Available online at www.sciencedirect.com
ScienceDirect
Computer Networks 51 (2007) 1–13
Computer Networks
www.elsevier.com/locate/comnet

Scalability and security in biased many-to-one communication

Francesc Sebé, Josep Domingo-Ferrer [*]

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

Received 21 June 2005; received in revised form 11 October 2005; accepted 21 November 2005
Available online 27 December 2005

Responsible Editor: Prof. C.B. Westphall

Abstract

In multicast communication, a source transmits the same content to a set of receivers. Current protocols for multicast follow a tree communication model which makes them scalable. This allows the set of receivers to be arbitrarily large. A large set of receivers (leaves) poses scalability problems when the multicast source (root) needs to collect data from the receivers. The literature on this subject is rather scarce. For a reverse multicast communication system to be scalable, it is necessary that intermediate multicast routers in the tree collect messages from their child nodes, aggregate them and send them back to their parent node. In this way, the root finally obtains a single message containing all data. Scalability also requires that aggregation of messages does not result in a size growth of the aggregated message. We focus on this problem with the extra requirements that leaf-to-root traffic should stay confidential and authentic. The few existing solutions satisfying all these requirements are such that, for a tree with n leaves, messages have an $O(n)$ constant length. Therefore, such proposals are only practical for moderate values of n. We propose here a new protocol offering confidentiality, integrity and authentication that is more efficient than previous literature when communication is biased, i.e., when the probabilities of sending '1' or '0' are clearly different. Messages have an $O(k \log k \log n)$ constant length, where k is an upper bound on the number of leaves transmitting the least probable bit in a certain time slot.
© 2005 Elsevier B.V. All rights reserved.

Keywords: Multicast; Many-to-one communication; Communications security

1. Introduction

Multicast is a communication paradigm in which a sender transmits the same data to a set of receivers (one-to-many communication). Multicast protocols should be scalable, i.e., they should stay practical even if the number of receivers is large. This is done using a tree communication model where the root corresponds to the data source, the leaves are the receivers and the intermediate nodes are multicast routers. In this structure, intermediate multicast routers receive the content from their parent node and retransmit it to their child nodes.

While scalability in root-to-leaf transmission is the *raison d'être* of multicast communication,

[*] Corresponding author. Tel.: +34 977558270; fax: +34 977559710.
E-mail addresses: francesc.sebe@urv.net (F. Sebé), josep.domingo@urv.net (J. Domingo-Ferrer).

scalability in leaf-to-root traffic is a much less investigated problem. The latter problem becomes relevant in several situations. Some examples follow:

- Real-time fee collection via micropayment in multicast content delivery. Subscribers are leaves and the content source is the root of the multicast tree.
- A network of sensors transmitting data to a control center. Sensors send data upon reception of a request multicast by the control center. Here, the control center should be able to cope with sensor input without being swamped.
- Keep-alive notifications sent by remote units to a control center upon a challenge multicast by the control center.
- On-line auctions where the auctioneer multicasts the current price and buyers signal whether they are willing to bid up.

In leaf-to-root communication, a large number of transmitting leaves can overwhelm the root with incoming traffic. This problem is known as *implosion* [6]. From the scalability point of view, a leaf-to-root communication protocol should be implosion-resistant. As stated in [9], for this requirement to be satisfied, messages from leaves must be aggregated as they traverse intermediate multicast routers on their way towards the root. It is also necessary that aggregation of messages does not increase their size. Two scenarios are possible:

Lossy aggregation. The message output by aggregation contains less information than the set of messages input to aggregation, so that the size of the output can stay the same as the size of each input.

Lossless aggregation. If no information loss is affordable during aggregation, then the only possibility left is for leaves to use a message length such that all information they transmit can be aggregated in a single message of that length (the message reaching the root). Of course, this implies that the actual informational content transmitted by leaves will be less than the bit-length of the messages they use.

If there are no confidentiality, integrity or authentication requirements, a trivial solution to lossless many-to-one binary communication is the following:

Protocol 0

1. Messages consist of $2n$ bits, where n is the number of leaves. The first two bits of the message are reserved for the payload of the first leaf, the third and fourth bits for the payload of the second leaf, and so on.
2. If a leaf wants to transmit a ‘1’ symbol up to the root, the leaf sends a full message where its reserved bits are ‘01’. If the leaf wants to transmit a ‘0’ symbol, it sends ‘10’ in its reserved positions. The rest of the positions are set to ‘0’.
3. Intermediate routers aggregate messages from their child nodes using bitwise OR as aggregation operator.
4. Upon reception of the aggregated $2n$-bit message, the root knows that a ‘1’ or ‘0’ symbol was transmitted by the ith leaf if its corresponding bits are ‘01’ or ‘10’, respectively.

In Protocol 0, a leaf that did not transmit anything is detected since its reserved bits are ‘00’. The remaining combination ‘11’ can be used for a third symbol in a ternary communication context. Note that the size of the message output by aggregation is the same as the size of each input message.

Protocol 0 is nicely simple, but unfortunately it does not provide any security to communications. Typical security requirements are:

Confidentiality. An intruder eavesdropping the path from a particular leaf to the root should be unable to discover the information transmitted by a particular leaf. In some cases, one is interested in keeping confidential not just what a leaf transmitted, but even whether the leaf transmitted at all. A way to achieve this second type of confidentiality is for all leaves to transmit always (if a leaf has no actual information to send in a particular time slot, it can send some kind of encrypted padding).

Integrity. The communication protocol should ensure that, if the leaf-to-root traffic is altered by an intruder, this will be detected.

Authentication. Only real leaves should be able to send information to the root. So, the protocol must ensure that injection of bogus traffic is detected.

The above requirements are easily justified using the examples suggested above:

- In fee collection for multicast transmission, confidentiality, integrity and authentication should be guaranteed for payment information sent by subscribers.
- In a network of sensors transmitting data to a control center, no intruder should be able to counterfeit (or even ascertain in some cases) the status information sent by a certain sensor.
- The previous example also applies to a case in which the control center just requests a keep-alive signal rather than status information.
- In an on-line auction, the need for bid authentication, and often for bid confidentiality, is obvious. The amount of the highest bid certainly needs to be known to all bidders, but there is no need for anyone but the auctioneer to know who offered this amount. If only buyers bidding up responded, their identity could be disclosed, especially if there is just one buyer bidding up. Therefore, all buyers should always respond to a bid request by the auctioneer (with a positive or negative encrypted reply).

Adding encryption to Protocol 0 is a way to obtain confidentiality. However, integrity and authentication require that the payload of each leaf be encrypted together with some redundancy. This redundancy should provide data sender authentication and resistance against malicious alterations. If one wishes the success probability of random alteration attacks to be negligible, then one should use R redundancy bits per payload bit where 2^{-R} is negligible. Assuming that a cryptosystem is used such that the ciphertext length is the same as the cleartext length (e.g., a flow cipher), the length of messages would be at least $(R+2)n$ bits. For example, when using MD5 [7] in the Transport Layer Security protocol [1], one has a 128-bit message authentication code, so $R = 128$ bits. However, this secure version of Protocol 0 is much more wasteful than existing proposals in the literature: the scheme [2] combined with the Okamoto–Uchiyama cryptosystem [4] provides implosion resistance, confidentiality, integrity and authentication with shorter messages (about $6n$ bits for large n) for binary leaf-to-root communications.

1.1. Contribution and plan of this paper

There is not much literature on how to achieve implosion resistance, confidentiality, integrity and authentication in lossless leaf-to-root multicast communication. Furthermore, the few existing solutions [2,3] convey a message to the root whose length is $O(n)$ bits (see Section 1), with n being the number of leaves.

We consider in this paper a scenario with biased binary communication, where the probabilities of leaves sending ‘1’ or ‘0’ symbols are clearly different (but both probabilities are constant for all leaves). An example of such a scenario could be a network of alarms, where most of the time slots leaves (alarms) send the “OK” signal. Assuming “OK” is encoded as ‘0’ and “Emergency” as ‘1’, the probability of a leaf sending ‘0’ is much greater than that of sending ‘1’. An on-line auction can be another example: only a minority of buyers is likely to bid up when the price increases, so the probability of a positive response (‘1’) is less than the probability of a negative one (‘0’).

Let k be an upper bound on the number of leaves (among the n overall leaves) that transmit the least likely symbol in a certain time slot. We present a solution providing confidentiality, integrity and authentication where all messages in the leaf-to-root flow have the same bitlength $O(k \log k \log n)$. For large n and moderate k, our algorithm outperforms previous $O(n)$ proposals.

The description of our solution is for binary transmissions. However, it can be easily extended for q-ary symbol transmission using the ideas of [3].

From the security standpoint, our proposal assumes that the root node shares a secret key with each leaf (like in [2,3]). It provides confidentiality, integrity and authentication in the sense explained above:

- An eavesdropper cannot use sniffed traffic to determine what or even whether a leaf transmitted.
- False data insertion, removal or alteration by intruders are detected.

Our construction consists of a hybrid proposal built on two leaf-to-root protocols:

- The first protocol is very efficient in length. It conveys a message to the root node whose length is fixed and independent of n and k (this is, $O(1)$). It provides confidentiality and allows detection of false data insertion, deletion or alteration with arbitrarily high probability. However, direct use of this protocol to decode an incoming transmission from the leaves (i.e., to determine whether each leaf transmitted ‘0’ or ‘1’) would take a time exponential in n at the root. The reason is that decoding ought to be done by exhaustively checking the correctness of all possible leaf transmissions, which is impractical and can result in a non-negligible error probability when the number of possible transmissions is large. Thus, rather than using this protocol for decoding, we will use it to check the correctness of the decoding output by the second protocol below.

- The second protocol results in a message length $O(k \log k \log n)$. It provides unary communication in the sense that each leaf has two options: transmit ‘1’ or transmit ‘0’. The root cannot distinguish a ‘0’ transmission from a non-transmission. This fact causes message deletion to be undetectable. If the message from a particular leaf is removed, the root will simply conclude that the leaf did not transmit ‘1’. Therefore, even though this second protocol provides confidentiality, it does not guarantee integrity: an intruder cannot insert valid bogus messages, but accidental or intentional data corruption or deletion is possible and undetectable. The decoding time at the root is at most quasi-linear in n, specifically $O(kn \log n)$, which is affordable. Aggregation at intermediate nodes is lossless as long as the capacity k is not exceeded and parameters have properly been selected.
- The final solution is a hybrid of the previous two: both protocols are used for transmission, the second protocol is used for decoding and the first protocol is used to check the correctness of decoding. Thus, the hybrid protocol takes the best properties of each of the two preceding protocols. Confidentiality, authentication and integrity are provided and messages have bitlength $O(k \log k \log n)$. Decoding time is $O(kn \log n)$ and the probability of undetected error is arbitrarily small. Exceeding the capacity k can lead to erroneous message decoding but this condition is detected.

Section 2 reviews the literature on leaf-to-root multicast transmission. Section 3 describes the first protocol. Section 4 contains a description of the second protocol, an error analysis, guidelines for parameter choice and a performance assessment. The hybrid combination of the first and second protocols is presented in Section 5, together with a tracing protocol to locate tampering nodes. Finally, Section 6 is a conclusion.

2. Previous work on leaf-to-root multicast transmission

In [9], a general framework is presented based on the principle that intermediate routers collect data from their child nodes, aggregate them and send them up to their parent node. This solution is valid as long as aggregated data do not grow in size. But requiring that the output of the aggregation be of constant size implies that, at some stage, the aggregation procedure may be forced to compress its inputs, which may not be possible without some information loss.

In [2,3], protocols providing constant message size and secure reverse bit and q-ary symbol transmission in a multicast tree are presented. Those solutions use additive privacy homomorphisms to perform lossless aggregation of encrypted data, which offers confidentiality, integrity and authentication. In those solutions, the length of messages is proportional to the number n of leaves in the tree.

Both [2,3] use a modular additive privacy homomorphism, that is, a cryptosystem that provides an operation $\oplus$ taking as input two ciphertexts $C_1 = E(m_1)$, $C_2 = E(m_2)$ so that $C_1 \oplus C_2 = E((m_1 + m_2) \bmod p)$ for a given modulus p. The privacy homomorphism used is required to be public-key, semantically secure and without ciphertext expansion. The latter property means that the output of combining two ciphertext inputs using $\oplus$ should be no longer than each individual input. The Okamoto–Uchiyama homomorphism [4] is one that fulfills all desired properties.

Both [2,3] are based on a transmission paradigm that can be sketched as

Initial setting:

- each leaf station U_i shares a secret key K_i with the root node;
- the root node publishes her public key PK_R corresponding to a modular additive privacy homomorphism with the aforementioned properties;
- each leaf U_i wishes to transmit a binary or q-ary value b_i.

Data transmission scheme in a time slot:

1. the root node multicasts a request to all n leaves;
2. each leaf U_i transmits b_i by:
   (a) computing m_i as a function of the request, its shared secret key K_i and data b_i;
   (b) computing $M_i = E_{PK_R}(m_i)$;
   (c) sending M_i up to its parent node.

Intermediate nodes (routers) aggregate received messages using the homomorphic operation $\oplus$ and send the resulting ciphertext up to their parent node. Finally, the root performs the last aggregation to get a message M that once decrypted allows her to obtain $m \equiv (\sum_{i=1}^{n} m_i)(\bmod\, p)$. From m the root is able to obtain $B = (b_1, \ldots, b_n)$. If m does not contain any value for b_i, this means that either U_i did not transmit anything or U_i's message was accidentally or maliciously removed. In this way, message corruption is detectable from m.

3. First protocol

The protocol in this section is one of the two components of the proposed solution. This protocol uses the same initial setting and data transmission scheme described in the previous section. Thus, like [2,3], it uses a privacy homomorphism to encrypt and aggregate values which is required to be public-key, semantically secure and without ciphertext expansion.

Message aggregation in this protocol is lossy. We describe the protocol here for binary communications where each leaf U_i wishes to send a binary value $b_i = \{0, 1\}$. The protocol does not allow the root to obtain the vector $B = (b_1, \ldots, b_n)$ sent by the leaves. Instead, the root must specify a candidate B' as an input parameter and then use the protocol to check whether $B \overset{?}{=} B'$.

The detailed transmission scheme works as follows:

Protocol 1 (*First protocol*)

1. A transmission request which contains a random value v is multicast by the root to all leaves.
2. Upon reception of the request, each leaf U_i does:
   (a) Compute $v_{i,0} = \mathscr{H}(\overline{v} \| K_i)$ and $v_{i,1} = \mathscr{H}(v \| K_i)$ ($\overline{v}$ denotes the bitwise NOT operation applied to v and $\mathscr{H}(\cdot)$ is a collision-free hash function returning a positive integer $<p$).
   (b) Use the privacy homomorphism to encrypt as $M_i = E_{PK_R}(v_{i,1} \cdot b_i + v_{i,0} \cdot \overline{b_i})$.
3. Intermediate nodes aggregate received messages using the homomorphic operation $\oplus$.
4. The root node does:
   (a) Perform the last aggregation to get M.
   (b) Decrypt $m =: D_{SK_R}(M)$, where SK_R is the root's private key corresponding to public key PK_R.
   (c) Compute $v_{i,0} = \mathscr{H}(\overline{v} \| K_i)$ and $v_{i,1} = \mathscr{H}(v \| K_i)$, for all i.
   (d) Given a vector $B' = (b'_1, \ldots, b'_n)$, find out whether B' is equal to the vector $B = (b_1, \ldots, b_n)$ transmitted by the leaves by verifying

$$m \overset{?}{\equiv} \sum_{i=1}^{n} (b'_i \cdot v_{i,1} + \overline{b'}_i \cdot v_{i,0})(\bmod\, p).$$

   (If the above check is positive, the root concludes that $B = B'$; otherwise it concludes that $B \neq B'$.)

The above protocol has several strong points:

- It allows the root to check the correctness of the bits sent by an arbitrarily large number of leaves.
- Checking one vector B' takes a O(n) cost.
- The size of messages is the size of a cryptogram containing a $|p|$-bit long cleartext message, regardless of the number n of leaves. If the Okamoto–Uchiyama [4] cryptosystem is used, the cryptogram length is $3|p|$ bits.
- Transmitted messages are encrypted with the root's public key, so that only the root can decrypt them. This ensures confidentiality.
- An intruder cannot insert a false message and have it accepted by the root as if it came from some leaf U_i. This is so because K_i is needed to compute $v_{i,0}$, $v_{i,1}$. Also, these values are different for each protocol instance (they depend on the random value v). In this way, messages captured in one protocol instance cannot be replayed in subsequent instances.
- Modification by an intruder of a message M_i coming from leaf U_i will be detected with overwhelming probability. The reason is that, since M_i is encrypted under the public key of the root, random alteration of M_i will result in an m that will not pass the check at Step (4d).
- Message removal is detected, because, under normal circumstances, the root should find a contribution by every leaf in Step (4d).
- The probability that a wrong vector B' yields the same m modulo p as the correct vector B is $1/p$. This probability can be made arbitrarily small by increasing p. For an l-bit long p, we have an exponentially small error probability 2^{-l}.

Note 1. A limitation of the above protocol is that at Step (4d) one needs a (externally obtained) candidate vector B' whose correctness can be checked. If there was no candidate, the only possibility would be exhaustive search until a B' was found that passed the correctness verification. But this would take $O(n2^n)$ time and, worse yet, the per-check error probability $1/p$ would accumulate into a non-negligible error probability for the overall $O(2^n)$ checks.

Later in this paper (Section 5), we will show that the first protocol above will be used in combination with a second protocol; the latter will provide the candidate B' and the first protocol will be used to check its correctness. Although the above protocol can easily be extended to accommodate non-transmission by a leaf in addition to binary transmission, we require that each leaf send '1' or '0' each time slot. Otherwise, message removal by some intruder could be interpreted by the root as non-transmission by the affected leaves. Also, as discussed in Section 1, this is necessary for confidentiality.

4. Second protocol

Just like the protocol in Section 3, the protocol below uses homomorphic aggregation at intermediate nodes. The requirements on the privacy homomorphism are the same: public-key, semantic security and no ciphertext expansion. Unlike the protocol in Section 3, the protocol below exploits the fact that we are focusing on biased binary communication. Without loss of generality, we assume in what follows that the probability of leaves transmitting a '1' symbol is less than the probability of their transmitting a '0' symbol. Let q be an integer parameter such that $2^q > k$, where k is an upper bound on the number of leaves that wish to transmit the least likely symbol, that is, to send $b_i = 1$ (this is a sufficient condition to avoid overflow during message aggregation, see Sections 4.1 and 4.2). Let r and l be two integer parameters whose selection also depends on k (their specific selection is discussed in Sections 4.3 and 4.4).

In this second protocol, transmission is as follows:

Protocol 2 (*Second protocol*)

1. The transmission request multicast from the root to the leaves contains a random value v, the three parameters q, r, l and an r-degree binary primitive polynomial $\mathbf{P}$.
2. Each leaf U_i does:
   (a) Generate a pseudo-random sequence $s = (s_1, \ldots, s_{2^r-1})$, where $s_j \in \{0,1\}$ for all $j \in \{1, \ldots, 2^r - 1\}$, using a linear feed-back shift register (LFSR) whose connection polynomial is $\mathbf{P}$. The initial LFSR state must be agreed by all leaves and the root (the all 1's vector is a possibility).
   (b) Generate U_i's identification sequence $I_i = (I_{i,1}, \ldots, I_{i,l})$ as l pseudo-random integers picked in the range $[0, 2^r - 2]$. The pseudo-random generator is seeded with K_i and v.
   (c) Generate U_i's final sequence $S_i = (S_{i,1}, \ldots, S_{i,l(2^r-1)})$ as
   $$S_i = (s^{(I_{i,1})} \| s^{(I_{i,2})} \| \cdots \| s^{(I_{i,l})}),$$
   where $s^{(d)}$ denotes a d-bit left circular shift of sequence s.
   (d) Pack S_i into an integer v_i by computing
   $$v_i = 2^{ql(2^r-1)} + \sum_{j=1}^{l(2^r-1)} 2^{q(j-1)} S_{i,j}. \quad (1)$$
   (Note that, if two integers v_1, v_2 pack S_1 and S_2, respectively, then $v_1 + v_2$ packs the sequence $S_1 + S_2$ defined as the term-wise addition of S_1 and S_2.)
   (e) Compute $m_i = v_i \cdot b_i$.
   (f) Use the privacy homomorphism to encrypt m_i as $M_i = E_{PK_R}(m_i)$. Even if $m_i = 0$, as it happens when $b_i = 0$ is transmitted, confidentiality is guaranteed: the semantic security requirement on the privacy homomorphism ensures that $E_{PK_R}(0)$ is indistinguishable from $E_{PK_R}(m_i)$ for $m_i \neq 0$ and also that successive encryptions of $m_i = 0$ are unlinkable.
   (g) Send M_i up to U_i's parent node.
3. Intermediate nodes aggregate received messages using the homomorphic operation $\oplus$.
4. The root node does:
   (a) Perform the last aggregation to obtain message M and decrypt it to get m.
   (b) Recover the number of leaves that have sent the least likely symbol ($b_i = 1$) as $\hat{k} = m \,\mathtt{div}\, 2^{ql(2^r-1)}$, where `div` denotes integer division. See Lemma 1 below for a justification.
   (c) Extract the sequence $S' = (s'_1, \ldots, s'_{l(2^r-1)})$ contained in m by computing $s'_j = (m \,\mathtt{div}\, 2^{q(j-1)}) \bmod 2^q$, for $j \in \{1, \ldots, l(2^r - 1)\}$. See Lemma 2 below for a justification.

(d) Parse sequence S' as the concatenation of l different (2^r-1)-long sequences, that is,

$$S' = (S'_1 \| S'_2 \| \cdots \| S'_l).$$

(e) Do $\forall t \in \{1,\ldots,l\}$, $\forall d \in \{0,\ldots,2^r-2\}$, if

$$\hat{k} + \sum_{j=1}^{2^r-1} (2S'_{t,j} - \hat{k})(2s_j^{(d)} - 1) > 0,$$

then $Rec[t][d] = 1$, else $Rec[t][d] = 0$.(Rec is a bidimensional table such that $Rec[t][d] = 1$ indicates that the tth subsequence S'_t of S' contains $s^{(d)}$; see Section 4.3 below for a justification.)

(f) Generate the sequences I_i (the root node can do this because she knows v and K_i for each leaf U_i).

(g) For each U_i, if $Rec[t][I_{i,t}] = 1$, $\forall t \in \{1,\ldots,l\}$ then recover $b_i = 1$, else recover $b_i = 0$.

(h) Return $B = (b_1,\ldots,b_n)$.

4.1. The role of parameter q

We now justify that, if q is such that $2^q > k$, and the number of leaves transmitting '1' is not greater than k, then the sequence S' obtained by the root at Step (4c) satisfies $S' = \sum_{i=1}^n b_i S_i$, i.e., S' is the aggregation of the sequences of leaves having transmitted $b_i = 1$.

Lemma 1 refers to Step (4b) of the second protocol above.

Lemma 1. *Let $\hat{k}$ be number of leaves having transmitted $b_i = 1$, that is, $\hat{k} = \sum_{i=1}^n b_i$. If $\hat{k} < 2^q$ and m is the result of decrypting the final aggregated message by the root, then $m\,\mathtt{div}\,2^{ql(2^r-1)} = \hat{k}$.*

Proof. According to expression (1), each nonzero integer $m_i = v_i$ (packing a sequence) that has been aggregated into m contributes a $2^{ql(2^r-1)}$ term plus a term

$$\sum_{j=1}^{l(2^r-1)} 2^{q(j-1)} S_{i,j} \leqslant \sum_{j=1}^{l(2^r-1)} 2^{q(j-1)} = \frac{2^{ql(2^r-1)} - 1}{2^q - 1} < \frac{2^{ql(2^r-1)}}{2^q} = 2^{q(l(2^r-1)-1)}.$$

Thus, if $\hat{k} < 2^q$ nonzero sequences are aggregated, m contains a first term which is $\hat{k}2^{ql(2^r-1)}$ plus a second term which is less than $2^{ql(2^r-1)}$. Therefore,

$$m\,\mathtt{div}\,2^{ql(2^r-1)} = \hat{k}.$$

Since $\hat{k} \leqslant k$, if $k < 2^q$ the above lemma holds. □

Lemma 2 refers to Step (4c) of the second protocol above.

Lemma 2. *Let $\hat{k}$ be number of leaves having transmitted $b_i = 1$, that is, $\hat{k} = \sum_{i=1}^n b_i$. If $\hat{k} < 2^q$ and m is the result of decrypting the final aggregated message by the root, then $(m\,\mathtt{div}\,2^{q(j-1)}) mod\, 2^q = \sum_{i=1}^n b_i S_{i,j}$ for $j \in \{1,\ldots,l(2^r-1)\}$.*

Proof. We have

$$m = \sum_{i=1}^n b_i v_i = 2^{ql(2^r-1)} \sum_{i=1}^n b_i + \sum_{i=1}^n \sum_{j=1}^{l(2^r-1)} 2^{q(j-1)} b_i S_{i,j}, \quad (2)$$

where v_i has been expanded using expression (1). Now take any $j' \in \{1,\ldots,l(2^r-1)\}$ and compute the integer division of Eq. (2) by $2^{q(j'-1)}$

$$m\,\mathtt{div}\,2^{q(j'-1)} = \left(2^{ql(2^r-1)} \sum_{i=1}^n b_i + \sum_{j=1}^{j'-1} \left(2^{q(j-1)} \sum_{i=1}^n b_i S_{i,j}\right) + \sum_{j=j'}^{l(2^r-1)} \left(2^{q(j-1)} \sum_{i=1}^n b_i S_{i,j}\right)\right) \mathtt{div}\,2^{q(j'-1)}. \quad (3)$$

Let us bound the second term on the right-hand side of Eq. (3):

$$\sum_{j=1}^{j'-1} \left(2^{q(j-1)} \sum_{i=1}^n b_i S_{i,j}\right) < \sum_{j=1}^{j'-1} 2^{qj} = \frac{2^{qj'} - 2^q}{2^q - 1} < \frac{2^{qj'} - 2^q}{2^q} = 2^{q(j'-1)} - 1. \quad (4)$$

The first bound in the above derivation comes from the lemma assumption that $\hat{k} < 2^q$, that is, which implies that the number of leaves transmitting $b_i = 1$ is less than 2^q or equivalently $\sum_{i=1}^n b_i S_{i,j} < 2^q$. Using bound (4), we have that the second term on the right-hand side of Eq. (3) vanishes after integer division by $2^{q(j'-1)}$, so we get

$$m\,\mathtt{div}\,2^{q(j'-1)} = 2^{q(l(2^r-1)-j'+1)} \sum_{i=1}^n b_i + \sum_{j=j'}^{l(2^r-1)} \left(2^{q(j-j')} \sum_{i=1}^n b_i S_{i,j}\right). \quad (5)$$

Finally, let us reduce Eq. (5) modulo 2^q to obtain

$$(m\,\mathtt{div}\,2^{q(j'-1)}) \bmod 2^q = \sum_{i=1}^{n} b_i S_{i,j'}. \qquad \square$$

4.2. Optimal choice of parameter q

We need $k < 2^q$ for the above correctness lemma to hold. At the same time, since the length of v_i grows with q, we are interested in taking the smallest possible value for q. This yields as optimal value $q = \lceil \log_2 k \rceil$.

4.3. The role of parameters r, l

We will show in this section that parameters r and l determine the probability of decoding error.

We first give a supporting lemma.

Lemma 3. *Let s be a pseudo-random binary sequence generated by an r-cell LFSR with a primitive connection polynomial. Let $s^{(d)}$ be a (d)-bit left circular shift of s and let $(A_0, \ldots, A_{2^r-2})$ be a vector of real numbers. If $S := \sum_{d=0}^{2^r-2} A_d s^{(d)}$ and $k = \sum_{d=0}^{2^r-2} A_d$, it holds that*

$$k + \sum_{j=1}^{2^r-1} (2S_j - k) \cdot (2s_j^{(d)} - 1) = A_d 2^r. \qquad (6)$$

Proof. An r-cell LFSR whose connection polynomial is primitive generates a pseudo-random sequence s, $s_j \in \{0,1\}$ whose period is $2^r - 1$ and whose auto-correlation function $\sum_{j=1}^{2^r-1}(2s_j - 1)\cdot(2s_j^{(d)} - 1)$ is $2^r - 1$ when d is a multiple of $2^r - 1$ (including $d = 0$) and -1 otherwise (see [5]).

Now, let us expand the left-hand side of Eq. (6) using the definitions of S and k and the above auto-correlation property:

$$\begin{aligned}
&k + \sum_{j=1}^{2^r-1} (2S_j - k) \cdot (2s_j^{(d)} - 1) \\
&\quad = k + \sum_{j=1}^{2^r-1} (2A_0 s_j^{(0)} - A_0) \cdot (2s_j^{(d)} - 1) + \cdots \\
&\qquad + \sum_{j=1}^{2^r-1} (2A_d s_j^{(d)} - A_d) \cdot (2s_j^{(d)} - 1) + \cdots \\
&\qquad + \sum_{j=1}^{2^r-1} (2A_{2^r-2} s_j^{(2^r-2)} - A_{2^r-2}) \cdot (2s_j^{(d)} - 1) \\
&\quad = k - A_0 - \cdots - A_{d-1} + A_d(2^r - 1) \\
&\qquad - A_{d+1} - \cdots - A_{2^r-2} = A_d 2^r. \qquad \square
\end{aligned}$$

Next, by the encoding construction in Step (2), the tth subsequence S'_t to be decoded by the root node at Step (4e) is the sum of $s^{(I_{i,t})}$ for all those leaves U_i that picked $I_{i,t}$ as the tth integer of their identification sequence *and* sent $b_i = 1$. Using the notation of Lemma 3 the number of such leaves is $A_{I_{i,t}}$. Adding for all possible values of $I_{i,t}$, we get the total number $\hat{k}$ of leaves that sent $b_i = 1$, that is,

$$\sum_{I_{i,t}=0}^{2^r-2} A_{I_{i,t}} = \hat{k}.$$

Now, note that what is done at Step (4e) is to use expression (6) on S'_t. If, for a particular d, the result $A_d 2^r$ is greater than 0, the root node infers that $A_d > 0$, i.e., there was at least one leaf contributing $s^{(d)}$ to S'_t; to indicate this, the root node sets $Rec[t][d] = 1$. On the other hand, a result $A_d 2^r = 0$ implies $A_d = 0$, from which the root node infers that no leaf contributed $s^{(d)}$; this is recorded as $Rec[t][d] = 0$.

From the arguments in the two previous paragraphs, it is natural for the root node to infer that U_i sent $b_i = 1$ if and only if $Rec[t][I_{i,t}] = 1$ for all $t \in \{1, \ldots, l\}$. This explains the way b_i is extracted at Step (4g) of the protocol.

As long as $\hat{k} < 2^q$, the contribution of a leaf sending $b_i = 1$ will always be correctly decoded. This is not the case for leaves sending $b_i = 0$. Let us now turn to the probability that the decoding procedure erroneously returns $b_i = 1$ when U_i sent $b_i = 0$.

Lemma 4. *If $\hat{k}$ out of n leaves of the multicast tree transmitted $b_i = 1$, the probability of incorrectly identifying those leaves is*

$$\epsilon = 1 - \left(1 - \left(1 - \left(\frac{2^r - 2}{2^r - 1}\right)^{\hat{k}}\right)^{l}\right)^{n-\hat{k}}.$$

Proof. Let $I_{i,t}$ be the tth pseudo-random integer of the identification sequence of a leaf U_i which transmitted $b_i = 0$. The probability that $I_{i,t}$ does not collide with the tth integer in the sequence of any of the $\hat{k}$ leaves which transmitted '1' is

$$\left(\frac{2^r - 2}{2^r - 1}\right)^{\hat{k}}.$$

Then, the probability of a collision in the tth integer of U_i is

$$1-\left(\frac{2^r-2}{2^r-1}\right)^{\hat{k}}.$$

The decoding procedure will return $b_i = 1$ for leaf U_i if all of its l integers collide. This happens with probability

$$\left(1-\left(\frac{2^r-2}{2^r-1}\right)^{\hat{k}}\right)^{l}.$$

Then, the probability that the decoding procedure returns $b_i = 0$ for leaf U_i is

$$1-\left(1-\left(\frac{2^r-2}{2^r-1}\right)^{\hat{k}}\right)^{l}.$$

Since we have $n-\hat{k}$ leaves transmitting '0', the decoding procedure will work properly if it returns $b_i = 0$ for all them, which happens with probability

$$\left(1-\left(1-\left(\frac{2^r-2}{2^r-1}\right)^{\hat{k}}\right)^{l}\right)^{n-\hat{k}}.$$

So, the probability of incorrect identification of all leaves transmitting '1' is

$$\epsilon = 1-\left(1-\left(1-\left(\frac{2^r-2}{2^r-1}\right)^{\hat{k}}\right)^{l}\right)^{n-\hat{k}}. \qquad \square$$

4.4. Optimal choice of parameters r and l

When selecting values for r and l, we must take the following into account:

- The total amount n of leaves of the multicast tree and the maximum amount k of stations that will transmit $b_i = 1$ during a time slot are fixed.
- Parameter q satisfying $2^q < k$ is already fixed too.
- We should be able to enforce a maximum acceptable value $\epsilon_{\max}$ for the error probability ϵ.

With the above constraints, the optimal choice for parameters r and l is the one minimizing the length $(2^r-1)l$ of sequences S_i while keeping the error probability less than $\epsilon_{\max}$. Thus, the optimal values of r and l are the solution of the following constrained minimization problem

$$\min_{r,l}(2^r-1)l$$

subject to r and l being positive integers and

$$1-\left(1-\left(1-\left(\frac{2^r-2}{2^r-1}\right)^{k}\right)^{l}\right)^{n-k} < \epsilon_{\max}.$$

4.5. Performance: sequence length

We investigate in this section the relationship between the error probability ϵ, the number of leaves n, the number of transmitting leaves $\hat{k}$ and the length $L = l(2^r-1)$ of sequences used in the protocol. Deriving such a relationship analytically is far from obvious, because r and l are the solution of a constrained optimization problem. Therefore, we will adopt a statistical approach to explore that relationship for reasonable values of n, $\hat{k}$ and $\epsilon_{\max}$.

Optimal r and l values are shown for $\epsilon_{\max} = 0.01$ and several values of $\hat{k}$ in Tables 1 and 2. The former table is for $n = 2000$ leaves and the latter table for $n = 4000$ leaves. For each $(\hat{k}, r, l)$ combination, the tables show the corresponding sequence length and error probability ϵ.

From the numerical values in Tables 1 and 2, we observe that, for fixed n and $\epsilon_{\max}$, values $L = l(2^r-1)$ and $\hat{k}$ in Table 1 present a Pearson correlation coefficient of 0.9996. For Table 2 this coefficient is 0.9988. Therefore, we can take the dependence between L and $\hat{k}$ as being linear.

Table 1
Optimal r and l values for $n = 2000$ and $\epsilon_{\max} = 0.01$ depending on $\hat{k}$

$\hat{k}$	r	l	$L = l(2^r-1)$	ϵ
40	6	17	1071	0.0057
80	7	17	2159	0.0049
120	7	25	3159	0.0090
160	8	16	4080	0.0093
200	8	20	5100	0.0093

Table 2
Optimal r and l values for $n = 4000$ and $\epsilon_{\max} = 0.01$ depending on $\hat{k}$

$\hat{k}$	r	l	$L = l(2^r-1)$	ϵ
40	6	18	1134	0.0055
80	7	17	2159	0.0099
120	7	27	3429	0.0070
160	8	12	4335	0.0090
200	8	22	5610	0.0058

Tables 3 and 4 show that L grows logarithmically with n. The correlation coefficient between $\log(n)$ and L is 0.9961 in Table 3 and 0.9999 in Table 4.

Now, at least for reasonable values of $\hat{k}$, n and $\epsilon_{\max}$, we have shown that L grows linearly with $\hat{k}$ and logarithmically with n. Since k is an upper bound of $\hat{k}$, we can assert that L grows like $O(k \log n)$.

4.6. Performance: message length and decoding time

Message length depends on the modulus of the privacy homomorphism used in the second protocol (quite unrelated to the O(1)-long modulus used in the first protocol, by the way). This modulus must be large enough to contain the addition of k integer values v_i whose bitlength is $O(ql(2^r - 1))$. Since q grows like $O(\log k)$ and $l(2^r - 1)$ grows like $O(k \log n)$, we can conclude that the modulus and thus the message length grow like $O(k \log k \log n)$.

This length should be compared to the $O(n)$ length required by the simplistic alternative described in Section 1 (secure version of Protocol 0). Note that an alternative unicast system where only the k stations transmitting '1' were active is not a valid comparison, because that system would offer no confidentiality.

The above choice of q, r and l can be used to quantify the decoding time of the second protocol:

1. We consider that Steps (4a) and (4b) run in O(1) time.
2. Sequence extraction at Step (4c) takes $O(l(2^r - 1))$ time, based on the number of s'_j extracted. With the above choice for l and r, this grows like $O(k \log n)$.
3. Step (4d) does no real computation, so it takes negligible time.
4. From the structure of its loop, Step (4e) takes $O(l(2^r - 1)^2)$ time, which is less than $O(l^2(2^r - 1)^2)$. With the above choice for l and r, this can be rewritten as $O(k^2 \log^2 n)$.
5. Step (4f) takes $O(nl)$ time (for each of n leaves an l-bit long sequence is generated), which is less than $O(nl(2^r - 1))$. With the above choice for l and r, this can be rewritten as $O(kn \log n)$.
6. Step (4g) is also $O(nl)$, and thus bounded by $O(kn \log n)$.

Thus, whenever $k \ll n$, the decoding time is dominated by $O(kn \log n)$.

Note 2. Our main goal is to avoid implosion at the root node. Therefore, our primary performance metrics is message length, which has been used to choose the values of parameters q, r and l in the above sections. However, we had to make sure that decoding time stays affordable. Fortunately, this turns out to be the case, since we get time quasi-linear in n.

Table 3
Optimal r and l values for $\hat{k} = 100$ and $\epsilon_{\max} = 0.01$ depending on n

n	r	l	$L = l(2^r - 1)$	ϵ
2500	7	21	2667	0.0073
5000	7	22	2794	0.0082
10000	7	23	2921	0.0090
20000	7	24	3048	0.0099
40000	7	26	3302	0.0059

Table 4
Optimal r and l values for $\hat{k} = 200$ and $\epsilon_{\max} = 0.01$ depending on n

n	r	l	$L = l(2^r - 1)$	ϵ
2500	8	21	5355	0.0065
5000	8	22	5610	0.0073
10000	8	23	5865	0.0082
20000	8	24	6120	0.0090
40000	8	25	6375	0.0098

5. Hybrid protocol

The second protocol above features efficient bit decoding while the first protocol is good at checking for correctness. We propose a hybrid protocol combining the first and second protocols in order to get the best of each:

Protocol 3 (*Hybrid protocol*)

1. The root multicasts a transmission request containing the challenges for the first and the second protocols. That is, a random integer v and also q, r, l and a primitive polynomial $\mathbf{P}$.
2. Upon reception of the request, each leaf U_i does:
   (a) Generate a message $M_{1,i}$ using the first protocol and $M_{2,i}$ using the second protocol.
   (b) Send the tuple $(M_{1,i}, M_{2,i})$ up to its parent node.

3. Intermediate nodes aggregate received messages by aggregating their first and second components independently. The first components are aggregated according to the privacy homomorphism of the first protocol. The second components are aggregated according to the privacy homomorphism of the second protocol.
4. The root node does:
   (a) Perform the last aggregation to obtain (M_1, M_2).
   (b) Decrypt this message to get m_1 and m_2 (using the appropriate privacy homomorphisms for M_1 and M_2).
   (c) Use the decoding procedure of the second protocol on m_2 to obtain $B' = (b'_1, \ldots, b'_n)$ (that is, identify the likely subset of leaves having transmitted '1').
   (d) Check the correctness of B' against m_1 as described in Step (4d) of the first protocol.
   (e) If everything is correct, multicast an acknowledgment message to all leaves; otherwise, call the tracing protocol (Protocol 4).

Already for moderately large n, simultaneously transmitting the messages of the first and the second protocols, as done in the hybrid protocol above, does not significantly increase the message length that would be transmitted by the second protocol alone. Indeed, messages in the first protocol are of constant length O(1), whereas those in the second protocol grow like $O(k \log k \log n)$. The same holds true for the decoding time: it takes $O(kn \log n)$ time for the root to compute B' using the second protocol, whereas the additional time required to check the correctness of B' using the first protocol is only $O(n)$.

If the above hybrid protocol yields successful correctness checks for several consecutive time slots, the root can use parameters (q, r, l) for a smaller k in the next transmission request to reduce message size and thus bandwidth usage. In case of overflow, a larger k may be desirable (see below). There are many possible feed-back policies to update (q, r, l) in order to increase/decrease k. In fact, this is a problem of control theory beyond the purpose of this paper, which seeks to give a general description of the hybrid protocol rather than of specific feed-back policies.

The final correctness check in the hybrid protocol can fail for at least two reasons:

- *Protocol malfunction*. This happens with probability at most $\epsilon_{\max}$ (Lemma 4).
- *Too many '1's*. The number $\hat{k}$ of leaves having transmitted '1' is larger than k. A natural reaction in this case would be to re-run the above hybrid protocol with parameters properly chosen for a larger k.
- *Junk traffic*. A leaf or an intermediate node are injecting incorrectly formatted traffic. This can be intentional or accidental.

Upon detection of failure in the hybrid protocol, the root runs the following tracing protocol to find out where corruption is being originated. Some assumptions are needed for this tracing protocol to work:

- Each intermediate node must store incoming messages until the corresponding aggregated message is acknowledged by the root.
- Each intermediate node or leaf must have a public/private key pair (e.g., RSA keys, [8]). The corresponding public key must be accepted by all its parent nodes, including the root node.
- Each node time-stamps and signs the message it sends up to its parent. The signature is also sent. Note that this done at the link level: those signatures are not forwarded to the root node under normal operation. When an intermediate router receives a message from one of its child nodes, it must check the child's signature. If it is not correct, then the message should not be accepted.

The tracing protocol is a recursive protocol initially called as *Tracing* (*root, final aggregated message*). Its pseudo-code is as follows:

Protocol 4 (*Tracing*(*explored router*, *explored corrupted message*)). The root node does:

1. Obtain from the 'explored router' the messages and signatures it has received from its child nodes (before aggregation).
2. Check the signatures on the 'explored corrupted message' and on the messages obtained from the 'explored router'. This step is omitted when the explored router is the root node.
3. If any of the signature checks fails, mark the 'explored router' as guilty. Go to Step (6).
4. Check whether the aggregation of the obtained messages matches the 'explored corrupted

message'. If not, mark the 'explored router' as guilty. Go to Step (6).

5. Decrypt the received messages and check their correctness:
   (a) If all messages are correct, conclude that corruption was due to overflow (more than k leaves sent a '1' symbol) or protocol malfunction (this can happen with probability at most $\epsilon_{\max}$, see Lemma 4). Exit the tracing algorithm.
   (b) For corrupted messages sent by leaves, mark those leaves as guilty.
   (c) For each intermediate router for which the correctness check has failed call *Tracing* (*intermediate router*, *aggregated message sent by router*).
6. Remove any guilty nodes from the multicast tree.

6. Conclusions

Scalability is a major issue in reverse multicast communication. Security, that is, confidentiality, integrity and authentication of leaf-to-root communication, is another important issue in many applications. The few previous secure proposals are such that, for a multicast tree with n leaves, the root node receives $O(n)$ bits of information each time slot.

We have focused on binary leaf-to-root communication, in which a leaf sends a binary symbol during a specific time interval. If the probabilities of transmitting '0' and '1' are different, our proposal exploits this bias to reduce the risk of implosion by conveying only an $O(k \log k \log n)$ bit long message to the root, where k is an upper bound on the number of leaves transmitting the least likely symbol and n is the total amount of leaves. Our solution combines a protocol with quasi-linear decoding time and non-verifiable correctness with a protocol that allows fast checking of the decoded solution with low error probability: the resulting hybrid protocol offers efficient decoding and verifiable correctness.

On the security side, it is not possible for an intruder to determine the symbol transmitted by a certain leaf (confidentiality); nor is it possible to alter transmission (integrity) or inject undetected bogus data in the leaf-to-root flow (authentication).

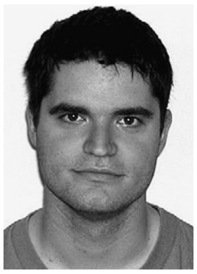
Extending the proposed solution for q-ary transmission is straightforward using the approach in [3].

Acknowledgements

The authors are partly supported by the Spanish Ministry of Science and Education through project SEG2004-04352-C04-01 "PROPRIETAS". We also acknowledge the comments of three anonymous referees, which were helpful to improve this paper.

References

[1] T. Dierks, C. Allen, The TLS Protocol. Version 1.0, Internet RFC 2246, January 1999. Available from: <http://www.ietf.org>.
[2] J. Domingo-Ferrer, A. Martínez-Ballesté, F. Sebé, Secure reverse communications in a multicast tree, in Lecture Notes in Computer Science, 3042 (2004) 807–816.
[3] A. Martínez-Ballesté, J. Domingo-Ferrer, F. Sebé, Secure many-to-one transmission of q-ary symbols, in: Lecture Notes in Computer Science, vol. 3283, 2004, pp. 85–91.
[4] T. Okamoto, S. Uchiyama, A new public-key cryptosystem as secure as factoring, in: Lecture Notes in Computer Science, vol. 1403, 1998, pp. 308–318.
[5] R.L. Peterson, R.E. Ziemer, D.E. Borth, Introduction to Spread Spectrum Communications, Prentice Hall International Editions, 1995.
[6] B. Quinn, K. Almeroth, IP Multicast Applications: Challenges and Solutions, Internet RFC 3170, September 2001. Available from: <http://www.ietf.org>.
[7] R. Rivest, The MD5 Message Digest Algorithm, Internet RFC 1321, April 1992. Available from: <http://www.ietf.org>.
[8] R.L. Rivest, A. Shamir, L. Adleman, A method for obtaining digital signatures and public-key cryptosystems, Communications of the ACM 21 (2) (1978) 120–126.
[9] 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, Catalonia. 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 25 international publications. His fields of interest are multicast security and information privacy. In 2003, He was a co-recipient (with Prof. Domingo-Ferrer) of a research prize from the Association of Telecom Engineers of Catalonia. In 2004, he was a visiting researcher at LAAS-CNRS, Toulouse, France.

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 field of activity is data security in a broad sense, from cryptographic protocols to data privacy, including network security. In 2003, he received 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 two patents and over 130 publications, one of which became an ISI highly-cited paper in early 2005. He was the co-ordinator of EU FP5 project CO-ORTHOGONAL (IST-2001-32012) and of several Spanish funded and US funded research projects. He has chaired the program committee of four international conferences and has served in the program committee of over 25 conferences on security. He is an Associate Editor of the Journal of Official Statistics and has been a Guest Editor of *Computer Networks, IJUFKS and Data Mining and Knowledge Discovery*. Prof. Domingo-Ferrer is a Senior Member of IEEE and in 2004 he was a Visiting Fellow at Princeton University.