Secure and Scalable Many-to-One Symbol Transmission for Sensor Networks

Alexandre Viejo, Francesc Sebé and Josep Domingo-Ferrer

Rovira i Virgili University
Department of Computer Engineering and Mathematics
UNESCO Chair in Data Privacy
Av. Països Catalans, 26, E-43007 Tarragona, Catalonia
E-mail: {alexandre.viejo,francesc.sebe,josep.domingo}@urv.cat

Abstract

In secure many-to-one communications, a set of nodes transmit data to the same receiver. Research in this area focuses on increasing scalability by minimizing the amount of data reaching the receiver. Current proposals for secure many-to-one transmission use costly cryptographic operations. In this paper we present a novel system for secure many-to-one symbol transmission in which nodes are only required to perform lightweight operations. This makes it suitable for implementation in resource-constrained scenarios such as sensor networks.

Key words: Many-to-one communications, secure communications, sensor networks

1 Introduction

Communications can be classified according to the number of involved senders and receivers. Single-sender paradigms are: *one-to-one* (unicast) in which a single sender transmits data to a single receiver; *one-to-all* (broadcast) in which one source sends data to all nodes of a network; and *one-to-many* (multicast) where a single source transmits to a given subset of nodes.

Efficient one-to-many (and one-to-all) communications are implemented using a tree communication model. The root of the tree is the source which sends the data, the intermediate nodes are the routers which receive the content from their parent node and retransmit it to their child nodes (replicating it for each child), and the leaves are the receivers. This model provides scalability because the number of receivers can be increased without increasing the workload nor the bandwidth needs at the source.

In this paper we focus on *many-to-one* communications. In this paradigm, a set of nodes transmit data to the same receiver. If the number of transmitting nodes is large, the receiver may be overwhelmed by the incoming traffic. This problem is known as implosion [1].

Implosion resistance is a challenging issue in the design of many-to-one communication protocols. Such protocols also follow a tree topology. In this case, the leaves are the senders; the intermediate nodes are routers that collect messages coming from their children and aggregate them into a single message that is transmitted up to their parent; finally, the root is the receiver. Scalability depends on the aggregation operation performed by intermediate routers.

In addition to their being scalable, many-to-one communications often need to be secure. Security requirements include *confidentiality* (an intruder should not be able to learn the transmitted data), *integrity* (any data alteration should be detectable by the receiver) and *authentication* (the source of the data should be verifiable by the receiver).

There is a general consensus that in scenarios where nodes are resource-constrained devices the high cost of public-key cryptography is usually not affordable. Researchers assume that in such scenarios symmetric cryptography and hash functions constitute the tools of choice to provide security. However, public-key technology can be selectively deployed in those environments too. In [2] the author argues that the RSA [3] public-key cryptosystem with a small public exponent and Rabin's [4] public-key cryptosystem have fast algorithms for encryption and digital signature verification which can be used on constrained devices. In contrast, their decryption and signature generation are slow and resource-demanding. Elliptic curve cryptosystems (ECC, [5]) provide not only lightweight encryption and signature verification, but also lightweight decryption and signature generation which make them suitable for resource-constrained devices.

Sensor networks are an example where secure many-to-one communications are required in low-cost and resource-constrained devices. Here, the sensor nodes (which may be very numerous) transmit data to a single collecting center. Security requirements arise when the networks are deployed in hostile areas.

1.1 Previous work

As stated in [6], the solution to implosion in many-to-one scenarios is obtained by intermediate routers combining received messages into a single message that is routed towards the base station. This process is called aggregation.

The authors in [6] present a general framework for scalable many-to-one communication where intermediate nodes collect messages from their children, aggregate them and send a single aggregated message up to their parent. In this way, the base station receives a single message containing all the readings from the leaves. This solution is scalable (permitting an unlimited amount of senders) as long as aggregated data do not grow in size. Requiring the output of the aggregation to be of constant size implies that some information loss must be tolerated.

Next, we give some examples of lossy data aggregation:

- If data is a temperature, different temperatures can be aggregated by computing their average. Information loss comes from the fact that the base station will not learn the temperature obtained by each node but only the average of all readings.
- If data is a counter, different counters can be aggregated by addition. Information loss comes from the base station not being able to learn the exact contribution by each node.
- If data sent is a binary value indicating an alarm, it can be aggregated using a logical OR operation. The base station will learn that an alarm has been raised somewhere but not exactly where.

The above lossy approaches do not impose strong computational requirements so they are suitable for a wide variety of scenarios including resource-constrained sensor networks. However, they cannot be used in scenarios where the root must know the specific data sent by each leaf: lossless communication is needed here.

There are some schemes in the literature which provide security in many-to-one lossless transmission. Such proposals can be divided into two categories which are reviewed below.

1.1.1 Secure acknowledgment

Secure acknowledgment schemes provide the source of a multicast communication with an undeniable and unforgeable proof that a given group of receivers have properly received a specific message. In these schemes, upon receiving the message, every receiver will answer with an acknowledgment (consisting of a digital signature) in case of correct reception or will not answer anything otherwise. These acknowledgments can be aggregated to achieve scalability. Such solutions only permit unary communications (a receiver can send an acknowledgment or stay silent) so that they are not appropriate for data transmission. The systems proposed in [7] and [8] fall into this category.

1.1.2 Secure symbol transmission

This class of schemes assume a tree communication model where the root multicasts a data request to the leaves. Upon reception of this request, the leaves react by sending one q-ary symbol each (data sent by each leaf can be modeled as an integer ranging from 1 to q). These messages will be aggregated by intermediate nodes. From the received message, the root will obtain the symbol sent by each leaf.

It can be proven that symbols sent by n leaves cannot be aggregated in a message whose length is less than $O(n)$ when all symbols have the same probability of being sent. Current research in secure symbol transmission focuses on designing systems whose actual message length is as short as possible (within the $O(n)$ length class). Note that due to this fact the amount of senders cannot be arbitrarily large.

The schemes proposed in [9] and [10] belong to this category. For q-ary alphabets, the former provides a message length that asymptotically tends to $3tn$ (where $q \leq 2^t - 1$) while the latter provides a message length asymptotically approaching tn (both message lengths are $O(n)$).

In [11], a proposal for biased binary ($q = 2$) communications is described. It offers an $O(k \log k \log n)$ message length with n being the number of leaves and k being an upper bound on the number of leaves that simultaneously wish to transmit the least likely bit.

The three schemes [9–11] provide a good bandwidth efficiency. Unfortunately, they require resource-demanding cryptographic operations at the nodes. Proposals [9] and [11] use homomorphic public-key cryptosystems which cannot be used in resource-constrained nodes. Regarding the scheme presented in [10], it requires an intensive use of multisignature generation and verification over Gap Diffie-Hellman groups, which makes it unaffordable in resource-constrained scenarios.

1.2 Contribution and plan of this paper

In this paper we propose a secure and scalable scheme designed to provide secure many-to-one symbol transmission in networks formed by low-cost devices (such as sensor networks). In our proposal, nodes only compute hash functions, simple arithmetic and logic operations, and a signature verification (which, as stated above, is affordable with elliptic curve cryptosystems [5], the Rabin cryptosystem [4] or RSA [3] with a small public exponent).

Our protocol provides transmission of q-ary alphabets with message length asymptotically tending to tn, where n is the number of leaves of the multicast tree and $q \leq 2^t - 1$.

Section 2 describes the new protocol. In Section 3 we analyze its security. Section 4 provides a performance analysis and an optimization of the proposed protocol to reduce the energy consumption at nodes due to communication. Finally, Section 5 is a conclusion.

2 Our proposal

Our protocol assumes a tree network where the root is the base station receiving data from the leaves (which may be sensor nodes). For the sake of simplicity, we assume that only the leaves send data. Intermediate nodes simply act as routers. Extending the proposed solution to accommodate data transmission from intermediate nodes is straightforward.

The base station (BS) is a full-fledged computer. Leaves and intermediate nodes are low-cost devices. The base station owns a private key SK_{BS}. The corresponding public key PK_{BS} is known and accepted as valid by all nodes in the tree. Let n be the number of leaves and U_i, $1 \leq i \leq n$, denote the leaves. Each leaf U_i shares a secret key K_i with the base station.

2.1 Many-to-one q-ary transmission

We represent each symbol from the q-ary alphabet by a different integer from the set $\{1, \ldots, q\}$. Parameter t is chosen as the smallest integer satisfying $q \leq 2^t - 1$. Parameter s is a security parameter (see Sections 3 and 4 for details about s). A protocol execution consists of the following steps:

(1) CHALLENGE. The base station generates a random value v and signs it to obtain $\{v\}_{SK_{BS}}$. The signed value is multicast by the base station to all leaves.
(2) MESSAGE GENERATION.
    (a) Upon receiving v and verifying its signature, each leaf U_i computes a pseudo-random t-bit sequence $(c_1, \ldots, c_t) \leftarrow lsb_t(\mathcal{H}(v||K_i))$, where $lsb_t(\cdot)$ is a function returning the t least significant bits of its argument, $\mathcal{H}$ is a one-way hash function and $||$ is the concatenation operator.
    (b) Each U_i computes a sequence $(d_1, \ldots, d_t)$ as follows. Let $(b_1, \ldots, b_t)$ be the binary representation of the q-ary symbol to be transmitted by U_i.

- If $(b_1, \ldots, b_t) = (c_1, \ldots, c_t)$ then $(d_1, \ldots, d_t) := (b_1, \ldots, b_t)$
  Else $(d_1, \ldots, d_t) := (b_1 \oplus c_1, \ldots, b_t \oplus c_t)$

Note that this step ensures that the sequence $(d_1, \ldots, d_t)$ does not have all its elements equal to 0. This all zeroes value is reserved to identify non-transmittal by leaves.

(c) U_i computes an s-bit pseudo-random integer σ_i as follows:

$$\sigma_i \leftarrow lsb_s(\mathcal{H}(d_1, \ldots, d_t || v || K_i))$$

(d) Each U_i generates a tn-bit sequence (n is the number of leaves) I_i and sets the bits from the subsequence between positions $t(i-1)+1$ and ti so that they match $(d_1, \ldots, d_t)$. The remaining bits are set to "0".

(e) U_i sends the pair (I_i, σ_i) up to its parent node.

(3) Message aggregation. An intermediate node R (or the base station) receives messages from its child routers/leaves and does the following:

(a) Store each received pair (I_j, σ_j) (they may have to be checked later).

(b) Once all expected messages $\{(I_j, \sigma_j)\}_j$ have been received, aggregate them by computing $I = \bigvee_j I_j$ ($\vee$ denotes the bitwise OR operation) and $\sigma = \sum_j \sigma_j \pmod{2^s}$.

(c) If R is not the base station, send (I, σ) up to its parent node. Else, this is the final aggregated message.

(4) Symbol extraction. From the final aggregated message (I, σ), the base station obtains, for each leaf U_i, the binary representation $(b_{i,1}, \ldots, b_{i,t})$ of the symbol sent by the leaf. It is obtained from the sequence $(d_{i,1}, \ldots, d_{i,t})$, previously generated by U_i (see Step 2b), which is contained in I. Then the base station computes the pseudo-random integer linked to $(d_{i,1}, \ldots, d_{i,t})$ (see Step 2c), which will be used to check the integrity of the whole aggregated message. We next give the pseudo-code related to this process:

```
(a) Let $i := 1$, $\omega := 0$.
(b) While $i \leq n$ loop
      • Compute $(c_1, \ldots, c_t) \leftarrow lsb_t(\mathcal{H}(v || K_i))$.
      • If $(I[t(i-1)+1], \ldots, I[ti]) = (c_1, \ldots, c_t)$ then $(b_{i,1}, \ldots, b_{i,t}) := (c_1, \ldots, c_t)$.
      • Else $(b_{i,1}, \ldots, b_{i,t}) := (I[t(i-1)+1] \oplus c_1, \ldots, I[ti] \oplus c_t)$.
      • Compute $\phi_i \leftarrow lsb_s(\mathcal{H}(I[t(i-1)+1], \ldots, I[ti] || v || K_i))$.
      • $\omega := \omega + \phi_i \pmod{2^s}$.
      • $i := i + 1$.
```

(c) If $\omega = \sigma$ then return $B = ((b_{1,1}, \ldots, b_{1,t}), \ldots, (b_{n,1}, \ldots, b_{n,t}))$, where $(b_{i,1}, \ldots, b_{i,t})$ is the binary representation of the symbol transmitted by U_i. The base station also multicasts a signed acknowledgment $\{\text{"}Ack\text{"} || v\}_{SK_{BS}}$ to the leaves. This message contains the challenge v to avoid replay attacks. Upon receiving this message, intermediate routers remove messages stored at Step 3a.

If $\omega \neq \sigma$ the base station launches the error-tracing procedure detailed in Section 2.2.

Figure 1 shows the message flow generated by a protocol execution in a simple scenario with a base station (BS), two intermediate nodes (R_1 and R_2) and four leaves ($U_1, \ldots, U_4$). In Figure 1.a the base station broadcasts a challenge to all leaves (Step 1 in the protocol execution). In Figure 1.b, message (1) sent by U_1 corresponds to the pair (I_1, σ_1) while message (2) sent by U_2 represents the pair (I_2, σ_2) (Step 2 in the protocol execution). Node R_1 constructs message (3), which corresponds to (I, σ), by aggregating messages (1) and (2) (Step 3 in the protocol execution). The same process occurs in the subtree rooted by R_2. The latter node constructs message (6) by aggregating messages (4) and (5), which correspond to the pairs (I_3, σ_3) and (I_4, σ_4), respectively. Eventually, the base station BS aggregates messages (3) and (6) to get the final aggregated message. After that, BS extracts the symbols transmitted by the leaves (Step 4 in the protocol execution).

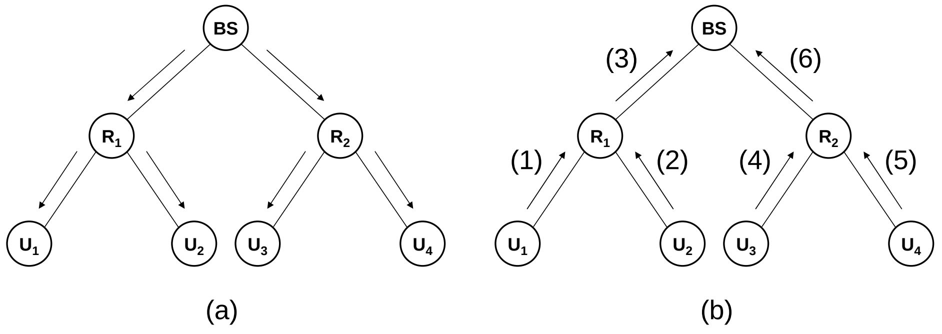

Fig. 1. Message flow in a protocol execution

2.2 Procedure to deal with corrupted messages

During symbol extraction, the base station checks the integrity of the received message. If this verification fails, the base station identifies the message as corrupted. The following procedure allows to remove the corrupting nodes from the tree.

(1) From the received I component, the base station computes the valid σ_i associated to each U_i:
  (a) For $i = 1$ to n do
    - $(d_{i,1}, \ldots, d_{i,t}) := (I[t(i-1)+1], \ldots, I[ti])$
    - $\sigma_i \leftarrow lsb_s(\mathcal{H}(d_{i,1}, \ldots, d_{i,t}||v||K_i))$
  (b) The base station sends to all nodes the signed message

$$\{I||\sigma_1, \ldots, \sigma_n||v\}_{SK_{BS}}$$

(2) Upon receiving $\{I||\sigma_1, \ldots, \sigma_n||v\}_{SK_{BS}}$ and verifying its signature and the

value v, each intermediate node R checks each stored message (I_j, σ_j) received from its children. For each (I_j, σ_j), R does:

(a) For each leaf U_i with nonzero contribution to I_j (that is, $(I[t(i-1)+1], \ldots, I[ti]) \neq (0, \ldots, 0)$), check that the contribution to I_j equals the contribution to I. If some of these checks fail, the point of corruption is above R's position and R stops the checking procedure.

(b) Else, R computes the sum modulo 2^s of the σ_i associated to each U_i who contributes to I_j. If the sum is equal to σ_j, the child who sent this message is considered innocent. If the result is different, that child is considered guilty.

When a malicious node has altered a message (note that a malicious node can corrupt a message at each protocol execution or only once in a while), all nodes located in the path from the corruption point to the root detect this corruption. However, the nodes detecting the corruption cannot decide whether corruption was caused by the child who sent the corrupted message or by another node located in the subtree rooted at this child. A simple solution would be to delete the entire suspicious subtree. This would entail the loss of a big part of the network due to a single corrupted message. We propose the following procedure to minimize the number of nodes to be eliminated:

(1) The base station and all intermediate nodes have a pre-loaded integer λ associated to each child.
  - Initially, an intermediate node assigns a value $\lambda = 1$ to those of its children that are leaves.
  - If a child is an internal node, the initial assigned value corresponds to the number of leaves in the subtree rooted at that child.

(2) When a node detects a corrupted message, it decrements the λ value assigned to the child from which this message comes from. Note that the λ values of all nodes located between the point of corruption and the base station will be decremented.

(3) When the λ value assigned to a child becomes zero, the node closes communication with such a suspicious child (thus pruning the subtree rooted at this child).

The initial value assignment ensures that a corrupted leaf is pruned the first time it sends a bad message ($\lambda = 1$). The initial value assignment for an internal node is done assuming that it always acts honestly so that it only needs to be removed from the tree after all the leaves in its subtree have already been removed (in this case no more messages will come from its subtree). This procedure ensures that a dishonest internal node will be removed, although not necessarily the first time it corrupts and forwards a message.

3 Security analysis

We next explain the adversarial model and the possible attacks the system has to be robust against. We refer to those attacks to justify the security properties achieved by our scheme: *confidentiality*, *authentication* and *integrity*. We also give the success probability of each possible attack.

3.1 Adversarial model

Our attacker model considers an adversary who can control nodes (thus turning them into compromised nodes) and who can also access communication lines to capture, modify and retransmit messages. The attacker's computational power does not permit her to break current computationally secure cryptosystems.

3.1.1 Possible attacks

An adversary can try the following attacks:

- Eavesdrop messages.
- Impersonate a certain leaf U_i.
- Alter the contribution of a certain leaf U_i.
- Remove the contribution of a certain leaf U_i.
- Disrupt the aggregation of messages received from some child nodes.

More specifically, an external attacker can try the first four attacks while an adversary which has compromised some nodes could also try the last one.

3.2 Attacks and security properties

Table 1 summarizes the success probabilities of each possible attack in scenarios where non-transmittal is disallowed. The subsections below justify the values in the table.

3.2.1 Message eavesdropping

This attack refers to the *confidentiality* property. An adversary eavesdropping messages of the form (I, σ) at some point (called sniffing point from now on) in the communication tree can discover, by observing the bits ranging from

Table 1
Possible attacks and their success probabilities

Attack	Success probability
Message eavesdropping	Not possible
Leaf impersonation	$1/((2^t - 1)2^s)$
Leaf contribution alteration	$1/2^s$
Leaf contribution removal	Not possible
Message aggregation disruption	Not possible

($I[t(i-1)+1]$ to $I[ti]$, the sequence $(d_{i,1}, \ldots, d_{i,t})$ transmitted by each leaf U_i located below the sniffing point. The attacker will be unable to decrypt this sequence because decryption requires knowledge of the secret key K_i.

The σ value does not provide any useful information to the attacker either.

From I, an adversary can determine which leaves transmitted and which ones did not. In applications where this fact causes information disclosure, non-transmittal should not be allowed.

3.2.2 Leaf impersonation

This attack refers to the *authentication* property. An intruder (who does not know K_i) trying to send a given symbol coming from U_i faces several difficulties.

First of all, the q-ary symbol is encrypted prior to encoding it inside I_i. Since the attacker does not know the encryption key, she can only fill the corresponding t bits in I_i randomly. The probability that decryption of those bits leads to the desired q-ary symbol is $1/(2^t - 1)$.

On the other side, the redundancy σ_i is also computed using K_i. In this way, the probability of randomly guessing the appropriate σ_i is $1/2^s$.

3.2.3 Leaf contribution alteration

This attack refers to the *authentication* and *integrity* properties. In case the attacker captures and alters the contribution of a leaf, the difficulty comes from the low probability of guessing σ, which is $1/2^s$. A sufficiently large value s exponentially reduces the chances of a corrupting attacker to stay undetected.

Table 2
Performance results

Concept	Cost
Message length (for $n \uparrow\uparrow$)	tn
Message length in the error-tracing procedure (for $n \uparrow\uparrow$)	$(t+s)n$
Cost of message generation	$O(n)$
Cost of message aggregation	$O(n^2)$
Cost of symbol extraction	$O(n)$
Cost of error tracing at intermediate nodes	$O(n^2)$

3.2.4 Leaf contribution removal

The contribution of U_i can be easily erased if σ_i is known. This fact will be considered as a non-transmittal by the base station. To detect these erasure attacks, non-transmittal should be disallowed.

In case the adversary does not know the σ_i value generated by U_i, she must guess the appropriate σ_i with a probability of success $1/2^s$.

3.2.5 Disruption of the aggregation of child node messages

A dishonest intermediate node can decide not to aggregate a message received from some of its child nodes. The base station will consider this fact as a non-transmittal. To detect this situation non-transmittal should be disallowed.

4 Performance analysis

We next evaluate the protocol performance in terms of message length and computational cost at the sensor nodes. We also give an optimization of the proposed protocol to improve the efficiency of the message length and reduce the energy consumption at nodes due to communication.

Table 2 summarizes the performance results obtained. The subsections below justify the values in that table.

4.1 Message length

Our protocol keeps the length of messages constant on their way from the leaves towards the base station. Each message consists of the pair (I, σ). Component I encodes the q-ary symbol transmitted by each leaf and its bitlength is tn, where n is the number of leaves of the multicast tree and $q \leq 2^t - 1$. Since t is a constant, the bitlength of I is $O(n)$. Component σ has a constant length of s bits. So its length is $O(1)$. In this way, messages have a bitlength $O(n)+O(1)$. For large values of n, this length asymptotically tends to tn. Note that this length is linear, which represents a limitation on the total amount of leaves which can participate in the network. However, it can be proven that symbols sent by n leaves cannot be aggregated in a message whose length is below $O(n)$ when all symbols have the same probability of being sent.

In the event of a corrupted message, the base station multicasts a message which contains $(I||\sigma_1, \ldots, \sigma_n||v)$. The first component has tn bits, the second one sn bits and the last one $O(1)$ bits. Thus, this special message has a bitlength of $O(n)$. For large values of n, this length asymptotically tends to $(t + s)n$.

4.2 Computational cost

Next, we analyze the time complexity of the protocol in four operations: message generation, message aggregation, symbol extraction and error tracing at intermediate nodes.

Message generation: Each message has two components (I_i, σ_i). Leaf U_i employs $O(n)$ time to generate the binary sequence I_i. The computation of σ_i does not depend on n, so it is $O(1)$. During message generation, each leaf verifies one signature and computes two hash functions and at most t bitwise XOR operations.

As mentioned in Section 1, there are fast algorithms for digital signature verification in resource-constrained environments. As a real example, the authors of [5] implement the Elliptic Curve Digital Signature Algorithm [12,13] on a MICA2 mote [14,15], designed by researchers at the University of California at Berkeley. This device offers an 8-bit, 7.3828-MHz ATmega 128L processor, 4 kilobytes (KB) of primary memory (SRAM), and 128 KB of program space (ROM). According to their results, an ECDSA signature verification in such a device takes about 24.17 seconds (this time can only be expected to decrease as technology progresses).

The remaining operations (hash functions and bitwise operations) take negligible time. Thus, our scheme is suitable for resource-constrained leaves.

Message aggregation: An intermediate node receives and aggregates messages. Aggregation of $\{(I_j, \sigma_j)\}_j$ into (I, σ) requires at most $O(n)$ bitwise OR operations over $O(n)$-long messages during the computation of I. Computation of the new σ requires at most $O(n)$ additions modulo 2^s (each with cost $O(1)$). This results in a maximum $O(n^2)$ cost. Note that intermediate nodes compute only bitwise operations. These operations are appropriate for resource-constrained nodes.

Symbol extraction: The base station extracts the contribution of each leaf in a vector $B = ((b_{1,1}, \ldots, b_{1,t}), \ldots, (b_{n,1}, \ldots, b_{n,t}))$ by processing component I (total length tn bits). Since t is a constant, this time is $O(n)$. Integrity checking does not increase this cost.

Since the base station is a full-fledged device, we consider that all operations executed in this step are affordable.

Error tracing at intermediate nodes: This procedure is only invoked in case of a corrupted message event. An intermediate node may need to verify a signature and check the value I_j sent by each of its children. Each check takes $O(n)$ time. Since the number of children may also be $O(n)$, the maximum time spent on this operation is $O(n^2)$. Checking σ_j has at most the same cost.

In this step we require one signature verification. As explained above, this operation is affordable on real sensor nodes like the MICA2 mote. In addition to that, intermediate nodes must execute $O(n^2)$ additions modulo 2^s. These operations are suitable for resource-constrained nodes too.

4.3 Message length optimization

Our system is designed for nodes that are resource and power-constrained devices. This motivates the need to reduce energy consumption as much as possible. Reducing the length of messages is one way to achieve this.

In our protocol, leaf U_i sends (I_i, σ_i) where I_i is a tn bit long binary sequence. Useful information within I_i is contained in bits located between positions $t(i-1)+1$ and ti. The remaining bits of I_i are set to 0.

This information could be represented in a more compact way using $\log n$ bits to code index i and t bits for useful information. In this way, the length of I_i would be $t + \log n$. Aggregation of vectors I_i would be done by concatenation. In this way, the length of a vector I containing data from j leaves would be $j(t + \log n)$ bits.

For small values of j this results in shorter messages than those described in our protocol above (*i.e.* when $j(t + \log n) < tn$). Low values of j appear at nodes that are far from the root. However, when j grows towards n this new coding results in longer messages than those described above.

Therefore using this alternative coding when j satisfies $j(t + \log n) < tn$ (near the leaves) and switching to the initial coding when messages get near the root is a way to minimize the length of transmitted data.

5 Conclusion

A scalable and lightweight protocol for secure many-to-one symbol transmission in tree-based networks has been presented. This proposal is the first one in the literature that provides this kind of communication for resource-constrained devices, which are quite common in sensor networks. Our scheme provides constant message length and $O(n)$ cost for symbol extraction at the base station, where n is the number of senders (leaves of the tree). Besides, it achieves the fundamental security properties: confidentiality, authentication and integrity.

Regarding computation, this protocol has been designed to be very lightweight at leaves and intermediate nodes:

(1) Leaves are only required to compute hash functions, bitwise operations and two signature verifications (the first one at the beginning and the last one at the end of each protocol execution).
(2) In a correct protocol execution, intermediate nodes are only required to aggregate messages which represent bitwise OR operations and additions modulo 2^s. In case of message corruption (whether intentional or accidental), intermediate nodes are required to verify one signature and check the messages received from their children. Message corruption is thwarted by removing corrupting nodes from the network.

Disclaimer and acknowledgments

The authors are with the UNESCO Chair in Data Privacy, but they are solely responsible for the views expressed in this paper, which do not necessarily reflect the position of UNESCO nor commit that organization. This work was partly supported by the Spanish Ministry of Education through projects TSI2007-65406-C03-01 "E-AEGIS" and CONSOLIDER CSD2007-00004 "ARES", and by the Government of Catalonia under grant 2005 SGR 00446.

References

[1] B. Quinn and K. Almeroth, "IP multicast applications: challenges and solutions", *Internet RFC 3170*, 2001. `http://www.ietf.org`

[2] Z. Benenson, "Authenticated Queries in Sensor Networks", *Lecture Notes in Computer Science*, vol. 3813, 2005, pp. 54-67.

[3] R. L. Rivest, A. Shamir and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems", *Communications of the ACM*, vol. 21, 1978, pp. 120-126.

[4] M. Rabin, "Digitalized signatures and public-key functions as intractable as factorization", MIT Laboratory for Computer Science TR-212, Jan. 1979.

[5] E.-O. Blaß and M. Zitterbart, "Towards Acceptable Public-Key Encryption in Sensor Networks", *ACM 2nd International Workshop on Ubiquitous Computing*, 2005, pp. 88-93.

[6] T. Wolf and S. Y. Choi, "Aggregated hierarchical multicast - a many-to-many communication paradigm using programmable networks", *IEEE Transactions on Systems, Man and Cybernetics - Part C*, vol. 33, no. 3, 2003, pp. 358-369.

[7] A. Nicolosi and D. Mazières, "Secure acknowledgment of multicast messages in open peer-to-peer networks", *3rd International Workshop on Peer-to-Peer Systems - IPTPS'04*, San Diego, CA, 2004.

[8] C. Castelluccia, S. Jarecki, J. Kim and G. Tsudik, "Secure acknowledgment aggregation and multisignatures with limited robustness", *Computer Networks*, vol. 50, no. 10, 2006, pp. 1639-1652.

[9] J. Domingo-Ferrer, A. Martínez-Ballesté and F. Sebé, "Secure reverse communications in a multicast tree", *Lecture Notes in Computer Science*, vol. 3042, 2004, pp. 807-816.

[10] F. Sebé, A. Viejo and J. Domingo-Ferrer, "Secure Many-to-One Symbol transmission for Implementation on Smart Cards", *Computer Networks*, vol. 51, no. 9, 2007, pp. 2299-2307.

[11] F. Sebé and J. Domingo-Ferrer, "Scalability and security in biased many-to-one communication", *Computer Networks*, vol. 51, no. 1, 2007, pp. 1-13.

[12] ANSI X9.62-1998, "Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)", American National Standard for Financial Services, American Bankers Association, 1999.

[13] D.J. Johnson, A.J. Menezes, S.A. Vanstone, "The Elliptic Curve Digital Signature Algorithm (ECDSA)", *International Journal of Information Security*, vol. 1, 2001, pp. 36-63.

[14] University of California Berkeley, "Tiny OS Hardware Designs", 2004. `http://www.tinyos.net/scoop/special/hardware`

[15] Crossbow Technology, Inc., "MICA2 Data Sheet", 2008. `http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICA2_Datasheet.pdf`