Split and Merge - an algorithm to implement security on the Internet

JOÃO PAULO PIMENTÃO, PEDRO A. C. SOUSA, ADOLFO STEIGER GARÇÃO
UNINOVA — Centre for Intelligent Robotics
Universidade Nova de Lisboa
Quinta da Torre — 2825-114 Caparica
PORTUGAL

Abstract: - The advent of new technologies will enhance the power of computing beyond imaginable scenarios (e.g. quantum computers), secure communication based on the present public key system is bound to be at risk within a short period of time.

In fact, in September 1999, RSA Laboratories reported that a team of researchers was able to determine the two prime numbers used to generate a single 512-bit RSA key [1]. In sequence, RSA warned companies to use 768 bit keys as the minimum requirement for achieving reliable security. It was only in February of 1998 that a 56-bit Data Encryption Standard (DES) key was cracked in 39 days [2].
In May of that 1999 Adi Shamir, one of the developers of the RSA algorithm introduced the project "Twinkle" that will deliver a machine capable of performing prime number generation (the core of the RSA algorithm) in speeds not yet achieved [3].

In this scenario, relying strictly on the lack of computational power for protecting information seems to be a battle that will be compromised. Nevertheless some people still argue that as the power of computing increases, so does the power to create more complex algorithms (such as increasing the number of bits in an RSA key) [4].

In our view, the security of a message should not rely on the ability to get a key, and therefore be able to decipher the message, but on the ability to get the message.

The system presented combines some of the up to date techniques of ciphering, with a scheme of splitting the message in parts and sending it to different destinations through different routes.

The power of public key scheme is used on the ciphering of the message, but the message is then split into a random number of parts and sent, via Internet to the destination.

Each of the nodes that receive a message acts in a similar way, thus creating a web of message fragments traveling the Internet through a myriad of routes which, in our opinion will make the process of getting hold of the entire message, virtually impossible.

This paper presents details the algorithms used in splitting, ciphering, deciphering and merging of the message and it introduces the concepts of neighbor and trusted hosts.

Key-Words: - Cryptography, Communication, Security, RSA, Internet, Distributed system.

1 Introduction

The power of Public Key cipher is based on the fact that the time to generate the equivalent keys needed to decipher the message is too large with current CPU speed, even when "super-computers" are considered. This, combined with the life expectancy of the messages would not allow enough time to enable message deciphering.

On the other end, in the near future, the power of computing will continue to be enhanced by several orders of magnitude, which will render today's "impossible" tasks into easily solved problems.

Most of the new developments in cipher methods are based on the use of more complex algorithms that hopefully will render key cracking into difficult tasks.

Since there is no expectation in the creation of an unbreakable algorithm the followed approach relies on denying perpetrators the access to essential parts of the message, which will prevent deciphering.

The basic idea is to split the message in parts and to be able to send the parts through a web of connections, from sender to receiver that will change with every new message.

Given the RSA algorithm, the deciphering of the message is based on the knowledge of the key, and the possession of the message. With the proposed process, even if the private key of the destination is known, deciphering will not be possible, because, the perpetrator will only have access to parts of the message and, therefore, the message will not be decipherable. In summary, having a set of parts of a

message with some holes in it will render deciphering virtually impossible.

The base idea proposed is to use RSA as an end-to-end algorithm, split the message into parts and send it through different ways. In fact, if enough breaking of the message is achieved (e.g. splitting until each letter is a fragment) the chance of being able to get the whole message out of a set of fragments is very slim.

2 The security neighborhood

One of the problems faced was on the sending and receiving ends of the message transmitting process. Even if, in the middle of the communication, one was able to send different parts through different routes, the sending and receiving ends would generally have only one point of contact through which the whole message would (sooner or later) pass.

To solve this problem, the concept of security neighborhood was introduced. The security neighborhood is represented by the set of neighbors, inside a security border (e.g. the facilities at the location of the sender) that are used at the first stage (and the last stage at the receiving end) of the message transmitting process.

With this neighborhood, any message leaving a facility will be seen (by someone looking at the communication traffic) as messages coming from a random set of sources (inside the border) to a random set of destinations; at the receiving end, messages will be seen as coming from a set of machines, outside the border, to a set of machines inside the border.

When we consider the number of messages flowing in the day-by-day traffic, the ability to select the correct fragments of a given message is strongly compromised.

3 Message Format and Parameters

The message security is achievable, with the proposed algorithm, by the tuning of two parameters: the number partitions and the number of hops that each message fragment has to go through before reaching the destination.

Based on the principle that each fragment of the message is sent to a different computer, the larger the fragmentation, the lesser the probability that the perpetrator can get hold of the whole message.

When the number of hops increases, the process will be more difficult to track, since fragments of the message will probably follow more and more distinct tracks, in order to create a web of tiny fragments of the original message traveling around the Internet.

2.1 **Increasing** security

It is usual that when one wants added security, the time it takes from the instant the message is created to the instant the message is read increases. It is so with the RSA algorithm that when security is at stake the solution is to raise the number of bits in the key; this will increase the amount of time needed both for ciphering and deciphering the message.

The proposed algorithm is no exception, increasing the message fragmentation or increasing the number of hops the message has to go through will necessarily lead to an increase of the time it takes until the message is read by the destination.

2.1 Denying information

The use of two types of parameter associated with the message seems to put too much information on the message itself. Although the number of hops is needed to determine when to send the fragment of message to the final destination, the number of times a message is split does not need to travel with the message.

In fact, the decision was to let each node define, via a random number, the number of times the message is split at that node. The use of random numbers at each node will render more complex the process of determining the number of bits flowing around. On the other end, the header of each fragment of the message will only contain information pertaining to that particular fragment of message and will not supply information regarding previous parts of the message.

When a fragment of message leaves a node in the system, going to another node, the format of the packet containing the fragment of message is the one presented in Fig. 1.

Besides the standard IP header that is used in the routing of the message and, therefore, sent in clear text, the packet contains:

- **the session key**, randomly generated at the node sending the packet, that is ciphered with the public key of the node that this packet is heading to,
- **the message packet header**, whose contents will be presented in the following paragraphs, ciphered with the session key, and

- **the fragment of the original message** ciphered both with the public key of the final destination and with the session key.

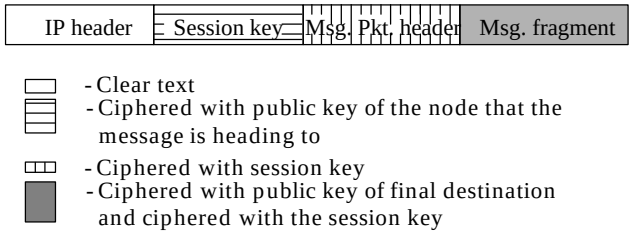

Fig. 1 - Format of a packet containing a fragment of the original message.

The message packet is composed of an header (message packet header) that is ciphered with the session key created expressly for this transfer, and a "Data" segment - message fragment - (which was originally ciphered with the public key of the final destination) ciphered with the session key.

Fig. 2 below details the format of the message packet header.

The message packet header is composed by:

- · **Header Size**: one byte that defines the size of the header block (up to 255 bytes);
- **Message Id**: the unique Id of the message as generated by the sender;
- **First byte**: the order number of the first byte of the data block that this packet contains, in the original message;
- **Last byte**: A Boolean value indicating if this block contains the last byte of the message;
- **Hops**: the number of nodes that this fragment of message still has to go through before being delivered to the final destination;
- **Source IP** and **Destination IP**: the IP addresses of the original sender of the message and of the final destination of the message, respectively;
- **D Size**: the number of bytes in the data fragment of the message.

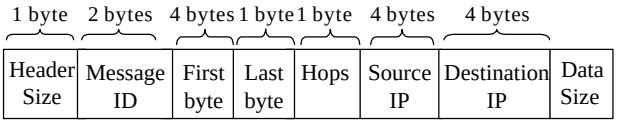

Fig. 2 - Format of the message packet header of a packet containing a fragment of the original message.

Some concern with the information that is made available at each node so, that even if a node is compromised, the limited information contained in (or passing through) a node will not (normally) supply information about the size of the message. An exception exists in the packet that contains the last byte of the original message.

4 Message splitting and reassembling

The purpose of this section is to give an analysis of what happens at the nodes regarding the determination of the number of hops, splitting and, later, at the destination, the reassembly of the message.

At each node, after validation of the number of hops that the message still has to flow through (field "Number of hops to go" in Fig. 3), the message is split into a random number of parts of random dimension.

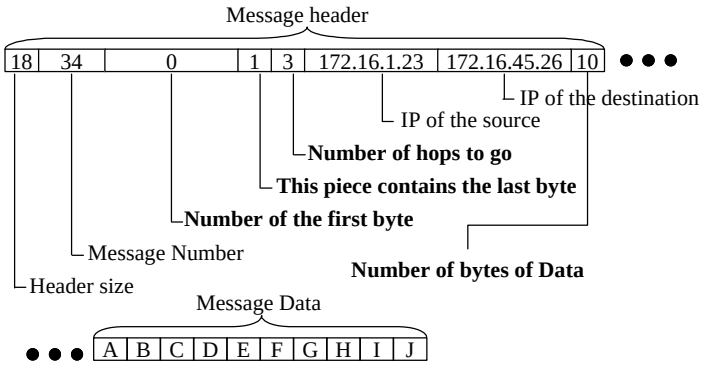

Fig. 3 - Message splitting process: the whole message.

In the resulting messages (Fig. 4), the field specifying the number of hops is decreased by one and the fields designating the number of the first data byte, the end of message marker and the field specifying the number of bytes in the data section of the message are updated. The header size is then update to reflect an eventual decrease in the number of bytes used.

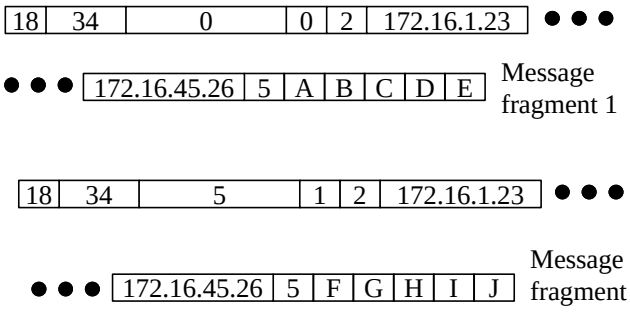

Fig. 4 - The result of splitting the message of Fig. 3 in two parts.

The use of the "Number of the first byte" field and of the "This fragment contains the last byte" field provides an added security since only when in possession of the last fragment of the message can one determine its full size. Since no coding is used to determine the degree of fragmentation of the message, the information about the number of fragments flowing around is scattered all over the nodes.

In Fig. 5 an example is presented of the re-fragmentation of the fragment 1 of the message, as it goes through another node in the system.

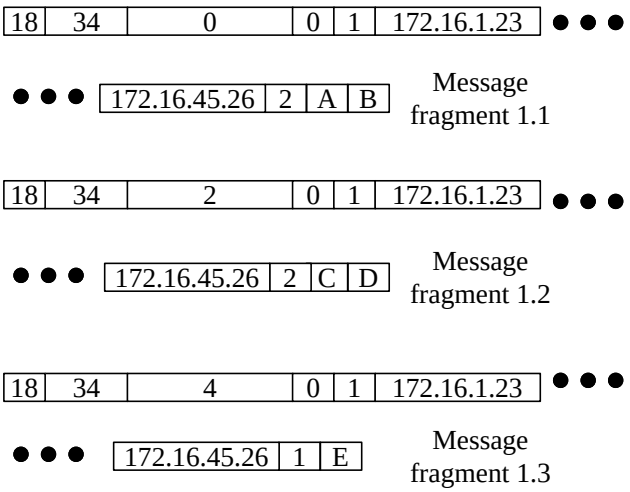

Fig. 5 - Example of second level of fragmentation.

The process of message reassembling only takes place at the destination node.

The destination node maintains a list of incoming messages fragments that is indexed by the sender's IP and sender's unique message ID (Fig. 6).

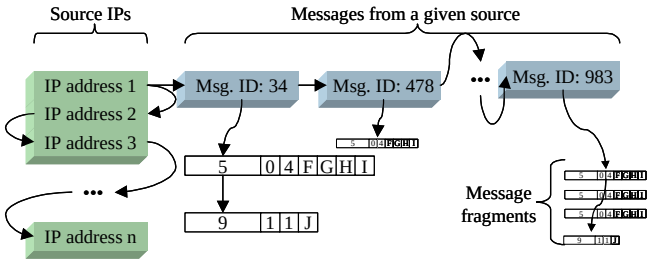

Fig. 6 - Reassembly of messages at the destination.

Every time a message fragment arrives at the final destination, the source IP address is looked up and, if it does not exist in the list of "Source IPs", it is added to it. Once the correct IP is found on the "Source IPs" list, the message number of the fragment is located in the list of messages already present.

If the message ID exists the pertinent information of the received message packet (Start byte, Last Byte, Size of Data, and Data) are added to the list of message fragments of that message, while preserving the list ordered by the "First Byte" field.

Only when all the fragments of the message are reassembled, can the message be deciphered by using the destination node's private key and delivered to the application it is destined to.

5 Hops, certificates and neighbors

As explained before in the document, the process of adding security to the message transfer is based on the number of nodes that the message has to pass through (hops) until it is delivered to the destination address. Increasing this number of nodes increases the fragmentation of the message, but it also increases the delay on the message transmission process.

At each node, the message, after being split is sent to a group of trusted nodes. The definition of the set of trusted nodes is the responsibility of each participating node.

The process will continue until the number of hops reaches one. When this happens, the message needs to be delivered to the destination.

In order to avoid the already stated problem of having the whole message seen passing through a given host, it was decided to establish a neighborhood of security around each node (namely at the facilities where the destination node is located) so that, when crossing the security border, the message fragments are seen as a set of messages coming from a group of computers on the Internet, to another group of computers inside the security border (Figure 7).

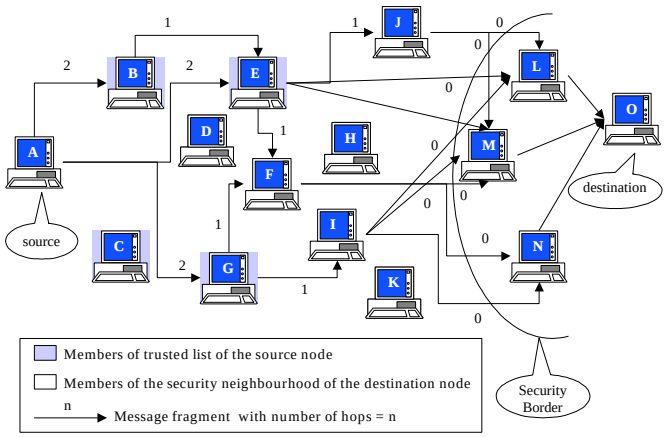

Figure 7 - Fragments, number of hops, trusted list and security neighborhood.

In Figure 7, consider that node A (source) intends to send a message to node O (destination). At this point the message is split into a random number of

parts, the list of trusted nodes of A is scanned and each fragment of the message is sent to a node on this trusted list.

Each of these receiving nodes looks at the number of hops and, until it reaches one, the message is sent to the trusted neighbors of the current node. When a node (e.g. I) receives a message fragment with the number of hops equal to one, it forwards the message to a set of nodes inside the border of security of the destination node O.

When the message fragment reaches a node with the number of hops equal to zero, it means that this node is inside the security border of the destination node. The message is not re-split at this time; it is just sent to the final destination.

The public keys of each node are kept in a PKI (Public Key Infrastructure). This PKI has been redesigned so that when requested, it would be able to supply a certificate containing the public keys of the nodes involved in the process, and also a list of the security neighbors of each node.

6 Security mechanisms

It has been said that in end-to-end communication RSA is used to cipher the message; nevertheless, the information that accompanies each part of the message (primary source, primary destination and part number) is too valuable to be sent in clear. In the beginning RSA was used in ciphering each part of the message, since it is one of the most secure methods available; however the process of ciphering and deciphering is too slow and it would create very large delays in the message transmission [5].

The solution encountered is the use of mixed key algorithms. In these algorithms a session key is randomly generated at the source; the message fragment is ciphered with this key and the key itself is ciphered with the public key of the destination; the ciphered key and the message fragment are then sent to the destination that will use its private key to reverse the process.

It should be therefore clear that each fragment of the message traveling around the network is ciphered with two keys; the public key of the destination and the session key randomly created for that fragment.

7 Conclusions and further work

This paper presents a method for the secure transfer of messages between computers using the Internet.

The chosen strategy is based on denying the access of the perpetrators to the message, in contrast with the traditional approaches of ciphering the message.

The work is still in progress and some further refinements will have to be made.

So far a prototype has been implemented as a proof of concept [6]. In this prototype the communication between nodes is based on the TCP/IP protocol [7]. The overhead presented by TCP, which guarantees safety on node-to-node communication, does not increase the guarantee of delivery of the total message, which will be compromised if a node breaks down.

It seems more adequate to use the UDP protocol and to implement an overall method of guaranteeing delivery that encompasses the whole message flow.

The results achieved in our lab tests are very promising and steps are being followed in order to develop the UDP implementation and to include new security mechanisms related with message partitioning and distribution.

References:

[1] Martyn Williams. E-Commerce Encryption Cracked; *Newsbytes, Daily News*, http://www.computeruser.com/newstoday/99/09/28/news3.html; September 28, 1999.

[2] Jim Kerstetter. RSA's encryption challenge solved in 39 days; *PC Week Online*, http://www.zdnet.com/zdnn/content/pcwo/0226/288730.html; February 28 1998.

[3] RSA Labs. Questions and Answers: Shamir's Factoring Device and RSA, http://www.rsasecurity.com/rsalabs/bulletins/twinkle_qa.html; 1999.

[4] Tanenbaum, A. *Computer Networks*, Prentice-Hall, 1996.

[5] Garfinkel and Simmson. *Web Security & Commerce*, O'Reilly, 1997

[6] Pimentão, J. *Sistema Split and Merge - Relatório de implementação*, UNINOVA, CRI-TR-03-00, 2000. (in Portuguese)

[7] Comer, D.; Stevens, D.. *Internetworking with TCP/IP Volume III: Client-Server Programming and Applications*, Prentice-Hall, 1993