IETF
INTERNET DRAFT

P. Cheng, J.A. Garay, A. Herzberg, H. Krawczyk
IBM T.J. Watson Research Center

28 November 1994

Modular Key Management Protocol (MKMP)

draft-cheng-modular-ikmp-00.ps

Abstract

This memo describes mechanisms and introduces a protocol for the management of cryptographic keys as required for the management of security associations in IPSP and IPv6. Our key management scheme adheres to a modular approach, namely, the scheme is separated into two modules: An upper module in which a long-lived (‘‘master’’) key is exchanged between the communicating parties, and a lower module, in which the already shared (master) key is used for the derivation, sharing and/or refreshment of additional short-lived keys to be used for the cryptographic transformations applied to the data. In this draft, we concentrate on the management module for short-lived keys, and indicate how proposed variants of public key-based master key exchange protocols can be accommodated in the upper module.

Status of this memo

This document is being submitted to the IETF IPSec Working Group for consideration as a method for the management of cryptographic keys as required for the management of security associations in IPSP and IPv6. Publication of this document does not imply acceptance by the IPSec Working Group of the concepts discussed. Comments are solicited and should be addressed to the authors and/or the working group's mailing list (ipsec@ans.net).

To learn the current status of any Internet-Draft, please check the ‘‘1id-abstracts.txt’’ listing contained in the internet-drafts Shadow Directories on ds.internic.net (US East Coast), nic.nordu.net (Europe), ftp.isi.edu (US West Coast), or munnari.oz.au (Pacific Rim).

Contents

1. Introduction

IPSP is a network-layer protocol that provides for the encapsulation of IP packets that are subject to cryptographic transformations like data integrity, authentication and encryption. These transformations require the generation, sharing and maintenance of cryptographic keys. In the vast majority of cases the cryptographic transformations applied to the data are based on symmetric keys as opposed to asymmetric keys used in public-key cryptosystems. The reason is usually efficiency, since public key operations are typically much slower than those of symmetric-key algorithms (e.g., compare RSA decryption with DES decryption). The domain of public-key is best suited for key distribution and signatures.

Our basic approach to the key management problem for IPSP is that any viable solution needs to follow a modular approach. We advocate the separation of a key management scheme into the following two modules: an upper module in which a long-lived (''master'') key is exchanged between the communicating parties; and a lower module, in which the already shared (master) key is used for the derivation, sharing and/or refreshment of additional short-lived keys to be used for the cryptographic transformations applied to the data, e.g., encryption and authentication. (Short-lived keys can be thought of as ''session'' keys, but the existence of a session is not necessarily required or implied.) The split into these two modules is not mandatory, and in fact there are systems that do not establish (at least explicitly) this separation. However, we argue here that for most scenarios - and IPSP is one of them - this explicit separation has a significant methodological and design value. (A more elaborate presentation of these fundamental issues is given in Section 2.)

Agreement on a standard for the sharing of master keys will not be straightforward due to the variety of existing techniques and very complex trade-offs (e.g., public-key algorithms, certification authorities, key distribution centers, etc.). In contrast, the management of short-lived keys once a shared master key already exists can be based on very simple, efficient and well-understood mechanisms. Moreover, while one can start working with IPSP even with manually-installed shared keys, a good management of the derived short-lived keys (including periodic refreshment) is essential for a secure working IPSP. Therefore, we propose that the standardization process begin and focus on the lower layer, allowing for the fact that the standardization of the upper layers would take more time.

This approach is consistent with the way the IPSEC Working Group has structured its development strategy for securing IP, namely, first come up with an agreed-upon network-layer encapsulation protocol (IPSP), and then complement it with a key management protocol. This methodology supports both a gradual development effort, as well as the independence of IPSP from the particular key management techniques. In the same way, a gradual and bottom-up independent approach is desirable for key management.

In this document we concentrate on the basic mechanisms for the management of short-lived keys. The description here omits, for the sake of clarity, some system details and leaves several issues open. These issues are intended to be completed through discussions in the IPSEC and IKMP Working Groups. Our protocol can also be extended to the derivation of master keys. In the Appendix, extensions of the protocol are presented for the derivation of the shared master keys from public key. Most importantly, the lower module protocol can also be combined with any other mechanism for master key distribution, such as key-distribution centers, manual installation, etc.

2. The Modular Approach to Key Management

We now state the basic functionality that needs to be provided by each of the key management modules.

LONG-LIVED KEY MODULE

Will provide a mechanism for authenticated sharing of a secret (symmetric) key between the communicating parties. This protocol would not assume the existence of an already shared key, but is instead based on some form of shared trust. This trust can be physical (e.g., manual/over-the-phone/etc. installation techniques), based on a trusted third party acting as a key distribution center (e,g,, the Kerberos model), or based on the possession by the parties of certified public keys of each other. Periodic (although not frequent) refreshment of long-lived keys needs also to be provided, but this can be accomplished by the basic exchange mechanism (i.e., long-lived key refreshment does not require to assume the existence of an already shared secret key between the parties). Since performance is not critical for this module, it could be implemented as a standard application and usually on top of TCP.

SHORT-LIVED KEYS MODULE

This is the ''busiest'' key management module. This module is responsible for the exchange of short-lived keys, the authentication of the parties, the derivation of ''data keys'' (the actual keys used for encryption and authentication of the IP payload), the periodic (and relatively frequent) refreshment of these keys, and the communication of SAID's for the indexing of the keys (especially in the desirable model where SAID's are chosen by the local implementation and need to be transmitted to the other party). The protocol should make sure that ''data keys'' are always ready to be used by IPSP, i.e., minimize the number of cases where IPSP gets a packet but does not have a valid key to encrypt/authenticate or decrypt/verify it. For performance reasons, and to support implementations on limited-capability systems (such as routers), this module should be implemented on top of UDP, and therefore has to deal also with packet loss and retransmission.

Notice that there is no duplication of functions between the two modules. Regardless of whether these modules are designed in a hierarchical way as we propose or not, the above functions need to exist in any secure and sound key management solution.

One important issue that is not addressed in the above modules is the negotiation between the parties of the security attributes for the security associations served by the exchanged keys. This attributes include the specific functions to be performed on the data (encryption only, authentication only, both of them, etc), the particular algorithms (CBC-DES, key-ed MD5, etc.), the frequency of the key updates (based on time, usage, unilateral decision by any party, etc.), support of timestamps in addition to nonces, interactive vs. non-interactive updates, and so on. The exact list of possible attributes is to be decided by the IPSEC Working Group, and usually involves trade-off considerations between flexibility and simplicity. Additionally, the ''location'' of this negotiation module will have to be decided on. It can be attached either to the master-key module or to the short-lived key module. Or, following our modular approach, it may reside ''in between'' these modules.

There is a full spectrum of reasons behind the conceptual and practical separation of the above two key management levels. A first classification of these issues is presented next. These considerations are in full compliance with the summary on Key Management Requirements resulting from the Twenty-Eight IETF meeting in November of 1993. Some of the elements below are a re-statement of these requirements, while others present complementary aspects (e.g., the need for a bottom-up development methodology):

FUNCTIONALITY

Whatever the master-key exchange mechanism is, a method for derivation of data keys from a shared key is necessary. Notice that multiple security transformations on the data require multiple keys (e.g, encryption and authentication). Moreover, even a single function may require more than one key (for example, stream cipher encryption may require directional keys, i.e., different keys depending on the direction of the data). Also, the communication of SAID's is essential for a model that supports unstructured and locally-generated SAID's.

SECURITY

There is a well-recognized need for frequent key updates. Frequent updates are not just a good cryptographic practice (e.g., to reduce the damage caused by a compromised key; to reduce the time/data available for cryptanalysis; etc.), but are of particular importance in cases where weak algorithms are used. (Weak algorithms will be part of the Internet's repertoire of cryptographic algorithms, both because of export rules as well as for efficiency needs.)

EFFICIENCY

Efficiency is a universal consideration. It is particularly important to have efficient key management protocols in computationally-constrained environments (e.g., weak/loaded devices; wireless devices; etc.). Inefficient methods would effectively lead to less key updates, with the corresponding impact on security. While techniques like Diffie-Hellman (complemented with authentication mechanisms) have an important role for long-lived key exchange, much more efficient techniques (e.g., based on MD5) can be used for the relatively frequent updates of data keys.

DEPLOYMENT and INTEROPERABILITY

A key management protocol needs to support and interoperate with the variety of existing key distribution technologies. Key distribution centers (e.g., Kerberos), and even manual key installation (e.g., in existing intra-enterprise firewalls) are popular and widely-deployed technologies. The view that only public key can provide a full-scale solution to key management for the Internet is well understood and agreed upon. However, equally well understood is that it is not possible to wait until a full certification infrastructure is in place to start providing security for IP. Therefore, not taking advantage of the above handy technologies would be irresponsible.

Note that what we are proposing is an *inclusive* approach, in which all these technologies may co-exist. Still, the short-lived key engine is common to (and independent from) all of them.

DEVELOPMENT METHODOLOGY

Finally, the specification of a short-lived key management protocol supports the *gradual development* of the standard, summarized in the following list:

1. The encapsulation protocol (IPSP);

2. short-lived/data key management (which assumes master key);

3. attribute negotiation (input to policy engine); and

4. master key management.

Note that the above model is broad enough to accommodate a variety of key management proposals, including SKIP (which basically contains the above two modules: a master key exchange based on Diffie-Hellman certificates, and a short-lived key exchange based on encrypting the data keys in the IPSP packet). The question behind the suitability of SKIP involves the need for non-interactive key exchange as opposed to interactive solutions. Our short-lived module contains a non-interactive option. In the Appendix, we also indicate how SKIP can be accommodated for the derivation of master keys.

3. Basic Short-Lived Key Management Model

A basic assumption that we make is that parties are (usually) able to establish periodic interactive communications (as opposed to just sending information in a one-way mode). This assumption, which does not hold true in a 100% of the cases, is however the dominant case for IPSP. Solutions for the one-way communication scenarios are provided too, but viewed as the exception and not as the rule. This distinction is important for assessing the desirable trade-offs between different solutions.

Another basic principle that we follow is that keys must have an expiration policy attached to them (based on usage, time, etc.). Moreover, good security cannot be achieved without secure periodic refreshment mechanisms. In particular, forward secrecy of session keys (i.e., the exposure of a particular key does not implies the compromise of subsequent keys) is a must, given the easiness with which past keys might be revealed (through careless discards, backups, auditing, etc.). A good key management protocol needs to be immune as much as possible to a careless user---‘‘only transparent security can be secure!’’

Key refreshments can be accomplished through interactive or non-interactive mechanisms. The first is done through ‘‘cryptographic handshakes’’ between the parties, namely, a simple authentication and key exchange mechanism. The latter is achieved using transformation on keys based on cryptographic one-way (or more precisely, pseudorandom) functions, or by the encrypted transmission of keys.

Interactive key refreshment has the advantage of providing keys that are fresh and independent from the past (the only dependence is to the master shared key). The cryptographic handshake serves also for direct authentication between the parties, and for the exchange of SAID’s (and well as of any other parameter, if required).

Non-interactive methods have the obvious advantage of reducing or annulling the need for periodic interaction between the parties. However, they have significant disadvantages with respect to interactive methods, like a) requiring some form of strong synchronization for the derivation of new keys (e.g., to avoid replay attacks), b) a stronger dependency with past keys, c) mechanisms to compensate for the lack of mutual negotiation and SAID exchange, and d) overhead to the IPSP packets, in case encrypted keys are attached to these packets. In general, in the IPSP scenario we see the savings of reduced interaction as not significant enough to justify the security disadvantages. However, we anticipate that in particular cases the availability of non-interactive solutions may be important. Therefore, provision for such non-interactive mechanisms is included in this proposal. Of course, this introduces some additional complexity and the WG has to decide on what is the right tradeoff.

4. The Lower Module

A main component of the lower module will be a cryptographic handshake protocol between the communicating parties. In addition to this interactive protocol for the update of short-lived keys, there will also be a non-interactive method for the same goal which may be used depending on the application, or in case the interactive protocol fails repeated trials. We first elaborate on the functional requirements of the interactive protocol. This protocol between the communicating parties will be used for:

1. Determining a shared short-lived key for use with a particular security association;

2. exchanging of SAID's corresponding to this (refreshed) security association; and

3. if necessary, establishing parameters of the secure associations, as refreshment frequency, the particular cryptographic transformations to be used, key lengths, etc.

The need for (1) is explained above. (2) is necessary in the model in which SAID's are determined by the receiver of information (e.g., decryptor, integrity verifier). (3) may or may not be necessary and will be discussed later.

The protocol itself is presented in Section 4.1. We remark that the same protocol can be used for the derivation of multiple keys, as explained below. Indeed, different keys are needed for the application of different cryptographic functions to the data. The short-lived keys derived by the above interactive protocol are in turn used for the derivation of ''data keys,'' for example, the specific keys applied to encryption or authentication of information. In principle, a data key can be the protocol-derived session key itself; however, since usually a different key is required for each cryptographic transformation (say, a key for DES-CBC and a key for key-ed MD5), then the actual data keys are themselves derived from the short-lived key. This will be done by using the short-lived key as an index to a pseudorandom function applied to a value which uniquely determines the algorithm for which the derived key is to be used. This assumes some agreed-upon unique identifier for these algorithms between the parties. In particular, these unique identifiers can be standardized so as to facilitate algorithm negotiation in general and key derivation in particular. These same mechanisms can also be used for the derivation of uni-directional keys (e.g., the encryption key used by A to send data to B, is different from the one used by B to send data to A).

To the interactive protocol proposed here for the update of the short-lived keys, we add the option of non-interactive key updates. Despite the intrinsic disadvantages of non-interactive updates (see Section 3), such an option may be necessary in those scenarios where handshakes between the parties are not possible, or even as a backup mechanism if the interactive protocol fails repeated trials and previously exchanged keys are expired. Notice, however, that in typical situations the failure to establish communication for the key update will also imply the lack of other forms of communication between the parties, and therefore the keys may not be really needed. The non-interactive options are presented in Section 4.2.

The interactive protocol requires no structure on the SAID field, and the receiver can choose it freely. When going into the implementation of more options, like the non-interactive key refreshment mentioned above, there is the possibility of having co-existing SAID's of both ‘‘structured’’ and ‘‘unstructured’’ nature, the former needed for the non-interactive communication. It is enough to have one bit distinguishing between both (although, in this case, even the unstructured SAID becomes 1-bit structured). Alternatively, the STPI field can be used for SAID extensions, as well as for communicating the key in the non-interactive method. We elaborate more on this in the sequel.

We now turn to the description of the interactive protocol, followed by the non-interactive option.

4.1. The Interactive Protocol

This design assumes a pair of IP nodes that already share a long-lived (‘‘master’’) secret key. It provides secure mechanisms for the periodic refreshment of keys and derivation of data keys as required for the different cryptographic functions used with a single security association.

The protocol requires minimal interaction and computational overhead, and has a compact format. (We stress that the extensions described in the Appendix for the derivation of master keys use the same basic format.) Regarding security, the approach guarantees ‘‘forward secrecy’’ for the short-lived keys, i.e., even if somehow an eavesdropper is eventually able to derive one of the short-lived keys, future short-lived (and, of course, master) keys are not compromised. This follows from properties of pseudorandom functions and the way we apply them.

We first present the high-level design, or ‘‘cryptographic skeleton’’ of the interactive protocol, including only the relevant information; additional information (e.g., parameters of the security associations) which requires authentication is omitted here for simplicity. We then specify the implementation in more detail.

There are two parties, S (for sender) and R (the receiver). S is the party that initiates the protocol. We first include a brief explanation of the terminology:

N_X A nonce (i.e., a random number) chosen by X.

K The shared master key.

MAC_K A Message Authentication Code (or function) which is applied to a piece of information for authentication using a secret key K. (Examples include block ciphers, e.g. DES, in MAC mode, or key-ed cryptographic hash functions, e.g., MD5 with prefixed/suffixed key.)

SK The short-lived key, outcome of the protocol.

f_K A pseudorandom function with index K. (Roughly speaking, pseudorandom functions are characterized by the pseudorandomness of their output, namely, each bit in the output of the function is unpredictable if K is unknown. Some of the functions used as MAC are also used as pseudorandom functions, e.g., DES in MAC mode. Some key-ed hash functions, as MD5, are also conjectured to be pseudorandom - although there exists less evidence for that than in the case of DES; the same is true for the use of these functions as MAC.)

We now turn to the process of establishing a shared short-lived key between S and R. This includes mutual authentication and the exchange of SK. The two-flow protocol is shown in Figure 1. It is assumed that S and R already share a master key K, as well as the nonce N_R. This nonce was sent in the previous run of the protocol from R to S. Initially, such a nonce is exchanged at the time of master key exchange.

4.1.1. Properties of the Interactive Protocol

The nonce serves as a challenge for guaranteeing the freshness of the authentication (i.e., to avoid replay attacks). Sharing a nonce is not essential: It can be replaced by use of time stamps (at the expense of requiring good clock synchronization) or by adding an extra flow to the protocol (at the expense of performance). The nonce also serves the purpose of alleviating the effect of a ''clogging'' attack (see below). In any case, the nonces do not require any secrecy, and can be transmitted in the clear.

As mentioned above, the use of time in the first flow is optional. One can let S send the time only (if, for instance, S does not have or did not keep N_R), in which case it will be R's decision (policy) whether to accept or not the time value for a valid authentication. In the case time is chosen to be used by some users, it may make sense only in the S to R flow (R will always have N_S for the

```
S                                              R

           N_S, MAC_K(N_S, N_R)

      ------------------------------------>

                                                R randomly chooses N'_R


           N'_R, MAC_K(N'_R, N_S)

      <------------------------------------

      Let T = MAC_K(N'_R, N_S)

               SK = f_K(T)
```

Figure 1: The interactive protocol for short-lived keys.

response). An adversary that takes advantage of this weak freshness mechanism and manages to re-send an old S-R flow that R accepts as fresh, will succeed in clogging R but not in finding the new key. The incorporation of N'_R, which is chosen by R, into the MAC expression T guarantees the freshness of the key. The adversary, not knowing K, cannot find it. We stress that this function of N'_R is necessary even if one decides that S will not keep N'_R for future use. An important aspect is that the flows of the protocol may be used for the parties' mutual authentication, regardless of a key exchange. In that case, the freshness of the time/nonce is essential.

Notice that the short-lived key SK is derived by each party by applying the (secret) function f_K to the value T; therefore, the key itself is never explicitly transmitted. This avoids the need for encrypting this key, as well as the need for explicitly authenticating it. The authenticity of SK is derived from the authenticity of the MAC expression T.

Finally, simplicity and being amenable to analysis and proof are important features of any cryptographic protocol. The protocol presented here is structured, simple, and thus easier to analyze. (Indeed, methods similar to those of [BGH+91, BR93] can be used to establish the protocol's desired cryptographic properties.)

NOTES: For the sake of simplicity, in the description of the protocol above we have omitted the following aspects, which need to be included in a complete specification. (See also Section 5: Pending Issues.)

1. In this high-level presentation, the same key K is used for both authentication and short-lived key derivation (i.e., as a key for the MAC, and as an index to the pseudorandom function f, respectively). However, a secure implementation of the protocol requires the use of two different keys for those two different purposes. These two keys can be readily derived from a single master key using a pseudorandom function. This separation of keys is intended to preserve the secrecy of the derived session keys, even if the authentication key happens to be compromised (e.g., via cryptanalysis - notice the availability of plaintext-cyphertext pairs to the attacker for the authentication key, but not for the pseudorandom function key).

2. The above description assumes a sender S that initiates the protocol and a receiver R. Any scenario in which both parties

can initiate the protocol will require the storage of two nonces to be used according to the initiating party.

3. A third ''ack'' flow can be added to the protocol. This message from S to R will notify R about the successful completion of the protocol. This will minimize the number of re-tries in IPSP communication due to the unavailability of keys. (More details on the ''ack'' packet and on the protocol's state machine description will be included in the next version of this document.)

4. Details related to the recovery from faults (e.g., failed authentication) need to be addressed.

4.1.2. Implementation

This section describes the implementation aspects of the interactive protocol in more detail. Importantly, we stress that the same packet format can be used for master key exchanges. Thus, the scheme has added functionality without the penalty of having to manage more packets.

The contents of the protocol's first flow are shown in Figure 2. Some of the details are not shown.

Where:

''Prot.type,vers.'' is used to indicate the protocol's type (e.g., interactive, non-interactive, IBM's, etc.) and its version number.

''Msg. type'' indicates the type of the message (e.g., from S to R, ack, etc.).

''SAID'' is IPSP's Security Association Identifier. ''Current SAID'' is the SAID that R chose for use during the current session and its transmission from S tells R to which security association this run of the protocol corresponds. ''Next SAID'' is the SAID that S is proposing for the next short-lived key.

''Other fields'' include information that must be authenticated, but whose inclusion depends on the scope of the key management protocol. Examples are identification information, tunnel parameters (e.g., tunnel lifetime), security transformations (i.e., what algorithms are to be used), etc.

```
  0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+     +
 |Prot.type,vers.|   Msg. type   |            Length             |     |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+     |
 |     Other fields' length      |           Reserved            |     |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+     |
 |                   S's IP address                              |     |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+     |
 |                   R's IP address                              |     |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+     |
 |                Current SAID  (32 bits)                        |     |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+    MAC
 |                 Next SAID  (32 bits)                          |     |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+     |
 |                   N_S  (64 bits)                              |     |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+     |
 |                   N_R  (64 bits)                              |     |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+     |
 |       Other fields (prot. mode, tunnel param., etc)           |     |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+     +
 |                   MAC  (128 bits)                             |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2: Contents of protocol's first flow.

The ''MAC'' field contains the result of MAC_K, where K is the shared master key, computed on all previous fields (or in all fields that require authentication). The position of N_R and N_S as the last fields in the computation of MAC is intentional. The effectiveness of these nonces as freshness guarantee is enhanced by fixing their offset relative to the beginning or end of the authenticated arguments.

The layout of the protocol's second flow is identical to the first flow above. Regarding the SAID fields, the role of the parties that choose/propose them is now interchanged.

4.2. The Non-interactive Protocol

In this section we describe an option for the refreshment of short-lived keys, which does not require handshake or interaction between the two communicating parties. Thus, the sender can choose this option when a short-lived key is not in place. This is indeed the method proposed in [A94] (SKIP), and in fact, SKIP is one of the variants that can be accommodated in our framework.

One of the disadvantages of these non-interactive solutions is the impossibility to use shared nonces between the parties. Thus, the use of counters or timestamps help against the ''replay'' attacks. Another consideration, in the case SKIP is used, is that some space in the packet format (e.g., STPI) must now be used by the sending party in order to communicate the key. Finally, the non-interactive option imposes some structure on the format of the SAID fields. While in the interactive protocol from the previous section the SAID field is completely left for the recipient to choose, this is not the case with non-interactive communication. (In fact, even the non-structured SAID becomes ''1-bit structured,'' in order to distinguish between the two types).

Here we propose another non-interactive version which does not transmit the short-lived key, and is therefore more efficient.

The idea is simple: If K is the master key shared between the parties, then the packet key is $f_K(NI, index)$, where:

NI (''Non-Interactive'') is a constant to distinguish from the master key(s) used by the interactive protocol; and

$index$ is a random number sent with the message. Alternatively, $index$ can be a counter (provided it does not repeat), or a timestamp (in which case we would assume that the adversary cannot modify the time in the communicating machines).

5. Pending Issues

Interactive vs. non-interactive short-lived key updates

The non-interactive update is to be used, for example, in case the interactive protocol fails repeated trials. The necessity of this backup mechanism needs to be decided in the working group and, if adopted, requires the resolution of the issues mentioned above (i.e., generation of SAID's for the non-interactively-derived keys, synchronization between the parties for switching to these keys, etc.)

Negotiation of algorithms/key expiration/key length

Both at the upper and lower modules. Whose component's (IPSP or IKMP) responsibility is it to check for key expiration?

Data keys

The short-lived keys are used to derive data keys for the different cryptographic functions, which may require different update frequencies. How is this to be handled?

Non-random keys

In our proposal short-lived and data keys are derived out of pseudorandom functions and therefore they are themselves (pseudo)random. Most shared-key algorithms require such random keys but some may require a specially structured key (e.g., a random prime number). Should there be a mechanism to exchange such keys? This can be done as an extension to the protocol presented here (e.g., by using the derived random key as a one-time pad for encryption of the structured key). Alternatively, the STPI field may be used for the encryption of the structured key under the short-lived key.

IPSP/IKMP interface

Is there a need for a standard IPSP/IKMP interface?

Multiparty associations

Multicast. What are the models to be supported? What would the SAID assignment policy be in that case? What other cases of ''a-symmetric'' traffic exist? How can the existing multicast proposals (e.g., GKMP [HMR94]) be accommodated in our framework?

Multiple tunnels

Our scheme can support the establishment and maintenance of multiple tunnels between the same two IP nodes. How often would these situations arise in practice? Notice that, if allowed, this will require the maintenance of a ‘‘tunnel-ID,’’ similar to a SAID.

Key termination

In addition to the regular key expiration/refreshment policy one may want in some cases the termination of an existing key (e.g., under suspicion of key compromise). Mechanisms for this need to be added to this draft.

Denial of service

We need an appropriate model for considering these attacks. How much is to be invested (performance, bandwidth) to prevent it? For example, there is no reason to protect against these attacks in IPSP if they are equally possible using other IP protocols.

References

[A94] A. Aziz, ''Simple Key-Management for Internet Protocols (SKIP),'' Internet Draft draft-ietf-ipsec-aziz-skip-00.txt, October 1994.

[BGH+91] R. Bird, I. Gopal, A. Herzberg, P. Janson, S. Kutten, R. Molva, and M. Yung. ''Systematic design of two-party authentication protocols,'' Advances in Cryptology, Proc. CRYPTO 91, August 1991.

[BGH+93] R. Bird, I. Gopal, A. Herzberg, P. Janson, S. Kutten, R. Molva, and M. Yung. ''Systematic design of a family of attack-resistant authentication protocols,'' IEEE Journal on Selected Areas in Communications (special issue on Secure Communications), 11(5):679--693, June 1993. See also a different version in Crypto 91.

[BR93] M. Bellare and P. Rogaway, ''Entity Authentication and Key Distribution,'' Advances in Cryptology, Proc. CRYPTO 93, August 1993

[HMR94] H. Harney, C. Muckenhirn and T. Rivers, ''Group Key Management Protocol (GKMP) Architecture,'' Internet Draft draft-harney-gkmp-arch-00.txt, September 1994.

A. Deriving Shared Master Keys from Public Key

We now present a protocol for the derivation of the shared master keys from public key. The protocol can be viewed as an extension of the short-lived key negotiation protocol presented above, and, in fact, the same packet formats can be used. (We remark that the short-lived key protocol can also be combined with any other mechanism for master key derivation, such as key-distribution centers, manual installation, etc.) Moreover, when exchanging a master key it is desirable to also have a mechanism to derive a short-lived key. This is obtained by combining the two protocols. This allows S and R to establish, starting with their public keys, both a master key AND a short-lived key in just two flows.

As in the case of the short-lived keys, a non-interactive option can also be accommodated in the master-key module. However, having options at this level is a more complex issue. We elaborate on this at the end of the section.

We now assume that the two parties, S and R, posses an authentic public key of each other, as well as a shared nonce. Again, sharing a nonce is not essential: as explained before, it can be replaced by the use of time stamps or by adding an extra flow to the protocol. Before we present the protocol, we need some additional terminology:

E_X RSA encryption with X's public key. We assume that the information is padded with a random string prior to encryption.

$SIGN_X$ X's RSA signature. More specifically, by $SIGN$ we mean first apply MD5 to the signed information, and then apply RSA (i.e., exponentiation with X's private key). Since RSA operations require an argument as long as its modulus, and the MD5 output is shorter than this modulus, then the RSA operation will be performed on the concatenation of MD5 and some other fields in the packet to complete the modulus length (probably, with added randomness and redundancy). Details TBD.

K The shared master key, outcome of the protocol.

The two-flow master key negotiation protocol is shown in Figure 3. S is the party that initiates the protocol. Initially, S and R share nonce N_R. N'_R is the nonce to be used next time, i.e., upon completion, S and R set N_R := N'_R.

```
S                                              R

S randomly chooses K_S, N_S

Let E_1  = E_R(K_S)


       E_1, N_S, SIGN_S(E_1, N_S, N_R)

     ------------------------------------>

                                                R randomly chooses K_R, N'_R

                                                Let E_2 = E_S(K_R)

       E_2, N'_R, SIGN_R(E_2, N'_R, N_S)

     <-------------------------------------


               K = K_S XOR K_R.
```

Figure 3: The master key protocol.

We now present some observations about the protocol. First notice that $SIGN_R$ in the return flow is not really necessary, it can be replaced by $MAC_K(E_2, N_S, N'_R)$. The advantage of this is efficiency (i.e., less exponentiations), at the price of not being homogeneous in both flows. The replacement of $SIGN$ by MAC does not hold for the first flow, where the signature is mandatory (otherwise anybody could choose K_S and compute $E_R(K_S)$ and $MAC_{K_S}(...)$).

R first verifies the signature, and only if this succeeds does it decrypt K_R (this reduces the computational overhead, for example, against a ''clogging'' attack, since signature verification is much cheaper than decryption).

The protocol is immune to the exposure of one of the private keys (indeed, an adversary needs to discover the private keys of both parties in order to derive the exchanged master key). For the vast majority of applications, this should be sufficient. However, the protocol can be strengthened to guarantee the maintenance of security even if both keys are exposed, at the expense of using Diffie-Hellman.

The master key protocol is in some sense minimal, since 2 flows are always needed, as well as secrecy and authentication each way (thus the 2 exponentiations). This can be made even cheaper by letting only one party choose the key (in which case the compromise of the private key of this party would compromise the exchanged key). The protocol uses 4 exponentiations in total (2 performed by each party). Using the replacement of $SIGN$ by MAC in the second flow mentioned above reduces the number to 3 (2 by S and 1 by R). By letting just one party choose the key, the number of exponentiations can be further reduced to 2 (1 per party). Our proposal is based on the variant with 3 exponentiations. We remark that the only differences with the short-lived key protocol are the $SIGN$ field in the first flow (instead of MAC), and the fields of encryption.

A.1. Non-Interactive Master Key Exchange

As mentioned before, albeit more complex, it is also possible to have a mixed (interactive and non-interactive) approach for master key derivation based on public-key. Contrary to the case of short-lived key exchange (where adding a non-interactive option to an interactive exchange protocol is simple and without costly side effects), supporting the option of a non-interactive exchange of master-keys has a significant cost to the system in terms of complexity and/or security.

A straightforward way would be to just have two options: the public-key protocol of Figure 3 and SKIP [A94]. However, a problem with this is that two kind of public-key algorithms/keys (RSA's and SKIP's) need to be supported.

However, if there exist a real requirement for non-interactive master-key exchange, we believe that the following is a more realistic approach. The basic idea is to use SKIP for the non-interactive cases, while parties that can afford the interaction (our thesis is that this constitutes most of the traffic) will use the SKIP-shared master key only as a symmetric key for authentication of a regular Diffie-Hellman exchange. The result of this Diffie-Hellman exchange will then be used as the master key for the purpose of deriving short-lived keys. This combination keeps all the advantages of SKIP (e.g., sending an encrypted message to a party that never had communication with you with only moderate packet overhead), but does not pay with all the disadvantages of SKIP for those parties that can afford interaction. For example, using the SKIP-shared key only as an authentication key implies that the exposure of one the party's private key does NOT expose ALL the traffic to and from this party; the discovery of the two parties' SKIP-shared key (a^{ij}) by an adversary only helps this adversary in ACTIVE impersonation attacks, but not in order to mount passive/off-line attacks, as is the case in SKIP. Hence, this improves over a SKIP-only solution. On the other hand, for parties that can execute interactive protocols this is less good than a pure interactive solution as the one described above in this Appendix. We defer a full treatment of this combination to the next version of this document.

Authors' Addresses

The four authors are with

IBM T.J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

FAX: +1 914 784-6205

Pau-Chen Cheng

Tel. +1 914 784-7446
E-mail: pau@watson.ibm.com

Juan A. Garay

Tel. +1 914 784-6852
E-mail: garay@watson.ibm.com

Amir Herzberg

Tel. +1 914 784-6981
E-mail: amir@watson.ibm.com

Hugo Mario Krawczyk

Tel. +1 914 784-6730
E-mail: hugo@watson.ibm.com