IPSEC Working Group INTERNET-DRAFT
Radia Perlman

draft-ietf-ipsec-ikev2-tutorial-01.txt
February 2003

Understanding IKEv2: Tutorial, and rationale for decisions
<draft-ietf-ipsec-ikev2-tutorial-01.txt>

Status of this Memo

Abstract

The main job of a protocol specification is to document how the protocol works. It is sometimes difficult to learn how a protocol works from such a document, because there are so many details, and the necessary formalism for accuracy makes a specification long and difficult to read. What also is usually lost in the process of creating an RFC for a protocol is documentation of the tradeoffs that were considered when making controversial choices. Sometimes it is possible to find this information on the email archives, but that is a daunting task. This document is intended to work both as a tutorial to understanding IKEv2, and a summary of the controversial issues, with the reasoning on all sides of each issue. If any differences in details exist between this document and the IKEv2 specification, the IKEv2 specification is authoritative.

1. Introduction

IKE (Internet Key Exchange) is the protocol which performs mutual authentication and establishes security associations (SAs) for IPsec. The base protocol of the first version of IKE was documented in RFCs 2407, 2408, 2409. Also, IKEv1 implementations incorporated additional functionality including features for NAT traversal, legacy authentication, and remote address acquisition, which were not documented in the base documents. The goal of the IKEv2 specification is to specify all that functionality in a single document, as well as simplify and improve the protocol, and fix various problems in IKEv1 that had been found through deployment or analysis. It was also a goal of IKEv2 to understand IKEv1 and not to make gratuitous changes. The intention was to make it as easy as possible for IKEv1 implementations to be modified for IKEv2, and to benefit from the experience gained from deployment of IKEv1.

IKEv2 preserves most of the features of the original IKE, including identity hiding, perfect forward secrecy, two phases, and cryptographic negotiation, while greatly redesigning the protocol for efficiency, security, robustness, and flexibility. This document is intended to be a readable description of all the concepts, rather than being a complete specification of all the details. It also explains reasoning on all sides of controversial issues.

For simplicity of description, we refer to the two parties in an IKE exchange as "Alice" and "Bob", where Alice is the initiator of the exchange. These names allow us to use the pronouns "she" and "he".

2.0 Overview of IKEv2

IKEv2 has an initial handshake in which Alice and Bob negotiate cryptographic algorithms, mutually authenticate, and establish a session key, creating an IKE-SA. Additionally, a first IPsec SA is established during the initial IKE-SA creation.

All IKEv2 messages are request/response pairs. It is the responsibility of the side sending the request to retransmit if it does not receive a timely response.

The initial exchange usually consists of two request/response pairs. (Additional request/response pairs might be needed for DOS protection, if Alice attempts to use a Diffie-Hellman group Bob does not support, or if Bob will authenticate Alice through some legacy mechanism such as a token card, OTP, or name/password.

The first pair negotiates cryptographic algorithms and does a Diffie-Hellman exchange. The second pair is encrypted and integrity

protected with keys based on the Diffie-Hellman exchange. In this exchange Alice and Bob divulge their identities and prove it using an integrity check generated based on the secret associated with their identity (private key or shared secret key) and the contents of the first pair of messages in the exchange. Also, the first IPsec SA is created.

After the initial handshake, additional requests can be initiated by either Alice or Bob, and consist of either informational messages or requests to establish another child-SA. Informational messages include such things as null messages for detecting peer aliveness, and deletion of SAs.

The exchange to establish a child-SA consists of an optional Diffie-Hellman exchange (if perfect forward secrecy for that child-SA is desired), nonces (so that a unique key for that child-SA will be established), and negotiation of traffic selector values which indicate what addresses, ports, and protocol types are to be transmitted over that child-SA.

3.0 Two Phases

In IKEv2 terminology, the first phase consists of the (usually 4) messages that create the IKE-SA and the first associated IPsec SA (known as a "child-SA"). Once the IKE-SA is created, it can be used for sending authenticated notification messages, reliable dead-peer detection, and inexpensive creation of additional child-SAs.

It was argued in [PK01] and [JFK] that having two phases was unnecessary and added complexity, and additional SAs between the same pair of nodes could be accomplished by creating additional IKE-SAs. However, child-SA creation is less expensive, and experience with IKEv1 showed the two phases to be useful, since there were scenarios in IPsec deployments in which a sufficient number of child-SAs between the same pair of nodes was desirable that the extra spec complexity was worth it for the efficiency of child-SA creation.

Why do people find it useful to create multiple IPsec SAs between the same pair of hosts?

* avoiding multiplexing multiple conversations over the same SA. Several years ago Bellovin pointed out that if encryption is done without integrity protection, there is a splicing attack whereby a process involved in one flow can, through an active attack, cause traffic for a different flow to be decrypted and delivered to the process in the first flow. Of course, nobody should be doing encryption without integrity protection. It is likely there is no similar flaw if integrity is used. But in a case where a router is

delivering traffic on behalf of multiple customers, and the data is going to another router in order to access other machines of those customers, the customers feel safer knowing that their traffic is being delivered with a different SA (and different key) than traffic between nodes belonging to other customers.

* different security properties of different flows. According to policy, some traffic might be only integrity-protected. Other traffic might be encrypted with a short key. Other traffic might be encrypted with a long key. Other traffic might use a vanity crypto algorithm designed by one of your customers, and it will make them happy if you use their algorithm for their traffic. In [PK01] it was argued that all traffic might as well be protected according to the needs of the traffic that requires the strongest protection. The counter-argument is that there might be performance reasons or legal reasons (or vanity reasons) why this is undesirable.

* different SAs for different classes of service. There might be different classes of service, such as priority classes, that might cause traffic for one class to travel much more slowly to the SA destination than other types of traffic to that SA destination. To avoid replay attacks, the recipient keeps track of which sequence numbers have been received. Typically, it only keeps track of the highest n sequence numbers, up to the highest sequence number it has seen on this SA, and data with sequence numbers lower than that are discarded. If different classes of service have widely different delivery times, the recipient would have to keep track of a larger, and possibly unbounded, set of sequence numbers.

In JFK it was envisioned that IKE would be used solely for setting up an SA, and there would be no IKE messages other than the initial handshake. In IKEv1, the second phase was used either for setting up an additional SA or for sending informational messages. Once the WG consensus was to keep the two phase structure, both uses of the second phase; creation of new child-SAs, and the ability to send reliable and authenticated informational messages were deemed important. The ability to send informational messages increases IKEv2's robustness by detecting error conditions, allowing rekeying, and detecting a dead peer, as well as being a potentially valuable feature for future functionality.

Note that the ability to create additional child-SA's is optional in IKEv2, so it is legal for an implementation to have the behavior envisioned in the JFK spec.

4.0 Perfect Forward Secrecy/Computation Tradeoff

The IKEv2 handshake includes nonces in addition to Diffie-Hellman

values. If each side chose a unique private Diffie-Hellman number for each exchange, there would be no need for nonces. It is reasonable for an implementation to choose less than perfect forward secrecy by reusing the Diffie-Hellman number (avoiding expensive exponentiations), since the nonces, which must be unique for each exchange, will ensure unique keys for each IKE-SA. Likewise, child-SAs established through an IKE-SA can choose perfect forward secrecy and generate and send Diffie-Hellman values, or simply use nonces to establish unique keys.

Note that even if Bob (or Alice) reuses his Diffie-Hellman value (call it "B"), there will still be perfect forward secrecy so long as Bob forgets B as soon as any SA based on B is closed. If Bob remembers B for, say, an hour longer than that, then perfect forward secrecy is only slightly affected, and really not affected in any practical sense.

5.0 Colocated Services

In some cases Bob might host many different services (e.g., distinct web sites with different identities). All these identities would have the same IP address, but would have different keys and certificates. Having Alice initiate a connection to Bob's IP address does not inform Bob who she wants to communicate with. Therefore, IKEv2 allows Alice to specify an identity for Bob. This feature was given the affectionate name "You Tarzan. Me Jane." by Hugh Daniel. The name is quite appropriate because in the same message in which Alice reveals her identity she requests a specific identity for Bob.

6.0 DOS protection

Photuris specified a mechanism known as a "stateless cookie", in order to avoid a certain type of DoS attack. The attack involves sending nuisance SA-creation-request packets to a server (Bob) from an unauthorized node with the purpose of exhausting the server's computation or memory resources. Such attacks typically are sent from forged source addresses, both to avoid prosecution and to make it difficult for these packets to be easily filtered. Photuris's cookie design was a way of assuring Bob that the SA-requester could receive at the IP address it claimed to be coming from before Bob devoted significant resources to authenticating the request and creating the SA.

A method of implementing stateless cookies is for Bob to have a secret S, that he shares with nobody and changes periodically. When he gets a request from IP address x with no cookie, he sends cookie=h(S,x) to x, but otherwise does nothing with the request. When he gets a request from IP address x with cookie=c, Bob verifies if

c=h(S,x) and if so, continues processing. (If Bob has recently changed S's, he might want to see if the cookie verifies with the old S if it fails with the new S).

OAKLEY had the stateless cookie be optional. If there was no attack, the protocol would avoid the extra round trip for the cookie exchange. If Bob was getting low on resources, perhaps because of an attack, Bob could refuse requests that didn't contain a cookie. IKEv2 uses the OAKLEY idea of having the cookie exchange be optional.

This aspect of IKEv2 was the subject of some debate in the WG. There were two alternatives for providing this feature. The chosen alternative was the OAKLEY-style optional extra round trip. It was possible, (as specified in JFK, suggested in [PK01], and specified in the next version of IKEv2 after the JFK and IKEv2 author teams decided to work together on a single document), to provide this feature without adding an additional round trip. The arguments for avoiding the extra round trip were:

* it saves a round trip

* it avoids forcing Bob to make a decision about whether he is under attack

The WG decided in favor of the additional round trip for this case because:

* it made the protocol much simpler, since after the initial pre-exchange, Bob is not stateless. As a result of the protocol being simpler, it was likely that future changes would not break the handshake, and that future functionality could be incorporated without a redesign.

* it makes message 3 shorter, since the mechanism by which Bob can be stateless is to have Alice repeat everything Bob would have needed to remember in message 3.

* This design makes it easy to defend against a "fragmentation attack", a DOS attack on an IKE exchange that was pointed out by Charlie Kaufman that could enable an attacker to prevent IKE exchanges from completing. Since message 3 in an IKE exchange tends to be long (it includes certificates), and IKE runs over UDP, it is likely that it will need to be fragmented. With the variant in which Bob is stateless until verifying message 3, (and it is message 3 in which Alice sends the cookie), an attacker could send fragments, exhausting Bob's reassembly resources, so that Bob's IKE would never get to see and verify the cookie. With the extra two messages for cookie exchange, all messages are sufficiently short so that

reassembly would not be required, and a fragmentation attack cannot prevent Bob from verifying Alice's cookie.

Once Bob has verified Alice's cookie, it is a fairly easy implementation trick to ensure the rest of the IKE exchange completes, even in the face of a fragmentation attack, by providing a side-channel from IKE to the reassembly code, whereby Bob can inform the reassembly code of preferred IP addresses (those that have returned a valid cookie).

7.0 Cryptographic Negotiation

In IKEv1, cryptographic negotiation was "a la carte", meaning that each algorithm (encryption, integrity protection/prf (prf=pseudorandom function), Diffie-Hellman group), was independently negotiated. Aside from being complex to understand, it also created an exponential expansion, since if there were k of one type of algorithm that could interwork with j of another, there had to be k*j seperate proposals. In the original IKEv2 design, the a la carte concept was kept, but the SA payload was simplified somewhat. Also, the exponential explosion of proposals was avoided by allowing sets of algorithms that could interwork together to be presented as a single proposal, and Bob could narrow the choices down to any one from the set. JFK, in contrast, had no negotiation of cryptographic algorithms, which was even simpler, but made it difficult to migrate to different algorithms in the future.

The IKEv2 and JFK authors together agreed that a compromise would be suites, as was done in SSL. With a suite, all parameters are encoded into a single suite number, and negotiation consists of offering one or more suites and having the other side choose. It was assumed this would be a noncontroversial decision, but unfortunately it turned out to be controversial. The arguments in favor of suites are:

* it is simpler and more compact to encode

The arguments in favor of a la carte are:

* it is more flexible

* there is the fear that there will be an exponential number of suites defined

* it is a gratuitous change from IKEv1 that made a lot of unnecessary work for implementations. Suites might have been OK if starting from scratch, but a la carte was easier for migrating from an IKEv1 code base.

Although there was sympathy with the a la carte supporters, a decision had to be made, and based on a straw poll at a WG meeting, the decision was to use suites.

8.0 Acquiring an IP address

When an endnode dials into a firewall, it is often the case that the endnode needs to be given an IP address. Even if the endnode has an IP address for where it is currently residing in the Internet, it may need an address specific for the network inside the firewall, since with its IPsec tunnel, the endnode will now be logically inside the firewall. There were two proposed methods of doing this:

* MODECFG, which involves a field in the IKE exchange in which Bob tells Alice an IP address, and

* DHCP-relay, which involves running DHCP over IKE or over an ESP connection set up specifically for this purpose.

The appeal of MODECFG is that it is very simple, and minimizes the number of messages and crypto operations in getting an IPsec session set up. The appeal of DHCP-relay is that it provides all of the flexibility and power of DHCP (including extensions that might be defined in the future) and does so in a way that appears to make it independent of the IKE specification. One thing that can be done with DHCP-relay is end-to-end authentication between the client and the DHCP server, for instance.

The use of MODECFG does not preclude the use of tunnelled DHCP for uses other than acquiring leases on IP addresses, and if there is functionality that can only be done using DHCP-relay, this may be done. The worry was that both MODECFG and DHCP-relay might be needed, and that even though DHCP-relay was more complex than MODECFG, if doing MODECFG meant implementations had to support both, doing DHCP-relay instead of MODECFG would mean less implementation effort. However, the decision was that for now, since MODECFG was simpler, higher performance, fully specified, and gave all currently-needed functionality, IKEv2 would assign addresses using MODECFG.

9.0 NAT Traversal

People love to hate NAT (Network Address Translation) gateways, but they are a fact of life in the Internet. NAT-Traversal [KSH01] designed a mechanism that was deployed in IKEv1, and IKEv2 copied the design. The NAT-Traversal design accommodated existing NATs as much as possible (without sacrificing security or significantly impacting performance).

NATs were originally invented primarily because of the shortage of IPv4 addresses, though there are other rationales. IP nodes that are "behind" a NAT have IP addresses that are not globally unique, but rather are assigned from some space that is unique within the network behind the NAT but which are likely to be reused by nodes behind other NATs. Some people consider the fact that the nodes in their network cannot be directly addressed from outside a security feature. And it even allows a network using some addressing scheme different from IPv4 to connect to the Internet.

9.1 The games NATs play

Generally, nodes behind NATs can communicate with other nodes behind the same NAT and with nodes with globally unique addresses, but not with nodes behind other NATs. When a node behind a NAT makes a connection to a node on the real Internet, the NAT gateway assigns the inner node a global IP address (which the Internet will route to the NAT box), keeps the mapping for the duration of the conversation (where "duration" has to be heuristically determined by the NAT box), and modifies the IP addresses in the header appropriately. For outgoing packets, the NAT box modifies the source address. For incoming packets, the NAT box modifies the destination address to be the address of the node on the internal network.

If the NAT box does not have a sufficiently large pool of global IP addresses to hand out a unique one to each node inside its net that is communicating outside, then NAT boxes translate based on UDP or TCP ports. This is known as a NAPT box.

NAPT boxes are foiled by ESP, because ESP encrypts the layer 4 header. Some NAPT boxes attempted to make ESP work by doing the following (assume the NAPT box has only one globally unique address, "C"):

* when it sees an ESP packet from inner node IP address A to outer node IP address B, the only extra information is the SPI=x. So the NAPT box rewrites the source address A to C, and keeps track of the fact that an ESP packet came from A recently, and there is no current mapping for an incoming SPI corresponding to x.

* when the NAPT sees an ESP packet from B to C, with an SPI=y that the NAPT has no mapping for, it assumes B is really trying to respond to A, and makes a mapping that (C,y)=A.

An additional problem with NATs and IKEv1 is that IKEv1 specified that IKE messages must be sent to port 500, and sent from port 500. If IKEv1 had specified that you initiate by sending to 500, but respond to whatever port you received an IKE packet from, things

would have been simpler. But NAPTs, understanding IKE will want both source and destination UDP ports to remain as 500 make a special case for packets on UDP port 500. If the ports are 500, the NAPT does not modify the UDP ports. Instead, the NAPT makes mappings based on the what used to be called the "cookie fields" in the IKEv1 payload (and in IKEv2 are called the SPIs).

The way it works is that when the NAPT box sees a packet on port 500, it looks at the SPI pair (ci,cr). If it does not yet have a mapping for that pair, the packet will have to be coming from inside the net, from IP address A, to IP address B. The NAPT box makes a mapping that IKE connection (ci,cr) is for node A. The box does not modify the UDP ports or the IKE SPIs. It merely records the SPI pair for its IP address mapping. Outgoing packets from A to that IKE connection will always get translated as having source address C. But incoming packets to C from that SPI pair will get translated to destination address A.

In the beginning of the IKE connection, Bob has not yet chosen a SPI, so the SPI pair will be (ci,0). If two nodes from inside the net initiate IKE connections to Bob simultaneously, and both choose SPI=ci, the NAPT would not be able to differentiate. But since SPIs are 8-bytes long, and recommended to be chosen at random, this is unlikely to happen.

9.2 NAT detection

NAT-Traversal was designed for enabling IKEv1 to work through NATs without requiring modifications of the NATs, and IKEv2 has pretty much copied the design. The first step is for Alice and Bob to notice that one of them is behind a NAT. In IKEv2 this is done by including two notify payloads in messages 1 and 2, called NAT-DETECTION-SOURCE-IP and NAT-DETECTION-DESTINATION-IP. These notify payloads consist of a one-way hash of the IP address and port. The reason it is a hash rather than the actual address is because some people have argued that the actual address of a node behind a NAT might be secret. Given there are only 32-bits in an IP address and the port is known to be equal to 500, it is possible to do a brute force search and discover the actual IP address, but having the payload convey the hash rather than the actual address is at least a nuisance to someone that would want to find the address.

Mysteriously, the NAT-DETECTION payloads are ignored by Bob. However, if Alice notices a discrepancy between the IP addresses in the header of the received message 2, and the hashes in the payloads, she reverts to NAT behavior.

9.3 So, you're behind a NAT. What do you do?

In NAT-mode, packets on a child-SA (e.g., ESP packets) are sent with UDP encapsulation. This means that instead of indicating the ESP header with an IP header protocol type (and having the ESP header immediately following the IP header), the IP header will indicate "UDP", and the presence of the ESP header will be indicated by using port 4500 in the UDP header.

NAPTs do not do any special-case processing with port 4500 (as they do with port 500). So, if there is a packet from internal node A to B, with UDP header, and ports 4500, and no mapping on the NAPT box yet for A, the NAPT box will make a mapping from A to (IP address C, port x), and overwrite the IP source address to C and the UDP source port to x. When Bob receives a packet to port 4500 (indicating IPsec UDP encapsulation), Bob responds to the port from which it was received, so will respond to IP address C, port x. When the NAPT box receives this packet from Bob to (C,x), it will translate the IP address from C to A, and the port from x to 4500.

In addition to sending the child-SA packets on 4500, IKE (once the NAT is detected) sends the remaining messages of the IKE handshake over port 4500 instead of 500 (and does not insist on receiving them from source port 4500). Using port 4500 for IKE wasn't strictly necessary but had some advantages:

* It enables the NAPT box not to do special case processing for IKE, and instead modify the UDP ports (as it would with anything else) instead of relying on the IKE SPIs, which was a somewhat fragile and very complex mechanism,

* It winds up creating fewer mapping entries in the NAPT box, since the same port mapping for UDP-encapsulated child-SA packets will work for the IKE exchange. (i.e., there is no need to keep mappings for IKE cookie pairs).

* Since NAT boxes are only using heuristics for how long to keep a mapping, if there were a different mapping for IKE than for the child-SA, it could be that the NAT would forget the UDP port mapping for the child-SA, but remember the IKE-SA cookie mapping. This would be bad because dead peer detection is done by sending IKE informational messages, which would indicate the SA was alive, but child-SAs would go into a black hole because the NAT box would no longer know how to map packets from B to A.

9.3 Encoding both IKE and ESP with port 4500

The mechanism for this encoding was copied from [HSSVC]. If a NAT is

detected, ESP packets will be sent with UDP encapsulation, using port 4500. Also, IKE packets will be sent to port 4500. How can the receiving end distinguish between an ESP packet and an IKE packet?

In either case the packet will start with an IP header, with protocol type=UDP (17). Then there is a UDP header, with destination port=4500. After the UDP header is either an ESP packet or an IKE packet. The first 4 bytes of an ESP packet is the SPI, which is not allowed to be zero. So, to distinguish an IKE packet from an ESP packet, if it is an IKE packet, the first 4 bytes after the UDP header are 0, and then the IKE packet.

With this encoding, the overhead for UDP encapsulation of ESP packets is minimized, and the extra 4 bytes of overhead is only on IKE packets, and there are not many of those (compared to data packets).

10.0 Identity Hiding

Some people argue that identity hiding is an exotic feature that cryptographers put into IKEv1 just because they could. In many cases, such as those where nodes are at a fixed IP address, the identity is not hidden.

And, there are different flavors of identity hiding. IKEv2 does identity hiding of both parties from passive attackers.

In theory (and in IKEv1) one could hide the identities from active attackers. With public encryption keys, if at least one side already knows the identity and public key of the other, then it is possible to protect both sides from any active attacker (assuming the encryption key is not escrowed or otherwise compromised). With pre-shared secret keys, assuming both parties already know who they expect to be speaking with (within a small set, perhaps), it is also possible to protect both identities from active attack. (But in fact, in IKEv1, this was too strong an assumption, and identities with the in-theory identity-hiding secret key protocol required, in practice, that the identities be the IP addresses.) Additionally, having n different protocols for slightly different security properties in IKEv1 was deemed to be too complex for any benefit it gained, so IKEv2 only supports public signature keys and pre-shared keys.

However, with public signature keys, one side or the other has to reveal its identity first (before the other side has proven its identity). Whichever side reveals its identity first, if it is talking to an active attacker, it will have revealed its identity to that attacker. In [PK01] it was argued that it was more important to protect the identity of the initiator, since in the client-server

model, the server would be at a fixed IP address and would not have a hideable identity. However, Charlie Kaufman later argued that a much easier attack is a polling attack, in which the attacker merely opens an IPsec connection to a node. If the responder reveals its identity first, then this simple attack, which is easier to mount than a passive attack, will reveal the identity at that address. If the model were changed to a strict client-server model in which clients never respond to connections, and server identities are not important to protect, then it is reasonable to have the responder reveal its identity first. The WG's decision was that they did not want to limit IPsec's use to a strict client-server mode.

To avoid a polling attack, (in which an active attacker simply initiates a connection to an IP address to find the identity associated with that IP address) IKEv2 has the initiator reveal its identity first. The active attack that IKEv2 has chosen not to deal with involves having someone impersonate Bob's IP address and discover the identities of parties that attempt to communicate with that IP address. This attack is difficult to mount and it is not obvious what benefit it would gain the active attacker. Alice has only initiated a connection to an IP address. If she is not speaking to the real Bob, she will discover this and break the connection. So the active attacker cannot prove she intended to speak to Bob; merely that she initiated an IPsec connection to a particular IP address.

11.0 Legacy Authentication

Alice might be a human, without a public key pair or a shared cryptographic key. She might be using a token card (such as SecureID), or a password.

There were several proposed extensions to IKEv1 for providing legacy authentication [XAUTH], [CRACK], [EAP]. Given they were all technically acceptable, one had to be chosen. The EAP protocol [EAP], designed within the pppext WG, was adopted for IKEv2.

To use EAP with IKEv2, Alice, in message 3, reveals her identity, but does not put in a certificate or an authentication payload. Bob, in message 4, reveals and proves his identity, and specifies what type of legacy authentication he wants, along with a text string to be displayed at the client end (such as "this is your challenge for your challenge/response token"). Alice can respond as Bob requests, or can NAK and suggest a different type of EAP authentication. The IKEv2 spec really just references the EAP specification, so the design and the types are defined within EAP. In IKEv2, mostly for illustrative purposes, 3 of the EAP types are mentioned; MD5-Challenge, OTP, and generic token card. In MD5-Challenge, the client must compute the

response, and not just mirror the text string. For OTP, depending on whether the implementation is based on human-with-paper or client-computed hash, the client either just sends the string the user types or treats it as a password and hashes it n times. In the third type, the client displays the text string to the human, which responds by typing something (perhaps the display from the token card), and the client sends that string back to the server. If the server is satisfied, it responds with what would have been the 4th message in the IKE handshake, choosing the selectors and cryptographic algorithms for the child-SA. Additional exchanges are allowed, for instance, if the user mistypes the value and the server gives the user an extra chance.

12.0 References

[CRACK] Harkins, D., Korver, B., Piper, D., "IKE Challenge/Response for Authenticated Cryptographic Keys", draft-ietf-ipsec-ike-crack-00.txt, October, 1999.

[EAP] Blunk, L. and Volibrecht, J., "PPP Extensible Authentication Protocol (EAP), RFC 2284, March 1998.

[HC98] Harkins, D., Carrel, D., "The Internet Key Exchange (IKE)", RFC 2409, November 1998.

[HKP99] Harkins, D., Korver, B., Piper, D., "IKE Challenge/Response for Authenticated Cryptographic Keys", draft-ietf-ipsec-ike-crack-00.txt

[HSSVC] Huttunen, A., Swander, B., Stenbert, M., Volpe, V., and DiBurro, L., "UDP Encapsulation of IPsec Packets", draft-ietf-ipsec-udp-encaps-06.txt, January 2003.

[JFK] Aiello, W., Bellovin, S., Blaze, M., Canetti, R., Ioannidis, J., Keromytis, A., Reingold, O., draft-ietf-ipsec-jfk-03, April 2002.

[KSH01] Kivinen, T., Stenberg, M., Huttunen, A., Dixon, W., Swander, B., Volpe, V., and DiBurro, L., "Negotiation of NAT-Traversal

in
the IKE", draft-ietf-ipsec-nat-t-ike-01.txt, October 2001.

[MSST98] Maughhan, D., Schertler, M., Schneider, M., and Turner, J.
"Internet Security Association and Key Management Protocol
(ISAKMP)", RFC 2408, November 1998.

[Orm96] Orman, H., "The Oakley Key Determination Protocol", RFC 2412, November 1998.

[PK01] Perlman, R., and Kaufman, C., "Analysis of the IPsec key exchange Standard", WET-ICE Security Conference, MIT, 2001,
http://sec.femto.org/wetice-2001/papers/radia-paper.pdf.

[Pip98] Piper, D., "The Internet IP Security Domain Of Interpretation for ISAKMP", RFC 2407, November 1998.

[XAUTH] Beaulieu, S., and Pereira, R., "Extended Authentication within IKE (XAUTH)", draft-beaulieu-ike-xauth-02.txt, October 2001.

Authors' Addresses

Radia Perlman
radia.perlman@sun.com
Sun Microsystems