Secure many-to-one transmission of q-ary symbols [⋆]

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

Universitat Rovira i Virgili - Tarragona, Catalonia, Spain
e-mail {`anmartin,fsebe,jdomingo`}`@etse.urv.es`

Abstract. There is a number of applications requiring a community of many senders to transmit some real-time information to a single receiver. Using unicast connections to send this traffic can result in data implosion and swamp the receiver. In [Domi04], a mechanism was proposed for secure bit transmission in large-scale many-to-one communications; we propose here an extension for securely sending q-ary symbols.
Keywords: Multicasting, Active network application, Network security.

1 Introduction

Several applications require a large group of senders to transmit some real-time information to a single receiver (many-to-one communication). A network of sensors sending status information to a control center is one example of such applications.

Many-to-one communication entails inherent scaling problems. Too many simultaneous senders transmitting data to the receiver may overwhelm or swamp the latter, a problem usually known as implosion. In addition to requiring solutions to implosion, some many-to-one applications require secure and real-time transmission. Security usually means that transmission from each sender to the receiver should be confidential and authentic.

1.1 Previous work

The best way to avoid the implosion problem is that intermediate nodes aggregate the information sent from the large set of senders to the unique

[⋆] 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".

receiver. A few contributions about aggregation of data streams in many-to-one communications can be found in the literature. In [Wolf03], a technique called aggregated hierarchical multicast is presented, whereby packets are aggregated in multicast nodes. This technique, similar to Concast is introduced as an aggregation mechanism that basically suppresses duplicate packets. It must be noticed that large data packets can be output from inner nodes, depending on the information sent and the number of senders attached to a node. However, the network layer seems to be the natural place to carry out the aggregation of information. According to this, as stated in [Wolf03], the aggregation operation of data packets inside the network requires the support of the network infrastructure in terms of processing resources. The scheme described in our paper also requires the support of an *active network* [Psou99].

1.2 Contribution and plan of this paper

In this paper, a scalable protocol for many-to-one communication is presented. The scheme proposed 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. From now on, the *senders* are the leaves in the routing tree, whereas the final *receiver* is the root of the tree. Our aim is to dramatically reduce the number of connections to the receiver, sending securely fixed-length aggregated packets.

The operation of the protocol can be summarized as follows: i) in order to receive the symbols from the senders U_i, a challenge message is multicast by the receiver to all senders, via the routing tree; ii) routers in the tree aggregate encoded messages M_i received from their child nodes/senders and send aggregated information up to their parent nodes; iii) the active node closest to the receiver, produces a final message containing all symbols σ_i transmitted by the senders. iv) the receiver is finally able to decode the aggregated symbols from the final message. In practice, a mapping between the application-level language and the symbol-level language is likely to be used, whereby sending a single word in the application-level language may require sending two or more symbols. This high-level mapping is out of the scope of this paper. The proposed protocols are described in detail in Section 2. Section 3 deals with the security of the proposed scheme, whereas performance issues are examined in Section 4. Finally, Section 5 contains some conclusions.

2 Secure aggregated symbol transmission

Our proposal is based on super-increasing sequences [Merk78] and probabilistic additive public-key privacy homomorphisms (PH, [Okam98]). The knapsack problem over a superincreasing sequence is used for symbol extraction from the aggregated message. On the other hand, *privacy homomorphisms* (PHs) are encryption transformations mapping a set of operations on cleartext to another set of operations on ciphertext. A PH is called *additive* when its set of cleartext operations contains addition. A PH is called *probabilistic* if the encryption algorithm 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. The Okamoto-Uchiyama [Okam98] probabilistic public-key cryptosystem (OUPH) has an additive homomorphic property.

2.1 Construction

Protocol 1 (Set-up).

1. The receiver chooses parameters l, u, where l will be used below and u is the number of senders. Let t be the bit length of each symbol to be transmitted.
2. The receiver computes tu intervals as follows:
$$\mathbf{I}_j = [I_j^{min}, I_j^{max}] = [(2^j - 2)2^l - 2^{j-1} + 2, (2^j - 1)2^l - 2^{j-1} + 1]$$
for $j = 1$ to tu. Each sender is assigned t intervals among the above; specifically $\mathbf{I}_{(i-1)t+1}$ to $\mathbf{I}_{it}$ correspond to the i-th sender.
3. The receiver generates a secret value k_i for each sender i, for $i = 1$ to u.
4. The receiver 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 $2I_{tu}^{max}$. After some manipulation, it can be checked that the lower bound on p is
$$p > (2^{tu} - 1)2^{l+1} - 2^{tu} + 2 \tag{1}$$
5. The receiver multicasts the public key PK of the PH and I_j^{min} for j=1 to tu. In addition, the receiver secretly sends k_i to each sender U_i, who should keep it confidential (storing it in a tamper-resistant device such as a smart card would seem appropriate).

Protocol 2 (Real-time symbol transmission).

1. *Transmission request.* A challenge message is multicast by the receiver to all senders. This challenge contains a random value v.
2. *Message generation.*
   (a) When a sender U_i receives the challenge message, she computes her own t values:

$$S_{ti-t+j} = I^{min}_{ti-t+j} + \mathcal{H}(v + j - 1 || k_i) \qquad (2)$$

   for j=1 to t 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 $S_{ti-t+j} \in \mathbf{I}_{ti-t+j}$, which in turn guarantees that the *entire sequence* $\mathcal{S} = \{S_j\}$ for $j = 1, \cdots, tu$ is *super-increasing.* Note that, since v and the parameters in Protocol 1 were chosen by the receiver, the latter can readily compute the subset of $\mathcal{S}$ corresponding to any sender. 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'. Now, U_i can transmit $2^t - 1$ different symbols by sending the encrypted sum of a subset chosen among the $2^t - 1$ non-empty subsets of $\{S_{ti-t+1}, \cdots, S_{ti}\}$[1].For instance, if the symbol σ is mapped to the sum of values S_{ti-t+1} and S_{ti-t+3}, sender U_i computes the following message:

$$M_i = E_{PK}(S_{ti-t+1} + S_{ti-t+3})$$

   where E_{PK} stands for the encryption function of the probabilistic additive public-key privacy homomorphism used.
   (b) Finally U_i sends M_i up to her parent node. The size of M_i is discussed in Section 4.
3. *Message aggregation.* Intermediate nodes receive messages from their child nodes/senders and do the following:
   (a) Once all expected messages $\{M_i\}_i$ have been received, the node 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 node sends M up to its parent node. The size of M is discussed in Section 4.

[1] Note that the encrypted sum of the empty subset cannot be used to encode a value because anyone can send it or guess it.

4. *Symbol extraction.* When the previous process completes, the receiver finally receives an aggregated message M, from which the transmitted symbols are extracted as follows:
   (a) The receiver constructs the entire super-increasing sequence $\mathcal{S} = \{S_j\}$ for $j = 1$ to tu using, for each sender U_i, Equation (2).
   (b) The receiver decrypts M using its private key of the PH to recover a value T which is used to solve the super-increasing knapsack problem and obtain the sequence $\mathcal{S}' = \{S_1, S_2, \ldots\}$ that yields the values sent by the senders. From these values, the symbol σ_i sent by every sender U_i is easily retrieved, solving the knapsack problem [Merk78].

3 Security

A basic assumption when analyzing security is correctness in protocol execution, *i.e.* that Protocol 2 is followed by all senders without deviations. If one or more senders deviate, symbol extraction at the reception might fail. Correctness in execution can be enforced if senders are forced to using a computing device trusted by the receiver (*e.g.* a smart card). The receiver can use Protocol 1 to force senders, by refusing to give the secret keys k_i to anyone except sender smart cards issued or trusted by the receiver.

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 sender in Protocol 2.

Proof: Now, assume that the intruder captures a message M sent by U_i during Protocol 2. 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 will not be succeed, because of the assumption on PH. $\square$

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: i) the probability of successfully impersonating another sender when sending a bit value to the receiver is 2^{-l}; ii) substituting a false message M' for a legitimate message $M \neq M'$ in the current transmission is at least as difficult as impersonation; iii) substituting a message M' for a legitimate message $M \neq M'$ in future transmissions using information from the current transmission is infeasible.

Proof: In the impersonation attack, an intruder who wants to impersonate a sender, needs to guess at least one of the values from $\mathcal{S}$ assigned to the sender so as to generate a *valid* message. These symbols are computed using a one-way collision-free hash function (see Equation (2)), thus the probability of the intruder randomly hitting the correct symbol is at most 2^{-l}.

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 $M = E_{PK}(S_a)$; for example, the intruder wants to transform M into the encrypted sum of any of the other values from $\mathcal{S}$ assigned to U_i. This requires the following steps: i) recover S_a from M, which is difficult; ii) compute any other symbol with knowledge of S_a, which is as difficult as mounting a successful impersonation attack (see above); iii) compute M'.
- 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 Equation (2)). $\square$

4 Performance

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

- The performance criterion considered is the bandwidth required by the aggregated traffic.
- We will consider an alternative system based on unicast transmissions from each sender to the receiver. 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.
- We will require that each symbol transmission 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.

In order to avoid the need for public-key encryption for a sender to send a confidential and authenticated symbol, we must assume that each sender U_i shares with the receiver a key k_i corresponding to a block cipher (*e.g.* AES). The message M containing the symbol σ will thus look like

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

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 sender U_i. Integrity is ensured by ck and ts (the time-stamp prevents replacing future transmissions with past transmissions).

When u senders simultaneously send their encrypted symbols with the benchmark unicast system, $u(B + \log_2 u)$ bits are received by the receiver, assuming that B is the block bitlength of the block cipher and $\log_2 u$ is the bitlength of the sender identifier U_i. We assume also that the bitlength of $\sigma||ts||ck$ 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 senders send their encrypted 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 receiver. 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 senders, p can be chosen close to its lower bound (1) while remaining large enough for factoring of $n = p^2p'$ to stay hard, as required by OUPH. Therefore, if we use the generalized bound (1) we have

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

It can be seen that Expression (3) is dominated by $3tu$ as the number of senders grows. Therefore, if the number u of senders 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 incoming connection to the receiver, whereas the unicast alternative requires u connections to the receiver, which calls for allocation of additional overhead bandwidth not included in the above comparison. Finally, it must be noticed that bandwidth reduction is achieved without increasing the computational burden at the receiver. Symbol extraction during Protocol 2 requires the receiver 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.

5 Conclusions

The thrust behind the design of the scheme in this paper is the need for large-scale secure real-time many-to-one communication, that is, transmission of information whose symbols should not be buffered but be securely sent as they are generated. Our scheme can be applied whenever a large number of sending 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. In the special case where the Okamoto-Uchiyama PH is used, the required incoming bandwidth at the receiver for u senders approximates $3tu$ bits when each sender 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 sender-receiver connections, where B is the block size of a block cipher.

References

[Domi04] J. Domingo-Ferrer, A. Martínez-Ballesté, F. Sebé, "Secure reverse communication in a multicast tree", *3rd IFIP-TC6 Networking*, LNCS 3042, pp.807-816, Springer-Verlag, 2004.

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

[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.

[Psou99] K. Psounis, "Active networks: Applications, security, safety and architectures", *IEEE Communication Surveys*, vol. 2, no. 1, pp. 1-16, 1999.

[Wolf03] T. Wolf and S. Y. Choi, “Aggregated hierarchical multicast - A many-to-many communication paradigm using programmable networks”, *IEEE Transactions on Systems, Man and Cybernetics - C*, vol. 33, no. 3, pp. 358-369, Aug. 2003.