On key control in key agreement protocols

Chris J. Mitchell* Mike Ward† Piers Wilson‡

18th March 1998

Abstract

In this paper we consider certain key establishment protocols specified in an international standard and a draft international standard. These protocols are all intended to provide joint key control, i.e. the protocols are designed to prevent either party choosing the key value. We show that this claim is suspect for most of the protocols concerned.

Index Terms: key agreement, key distribution, authentication protocol

1 Introduction

As defined in ISO/IEC DIS 11770-3, [1], key agreement is the process of establishing a shared secret key between two entities A and B in such a way that neither of them can predetermine the value of the key. DIS 11770-3 specifies seven key agreement protocols. In addition, two protocols in ISO/IEC 11770-2, [2], are apparently designed to provide joint key control; they certainly enable both parties to influence the value of the key.

*Information Security Group, Royal Holloway, University of London, Egham, Surrey TW20 0EX, UK.

†APACS, Mercury House, Triton Court, 14 Finsbury Square, London EC2A 1BR, UK.

‡Vodafone Ltd., The Courtyard, 2–4 London Road, Newbury, Berkshire RG14 1JX, UK.

There are a number of practical circumstances in which it may be useful to use a key agreement protocol. For example, neither party may trust the other to choose a suitable key, and use of a key agreement protocol enables them to choose a mutually acceptable shared key. Moreover it is often recommended to use key agreement to prevent one party having any kind of advantage over the other.

2 The basic mechanism

In both the mechanisms in ISO/IEC 11770–2 (Key Establishment Mechanisms 5 and 6), and in six out of the seven key agreement mechanisms in ISO/IEC DIS 11770–3, i.e. all but Key Agreement Mechanism 1, the method for agreeing the key between the two parties operates in much the same way. Both parties choose a secret 'key component', and these secret components, or a function of them, are exchanged. The shared secret key is then computed as a one-way function of the two key components. In principle this provides key agreement since neither party has any means of controlling the final value of the key.

3 A practical limitation

In practice, one party, say B, will receive the key component (or a function of it) of the other party, say A, before B sends its component back to A. This will put B at an unfair advantage. In particular, if B is prepared to perform approximately 2^s computations of the one-way function used to combine key components prior to sending a response to A, then B will be able to choose s bits of the shared secret key. To do this B first nominates s bit positions in the shared key and chooses values for these s bits. B now generates a series of random values for its key component, and combines each with the key component

information supplied by A. As soon as an output key which has the s nominated bits equal to the chosen values is found, then the corresponding key component value is selected. After 2^s such random selections, the chances are that an appropriate value will have been found.

Of course the need for B to perform 2^s computations prior to returning a response to A, will typically severely limit the value of s. This holds since A will often only accept a response if it arrives within a short space of time (e.g. a second). However, if a hash-function such as SHA–1 is used to combine the keys, then given the availability of high-speed implementations of such functions, it may be possible to test as many as $10^4 - 10^5$ key components in a second or so, allowing as many as 16 bits of the shared key to be chosen by B. In situations where a reply is not expected so quickly, larger values of s will be possible.

4 Some specific examples

We now consider two examples of how this general point relates to specific key agreement mechanisms from [1, 2]. A slightly simplified version of Key Establishment Mechanism 5 from [2] involves the following exchange of messages between A and B:

M_1: $A \rightarrow B$: $e_{K_{AB}}(T_A||ID_B||F_A)$

M_2: $B \rightarrow A$: $e_{K_{AB}}(T_B||ID_A||F_B)$

where $e_{K_{AB}}(X)$ denotes the encryption of data X using the secret key K_{AB} (shared by A and B), $X||Y$ denotes the concatenation of data items X and Y, T_A and T_B are timestamps, ID_A and ID_B are identifiers for entities A and B respectively, and F_A and F_B are key components chosen by A and B respectively. At the conclusion of the protocol

A and B calculate the shared secret key as $K = w(F_A||F_B)$ where w is a one-way function. Using the argument from the previous section it should be clear that B has an advantage over A in this mechanism, as long as B waits to receive M_1 before sending M_2.

A second example is provided by an implementation of Key Agreement Mechanism 5 from [1] (this is the Matsumoto-Takashima-Imai $A(0)$ protocol). This involves the following exchange of messages between A and B:

M_1: $A \rightarrow B$: $g^{r_A} \bmod p$

M_2: $B \rightarrow A$: $g^{r_B} \bmod p$

where r_A and r_B are random key component values chosen by A and B respectively, g is a primitive element mod p, and p is a 'safe' prime (where g, p are globally agreed).

To use this protocol A and B must have private/public key pairs (h_A,p_A), (h_B,p_B) respectively, where $p_A = g^{h_A} \bmod p$ and $p_B = g^{h_B} \bmod p$, and A and B must have trusted copies of each others' public keys. At the conclusion of the protocol A calculates the shared secret key as

$$K = w(p_B^{r_A}||(g^{r_B})^{h_A})$$

(working mod p), where w is a one-way function. B calculates the same key as

$$K = w((g^{r_A})^{h_B}||p_A^{r_B}).$$

It should again be clear that B has an advantage over A (as long as B waits to receive M_1 before sending M_2).

5 Providing genuine key agreement

To remove the problem identified above we need to ensure that both parties choose their own key components before seeing the other party's component. This could be (partially)

achieved by requiring the recipient of one component to respond with its component in a very short space of time. Alternatively, a Trusted Third Party could act as a mediator, and require both components to be supplied to it before passing them on.

Clearly both these approaches have shortcomings. A more practical approach involves one party (B say) generating their component, and also calculating a hashed version of this component using a one-way hash function h. This hash-code, which we call a *commitment*, is passed to the other party (A). A now generates its component and passes it to B. B now passes its component back to A, who checks that it matches the previously provided commitment. Observe that, as long as h is one-way and components contain sufficiently many bits, A cannot misuse B's commitment to help choose its component. Moreover, B is forced to choose its component before seeing A's component.

6 Conclusions

It has been shown that certain key agreement protocols, either standardised or approaching standardisation, are not strictly 'fair', in that one party can exert some control over the mutually agreed key. A general approach to avoiding this problem through the use of *commitments*, a well-established concept in the study of cryptographic protocols, has been described.

References

[1] International Organization for Standardization, Genève, Switzerland. *ISO/IEC 2nd DIS 11770–3, Information technology—Security techniques—Key management; Part 3: Mechanisms using asymmetric techniques*, July 1997.

[2] International Organization for Standardization, Genève, Switzerland. *ISO/IEC 11770–2, Information technology—Security techniques—Key management—Part 2: Mechanisms using symmetric techniques*, 1996.