Secure Reverse Communication in a Multicast Tree *

Josep Domingo-Ferrer, Antoni Martínez-Ballesté and Francesc Sebé

Dept. of Computer Engineering and Maths,
Universitat Rovira i Virgili,
Av. Països Catalans 26,
E-43007 Tarragona, Catalonia, Spain
e-mail {jdomingo,anmartin,fsebe}@etse.urv.es

Abstract. Multicast content delivery can be expected to become a major source of revenue with the increase of private broadband fixed and mobile communications. Several multicast applications require the receivers to securely send some real-time information back to the source, which leads to a many-to-one communication scenario. Using unicast connections to send this reverse traffic results in a data implosion which may swamp the source with incoming communication. In order to avoid this problem, a method for aggregating the information as it is being sent back to the source is presented in this paper. Confidentiality and authentication are guaranteed at the symbol level for this reverse real-time traffic.
Keywords: Cryptography, Security and privacy, Multicasting.

1 Introduction

Multicast communication is a paradigm with one source and a group of receivers. Examples of multicast communication are cable TV, Internet live transmissions, pay-per-view video on demand, etc.

A multicast system can be represented as a tree structure where the root node is the content source, the leaves are users and the intermediate nodes are multicast routers. A multicast router receives the content from its parent node and forwards it to its child nodes.

Multicast applications usually result in one-to-many communication from the source to the users. Additionally, some multicast applications may require the source of the multicast tree to collect data from all users, which results in *many-to-one* communication [Mill99]. Many-to-one applications entail inherent scaling problems. Example data that may need to be collected by the source are network monitoring information, fee collection in case of pay-per-view or pay-as-you-watch transmissions, sensor information, resource discovery, etc. Too many

* This work has been partly supported by the Spanish Ministry of Science and Technology and the European FEDER fund under project TIC2001-0633-C03-01 "STREAMOBILE".

simultaneous users sending data to the source can potentially overwhelm the latter, a situation usually known as implosion problem [Quin01]. In addition to requiring solutions to implosion, some many-to-one applications require secure and real-time transmission.

1.1 Contribution and plan of this paper

We propose in this paper a scalable and secure protocol for reverse many-to-one communication in a multicast tree. Our proposal is based on super-increasing sequences and probabilistic additive public-key privacy homomorphisms. Since our scheme is designed for real-time reverse traffic, it is assumed that information cannot be buffered but should be sent symbol by symbol as these are generated by the user. Therefore, security is provided at the symbol level and consists of communications secrecy and user authentication.

Section 2 gives some background on super-increasing sequences and probabilistic additive public-key privacy homomorphisms. Section 3 describes the mechanism we propose for many-to-one communication. Section 4 deals with the security of the proposed scheme. A performance analysis is reported in Section 5. Section 6 contains some conclusions and suggestions for future work.

2 Background concepts

The basic tools used in our proposal are described next.

2.1 Super-increasing sequences and the knapsack problem

Given a sequence of positive integers $\mathcal{S} = \{S_1, S_2, \ldots, S_{m-1}, S_m\}$ and a value T which is the sum of some elements of $\mathcal{S}$, the knapsack problem [Merk78] consists of finding a subset $\mathcal{S}' = \{S_a, S_b, \ldots, S_j\}$, of $\mathcal{S}$ whose sum equals T.

The general knapsack problem is known to be an NP-complete problem, but there are some cases in which the problem can be solved polynomially. This is the case when the sequence $\mathcal{S}$ is *super-increasing*.

A sequence of positive integers $\mathcal{S} = \{S_1, S_2, \ldots, S_{m-1}, S_m\}$ is super-increasing if every term is greater than the sum of all previous terms, *i.e.*

$$S_k > \sum_{j=1}^{k-1} S_j$$

Given a super-increasing sequence, the knapsack problem can easily be solved using the following recursive procedure:

1. Initially, all values S_i greater than T are marked as not being part of the solution.
2. Take the largest unmarked element $S_j \in \mathcal{S}$ and check whether $S_j \leq T$. If the check is positive, mark S_j as being part of the solution and let $T := T - S_j$. Otherwise mark S_j as not being part of the solution and leave T unaltered.

3. Solve the problem for the remaining unmarked elements in $\mathcal{S}$.
4. The problem is correctly solved if and only if T equals 0 when all elements in $\mathcal{S}$ have been marked.

The scheme in this paper uses super-increasing sequences to aggregate integer values in a reversible way.

2.2 Additive privacy homomorphisms

Privacy homomorphisms (PHs) are encryption transformations mapping a set of operations on cleartext to another set of operations on ciphertext. Basically, PHs are encryption functions $E : CT \rightarrow CT'$ allowing a set F' of operations on a ciphertext domain CT' to be carried out without knowledge of the decryption function D. Knowledge of D allows the result of the corresponding set F of operations on a cleartext domain CT to be retrieved. A PH is called *additive* when its set F of cleartext operations contains addition. A PH is called *probabilistic* if the encryption algorithm E involves some random mechanism that chooses the ciphertext corresponding to a given cleartext from a set of possible ciphertexts.

Privacy homomorphisms that will be used in our proposal below must be additive, probabilistic and public-key. In [John02], it is shown that additive PHs are insecure in front of known-cleartext attacks if used for signing. Thus, in our scheme the public-key additive PH is only used for encryption. We next give an example of a probabilistic additive public-key PH.

Example 1. The Okamoto-Uchiyama [Okam98] probabilistic public-key cryptosystem (OUPH) has an additive homomorphic property. This probabilistic public-key cryptosystem is proven to be as secure as the intractability of factoring $n = p^2p'$ against passive adversaries, where p and p' are two large primes.□

3 Secure many-to-one bit transmission

This section describes our proposal for secure reverse many-to-one communication in a multicast context. A basic construction is first described which allows transmission of one binary or ternary symbol. A generalization is then presented for transmission of a q-ary symbol or a block of bits.

3.1 The basic construction

The construction consists of a set-up protocol to be run before any transmissions are started, and a transmission protocol to be run for each symbol transmission.

Protocol 1 (Set-up).

1. *The source chooses parameters l, u, where l will be used below and u is the number of users.*

2. *The source generates* $2u$ *intervals as follows:*

$$\mathbf{I}_1 = [1, 2^l - 1]$$

$$\mathbf{I}_j = [I_j^{min}, I_j^{max}] = [(2^j - 2)(2^l - 1) + 2^{j-1} - 1, (2^j - 1)(2^l - 1) + 2^{j-1} - 1]$$

for $j = 2$ *to* $2u$.

3. *The source generates* u *keys* k_i*, for* $i = 1$ *to* u*, corresponding to a block cipher (*e.g. *AES).*
4. *The source generates a key pair for a probabilistic additive public-key privacy homomorphism such that its cleartext space is* $CT = \{0, 1, 2, \cdots, p-1\}$ *where* p *should be larger than* $2I_{2u}^{max}+1$*. After some manipulation, it can be checked that the lower bound on* p *is*

$$p > (2^{2u} - 1)(2^{l+1} - 1) \tag{1}$$

*(*E.g.*, for* $u = 500$ *users,* p *should have* $O(10^3)$ *bits.)*

5. *The source multicasts the public key* PK *of the PH and* I_{2i-1}^{min} *and* I_{2i}^{min} *for* $i = 1$ *to* u*. In addition the source secretly sends* k_i *to each user* U_i*, who should keep this key confidential (storing it in a tamper-resistant device such as a smart card would seem appropriate).*

After set-up, the normal operation of the scheme consists of many-to-one transmissions of binary or ternary symbols. In order to collect a binary or ternary symbol from each user, the following four-step protocol is used:

Protocol 2 (Many-to-one binary or ternary transmission).

1. Transmission request*. A challenge message is multicast by the source to all users. This challenge contains a random value* v*.*
2. Message generation*.*
   (a) *When a user* U_i *receives the challenge message, she computes*

$$S_{2i-1} = I_{2i-1}^{min} + \mathcal{H}(v||k_i)$$

$$S_{2i} = I_{2i}^{min} + \mathcal{H}(v+1||k_i) \tag{2}$$

where $\mathcal{H}$ *is a one-way collision-free hash function yielding an* l*-bit integer as output. This condition on the output of* $\mathcal{H}$ *ensures that* $S_{2i-1} \in \mathbf{I}_{2i-1}$ *and* $S_{2i} \in \mathbf{I}_{2i}$*, which in turn guarantees that the sequence* $\mathcal{S} = \{S_j\}$ *for* $j = 1, \cdots, 2u$ *is* super-increasing*. On the other hand, condition (1) ensures that no overflow in* CT *will occur when adding encrypted terms of the super-increasing sequence over the ciphertext space* CT'*.*

   (b) *Next, each* U_i *generates her message as follows: i) If she wants to transmit a* 0 *bit value, she generates the message* $M_i = E_{PK}(S_{2i-1})$ *where* $E_{PK}(\cdot)$ *stands for the encryption function of the probabilistic additive public-key privacy homomorphism used. ii) If she wants to transmit a* 1 *bit value, she generates* $M_i = E_{PK}(S_{2i})$*. iii) In case of transmitting*

ternary symbols, a third symbol (other than 0 and 1) could be transmitted if U_i sent $M_i = E_{PK}(S_{2i-1} + S_{2i})$.
Note that, since a probabilistic cryptosystem is being used, the same cleartext message can result in different encrypted messages. Finally U_i sends M_i up to her parent router.

3. Message aggregation. *Intermediate routers receive messages from their child routers/users and do the following:*
   (a) *Once all expected messages $\{M_i\}_i$ have been received, the router aggregates them as $M = \sum_i' M_i$, where $\sum'$ stands for the ciphertext operation of the privacy homomorphism corresponding to cleartext addition.*
   (b) *The router sends M up to its parent router.*
4. Symbol extraction. *When the previous process completes, the source finally receives an aggregated message M, from which the transmitted symbols are extracted as follows:*
   (a) *The source constructs the super-increasing sequence $\mathcal{S} = \{S_j\}$ for $j = 1$ to $2u$ using, for each user U_i, equations (2).*
   (b) *The source decrypts M using its private key to recover a value T which is used to solve the super-increasing knapsack problem and obtain the sequence $\mathcal{S}' = \{S_a, S_b, \ldots, S_j\}$ that yields the symbols sent by the users. Specifically, there are four possible cases for each user U_i: i) If $S_{2i-1} \notin \mathcal{S}'$ and $S_{2i} \notin \mathcal{S}'$, then U_i sent nothing. ii) If $S_{2i-1} \in \mathcal{S}'$ and $S_{2i} \notin \mathcal{S}'$, then U_i sent a bit value 0. iii) If $S_{2i-1} \notin \mathcal{S}'$ and $S_{2i} \in \mathcal{S}'$, then U_i sent a bit value 1. iv) If $S_{2i-1} \in \mathcal{S}'$ and $S_{2i} \in \mathcal{S}'$, then this is an error condition in a binary transmission. However, if using ternary symbols, this configuration can be used to send the third symbol.*

3.2 A generalization for q-ary or block transmission

The basic construction given above can be generalized as follows to accomodate transmission of q-ary symbols or blocks of bits:

1. During Protocol 1 generate and publish tu intervals $\mathbf{I}_j$ rather than $2u$ intervals. Condition (1) must be modified by replacing $2u$ with tu.
2. During Protocol 2:
   (a) Each user U_i takes t consecutive intervals $\mathbf{I}_{ti-t+1}, \cdots, \mathbf{I}_{ti}$ and generates t terms the super-increasing sequence as:

$$S_{ti-t+j} = I^{min}_{ti-t+j} + \mathcal{H}(v + j - 1||k_i)$$

   for $j = 1$ to t.
   (b) Now U_i can transmit $q = 2^t - 1$ different values by sending the encrypted sum of a subset chosen among the $2^t - 1$ non-empty subsets of $\{S_{ti-t+1}, \cdots, S_{ti}\}$ Note that the encrypted sum of the empty subset (*i.e.* $E_{PK}(0)$) cannot be used to encode a value in a secure transmission because anyone can send it (no authentication) or guess it (no confidentiality). Thus, U_i can either transmit a q-ary symbol or a block of $t - 1$ bits (the first option is clearly less wasteful).

(c) Message aggregation in the generalization stays the same as in Protocol 2.
(d) Symbol extraction by the source in the generalization must use the same mapping between subsets and q-ary symbol values used by users during message generation. This mapping assumed to be public.

4 Security

We next state the security properties of our scheme, which are proven in the Appendix.

Property 1 (Confidentiality). *If a secure probabilistic additive public-key PH is used in which there is a negligible probability of obtaining the same ciphertext as a result of two independent encryptions of the same cleartext, then an intruder cannot determine the symbol transmitted by a user in Protocol 2.*

Property 2 (Authentication). *If a secure public-key PH and a one-way collision-free hash function with l-bit output are used, the following holds:*

1. *the probability of successfully impersonating another user when sending a bit value to the source is 2^{-l};*
2. *substituting a false message M' for a legitimate message $M \neq M'$ in the current transmission is at least as difficult as impersonation;*
3. *substituting a message M' for a legitimate message $M \neq M'$ in future transmissions using information from the current transmission is infeasible.*

5 Performance

Before presenting the performance comparison below, some preliminary remarks are required:

- The performance criterion considered is the bandwidth required by the reverse traffic.
- In order to benchmark the performance of our system, we will consider an alternative system based on unicast transmissions from each user to the source. Like in our system, the unicast transmissions in the benchmark system will be symbol-wise. We assume that the communication is real-time, so that symbols are transmitted as they are generated, rather than being buffered and transmitted in batches.
- We will require that each symbol transmission in the alternative unicast system has the same security properties as transmissions in our system.
- For the sake of concreteness, we will use OUPH as a privacy homomorphism in this section.

5.1 A benchmark unicast system

In order to avoid the need of public-key encryption for a user to send a confidential and authenticated symbol, we must assume that each user U_i shares with the source a key k_i corresponding to a block cipher (*e.g.* AES).

The message M containing the symbol b will thus look like

$$M = E_{k_i}(b||ts||ck), U_i$$

where $E_{k_i}(\cdot)$ stands for the encryption function of the block cipher, ts is a time-stamp, ck is a checksum and U_i is the identity of user U_i. Integrity is ensured by ck and ts (the time-stamp prevents replacing future transmissions with past transmissions).

5.2 Comparison

When u users simultaneously send their encrypted symbols with the benchmark unicast system, $u(B + \log_2 u)$ bits are received by the source, assuming that the B is the block bitlength of the block cipher and $\log_2 u$ is the bitlength of the user identifier U_i. We assume also that the bitlength of $b||ts||ck$ is less than or equal to B. For a block cipher such as AES, at least one has $B = 128$, so the previous assumption is reasonable.

When u users send their encrypted bits/symbols with our system, all symbol transmissions are eventually aggregated into a single message

$$M = \prod_i M_i \pmod{n}$$

which is the only one reaching the source. M can be at most n, so its length is $\log_2 n$. Equivalently, the bitlength of M is

$$|M| = \log_2 n = \log_2(p^2p') = 2\log_2 p + \log_2 p' = 3\log_2 p$$

where we have used that, in OUPH, $n = p^2p'$ with $|p| = |p'|$. Now, already for a moderate number u of users, p can be chosen close to its lower bound (1) while remaining large enough for factoring of $n = p^2p'$ to be hard, as required by OUPH. Therefore, if we use the generalized bound (1) with tu instead of $2u$, we have

$$|M| \approx 3\log_2[(2^{tu} - 1)(2^{l+1} - 1)] = 3\log_2(2^{tu} - 1) + 3\log_2(2^{l+1} - 1) \quad (3)$$

It can be seen that expression (3) is dominated by $3tu$ as the number of users grows. Therefore, if the number u of users is moderate to large and if the symbol bitlength is $t < (B + \log_2 u)/3$, the bandwidth $3tu$ required by our scheme is *less* than the bandwidth $u(B + \log_2 u)$ required by the benchmark unicast system. Since typical block sizes are as large as $B = 64, 128, 192$ or 256, the previous assumption on the symbol bitlength is reasonable.

Besides, our proposal only requires one reverse incoming connection to the source, whereas the unicast alternative requires u reverse connections to the source, which calls for allocation of additional overhead bandwidth not included in the above comparison.

Note 2. The primary aim of our proposal is reverse bandwidth reduction. It must be noticed that this is achieved without increasing the computational burden at the source. Symbol extraction during Protocol 2 requires the source to build tu terms of a super-increasing sequence and to solve a super-increasing knapsack problem. The computational cost of doing this is similar to the cost of the u block decryptions required by the unicast benchmark.

6 Conclusions and example applications

The thrust behind the design of the scheme in this paper was the need to securely send real-time reverse information in multicast scenarios, that is, information whose symbols should not be buffered but be sent as they are generated. This requirement not being exclusive of multicast, our scheme can be applied whenever a large number of users or devices must communicate in real-time with a single node and there is a risk that the incoming bandwidth available at the receiving node may be a bottleneck. Example applications include:

- *Secure multicast.* The scheme presented can be used for users to securely send keepalive messages to the source, who can keep track of who is logged on. A step further is to use the scheme for real-time pay-per-view multicast: the users send payment information back to the source, and stop receiving multicast contents if they stop sending payment.
- *Secure collection of control information.* A control center securely collects periodical status information from a large number of sensors or other devices. This is similar to sending reverse traffic in a multicast scenario.

The proposed scheme uses super-increasing sequences and probabilistic additive public-key homomorphic encryption to aggregate traffic. It guarantees confidentiality and authentication of the transmitted q-ary symbols. Thanks to aggregation, the source only needs to establish one incoming connection. In the special case where the Okamoto-Uchiyama PH is used, the required incoming bandwidth at the source for u users approximates $3tu$ bits when each user securely transmits one q-ary symbol at a time, with $q = 2^t - 1$. This is not so far from the $u \log_2 q \approx tu$ bits required for *insecure* transmission of u q-ary symbols. Achieving the same security properties using unicast transmissions would typically need Bu bits split in u user-source connections, where B is the block size of a block cipher.

Appendix

Proof (Property 1): Without loss of generality and to keep the proof simple, we can restrict ourselves to the basic construction given in Section 3.1, which

corresponds to the case $t = 2$ of the general construction sketched in Section 3.2. Generalizing the proof to any t is cumbersome but straightforward. Now, assume the intruder captures a message M sent by U_i during Protocol 2. This message is either $E_{PK}(S_{2i-1})$, $E_{PK}(S_{2i})$ or $E_{PK}(S_{2i-1} + S_{2i})$. Decryption of M is not possible because the PH is secure and the intruder does not have access to the private key.

Exhaustive search of the cleartext carried out by M is the other attack strategy to be examined. Now, exhaustive search of the sequence values S_{2i-1} or S_{2i} by encrypting candidate values and comparing the result to M is not feasible because the PH is probabilistic and there is a negligible probability that two independent encryptions of the same cleartext yield the same ciphertext. Therefore, the above comparison (and thus exhaustive search) will fail with overwhelming probability. □

Proof (Property 2): By the same argument as for Property 1, we need only prove this property for the basic construction. In the impersonation attack, an intruder who wants to impersonate user U_i tries to generate a message $E_{PK}(S_{2i-1})$, $E_{PK}(S_{2i})$ or $E_{PK}(S_{2i-1} + S_{2i})$, Now, the intruder needs to compute S_{2i-1} or S_{2i}. Each term S_j of the super-increasing sequence $\mathcal{S}$ is pseudo-randomly chosen within an interval $\mathbf{I}_j$ containing 2^l integer values. The choice is made using a one-way collision-free hash function of the challenge and the secret key k_i unknown to the intruder, as shown in equations (2). Thus, the probability of the intruder randomly hitting S_j is at most 2^{-l}. Remark that exhaustive search is not feasible, since there is no way of checking whether the right S_j has been hit (there is no way for the intruder to make sure whether the message generated with the candidate S_j is correct).

A substitution attack can be mounted in the current transmission or in future transmissions:

- In the current transmission, assume the intruder wants to substitute a false message M' for an authentic message M sent by U_i, with $M' \neq M$. Without loss of generality, let $M = E_{PK}(S_{2i})$; the intruder wants to transform M into $M' = E_{PK}(S_{2i-1})$ or $M' = E_{PK}(S_{2i-1} + S_{2i})$. This requires the following steps: i) recover S_{2i} from M; ii) compute S_{2i-1} with knowledge of S_{2i}; iii) compute M'. Thus, even if decrypting M at step i) was easy (which it is not), solving step ii) is as difficult as mounting a successful impersonation attack (see above).
- A second possibility is for an internal intruder to use information derived from a current transmission of a message by U_i to alter future messages sent by U_i. But this is infeasible, because in subsequent executions of Protocol 2, a different super-increasing sequence will be used to encode the messages which does not depend on the current super-increasing sequence (see equations (2)).

□

References

[John02] R. Johnson, D. Molnar, D. Song and D. Wagner, "Homomorphic signature schemes", in *Topics in Cryptology - CT-RSA 2002*, ed. B. Preneel, LNCS 2271, Berlin: Springer Verlag, pp. 244-262, 2002.

[Merk78] R. C. Merkle and M. Hellman, "Hiding information and signatures in trapdoor knapsacks", in *IEEE Transactions on Information Theory*, vol. 24, no. 5, pp. 525-530, 1978.

[Mill99] C. K. Miller, *Multicast Newtorking and Applications*. Reading MA: Addison Wesley, 1999.

[MSEC03] Multicast Security Working Group (MSEC WG). `http://www.securemulticast.org`

[Okam98] T. Okamoto and S. Uchiyama, "A new public-key cryptosystem as secure as factoring", in *Advances in Cryptology - EUROCRYPT'98*, ed. K. Nyberg, LNCS 1403, Berlin: Springer-Verlag, pp. 308-318, 1998.

[Quin01] B. Quinn and K. Almeroth, "IP multicast applications: challenges and solutions", Internet RFC 3170, Sept. 2001. http://www.ietf.org.