Secure multicast transport

Mick Seaman

This note discusses the mechanisms that secure the distribution of keys in KSP [1]. These mechanisms can also support other multicast protocols. Performance related aspects of parts of the KSP design are also discussed in [2].

Two-party protocol

Consider the well-known protocol

$S_A \rightarrow \{A, R_A\}_M$ (2-1)

$S_B \rightarrow \{B, R_B, A, R_A\}_M$ (2-2)

$S_{A:}\ K = R_A \oplus R_B$ (2-3)[1]

$S_A \rightarrow \{A, R_A, B, R_B\}_M$ (2-4)

$S_{B:}\ K = R_B \oplus R_A$ (2-5)

or one of its close relatives.

In this protocol, stations S_A and S_B exchange random numbers R_A and R_B to establish a common pair-wise key, K. Each of the messages is protected by a master key M. If that key has only been entrusted to parties that can be trusted to operate the protocol correctly, then the protocol:

a) proves mutual possession of the master key
b) proves liveness, i.e. the stations possessing the key are operational
c) results in a shared key, that can be used to protect subsequent communication between S_A and S_B.

The subsequent communication could itself be used to protect the exchange of further keying material. However for this communication to be generally robust, it needs to be protected against attacks that misorder or replay messages. This can be simply done by the addition of a sequence number, initialized to zero or one when the nonce R is chosen, to each message. So the communication might proceed as follows:

$S_B \rightarrow \{A, B, N_B, D_B\}_K$ (2-6a)

$S_A \rightarrow \{B, A, N_A, D_A\}_K$ (2-7a)

Here the symbol N is used to denote each sequence number. The data exchanged is D.

If proof of ongoing timeliness of this conversation is required, to protect against an attacker delaying data to disrupt the operation of a configuration protocol, this can be easily achieved by each of the stations reflecting back the last N value received from the other:

$S_B \rightarrow \{A, N_{A-}, B, N_B, D_B\}_K$ (2-6)

$S_A \rightarrow \{B, N_B, A, N_A, D_A\}_K$ (2-7)

Some of the data D may be reflected as well, but that is outside the scope of this discussion.

[1] The symbol $\oplus$ denotes 'exclusive-or'

N-party protocol

It is clear from the basic two-party protocol that knowledge of the variables R_A and R_B is sufficient to derive K, while possession of K implies possession of M. So, assuming that any attacker has full control of the communication channel and a transcript of all past messages, the message

$\{ ... \}_K$ (i)

with message body "..." is equivalent, from the point of view of protection and authentication of communication between S_A and S_B, to the message

$\{ R_A, R_B, ... \}_M$ (ii)

up to the point that there is a risk of key M having been used too many times.

Likewise, the message

$\{ N_A, N_B, ... \}_K$ (iii)

is equivalent to

$\{ R_A, N_A, R_B, N_B, ... \}_M$ (iv)

The objective of the KSP's multicast key distribution mechanisms is to replace multiple two-party exchanges, between a number of stations, with a multicast exchange, thus using a protocol that is O(n) in the number of messages sent rather than $O(n^2)$ – n being the number of participating stations. If, for example, there are three participating stations, S_A, S_B, and S_C then instead of using the two party protocol to derive

$K_{AB} = R_{AB} \oplus R_{BA}$

$K_{BC} = R_{BC} \oplus R_{BC}$

$K_{CA} = R_{CA} \oplus R_{AC}$

and then sending messages

$\{A, N_A, B, N_B,... \}_{KAB}$

$\{A, N_A, C, N_C,... \}_{KAC}$

$\{B, N_B, A, N_A,... \}_{KAB}$

$\{B, N_B, C, N_B,... \}_{KBC}$

$\{C, N_C, A, N_A,... \}_{KAC}$

$\{C, N_C, B, N_B,... \}_{KBC}$

the n-party protocol establishes R_A, R_B, and R_C, and sends messages of the form

$\{A, R_A, N_A, R_B, N_B, R_C, N_C,... \}_M$

$\{B, R_A, N_A, R_B, N_B, R_C, N_C,... \}_M$

$\{C, R_A, N_A, R_B, N_B, R_C, N_C, \ldots\}_M$

The gain in simplicity over a protocol that attempts to reconstruct multicast capability from n-way point to point dialogue is actually greater than appears from simply counting the messages required to convey fixed data, as anyone who has attempted this general conversion knows. In practice a restriction to the use of point-to-point secure capabilities is probably better handled by accepting the complexity and performance impact of electing a designated station, which then allows an O(n) protocol solution.

To allow a straight forward comparison between the n-party and two-party protocols, it is convenient to examine the former as supporting a contributory key agreement protocol, as the two party protocol does, rather than as a transport for key distribution. For three parties:

$S_A \rightarrow \{A, R_A, N_A\}_M$ (n-1A)

$S_B \rightarrow \{B, R_B, N_B\}_M$ (n-1B)

$S_C \rightarrow \{C, R_C, N_C\}_M$ (n-1C)

$S_A \rightarrow \{A, R_A, N_{A+}, R_B, N_B, R_C, N_C\}_M$ (n-2A)

$S_B \rightarrow \{B, R_A, N_A, R_B, N_{B+}, R_C, N_C\}_M$ (n-2B)

$S_C \rightarrow \{C, R_A, N_A, R_B, N_B, R_C, N_{C+}\}_M$ (n-2C)

$S_{A:}\ K_B = R_A \oplus R_B \oplus R_C$ (n-3A.B)

$S_{A:}\ K_C = R_A \oplus R_B \oplus R_C$ (n-3A.C)

$S_{B:}\ K_A = R_A \oplus R_B \oplus R_C$ (n-3B.A)

$S_{B:}\ K_C = R_A \oplus R_B \oplus R_C$ (n-3B.C)

$S_{C:}\ K_A = R_A \oplus R_B \oplus R_C$ (n-3C.A)

$S_{C:}\ K_B = R_A \oplus R_B \oplus R_C$ (n-3C.B)

and as a result of execution of the protocol not only do S_A, S_B, and S_C have the same values for the keys K_A, K_B, and K_C to be used for data transmission, but $K_A = K_B, = K_C$.

It is of course unlikely that the protocol, in the absence of a master clock and station synchronization – neither of which is naturally available on a LAN, would result in a sequence of messages so neatly organized into two rounds of transmission followed by the key calculations. This doesn't matter: the calculations of K_A at (n-3B.A) and (n-3C.A) only depend on receipt of the message sent at (n-2A), and similar observations apply to the calculations of K_B and K_C.

The conditions for assigning values to keys in this protocol are the same as, and serve to illustrate, the conditions for accepting data conveyed by the more general application of the protocol. A station S_X only accepts data from a station S_Y if that is transported in a message that:

a) contains R_X and an acceptably recent N_X [2]

and

b) if a message containing R_Y has previously been received, then N_Y is greater than the N_Y received in that previous message[3].

Of course the same conditions cannot be applied to recording a received R_Y, N_Y tuple for inclusion in subsequent messages, or the protocol would never get started. In this case the appropriate checks are simply that either

a) there is no existing record of R_Y

or

b) N_Y is greater than that currently recorded for R_Y.

It is worth noting in passing that although KSP is deliberately based on key selection and distribution, rather than on contributory key agreement, and thus uses the N-party protocol describe here purely as a secure multicast transport, the use of a contributory key agreement protocol based on the foregoing description may just meet our goals for MACsec[4], and may be somewhat easier to prove secure. In any case it is always nice to have realistic alternatives to spur examination of assumptions. I believe some improvements are required for the result to be satisfactorily robust, but that can be achieved without diminishing the security properties of the basic N-party protocol just described[5] [6].

KSP Terminology and Design

It should be readily apparent that the foregoing is a explanation of how the distribution of keys is secured in KSP, rather than a description of the KSP design process. Hopefully the explanation will make the design more intelligible to those familiar with the two-party key exchange protocol[7]. There follows a description of the correspondence between the above terminology and that used in KSP [1]. This description naturally leads into a summary of the design approach actually used to develop the key transport component of KSP.

In KSP, each of the random values R is referred to as an "Member Identifier", MI, and the corresponding N value as the "Member Age", MA. The latter reflects the fact that the MA values are incremented with reference to a local clock so it is easy for a station S_X to determine whether a reflect value of MA_X by S_Y guarantees

[2] I don't believe checking N_X is required if proof of timeliness of data delivery is not an objective. I don't believe that there is any requirement to verify that the N_X received is at least as recent as any other N_X parroted back from S_Y, though caution may be required to ensure that this is not implied by a proof.

[3] Implementation of the timeliness check on N_X helps here as it limits the time for which a record of receipt of R_Y needs to be retained.

[4] I think KSP can scale to more participants, but KAP (Key Agreement Protocol) may prove adequate for providing bridging requirements.

[5] And that this contention is itself easy to prove.

[6] [3] provides some food for thought.

[7] I should acknowledge the very useful discussion with John Viega at the Portland meeting , which encouraged me to write this note. However the claims and presentation here, together with any deficiencies, are my own fault.

that the message from S_Y has been delivered without undue delay.

The design philosophy of KSP is that each participating station participates within the protocol as an "instance" of itself, the duration of the instance representing a continuous period during which the station can remember both its "Instance identifier" ("member identifier") and all values derived from or subordinate to that instance. Instance identifiers are chosen at random, and a new identifier is always chosen after a station has been reset and usually after each time that station powers up. A new instance identifier is also chosen if any of the number spaces derived from or related to the instance identifier is exhausted or close to exhaustion[8]. Instance identifiers, i.e. member identifiers, are chosen from a space so large that they are vanishingly unlikely to be reused by accident.

A newly chosen member identifier, MI, together with the space of all possible "member ages" ("instance ages"), MA, and a given master key M represents a set of problems, i.e. encryption or integrity protection of a message containing the tuple MI, MA with the key M, that are deemed to be practically impossible (from the protocol's point of view) unless the station solving the problem possesses key M.

No station S_Y, possessing M and executing the protocol correctly[9], includes an MI_X, MA_X where $MI_X \neq MI_Y$ in a message before it has received that MI_X, MA_X in a message from another station. Given the size of the MI_X number space we are justified, within the probabilistic guarantees provided by the protocol, in assuming that $MI_X \neq MI_Y$ if $S_X \neq S_Y$. Hence if S_X transmits MI_X, MA_X tuples in MA_X order, and receives an MI_X, MA_{Xn} tuple in a message, with a value of MA_{Xn} not transmitted by S_X earlier than a known time before the current value of MA_X, then S_X can be sure the message was transmitted by a station possessing M within the interval between that time and the time of receipt. Because KSP is idempotent in respect of the further data carried in protocol messages, that is a sufficient security guarantee.

However the guarantee can be tightened, without additional message fields, by considering the receipt of pairs of tuples MI_X, MA_X ; MI_Y, MA_Y by S_X from (or apparently from) S_Y and not only applying the timeliness check to but also requiring the values of MA_X and MA_Y to be not less than the values of those variables in any prior message containing MI_X and MI_Y. This tighter guarantee protects the rest of KSP from replay attacks.

In KSP the station identifiers, referred to as A, B, C, .. in the protocol descriptions above, are actually the Secure Channel Identifiers (SCIs) used in the MACsec protocol. Since MI_Y (corresponding to R_Y above) does not appear in the MACsec data frames transmitted by S_Y, and the key used by S_Y could for a period differ from the key used by some S_Z, it is necessary to bind MI_Y to SCI_Y. Obviously, in line with the guarantees above, a station S_X wishing to receive from S_Y does not perform that binding unless the binding has been received in a protected message purportedly from the station with MI_Y and SCI_Y and meeting the MI_X and MI_Y guarantees described above.

References and background

[1] A distributed fault-tolerant group key selection protocol for MACsec.

KeySelectionProtocol-seaman-v03.pdf

[2] Mick Seaman. Key exchange with packet loss, delay, and misordering.

[3] Steiner, Tsudik, Waidner. Key Agreement in Dynamic Peer Groups. See particularly section 2.

http://citeseer.ist.psu.edu/cache/papers/cs/...steiner00key.pdf

[8] Although those number spaces may be exceedingly large and thus almost "impossible" to exhaust, sound protocol design requires that the protocol recover from any state (whether thought impossible or not) to a known state following a known bounded time during which all messages conform to the protocol and are received by their intended recipients. Reusing an instance identifier ensures that this rule is not broken by the possibility of an exhausted space.

[9] Correctness of any instance of KSP depends on all members possessing M and participating in that instance executing the protocol correctly.