Weaknesses in a leakage-resilient authenticated key transport protocol

Qiang Tang and Chris J. Mitchell
Information Security Group
Royal Holloway, University of London
Egham, Surrey TW20 0EX, UK
{qiang.tang, c.mitchell}@rhul.ac.uk

3rd September 2005

Abstract

In this paper we demonstrate the existence of a number of weaknesses in a leakage-resilient authenticated key transport (RSA-AKE) protocol due to Shin, Kobara and Imai.

1 Introduction

Recently, Shin, Kobara and Imai proposed a leakage-resilient authenticated key transport protocol [2] (referred to as the RSA-AKE protocol) to be used in an client-server environment, where

1. A client C, who can only remember a password, wishes to communicate with several servers.
2. C has insecure devices with very restricted computing power and built-in memory capacity. The servers have significant computing power, but they might be compromised.
3. Neither a PKI (Public Key Infrastructure) nor a TRM (Tamper-Resistant Module) is available.

Shin, Kobara and Imai claim that the RSA-AKE protocol is provably secure in the random oracle model [2]. However, we show that, despite this, the RSA-AKE protocol suffers from potential security problems in the intended environment of use.

The rest of this paper is organised as follows. In Section 2 we review the the RSA-AKE protocol. In section 3 we demonstrate weaknesses in the RSA-AKE protocol. In the final section, we conclude this paper.

2 Review of the RSA-AKE protocol

Suppose a client C shares password π with the server S. Without loss of generality, we suppose that C possesses identity ID_C, and S possesses identity ID_S. Additionally, we suppose that f is a full-domain hash function [3], and h_i $(1 \leq i \leq 4) : \{0,1\}^* \rightarrow \{0,1\}^k$ are four different hash functions, where k is a security parameter. Throughout this paper, $f(x_1, \cdots, x_n)$ and $h_i(x_1, \cdots, x_n)$ $(1 \leq i \leq 4)$ represent computing the hash value on the concatenation of messages x_j, $1 \leq j \leq n$.

At the initialisation stage, S generates its RSA public/private key pair (e, N) and (d, N), and sends (e, N) to C. C registers a password verifier $p_1 = (\alpha_1 + \pi) \bmod N$ at S, where α_1 is randomly selected from Z_N. C stores α_1 and (e, N) on some insecure device such a PDA, which is not necessarily securely protected and may leak the stored information. S stores p_1 and (d, N) in its database, which is also not necessarily securely protected and may leak the stored information (both p_j and (d, N)). Note that the values of p_j, $j = 1, 2, \cdots$ are defined recursively — see step 4 below. Finally, C and S both also store a counter j, initially set to 1.

In the j-th $(j \geq 1)$ execution of the RSA-AKE protocol, C and S perform as follows.

1. C first computes the password verifier $p_j = \alpha_j + \pi \bmod N$. Note that the values of α_j, $j = 1, 2, \cdots$ are defined recursively — see step 3 below. Then C chooses a random $x \in Z_N^*$ and computes $W = f(j, p_j)$, $y = x^e \bmod N$, and $z = y \cdot W \bmod N$. Finally, C sends ID_C, j, z to S.

2. S first checks whether j is the correct counter value. If the check succeeds, S computes $y' = z \cdot W^{-1} \bmod N$, $x' = (y')^d \bmod N$, and $V_S = h_1(ID_C, ID_S, j, z, p_j, x')$, and then sends ID_S, V_S to C. Otherwise, S terminates the protocol execution.

3. After receiving S and V_S, C first checks whether the following equation is valid:
$$V_S = h_1(ID_C, ID_S, j, z, p_j, x)$$
If the check succeeds, C computes and sends V_C to S, where
$$V_C = h_2(ID_C, ID_S, j, z, p_j, x)$$
Otherwise, C terminates the protocol execution.

C computes the session key as $SK_j = h_3(ID_C, ID_S, j, z, p_j, x)$, and replaces the stored data α_j with α_{j+1}:

$$\alpha_{j+1} = \alpha_j + h_4(ID_C, ID_S, j, z, p_j, x) \bmod N$$

C sets the counter value to $j + 1$.

4. After receiving V_C, S first checks whether the following equation is valid:

$$V_C = h_2(ID_C, ID_S, j, z, p_j, x')$$

If the check succeeds, S computes the session key as

$$SK_j = h_3(ID_C, ID_S, j, z, p_j, x'),$$

and replaces the password verifier p_j with p_{j+1}:

$$p_{j+1} = p_j + h_4(ID_C, ID_S, j, z, p_j, x') \bmod N$$

S sets the counter value to $j+1$. Otherwise, S terminates the protocol execution as a failure.

3 Possible weaknesses in the RSA-AKE protocol

Shin, Kobara and Imai [2] claim that the RSA-AKE protocol is provably secure in the random oracle model under the notion of LR-AKE security, where an adversary is given the client's stored secret and the server's RSA private key.

However, we nevertheless show that the RSA-AKE protocol suffers from certain potential security problems. It is, however, important to note that some of these vulnerabilities are outside the scope of the security model used in [2].

1. We show that, given the client's stored secret and the server's RSA private key, an adversary can mount an offline dictionary attack.

Lemma 3.1. *Given the client's stored secret and the server's RSA private key, an adversary can mount an offline dictionary attack.*

Proof. Without loss of generality, we suppose that the adversary is given α_j and (d, N) just before the j-th run of the RSA-EKE protocol.

During the j-th run of the RSA-EKE protocol, the adversary collects j, z, and V_S. The adversary then performs the following steps:

(a) The adversary guesses a possible password π^*, computes $p_j^* = \pi^* + \alpha_j \bmod N$, $x^* = ((f(j, p_j^*))^{-1} \cdot z)^d \bmod N$, and tests whether $V_S = h_1(ID_C, ID_S, j, z, p_j^*, x^*)$ holds.

(b) If the test succeeds, the adversary confirms that $\pi^* = \pi$ and stops; otherwise go to the first step.

It is straightforward to verify that the above attack will succeed in identifying the correct password with very high probability. □

2. Observe that p_j is the only secret used for authentication in the j-th run of the RSA-AKE protocol. So, if the attacker has compromised S and obtained p_j, then he can successfully impersonate C to S in the subsequent protocol executions without the need to have access to π. If this occurs, the legitimate client will no longer be able to authenticate himself, because the password verifier held by S will change. However, if the legitimate client authenticates himself before the attacker uses the stolen p_j, then the attacker cannot launch the above attack because the stolen password verifier p_j will no longer be valid.

   This attack means that leakage of p_j from S may enable an attacker to mount an impersonation attack. Hence the RSA-AKE protocol does not appear to be suitable for use in environments where the server is not securely protected.

3. Shin, Kobara and Imai point out that measures should be adopted to restrict an attacker's ability to replace the RSA public key (e, N) on the client's device; otherwise they show that an eth-residue attack can be mounted. They also propose a means to thwart the eth-residue attack if the attacker does succeed in replacing the RSA public key (e, N) with (e', N'). However, we show below that, in some extreme circumstances, more serious vulnerabilities exist in this case.

   Suppose, for example, that the attacker has obtained α_j and replaced the RSA public key (e, N) with (e', N'), where $e' = \phi(N')$, just before the j-th execution of the RSA-AKE protocol. In this case, the attacker can exhaustively search for the password using the intercepted message z. This is because $x^{e'} \bmod N' = 1$ for every x (since $e' = \phi(N')$), and hence $z = f(j, \pi + \alpha_j) \bmod N'$. That is, the only unknown value used to compute z is π. The measures proposed in [2] do not eliminate this vulnerability.

4. Shin, Kobara and Imai suggest that C can use the same password π with a number of servers. However, it is potentially dangerous to do this. Suppose the client shares the same password with m servers S_i $(1 \leq i \leq m)$. Then an attacker can successfully guess the password with a probability p by mounting n/m dictionary attacks in parallel

at each server S_i ($1 \leq i \leq m$), while he would need to mount n dictionary attacks against one specific server in order to achieve the same goal. This attack means that the client might need to change his password much more frequently (if m is very large) in order to prevent undetected dictionary attacks.

This attack is of particular concern in environments where m is large and the client chooses the password from a small password set, e.g. based on personal preferences.

4 Conclusions

In this paper we have demonstrated certain weaknesses in a leakage-resilient authenticated key transport protocol.

References

[1] J. Camenisch, U. M. Maurer, and M. Stadler. Digital payment systems with passive anonymity-revoking trustees. In E. Bertino, H. Kurth, G. Martella, and E. Montolivo, editors, *ESORICS '96: Proceedings of the 4th European Symposium on Research in Computer Security*, pages 33–43. Springer-Verlag, 1996.

[2] S. Shin, K. Kobara, and H. Imai. Efficient and leakage-resilient authenticated key transport protocol based on RSA. In J. Ioannidis, A. D. Keromytis, and M. Yung, editors, *Applied Cryptography and Network Security, Third International Conference, ACNS 2005, New York, NY, USA*, volume 3531 of *Lecture Notes in Computer Science*, pages 269–284. Springer-Verlag, 2005.

[3] D. Stinson. *Cryptography Theory and Practice*. CRC Press, Inc., second edition, 2002.