. Therefore, it is first necessary to derive the probability for a node to forward a packet. If C (or D) decides to authenticate the messages, its *forwarding probability* f_C (resp. f_D) is defined as the opposite of the probability that a packet received by C (resp. D) has been corrupted.

It is assumed as well that all nodes having more than one incoming link are coding nodes and nodes with a single incoming link are basic relays. The authentication strategy of a node i and the probability P_{ki} determine the forwarding decision f_i with P_{ki} the probability that a received message from neighbor k is corrupted. A relay authenticating a message forwards it with probability $f_i = (1 - P_{ki})$.

An authenticating coding node forwards a message depending on its mode of operation. If it performs XAF, it only forwards an encoded packet when all received messages are correct. If a single incoming packet has been corrupted, all packets arriving in the same time slot are dropped. Consequently, node C decides to forward its packet with probability $f_i = \prod_{k \in \mathcal{N}_1(k)} (1 - P_{ki})$ in XAF mode, where $\mathcal{N}_1(k)$ is the set of 1-hop neighbors of i.

A coding node using AXF is able to authenticate individually all incoming packets. As a consequence, only the good packets are XOR-ed together and forwarded. The probability for a node with AXF to forward a packet is equal to the probability $f_i = 1 - \prod_{k \in \mathcal{N}_1(k)} P_{ki}$.

The probability of receiving a corrupted message P_{ki} depends on the previous hops on the path from a source to i, i.e. on the forwarding probabilities of the previous hop relays and their modes of operation. It is possible to compute in parallel the values of f_i and P_{ki}, starting with the derivation of the f_i's for the nodes directly connected to the sources. Then, P_{ki} can be derived for all nodes located at 2 hops from the sources, which gives the information needed to derive the f_i values for these 2-hop distant nodes.

Table V gives the derivation of the forwarding probabilities (f_C, f_D) for all the six authentication strategies. For instance, consider S_5. Node C forwards a packet with probability $f_C = (1 - p_{AC})(1 - p_{BC})$ and node D forwards a packet with probability $1 - p_{CD}$ since all the packets it receives from C can only be polluted on link (C, D).

Energy derivation. As shown before, forwarding probabilities are derived from the strategies. Knowing these forwarding probabilities, it is possible to count the average number of messages that are transmitted and received in the network when the sources A and B transmit one message

Table V. Derivation of the forwarding probabilities (f_C, f_D) for the six S_i (butterfly).

Strategy	f_C; f_D
S_0	$f_C = f_D = 1$
S_1	$f_C = 1$; $f_D = (1 - p_{AC})(1 - p_{BC})(1 - p_{CD})$
S_2	$f_C = 1 - p_{AC} \cdot p_{BC}$; $f_D = 1$
S_3	$f_C = 1 - p_{AC} \cdot p_{BC}$; $f_D = 1 - p_{CD}$
S_4	$f_C = (1 - p_{AC})(1 - p_{BC})$; $f_D = 1$
S_5	$f_C = (1 - p_{AC})(1 - p_{BC})$; $f_D = 1 - p_{CD}$

DOI: 10.1002/cpe

Table VI. Energy costs in (10^{-4} J) for the atomic actions performed in the network.

One message transmission	Q_T	0.556851
One message reception	Q_R	0.7995405
XOR between two messages	Q_{XOR}	0.00003135
UHF-based MAC (one message)	Q_A	1.686154

each. It is possible to derive a precise energy consumption related to the security processing at nodes C and D as well.

To define our problem, we need to know the cost of each atomic action in the network, namely Q_T, Q_R, Q_{XOR} and Q_A. Table VI gives the corresponding values measured using eSimu for the system architecture described in Section 5.2.

The energy costs of XAF and AXF are different. The authentication cost at node i for the XAF mode is equal to the cost of authenticating a single packet ($Q_{XAF}(i) = Q_A \cdot \mathbf{P}_i^{Rec}$) since authentication is performed on the XOR-ed version of the incoming packets. $\mathbf{P}_i^{Rec} = 1 - \prod_{k \in \mathcal{N}_1(i)}(1 - f_k)$ is the probability of node i to receive at least one packet from its one hop neighbors.

$Q_{AXF}(i)$ is a function of the number of packets received at node i since each incoming packet is authenticated individually. It is derived as: $Q_{AXF}(i) = Q_A \cdot N_i$, where N_i is the average number of received messages at node i. N_i is derived according to $N_i = \sum_{k \in \mathcal{N}_1(i)} \mathbf{P}_k$, where $\mathbf{P}_k$ is the probability for a message (corrupted or not) to be sent by a 1-hop neighbor node k. This probability is derived knowing the f_i values for all the paths between the sources and the destinations that use relay k.

The total energy spent by the system with strategy S_j is denoted by $E(S_j)$. It derives the energy spent by the transmission of two messages sent by the sources $\mathcal{O} = \{A, B\}$ to the destinations $\mathcal{D} = \{E, F\}$ for S_j. It is defined by

$$E(S_j) = E_{\mathcal{O}}(S_j) + E_{\mathcal{R}}(S_j) + E_{\mathcal{D}}(S_j) \tag{3}$$

where $E_{\mathcal{O}}(S_j)$, $E_{\mathcal{R}}(S_j)$ and $E_{\mathcal{D}}(S_j)$ are the energy costs relative to the source, relay and destination actions, respectively.

All sources are computing an authentication code for transmitting a message, we have:

$$E_{\mathcal{O}}(S_j) = |\mathcal{O}| \cdot Q_T = 2(Q_A + Q_T).$$

At the destination, energy is consumed by packet reception and authentication. Hence, we have:

$$E_{\mathcal{D}}(S_j) = \sum_{d \in \mathcal{D}} N_d \cdot (Q_R + Q_A),$$

where N_d is the number of messages received at destination d for strategy S_j.

The energy expenditure for the relays $\mathcal{R}$ is the sum of the energy spent by each relay in receiving, transmitting and authenticating when required. It is given by:

$$E_{\mathcal{R}}(S_j) = \sum_{i \in \mathcal{R}} N_i \cdot Q_R + f_i \cdot Q_T + a_i \cdot E_A(i),$$

where a_i is a binary variable which is equal to one if node i authenticates and zero otherwise.

The function $E_A(i)$ gives the energy consumption related to the authentication. If node i is a relay node (no coding) and is authenticating, $E_A(i) = Q_A$. If the node is a coding node that is authenticating, we have

$$E_A(i) = Q_{XOR} \cdot (N_i - 1) + t_i \cdot Q_{XAF}(i) + (1 - t_i) \cdot Q_{AXF}(i), \tag{4}$$

where t_i is a binary variable which is equal to one if node i uses XAF and zero if i uses AXF.

Table VII gives the closed-form expressions for the average energy expenditure of the network for the six authentication strategies.

DOI: 10.1002/cpe

Table VII. Closed-form expressions of the energy expenditure $E(S_j)$ for the six strategies.

Strategy	$E(S_j)$
S_0	$4\cdot Q_T+5\cdot Q_R+4\cdot Q_A$
S_1	$[3+(1-p_{AC})(1-p_{BC})(1-p_{CD})]Q_T+[3+2\cdot(1-p_{AC})(1-p_{BC})(1-p_{CD})]Q_R$ $+[3+2(1-p_{AC})(1-p_{BC})(1-p_{CD})]Q_A$
S_2	$[4-2\cdot p_{AC}\cdot p_{BC}]Q_T+[5-3\cdot p_{AC}\cdot p_{BC})]Q_R+Q_{XOR}$ $+[3+2(1-p_{AC}\cdot p_{BC})]Q_A$
S_3	$[2+(1-p_{AC}\cdot p_{BC})(2-p_{CD})]Q_T+[2+(1-p_{AC}\cdot p_{BC})(3-2p_{CD})]Q_R$ $+Q_{XOR}+[4+2(1-p_{AC}\cdot p_{BC})(1-p_{CD})]Q_A$
S_4	$[2+2(1-p_{AC})(1-p_{BC})]Q_T+[2+3(1-p_{AC})(1-p_{BC})]Q_R$ $+Q_{XOR}+[3+2(1-p_{AC})(1-p_{BC})]Q_A$
S_5	$[2+(1-p_{AC})(1-p_{BC})(2-p_{CD})]Q_T+[2+(1-p_{AC})(1-p_{BC})(3-2p_{CD})]Q_R$ $+Q_{XOR}+[4+2(1-p_{AC})(1-p_{BC})(1-p_{CD})]Q_A$

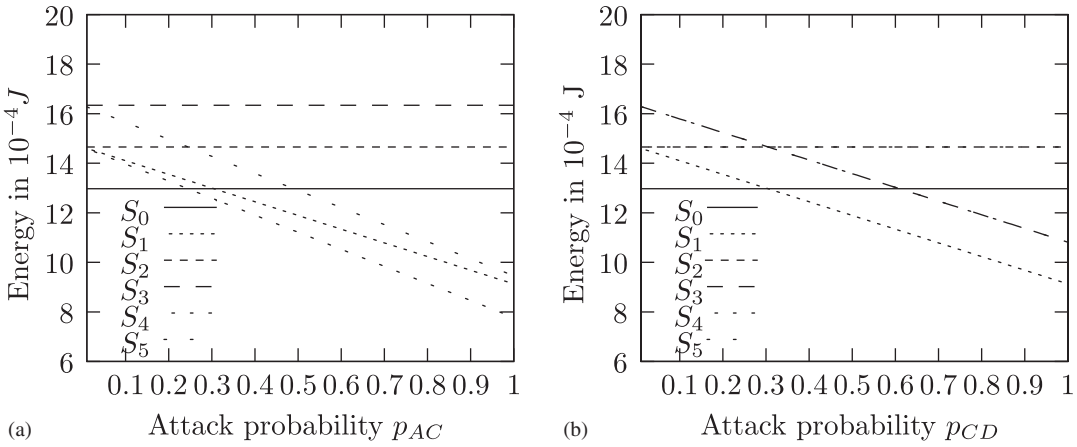

Figure 13. Energy consumption for a *single* attack localized on the link (A, C) (a) or (C, D) (b). On (b), the curves for S_2 and S_4 overlap and the ones for S_3 and S_5 overlap as well.

Results. We focus on three different scenarios of attacks:

- The attack targets a *single link*. Since the network is symmetrical, only the cases of an attack on links (A, C) and (C, D) are considered (results on Figure 13).
- The attack targets *two links*. Again, only the cases where attacks are on the pair of links $(A, C)/(B, C)$ and $(A, C)/(C, D)$ are considered for symmetry reasons. In both cases, we assume that the attacks on both links arise with the same probability, i.e. $p_{AC}=p_{BC}=p$ or $p_{AC}=p_{CD}=p$ (results in Figure 14).
- The attack targets the *complete forwarding network*. In this case, all links are attacked with the same probability, i.e. $p_{AC}=p_{BC}=p_{CD}=p$ (see Figure 15).

All subsequent figures show the values of $E(S_j)$ for each authentication strategy as a function of the probability of attack p.

In all figures, the strategies S_2 and S_3 are always the most expensive ones, whatever the value of the probability of attack in the network and the localization of the attacks. This is related to the fact that with AXF, node C is transmitting more often messages in AXF mode than with XAF since it authenticates each incoming packet individually and only forwards the packets that are not corrupted. Moreover, AXF performs $K-1$ times more checks than XAF if there are K incoming packets. AXF is the most costly strategy in energy, but it has be important benefit of providing the best packet throughput from the sources to the destinations and it permits to localize the attack on

DOI: 10.1002/cpe

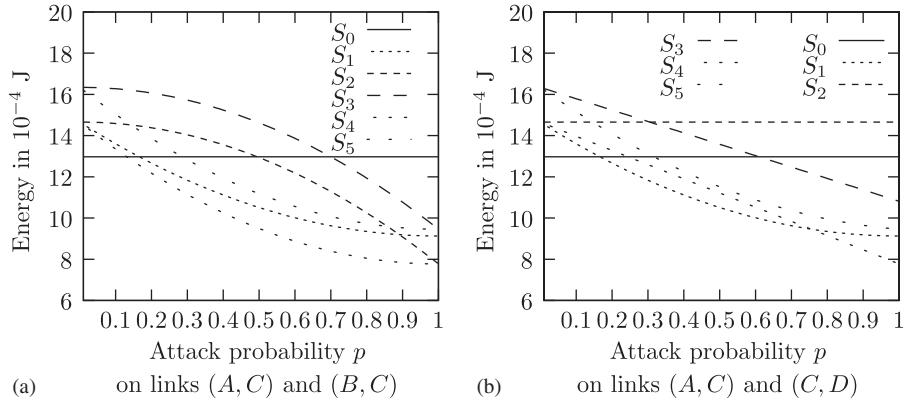

Figure 14. Energy consumption (in 10^{-4}) for an attack localized on the two pair of links $(A, C)/(B, C)$ (top) and $(A, C)/(C, D)$ (bottom).

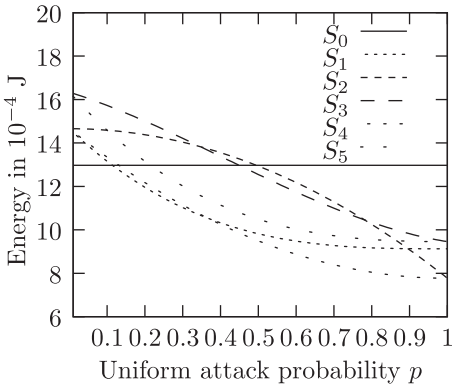

Figure 15. Energy consumption (in 10^{-4}) for a uniform probability of attack on all links p.

the incoming links. The average probability for E and F to recover the messages of A and B is given by

$$P = \tfrac{1}{2} f_c (1 - p_{CD})(2 - p_{AC} - p_{BC}), \tag{5}$$

where

$$f_c = \begin{cases} 1 - p_{AC}\, p_{BC} & \text{for AXF,} \\ (1 - p_{AC})(1 - p_{BC}) & \text{for XAF.} \end{cases}$$

Remark 7
From this expression, it is clear that the throughput of XAF is always below the throughput of AXF.

Strategy S_4, i.e. C authenticates with XAF, giving the best energy performance. Here, C transmits if all received packets are notcorrupted, which clearly reduces the throughput of data when an attack arises before C. The drawback of this approach is that it makes the network more vulnerable to DoS.

It is interesting to look at the energy performance for high and low probabilities of attack. For a low probability of attack, it is strategy S_1 that is the most energy efficient because there are only a few packets that are corrupted and the forwarding of these packets does not cost much to the network. Plus there is no energy wasted for unnecessary security checks. For a high attack probability, it is either S_4 or S_1 that consumes the minimum energy. In S_4, C drops the corrupted

DOI: 10.1002/cpe

packets and sometimes the uncorrupted packets that have been tainted by corrupted ones (Xor). This is the case for all scenarios of attacks but one, which is the case where the single link (C, D) is targeted. For this specific scenario, C has no impact on the overall network performance since the attack happens on its outgoing link. In this case, it is the strategy $\mathscr{S} = (a_C, a_D) = (0, 1)$ where D is authenticating which is more energy efficient than a basic forwarding.

The cross-points for curves S_0 and S_4 gives the probability of attack at which it becomes beneficial to start authenticating packets in XAF mode on the coding node C. For instance, for the case of a single link attack on (A, C), the cross-point is at $p_{AC} = 0.24$ and for the uniform case, it is at $p = 0.13$. The more the links are targeted, the sooner nodes should start to authenticate messages to minimize the global energy expenditure. For the scenario targeting the single link (C, D), the cross-point, where D should authenticate is $p_{CD} = 0.30$.

Figure 13 represents the case where there is only one localized attack either before the coding node C or between C and D. When p_{AC} is non-null, the attack is detected by C, which provides the lowest number of transmissions in the network. For the second case ($p_{CD} \neq 0$), security strategies where C authenticates are too expensive as already shown earlier. A better approach is to have D monitor the packet flow. Having both C and D monitors incurs an addition of at least 20% in energy with no additional security benefit (here we compare S_1 to S_5). From Figure 13, it is confirmed that it is important to detect the single link attack as early as possible. However, the naive solution where all nodes perform security checks and detect attacks at the earliest point in the network is not energy efficient. Indeed, all nodes that are located before the attacked link never perform successful checks and energy is wasted.

When two links are attacked in Figure 14, two situations are considered. In the first case, all the incoming links of C are attacked. The strategy where C monitors only using XAF is really energy efficient because it uses only one authentication to capture all corrupted packets. In the second case, a path ending in D is attacked, i.e. (A, C), (C, D) or (B, C), (C, D). Let us assume that the path $(A, C)(C, D)$ is attacked, it is still the strategy where C monitors with XAF which is the most efficient overall. However, it is interesting to note that in this case, up to a probability of attack of about one half, this strategy is as efficient as the one where only D authenticates and forwards. For a lower probability of attack, it is more interesting to have D detecting all threats than C since it captures the attacks on the whole path in one check. For higher probability, the gain in energy by node C stopping corrupted transmissions becomes more important.

Figure 15 provides the results for a uniformly distributed probability of attack on all links of the network. The curves on this plot are similar to the curves of Figure 14(a) while they are different from the ones of Figure 14(b). Hence, the scenario where the three links are attacked is similar to the one where the attacker targets two links before C. The common feature between the similar scenarios is that the two incoming links of the coding node can be attacked. Uniform probability of attack is really impacted by the actions of the coding nodes because they gather packets coming from different origins. For the case of uniform probability of attack, it is important to enable authentication on coding nodes and not on relay nodes from the energy point of view.

7. RELATED WORKS ON POLLUTION ATTACKS

Pollution attacks first appear in Peer-to-Peer (P2P) applications [31]: malicious users inject unusable or meaningless data in the system to frustrate the users. These attacks reappear with the new networking paradigm that are rateless codes [3], data aggregation or network coding [2]. All these applications share in common the fact that the nodes of the network transmit (linear) combinations of the data rather than the raw data. Therefore, protecting the network against pollution attacks is a challenging task: the nodes need to check that they are dealing with correct data despite the transformations. An overview of the challenges of pollution attacks in network coding applications can be found in [8, 9].

Homomorphic signatures [32–35] have been proposed to solve the problem of pollution attacks. These techniques are directly inspired by homomorphic encryption schemes. However, they require

DOI: 10.1002/cpe

to compute exponentiation which is far beyond the capacity of a node and despite the use of optimization technique such as batch computation [33].

Agrawal and Boneh have proposed in [16] to use homomorphic MACs (linear) to thwart pollution attacks in network coding applications. Their construction is based on an extension of the work of Krawczyk [23] as in this paper. Independently, Znaidi *et al.* in [17] apply the same extension of Krawczyk's work to the problem of data aggregation. From a security point of view, aggregation is very similar to network coding. The scheme of Agrawal and Boneh [16] has been since extended in [19]. However, all these works do not consider the threat of energy exhaustion implied by pollution attacks. We unveil all the possibilities offered by linear MACs to save as much energy as possible.

8. CONCLUSION AND OPEN PROBLEMS

We have studied the efficiency of MACs on sensor nodes in terms of delay and energy. From this study, we have capitalized to propose and study the natural modes of operation of MACs. We show that in an energy efficient scheme fighting pollution attacks: a good design consists of a combination of relays using AUTHENTICATE-XOR-FORWARD or XOR-FORWARD modes. An ongoing work consists of studying larger networks with more complex combinations of the data. An optimization process needs to be used to attribute which nodes use AUTHENTICATE-XOR-FORWARD, XOR-AUTHENTICATE-FORWARD or XOR-FORWARD (or their equivalent). DoS attacks against the XAF mode can be defeated by using group testing and an appropriate use of linearity.

Linear MACs perform well in our context because of the linearity of the considered transformations. However, some network-coding problems require to use non-linear methods [36]. For this class of problems, the only solution available against pollution attacks is AUTHENTICATE-XOR-MAC-FORWARD. Well-adapted authentication schemes are still to be discovered in this case.

An important part of the existing work on in-network transformations for WSNs is dedicated to data aggregation. If many solutions exist [17, 37–39], they essentially concern linear aggregations, e.g. the aggregation function is the mean. The recent advances in source coding or compressed sensing have introduced new non-linear aggregation functions that must also be studied. If the confidentiality of source coding has been studied, integrity protection for these schemes is still an open problem.

APPENDIX A: UNIVERSAL HASH FUNCTIONS

A universal hash function is a family of functions indexed by a parameter called the key and it must verify that the probability over all keys that all distinct inputs collide is small. This notion was introduced by Carter and Wegman in [40]. A relaxed notion called ε-almost universal was also introduced in [41]:

Definition A.1
Let f_k be a function of an (ℓ, n)-family H from an ℓ-bit set to an n-bit set with the parameter k taken in a set $\mathscr{K}$. The family H is ε-almost universal if the probability of collisions for a random distribution of the value k over the set $\mathscr{K}$ (i.e. $Pr_k(f_k(M) = f_k(M')), \forall k \in_R \mathscr{K}$) is smaller than ε.

Definition A.2
We also say that a family of functions H is $\oplus$-linear if for all M, M', we have $f_k(M \oplus M') = f_k(M) \oplus f_k(M')$ for all instance f_k in H.

A family H of functions that is at the same time ε-almost universal and $\oplus$-linear is said to be ε-almost XOR universal (ε-AXU). In this case, it must verify that the associated differential probability for a random distribution of the value k over the set $\mathscr{K}$ is bounded by ε, i.e. $\forall(M, M', a), Pr_k(f_k(M) - f_k(M') = a) \leq \varepsilon$.

DOI: 10.1002/cpe

The universal hash functions can be used for message authentication if the output is processed with another function. An MAC design using such a family of functions assumed the following scenario: the parties have already exchanged their secret key k, then to exchange a message M of length ℓ, the sender sends M and the corresponding *tag*:

$$h_k(M) = f_k(M) \oplus r. \tag{A1}$$

The shared secret key k is thus composed of a particular f_k function drawn randomly from an (ℓ, n)-family of hash functions and a random pad r. At reception, the receiver verifies the tag $h_k(M)$, corresponding with the MAC that is recomputed and checked for consistency. In practice, the fingerprint $f_k(M)$ is encrypted with a stream cipher which generates r.

Krawczyk has shown in [23] that the design of MAC of the above kind (i.e. combined with a one-time pad) requires only to have a family of functions that is ε-almost universal. Moreover, the family of functions can also be $\oplus$-linear. Thus, we can used for message authentication family of functions that are ε-almost XOR universal leading to homomorphic MAC constructions.

Many universal, ε-almost universal and ε-AXU hash families have been proposed in the literature to build MACs (see, for example, [18, 42, 43]). We mainly focus here on one proposal done by H. Krawczyk in 1994 in [23] that is really suitable for constraint environments.

A.1. Toeplitz hashing

In the same article, Krawczyk introduced a second construction based upon random matrices. More precisely, given A, a boolean Toeplitz matrix of size $n \times \ell$ (i.e. each lower diagonal is fixed, i.e. if $k - i = l - j$ for all indices then $A_{i,j} = A_{k,l}$) and given a message M of size ℓ, the universal hash function $h_A(M)$ is the binary multiplication of the matrix A by the column vector composed of message bits of M: $h_A(M) = A \cdot M$.

A simple and practical method to build such matrices is used for a linear feedback shift register (LFSR): from a particular irreducible polynomial of degree n over $\mathbb{F}_2 q(x)$, we build the corresponding LFSR of size n bits. The output sequence is denoted $s_0, s_1, \ldots$. The initial state of the LFSR $s = (s_0, s_1, \ldots, s_{n-1})$ represents a part of the secret key. More precisely, for each irreducible polynomial $q(x)$ and for each non-zero initial state of the LFSR, we associate the hash function $h_{q,s}(M)$ defined as the linear combination $\bigoplus_{j=0}^{\ell-1} M_j \cdot (s_j, s_{j+1}, \ldots, s_{j+n-1})$, where M_j is the bit number j of M. In other words, at each clock, the LFSR updates its internal state taking into account each message bit. This hash functions family is $\oplus$-linear, ε-almost universal (with $\varepsilon \leq (n+\ell)/(2^{n-1})$) and ε-almost XOR universal (with $\varepsilon \leq (\ell/(2^{n-1}))$).

ACKNOWLEDGEMENTS

This work is an extended version of NSS 2010.

REFERENCES

1. Pradhan SS, Ramchandran K. Distributed source coding using syndromes (DISCUS): Design and construction. *IEEE Transactions on Information Theory* 2003; **49**(3):626–643.
2. Ahlswede R, Cai N, Li SYR, Yeung RW. Network information flow. *IEEE Transactions on Information Theory* 2000; **46**(4):1204–1216.
3. Luby M. Lt codes. *Symposium on Foundations of Computer Science—FOCS 2002*. IEEE Computer Society: Silver Spring, MD, 2002; 271.
4. Katti S, Rahul H, Hu W, Katabi D, Médard M, Crowcroft J. XORs in the air: Practical wireless network coding. *IEEE/ACM Transactions on Networking* 2008; **16**(3):497–510.
5. Micali S, Ohta K, Reyzin L. Accountable-subgroup multisignatures: Extended abstract. *ACM Conference on Computer and Communications Security—CCS 2001*. ACM: Philadelphia, PA, U.S.A., 2001; 245–254.
6. Intanagonwiwat C, Govindan R, Estrin D. Directed diffusion: A scalable and robust communication paradigm for sensor networks. *ACM/IEEE International Conference on Mobile Computing and Networking—MobiCom'00*. ACM: Boston, MA, U.S.A., 2000; 56–67.
7. Perrig A, Stankovic JA, Wagner D. Security in wireless sensor networks. *Communication of the ACM* 2004; **47**(6):53–57.

DOI: 10.1002/cpe

8. Dong J, Curtmola R, Nita-Rotaru C. Practical defenses against pollution attacks in intra-flow network coding for wireless mesh networks. *ACM Conference on Wireless Network Security—WiSec '09*. ACM: New York, 2009; 111–122.
9. Dong J, Curtmola R, Nita-Rotaru C. Secure network coding for wireless mesh networks: Threats, challenges, and directions. *Computer Communication* 2009; **32**(17):1790–1801.
10. Krawczyk H, Bellare M, Canetti R. HMAC: Keyed-hashing for message authentication, 1997; RFC 2104.
11. Preneel B, van Oorschot PC. MDx-MAC and building fast MACs from hash functions. *Advances in Cryptology—CRYPTO '95* (*Lecture Notes in Computer Science*, vol. 963). Springer: Berlin, 1995; 1–14.
12. Black J, Rogaway P. CBC MACs for arbitrary-length messages: The three-key constructions. *Journal of Cryptology* 2005; **18**(2):111–131.
13. Nevelsteen W, Preneel B. Software performance of universal hash functions. *Advances in Cryptology—EUROCRYPT '99* (*Lecture Notes in Computer Science*, vol. 1592). Springer: Berlin, 1999; 24–41.
14. Black J, Halevi S, Krawczyk H, Krovetz T, Rogaway P. UMAC: Fast and secure message authentication. *Advances in Cryptology—CRYPTO '99* (*Lecture Notes in Computer Science*, vol. 1666). Springer: Berlin, 1999; 216–233.
15. McGrew DA, Viega J. The security and performance of the Galois/counter mode (GCM) of operation. *Progress in Cryptology—INDOCRYPT 2004* (*Lecture Notes in Computer Science*, vol. 3348). Springer: Berlin, 2004; 343–355.
16. Agrawal S, Boneh D. Homomorphic MACs: MAC-based integrity for network coding. *Applied Cryptography and Network Security—ACNS 2009* (*Lecture Notes in Computer Science*, vol. 5536). Springer: Berlin, 2009; 292–305.
17. Znaidi W, Lauradoux C, Minier M. Aggregated authentification (AMAC) using universal hash functions. *International ICST Conference on Security and Privacy in Communication Networks—SecureComm 2009*, Athens, Greece (*Lecture Notes of the Institute for Computer Sciences*, *Social Informatics and Telecommunications Engineering*, vol. 17). Springer, 2009; 248–264.
18. Sarkar P. A new universal hash function and other cryptographic algorithms suitable for resource constrained devices. *Cryptology ePrint Archive*, Report 2008/216, 2008. Available at: http://eprint.iacr.org/ (January 2011).
19. Yu Z, Wei Y, Ramkumar B, Guan Y. An efficient scheme for securing xor network coding against pollution attacks. *IEEE INFOCOM 2009*, Rio de Janeiro, Brazil, 2009; 406–414.
20. Fontaine C, Galand F. A survey of homomorphic encryption for non-specialists. *EURASIP Journal on Information Security* 2007; 10, Article ID 13801.
21. Castelluccia C, Chan ACF, Mykletun E, Tsudik G. Efficient and provably secure aggregation of encrypted data in wireless sensor networks. *ACM Transactions on Sensor Networks* 2009; **5**(3):1–36.
22. Xiao Y, Rayi VK, Sun B, Du X, Hu F, Galloway M. A survey of key management schemes in wireless sensor networks. *Computer Communications* 2007; **30**(11–12):2314–2341.
23. Krawczyk H. LFSR-based hashing and authentication. *Advances in Cryptology—CRYPTO '94* (*Lecture Notes in Computer Science*, vol. 839). Springer: Berlin, 1994; 129–139.
24. Zenner E. Nonce generators and the nonce reset problem. *International Conference on Information Security—ISC 2009* (*Lecture Notes in Computer Science*, vol. 5735). Springer: Pisa, Italy, 2009; 411–426.
25. Bellare M, Rogaway P. Optimal Asymmetric Encryption. *Advances in Cryptology—EUROCRYPT '94* (*Lecture Notes in Computer Science*, vol. 950). Springer: Perugia, Italy, 1994; 92–111.
26. Du DZ, Hwang FK. *Combinatorial Group Testing and its Applications* (*Applied Mathematics*). World Scientific Publishing Company: Singapore, 2000.
27. Zaverucha GM, Stinson DR. Group testing and batch verification. *International Conference on Information Theoretic Security—ICITS 2009* (*Lecture Notes in Computer Science*, vol. 5973). Springer: Shizuoka, Japan, 2009; 140–157.
28. Johnson R, Molnar D, Song DX, Wagner D. Homomorphic signature schemes. *Topics in Cryptology—CT-RSA 2002* (*Lecture Notes in Computer Science*, vol. 2271). Springer: San Jose, CA, U.S.A., 2002; 244–262.
29. Fraboulet A, Chelius G, Fleury E. Worldsens: Development and prototyping tools for application specific wireless sensors networks. *ACM/IEEE International Conference on Information Processing in Sensor Networks—IPSN 2007*. ACM: New York, 2007; 176–185.
30. Fournel N, Fraboulet A, Feautrier P. Embedded software energy characterization: Using non-intrusive measures for application source code annotation. *Journal of Embedded Computing* 2009; **3**(3):10.
31. Liang J, Kumar R, Xi Y, Ross KW. Pollution in p2p file sharing systems. *IEEE INFOCOM 2005*. IEEE: New York, 2005; 1174–1185.
32. Krohn MN, Freedman MJ, Mazières D. On-the-fly verification of rateless erasure codes for efficient content distribution. *IEEE Symposium on Security and Privacy—S&P 2004*. IEEE Computer Society: Silver Spring, MD, 2004; 226–240.
33. Gkantsidis C, Rodriguez P. Cooperative security for network coding file distribution. *IEEE INFOCOM 2006*. IEEE: New York, 2006.
34. Charles D, Jain K, Lauter K. Signatures for network coding. *International Journal in Information and Coding Theory* 2009; **1**(1):3–14.
35. Boneh D, Freeman D, Katz J, Waters B. Signing a linear subspace: Signature schemes for network coding. *Public Key Cryptography—PKC 2009* (*Lecture Notes in Computer Science*, vol. 5443). Springer: Berlin, 2009; 68–87.
36. Riis S. Linear versus non-linear boolean functions in network flow. *Conference on Information Sciences and System—CISS 2004*, Princeton, NJ, U.S.A., 2004.

DOI: 10.1002/cpe

37. Przydatek B, Song D, Perrig A. SIA: Secure information aggregation in sensor networks. *ACM International Conference on Embedded Networked Sensor Systems—SenSys 2003*, Los Angeles, CA, U.S.A., 2003.
38. Hu L, Evans D. Secure aggregation for wireless networks. *Workshop on Security and Assurance in Ad hoc Networks*, Orlando, FL, U.S.A., 2003; 384–394.
39. Chan H, Perrig A, Song D. Secure hierarchical in-network aggregation in sensor networks. *ACM Conference on Computer and Communications Security—CCS '06*. ACM: New York, 2006; 278–287.
40. Carter L, Wegman MN. Universal classes of hash functions (Extended Abstract). *ACM Symposium on Theory of Computing—STOC '77*. ACM: New York, 1977; 106–112.
41. Carter L, Wegman MN. Universal classes of hash functions. *Journal of Computer and System Sciences—JCSS* 1979; **18**(2):143–154.
42. Shoup V. On fast and provably secure message authentication based on universal hashing. *Advances in Cryptology—CRYPTO '96* (*Lecture Notes in Computer Science*, vol. 1109). Springer: Berlin, 1996; 313–328.
43. Handschuh H, Preneel B. Key-recovery attacks on universal hash function based MAC algorithms. *Advances in Cryptology—CRYPTO 2008* (*Lecture Notes in Computer Science*, vol. 5157). Springer: Berlin, 2008; 144–161.

DOI: 10.1002/cpe