Notes on weaknesses in a leakage-resilient authenticated key transport protocol

SeongHan Shin, Kazukuni Kobara, and Hideki Imai

Institute of Industrial Science, The University of Tokyo,
4-6-1 Komaba, Meguro-ku, Tokyo 153-8505, Japan
shinsh@imailab.iis.u-tokyo.ac.jp, {kobara,imai}@iis.u-tokyo.ac.jp
http://imailab-www.iis.u-tokyo.ac.jp/imailab.html
June 23, 2005

Abstract. Tang et al., [1] have showed weaknesses in a leakage-resilient authenticated key transport (so-called RSA-AKE) protocol [2] and then proposed an enhanced protocol. The objective of this paper is two-fold. First, we clarify some ambiguities that may cause misunderstandings on the RSA-AKE protocol by [1]. Second, we show that Tang's protocol is *insecure* against a weaker adversary who gets the client's stored secret: the adversary can retrieve the password with off-line dictionary attacks after eavesdropping only one message. Note that the RSA-AKE protocol is secure against an adversary who gets the client's stored secret and the server's RSA private key.

1 Tang's observations on the RSA-AKE protocol

Here we clarify some ambiguities that may cause misunderstandings on the RSA-AKE protocol by [1]. Please, refer to Section 2 of [1] or the conference paper [2] for the RSA-AKE protocol. For an easier discussion, we paste Section 3 of [1] here and explain about some ambiguities.

Here are three observations Tang claimed in [1].

1. Observe that p_{ij} is the only secret used for authentication in the j-th run of the RSA-AKE protocol. So, if the attacker has compromised $\mathcal{S}_i$ and obtained p_{ij}, then he can successfully impersonate $\mathcal{C}$ to $\mathcal{S}_i$ in the subsequent protocol executions without the need to have access to pw. If this occurs, the legitimate client will no longer be able to authenticate himself, because the password verifier held by $\mathcal{S}_i$ will change. However, if the legitimate client authenticates himself before the attacker uses the stolen p_{ij}, then the attacker cannot launch the above attack because the stolen password verifier p_{ij} will no longer be valid.
   This attack means that leakage of p_{ij} from $\mathcal{S}_i$ may enable an attacker to mount an impersonate attack. Hence the RSA-AKE protocol does not appear to be suitable for use in environments where the server is not securely protected.

2. Suppose, for example, that the attacker has obtained α_{ij} and replaced the RSA public key (e, N) with (e', N'), where $e' = \varphi(N')$, 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 $e' = \varphi(N')$), and hence $z = f(j, pw + \alpha_{ij}) \bmod N'$. That is, the only unknown used to compute z is pw.
3. Shin, Kobara and Imai suggest that $\mathcal{C}$ can use the same password pw with a number of servers $\mathcal{S}_i$ $(i \geq 1)$. However, it is potentially dangerous to do this. Suppose the client shares the same password with m servers $\mathcal{S}_i$ $(1 \leq i \leq m)$. Then an attacker can successfully guess the password with a probability p by mounting n/m dictionary attacks in parallel at each server $\mathcal{S}_i$ $(1 \leq i \leq m)$, 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.

Here are our explanations in order to clarify some ambiguities.

As for the first observation: We already mentioned that the RSA-AKE protocol is insecure against the leakage of stored secret p_{ij} (see Table 1 and Section 5.1 of [2]). As Tang said, this attack is only possible until the client successfully run the RSA-AKE protocol with the server.
A more important point to be noted is that the RSA-AKE protocol is designed to minimize the effect caused by the leakage of the stored secrets. That means, the leakage of p_{ij} from one server doesn't affect the other runs of the protocol between the client and the remaining servers where the client, remembering only one password, is communicating with many servers! For example, if an adversary $\mathcal{A}$ gets the verification data p_{ij} from server $\mathcal{S}_i$ the adversary, of course, impersonates the client to the server $\mathcal{S}_i$. However, the adversary $\mathcal{A}$ cannot impersonate the client to the remaining servers S_k $(1 \leq i, k \leq m$ and $i \neq k)$ where m is the number of servers the client is communicating with. And the adversary $\mathcal{A}$ cannot impersonate the remaining servers S_k to the client as well.

As for the second observation: A simple check can prevent the attack that the client doesn't send z if the encryption of x (i.e., x^e) is 1. We omitted this check because it is a common consensus for computing a value in a finite field (we think).
The RSA-AKE protocol is especially designed for the client's devices with very-restricted computing power so that e should be a small prime (e.g., 3 or $2^{16} + 1$) (see Introduction and Section 3.1 of [2]). Checking whether e is 3 or $2^{16} + 1$ is not difficult for the client.

As for the third observation: Unfortunately, Tang et al., seems missing the following points with respect to the RSA-AKE protocol. (For more details, see Figure 1 and Section 3.1 of [2].)
First, in order to do parallel on-line dictionary attacks to servers $\mathcal{S}_i$ ($1 \leq i \leq m$), an adversary first should obtain the client's stored secret α_{ij} and all of the servers' RSA private keys (d_i, N_i) ($1 \leq i \leq m$). Second, in order to continue parallel on-line dictionary attacks, the adversary should obtain the client's stored secret α_{ij} successively. If the adversary $\mathcal{A}$ does not have $\alpha_{i(j+1)}$ in the $(j+1)$-th run of the RSA-AKE protocol, $\mathcal{A}$ cannot do on-line attacks any more since the secret α_{ij} is useless at the end of the protocol. Think of a client who accesses a web-mail server every 6 hours using the RSA-AKE protocol, the adversary $\mathcal{A}$ can do on-line attacks for the period with α_{ij} and (d, N) and then $\mathcal{A}$ should steal the next $\alpha_{i(j+1)}$ to continue on-line attacks for the next same period.

2 The insecurity of Tang's protocol

We briefly show the initialization and the actual execution of Tang's protocol.

In the initialization stage, server $\mathcal{S}_i$ generates its RSA key pair $((e, N), (d, N))$ and two large primes p and q, where $p = 2q+1$ and $q > N$. Also, server $\mathcal{S}_i$ generates two generators g_1 and g_2 of a multiplicative subgroup of order q in Z_p^* where $g_2 = g_1^{x*}$ and x^* is randomly selected form Z_q^*. Finally, $\mathcal{S}_i$ sends $(e, N), p, g_1$ and g_2 to client $\mathcal{C}$ who remembers many passwords pw_i according to servers $\mathcal{S}_i$ ($1 \leq i$) where $pw_i \neq pw_j$ if $i \neq j$. The client $\mathcal{C}$ registers a password verifier $g_2^{p_{i1}} \bmod p$ at server $\mathcal{S}_i$ where $p_{i1} = \alpha_{i1} + pw \bmod q$ and α_{i1} is randomly selected from Z_q. At the end of the initialization stage, $\mathcal{C}$ stores $p, g_1, g_2, \alpha_{i1}, (e, N)$ and a counter j (initially set to 1) on some insecure devices that are not necessarily securely protected and may leak the stored information. $\mathcal{S}_i$ stores $p, g_1, g_2, g_2^{p_{i1}}, (d, N)$ and a counter j (initially set to 1) in its database, which may leak the stored information as well.

In the j-th ($j \geq 1$) execution of Tang's protocol, $\mathcal{C}$ and $\mathcal{S}_i$ perform as follows.

1. $\mathcal{C}$ first computes $p_{ij} = \alpha_{ij} + pw \bmod q, t_1 = g_1^{p_{ij}} \bmod p$. Then $\mathcal{C}$ chooses a random $x \in \mathbb{Z}_N^\star$ and computes $W = f(j, t_2), y = x^e \bmod N$, and $z = y \cdot W \bmod N$. Finally, $\mathcal{C}$ sends ID_C, j, z, t_1 to $\mathcal{S}_i$.

We omit the remaining procedures of the actual execution since we need the first message in order to show the insecurity of Tang's protocol (see Section 4.1 of [1] for the whole description).

The insecurity of Tang's protocol: Suppose an adversary who gets the client's stored secret $(p, g_1, g_2, \alpha_{ij}, (e, N), j)$ before the j-th execution. Only eavesdropping the first message (i.e., t_1), the adversary can compute the password pw with off-line dictionary attacks: $t_1 = g_1^{\alpha_{ij}+pw} \bmod p$. Then the adversary can freely impersonate the client to server $\mathcal{S}_i$ (the exact same attack is possible to the remaining servers).

Note that the RSA-AKE protocol is secure against an adversary who gets the client's stored secret and the server's RSA private key.

Acknowledgements

The main motivation of this paper is the work that was proposed by Tang et. al., in [1], and we show here our comments on Tang's work. In fact, although Tang received our comments (on Section 3 of [1]) before he submitted his work to ePrint server, he wrote in his acknowledgements as if we supported his work while the truth is different.

References

1. Q. Tang and Chris J. Mitchell, "Weaknesses in a leakage-resilient authenticated key transport protocol", `http://eprint.iacr.org/2005/173`, June 10, 2005.
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, *The Third International Conference on Applied Cryptography and Network Security (ACNS2005)*, LNCS 3531, pages 269-284. Springer-Verlag, USA, 2005.