Cryptanalysis of the Yeh-Sun password-based authentication protocols

Chris J. Mitchell and Qiang Tang

Technical Report
RHUL–MA–2004–4
29 November 2004

Department of Mathematics
Royal Holloway, University of London
Egham, Surrey TW20 0EX, England
http://www.rhul.ac.uk/mathematics/techreports

Abstract

Two authentication protocols proposed by Yeh and Sun are analysed and shown to possess serious security defects.

1 Introduction

Yeh and Sun [2] have proposed two trusted third party (TTP)-aided client-server mutual authentication protocols. These protocols are designed for the case where a client shares a password with the TTP, the server shares a secret key with the TTP, and the TTP has an asymmetric encryption key pair for which a reliable copy of the public key is available to the client. The two protocols, known as KTAP and KAAP, cover the cases where either the TTP generates the key and distributes it to the client and server (KTAP), or the client and server agree on a key without any party choosing it (KAAP).

It is claimed in [2] that the protocols are secure against a range of possible attacks, although no formal evidence of security is offered. In this short paper we show that both protocols suffer from serious flaws, particularly KTAP.

2 The two protocols

We now briefly describe the two protocols. In these descriptions we use $\{x\}_P$ to denote the asymmetric encryption of the data string x using the public key P, and $[x]_K$ to denote the symmetric encryption of the data string x using the secret key K. We also use h to denote a one-way hash-function, i_X to denote an identifier for entity X, and $||$ to denote concatenation of data strings. We moreover suppose that all the parties have agreed on the cryptographic algorithms (and the details of the protocol) in advance. We denote the parties in the protocols by C (client), S (server) and T (TTP). Finally we suppose that the public encryption key of the TTP is P_T, that the TTP and the client share a secret password p_C, and that the TTP and the server share a secret key K_S.

Note that in both the protocol descriptions we implicitly assume that if any of the checks fail then the party concerned aborts the protocol and does not send any more messages.

2.1 The Key Transfer Authentication Protocol (KTAP)

The four messages of the KTAP protocol are as follows.

1. $C \rightarrow T \quad \{i_C||i_S||p_C||K_C\}_{P_T}$
2. $T \rightarrow S \quad [i_C||i_S||\,[i_C||K]_{K_C}\,||K]_{K_S}$
3. $S \rightarrow C \quad [i_C||K]_{K_C}\,||\,[i_S||r_S]_K$
4. $C \rightarrow S \quad r_S$

The processing associated with the messages is as follows. The client generates and stores a random secret key K_C and then uses the public key P_T of the TTP to generate message 1.

On receipt of message 1, the TTP first decrypts it (using the TTP's private decryption key), and verifies the correctness of the client password p_C (using i_C to identify the client). The TTP then chooses a random secret session key K (to be shared by C and S) and uses the server secret key K_S and the client-selected key K_C to generate message 2 (where i_S received in message 1 is used to identify the server).

When the server receives message 2, it is decrypted using the server secret key K_S. The server then checks that i_S is correctly included and generates a random 'nonce' r_S. The server then assembles message 3 by combining the 'inner' encrypted string from message 2 with a string encrypted using the key K, as recovered from message 2.

When the client receives message 3, the first part is decrypted using the key K_C (generated and stored at the beginning of the protocol). The client checks that i_C is correctly included, and then uses the recovered key K to decrypt the second part of the message. The correctness of i_S is then checked, at which point (if all the checks have passed) C has authenticated S. Finally, the client sends r_S to the server as message 4.

Receipt of message 4 enables the server to authenticate the client. At this point both client and server possess an authenticated shared secret key, K.

2.2 The Key Agreement Authentication Protocol (KAAP)

This protocol requires the client and server to have agreed on Diffie-Hellman domain parameters, namely a large prime modulus p and a base g, where g has multiplicative order q modulo p and q is a large prime. The four messages of the KAAP protocol are as follows.

1. $C \rightarrow T \quad \{i_C||i_S||p_C||r_C||g^x\}_{P_T}$
2. $T \rightarrow S \quad [i_C||g^x]_{K_S}$

3. $S \rightarrow C \quad g^y || [i_S || r_S]_K$
4. $C \rightarrow S \quad r_S$

The processing associated with the messages is as follows. The client generates and stores random values x and r_C, and then uses the public key P_T of the TTP to generate message 1 (where g^x is computed mod p).

On receipt of message 1, the TTP first decrypts it (using the TTP's private decryption key), and verifies the correctness of the client password p_C (using i_C to identify the client). The TTP then uses the server secret key K_S to generate message 2 (where i_S received in message 1 is used to identify the server).

When the server receives message 2, it is decrypted using the server secret key K_S. The server then generates a random nonce r_S and a random value y, and computes the secret session key $K = (g^x)^y \bmod p$. The server then assembles message 3 using the newly computed key K.

When the client receives message 3, the value g^y is used to obtain $K = (g^y)^x \bmod p$, which is then used to decrypt the remainder of the message. The client checks that i_S is correctly included, at which point the client has authenticated the server. The client concludes by sending r_S back to the server as message 4.

Receipt of message 4 enables the server to authenticate the client. At this point both client and server possess an authenticated shared secret key, K.

3 Cryptanalysis

We next describe a number of security issues with both protocols. The majority of these issues arise from the fact that in both cases the compromise of a transient or session key enables the protocol to be compromised. Such attacks have long been known — see, for example, Denning and Sacco [1].

Before discussing these issues in detail note that it is important for both protocols that the symmetric encryption technique used should be sufficiently robust to resist attacks based on known plaintext-ciphertext pairs. This is because message 2 in both protocols provides a dishonest client C with a known plaintext-ciphertext pair for the key K_S.

Note also that both protocols rely for their security on the secrecy of a transient key during execution of the protocol. If, K_C (in KTAP) or g^x (in KAAP) becomes known to an eavesdropper E, then E can use this knowledge to impersonate S to C in this instance of the protocol. This is a typical and uncontroversial assumption for an authentication protocol; however, we show below that if such a transient secret is ever disclosed, even long after execution

of the protocol, then serious attacks are possible.

3.1 Analysis of KTAP

The KTAP protocol possesses the following serious weaknesses.

1. Suppose a fraudulent third party, E say, who has intercepted message 2 of the protocol, learns the session key K sent in that message. (This may occur as a result of the use of the key K in subsequent communications between C and S). Entity E can now impersonate C to S any number of times. This is achieved by E forwarding the intercepted message 2 to S. The server S then responds with message 3, containing a new random nonce r'_S encrypted using the 'old' key K. E intercepts this message and can decrypt the second part to obtain r'_S. This can then be used to impersonate C to S.

2. Suppose E has intercepted messages 1 and 2 of the protocol and has, by some means, learnt the key K_C sent in message 1 and used to encrypt part of message 2. How this might be achieved is outside the scope of this paper; however, note that message 3 will provide an interceptor with a matching pair of ciphertext and (partial) known plaintext, where the ciphertext is encrypted using K_C. This could provide the basis for an exhaustive search for K_C, especially if K_C is not well chosen (this point is considered in more detail below).

   This situation gives rise to two separate vulnerabilities.

   (a) E can use this value of K_C to impersonate C to S. This arises trivially since E can decrypt the copy of message 2 encrypted using K_C, and thereby recover K. The attack described above can then be employed.

   (b) If the password p_C is poorly chosen, then knowledge of K_C will enable p_C to be discovered from message 1 using an exhaustive search (since all other data included in this message is known to E). Compromise of the password p_C will enable the attacker to impersonate C freely.

Over and above these weaknesses, we note two further anomalies in the protocol as described in [2].

- Firstly observe that Yeh and Sun appear to be assuming that symmetric and asymmetric encryption provide integrity protection for an encrypted message. This requirement is, however, never explicitly stated.

Moreover, although modern asymmetric encryption schemes provide the non-malleability property, which will guarantee this integrity protection, the same is not true for any widely used symmetric encryption schemes. That is, implementing the protocol as specified (without providing the necessary integrity protection) will result in an insecure scheme.

- Secondly note that the key labelled K_C in the description of KTAP is described as a random value in [2]. However, the protocol requires it to be used as a key, and hence we describe it as a key in this paper. This means that it is a fundamental protocol requirement that C must be capable of generating sound keys. If not, i.e. if K_C is not chosen well, then deducing K_C and enabling the second attack above may be possible. Also, if A is capable of generating sound keys, then there seems no reason not to employ the KAAP protocol (that is, there seem to be no advantages from the use of KTAP).

3.2 Analysis of KAAP

KAAP possesses the following potentially significant security vulnerability. Suppose E has intercepted message 2 from one instance of the protocol, and has (by some means) determined the values of both x and g^x used in this protocol instance. Then E can use these values to impersonate C to S any number of times. This is achieved by E forwarding the intercepted message 2 to S. The server S then responds with message 3, containing a new public transient value $g^{y'}$ and a new random nonce r'_S encrypted using the key $K' = g^{xy'}$. E intercepts this message and can combine the known value of x with $g^{y'}$ to deduce K, which can then be used to obtain the cleartext nonce r'_S. This can then be used to impersonate C to S.

It is, however, important to note that it may be difficult for an attacker to obtain both x and g^x, since g^x is only sent in encrypted form, and x is never transmitted; moreover, as long as p and q are sufficiently large, determining x from g^x is computationally infeasible. Thus this vulnerability is of somewhat less significance than the vulnerabilities in KTAP.

Over and above this weakness, we note that, as for KTAP, the authors appear to be assuming that the symmetric and asymmetric encryption schemes used provide data integrity protection, although this is not an explicit requirement.

4 Conclusions

Serious security issues have been identified in both the protocols specified in [2]. As a result, use of these protocols cannot be recommended. More generally, protocols not possessing any solid evidence of security should only be used with great care.

References

[1] D. E. Denning and G. M. Sacco. Timestamps in key distribution protocols. *Communications of the ACM*, **24**:533–536, 1981.

[2] H.-T. Yeh and H.-M. Sun. Password-based user authentication and key distribution protocols for client-server applications. *The Journal of Systems and Software*, 72:97–103, 2004.