Available online at www.sciencedirect.com

ScienceDirect

Computer Networks 51 (2007) 2299–2307

Computer Networks

www.elsevier.com/locate/comnet

Secure many-to-one symbol transmission for implementation on smart cards

Francesc Sebé [*], Alexandre Viejo, Josep Domingo-Ferrer

Rovira i Virgili University of Tarragona, Dept. of Computer Engineering and Maths, Av. Països Catalans 26, E-43007 Tarragona, Catalonia, Spain

Available online 31 January 2007

Abstract

Multicast communication is arguably the most promising paradigm to enable mass Internet transmission of live events, or any other content sent on the network from a single source to a large community of receivers. A tree model is normally used to represent this type of communication. Scalability problems arise when communication is many-to-one (leaf-to-root) rather than one-to-many (root-to-leaf): this is the case when the root collects data (sensor information, usage data, micropayments in real-time pay-per-view, etc.) from the leaves. The matter is further complicated if there are confidentiality and authenticity requirements on the leaf-to-root traffic. We present here a new method offering security and scalability in many-to-one communication which outperforms previous proposals in the literature: it is more general, it saves more bandwidth and it is computationally simpler. In particular, computation at the leaves is simple enough to be encapsulated in a smart card, which is attractive to protect the key material held by the leaves.

Keywords: Multicast; Many-to-one communication; Scalability; Security; Smart cards

1. Introduction

Multicast is a communication paradigm where a single sender transmits the same data to a group of receivers (one-to-many communication). This is the most cost-effective transmission model for large scale real-time multimedia transmission (for instance, radio and TV). Reverse multicast is a paradigm where several sources send data to a single receiver (many-to-one transmission).

Multicast and reverse multicast are usually modeled using a tree. In multicast the root is the source transmitting the data, the leaves are the receivers, and the intermediate nodes are the multicast routers receiving the content from their parent node and retransmitting it to their child nodes. This communication model provides scalability—the main objective of multicast communication—in the sense that the number of receiving leaves can be arbitrarily large. Security in multicast communications has been widely addressed in the literature [1].

Reverse multicast is much less studied. When the leaves act as senders and the root is the only receiver, the latter is very likely to be swamped if too many leaves transmit simultaneously, a

[*] Corresponding author.
E-mail addresses: francesc.sebe@urv.cat (F. Sebé), alexandre.viejo@urv.cat (A. Viejo), josep.domingo@urv.cat (J. Domingo-Ferrer).

problem known as implosion [8]. In addition to the scalability requirement of implosion-resistance, many-to-one transmission sometimes has security requirements such as confidentiality, authentication, integrity and non-repudiation.

Examples of applications requiring secure many-to-one communication in the above sense are

- A system sending TV and radio content as a multicast stream where the source collects statistical information about customer preferences. Here the source periodically queries the receivers about the program they are currently viewing/listening to. Obviously, information about TV and radio preferences is confidential. It is also desirable to prevent intruders from altering customer answers by unlawful data modification or injection; therefore, integrity and authentication are needed.
- A sensor network where a control center receives data from a vast quantity of sensors in real time. This kind of setting is usual in many critical infrastructures (nuclear power plants, chemical factories, etc.).

Cryptography permits secure data transmission over insecure communication channels. The security of cryptographic protocols is critically dependent on key management: access to secret keys by an intruder would compromise the security of the system. Properly managing keys is not easy if these are stored in devices accessible to attackers. A better option is to store cryptographic key material in tamper-resistant hardware such as smart cards. However, the limited computational power of smart cards requires careful design of the communication and cryptographic protocols based on them.

In the example of data collection of TV or radio preferences, a straightforward option would be to directly store cryptographic keys in the content receiver. The drawback of doing so is that receiver devices are vulnerable to attacks aiming at extracting the keys from them. Thus, a more secure option is to keep the cryptographic keys in a smart card plugged into the receiver.

1.1. Previous work on reverse multicast communication

In [10], a general framework for scalable many-to-one communication is presented. In that framework, intermediate nodes collect messages from their children, aggregate them and send a single aggregated message up to their parent. In this way, the root receives a single aggregated message. To meet the scalability requirement, aggregation of messages should not increase the length: the aggregated message should be no longer than the aggregation inputs. This is only possible if message aggregation tolerates some information loss. The shortcoming of the framework presented in [10] is that it does not address security.

Several proposals for secure many-to-one communications exist in the literature. They can be divided into two categories: *secure acknowledgment* and *secure symbol transmission*.

1.1.1. Secure acknowledgment

These schemes provide the root with an undeniable and unforgeable proof that a certain set of leaves have received a specific content. The information sent by the leaves to the root is unary in the sense that, after receiving a piece of data, every leaf will either respond with a positive acknowledgment (a digital signature) in case of correct reception or will stay silent otherwise.

The systems proposed in [6,4] fall into this category. The former uses the multisignature scheme in [2] constructed over a Gap Diffie–Hellman group (GDH) [3]. The latter is a construction whose security rests on the hardness of the discrete logarithm problem. Both solutions provide non-repudiation and are scalable ($O(n)$ message length) as long as the set of acknowledging leaves remains stable.

These systems satisfy no security properties beyond non-repudiation. For instance, the root is unable to distinguish a voluntary non-transmission from malicious erasure of acknowledgements by intruders. The authors in [6,4] leave this issue for future work. Thus, integrity is not ensured. Confidentiality is not achieved either since any intruder listening to the communication can ascertain which leaves are acknowledging and which are not.

1.1.2. Secure symbol transmission

Schemes in this category permit the leaves of a multicast tree to securely send q-ary symbols to the root.

In [5] the authors propose a system using superincreasing sequences and additive privacy homomorphisms. The length of messages is $O(n)$, where n is the number of leaves of the multicast tree. If implemented using the Okamoto–Uchiyama cryptosystem [7] for binary transmissions the message length asymptotically tends to $6n$. The scheme is easily extensible to accommodate q-ary alphabets with message length tending to $3tn$ (where $q \leqslant 2^t - 1$).

The proposal in [9] reduces the message length with respect to [5] for biased binary communication—i.e., where the probability of leaves transmitting a '1' symbol is less than the probability of their transmitting a '0' symbol. This scheme 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 wish to transmit simultaneously the least likely symbol. Both systems provide confidentiality, authentication and integrity. Non-repudiation is not provided.

In spite of their bandwidth efficiency, both proposals present a high computational cost. Both use additive public-key privacy homomorphisms, whose cleartext message length grows like $O(n)$ for [5] and $O(k \log k \log n)$ for [9]. The costly cryptographic operations on long messages required by these schemes render them ill-suited for implementation on resource-limited hardware like smart cards.

Regarding integrity, both systems permit detection of data corruption but identifying the corrupting nodes is not straightforward. This must be done using a tracking procedure described in [9] (which can also be applied for [5]) in which the root traces and identifies corrupting nodes.

1.2. Multisignatures over Gap Diffie–Hellman groups

The construction we propose uses multisignatures over a Gap Diffie–Hellman group. Next, we briefly introduce its mathematical background. A Gap Diffie–Hellman (GDH) group G is an algebraic group of prime order q for which no efficient algorithm can compute g^{ab} for random $g^a, g^b \in G$, but such that there exists an efficient algorithm $D(g^a, g^b, h)$ to decide whether $h = g^{ab}$. Let 1_G be the neutral element of G. GDH groups are suitable for public-key cryptography. The secret key is a random value $x \in \mathbb{Z}_q$ and its corresponding public key is $y \leftarrow g^x$. The signature on a message m is computed as $\sigma \leftarrow \mathscr{H}(m)^x$ ($\mathscr{H}$ is a cryptographic one-way hash function). The validity of a signature can be tested by checking $D(y, \mathscr{H}(m), \sigma)$.

GDH groups are convenient to compute multisignatures. Given two signatures of the same message m under two different public keys y_1, y_2, a signature of m under the combined public key $y \leftarrow y_1 y_2 = g^{(x_1+x_2)}$ can be obtained as $\mathscr{H}(m)^{x_1} \mathscr{H}(m)^{x_2} = \mathscr{H}(m)^{x_1+x_2}$.

1.3. Contribution and plan of this paper

We propose a scalable and secure protocol to provide many-to-one symbol transmission. Our proposal outperforms [9] in non-biased binary communication, that is, if yields a $O(n)$ message length shorter than $O(k \log k \log n)$ when $k \backsim n/2$. This is natural because [9] was designed for biased scenarios. Furthermore, our proposal improves on [5] in the following aspects:

- It offers non-repudiation.
- Its message length tends asymptotically to $2n$ for binary transmissions, which is better than the $6n$ achieved by [5]. For q-ary ($q \leqslant 2^t - 1$) transmissions, our message length tends to tn, which is better than the $3tn$ offered by [5].
- It reduces the computational cost at the leaves. Public-key cryptographic operations are reduced to a signature computation. This permits implementation of the leaf cryptographic functionality on a smart card.
- Intermediate routers can check the correctness of received messages prior to aggregating them. In this way, message corruption (whether intentional or accidental) can be detected immediately without requiring any extra error tracking procedure.

Section 2 describes the new protocol. Section 3 is a security analysis. A performance analysis is offered in Section 4. Error handling is discussed in Section 5. Section 6 describes a generalization of the protocol for q-ary transmission. Conclusions and future research directions are summarized in Section 7.

2. The new protocol

Our protocol assumes a tree communication model in which the root is the final receiver, internal tree nodes are reverse multicast routers and the leaves correspond to senders. The root S has a private key x_S and its corresponding public key $y_S \leftarrow g^{x_S}$. This public key is accepted by all the nodes of the tree (it may be certified by some recognized certification authority). Each leaf U_i has several private/public key pairs. The public keys are accepted as valid by intermediate routers and the root (they may be certified). Each node of the multicast tree knows its parent node and the public keys of nodes belonging to the subtree rooted at it. Each node also knows the public key of the root.

2.1. Reverse bit transmission

In this section we detail our proposal for binary communication, where each leaf U_i transmits a "0" or "1" bit (denoted by b_i). We assume the multicast tree contains n leaves U_i, $1 \leqslant i \leqslant n$. Each leaf U_i has two secret keys $x_{i,a}$ and $x_{i,b}$. Its corresponding public keys are $y_{i,a} \leftarrow g^{x_{i,a}}$ and $y_{i,b} \leftarrow g^{x_{i,b}}$. We also require each leaf U_i to share a secret key K_i with the root. This value can be agreed upon by using the Diffie–Hellman key exchange protocol. In this way, leaf U_i obtains K_i from one of its private keys and the root's public key, that is, $K_i = (y_S)^{x_{i,a}}$; the root can also obtain K_i by computing $K_i = (y_{i,a})^{x_S}$. The protocol works as follows:

(1) Challenge. The root multicasts to the leaves a challenge consisting of a random value v (v may include a description on the requested information.).
(2) Message generation
  (a) Upon receiving v, each leaf U_i computes a pseudo-random bit $c_i \leftarrow lsb_1(\mathscr{H}(v||K_i))$, where $lsb_1(\cdot)$ is a function returning the least significant bit of its argument.
  (b) If $c_i \oplus b_i = 1$ then U_i computes $\sigma_i := \mathscr{H}(v)^{x_{i,a}}$. If $c_i \oplus b_i = 0$ then U_i computes $\sigma_i := \mathscr{H}(v)^{x_{i,b}}$.
  (c) If $c_i \oplus b_i = 1$ then U_i generates a $2n$-bit sequence I_i so that its $2i$th bit is "1". The rest of the bits are set to "0".If $c_i \oplus b_i = 0$ then U_i generates a $2n$-bit sequence I_i so that its $(2i-1)$th bit is "1". The rest of the bits are set to "0".
  (d) U_i sends the pair (I_i, σ_i) up to its parent node.
(3) Message aggregation. An intermediate router R or the root S receives messages from its child routers/leaves and does the following:
  (a) For each received pair (I_j, σ_j):
    (i) Let $i := 1$. Let $y := 1_G$.
    (ii) While $i \leqslant n$ loop.
      • If $I_j[2i] = 1$ and $I_j[2i-1] = 0$ then $y := y \cdot y_{i,a}$.
      • If $I_j[2i] = 0$ and $I_j[2i-1] = 1$ then $y := y \cdot y_{i,b}$.
      • If $I_j[2i] = 1$ and $I_j[2i-1] = 1$ then $ERROR$.[1]
      • $i := i + 1$.
    (iii) It checks $D(y, \mathscr{H}(v), \sigma_j)$. If this check fails, then $ERROR$.
  (b) Once all expected messages $\{(I_j, \sigma_j)\}_j$ have been received and checked (for the sake of simplicity, we describe the protocol assuming no errors were found), R or S aggregate them by computing $I := \vee_j I_j$ ($\vee$ denotes the bit-wise OR operation) and $\sigma := \prod_j \sigma_j$.
  (c) If the aggregating node is an intermediate node R it sends (I, σ) up to its parent node. Else, if it is the root S this is the final aggregated message.
(4) Symbol extraction. From the final aggregated message (I, σ), the root S obtains the bit sent by each leaf as follows:
  (a) Let $i := 1$.
  (b) While $i \leqslant n$ loop.
    • Compute $c_i \leftarrow lsb_1(\mathscr{H}(v||K_i))$.
    • If $I[2i] = 1$ and $I[2i-1] = 0$ then $d_i := 1$ and $b_i := d_i \oplus c_i$.
    • If $I[2i] = 0$ and $I[2i-1] = 1$ then $d_i := 0$ and $b_i := d_i \oplus c_i$.
    • If $I[2i] = 0$ and $I[2i-1] = 0$ then $b_i := NULL$.
    • $i := i + 1$.
  (c) Return $B = (b_1, \ldots, b_n)$.

Note. No verification of the signature σ is needed during the extraction step, because σ is the aggregation of signatures σ_j which have been verified at each aggregation step (in the last aggregation step, verification has been carried out by the root itself).

3. Security analysis

We next analyze how our proposal provides confidentiality, authentication, integrity and non-repudiation.

3.1. Confidentiality

The confidentiality property refers to the fact that only the root should be able to obtain vector $B = (b_1, \ldots, b_n)$ containing the bit transmitted by each leaf. An intruder eavesdropping messages of

[1] Section 5 describes how to handle erroneous situations.

the form (I, σ) can determine for each leaf U_i located below the sniffing point in the tree whether the leaf transmitted $\sigma_i = \mathscr{H}(v)^{x_{i,a}}$, $\sigma_i = \mathscr{H}(v)^{x_{i,b}}$ or did not transmit by observing I (exactly the bits at $I[2i]$ and $I[2i-1]$).

From knowledge of σ_i the intruder is able to determine $c_i \oplus b_i$. But since c_i is only known to U_i and the root (it is computed from the challenge v and the shared secret key K_i), no one else is able to determine the transmitted value b_i.

Note that an intruder can determine which leaves did not transmit. In applications where this fact causes information leakage, non-transmission should not be permitted.

3.2. Authentication

This property requires that intruders cannot generate false messages that will be accepted as valid by the system. The creation of a message that will be accepted as authentic coming from U_i requires knowledge of its private key $x_{i,a}$ or $x_{i,b}$. This is because the message sent by U_i includes a signature σ_i over $\mathscr{H}(v)$ computed from $x_{i,a}$ or $x_{i,b}$. As long as secret keys are not compromised and the signature scheme is unforgeable (a valid signature can only be computed if the secret is known) the system provides authentication. The use of a different challenge v at each execution prevents replay attacks.

3.3. Integrity

This property requires being able to detect substitution or suppression of messages by an intruder. Given a message (I, σ), the field σ is a multisignature on $\mathscr{H}(v)$. Without loss of generality, let us take the case of a leaf U_i whose message has been aggregated into (I, σ) and assume that U_i transmitted $\sigma_i = \mathscr{H}(v)^{x_{i,a}}$. Further, assume that the value $\sigma_i = \mathscr{H}(v)^{x_{i,a}}$ is known to an attacker who could have obtained it by capturing the first message sent by U_i.

An attacker wishing to replace U_i's contribution with $\sigma'_i = \mathscr{H}(v)^{x_{i,b}}$ needs to replace σ with σ' so that $\sigma' = \sigma\mathscr{H}(v)^{x_{i,b}}(\mathscr{H}(v)^{x_{i,a}})^{-1}$. If this were possible, an attacker able to compute σ' would get $\mathscr{H}(v)^{x_{i,b}} := \frac{\sigma'\sigma_i}{\sigma}$ which is a signature on $\mathscr{H}(v)$ which would be validated using the public key $y_{i,b}$. This would contradict the unforgeability property of the GDH signature. In this sense, the system provides integrity against malicious alteration of the value b_i sent by a given leaf.

An intermediate node could dishonestly decide to suppress and not aggregate a message received from some of its child nodes. This act would be interpreted by the root as a non-transmission. Also, the contribution by U_i could be suppressed by an attacker who knew the value σ_i ($\mathscr{H}(v)^{x_{i,a}}$ or $\mathscr{H}(v)^{x_{i,b}}$). This can be done by computing $\sigma' = \sigma(\sigma_i)^{-1}$ (the corresponding alteration of I is trivial).

In order to avoid these suppression attacks, the protocol should not permit non-transmissions. In this way, if the root gets nothing from a leaf, a suppression attack is signalled.

3.4. Non-repudiation

This property requires that no leaf be able to deny having sent a given value that has been received by the root. A valid message (I, σ) reaching the root contains a value σ that is a multisignature on $\mathscr{H}(v)$. The non-repudiation property of the multisignature scheme guarantees that each leaf having contributed to the signature cannot deny having signed under the private key corresponding to one of its two public keys $y_{i,a}$ or $y_{i,b}$. Therefore, a leaf U_i cannot repudiate the value $c_i \oplus b_i$ she sent. Deriving non-repudiation on b_i from $c_i \oplus b_i$ requires prior declaration by U_i of the procedure used to obtain c_i. This procedure must later be reconstructable under observation of a third party.

4. Performance analysis

Performance will be analyzed in terms of message length, computational cost and implementability on smart cards.

4.1. Message length

In our proposal, the length of messages stays constant in the direction from the leaves towards the root. A message consists of the pair (I, σ). The bitlength of component I is $2n$ (being n the number of leaves) while the component σ is a multisignature constructed over a Gap Diffie–Hellman group [3]. This group can be constructed over non-supersingular elliptic curves to get a bitlength of approximately 170 bits which provides a security level similar to 320-bit DSA signatures or 1024-bit RSA signatures [3]. Thus, messages have a bitlength $2n + O(1)$. For large values of n, this length tends asymptotically to $2n$. This improves on the message length offered by [5], which tends asymptotically to $6n$.

4.2. Computational cost

The calculations performed by the protocol can be classified into four categories: message generation, message verification, message aggregation and data extraction. We next quantify the computation in each category.

Generation: Generation of (I_i, σ_i) by leaf U_i takes time $O(n)$ to generate the binary sequence I_i plus the time to compute the signature σ_i. Since this latter time does not depend on n, we take it as $O(1)$ in our analysis.

Verification: An intermediate node receives and checks messages $\{(I_j, \sigma_j)\}_j$ from its child nodes. For each (I_j, σ_j), the node computes y and then checks the validity of the multisignature σ_j. Computation of y requires one operation over the GDH group for each leaf that contributed to the message. The verification time of the signature does not depend on n, so we take it $O(1)$. Since there are $O(n)$ leaves in the tree, the overall amount of multiplications spent by one node computing the y's for all $\{(I_j, \sigma_j)\}_j$ is at most $O(n)$. The number of signatures to be verified depends on the number of child nodes of the intermediate router. In any case, this amount cannot grow faster than $O(n)$.

Aggregation: 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 and at most $O(n)$ operations (each one with cost $O(1)$) over the GDH group to compute the new multisignature. This results in a maximal $O(n^2)$ cost.

Extraction: Finally, the extraction of vector $B = (b_1, \ldots, b_n)$ by the root node is done by processing component I in time $O(n)$.

Table 1 compares our system with [5]. The cubic cost ($O(n^3)$) of message generation and data extraction in [5] is due to the encryption of $O(n)$ long messages using the Okamoto–Uchiyama homomorphic cryptosystem.

Table 1
Performance comparison with [5]

	Our proposal	[5]
Message length (for $n\uparrow\uparrow$)	$2n$	$6n$
Cost of message generation	$O(n)$	$O(n^3)$
Cost of message aggregation	$O(n^2)$	$O(n^3)$
Cost of data extraction	$O(n)$	$O(n^3)$

4.3. Implementability on smart cards

In our system, cryptographic private keys are stored at the root and leaves. The root is the central entity of the system, so we can assume it stays in a secure environment. This is not the case for the leaves. They are located on the customer side, so they could be attacked and their keys compromised. Therefore, it is necessary to store the keys of the leaves in tamper-resistant hardware, such as smart cards.

The system for collecting information on TV or radio preferences could be implemented as follows. Each leaf U_i has a smart card containing both private keys $x_{i,a}$ and $x_{i,b}$ and the root public key y_S. From $x_{i,a}$ and y_S it computes K_i. The challenge v sent by the root could carry a query asking whether the leaf is currently tuning a given channel. From this query, the smart card internally generates the answer b_i and computes and returns σ_i and $c_i \oplus b_i$ (step 2b of the protocol). Outside the smart card, from $c_i \oplus b_i$, the leaf would then generate the sequence I_i. Finally, the message (I_i, σ_i) would be sent up to its parent node.

Note that the only costly operation computed inside the smart card is a signature (taking $O(1)$ time). The $O(n)$ generation of I_i is performed off-card where more computational power may be available.

In a q-ary environment, the query sent by the root could directly ask about the channel that is being tuned. We will detail below how to extend our system for q-ary symbol transmission.

5. Error handling

Upon message reception, intermediate nodes perform several checks on the messages (I_j, σ_j) received from their child nodes prior to composing the aggregated message they will transmit. The checks that are always performed are

- Check that I_j does not contain $I_j[2i] = I_j[2i-1] = 1$ for any U_i (message generation does not permit this situation).

- Check that multisignature σ_j is consistent with the aggregated public key y computed from I_j.

If some of the above checks fail, the node will consider its sender (one of its child nodes) liable. This is because the child node either ought to have detected and reported these problems when performing its checks (if it was an intermediate node) or is causing the problems itself. In particular, if the child node is a leaf, it should have constructed an error-free message. Upon identification of a disrupting node, appropriate measures are taken against it (for instance, removal of the node from the multicast tree).

If non-transmission by leaves is not permitted, some additional requirements arise:

- First of all, an intermediate node has to receive one message (I_j, σ_j) from each of its children. If some of them are missing this will be interpreted as a malicious non-transmission by the corresponding children.
- Each intermediate node needs to know the list of leaves present in the subtree rooted at each of its child nodes. When checking each message (I_j, σ_j), it needs to check that all leaves present in this subtree are contributing. If this is not the case, this node will consider its sender child liable for having suppressed such contributions.

Note that neither the reception of a corrupted message nor a non-reception may be caused by the sender, but by an attacker disrupting the communication link between the sender and the receiver. In any case, the receiver cannot distinguish between the two situations. The receiver simply perceives that messages coming from that child are not reliable any more; upon this, the receiver can take the appropriate measures.

6. Generalization to q-ary transmission

The system can easily be generalized from binary to q-ary communications. We will represent each symbol from the q-ary alphabet by a different integer from the set $\{1, \ldots, q\}$. First of all, the smallest integer t meeting $q \leqslant 2^t - 1$ is chosen. Each leaf U_i has t secret keys $x_{i,1}, \ldots, x_{i,t}$, with their corresponding public keys $y_{i,1} \leftarrow g^{x_{i,1}}, \ldots, y_{i,t} \leftarrow g^{x_{i,t}}$ accepted as valid by intermediate routers and the root. As in the binary protocol, the root S shares a secret key K_i with each leaf.

The generalized protocol is as follows:

(1) CHALLENGE. The root multicasts a challenge consisting of a random value v (v may include a description of the requested information).
(2) MESSAGE GENERATION
  (a) Upon receiving the challenge v, each leaf U_i computes a pseudo-random t-bits sequence $(c_1, \ldots, c_t) \leftarrow lsb_t(\mathscr{H}(v \| K_i))$, where $lsb_t(\cdot)$ is a function returning the t least significant bits of its argument.
  (b) Let $(b_1, \ldots, b_t)$ be the binary representation of the symbol to be transmitted. Leaf U_i computes the sequence $(d_1, \ldots, d_t)$ by doing:
    - If $(b_1, \ldots, b_t) = (c_1, \ldots, c_t)$ then $(d_1, \ldots, d_t) := (b_1, \ldots, b_t)$. Else $(d_1, \ldots, d_t) := (b_1 \oplus c_1, \ldots, b_t \oplus c_t)$.
  (c) U_i generates a tn-bit sequence (where n is the number of leaves) I_i and sets the bits from the subsequence ranging from the $t(i-1)+1$ to the ti positions so that they match $(d_1, \ldots, d_t)$. The remaining bits are set to "0".
  (d) U_i computes $\sigma_i := \mathscr{H}(v)^{\sum_{p=1}^{t} d_p \cdot x_{i,p}}$.
  (e) U_i sends the pair (I_i, σ_i) up to its parent node.
(3) MESSAGE AGGREGATION. An intermediate router R or the root S receives messages from its child routers/leaves and does the following:
  (a) For each received pair (I_j, σ_j):
    (i) Let $i := 1$. Let $y := 1_G$.
    (ii) While $i \leqslant n$ loop.
      - $y := y \cdot \prod_{p=1}^{t} y_{i,p}^{I_j[t(i-1)+p]}$.
      - $i := i + 1$.
    (iii) It checks $D(y, \mathscr{H}(v), \sigma_j)$. If this check fails, then *ERROR*.
  (b) Once all expected messages $\{(I_j, \sigma_j)\}_j$ have been received, R aggregates them by computing $I := \vee_j I_j$ ($\vee$ denotes the bit-wise OR operation) and $\sigma := \prod_j \sigma_j$.
  (c) If R is an intermediate node, it sends (I, σ) up to its parent node. Else, if it is the root, this is the final aggregated message.
(4) SYMBOL EXTRACTION. From the final aggregated message (I, σ), the root obtains the symbol sent by each leaf as follows:
  (a) Let $i := 1$.
  (b) While $i \leqslant n$ loop.

- Compute $(c_1, \dots, c_t) \leftarrow lsb_t(\mathscr{H}(v||K_i))$.
- If $(I[t(i-1)+1], \dots, I[ti]) = (c_1, \dots, c_t)$ then $(b_{i,1}, \dots, b_{i,t}) := (c_1, \dots, c_t)$.
- Else $(b_{i,1}, \dots, b_{i,t}) := (I[t(i-1)+1] \oplus c_1, \dots, I[ti] \oplus c_t)$.
- $i := i + 1$.

(c) Return $B = ((b_{1,1}, \dots, b_{1,t}), \dots, (b_{n,1}, \dots, b_{n,t}))$ ($(b_{i,1}, \dots, b_{i,t})$ is the binary representation of the symbol transmitted by U_i).

The security and cost analysis of this extension is not included since it would be done in the same manner described for the binary protocol. In this case, the message length tends asymptotically to tn.

Note that step 2b above ensures that the sequence $(d_1, \dots, d_n)$ does not have all its elements equal to 0. If this were the case, the signature σ_i would equal 1 and would lose its non-repudiation and integrity properties.

7. Conclusions and future work

We have proposed a scalable tree-based protocol for secure many-to-one communication where the length of the aggregated message reaching the root is O(n), where n is the number of senders (leaves). The protocol uses multisignatures and offers the four basic security properties: confidentiality, integrity, authentication and non-repudiation. We have shown how to generalize the protocol from binary to q-ary communication.

From the performance point of view, the new protocol achieves a lower message length than [9] for non-biased binary communication (where the number of leaves transmitting “0” and “1” symbols at each time slot is similar), which is natural because [9] was designed for biased scenarios. It also outperforms [5] both in message length and computational cost. In what regards computation, the protocol is quite sparing at the leaves and the root:

- The cryptographic work required from leaves in our protocol is basically one signature, so the cryptographic functionality and the relevant keys can be securely and conveniently confined inside a tamper-resistant smart card.
- The protocol permits intermediate nodes to check the correctness of the received messages prior to aggregating them. In this way, message corruption (whether intentional or accidental) can be detected immediately without requiring any extra error tracking procedure.

Future work will focus on the design of secure many-to-one protocols for scenarios where nodes have limited computational resources. This fact may prevent the protocols from using some costly cryptographic operations such as public key encryption or digital signatures.

Acknowledgments

The authors are partly supported by the Spanish Ministry of Science and Education through project SEG2004-04352-C04-01 “PROPRIETAS” and by the Government of Catalonia under grant 2005 SGR 00446.

References

[1] M. Baugher, R. Canetti, L. Dondeti, F. Lindholm, Multicast security (MSEC) group key management architecture, Internet RFC 4046, 2005, http://www.ietf.org.

[2] A. Boldyreva, Efficient threshold signatures, multisignatures and blind signatures based on the Gap-Diffie–Hellman-group signature scheme, Lecture Notes in Computer Science 2567 (2003) 31–46.

[3] D. Boneh, B. Lynn, H. Shacham, Short signatures from the Weil pairing, Lecture Notes in Computer Science 2248 (2001) 514–532.

[4] C. Castelluccia, S. Jarecki, J. Kim, G. Tsudik, Secure acknowledgment aggregation and multisignatures with limited robustness, Computer Networks 50 (10) (2006) 1639–1652.

[5] J. Domingo-Ferrer, A. Martínez-Ballesté, F. Sebé, Secure reverse communications in a multicast tree, Lecture Notes in Computer Science 3042 (2004) 807–816.

[6] A. Nicolosi, D. Mazieres, Secure acknowledgement of multicast messages in open peer-to-peer networks, in: 3rd International Workshop on Peer-to-Peer Systems – IPTPS’04, San Diego, CA, 2004.

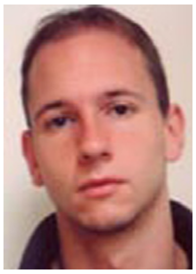
[7] T. Okamoto, S. Uchiyama, A new public-key cryptosystem as secure as factoring, Lecture Notes in Computer Science 1403 (1998) 308–318.

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

[9] F. Sebé, J. Domingo-Ferrer, Scalability and security in biased many-to-one communication, Computer Networks 51 (1) (2007) 1–13.

[10] T. Wolf, S.Y. Choi, Aggregated hierarchical multicast – a many-to-many communication paradigm using programmable networks, IEEE Transactions on Systems, Man and Cybernetics – Part C 33 (3) (2003) 358–369.

Francesc Sebé is a tenure-track Assistant Professor in Telematics Engineering at Rovira i Virgili University of Tarragona. He got his M.Sc. in Computer Engineering from Rovira i Virgili University in 2001 and his Ph.D. in Telematics Engineering from the Polytechnical University of Catalonia, Barcelona, in 2003. He has participated in several European and Spanish funded research projects. He has authored over 40 international publications. His fields of interest are cryptography and information privacy. In 2003, he was a co-recipient of a research prize from the Association of Telecom Engineers of Catalonia. In 2004, he was a visiting researcher at LAAS-CNRS, Toulouse, France.



Alexandre Viejo is currently working towards his Doctorate in Telematics Engineering from Polytechnical University of Catalonia. He got his M.Sc. in Computer Engineering from Rovira i Virgili University of Tarragona in 2005. His fields of interest are cryptography and information privacy.

Josep Domingo-Ferrer is a Full Professor of Computer Science at Rovira i Virgili University of Tarragona, Catalonia. He received with honors his M.Sc. and Ph.D. degrees in Computer Science from the Autonomous University of Barcelona in 1988 and 1991 (Outstanding Graduation Award). He also holds a M.Sc. in Mathematics. His fields of activity are data privacy, data security and cryptographic protocols. In 2003, he was a co-recipient of a research prize from the Association of Telecom Engineers of Catalonia. In 2004, he got the TOYPS'2004 Award from the Junior Chambers of Catalonia. He has authored 3 patents and over 170 publications, one of which became an ISI highly cited paper in early 2005. He was the co-ordinator of EU FP5 project CO-ORTHOGONAL and of several Spanish funded and US funded research projects. He has chaired or co-chaired eight international conferences (including two CARDIS conferences) and has served in the program committee of over 42 conferences on privacy and security. He is an Associate Editor of three international journals and has been a Guest Editor of *Computer Networks*, *IJUFKS* and *Data Mining and Knowledge Discovery*. In 2004, he was a Visiting Fellow at Princeton University. He is the Secretary of IFIP WG 8.8 on Smart Cards and he is the founder and chairholder of the forthcoming UNESCO Chair in Data Privacy.