# INTERNET KEY EXCHANGE (IKE)

## R OUTLINE

tion
ory
Protocol
Key Management for Internet
s (SKIP)
ses: IKE Modes

sion 2 - The Future of IKE
coding

## 2.1. INTRODUCTION

The Internet Key Exchange (IKE) protocol is the main part of the IPSEC implementation, and is used to negotiate secret key material between two parties, called the initiator and responder. The key material is the result of the protocol, and is used to create the Security Associations (SAs) that define how the traffic between the two hosts are to be protected. The IKE also provides mutual authentication by authenticating both of the parties to each other; Both of the parties need to provide a digital signature in the key exchange protocol that the other party will verify. A successful verification means that the other party is authenticated. In order to be able to verify the signature also the public key (certificate) needs to be trusted, and verified. If certificates are not available, the trust can be gained also with a pre-shared-key.

## 2.2. IKE HISTORY

The Internet Key Exchange (IKE) protocol, described in RFC 2409, is a key management protocol standard which is used in conjunction with the IPsec standard. IPsec can be configured without IKE, but IKE enhances IPsec by providing additional features, flexibility, and ease of configuration for the IPSec standard.

IKE is a hybrid protocol based on the Internet Security Association and Key Management Protocol (ISAKMP), described in RFC 2408. The IKE protocol implements parts of the two other key management protocols—Oakley, described in RFC 2412, and SKEME. The protection policy within SAs is negotiated and established with the help of the ISAKMP protocol, and keying material (session keys

for encryption and packet authentication) is agreed on and exchanged with the use of Oakley and SKEME protocols.

**ISAKMP**—The Internet Security Association and Key Management Protocol is a protocol framework that defines payload formats, the mechanics of implementing a key exchange protocol, and the negotiation of a security association. ISAKMP is implemented according to the latest version of the "Internet Security Association and Key Management Protocol (ISAKMP)" standard.

**Oakley**—A key exchange protocol that defines how to derive authenticated keying material.

**Skeme**—A key exchange protocol that defines how to derive authenticated keying material, with rapid key refreshment.

The Internet Security Association and Key Managment Protocol (ISAKMP) establishes a secure management session between IPsec peers, which is used to negotiate IPsec SAs. ISAKMP provides the means to do the following :

- Authenticate the remote peer
- Cryptographically protect the management session
- Exchange information for key exchange
- Negotiate all traffic protection parameters using configured security policies.

Therefore, the goal of ISAKMP is the establishment of an independent security channel between authenticiated peers in order to enable a secure key exchange and the negotiation of IPsec SAs between them.

Oakley is originally a free-form protocol that allows each party to proceed with the exchange at its own speed. IKE borrowed this idea from Oakley, and defines the mechanisms for key exchange in different modes over the IKE (ISAKMP) session. Each protocol produces a similar result—an authenticated key exchange, yielding trusted keying material used for IPsec SAs. Oakley, within IKE, determines AH and ESP keying material (authentication and encryption session keys) for each IPsec SA automatically, and by default uses an authenticated Diffie-Hellman algorithm to accomplish this.

Diffie-Hellman algorithm was discovered in 1976 by Whitfield Diffie and Martin Hellman. It gets its security from the difficulty of calculating the discrete logarithms of very large numbers. The Diffie-Hellman algorithm is used for secure key exchange over insecure channels and is used a lot in modern key management to provide keying material for other symmetric algorithms, such as DES or keyed-MD5 (HMAC).

## 2.3. PHOTURIS PROTOCOL

Photuris establishes short-lived session-keys between two parties, without passing the session-keys across the internet. These session keys directly replace the long lived secret-keys (such as passwords and passphrases) that have been historically configured for security purposes. The basic photuris protocol utilizes these existing previously configured secret-keys for identification of the parties. This is intended to speed deployment and reduce administrative configuration changes.

### 2.3.1. Protocol Overview

The photuris protocol consists of several simple phases:

1. A "Cookie" Exchange guards against simple flooding attacks sent with bogus IP sources or UDP Ports. Each party passes a "Cookie" to the other.

```
                                        make privacy key(s)
                                        mask/encrypt message
                    [make SPI session-keys in each direction]
SPI User                                SPI Owner
= = = = = =                             = = = = = =
SPI_Needed                  →
    list SPI attribute(s)
    make validity key
    authenticate
    make privacy key (s)
    make/encrypt message
                                ←   SPI_Update
                                        make SPI
                                        pick SPI attribute(s)
                                        make SPI session-key (s)
                                        make validity key
                                        authenticate
                                        make privacy key (s)
                                        mask/encrypt message
```

Either party may initiate an exchange at any time. For example, the initiator need not be a "caller" in a telephony link.

The initiator is responsible for recovering from all message losses by retransmission.

### 2.3.2. Security Parameters

A photuris exchange between two parties results in a pair of SPI values (one in each direction). Each SPI is used in creating separate session-key (s) in each direction.

The SPI is assigned by the entity controlling the IP destination: the SPI Owner (receiver). The parties use the combination of IP, Destination, IP (Next Header) protocol, and SPI to distinguish the correct Security Association.

When both parties initiate Photuris exchanges concurrently, or one party initiates more than one Photuris exchange, the Initiator Cookies (and UDP Ports) keep the exchanges separate. This results in more than one initial SPI for each Destination.

## 2.4. SIMPLE KEY MANAGEMENT FOR INTERNET PROTOCOLS (SKIP)

Most key distribution and management schemes and protocols are session-oriented, meaning that they establish and maintain an SA that lasts at least a certain amount of time. In contrast, many network and Internet layer protocols, such as IP and CLNP, are connectionless in nature and provide a connectionless datagram delivery service. To use a session-oriented key distribution scheme with IP, one has to crete and implement a pseudosession layer underneath IP for the establishment and updating of IP traffic authentication and encryption keys. In fact, this layer is represented by the SA. While this is possible, it may be far less cumbersome to use a scheme that does not entail the overhead of such a pseudosession layer.

Against this background, Sun Microsystems developed and proposed a key distribution and management scheme and protocol that is not session-oriented, meaning that it does not make use of SAs, in the early 1990s. As mentioned, the resulting key distribution and management scheme and protocol was named SKIP and is being marketed by Sun Microsystems. You may

refer to the SKIP home page to learn about the availability of SKIP products and their interoperability with other VPN products.

Contrary to most other key distribution and management schemes and protocols that have been submitted to the IETF IPSEC WG, SKIP is not session-oriented. Instead of having the communicating peers establish and maintain an SA, SKIP uses packet-specific encryption keys that are communicated inband with the IP packets (in cryptographically protected form). More specifically, SKIP uses predistributed and authenticated Diffie-Hellman public keys to have two entities share an implicit master key from which a key encryption key (KEK) may be derived. The KEK, in turn, is used to encrypt a randomly chosen packet key and the packet key is used to encrypt the IP packet. Finally, the encrypted packet key is prepended to the encrypted IP packet.

Let us assume that all SKIP entities share a large prime p and a generator g of $Z_p^*$. We further assume that principal A has a private Diffie Hellman key $X_A$ and a corresponding public key $Y_A = g^{xA} \pmod p$, and that principal B has another private key $X_B$ an a corresponding public key $Y_B = g^{xB} \pmod p$. Both public keys are distributed in the form of public key certificates and retrieved using a protocol, such as the Certificate Discovery Protocol (CDP), specified for SKIP. According to the Diffie-Hellman key exchange, A and B then share the secret key $K_{AB} = g^{xAxB} \pmod p$. This key is implicit and does not need to be communicated explicitly to either entity (i.e., each entity can compute the shared secret based on knowledge of the other entity's identity and public key certificate). This mutually authenticated long-term secret key $K_{AB}$ can be used to derive a master key or KEK $K_{AB}$ to provide IP-packet based authentication and encryption. $K_{AB}$ can be derived, for example, by taking the low-order key size bits of $K_{AB}$. However, there are at least two reasons for updating the KEK periodically:

1. It minimizes the exposure of any given KEK, making cryptanalysis more difficult.
2. Updating the KEK prevents the reuse of compromised packet keys.

The KEK is updated by sending in the packet a counter n that only increments and is never decremented $K_{AB}$ then becomes a function of this counter n as follows: $K_{ABn} = h(K_{AB}, n)$. Again h refers to a one-way hash function, such as MD5, SHA-1, or RIPEM. A simple and stateless way of implementing n is to let n be equal to the number of time units since the more recent of A's or B's certificates. In the absence of a clock, n can be constructed simply by using a counter.

To encrypt an individual IP packet, A randomly generates a packet key $K_p$ and digitally envelopes the packet with this key and the KEK that he or she implicitly shares with the recipient B. More accurately, A encrypts both the packet with $K_p$ and $K_p$ with $K_{AB}$, and encapsulates both ciphertexts in a new IP packet. Note that because $K_{AB}$ can be cached for efficiency, it allows packet keys to be modified very rapidly without incurring the computational overhead of public key operations. The packet keys can be changed as frequently as desired in line with a key management policy. If necessary, packet keys can be changed on a per-packet basis. When B receives the encrypted packet, he or she looks up A's certificate (using, for example, the SKIP CDP). Using the corresponding long-term public key and his or her own long-term private key, B can compute $K_{AB}$ and hence $K_{AB}$. Using $K_{AB}$, B can finally decrypt both $K_p$ and the entire IP packet.

Figure 2.1 illustrates the use of SKIP to encrypt unicast and multicast IP packets. The use of SKIP to encrypt unicast packets has been described. The traditional approach for key distribution in a multicast environment is to set up and run a KDC that distributes the traffic keys to the legitimate and authorized members of a group. This one shared key is then used by the group members to encrypt and decrypt data. There are at least two problems related to this approach.

1. Key change cannot be done efficiently and does not scale well to large groups. Key change policies need to be a function of the amount of data encrypted with a given key, and not merely a function of time alone. This means that for highspeed data transmission links, the keys must be updated far more frequently than for slower links, for the same key change policy. For a large number of members in a multicast group, however, it may become increasingly difficult or prohibitive for a multicast KDC to be rapidly supplying updated keys to all group members.
2. Use of the same key by all members of a multicast group precludes the use of certain stream ciphers. This is because using the same key will result in the same key stream being used to encrypt different plaintexts. Because key stream reuse is catastrophic to the security of some stream ciphers, it should be avoided generally. Nevertheless, it is important to allow stream ciphers to be used in conjunction with IP multicast. A common use of IP multicast is videoconferencing. Because video is a demanding application in terms of speed and throughput, it is important to allow the use of ciphers that can be efficiently implemented in software. Some of the most widely used and efficient ciphers are stream ciphers such as RC4. Because of key stream reuse issues, it is not possible to use a stream cipher like RC4 for IP multicast if all members of the multicast group use the same traffic key.

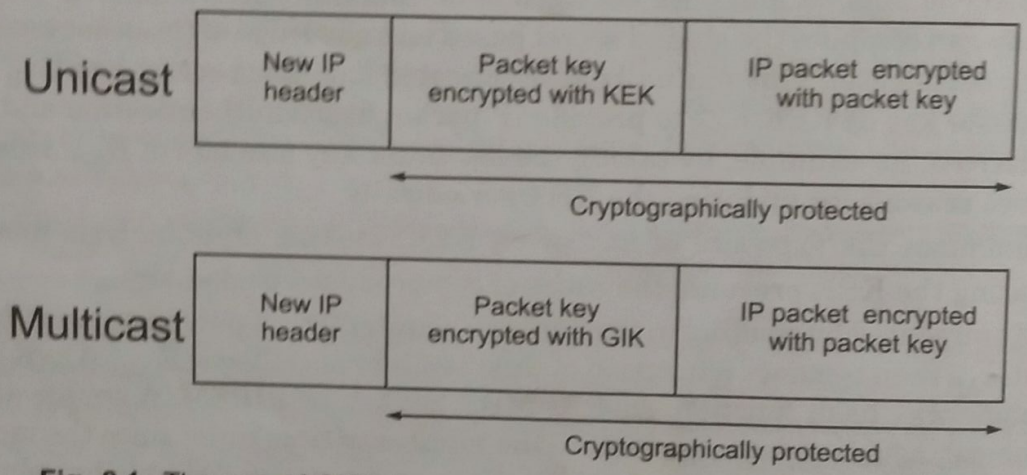



**Fig. 2.1.** The use of SKIP to encrypt unicast and multicast IP packets.

## 2.5. IKE PHASES: IKE MODES

IKE has several modes which define how the actual key exchange precedure is to be done. Two of the most common modes arr Main Mode and Aggressive Mode. There are also other modes like Base Mode and New Group Mode, but they are seldom used, and vendors usually does not support them at all.

The difference of Main Mode (MM) and Aggressive Mode (AM) is in length of the procedure. The AM can be completed with only 3 packets, where MM takes 6 packets to complete. Which ever mode is chosen these modes are called Phase-1 modes in IKE.

There is also a Phase-2 mode, called the Quick Mode (QM). The Phase-2 supports only one mode and it is called the Quick Mode. The Quick Mode takes 3 packets to complete.

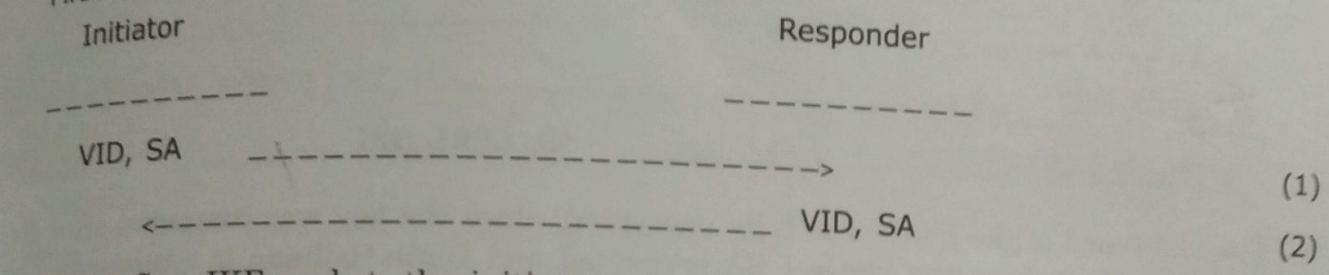
### 2.5.1. Phase-1 Mode

The IKE negotiation always starts by executing the Phase-1 of the protocol. This section describes the procedure when performing Main Mode. The examples assume that digital signatures (certificates) are used in authentication.

The purpose of the Phase-1 is to authenticate the peers to each other, and provide protection for the upcoming Phase-2 negotiation. The Phase-1 is used to exchange proposals, vendor specific information, certificates, and it is also used to perform the mutual authentication, which is the main purpose of the Phase-1 negotiation. The result of the Phase-1 can be called in many ways: Phase-1 SA, ISAKMP SA, IKE SA, etc. They all mean the same thing. The Phase-1 SA is also used to protect the actual Phase-2 negotiation, described subsequently. The Phase-1 SA can also be used to protect other informational notifications that may be sent in IKE (such as SA delete notifications).

The Phase-1 Main Mode is performed as follows:

First & second packets:



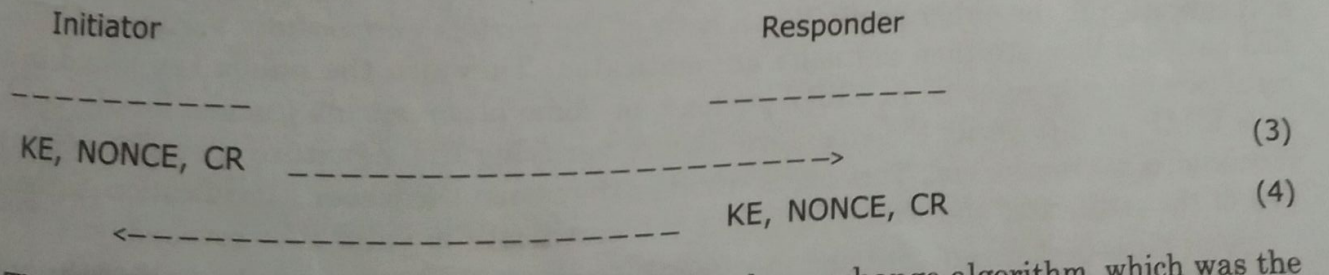

In the first IKE packet, the initiator may send zero (0) or more Vendor ID Payloads (VID), that each include a Vendor ID. The Vendor IDs are usually used to tell the responder what kind of extensions the initiator supports. IKE does not provide any other mechanism to tell what extensions initiator supports. The actual data of the Vendor ID is usually a message digest of some ASCII text.

The SA payload is mandatory and it is used to list the security properties the initiator supports. It includes the ciphers, hash algorithms, key lengths, lifetimes, and other information. It is possible to send only one (1) SA payload in Phase-1.

The responder may also send zero (0) or more VID payloads in its reply to the initiator. The responder must also include SA payload in its reply. The SA payload responder sends includes the security properties it selected from the initiator's security property list (the SA payload).

These packets are not encrypted, since there are no key to encrypt them with.

Third & fourth packets:



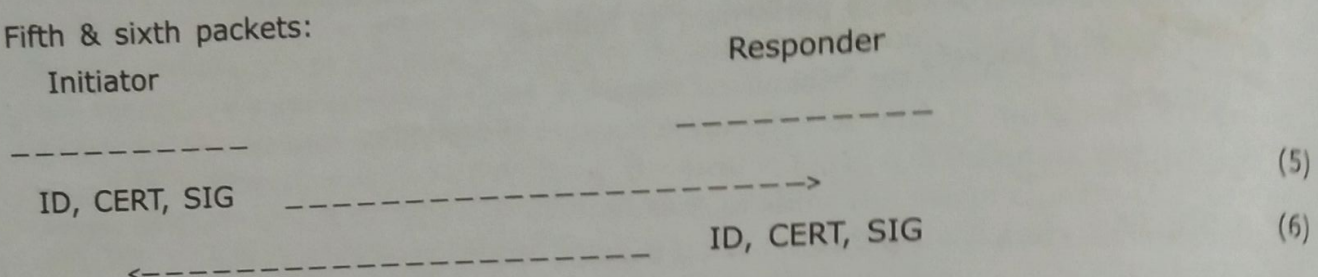

The IKE protocol is based on the Diffie-Hellman key exchange algorithm, which was the first ever invented algorithm that use public key cryptography (in 1974). The third packet is used to exchange the Diffie-Hellman public keys inside a Key Exchange (KE) payload. The Diffie-Hellman public keys are created automatically every time the Phase-1 negotiation is performed, and they are destroyed automatically after the Phase-1 SA is destroyed.

The responder also sends its Diffie-Hellman public key in the fourth packet to the initiator.

There is also a NONCE payload that is sent in third and fourth packet which is used (with other information) in the IKE to compute the secret data for the Phase-1 SA. The NONCE payload includes random data from random number generator.

The CR payload is a Certificate Request payload and is used to request for certificates by a specific CA. The CR payload includes the name of the CA for which it would like to receive the remote's end entity certificate (peer certificate). If empty CR payload is received, it means that it requests any certificate from any CA. The CR payload is usually sent in the third and fourth packets, but it can be sent also in first and second packet. In our example, they are sent in third and fourth packet.

These packets are not encrypted, since there are no key to encrypt them with.




The last round of the Phase-1 is fully encrypted, since the key was computed after the third and fourth packets (the Phase-1 SA is created and Diffie-Hellman is computed). The last round is used to send Identification (ID) payload, zero (0) or more Certificate payloads (CERT), which each include one certificate (or CRL), and the Signature payload (SIG) which is the digital signature that the other party must verify.

The ID payload is used to tell the other party who you are, and it also can be used to make policy decisions and to find the certificate of the remote end. The ID may be IP address, FQDN, email address, or something similar.

The CERT payload is optional payload, but usually if it is not sent the result of the IKE is "Authentication Failed" error. The CERT payload includes the sender's end entity certificate, but it is also possible to send CRL inside a CERT payload. The CERT payload is optional because it is possible that the remote end has cached the public key locally, and does not need to receive the CERT payload in the negotiation. Usually implementations do not cache it locally and in this case failing to send CERT payload also causes failure of the IKE negotiation.

The SIG payload includes the digital signature computed with the private key of the corresponding public key (usually sent inside the CERT payload), and provides the authentication to the either party. When both of the parties successfully verify each other's SIG payloads they are then mutually authenticated. They use the public key found in the certificate (usually received in CERT payload, or some other means (cached locally, fetched from LDAP, etc)) to verify the signature. Before verifying the signature they also verify the certificate of the remote end. They check whether they trust the issuer (Certification Authority, CA) of the certificate, and they verify that the certificate is valid (not revoked, etc).

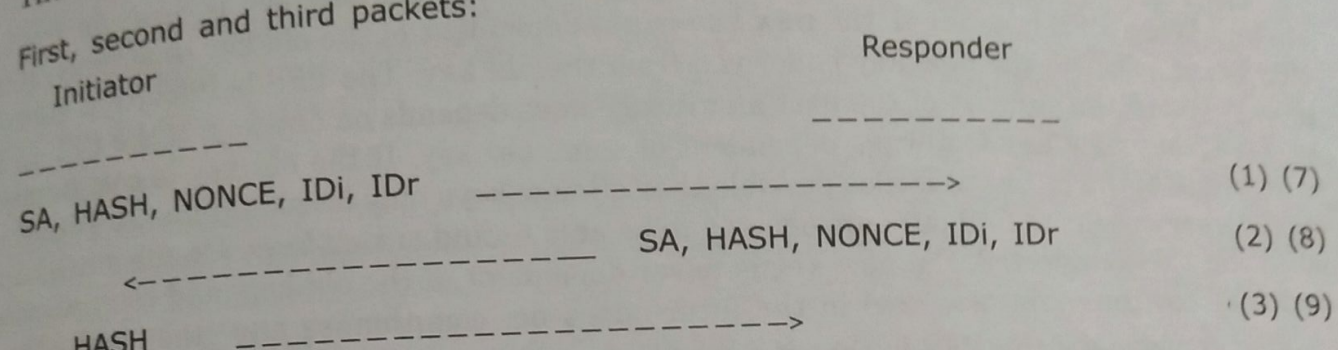
After those packets are sent and the digital signatures are successfully verified the result of this Phase-1 negotiation is the Phase-1 SA, which can be used to protect other packets sent in the IKE, such as the packets of the Phase-2 negotiation. This also completes the Phase-1 negotiation successfully.

### 2.5.2. Phase-2 Mode

After the Phase-1 is successfully completed, the Phase-2 negotiation can proceed. The purpose of the Phase-2 exchange is to provide, and refresh the key material that is used to create the Security Associations (SAs) to protect the actual IP traffic with IPSEC. The Phase-2 exchange is protected with the Phase-1 SA by encrypting the Phase-2 packets with the key

material derived from the Phase-1. The Phase-2 also provides proposal list which defines the actual ciphers. HMACs, hash algorithms and other security properties that are used in the protection of the IP traffic. The proposal that was proposed in the Phase-1 is merely for protection of traffic under the Phase-1 SA (like the packets of the Phase-2), and not for the actual IP traffic. The Phase-2, also called the Quick Mode, is for the protection of the IP traffic.

The Phase-2 Quick Mode is performed as follows:




The Phase-2 Quick Mode is three (3) packets long and it includes several payloads which relates to the key generation. The HASH and NONCE payloads are the keying material which are exchanged, and they are used to create the new key pair. The NONCE payload includes always random data.

The SA payload is the Phase-2 proposal list which includes the ciphers, HMACs, hash algorithms, life times, key lengths, the IPSEC encapsulation mode (ESP, AH, etc) and other security properties. Note that it is possible to send more than one SA payloads in Phase-2, although usually only one is sent.

The ID payloads, marked here as IDi and IDr, for initiator's ID and responder's ID, respectively, are optional in Phase-2. Usually, IKE implementations do send the ID payloads in Phase-2 since they can be easily used to make local policy decisions. However, as noted, they arc not mandatory and can be omitted. The IDi is the initiator's ID, usually IP address or similar, and the IDr is the respondenr's ID, usually IP address, IP range or IP Subnet. Both of the initiator and responder usually use the ID payloads to search the local policy for matching connection. The ID payloads in the Phase-2 are also called as "proxy IDs", "pseudo IDs" or similar, since they do not necessary represent the actual negotiator (for example when Security Gateway (SGW) negotiates on behalf of some client).

After the Phase-2 is completed by sending the last packet, the result of the Phase-2 is two Security Associations (SAs). One is for inbound traffic, the other is for outbound traffic. This also completes the IKE key exchange for basic key exchange.

## 2.6. REKEY

The rekey, or key re-generation is a process where new key material is created for protecting the IP traffic. The main cause of rekey in IPSEC is the expiration of the Security Association (SA) which is used to protect the traffic. The time when SA expires is dictated by the life time of the SA, which can be negotiated during the Phase-1 and Phase-2, for Phase-1 SA and Phase-2 SAs, respectively. The rekey is also performed because of security reasons. Longer the same key is used, more insecure the key becomes. Also the risk that an adversary is able to retrieve the old key is an cause of rekey. If the old key is compromised or about to become compromised, the rekey is performed.