Internet Security

Research paper

Topic:
The use of RSA within ESP and AH

Author: Ralf Baier

Table of contents

1 Introduction

At first, a definition of ESP and AH has to be given. ESP stands for Encapsulation Security Payload and AH is the abbreviation of Authentication Header. Both are Protocols of the IPSec protocol family which enhance the common IP protocol by a couple of security features. ESP and AH, topic of this research paper, provide three different security goals:

- **Data integrity:**
  Data integrity means, that the data of a packet hasn't been manipulated in some kind of way, while it's way from sender to receiver. The packet, the receiver got is exactly the packet, the sender has sent out.
- **Data authentication:**
  Data authentication means, that the receiver of a message can verify the identity of the sender. He can verify, if the packet has really been sent out by the person he believes.
- **Data confidentiality:**
  Confidentiality takes care about eavesdropping. By using encryption and decryption algorithms, sender and receiver can be sure that nobody can read the data they send to each other.

Data Integrity and Data Authentication can be done by AH and ESP, while data confidentiality is only implemented in the ESP protocol.
This research paper only cares about data integrity and data authentication. Data confidentiality can be done in its current form without the need of RSA.

2 Data authentication / integrity in AH and ESP

First one should think about the things, authentication and integrity have in common. Now, it's almost the same, at least it can be done by the same algorithms and in just one step which verifies authentication and integrity simultaneously. If someone gets a message, and wants to check, if the packet is authentic, he automatically will check the data integrity because to verify the integrity of a data packet, a method is needed to make verification possible. This method has to be known by both communication partners. Let us consider a classical sender - receiver scenario. Let the name of the sender be Bob and the receivers name Alice.
In the current AH and ESP implementations (see RFC 2407, 2857, 3566) this is be done by a so called HMAC[1]. The principle is very simple. Bob and Alice share a secret key k. Nobody else than Bob and Alice knows k. The knowledge of k gives them the opportunity to do data authentication and data integrity. Before sending a packet, bob puts the secret key k, together with the message, he wants to

[1] Hash Based Message Authentication Code

transmit, into an hash algorithm like SHA1[2] or MD5[3]. With this step, he creates a hash value that represents something like a fingerprint for this combination of k and the message. He puts the message into an AH or ESP packet and writes the hash value into the corresponding header. Some additional parameters like the used hash algorithm are also putted in.

When Alice now receives the packet, she does the same steps, Bob did. She puts k and the message she got, into the specified hash Algorithm and also calculates the hash value of that key – message combination. Then she compares it to the hash value she read out of the data packet. If the two values match, she can be sure, that the message is from Bob and hasn't been modified in any kind of way after Bob sent it.

The security is based on the design of hash functions. It is characteristic, that nobody can calculate the input from the output, so it is not possible to find k when a hash value is known, even then, when the message is also known what mostly will be the case. This creates the data authentication feature, because only Bob and Alice can produce the same hash values. The data integrity is ensured by the fact that even a little change in the message would create a completely different hash value which would not match the hash value that Alice calculates.

3 Using HMAC in group traffic

In group traffic like IP Multicast, there are some problems in using HMAC. This is due to the fact, that everybody in the group has to know the secret key k. Let be Eve another person, joining our communication. We now have three communication partners and can now call it group traffic. The members of the group can now send messages to each other, always protected with a HMAC, produced by using the shared secret key k. Somebody in the group who receives a data packet can do the same steps to verify the message, like it has been done in chapter 2. If he verifies the message correctly, he can be sure, that the message is from a group member. An outstanding attacker can not spoof the identity of somebody in the group because of not knowing k.

In the group itself, the unique authentication is not given. Every group member knows k, so for example Eve could spoof the identity of Bob, she just has to produce a IPSec packet with Bobs IP address and a valid group HMAC. This packet she sends to Alice who verifies it as correct and because of the IP address she assigns it to Bob. As long as everybody in a group uses the same secret key, the authentication of a data packet is not guaranteed.

4 Asymmetric authentication systems

The problem of inner group data authentication can be solved by using asymmetric algorithms for authentication. Every member in the group creates a pair of keys, a secret key e and a public key d.

[2] Secure Hash Algorithm 1
[3] Message Digest 5

The private key is kept secret to the person who created the key pair, while the public key is shared to every other member of the group.
The part of the sender is called signing and is done as follows:

1. Calculate a hash value out of the message which to be sent.
2. Encrypt the hash value with the private key.
3. Put hash value, message and details about the used algorithms into the data packet.
4. Send the packet to the receiver(s).

The encrypted hash value is also called “digital signature”.

After getting the data packet, the receiver has to do the following steps:

1. Read out the algorithm details, the message and the encrypted hash value
2. Use the public key of the sender to decrypt the received encrypted hash value
3. Calculate the hash value of the message and compare it to the hash value calculated in step 2. If the hashes match, the message is authenticated successfully, otherwise it will be rejected.

Because only the private key can be used to create a message that is assigned to a certain person, even identity spoofing by group members is not possible any more, as long as the private key is only known by the group member that is the original owner of the key.

Unfortunately IPSec currently only defines symmetric authentication algorithms like HMAC. Due to this, Brian Weiss of Cisco systems has suggested to use RSA as asymmetric authentication algorithm in his internet draft “The Use of RSA Signatures within ESP and AH” written in December 2003.

5 The RSA Algorithm

5.1 Reasons for using RSA as asymmetric encryption algorithm

There are a couple of asymmetric encryption algorithms out there, the most widely used one is the RSA algorithm. The algorithm has been developed by Ron Rivest, Adi Shamir and Leonard Adleman in 1977. It’s security is based on the fact, that no efficient algorithms are known, to factorize the product of two large integers in a acceptable timeframe.
Some reasons for its popularity are, that the algorithm is very robust and well researched by mathematicians all over the world, it can be implemented for free, because there are no intellectual property claims any more. The patent expired on 20th September 2000. The algorithm is commonly supported in

hardware, and the signature verification is relatively efficient compared to other asymmetric encryption algorithms.

5.2 Principle of RSA

Key generation process

Choose two large primes randomly:	p, q
Calculate the modulus:	$n = p \cdot q$
Calculate Eulers Phi function:	$\varphi(n) = (p-1)\cdot(q-1)$
Choose d randomly with:	$\gcd(d, \varphi(n)) = 1$
Calculate the inverse e of d	$e = [d]^{-1} \bmod \varphi(n)$
Public Key:	(d, n)
Private Key:	(e, n)

Encryption Decryption

Let be h the hash value to encrypt :

Encryption:	$c = h^e \bmod n$
Decryption:	$c = h^d \bmod n$

5.3 Performance considerations

As mentioned in chapter 5.2, RSA uses large primes and with that, it generally has to calculate with large numbers. Because of the fact, that general purpose processors have limited register sizes, they have to simulate big integer arithmetic by software. This makes RSA encryption slow. Even in specialized hardware it is relatively slow, compared to the symmetric HMAC algorithms. This can make RSA authentication within AH or ESP to a killer for applications that have high requirements in bandwidth.
Digital RSA signatures also need more space in an IPSec packet than HMAC´s do, this can cause more packet fragmentations, what also can influence the performance of a network in a negative way.

Fortunately the progress in computer technology provides faster processors from year to year, so that RSA encryption sometime will be such fast, that the caused performance drawdown can be ignored. Today, hardware accelerators for RSA encryption are available if needed.

For today's networks, the following assumption can be made: The RSA authentication method is best suited for networks where the sender has substantial amount of processing power whereas the receivers don't have and for networks where traffic is small enough, that the additional authentication tag (digital signature) does not cause packet fragmentations.

5.4 Performance optimization

RSA encryption and decryption is very costly in terms of processing time. Due to the performance drawdown, one can think about possibilities to speed up the computation.

A general data connection will usually last for hours perhaps days or more. The usual use of RSA is the encryption or signing of digital documents which have to be secure over many years. For that reason, a modulus size of 1024 bit is recommended nowadays.
A bigger modulus size is more secure because the time to factorize the prime factors p and q out of the modulus n is much higher. But a larger modulus causes slower computation.
When considering a usual data connection, we recognize, that it lasts in average for hour's maybe days. Mostly the timeframe is much smaller than years. Based on this assumption, a smaller modulus can be used for data connections. The use of those "weaker" keys is absolutely legitimate, as long as the size of the modulus is large enough to prevent an possible attacker from factorizing the modulus while the duration of the connection. This can shrink the processing time for doing RSA operations dramatically.
For every connection, a new key pair is created, or an older key can be used for a couple of days or months, as long as the timeframe is smaller than the approximated factorizing time.

Another way to speed up the computation, at least at the side of one of the communication partners, can be found in the key generation scheme of RSA (refer chapter 5.2). The exponent d for the decryption process can be chosen freely, as long as it has no common divisor with Eulers phi function, except the 1. This gives the opportunity to choose d small. A small d will lead to a faster exponentiation process because the amount of multiplications to be done is lower. A low decryption exponent doesn't influence the security of the algorithm in a negative way, of course it should not be 1 because this would lead to identical public and private keys. Often the exponent 0x010001 is used as can be seen in some certificates.
The encryption exponent e depends on d and there is no way to influence it, it will usually be a big number. So, the speed of the computation is also some kind of asymmetric. This feature could also be mirrored and used to keep e small, while d is big, but in practice it is mostly used to keep d small also

due to the fact that the user of e also knows p and q and so can speed up the computation by other ways like the chinese remainder theorem[4].
The small d is the reason for the statement, that the signature verification is very efficient, as it has been given some chapters before.

5.5 Key management

A key management mechanism negotiating the use of RSA signatures has to include the length of the RSA modulus. There are many different hardware devices out there, which have different capabilities. A device that is not able to handle the verification of signatures with larger key sizes should have the opportunity to decline the connection.
When using a group key management system such as GDOI[5], the public key should be sent as part of the key download. If the group has multiple senders, the public key of each sender should be sent as part of the key download policy.

It is very important, that the public keys are transmitted in a trusted manner. The receiver has to have the opportunity to have trust in the origin of the public key. Let us assume, that Alice wants to download Bob’s public key. This download request is been hijacked by Eve who creates a key pair of her own. She then sends one key of the key pair to Alice and so makes her believe that the key is from Bob. If Alice now trusts in that key, Eve can spoof Bob’s identity and fake as much messages as she likes. This problem is a general problem in those cryptographic systems, the algorithms can not create trust. The user of the public key is responsible for having trust into a certain public key.

6 Attacks

In RFC 3552 some basic attacks are described. Now this document will discuss the possible threats, caused by those attacks, when using an RSA authentication system.
IPSec already has some security features which provide countermeasures to some of those attacks. First of all, the concept of security associations. Every IPSec connection has got an entry in a security association database on sender and receiver side. Everybody who wants to send a packet with a faked massage, would have to know some details about the parameters of the security association, for example the unique security parameter index, a unique number, which is also part of every IP packet sent. A receiver gets the IP packet, reads out the SPI and looks it up in the security association database if there is an entry, he accepts the packet. If not, the packet will be dropped.
Another feature is the use of a sequence number which is synchronized between sender and receiver. Sender and receiver negotiate the starting sequence number before establishing the connection. When

[4] Mathematical transformation which makes it possible to calculate the exponentiation mod n in the fields mod q and mod p, where numbers are smaller.
[5] Group Domain Of Interpretation

sending, the sequence number is put into the packet and the sequence number counter of the sender is incremented. Receiving the packet, the receiver reads out the sequence number and compares it to its internal counter. If the values match, he accepts the packet and also increments his counter (of course other details like the digital signature, also have to be valid), otherwise he drops the packet.
Authentication then additionally makes it impossible, that an attacker could forge a valid IP packet, even if he would know the correct security association and sequence number details. But for the following attack considerations, let us assume, that our attacker is in possession of correct connection parameters, or that our attacker is a member of the group and so of course knows these parameters.

6.1 Replay attack

When doing a replay attack, an attacker records some data packets of a sender and then tries to send copies of those packets to the receiver at a later time.
This attack is prevented by the sequence number of IPSec. The receiver will verify the packets correctly because the authentication is guaranteed. The packets seem to be from the correct sender – because they are copies of original packets. But when looking at the sequence number, the receiver will detect, that the packets has already been received and so will be dropped. The replay attack will fail. Generally the receiver should check the sequence number first, because this step is faster than doing the verification of the RSA signature.

6.2 Message modification attack

In message modification, an attacker tries to modify the data of a IPSec packet while the sending process. This attack will fail because the digital signature then won’t fit the hash value that will be calculated out of the data itself. The receiver would detect the mismatch of the hash values and drop the packet.
To create a correct digital signature to the modified data, the attacker would have to know the private key of the sender, he doesn’t know.

6.3 Man in the middle attack

The attacker manages it to be between sender and receiver. Every sent packet he also can read and modify in any way. This attack is a concern in many cryptographic algorithms. But when recalling the fact, that a valid digital signature can only be produced, when the private key is known, we see, that a man in the middle attack will also fail, at least if the public key has been shared in a trusted manner and if the key pair is strong enough to stand a factorization attack while the duration of the connection.

6.4 Denial of service attack

Especially when using RSA as authentication scheme, prevention of denial of service attacks is very important. Because of the performance problems described in chapter (refer Chapter 5.3). An attacker could easily bring a receiver to its knees by permanently sending many packets, the receiver has to verify. In a multicast group, even all members would be under attack simultaneously.
So a couple of countermeasures have to be taken to decrease the danger of denial of service attacks to an acceptable level.

When receiving a packet, the receiver should first check security association and sequence number of the received packet. These two steps are not much time consuming, and are able, to filter DOS packets out which don't have correct connection parameters.

But an attacker how is able to create correct parameters is still able to do a DOS attack. So every AH or ESP packet should be encapsulated into a wrapper AH packet with HMAC as authentication scheme. This HMAC uses a pre shared key, all group members know. It is much faster in terms of verifying. An attacker who is not member of the group, is not able to force the receiver to do the RSA verification, because he will do the much faster HMAC verification before and drops the packet as not originated by a group member.

Only an attacker who knows the secret shared HMAC key would now be able to cause the receiver to do the RSA verification process and so, start a DOS attack, but due to the fact that only members of the group know the secret key, this threat is acceptable.

7 Conclusion

RSA is a good way to provide data authentication and data integrity when used within AH or ESP. The most important feature compared with the usual methods like HMAC is, that even in group traffic, the authentication und integrity of a data packet is guaranteed.
The disadvantage of the RSA scheme is that it is much slower than HMAC what can cause problems in some applications. Fortunately there are a couple of ways to speed up the computation. It resists most common attacks when implemented in a reasonable way.

8 References

Kent, S., "IP Authentication Header", draft-ietf-ipsec-rfc2402bis-05.txt, September 2003.

Kent, S., "IP Encapsulating Security Payload (ESP)", draft-ietf-ipsec-esp-v3-06.txt, July 2003.

http://www.iana.org/assignments/isakmp-registry

Kent, S., "Security Architecture for the Internet Protocol", draft-ietf-ipsec-rfc2401bis-00.txt, October 2003.

Thayer, R., Doraswamy, N., and R. Glenn, "IP Security Document Roadmap", RFC 2411, November 1998.

Jonsson, J., B. Kaliski, "Public-Key Cryptography Standard (PKCS) #1: RSA Cryptography Specifications Version 2.1", RFC 3447, February 2003.

Madson, C., and R. Glenn, "The Use of HMAC-SHA-1-96 within ESP and AH", RFC 2404, November 1998.

Delfs, H., Knebl, H., Introduction to Cryptography, Principles and Applications.
Springer, Berlin, 2001.

Beutelspacher, A., Schwenk, J., Wolfenstetter, K.D., Moderne Verfahren der Kryptographie. Vieweg, Wiesbaden, 1995.

Buchmann, J., Einführung in die Kryptographie. (2. Aufl.) Springer, Berlin, 2001