

# ISAKMP/Oakley White Paper

## ***Industrial strength key exchange***

ISAKMP/Oakley, the IPsec group's answer to protocol negotiation and key exchange through the Internet, is actually a hybrid protocol. It integrates the Internet Security Association and Key Management Protocol (ISAKMP) with a subset of the Oakley key exchange scheme. This combined technique is now referred to as Internet Key Exchange (IKE).

ISAKMP/Oakley provides a way to

- agree on which protocols, algorithms, and keys to use (negotiation services) ensure from the beginning of the exchange that you're talking to whom you think you're talking to (primary authentication services)
- manage those keys after they've been agreed upon (key management)
- exchange those keys safely.

When hosts on a network communicate using IPsec, they create Secure Associations (SA) for session management. Key exchange is a closely related service to SA management. When you need to create an SA, you need to exchange keys. So ISAKMP/Oakley wraps them both up together, and delivers them as an integrated package.

## ***Manual key exchange***

There is one other way to exchange keys. IPsec specifies that compliant systems support manual keying as well. That means if you wish to use manual (face-to-face) key exchange for certain situations, you still can. But IPsec's designers also assume that in most situations, for most large enterprises, this would be impractical. So ISAKMP/Oakley, the safe way to negotiate SAs and exchange keys through public networks, will probably do most of the work for most of the world.

## ***ISAKMP/Oakley Phases***

ISAKMP/Oakley functions in two phases. In phase one, two ISAKMP peers establish a secure channel for doing ISAKMP (called the ISAKMP SA). In phase two, those two peers negotiate general purpose SAs.

An ISAKMP peer is an IPSEC-compliant node capable of establishing ISAKMP channels and negotiating SAs. It might be the computer on your desktop, or something called a security gateway that negotiates security services for you.

## ***ISAKMP/Oakley modes***

Oakley provides three modes of exchanging keying information and setting up ISAKMP SAs—two for ISAKMP phase one exchanges, and one for phase two exchanges.

### **Main mode –**

Main mode accomplishes a phase one ISAKMP exchange by establishing a secure channel.

## **Aggressive mode –**

Aggressive mode is another way of accomplishing a phase one exchange — it's a little simpler and a little faster than main mode, and does not provide identity protection for the negotiating nodes, as they must each transmit their identity before having negotiated a secure channel through which to do so.

## **Quick mode –**

Quick mode accomplishes a phase two exchange by negotiating an SA for general-purpose communications.

ISAKMP/Oakley actually has one other mode, called new group mode, which doesn't really fit into phase one or phase two (see Diffie-Hellman in this paper).

## ***Establishing a secure channel for negotiation***

To establish an ISAKMP SA, the initiating node proposes five things:

- an encryption algorithm (to protect data)
- a hash algorithm (to reduce data for signing)
- an authentication method (for signing data)
- information about a group over which to do a Diffie-Hellman exchange
- a pseudo-random function (PRF) used to hash certain values during the key exchange for verification purposes (this is optional, you can also just use the hash algorithm)

## ***Perfect forward secrecy***

One of the most annoying things about passing encrypted data around a public network is the number of opportunities an attacker has to get hold of encrypted material. You can reduce the risk of their ever deciphering it by using larger and larger keys. But the larger the key, the slower and more complex the encryption and this can impair network performance.

A good compromise solution is to use reasonably large keys, and to keep changing them. But this presents difficulties too. You need ways to generate those new keys so that the person at the other end can agree on it as well. But, to generate your new key, you can't use either the key you're changing from, or material used to generate the key you're changing from.

The point being that if you do, and then someone gets ahold of the current key, that person can then easily deduce your new key. What you need is a method of generating a new key that is in no way dependent on the value of the current key. So if someone gets hold of your current key, it only gives them a small part of the overall picture, and they would have to break yet another entirely unrelated key to get the next part. Cryptographers call this concept "perfect forward secrecy". And yes, there are ways of doing this through the net. The method ISAKMP/Oakley uses is called Diffie-Hellman.

## ***Diffie-Hellman***

A Diffie-Hellman exchange works like this: two people independently and randomly generate values much like a public/private key pair. Each sends their public value to the other (using

authentication to close out the man-in-the-middle). Each then combines the public key they received with the private key they just generated, using the Diffie-Hellman combination algorithm.

The resulting value is the same on both sides, and therefore can be used for fast symmetric encryption by both parties. But no one else in the world can come up with the same value from the two public keys passed through the net, since the final value also depends on the private values, which remain secret.

You can use the derived Diffie-Hellman key either as a session key for subsequent exchanges, or to encrypt yet another randomly generated key, which you can then pass through the net quite safely.

Note that yes, you do need authentication to protect even Diffie-Hellman exchanges against the man-in-the-middle. Diffie-Hellman alone does not solve this problem. It would be complicated, but without authentication a man-in-the-middle could use an active attack to get in on the action and plant his own keys.

But if the key exchange mechanism you're using is protected by an authentication scheme, Diffie-Hellman allows you to generate new shared keys to use for symmetric encryption which are independent of older keys—providing perfect forward secrecy. And since symmetric encryption techniques are a lot faster, this can be quite useful in network communications.

You may wish to agree on a few things to do a Diffie-Hellman exchange in the first place. That's what the Diffie-Hellman parameter in the ISAKMP SA is for.

The parameter contains information on a group to perform the Diffie-Hellman exchange. The group consists of generation material used for coming up with keys. You need two numbers: a large known prime number, and a seed.

By default, ISAKMP/Oakley specifies two groups that are always available for selection by both parties. If they wish to add new groups to select from, they may do so at their option. And that's what the ISAKMP/Oakley new group mode is used for.

## ***The pseudo-random function***

The pseudo-random function (PRF) is also probably worth explaining briefly. In the context of IPsec, a PRF is really just another name for a hash message authentication code (HMAC) function.

Providing a way to check the integrity of information transmitted over or stored in an unreliable medium is a prime necessity in the world of open computing and communications. Mechanisms that provide such an integrity check based on a secret key are usually called "message authentication codes" (MAC). Typically, message authentication codes are used between two parties that share a secret key in order to validate information transmitted between these parties.

The PRF may be negotiated, but if it is not, the HMAC version of the negotiated hash algorithm is used as a pseudo-random function (as defined in IETF RFC 2104). HMAC can be used with any iterative cryptographic hash function, e.g., MD5, SHA-1, in combination with a secret shared

key. The cryptographic strength of HMAC depends on the properties of the underlying hash function. In ISAKMP/Oakley, you can use the PRF both for authentication purposes and to generate additional key material (as a randomizer). Keys need to be chosen at random or using a cryptographically strong pseudo-random generator seeded with a random seed. Periodically the keys must be refreshed.

## ISAKMP/Oakley modes

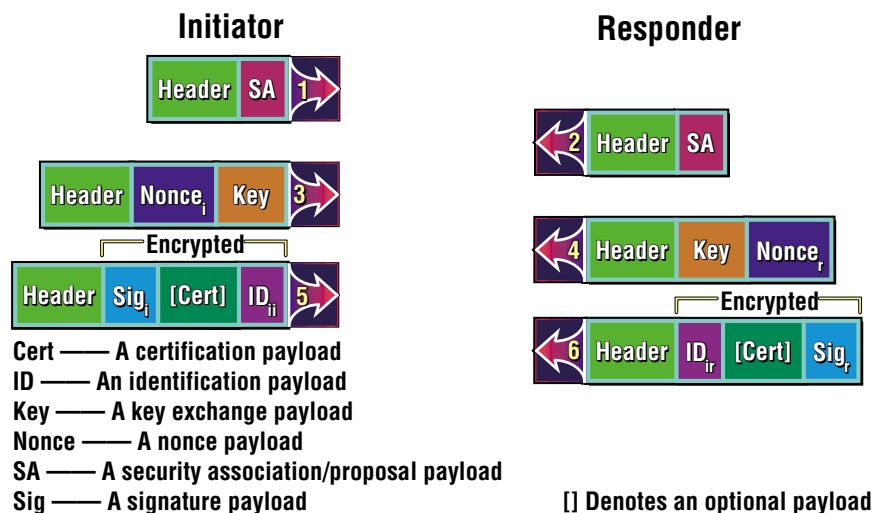
### Main mode

Main mode provides a mechanism for establishing the first phase ISAKMP SA, which is used to negotiate future communications. Remember, the objective here is to agree on enough things (authentication and confidentiality algorithms, hashes, and keys) to be able to communicate securely long enough to set up an SA, which will be used for future communication. The steps in full will be

- Use main mode to bootstrap an ISAKMP SA for temporary communication
- Use quick mode within that ISAKMP SA to negotiate a general SA
- Use that SA to communicate from now until it expires

The first step, securing an ISAKMP SA using Main mode, occurs in three two-way exchanges between the SA initiator and the recipient. In the first exchange (1 and 2 in the illustration below), the two agree on basic algorithms and hashes. In the second (3 and 4 in the illustration below), they exchange public keys for a Diffie-Hellman exchange, and pass each other a nonce — a random number the other party must sign and return to prove their identity. In the third (5 and 6 in the illustration below), they verify those identities, and that exchange has gone through.

So the following is how ISAKMP/Oakley establishes its own ISAKMP SA, step by step, using Main mode, and established digital signatures for authentication.



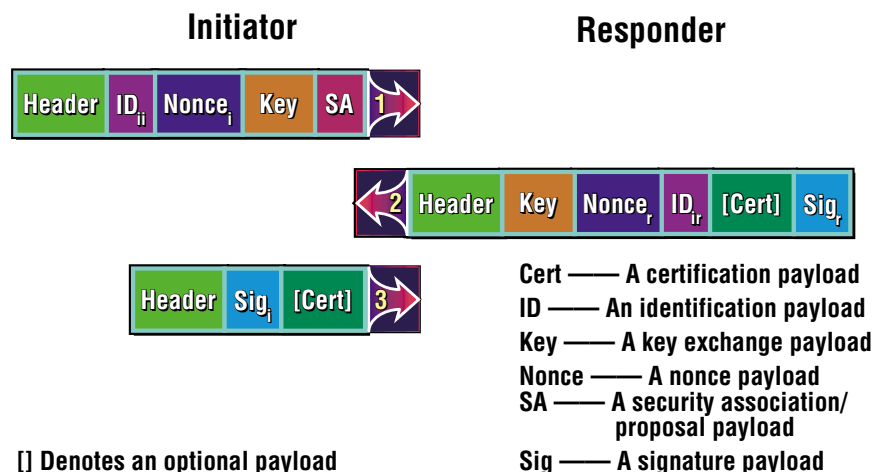
In all of these steps, an ISAKMP header preceding the rest of the packet identifies the step being taken. Each of the pieces is carried in its own payload, but you can pack any number of these payloads into a single ISAKMP packet.

The parties actually use the shared key in three permutations, once they derive it. Both parties have to hash it three times—generating first a derivation key (to be used later for generating additional keys in Quick mode), then an authentication key (for authentication), and then, finally the encryption key to be used for the ISAKMP SA.

## Aggressive mode

Aggressive mode provides the same services as Main mode. It establishes the original ISAKMP SA. It looks much the same as Main mode except that it is accomplished in two exchanges, rather than three, with only one round trip, and a total of three packets rather than six.

In aggressive mode, the proposing party generates a Diffie-Hellman pair at the beginning of the exchange, and does as much as is practical with that first packet—proposing an SA, passing the Diffie-Hellman public value, sending a nonce for the other party to sign, and sending an ID packet which the responder can use to check their identity with a third party. The responder then sends back everything needed to complete the exchange—really an amalgamation of all three response steps in Main mode, and all that's left for the initiator to do is to confirm the exchange.



The end result is that an Aggressive mode exchange attains the same goal as a Main mode exchange, except that Aggressive mode does not provide identity protection for the communicating parties. That is to say, in Aggressive mode, the parties exchange identification information prior to establishing a secure SA in which to encrypt it. So someone monitoring an aggressive exchange can actually identify who has just formed a new SA.

The advantage of Aggressive mode, however, is speed.

## Quick mode

Once two communicating parties have established an ISAKMP SA using Aggressive mode or Main mode, they can use Quick mode.

Quick mode has two purposes: negotiating general IPSec security services, and generating fresh keying material.

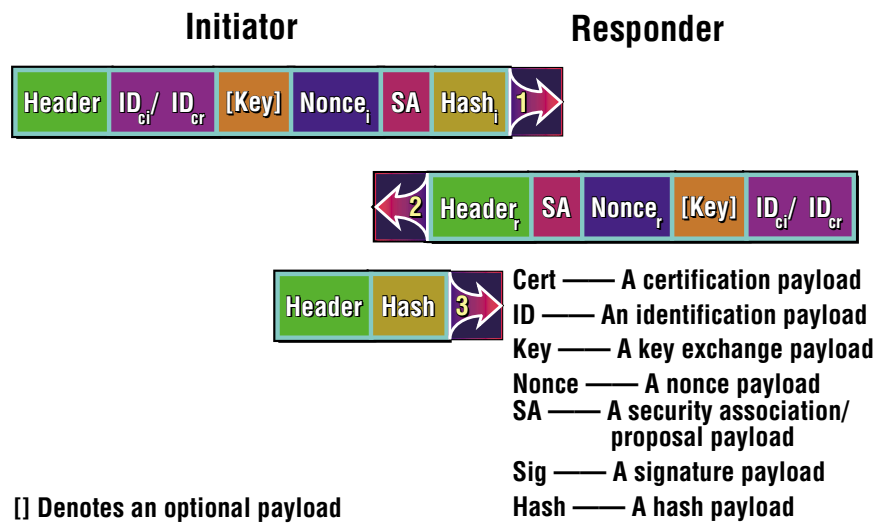
Quick mode is considerably simpler than either Main or Aggressive mode. Since it's already inside a secure tunnel (every packet is encrypted), it can also afford to be a little more flexible.

Quick mode packets are always encrypted, and always start with a hash payload. The hash payload is composed using the agreed-upon PRF and the derived authentication key for the ISAKMP SA. The hash payload is used to authenticate the rest of the packet. Quick mode defines which parts of the packet are included in the hash.

Key refreshing can be done one of two ways. If you don't want or need perfect forward secrecy, Quick mode can just refresh the keying material already generated (in Main or Aggressive mode) with additional hashing. The two communicating parties can exchange nonces through the secure channel, and use these to hash the existing keys.

If you do want perfect forward secrecy, you can still request an additional Diffie-Hellman exchange through the existing SA and change the keys that way.

Basic Quick mode is a three-packet exchange, like Aggressive mode.



If the parties do not require perfect forward secrecy, the initiator sends a packet with the Quick mode hash, with proposals and a nonce. The respondent then replies with a similar packet, but generating their own nonce and including the initiator's nonce in the Quick mode hash for confirmation. The initiator then sends back a confirming Quick mode hash of both nonces, completing the exchange. Finally, both parties perform a hash of a concatenation of the nonces, the SPI, and the protocol values from the ISAKMP header that initiated the exchange, using the derivation key as the key for the hash. The resulting hash becomes the new password for that SA. If the parties do require perfect forward secrecy, the initiator first generates a public/private key pair, and sends the public key along with the initiation packet (along with the hash and nonce). The recipient then responds with their own public key and nonce, and both parties then generate the shared key using a Diffie-Hellman exchange—again fully protected by the Quick mode hashes, and by full encryption within the ISAKMP SA.

## ***Negotiating the SA***

After all those exchanges to generate the ISAKMP SA, actually establishing the general purpose SA is relatively simple.

To generate a new SA, the initiator sends a Quick mode message through the ISAKMP SA requesting the new SA. A single SA negotiation actually results in two SAs — one inbound, to the initiator, and one outbound. Each IPsec SA is one way and the node on the receiving end of that SA always chooses its own SPI to ensure that's the only SA for which it's using that reference.

So, using Quick mode, the initiator tells the respondent which SPI to use in future communications with it, and the respondent follows up with its own selected SPI.

Each SPI, in concert with the destination IP address, uniquely identifies a single, IPsec SA.

However, these SAs are always formed in pairs (inbound and outbound), and these pairs have identical parameters (keys, authentication and encryption algorithms and hashes)—apart from the SPI itself.

## ***So how do you know who is who?***

We're almost to the end of explaining how the various parts of IPsec work together to make your communications safe. But we haven't yet explained how you verify that people are who they say they are in the first place.

Which brings us to the third party.

## ***Certificates and third party authentication***

IPsec's sophisticated authentication infrastructure offers the most robust and scaleable approach to authentication yet devised: certificates and certificate authorities.

Using certificates delegates authentication services to specific 'certificate authority' servers, from which security administrators broadcast authentication information—with which nodes must identify themselves.

IPsec certificate authorities use industry standard X.500 database servers to serve their information, and take advantage of X.500's built-in support for distributed databases to make setting up large, decentralized authentication server networks easy.

Certificates solve once intractable scalability problems, and further insulate users and security administrators from the complex inner working of authentication and encryption algorithms, making even complex secure VPNs involving multiple trading partners and millions of users easy and intuitive to maintain.

## ***Shared secret authentication for smaller secure VPNs***

For smaller, less complex secure VPNs, IPSec specifies simpler 'shared secret' authentication options. Shared secrets allow you to set up the secure VPN without a certificate infrastructure.

## ***Joe sent me – third party verification via a CA***

The final component of the IPSEC-compliant secure VPN, in most implementations, is something called a Certificate Authority, or CA.

A CA is a trusted third party—someone whose identity you can already prove, and who can vouch for the identity of people with whom you're trying to communicate.

The idea is easy enough to relate to the real world. The CA is like a public figure (for example, a notary public) that you know well enough to trust, and who vouches for other people.

In the world of online verification, the CA software issues certificates tying three things together:

- someone's identity (anything that uniquely identifies them to you in a meaningful way so you can't confuse them with someone else (i.e. their name and place of residence))
- the public key that person uses to sign their identity to online documents
- the CA's public key (used to sign and authenticate the entire package)

The CA is the defense against that man-in-the-middle working his way into key exchanges.

Whenever you initiate an exchange with someone, he has to sign it with his digital signature. And then you in turn can check that signature against the one on record with the CA. They have to match. You then check that certificate's signature with the CA's signature. They have to match too.

What stops the man-in-the-middle from forging the CA's signature, and turning out his own certificates? Mostly smart distribution. Because it's a highly public entity, and a highly useful value, the CA's public key can be on record in a number of places—making it difficult for anyone to get away with forging it.

You also usually design the CA's signature with strong encryption and an extremely large key-space.

This makes it long lived, so you can distribute it any number of ways, including on the disk on which the verification software is distributed. It makes it difficult indeed (provided the system is vigilant about checking) to introduce counterfeit signatures into the system.

ISAKMP/Oakley provides for third party verification using the established industry-standard X.509 format certificate.

So it looks like this in actual implementation: when someone initiates an ISAKMP/Oakley exchange with you, or responds to one you have initiated, you first send him data he has to sign with his digital signature. You can then connect with the CA (which is a specialized X.500 server designed for this task), and request a copy of the certificate belonging to that person. You first check the certificate signature against your copy of the CA's signature to make sure the CA

really issued it. Provided it checks out, you then look up the signature attached to the certificate (the one belonging to the person you're trying to initiate an exchange with), and check the data they've just signed against it.

It's really a chain of trust—beginning with your trust in the CA's signature, which is then used to verify someone else's.

All of this may look complicated, but it's part of the beauty of IPSec that this can all be done in the background—quickly, quietly, and painlessly. It's far less trouble than checking the signature on the back of a credit card, in that your verification system checks for forgeries for you. You don't even have to think about it.

## ***Robust, scaleable key exchange***

Altogether, IPSec's ISAKMP/Oakley system offers network users a great deal that other schemes have had difficulty delivering. But most critically, IPSec's designers built the system's key exchange from the ground up to be genuinely scaleable—so it works with any size of system.

So to put all that together, with an IPSec secure VPN

- you can form SAs with a range of attributes, layering your network
- you can build multiple domains of communication within the same secure VPN
- you can update keys as frequently as you wish, and get the best of both worlds in doing so—perfect forward secrecy, and a silent, painless background mechanism the user doesn't have to think about
- you can do this in any IP network environment, of any size, for any size of enterprise
- and finally, you can do this with anyone using IPSec technology, creating new SAs with whomever you wish to communicate