Comments on the S/KEY user authentication scheme

Liqun Chen* and Chris J. Mitchell
Computer Science Department, Royal Holloway, University of London,
Egham, Surrey TW20 0EX, England.

{liqun,cjm}@dcs.rhbnc.ac.uk

Abstract

We give a critical analysis of the security properties of the S/KEY user authentication system.
Index Terms: authentication, password, user authentication

1 Introduction

In this paper we consider the security properties of the S/KEY user authentication system, details of which have been published as an Internet RFC, [3]. A complete software implementation of S/KEY has also been made publicly available (see [3]).

The S/KEY scheme, which is closely based on a scheme devised by Lamport, [5], has been designed to provide users with 'one-time passwords', which can be used to control user access to remote hosts. Of course, as with any such system, after the user authentication process is complete, i.e. after the one-time password has been sent across the network, no protection is offered against subversion of the link by third parties. This fact is pointed out in [3]. Indeed it is stated there that the S/KEY scheme 'does not protect a network eavesdropper from gaining access to private information, and does not provide protection against "inside" jobs or against active attacks where the potential intruder is able to intercept and modify the packet stream'.

However it is claimed that S/KEY 'is not vulnerable to eavesdropping/replay attacks'. In this paper we show that, depending on the definition of 'replay attack', the S/KEY scheme can fail to provide this property, which is not true of some other schemes of this general type.

*The work of this author has been funded by the European Commission under ACTS project AC095 (ASPeCT).

2 Outline of scheme

The S/KEY scheme operates in the following general way. For the purposes of this discussion we assume that the user has access to some kind of computing device with both an input and an output device (e.g. a keyboard and screen) and which can be used to perform certain calculations.

The user possesses a 64-bit secret key. This key is derived from a 'pass-phrase' of arbitrary length, thus avoiding the need for it to be stored by the user's hardware. The user and host share an implementation of a one-way function which takes a 64-bit input and gives a 64-bit output. The function built into the public domain version of S/KEY is based on the MD4 hash-function (see, for example, Internet RFC 1320, [6]), although it could be based on any other suitable function (e.g. SHA or RIPEMD–160).

The key is used to generate a sequence of 'one-time passwords' in the following way. Each host which the user wishes to access is assigned a 64-bit seed value and a 'count' value, which will initially be set to some fixed value (e.g. 1000). For each user wishing to gain access, the host retains three things:

- a copy of the seed for that user,
- the current 'count' value for that user, and
- the previous one-time password for that user.

When a user wishes to gain access to the host, the following procedure is followed.

1. The host decrements its stored counter for that user, and sends the value of this decremented counter, c say, to the user in conjunction with the seed value, D say.
2. On receipt of D and c, the user takes D and bit-wise exclusive-ors it with its 64-bit key to obtain a 64-bit value S. The one-way function shared by the user and host is then recursively applied to S a total of c times to obtain the 64-bit one-time password P. The value P is then sent back to the host.
3. On receipt of P, the host applies the one-way function to P once, and compares the result with its stored 'old password'. If the two values agree then the user is authenticated and the 'old' stored password is replaced with P. Otherwise the user is rejected.

Setting the system up will require the user to enter his/her pass-phrase into the host, where the initial count can be chosen, and the initial password computed and stored.

It is important to note that the host does not need to retain any secrets, since it only keeps the 'old' password, from which the new password cannot be derived.

3 Challenge-response protocols and predictable challenges

The scheme can be thought of as a special type of challenge-response scheme based on the use of a cryptographic check function, as specified in clause 5.1.2 of ISO/IEC 9798–4, [4]. In this protocol, entity B wishes to verify the identity of entity A. To achieve this the following two messages are exchanged.

1. B sends A a challenge R.
2. A sends to B a response equal to $f_K(R)$, where K is a shared secret key and f is a cryptographic check function.

We can make S/KEY fit this general model by equating:

- the challenge R with the concatenation of c and D, and
- the function $f_K(R)$ with $g^c(K \oplus D)$, where g is the one-way function used by S/KEY, K is the 64-bit user key, and $\oplus$ denotes bit-wise exclusive-or.

As a requirement for use of this 'unilateral authentication' protocol, ISO/IEC 9798–4 specifies that the challenge R should be unpredictable by any third party. However, in the S/KEY scheme this value is not unpredictable, since successive challenges from a particular host to a particular user consist of an identical seed, and count values differing by one. This leads to a possible problem, which we now describe.

First note that it has been observed by Gong, [2], that if a challenge is predictable in a challenge-response protocol (e.g. if a counter is used), then unless the challenge is cryptographically protected then a type of 'suppress-replay attack' (otherwise known as a 'preplay attack') becomes possible; the use and limitations of 'predictable nonces' has also been discussed in [1]. The general idea of the attack (in the context of the ISO/IEC 9798–4 protocol) is as follows.

User C wishes to impersonate A to B. Suppose also that C is able to predict the next challenge R to be chosen by B. C now impersonates B to A and asks A to authenticate itself; C chooses as a challenge the next challenge R which will be chosen by B when 'genuinely' authenticating A. When received, the response to R sent by A is memorised by C. At some future time C can now impersonate A to B, using this memorised response.

This immediately applies to the S/KEY system, since, by monitoring exchanges between a particular host and user, an interceptor will be able to predict the exact future values of the seed and count to be used. If the interceptor is able to impersonate the host to the user, the interceptor can obtain 'one time passwords' which are guaranteed to be valid at some point in the future.

In fact, things are actually even worse for the S/KEY system than we have so far implied. If the interceptor is ever able to impersonate the host to the user on one single occasion, then the system is effectively completely broken. This is because the impersonator of the host can supply the 'current' seed value together with a count of 1. Knowledge of the response to this challenge will readily enable all 'one time passwords' based on this particular seed value to be computed (given knowledge of the S/KEY one-way function).

4 Other possible schemes

Of course, it would be ideal to have a scheme having the main advantage of S/KEY (i.e. not requiring storage of user secrets by the host) combined with the ability to resist host-impersonation attacks of the type described. This seems difficult to provide without making additional assumptions about the capabilities of the host and/or the user. We briefly consider two possibilities, although neither of them are 'ideal' since they put additional requirements on the user and/or host.

First observe that, if the user can store the counter value c and the seed D for the host, then there is no longer a need for the host to send a challenge. The user simply decrements his/her stored counter c, and sends the host the three values: c, D and $g^c(K \oplus D)$ (which the host can verify as long as c is less than its stored value), c.f. clause 5.1.1 of ISO/IEC 9798–4, [4]. This scheme is still open to a limited form of the host-impersonation attack, although the most serious version, where the impersonator obtains the value of $g^1(K \oplus D)$, is no longer possible.

Second suppose that the host has a digital signature key pair, the private signature-generating part of which it can store securely, and that every user is capable of storing a copy of the host's public signature-verification key. The host can now use its private signature key to sign the challenge (consisting of c and D) and the user will not generate and send its response unless the signature is valid, c.f. [1]. This scheme now appears secure against host-impersonation, although it does require the host to store a secret, albeit a single secret and not secret information for each individual user.

5 Conclusions

The main advantage of the S/KEY scheme is that it avoids the need for the host to retain a user secret (although the current user 'count', the 'seed' and the 'old password' must be protected against unauthorised change). The main disadvantage is the possibility of the type of host-impersonation attack we have described, which a conventional challenge-response scheme based on unpredictable challenges would avoid, although such schemes do require the host to retain user secrets. Thus a careful choice should be made between S/KEY and other schemes, depending on whether the risk of the host storing user secrets or the threat of the host-impersonation attack is deemed greater.

References

[1] L. Chen, D. Gollmann, and C. Mitchell. Tailoring authentication protocols to match underlying mechanisms. In *Proceedings of the Australiasian Conference on Information Security and Privacy, June 1996*. Springer-Verlag, Berlin, 1996 (to appear).

[2] L. Gong. Variations on the themes of message freshness and replay. In *Proceedings: Computer Security Foundations Workshop VI*, pages 131–136. IEEE Computer Society Press, Los Alamitos, California, June 1993.

[3] N. Haller. *The S/KEY one-time password system.* Bellcore, February 1995. Internet RFC 1760.

[4] International Organization for Standardization, Genève, Switzerland. *ISO/IEC 9798–4, Information technology — Security techniques — Entity authentication — Part 4: Mechanisms using a cryptographic check function*, March 1995.

[5] L. Lamport. Password authentication with insecure communication. *Communications of the ACM*, **24**:770–772, 1981.

[6] R. Rivest. *The MD4 Message-Digest Algorithm.* MIT Laboratory for Computer Science and RSA Data Security Inc., April 1992. Internet RFC 1320.