# Internet Key Exchange Version 2

in the context of the lecture *Internet Security*
at the *GSO university of applied sciences*

Stefan Zech

Stefan.Zech@student.fh-nuernberg.de

June 29, 2004

## Preamble

In context of studying computer science at the Georg-Simon-Ohm university of applied sciences I have choosen to take part at the subject *Internet Security* at Prof. Dr. Trommler. Because I'm very interested on security aspects this decision was perfect.

My challenge at this subject was to incorporate into the Internet Key Exchange Protocol version 2 based on the internet draft[1] about this protocol. After that i had to pass this knowledge to the other students by a presentation and this composition.

Overall the subject *Internet Security* and all the presentations of the other students where very interesting and I have learnd much about technologies to provide security in networks. But I also have learned much about the unsolved problems in connection with security. This will help me to be more critically on security aspects in future.

Stefan Zech

# Contents

# List of Figures

# 1 What is IKEv2

The *IP Security Protocol (IPsec)* is used to communicate in a secure way over a network. To achieve such a secure connection, *IPsec* must provide confidentiality, data integrity, access control and data source authentication the IP datagrams. This services are provided by maintaining shared state between the source and the sink of an IP datagram. The *Internet Key Exchange protokol (IKE)* provides the service to establish this shared state between the source and the sink. But also IKE makes sure that this shared state keeps secure. Many details of *IKE* are based on *ISAKMP*[6] and the headers are very simmilar.

To establish this shared state, *IKE* performs mutual authentication between two parties and establishes an IKE security association. By establishing this security association, *IKE* negotiate a shared secret and the cryptographic algorithms to be used by the communication between the two parties. Also IKE negotiate the keys for the cryptographic algorithm, and from time to time it changes the key for more security.

But this protocol is also designed to be resistend to most attacks from hackers. Even a limited DoS protection is available.

# 2 IKE Exchanges

In this section I will show the exchanges of the *IKE* protocol. The first are the negotiation IKE exchanges. They are used to initiate an IPsec conversation using IKE. The second exchange is the *CREATE_CHILD_SA* exchange which generates child *Security Associations*. The last exchanges is the *INFORMAL Exchange*. This Exchange is used to to send information payloads like errors over the *IKE* protocol.

## 2.1 Negotiating an IKE Exchange

Communication using IKE always begins with the *Initial Exchanges*. This *Initial Exchanges* are *IKE_SA_INIT* and *IKE_AUTH*. In figure 1 the messages of the standard *Initial Exchanges* are diagramed.

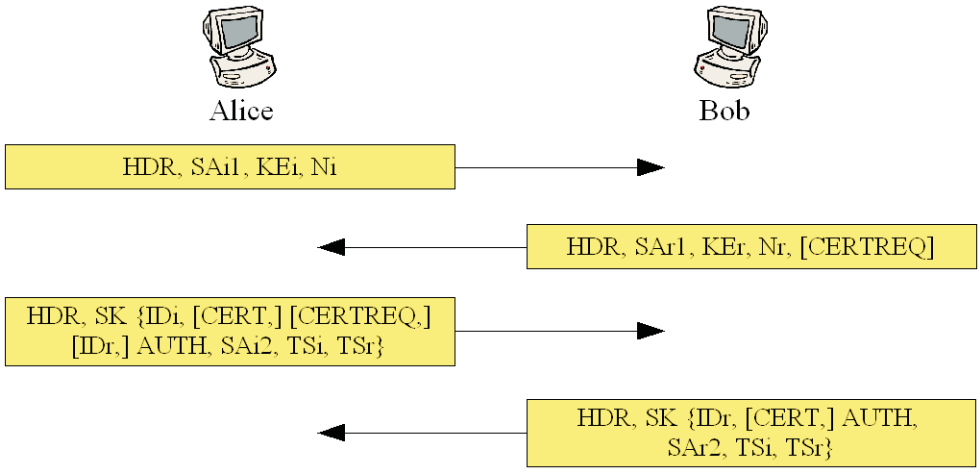


***Figure 1:*** *Initial Exchanges*

### 2.1.1 IKE_SA_INIT exchange

The first pair of messages are the *IKE_SA_INIT* exchange and negotiate the cryptographic algorithms, the exchange nonces and do a Diffie-Hellman exchange. The first message will be send by the initiator (Alice). Like all *IKE* datagrams it begins with the *IKE Header (HDR)*. This header contains the *Security Parameter Indexes (SPIs)*, the version numbers and some flags. The next payload of the first message is the first *Security Association Payload (SAi1)* of Alice. The *SAi1* payload states the cryptographic algorithms alice supports for the *IKE_SA*. The next payload is the *Key Exchange Payload (KEi)* and includes the Diffie-Hellman value for the Diffie-Hellman exchange. The last payload of the first message is the *Nonce Payload (Ni)* and includes a random number.

When Bob receives such a initial message he respond with the first *Security Association Payload (SAr1*, the *Key Exchange Payload (KEr)* and also a *Nonce Payload (Nr)*. The *SAr1* payload of Bob states Alice whith cryptographic algorithm he has choosen and his *KEr* payload completes the Diffie-Hellman exchange. Optionally he can send a *Certificate Request Payload (CERTREQ)*, if he want get Alice's certificate to authenticate her.

Now Alice and Bob have all information for a cryptographic protected conversation and all messages after the *IKE_SA_INIT* exchange are protected with the negotiated cryptographic algorithm. Only the headers of the following messages are not encrypted. At figure 1 we see that the payloads of the *IKE_AUTH* exchange (second message pair) are encapsulated in an *Encrypted Payload (SK)*.

### 2.1.2 IKE_AUTH

The *IKE_AUTH* exchange begin with a request message of Alice, too. The first payload of this request is the *Identification Payload of the initiator (IDi)*. Alice asserts her identity with the *IDi* Payload. This payload can be her IP address, a fully-qualified domain name or email address and so on. The different ID types are listed at the internet draft of *IKEv2*[1]. The next three payloads are optionally. The *Certificate Payload (CERT)* is only necessary if Bob sent an *CERTREQ*. The *CERTREQ* of Alice is only necessary if she want to get a certificate of Bob. The *Identification Payload of the responder (IDr)* is only necessary, if Bob has more than one identities and Alice want to communicate with a specific identity of Bob. The *Authentication Payload (AUTH)* Alice uses for authenticate her identity. The value of this *AUTH* payload is calculated with a shared secret. The second *Security Association payload (SAi2)* starts the negotiation of a *Child Security Association (CHILD_SA)*, because data packages of *IPsec* can only be sent within an *CHILD_SA*. This *SAi2* payload again contain an offer of cryptographic algorithms which can be used for a *CHILD_SA*. At last the *Traffic Selector (TSi, TSr)* payloads include policies like the port range or the IP adress range where it is allowed to send *IPsec* packages. In this first Message of the *AUTH* exchange Alice tells Bob her policies for both directions.

Finally Bob respond the request of Alice. The first payload now is Bobs *Identity Payload (IDr*. If Alice has sent a *CERTREQ*, the next payload is Bobs certificate. Then bob sends an *AUTH* to protect the integrity of the second message. The last to payloads are Bob's *Traffic Selector payloads (TSi, TSr)*. Now Alice and Bob know all policies sent by the *Traffic Selector payloads* and they use the intersection of all policies.

## 2.2 CREATE_CHILD_SA exchange

The *Create Child Security Association exchange (CREATE_CHILD_SA)* is used if a new *Child Security Association* is needed. It may also initiated by either end of the *IKE_SA* after the *Initial Exchanges* are completed, because there must be *CHILD_SAs* in both directions to communicate over IPsec.

In figure 2 the message pair of this exchange is diagramed. First the initiator Alice send an request to Bob. The first Payload in the graphic is the optional payload *Notify payload (N)*. It is used in the case of rekeying an *CHILD_SA* to identify the *SA* being rekeyed. The *Security Association (SA)* payload includes a list of supported cryptographic algorithms equally to the first *SA* payload at the *Initial Exchange*. The next payload is the *Nonce payload (Ni)* of Allice. At the end there are three optionally payloads, the *Key Exchange payload (KEi)* and the *Traffic Selector payloads (TSi, TSr)*. The *KEi* payload is needed if a new Diffie-Hellman exchange should performed. The *TSi* and *TSr* payload is needed if the policies for the traffic have been changed. As always Bob

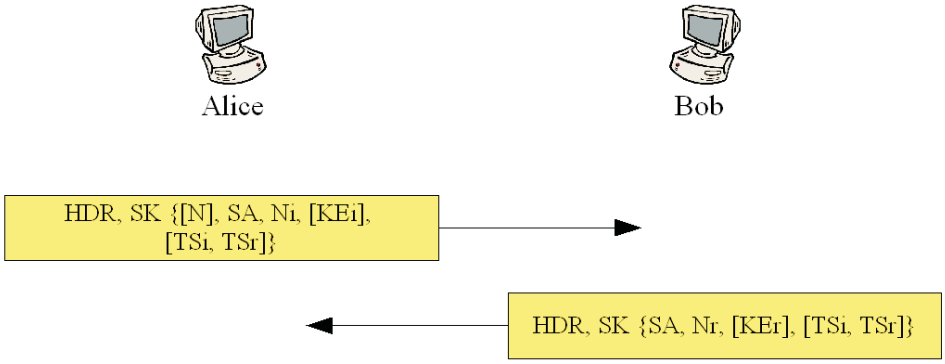


***Figure 2:*** *Create Child Security Association exchange*

respond to the request. This response includes again an *SA* with the choosen cryptographic algorithm. And Bob also respond a *Nr*. Optionally are the *KEr*, the *TSi* and *TSr* payloads. They only needed if the request includes that payloads.

## 2.3 INFORMATIONAL exchange

The last exchange is the *INFORMATIONAL* exchange. It is used for reconfiguration of a *Security Association (SA)*, for error handling and to send an *Delete Payload (D)* to delete a *SA*. In figure 3 this exchange is diagramed. It is a very simple exchange. Again there is a message pair of a request and a response. The response is only an anknowledgement in most cases. This messages only

include one or more informational payloads of different types, and all payloads are encrypted.

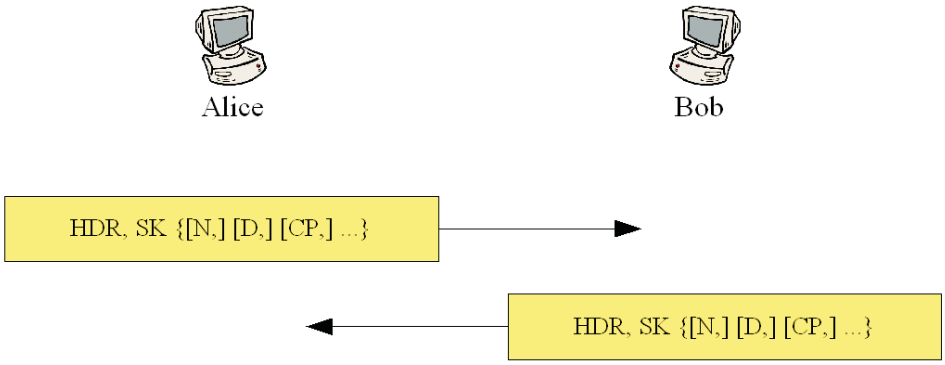


***Figure 3:*** *INFORMATIONAL exchange*

# 3 IKEv2 details and variations

In this Section I describe the most important details and variations of the *IKE* protocol. It is not a complete list of all details and variations, but it is enought for a detailed overview of the protocol.

## 3.1 Retransmission Timers

If a request message was sent and there is no response during a delay time the initiator must repeat his request. But only requests have to repeat. It is only allowed to repeat a response if the initiator has repeatet the request. If there is no response after a few retransmissions the initiator must deem that the *IKE Security Association (IKE_SA)* has failed and must delete the it.

## 3.2 Sequence Numbers for Message ID

Every *IKE* message contains a *Message ID* as part of it's fixed header. This Message ID is used to match up requests and responses, and to identify retransmissions of messages. This *Message IDs* are incremented from message to message. If now a message is received with the same *Message ID* of a Message before the receiver can drop this message. Thus this *Message IDs* are a protection against message replay attacks.

## 3.3 Window size for overlapping requests

This mean that an initiator can send multiple requests before getting a response to any of them. The number of this messages depends on the implementation of the *IKE* protocol. This window size is necessary to maximize the throughput of messages. However this makes the implementation of the protocol a little more complex but rather efficient.

## 3.4 State Synchronization and Connection Timeouts

In case that one endpoint of the *IKE Security Association (IKE_SA)* crashes the other endpoint must be able to detect this and has to close the connection so that not to waste network bandwidth by sending packets over discarded *SAs*. A hint to such a crashed endpoint can be routing information (e. g. ICMP messages) or IKE messages that arrived without cryptographic protection. But before the receiver of this messages conclude that the *IKE_SA* has faild he has to check this by sending an empty *INFORMATIONAL* message. Only if the other node doesn't respond after a few of this messages the *IKE_SA* has failed. This checking reduces the risk of *Denial of Service (DoS)* attacks.

### 3.5 Cookies

In the *IKE version 2* protocol *Cookies* are used for a limited *DoS* protection in case of forged source IP addresses. This mean that a *DoS* attacker can send a huge amount of requests for an *Initial Exchange* with forged IP addresses. Than the attacked node has to generate a respond with all payloads for all requests. But if a node detects such a huge amount of half-open *IKE_SAs*, it can send a response with only one payload like it is diagramed in figure 4. This payload is a *Notify payload (N)* which tells the initiator Alice, that she has to send a *Cookie* first. This is very effective because the calculation of the payloads of the response without *Cookies* needs a huge amount of CPU time. Whereas the response with the *Notify payload* needs minimal CPU time. After this response Alice has to retransmit the request with a additional payload for the *Cookie*. After this request the Initial Exchanges continue like in standard case. Thus an attacker needs great many messages to be successful with the *DoS* attack.

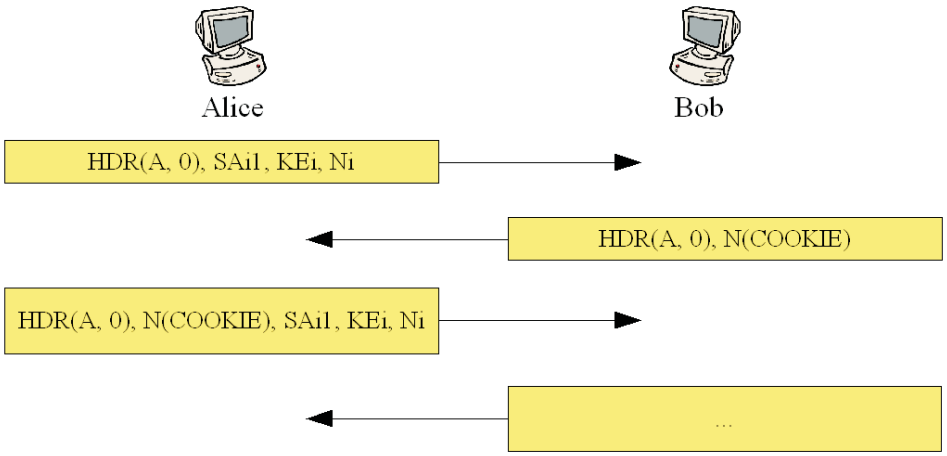


***Figure 4:*** *Initial Exchanges with Cookies*

### 3.6 Rekeying

The keys of a *Security Association (SA)* should only be used for a limited amount of time. This is useful to reduce the danger of hacked keys. To change the keys after their lifetime *Rekeying* is necessary. This can be done by delete the old *SA* and establish a new one or by negotiating new keys für the current *SA*. The second way of *Rekeying* is optional but should be implemented in an *IKE* implementation.

### 3.7 Traffic Selector Notification

A *IPsec* subsystem uses a *Security Policy Database (SPD)* to decide which messages have to be protected with *IPsec*. But the two endpoints can have different policies in their *SPD*. To negotiate a common subset of this policies the *Traffic Selector payloads (TSi, TSr)* needed. The *TSi* payload for packages from the initiator to the responder and the *TSr* payload for the opposite direction. This payloads are needed by negotiating an *Securita Association (SA)* and to update the *SPD*.

### 3.8 Nonces

The *Nonce (Ni, Nr)* Payloads are used to improof the security of the keys for the cryptographic algorithms. This *Nonces* are only random numbers and must be at least 128 bits in size and must be at least half the key size of the negotiated *pseudo-random function (prf)*. Each node creates a *Nonce*. Then each node uses the two *Nonces* as inputs to the cryptographic functions for generating the keys. This add freshness to the key derivation technique and so it supports more security.

### 3.9 Handling of keys

There are many hints and rules in the internet draft[1] of the *IKEv2* protocol for handling keys. In this section I give a little overview about the most important rules.

- If a *Security Association (SA)* has been closed all keys and secrets of this *SA* must be deleted immediately. They must deleted in the memory and on the harddisk! This must be done to avoid that an attacker steals this keys and secrets to decrypt the sent data afterwards.
- Computing *Diffie-Hellman* exponentials needs a lot of CPU time, but reusing such an exponential is not allowed.
- There are much rules and hints for the process of generating key material, too. But this is shown in detail at the internet draft[1].

### 3.10 Authentication of the IKE_SA

When not using extended authentication, the peers authenticated by each sign a block of data. The algorithm for this sign is a *Pseudo Random Function (prf)* and is based on a shared secret. There is a hint in the internet draft[1] that it is unsecure to use a secret which is derived from a users password, because the

risk of dictonary attacks is to big. The algorithm of the *prf* for the sign is not defined at the draft and it is not necessary that the two endpoints use the same *prf* for the sign.

### 3.11 Extended Authentication Protocol Methods

This authentication uses methods defined in the *RFC 2284* [2] of *EAP*. This authentication method uses public key signatures and shared secrets and supports more security than the normal Authentication. To perform this authentication a *EAP* payload is used. But this authentication is also more complex and needs four message pairs at the *IKE Initial Exchanges*. The operating sequence of this authentication is described in detail in the internet draft[1]. One of the common features of the *Extended Authentication Protocol Methods* is that this methods are more robust against dictonary attacks than the normal Authentication.

### 3.12 Requesting an internal address on a remote network

At a endpoint to security gateway scenario most commonly an endpoint may need an IP address in the network protected by the security gateway. This IP address must be dynamically assigned.

The *IKE* protocol uses a *Configuration Payload (CP)* for requests to a *IPsec Remote Access Server (IRAS)*. This server is responsible to assign a IP address to the *IPsec Remote Access Client (IRAC)*. This exchange is shown in figure 5. First the *IRAC* sends a request in the *IKE_AUTH* exchange which includes the

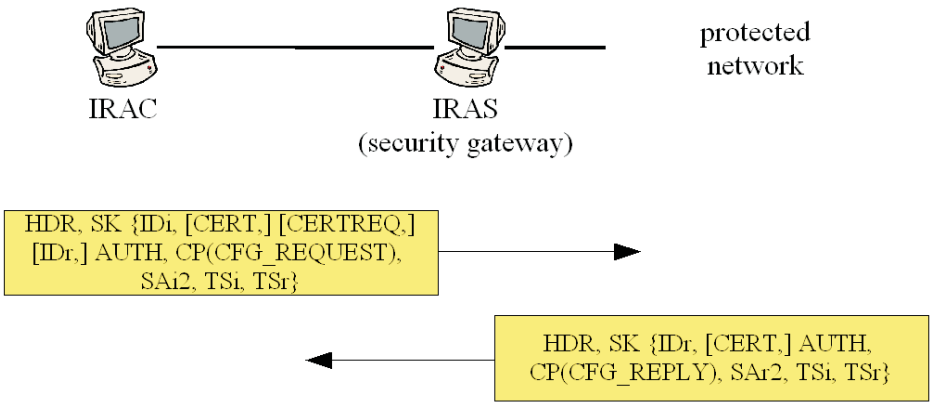


***Figure 5:*** *Request internal address on a remote network*

additional *CP* payload with the configuration request to the *IRAS*. Than the *IRAS* may procure an network setting for the *IRAC* such as a *DHCP/BOOTP* server in a response. After this response the *IRAC* reconfigures it's network

settings and can pass the security gateway through the originated tunnel. The security gateway and the *IRAS* can also be located in different nodes.

### 3.13 Error Handling

In an *IKE* processing there are many kinds of errors that can occur. But if a node receives error messages outside an *Security Association* and so without cryptographic protection the node must be very carefully with this messages. For diagnosing the problem it should response this messages, but only if the arrived messages are marked as requests. But if a node receives numerous requests without cryptographic protection it must limit the rate of this responses, because this messages can be a trial of a *DoS* attack. If a node receives encrypted errors it can audit the messages and can response them to fix the problem.

### 3.14 NAT traversal

If a node is behind a *Network Address Translator (NAT)* there are two problems. The first problem is that the *NAT* translates the source IP address. But this IP address is cryptographic protected by a checksum in the encrypted payload. The second problem is that the *NAT* also translates TCP and UDP port numbers. But IKE normally communicates only on port 500 and port 4500.

But this problems have been born in mind by designing the *IKE* protocol. The solution of the first problem is that the nodes of the *IKE* processing can detect a *NAT* traversal by *Notify payloads (N)* of type *NAT_DETECTION_SOURCE_IP* and *NAT_DETECTION_DESTINATION_IP*. If there is a NAT between there must be an additional *Traffic Selector (TSi, TSr)* payload in each message which includes the original IP address to fix the checksum. The fix for the second problem is a little bit easier. A node must also accept *IKE* messages from other source ports than 500 and 4500 if a *NAT* is between.

# 4 IKEv2 headers

In this section I explan the two most important headers of the *IKE version 2* protocol, the *IKE Header* and the *Generic Payload Header*.

## 4.1 IKE Header

All messages of the *IKE protocol* are beginning with the *IKE Header*.

In figure 6 we see that this header beginns with the *Security Parameter Index (SPI)* of the initiator and the responder. This *SPIs* are values chosen by the initiator and the responder to identify a unique *IKE Security Association (IKE_SA)*.

Then the *Next Payload* field follows. It identifys the type of the first payload in the *IKE* message. If it is zero there is no *Payload* inside the message.

The next fields are the *Major Version (MjVer)* and the *Minor Version (MnVer)* fields to identify the version of the *IKE protocol*.

The field *Exchange Type* identify the current exchange. This can be a *IKE_SA_INIT*, a *IKE_AUTH*, a *CREATE_CHILD_SA* or a *INFORMATIONAL* exchange. In future the *IKE protocol* can be supplemented with other exchange types.

The *Flag* field includes some flags which are described in the internet draft[1].

The *Message ID* field includes the sequenced Message ID like it is describet in Section 3.2.

The last field is the *Length* field which simply includes the total message length.

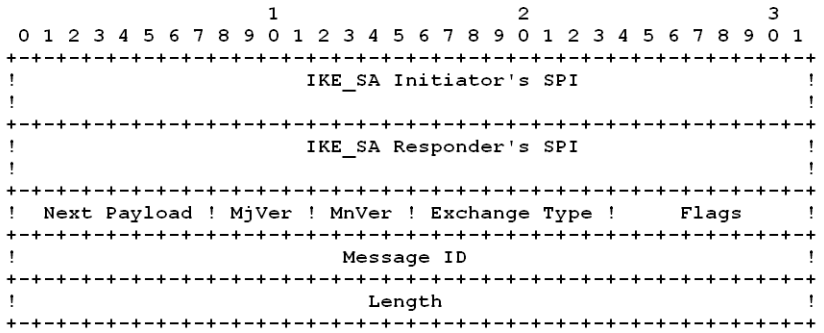


***Figure 6:*** *IKE version 2 Header*

## 4.2 Generic Payload Header

Each payload begins with the *Generic Payload Header* like it is displayed in figure 7. This is a very simple header. It begins with the *Next Payload* field which identifys the type of the next payload. The second field is the *Critical Bit (C)*. This bit is set to zero if the sender wants the recipient to skip this payload if he does not support the *Payload Type* of this payload. If this bit is 1 and the recipient doesn't support the *Payload Type*, the recipient must respont an error. The third field is reserved vor future and must be set as zero. The last field is the *Payload Length* which simply includes the length of the payload.

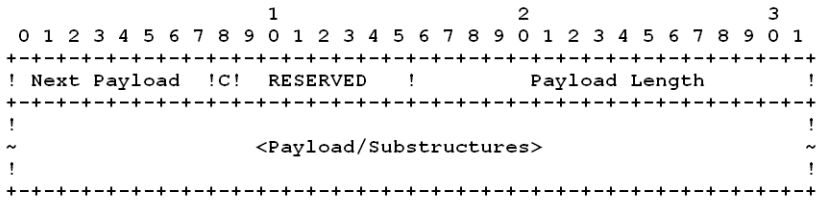


***Figure 7:*** *Generic Payload Header*

# 5 Conclusion

After all I would say the *IKE version 2* protocol is a very complex and well designed protocol which considered a lot of security aspects. Also the protocol have been born in mind of a lot of special cases like the *NAT Traversal* or the *Security Gateway*.

But this complexity of the protocol can also lead to bad implementations because the developerst must attend to a huge numbers of details and this can end in security holes. But it would be very hard to find a simpler protocol with the same functional range and the same security level like the *IKE protocol*.

# References

[1] Charlie Kaufman, *Internet Draft: Internet Key Exchange (IKEv2) Protocol*, http://www.ietf.com, revision 13, March 22, 2004

[2] L. Blunk and J. Vollbrecht, *RFC 2284: PPP Extensible Authentication Protocol (EAP)*, http://www.ietf.com, March 1998

[3] S. Kent and R. Atkinson, *RFC 2401: Security Architecture for the Internet Protocol*, http://www.ietf.com, November 1998

[4] S. Kent and R. Atkinson, *RFC 2402: IP Authentication Header*, http://www.ietf.com, November 1998

[5] S. Kent and R. Atkinson, *RFC 2406: IP Encapsulating Security Payload (ESP)*, http://www.ietf.com, November 1998

[6] D. Maughan, M. Schertler, M. Schneider and J. Turner, *RFC 2408: Internet Security Association and Key Management Protocol (ISAKMP)*, http://www.ietf.com, November 1998

[7] D. Harkins and D. Carrel, *RFC 2409: The Internet Key Exchange (IKE)*, http://www.ietf.com, November 1998