Computer Network Security Essay

IKEv2: Internet Key Exchange over IPSec

Author: Luca Campa

Contents

1 Introduction: general overview of IPSec and IKEv2

Internet Protocol Security (IPSec) is a network protocol used to secure the communication between two communicating parties over the network. Being a security network protcol, it requires mechanisms for establishing mutual authentication, chosing cipher suites for encryption and integrity protection and creating a communicating session secured by negotiated cryptographic keys.
IP Security provides confidentiality, data integrity, mutual authentication and access control to IP Datagrams thanks to the cryptographic algorithms mainteined withing a shared session state.
IKE performs mutual authentication between the two parties involved in the communication and establishes an IKE Security Association (SA) which is a container of secret information used to establish other SAs (called Child-SAs) and to protect the traffic that they carry.
IPSec can be directly implemented on an host or on a gateway: the main difference is that, on the host we are securing the communication going to or coming from that particular host, whilst on the gateway we are securing all the communication from and to the network under the gateway.
IPSec can be used with two main modes:

- Transport Mode: encrypts the payload and performs the integrity protection of the header and the encrypted payload
- Tunnel Mode: encrypts the payload and the first IP Header and performs the integrity protection of the all packet.

2 IKEv2: Protocol details

2.1 Ports and packet handling

IKE normally listens and sends on UDP port 500, but also on UDP port 4500 with a slightly different format. IKE is designed in order to provide recovery from transmission errors, packet replay and packet forgery and to minimize the effects of DOS attacks.
IKEv2 is designed for short messages which do not require fragmentation at the IP Level. Indeed, it does not define mechanisms to deal with fragmentation of large messages, such as the X.509 Certificates, because IP Fragmentation opens an implementation to Denial-Of-Service attacks. In general, IKEv2 implementations must be able to send, receive and process messages at least of 10kB, but they should be able to deal with packets up to almost 24kB.
The main technique to avoid fragmentation in case of certificates is to use the "Hash and URL" format (URL of the server containing the certificate in DER format and the HASH of the certificate itself for checking validity).

IKE messages sent and received on port 4500 need to begin with the prefix of four zeros in order to permit the service to understand the type of packet. That port is commonly used in IPSec in order to deal with NAT Traversal Communications.

2.2 Sequence Numbers for Message ID

Every IKE message contains a message ID (sequence number) as part of its fixed header which is used to uniquely identify a message withing a particular SA. This message ID is used to match

up requests and responses and to identify retransmission of messages. The message ID is a 32-bit quantity, which is 0 for the IKE_SA_INIT messages and it is incremented for each subsequent exchange. After Rekeying the sequence number is reset to 0.
Each endpoint of the IKE SA mantains two counters:

- one counter for the message ID to be used for the next request an endpoint initiates
- one counter for the message ID an endpoint expects to receive from the other party.

That means that at most 4 messages in the communication will have the same message ID and they are identified by the Initiator and Response flags in the message header: given two parties A and B, the messages with the same sequence numbers will be:

- the request from A to B
- the response from B to A
- the request from B to A
- the response from A to B

Message IDs are cryptographically protected during communications in order to provide protection against replay attacks. Moreover, if they reach the maximum possible value ($2^{32} - 1$), a rekeying is performed.

2.3 Slide Window technique

The sliding window technique, together with the sequence numbers, is used to keep track of the trasmitted messages and to provide protection against replay attacks.

How it works? The window size is always 1 until the initial exchanges are completed. The SET_WINDOW_SIZE message is used to communicate the other party how many messages the sender premises to keep. An IKE endpoint must wait for a response to each of its messages before sending a subsequent message. Otherwise, if we want to increase the throughput of the communication, the sender can send multiple messages only if it receives from the other party a SET_WINDOW_SIZE notification indicating it is able to keep track of multiple messages (the windows size has been increased). These messages are not supposed to reach the destination in the wanted order because of the unreliablity of the network and the UDP protocol used. Then, in order to avoid deadlocks, an IKE endpoint must be prepared to accept and process unordered requests. Of course, the endpoint must not exceed the peer's stated window size, meaning that the initiator can make a request with sequence number X only if it has received responses to all requests up to $X - W$ (W is the window size). The window size cannot be decreased. It can be increased with a SET_WINDOW_SIZE notification message or reset to 1 thanks to rekeying. When a peer receives a message outside the window size it must inform the other party about that event through an INVALID_MESSAGE_ID notification.

2.4 Retransmission Timers

If a timeout occurs, the initiator of the communication is responsible for the retransmission of that message. A responder must never retransmit a response unless it receives a retransmission of the request from the initiator. In order to make this process coherent, the initiator must remember each request until it receives the corresponding response (remember that in IPSec, each request needs a response with the same message ID). On the other hand, the receiver should remember each response until it receives a request with a message ID (sequence number) larger or equal to the the sequence number of the last response plus its window size. In order to not fulfill the entire memory, responders are allowed to forget responses after a timeout of several minutes. An important thing to notice is that the retransmitted message will have the same message ID of the original message.

2.5 IKE SA SPIs and Cookies

One of the main important characteristics of IPSec is the use of Security Parameter Indexes in order to link a particular policy and a particular endpoint to a specific Security Association. The two initial fields (8 bytes each) in the header are the IKE SPIs. They are used as identifiers of a specific SA. It is important that these values are unique in order to avoid ambiguities. An SPI value of 0 means that the remote SPI value is not yet known by the sender. SPIs are used to redirect the messages to the correct SA, avoiding the possibility that an other SA is able to read messages except from the ones belonging to it. Incoming IKE packets are mapped to the correct IKE SA only by using the packet's SPI and not by using the source IP Address. Unlike ESP or AH packets, IKE packets contains both the sender and the receiver SPIs. Thanks to the Initiator and Receiver flags within the header, an endpoint is able to understand what SPI to use in order to identify the correct SA.

Attacks Two possible attacks against IKE are the State and CPU exhaustion. The target is flooded with session initiation requests from forged IP addresses that will consume the target resources. Possible mitigations to this are the use of minimal CPUs and the commition of state to an SA only after the initiator has proved it is able to receive packets at the address from which it claims to be sending them. These attacks will leave a lot of half-open IKE SAs. In order to avoid such kind of attacks, IKEv2 introduces the use of COOKIES. When a responder detects a large number of half-open IKE SAs, it should reply to the IKE_SA_INIT requests with a response containing a COOKIE notification. The initiator must then retry the IKE_SA_INIT request including the COOKIE notification as the first payload.

Properties and generation of COOKIES Cookies are used to avoid flooding and DOS attacks. This piece of data must be between 1 and 512 bytes in length. Each party should implement a specific cookie generation method and a cookie validation method. Modern implementations do not require any saved state to validate and recognize correct cookies sent from the other party.
The standard does not specify a specific method to generate cookies (this is not the best choice, because wrong implementations could expose them to possible attacks), but a possible guideline could be:

$$\text{Cookie} = \langle\text{VersionIdOfSecret}\rangle \mid \text{Hash}(\text{Ni} \mid \text{IPi} \mid \text{SPIi} \mid \langle\text{secret}\rangle)$$

where secret is a randomly generated secret known only to the responder and periodically changed (frequently) in order to avoid MITM attacks. Of course, <VersionIdOfSecret> must be changed together with the secret.
The cookie is recomputed when the IKE_SA_INIT arrives the second time and compared to the received cookie. If it matches, it is accepted. Using Nonces in the computation ensures that an attacker cannot successfully forge such message. Moreover, incorporating the SPI of the initiator prevents an attacker from fetching the same request with different SPIs (it avoids the creation of a lot of half-open IKE SAs, meaning DOS attacks).
In order to deal with the unreliability of the networks, IKEv2 implementations should be able, for a short time, to process cookies generated with the old cookie after a new one is generated. When one party receives an IKE_SA_INIT request containing an invalid cookie, that party must ignore the cookie and process the message in the usual way. The last remark was thinked in order to deal with busy servers, but unfortunately it opened the implementations to the so-called SLOTH attack (if vulnerable algorithms are used). The initiator should limit, usually with exponential back-off, the number of cookie exchanges before giving up with the connection.

2.6 Rekeying

IKE, ESP, AH Security Associations use secret keys that must be used only for a limited amount of time or to protect a limited amount of data. The standard does not define a specific time limit, but the implementers need to take care about this issue. The reestablishment of Security Associations which can happen after they expire is called Rekeying. SAs should be rekeyed proactively, meaning that the new SA should be established before the old one expires. IKEv1 and IKEv2 differ a lot in the way they handle the SA lifetimes. In IKEv1, SA lifetimes were negotiated, while in IKEv2 each end of the SA is responsible for enforcing its own lifetime policy and rekeying the SA when necessary. An other important difference between those implementations is that IKEv2 allows parallel SAs with the same Traffic Selectors between common endpoints. That is done with the purpose of supporting the rekeying of old SAs and to provide different QoS (Quality of Service).
When an old SA is rekeyed, the response should continue to send trffic on the old SA until one of those events occur:

- It has received a cryptographically valid message on the newly created SA which will replace the old one.
- It has received an IKE request to close the replaced SA.

Moreover, the initiator can send dummy ESP messages on the newly created SA in order to assure the responder that the initiator is ready to receive messages on it.
Notice that if the two ends have the same lifetime policies, it is possible that both will initiate a rekeying at the same time. This would raise to the instatiation of multiple reduntant SAs. To reduce the probability of that event, the timing of rekeying should be delayed by a random amount of time.

How does it work? The process of rekeying is the same for IKE SAs and the Child SAs:

- To rekey an IKE SA, create a new equivalent IKE SA within the old IKE SA (in that way, the new one will hereditate all the Child SAs of the old one). When the creation of such IKE SA is done, delete the old one.

- To rekey a Child SA, create a new equivalent Child SA with the same Traffic Selectors and Algorithms of the previous one, then delete the old one.

2.7 Traffic Selectors

When IPSec receives an IP Packet that matches an entry within the SPD (Security Policy Database), it encapsulate the packet in order to protect its content depending on the rules described by the chosen entry.

Traffic Selector payloads allow endpoints to exchange some of the information from their SPD to their peers. Their objectives are:

- Guiding the dynamic update of the SPD
- Specifying the selection criteria for packets that will be forwarded over the newly set up SA.

Two Traffic Selector payloads (TSi - initiator and TSr - responder) appear in each of the messages in the exchange that creates a Child SA pair. Each TS payload contains one or more traffic selectors which carry the information coming from the chosen entry of the SPD: an address range (IPv4 or IPv6), a port range and an IP Protocol ID. TSi specifies the source address of traffic from the initiator of the Child SA (or, in the packet from the other party, the destination address of the traffic forwarded from the responder). In the same way, TSr specifies the destination address of the traffic forwarded from the initiator (or, in the packet of the other party, the source address of the traffic from the responder). Usually the traffic selectors coming from the initiator are only a proposal for the responder. The responder will have to choose a subset (even all or one) of these proposals. In particular, the responder will perform the choice as follows:

- If the responder's policy does not allow it io accept any part of the proposed Traffic Selectors, it responds with a TS_UNACCEPTABLE notification message.
- If the responder's policy allows the entire set of trsffic covered by TSi and TSr, the responder can return the same TSi and TSr values to the initiator.
- If the responder's policy allows it to accept the first traffic selector of TSi and TSr, the responder must restrict the Traffic Selectors to a subset that includes the initiator's first choice.
- If the responder's policy does not allow it to accept the first selector of TSi and TSr, the responder will answer with an acceptable subset of the received TSi and TSr.

Of course, when creating a new SA, the initiator needs to avoid proposing Traffic Selectors that violate its own policy. If this rule is not followed, valid traffic may be dropped.

2.8 Nonces

Each IKE and CHILD SAs initialization messages (IKE_SA_INIT and CREATE_CHILD_SA) contain a nonce. These are useful to add freshness to the key derivation technique used to obtain keys for Child SAs and to ensure the creation of strong pseudorandom bits from the Diffie-Hellman key. IKEv2 requires nonces to be randomly chosen, at least 128 bits in size and at least long as half of the key size of the negotiated PRF. Notice that the nonces are chosen before the negotiation of the PRF, then the nonce's length must be long enough for all the proposed PRFs.

2.9 Generating Keying Material

During the IKE SA initialization, four cryptographic components are negotiated: the encryption algorithm, the integrity protection algorithm, the Diffie-Hellman group and a Pseudorandom Function (PRF). The PRF is also used in order to generate the keys for all of the cryptographic algorithms used in both the IKE SA and the Child SAs. Keying material will be derived as the output of the negotiated PRF algorithm. Since the amount of data needed may be greater than the size of the output of the PRF, it is used iteratively. This means that the keys are taken from the pseudorandom stream generated from the inputs to the negotiated PRF. This process is defined as:

- Stream = S1 | S2 | S3 ...
- S1 = PRF(K, IV | 0x01)
- S2 = PRF(K, S1 | 0x02)
- S3 = PRF(K, S2 | 0x03)

The constant concatenated at the end of each PRF function is a single byte because the iteration is not supposed to go beyond 255.

Keying material for the IKE SA The shared keys which are saved into the state of the IKE SA are:

- SK_d: used for deriving new keys for the Child SAs of that IKE SA
- SK_{ai}: the intiator's key for integrity protection
- SK_{ar}: the responder's key for integrity protection
- SK_{ei}: the initiator's encryption key
- SK_{er}: the responder's encryption key
- SK_{pi}: the initiator's key used when generating the authentication payload
- SK_{pr}: the responder's key used when generating the authentication payload

They are computed in the same order of the previous list by applying the PRF iteratively (call it PRF+):

$$\text{PRF+(SKEYSEED, Ni | Nr | SPIi | SPIr)} = \{\ SK_d|SK_{ai}|SK_{ar}|SK_{ei}|SK_{er}|SK_{pi}|SK_{pr}\ \}$$

where

$$\text{SKEYSEED} = \text{PRF(Ni | Nr, } g^{ir})$$

g^{ir} is the derived Diffie-Hellman key. The two directions of traffic flow use different keys, then there are two keys for encryption and integrity protection from initiator to responder and viceversa.

Keying material for the Child SAs The first Child SA is created by the IKE_AUTH exchange and additionally Child SAs can be created in future with the CREATE_CHILD_SA exchanges. Keying material for them is created as:

$$\text{KEYMAT} = \text{PRF+}(SK_d, \text{Ni} \mid \text{Nr})$$

where Ni and Nr are the nonces exchanged during the IKE_SA_INIT if it is the first Child SA or fresh nonces in the case of additional Child SAs. If the CREATE_CHILD_SA exchanges include optional Diffie-Hellman payloads, the key material is generated by using the new shared Diffie-Hellman key as follows:

$$\text{KEYMAT} = \text{PRF+}(SK_d, g^{ir} \text{ (new)}, \text{Ni} \mid \text{Nr})$$

A single CHILD_SA negotiation can result in multiple security associations, then we need to obtain keys for each Child SA. Keying material must be taken from the extended KEYMAT by following these rules:

1 All keys for the SAs carrying data from the initiator to the responder are taken before SAs going from the responder to the initiator.

2 If multiple IPsec protocols are notiated, keying material for each Child SA is taken in the order in which the protocol headers appear in the packet.

3 If the IPsec protocol requires multiple keys, they need to be taken in the order specified by their specification.

2.10 Authentication

When not using extensible authentication methods, the peers authenticate themself to the other party by signing a block of data.

- The responder will sign the bytes of its response to the IKE_SA_INIT message from the first byte of the first SPI to the last payload's byte appended with the initiator's Nonce (Ni) and the value PRF(SK_{pr}, IDr). Notice that these two values are not trasmitted alongside the signature.
- The initiator will sign the bytes of the IKE_SA_INIT request from the first byte of the first SPI to the last payload's byte appeded with the responder's nonce (Nr) and the value PRF(SK_{pi}, IDi).

It is extremely important that each side signs the nonce of the other party.

The signature can be generated by using public key encryption (which usually requires a certificate in order to bind a specific identity to the corresponding key) or by using symmetric encryption. In the latter case, it is mandatory that the shared key is generated by a good source of randomness and not only from a user-chosen password which is not a good source of entropy. Using such kind of source would expose an implementation to dictionary attacks. While in the case of public key signature schemes, the AUTH payload is the common application of the algorithm, in the case of a pre-shared key, the AUTH payload is computed as:

- For the initiator:

$$\text{AUTH} = \text{PRF}(\ \text{PRF}(\text{Shared Secret, padding (17 ascii characters)}),\ \text{<InitiatorSignedBytes>})$$

- For the responder:

$$\text{AUTH} = \text{PRF}(\ \text{PRF}(\text{Shared Secret, padding (17 ascii characters)}),\ \text{<ResponderSignedBytes>})$$

If the password based system is used, the authentication process should use EAP methods (Extensible Authentication Protocol). If the EAP is key-generating the shared secret is a computed Master Session Key (MSK). For non-key generating methods, use SK_{pi} and SK_{pr} for the shared secret in the two AUTH computations. EAP is used in order to provide mutual authentication and access control between the two parties before exchanging the AUTH payload. Non-Key generating methods should be avoided if they are used in protocols which does not support server-authenticated tunnels.

2.11 Internal Address Assignment

The most common user scenario is the endpoint-to-security-gateway communication, which means an endpoint want to securely connect to the network protected by the security gateway. In this situation, the endpoint commonly asks for the assignment of an internal IP Address which has to be chosen dinamically. A request for such a temporary address can be included in any request to create a Child SA by including a specific Configuration Payload (CP). However, it is usual to assign one IP Address during the mutual authentication phase (IKE_AUTH exchanges). The two parties involved are usually called IRAC (Internet Remote Access Client) and IRAS (Internet Remote Access Server - the gateway). During the mutual auhtentication phase, the IRAC must request the IRAS-Controlled address in the IKE_AUTH exchange. The IRAS will answer with an address chosen from a various number of sources (commonly DHCP).

How it works? An initiator can ask for the assignment of an IP address by sending a Configuration Payload within the IKE_AUTH message, before the SA payload. The Configuration Payload should contain at least the type (attribute) of required address (IPv4 or IPv6). The server (IRAS) will add to the response a Configuration Payload containing the required network information (IP, NETMASK, SUBNET).

3 IKEv2: Protocol exchanges

3.1 Common traffic flow

IKE communication always begins with four messages: two IKE_SA_INIT and two IKE_AUTH messages. They compose what in IKEv1 is called Phase 1. They are four in an ideal scenario. Sometimes can happen that retransmissions are required or additional messages with cookies or keys are necessary to solve some issues during the communication. The first pair of messages (IKE_SA_INIT) are used to negotiate cryptographic algorithms, exchange nonces and, if necessary,

perform a Diffie-Hellman Exchange (when not performed by using other methods like EAP). All the subsequent messages are encrypted and integrity protected through the cryptographic algorithms and keys negotiated during this process.

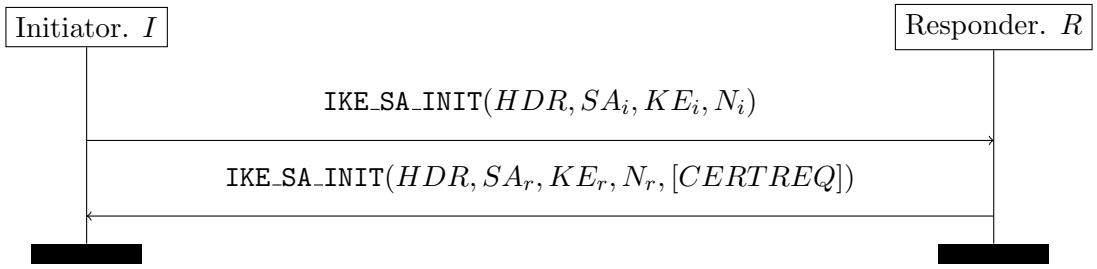
The initiator sends:

- the Header containing the SPIs and the other fields explained in the next Section.
- the SA payload which contains the cryptographic suites the initiator supports for the IKE SA to be instantiated.
- the Diffie-Hellman public key of the initiator
- the Initiator's Nonce

The responder answers with:

- the header containing also its SPI
- the SA payload which contains the chosen cryptographic suite
- the responder's Diffie-Hellman public key
- the responder's Nonce
- optionally a Certificate Request

At this points, both the peers can generate the SKEYSEED from which all the other keys are derived for that IKE SA (see *Section 2.9*). Remember that a separate encryption and integrity protection key is generated for each direction.



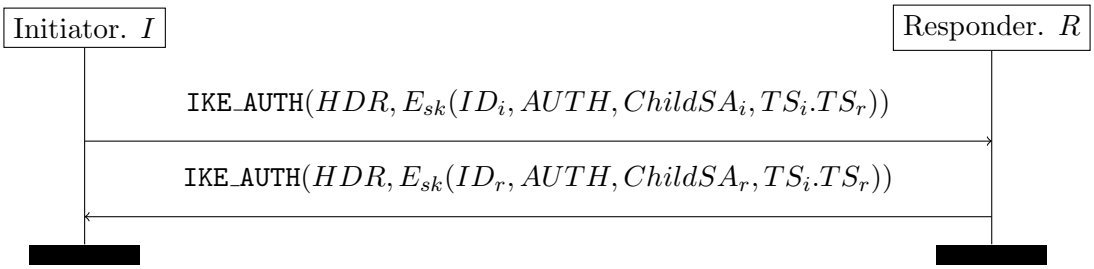
The second pair of messages (IKE_AUTH) is used to perform the authentication of the previous messages, exchange identities (and optionally certificates) and establish the first Child SA within the created IKE SA. Parts of these messages (see *Section 2.10*) are encrypted and integrity protected with the keys established through the IKE_SA_INIT exchange. In this way IPSec protects the identities from eavesdroppers and all the fields in all the messages are authenticated. A MITM attacker who cannot complete the IKE_AUTH exchange cannot see the identity of the initiator.

The initiator sends:

- the header containing the initiator and responder's SPIs.
- An encrypted payload containing:
  - its identity
  - the Certificate (if requested, usually with the "Hash and URL" technique)
  - a Certificate Request (optional)
  - the responder's identity it wants to talk with (optional)
  - the authentication payload used for integrity protection and identity validation
  - the proposals for the Child SA to be generated within the Security Association Payload
  - the proposals for the Traffic Selectors

The responder sends:

- the header containing the initiator and responder's SPIs.
- An encrypted payload containing:
  - its identity
  - the Certificate (if requested, usually with the "Hash and URL" technique).
  - the authentication payload for integrity protection and identity validation
  - the chosen proposal for the new Child SA which is going to be created.
  - the chosen Traffic Selectors.



At this point an IKE SA and a Child SA were created. If the peers want to create new Child SAs, they can use the CREATE_CHILD_SA exchange. Moreover, that exchange can be used for 3 different purposes:

- Creating a new Child SA
- Rekeying an IKE SA
- Rekeying a Child SA

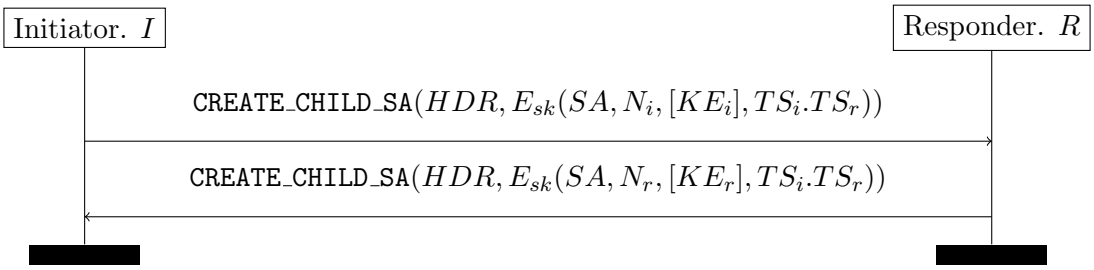
The rekeying process is explained in *Section 2.6*.
As for the creation of new Child SAs, the initiator sends:

- the header containing the initiator and responder's SPIs.

- An encrypted payload containing:
  - SA proposals (cryptographic suites to be used to secure the new Child SA).
  - A Nonce
  - Optionally a new Diffie-Hellman Public Key (frequently changing that value is suggested in order to increase the security of the protocol - it is called Ephemeral Diffie-Hellman).
  - the proposed Traffic Selectors for the new Child SA

The responder sends:

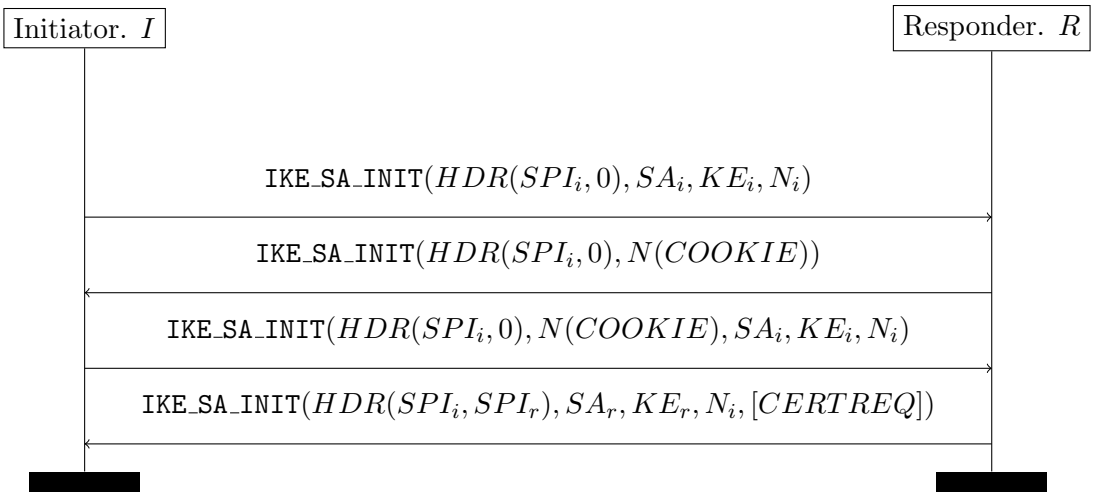
- the header containing the initiator and responder's SPIs.
- An encrypted payload containing:
  - the accepted offer for the SA
  - a Nonce
  - optionally a new Diffie-Hellman Public Key
  - the chosen Traffic Selectors



At the end of this exchange, a new Child SA is instantiated and ready to be used.

3.2 Retransmission with Cookies

When a responder detects a large number of half-open IKE SAs, it should reply with a sligthly different IKE_SA_INIT message. The new message, with the same message ID will contain a Cookie Notification (see *Section 2.5*). When the initiator receives an IKE_SA_INIT message containing a Cookie it must retry the previous IKE_SA_INIT request including as the first payload the received Cookie and all the other payloads unchanged with respect to the first request. Even the sequence number will be always the same, in particular it will be 0 for all the four messages.



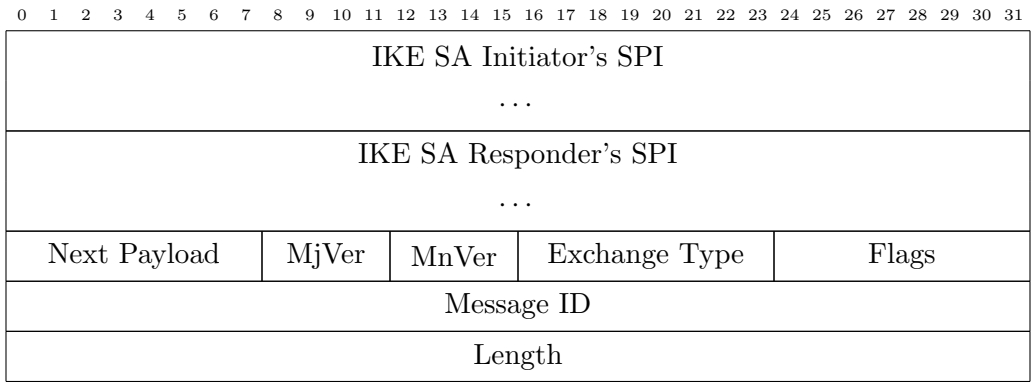
4 IKEv2: Packets' Format

In this section, the format of some of the packets exchanged between the peers are explained in detail. In particular, every field will be associated with a brief description.

4.1 Header Format

There are two header formats involved: one is used for the IKE messages, the other one for the common exchanges.

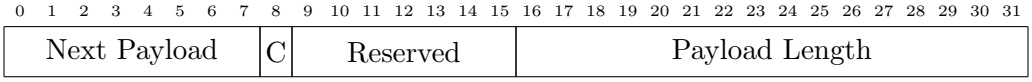
IKE Header



- Initiator's SPI (8 bytes): value chosen by the initiator to identify a unique IKE Security Association. It cannot be 0.
- Responder's SPI (8 bytes): value chosen by the responder to identify a unique IKE SA. In the first message (from the initiator to the responder) it must be 0 because it is not known to the initiator.
- Next Payload: the type of payload which follows the header

- Major Version: the major version of the IKE protocol in use
- Minor Version: the minor version of the IKE protocol in use.
- Exchange Type: indicates the type of message being used:
  - 34: IKE_SA_INIT
  - 35: IKE_AUTH
  - 36: CREATE_CHILD_SA
  - 37: INFORMATIONAL
- Flags: Options for the message
- Message ID: the message identifier used to control retransmission and prevent replay atacks.
- Length: bytes length of the whole message (header + payload)

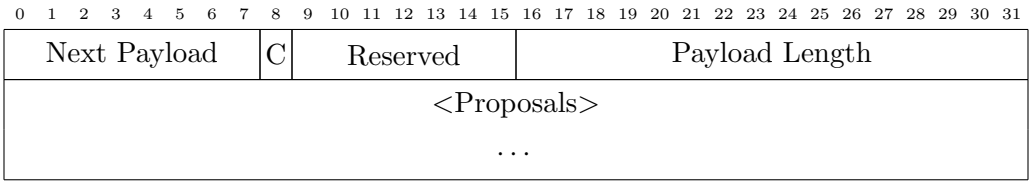
Generic Header



- Next Payload: the type of the first payload which follows the header
- Critical (C - 1 bit):
  - 0: the sender wants the recipient to skip this payload if it is not able to understand the payload type code in the Next Payload field.
  - 1: the sender wants the recipient to reject all the message if it is not able to understand the payload type in the Next Payload field.
- Reserved (7 bits)
- Payload Length (2 bytes): length of the current payload plus the length of the header

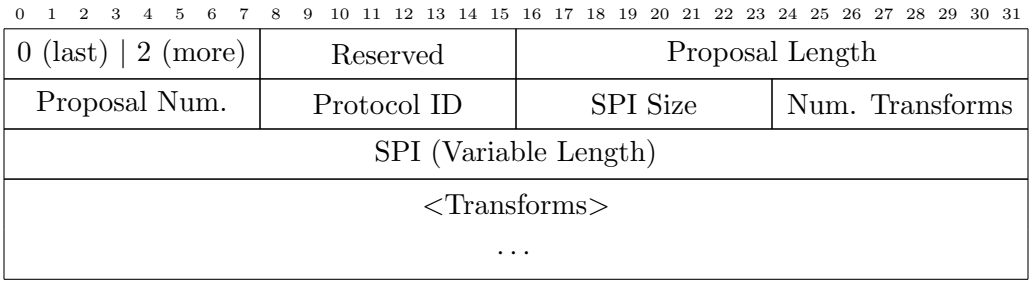
4.2 Security Association (SA) Payload Format

The security association payload is used within the IKE SA and Child SA exchanges to negotiate the attributes of the Security Associations. An SA payload can contain multiple proposals, ordered from the most preferred to least preferred. Each proposal contains a single protocol, which can contain multiple transforms. Each transform can contain multiple attributes. All this things are nested such that the Payload Length of an SA Payload includes the combined contents of the SA, Proposals, Transforms and Attributes. With respect to IKEv1, the syntax of Security Associations, Proposals, Transforms and Attributes is based on ISAKMP with differences in the semantics in order to allow for multiple possible combinations.



- Proposals: one or more proposal substructure (explained in the next paragraph)

Proposal Substructure



- 0 (last) or 2 (more): specifies if this is the last proposal substructure or not in the SA
- Reserved
- Proposal Length (2 bytes): length of the proposal (including transforms and attributes)
- Proposal Number (1 byte): the first proposal will have number 1 associated. The subsequent proposals must be one more than the previous one.
- Protocol ID: specifies the IPSec protocol for the current negotiation
  - 1: IKE
  - 2: AH
  - 3: ESP
- SPI Size (1 byte): initially it is 0, whilst in the subsequent negotiations it is the byte length of the SPI of the corresponding protocol (8 for IKE, 4 for ESP and AH)
- Number of Transforms: the number of transforms within this proposal
- SPI (variable length): sending peer's SPI
- Transforms: one or more transform substructure (explained in the next paragraph)

Transform Substructure

0 1 2 3 4 5 6 7	8 9 10 11 12 13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
0 (last) | 3 (more)	Reserved	Tranform Length
Transform Type	Reserved	Tranform ID
<Transform Attributes> ...		

- 0 (last) or 3 (more): specifies whether this is the last transform substructure or not in the proposal
- Reserved
- Transform Length: the byte length of the Transform (including the attributes)
- Transform Type: the type of transform. Different Protocols support different transform types. IKE's transform types are:
  - 1: Encryption Algorithm (ENCR)
  - 2: Pseudorandom function (PRF)
  - 3: Integrity Algorithm
  - 4: Diffie-Hellman group
- Reserved
- Transform ID: identifies the specific instance of the Transform Type being proposed. Some of the IDs for each transform type are shown in *Section 7.1.*
- Transform Attributes: additional substructure where the characteristics of the trasform are specified.

An example of Security Association Payload (shown as a tree) is the following:

SA Payload

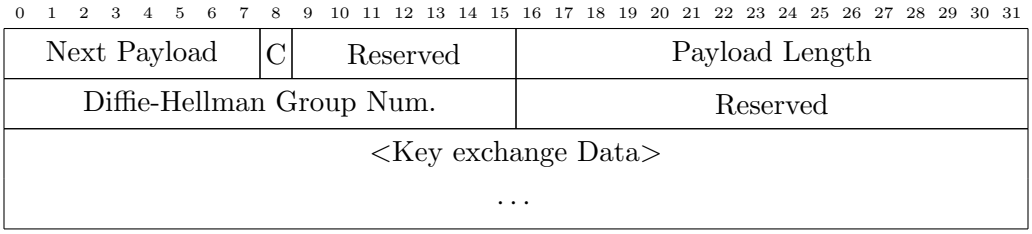
- **Proposal** {0, Proposal Num = 1, Protocol ID = 3 (ESP), SPI Size = 4, Num. Transforms = 4}
  - **Transform** {3, Transform Type = 1 (ENCR), Transform ID = 12 (ENCR_AES_CBC)}
    - Attribute {Key Length = 128 }
  - **Transform** {3, Transform Type = 1 (ENCR), Transform ID = 12 (ENCR_AES_CBC)}
    - Attribute {Key Length = 192 }
  - **Transform** {3, Transform Type = 3 (INTEG), Transform ID = 7 (AUTH_HMAC_SHA1_160)}
  - **Transform** {0, Transform Type = 3 (INTEG), Transform ID = 8 (AUTH_AES_CMAC_96)}

If there are multiple transforms with the same transform type, the proposal is an OR of those Transforms. If there are multiple transforms with different Transform Types, the proposal is an AND of the different types. In the latter example there are 2 transforms of type 1 (Encryption

Algorithm) and 2 of type 3 (Integrity Algorithm). That means we must choose one algorithm from the first group and one from the second group. As you probably noticed, a transform can have one or more attributes. This is necessary when an algorithm can be used in more than one way. If we want to propose the same algorithm with different attributes, we must use multiple transforms of the same type and with the same Transform ID, but with different attributes payload.
During Security Association negotiation, initiators present offers to responders which must select a complete set of algorithms and parameters from the offers. If multiple proposals are offered, the responder should choose only a single one. Notice that the initiator sends the offers ordered from the most preferred to the least preferred.

A slightly different procedure can happen when negotiating the Diffie-Hellman Group. The initiator should guess the responder's preferred group in order to proceed. If it does not happen in the first message, the responder must answer with a value in the correct group and with the indication of what is the correct group to be used.

4.3 Key Exchange Payload Format



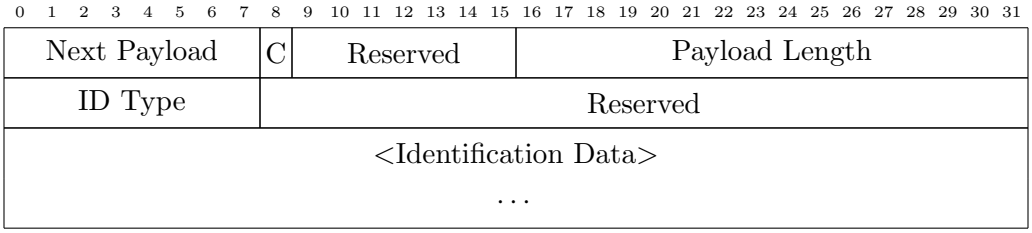
- Diffie-Hellman Group Number: it identifies the Diffie-Hellman group in which the Key Exchange take place. This group must match a Diffie-Hellman group in a proposal of the Security Association payload that is sent in the same message. If none of the proposals in the Security Association payload specificed a group, the Key Exchange Payload must be omitted, meaning that the two peers will not use the Diffie-Hellman key exchange to generate the keys. The IDs of these groups are shown in *Section 7.2*.

- Key Exchange Data: the public Diffie-Hellman key of one peer which has to be of the same length of the modulus of the chosen group. If necessary, it can be prepended with zero bits in order to make it of the correct length.

4.4 Identification Payload Format

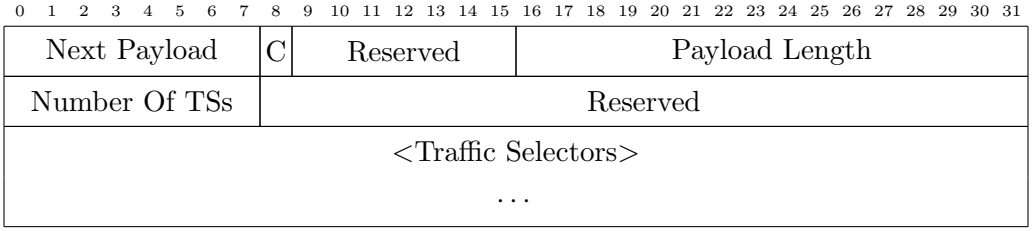
The identification payload is used by peers to assert the identity of the other party. In particular, thanks to authentication mechanisms, one peer can assert the identity of the other one and activate appropriate access control policies. It is important to notice that the content of the identification payloads is used only to fetch the policy and the authentication data related to the other party, then, if IP addresses are used, it is not required to match the IP Address specificed in the IP Header. The link between the SPD entries and the IKE Security Association Management is provided by an additional database (commonly not citied): the PAD (Peer Authentication Database) is aimed

to describe the use of the identification payload in IKEv2 and to provide a formal model for the binding of identities to policies.



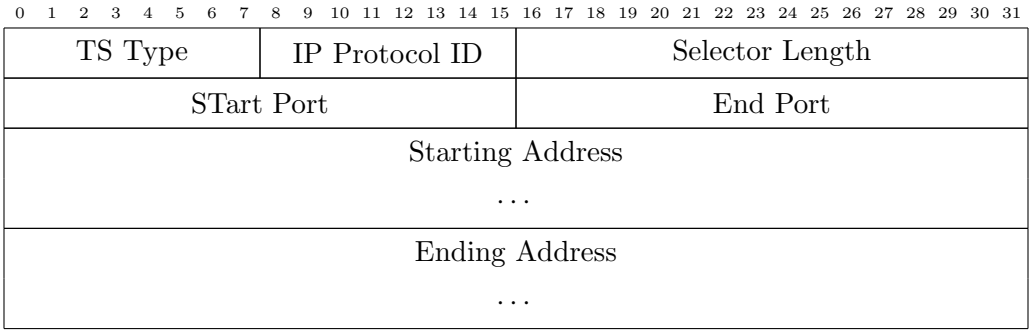
4.5 Traffic Selector Payload Format

Traffic Selector Payloads allow peers to identify traffic flows to be processed by IPSec Services which will apply the appropriate security measures and policies.



- Number of TSs (1 byte): number of traffic selectors being provided
- Reserved
- Traffic Selectors: one or more Traffic Selector Substructures (explained in the next paragraph)

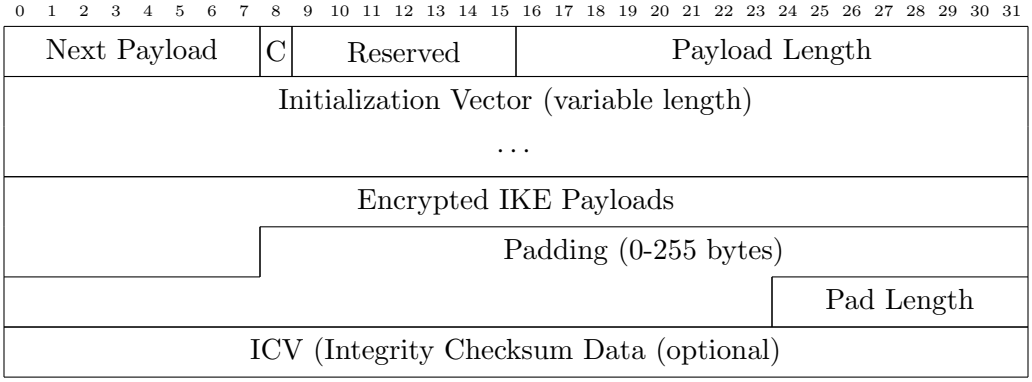
Traffic Selector Substructure



- TS Type (1 byte): it specifies the type of of Traffic Selector (*TS_IPV4_ADDR_RANGE* or *TS_IPV6_ADDR_RANGE*)
- IP Protocol ID (1 byte): 0 means that the protocol is not relevant and the peer accepts all protocols. Other values for UDP, TCP and ICMP.

- Selector Length: the length of the entire substructure, inclunding the header
- Start Port: the smallest port number allowed by the Traffic Selector
- End Port: the largest port number allowed by the Traffic Selector
- Starting Address: the smallest address allowed by the Traffic Selector
- Ending Address: the largest address allowed by the Traffic Selector

4.6 Encrypted Payload Format



- Next Payload: the payload type of the first embedded payload. Remember that this kind of payload is the last one and it is linked to the first one as a circle in that way.
- Payload Length: the length of the entire payload (header, IV, Payloads, Padding, Pad Length and ICV)
- Initialization Vector: senders must select a new unpredictable IV for every message in order to guarantee security. The length of this field depends on the chosen algorithms.
- Padding: It is used to make the combination of payloads, padding and pad length a multiple of the encryption block size specified by the chosen algorithm.
- Integrity Checksum Data: is the cryptographic checksum of the entire message starting with the fixed IKE header through the pad length. It must be computed over the encrypted data and its length depends on the negotiated integrity algorithm.

5 Security Measures and Considerations

5.1 Disclosure of configuration information

The protocol is designed to minimize the disclosure of configuration information. It is unavoidable during the first phase and exchanges (IKE SA Init) when the two parties need to discover what IKE version to use and what cryptographic suites are available and supported.

5.2 Key Strength

The strength of IKE key derived from a Diffie-Hellman exchange depends on the inherent strength of the chosen group, on the size of the exponenet and on the entropy provided by the Random Number Generators used (PRFs). Then, even if we choose the strongest Diffie-Hellman group, the strength of the result is limited by the inherent strength of the other components (negotiated algorithms: PRF, etc..) involved in its computation. The security of this protocol is then critically dependent on the randomness of the randomly chosen parameters, which should be generated from a pure random (or at least seeded pseudorandom) source.
As discussed before, IKE can work with pre-shared keys too. That means that the implementers MUST take care about the randomness of these secrets. Deriving a shared secret from a password, name, or other low-entropy sources should not be permitted.

A common practice (that MUST be implemented) is to repeat the Diffie-Hellman exchanges during rekeying because the repeated use of the same key represents a huge vulnerability, in the sense that all SAs created with that key would become vulnerable to cryptoanalysis of a single key. Implementers should set a limit on CREATE_CHILD_SA exchanges between exponentiations.

5.3 Disclosure of Internal IP Addresses

When using IPSec with NAT Networks, the internal addresses are hided by hashing the IP and the associated ports. In the case of IPv4, the length of such kind of addresses is 32 bits, then it is possible to an attacker to perform a bruteforce attack and discover what is the internal IP Address ($O(2^{32})$). With an educated guess the attack would become more efficient by reducing the number of computed hashes. That is inhevitable if IPv4 is used.

5.4 Non-key generating EAP method and use of tokens

When using EAP authentication method that does not generate a shared key for protecting AUTH payloads, MITM attacks and impersonations are possible. For example, if it generates an authentication token instead of a shared key, it could be intercepted by an attacker which would use it in order to impersonate one of the parties or to perform a MITM attack. Then, use of non-key generating EAP methods should be avoided where possible. If needed, this method can be used by using a protected tunnel where the initiator will be able to validate the certificate of the responder before starting the EAP authentication.

5.5 Fragmentation and DDOS attacks

Fragmentation is a possible attack vector for a Denial-Of-Service attack. If possible, it should be avoided. This problem usually happens when the two parties exchange directly their certificates. A common practice, which is used to reduce the chances of a DOS attack, is to apply the "Hash and URL" encoding instead of directly sending the whole certificate. Indeed, the attackers could prevent the exchanges from completing by exhausting the buffer used for reassamble the fragments.

5.6 Replay Attacks

An attacker could intercept some messages and try to resend them to that party inducing it to perform past exchanges: for example, it is common to use a replay attack to induce the other party to reuse previous (maybe discovered) keys. IKEv2 solves that issue by using Message IDs (sequence numbers) together with Sliding Windows of dinamically changed sizes and cookies in order to keep track of the already received messages. That system is discussed in Chapter 2.

6 Known Attacks

Common attacks against IKEv2 implementations are:

- DDOS attacks
- Key-Reusing attacks
- Wrong configuration of the protocol

As you can notice, the protocol itself is not vulnerable, or better, it was proved to be secure thanks to Tamarin and ProVerif. The vulnerabilities comes from erroneous configurations of the protocol by the administrators. Fortunately, IKEv2 disables by default weak Diffie-Hellman groups and weak algorithms, but a lot of homemade implementations are yet vulnerable cause they do not follow all the constraints. One example is the SLOTH attack, which was born only for TLS implementations. Wrong configurations of IKEv2 (or IKEv1) can raise to its application to IPSec Protocol too, in particular by performing a MITM attack. It is not considered a vulnerability of the Internet Key Exchange protocol because it relies on the wrong choices of the administrator, which can enable weak algorithms and policies.

7 Appedix

7.1 Cipher Suite Choices

Transform Type 1 (Encryption Algorithms)

Name	ID
ENCR_DES_IV64	1
ENCR_DES	2
ENCR_3DES	3
ENCR_RC5	4
ENCR_IDEA	5
ENCR_CAST	6
ENCR_BLOWFISH	7
ENCR_3IDEA	8
ENCR_DES_IV32	9
ENCR_NULL	11
ENCR_AES_CBC	12
ENCR_AES_CTR	13
ENCR_AES_CCM_8	14
ENCR_AES_CCM_12	15
ENCR_AES_CCM_16	16
ENCR_AES_GCM_8	18
ENCR_AES_GCM_12	19
ENCR_AES_GCM_16	20

Transform Type 2 (Pseudorandom functions)

Name	ID
PRF_HMAC_MD5	1
PRF_HMAC_SHA1	2
PRF_HMAC_TIGER	3
PRF_AES128_XCBC	4
PRF_HMAC_SHA2_256	5
PRF_HMAC_SHA2_384	6
PRF_HMAC_SHA2_512	7
PRF_AES128_CMAC	8
PRF_HMAC_STREEBOG_512	9

Transform Type 3 (Integrity Algorithms)

Name	ID
AUTH_HMAC_MD5_96	1
AUTH_HMAC_SHA1_96	2
AUTH_DES_MAC	3
AUTH_KPDK_MD5	4
AUTH_AES_XCBC_96	5
AUTH_HMAC_MD5_128	6
AUTH_HMAC_SHA1_160	7
AUTH_AES_CMAC_96	8
AUTH_AES_128_GMAC	9
AUTH_AES_192_GMAC	10
AUTH_AES_256_GMAC	11
AUTH_HMAC_SHA2_256_128	12
AUTH_HMAC_SHA2_384_192	13
AUTH_HMAC_SHA2_512_256	14

Transform Type 4 (Diffie-Hellman groups) See next section.

7.2 Diffie-Hellman groups

There are multiple Diffie-Hellman groups that can be used with IKEv2. Newer implementations tend to use Elliptic Curve's groups in order to reach higher security with shorter keys. The standard defines 30 groups, identified by their ID (used in the configuration process).

ID	Group	Scope
1	768 bit prime	IKEv1 - IKEv2 Deprecated Commonly used with DES
2	1024 bit prime	IKEv1 - IKEv2 Deprecated
3	EC group over $GF(2^{155})$	Not available for use in modern IKE implementations.
4	EC group over $GF(2^{185})$	Not available for use in modern IKE implementations.
5	1536 bit prime	IKEv1 - IKEv2 Deprecated
6	EC group over $GF(2^{163})$	Not available for use in modern IKE implementations.
7	EC group over $GF(2^{163})$	IKEv1 Deprecated
8	EC group over $GF(2^{283})$	Not available for use in modern IKE implementations.
9	EC group over $GF(2^{283})$	Not available for use in modern IKE implementations.

10	EC group over $GF(2^{409})$	Not available for use in modern IKE implementations.
11	EC group over $GF(2^{409})$	Not available for use in modern IKE implementations.
12	EC group over $GF(2^{571})$	Not available for use in modern IKE implementations.
13	EC group over $GF(2^{571})$	Not available for use in modern IKE implementations.
14	2048 bit prime	IKEv2 MINIMIUM Acceptable
15	3072 bit prime	Not available for use in modern IKE implementations.
16	4096 bit prime	Not available for use in modern IKE implementations.
17	6144 bit prime	Not available for use in modern IKE implementations.
18	8192 bit prime	Not available for use in modern IKE implementations.
19	256-bit random elliptic curve	IKEv2
20	384-bit random elliptic curve	IKEv2
21	521-bit random elliptic curve	IKEv2
22	1024-bit modulus with 160-bit prime order subgroup	Not available for use in modern IKE implementations.
23	2048-bit modulus with 224-bit prime order subgroup	Not available for use in modern IKE implementations.
24	2048-bit modulus with 256-bit prime order subgroup	IKEv2
25	192-bit Random ECP Group	Not available for use in modern IKE implementations.
26	224-bit Random ECP Group	Not available for use in modern IKE implementations.
27	224-bit Random ECP Group	Not available for use in modern IKE implementations.
28	256-bit Brainpool ECP group	Not available for use in modern IKE implementations.
29	384-bit Brainpool ECP group	Not available for use in modern IKE implementations.
30	512-bit Brainpool ECP group	Not available for use in modern IKE implementations.

References

[1] https://it.wikipedia.org/wiki/IPsec

[2] https://it.wikipedia.org/wiki/Internet_key_exchange

[3] https://www.rfc-editor.org/rfc/rfc5996

[4] https://tex2e.github.io/rfc-translater/html/rfc7296.html

[5] https://access.redhat.com/blogs/product-security/posts/sloth