Explicit Communication Revisited: Two New Attacks on Authentication Protocols

Martín Abadi
Systems Research Center
Digital Equipment Corporation
ma@pa.dec.com

March 9, 1997

Abstract

SSH and AKA are recent, practical protocols for secure connections over an otherwise unprotected network. This paper shows that, despite the use of public-key cryptography, SSH and AKA do not provide authentication as intended. The flaws of SSH and AKA can be viewed as the result of their disregarding a basic principle for the design of sound authentication protocols: the principle that messages should be explicit.

1 Introduction

SSH and AKA are two recent, practical protocols for secure connections over an otherwise unprotected network [Ylö96a, SSH96]; for example, they enable users to log into remote hosts. Both of the protocols rely on public-key cryptography for authentication (specifically, on RSA [RSA78]). In addition to authentication, each of the protocols has other significant goals, such as confidentiality.

This paper presents attacks on both of the protocols: it shows that SSH allows clients to be impersonated, while AKA allows servers to be impersonated. These attacks concern the SSH protocol as specified in the June 1996 Internet draft [Ylö96b], and the AKA protocol presented in the paper that introduces AKA [SSH96]. SSH is widely used; the version deployed is fortunately not that of the Internet draft.

The common weakness of SSH and AKA is the lack of explicitness in messages. Specifically, the signed statements of the protocols do not include some information that may be presumed obvious from context. The absence of this information opens the door to the attacks.

These attacks should not be very surprising—in particular, Needham and the author have described a few similar ones [AN96]. Beyond documenting the attacks on SSH and AKA, the purpose of this paper is to provide additional evidence in favor of the following basic principle:

> Every message should say what it means: the interpretation of the message should depend only on its content. It should be possible to write down a straightforward English sentence describing the content—though if there is a suitable formalism available that is good too.

This paper is a short sequel to the paper where the basic principle is introduced [AN96]. Additional principles for public-key protocols appear in the work of Anderson and Needham [AN95].

Notation This paper assumes some familiarity with cryptography, and in particular with the elementary properties of public-key cryptosystems and of one-way hash functions; this background material is covered by Schneier [Sch96], for example. The notation used is the same for both protocols (but differs from the notation of Ylönen and Safford et al. [Ylö96a, Ylö96b, SSH96]). The uniformity of notation is intended to highlight the similarities and differences between the protocols.

- A represents the client, B the server, and C the attacker. In messages, A stands for A's name.

- K_A is A's public key and K_A^{-1} is A's private key.

- K_{Bh} and K_{Bs} are two public keys for B, and K_{Bh}^{-1} and K_{Bs}^{-1} the corresponding private keys. In SSH terminology, K_{Bh} is a long-term host key, and K_{Bs} is a shorter-term server key. In AKA, K_{Bs} is in fact a one-time key.

- $\{X\}_K$ represents X encrypted under K; anyone who knows $\{X\}_K$ and the inverse of K can obtain X. For shared-key cryptosystems, such as DES [DES77], each key is its own inverse. For the RSA public-key cryptosystem, the public key and the private key in a key pair are inverses of one another. In particular, if K is a private key, then $\{X\}_K$ is the result of signing X with K.

Sometimes $\{X\}_K$ includes checkable redundancy that protects the integrity of X, but we do not display this redundancy in our notation.

- H is a one-way hash function and $\oplus$ is the exclusive-or operation.
- N_A and N_B are random nonces created by A and B, respectively. In AKA, but not in SSH, these are intended to be secret.

It is assumed that, given A and K_A, the server B can verify that K_A is the public key for A and that A is a legitimate client. The protocol descriptions below (and those of [Ylö96b, SSH96]) do not explicitly represent the certificates that may convey this information. Similarly, it is assumed that the client A can verify that K_{Bh} is the host public key for the intended principal; but A has no information about K_{Bs} beyond that obtained by running the protocols.

2 SSH

The SSH protocol has several options. Here we discuss only the case in which the client is authenticated using public-key cryptography. In this case, the protocol is:

Message 1 $A \rightarrow B:$ N_A
Message 2 $B \rightarrow A:$ N_B
Message 3 $B \rightarrow A:$ $K_{Bh},\ K_{Bs}$
Message 4 $A \rightarrow B:$ $\{\{H(\mathit{previous\ msgs.}),\ K\}_{K_{Bs}}\}_{K_{Bh}}$
Message 5 $A \rightarrow B:$ $\{A,\ K_A,\ \{H(A, N_A, N_B)\}_{K_A^{-1}}\}_{K'}$

The goal of these messages is to establish K as a session key for subsequent communication between A and B. In Messages 1 and 2, A and B exchange nonces. In Message 3, B provides its public keys. In Message 4, A sends K to B. This message is encrypted under B's public keys for confidentiality. It also includes a hash of the previous messages, as a tie to the protocol run. In Message 5, A provides its name and public key, and signs its name and the nonces. This message is protected by shared-key encryption with a key K' derived from K, N_A, and N_B.

A basic weakness in this protocol is that A's signed statement

$$\{H(A, N_A, N_B)\}_{K_A^{-1}}$$

does not contain enough information explicitly. The statement is intended to imply that K, N_A, and N_B can be used as the basis for a session between A and B; however, there is no trace of B or K in the statement. This weakness permits an attack.

In the attack, a server C with public keys K_{Ch} and K_{Cs} is able to impersonate A in a session with B. The attack is possible whenever A starts a session with C. When this happens, C immediately starts a session with B, making sure that the nonces are the same in both sessions; so C is able to use A's signed statement in the session with B.

Message 1	$A \rightarrow C:$	N_A
Message 1'	$C \rightarrow B:$	N_A
Message 2	$B \rightarrow C:$	N_B
Message 2'	$C \rightarrow A:$	N_B
Message 3	$B \rightarrow C:$	$K_{Bh},\, K_{Bs}$
Message 3'	$C \rightarrow A:$	$K_{Ch},\, K_{Cs}$
Message 4	$A \rightarrow C:$	$\{\{H(\mathit{previous\ msgs.}),\, K\}_{K_{Cs}}\}_{K_{Ch}}$
Message 4'	$C \rightarrow B:$	$\{\{H(\mathit{previous\ msgs.'}),\, K\}_{K_{Bs}}\}_{K_{Bh}}$
Message 5	$A \rightarrow C:$	$\{A,\, K_A,\, \{H(A, N_A, N_B)\}_{K_A^{-1}}\}_{K'}$
Message 5'	$C \rightarrow B:$	$\{A,\, K_A,\, \{H(A, N_A, N_B)\}_{K_A^{-1}}\}_{K'}$

At this point, B believes that it has started a session with A using K, while in fact C has K as well; A cannot notice any immediate errors.

The attack is thwarted if A signs additional fields, including some identifier for B (name, certificate, public keys) and some identifier for the session key (such as a hash of the key). Such additions have recently been made in response to the attack.

3 AKA

Like SSH, AKA has several variants, but all of them have similar messages. We discuss a typical one (the third one of [SSH96, p.181]):

Message 1	$A \rightarrow B:$	K_A
Message 2	$B \rightarrow A:$	$K_{Bh},\, K_{Bs}$
Message 3	$A \rightarrow B:$	$\{N_A,\, A\}_{K_{Bs}}$
Message 4	$B \rightarrow A:$	$\{N_B\}_{K_A}$
Message 5	$B \rightarrow A:$	$\{\{N_A\}_{K_{Bh}^{-1}}\}_{K_A}$
Message 6	$A \rightarrow B:$	$\{H(N_B)\}_{K_{Bs}}$

The goal of these messages is to establish $N_A \oplus N_B$ as a session key for subsequent communication between A and B. In Messages 1 and 2, A and B exchange public keys. In Messages 3 and 4, A and B exchange the nonces N_A and N_B; these are their contributions to the session key. In Message 5, B signs N_A. Finally, in Message 6, A sends an encrypted hash of N_B. One may reason that only A and B know the resulting session key, because of the use of public-key encryption; in particular, in Messages 3, 4, and 5, the secrecy of the nonces is protected.

In this protocol, it is B who signs, rather than A. But, mutatis mutandi, AKA has the same flaw as SSH, because B's signed statement does not contain enough information. Specifically, the statement proves possession of N_A, but does not mention N_B or the names of the principals. This weakness allows C to impersonate B in a session with A.

An attack is possible whenever A and B initiate a session, provided C can intercept and replace some messages. C need not even be a legitimate principal in the system.

Message 1	$A \rightarrow B :$	K_A
Message 2	$B \rightarrow \ldots :$	K_{Bh}, K_{Bs}
Message 2'	$C \rightarrow A :$	K_{Bh}, K_{Cs}
Message 3	$A \rightarrow \ldots :$	$\{N_A, A\}_{K_{Cs}}$
Message 3'	$C \rightarrow B :$	$\{N_A, A\}_{K_{Bs}}$
Message 4	$B \rightarrow \ldots :$	$\{N_B\}_{K_A}$
Message 4'	$C \rightarrow A :$	$\{N_C\}_{K_A}$
Message 5	$B \rightarrow A :$	$\{\{N_A\}_{K_{Bh}^{-1}}\}_{K_A}$
Message 6	$A \rightarrow \ldots :$	$\{H(N_C)\}_{K_{Cs}}$

C should intercept Message 2, and replace the server key K_{Bs} with one of its own, K_{Cs}. This change allows C to read A's nonce N_A, sent in Message 3; after receiving Message 3, C should produce a similar Message 3' for B (under K_{Bs}). C should intercept Message 4 as well, and create a replacement message with a nonce it knows, N_C; but C does not need to read N_B. Messages 5 and 6 are not affected, except for the fact that C may want to intercept Message 6. At the end of this exchange, C cannot continue the session with B; on the other hand, A believes that $N_A \oplus N_C$ is a session key for communication with B, when in fact C has $N_A \oplus N_C$.

This attack can be prevented in the usual manner: B should sign more information. As in SSH, such additions have recently been made.

4 Conclusion

The attacks on SSH and AKA exploit a frequent weakness: the absence of sufficient information in signed statements. Hopefully, these attacks will remind the designers of future protocols that it is generally prudent to include all relevant fields in signed statements, and that it is probably better to err on the side of including too much rather than too little.

As an extreme example, certain signed statements in the SSL v3.0 protocol include hashes of all previous messages [FKK96]. These hashes impede attacks analogous to those described in this paper, found by the author on drafts of SSL v3.0 and of an earlier version of SSL. The SSL approach seems rather effective. Although the SSL approach can be applied to many other protocols without much thought, it is always preferable to think carefully about the meaning of signed statements.

Acknowledgements

Robert Morris suggested looking at SSH, and checked the attack on SSH described here. David Wagner has found other attacks on AKA; the attack described here was discovered while reading his explanation of AKA.

References

[AN95] Ross Anderson and Roger Needham. Robustness principles for public key protocols. In *Proceedings of Crypto '95*, 1995.

[AN96] Martín Abadi and Roger Needham. Prudent engineering practice for cryptographic protocols. *IEEE Transactions on Software Engineering*, 22(1):6–15, January 1996.

[DES77] Data encryption standard. Fed. Inform. Processing Standards Pub. 46, National Bureau of Standards, Washington DC, January 1977.

[FKK96] Alan O. Freier, Philip Karlton, and Paul C. Kocher. The SSL protocol: Version 3.0. Available at `http://home.netscape.com/eng/ssl3/ssl-toc.html`, March 1996.

[RSA78] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.

[Sch96] B. Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C.* John Wiley & Sons, Inc., second edition, 1996.

[SSH96] David Safford, Douglas Schales, and David Hess. Texas A&M University anarchistic key authorization (AKA). In *Proceedings of the Sixth USENIX Security Symposium*, pages 179–185, July 1996.

[Ylö96a] Tatu Ylönen. SSH—Secure login connections over the Internet. In *Proceedings of the Sixth USENIX Security Symposium*, pages 37–42, July 1996.

[Ylö96b] Tatu Ylönen. SSH transport layer protocol. Internet draft, at `ftp://ds.internic.net/internet-drafts/draft-ietf-tls-ssh-00.txt`, June 13, 1996.