Attacks on Two-way RSA Key Agreement Protocol

Mariskha Tri Adithia
mariskha@unpar.ac.id

January 28, 2012

Abstract

In this paper we discuss a small subgroup attack, a triangle attack, and an unknown key-share attack, and their application on the two-way RSA key agreement protocol. We explain the assumptions for exploring these attacks and the practical circumstances where the assumptions hold. Finally, the countermeasures to avoid these attacks will also be discussed.

1 Introduction

A key agreement protocol is a protocol which enables two or more parties to establish a shared secret key. In the shared key establishment process, all involved parties contribute information that are combined to obtain the shared secret key. By using this protocol, the parties can share keys freely and securely over an insecure medium. In general, a public-key cryptography is used as a building block for implementing a key agreement protocol. The first key agreement protocol based on a public key cryptography is a Diffie-Hellman key agreement which uses an exponential technique in generating the shared secret key.

In this paper we discuss the two-way RSA key agreement protocol. This protocol is based on a factoring technique. This protocol makes use of the RSA public key cryptosystem that was invented by R. L. Rivest, A. Shamir and L. Adleman in 1978; the security provided is based on on factoring large numbers. Nowadays, the protocol is used in electronic commerce protocols, and is believed to be secure given sufficiently long keys.

There are two types of attacks which are available against this protocol, namely passive attacks and active attacks. In this paper, we only discuss the active attacks, especially a small subgroup attack, a triangle attack, and an unknown key-share attack. Later in this paper, we show that the triangle attack and the unknown key-share attack can successfully be applied on the two-way RSA key agreement protocol while this is not the case for the small subgroup attack. The main ideas of the unknown key-share attack, the triangle attack, and the small subgroup attack, and also the application on two-way RSA key agreement protocol are discussed later.

This paper is organized as follows: in Section 2, we discuss the RSA cryptosytem and its application on the two-way RSA key agreement protocol. Section 3 provides detailed information about the unknown key-share attack, the triangle attack, and the small subgroup attack, and how to apply the attacks on two-way RSA key agreement protocol. The countermeasures taken to avoid the attack are discussed in Section 4. Lastly, Section 5 gives the conclusions and remarks of the entire paper.

2 Two-way RSA key agreement protocol

First, we discuss about the RSA cryptosystem before we look through the two-way RSA key agreement protocol. The RSA cryptosystem is used for privacy or signature. In our case, we only consider the RSA cryptosystem for privacy. Furthermore, the RSA cryptosystem makes use of the following facts:

- Computing exponentiation modulo a composite number n is simple. For example, it is simple to compute c from $c \equiv m^e (\text{mod } n)$ for given m and e.
- Taking roots modulo a large, composite number n is difficult. For example, it is difficult to compute m from $c \equiv m^e (\text{mod } n)$.
- If the prime factorization of n is known, the problem of taking roots modulo n is feasible.

We continue with setting up the system. Suppose that the parties who follows the protocol are denoted with user U . Based on [5] there are three main steps required as follows:

- Step 1 - computing the modulus n_U
  Each user U chooses two large prime numbers, say p_U and q_U. Next let $n_U = p_U q_U$.
- Step 2 - computing exponents e_U and d_U
  User U chooses an integer e_U with $1 < e_U < \varphi(n_U)$ and $\gcd(e_U, \varphi(n_U)) = 1$. $\varphi(n_U)$ is an Euler's Totient Function, namely the number of integers between 1 and n_U which are coprime with n_U. Here $\varphi(n_U) = (p_U - 1)(q_U - 1)$. Next, user U can compute d_U which satisfy $e_U d_U \equiv 1 (\text{mod } \varphi(n_U))$.
- Step 3 - making public e_U and n_U
  Each user U makes e_U and n_U public while d_U stays secret.

The protocol works as follows: say that Alice wants to send a secret message m to Bob with $1 < m < n_B$. She encrypt her message by using Bob's public exponent by computing $c \equiv m^{e_B} (\text{mod } n_B)$ and sends c to Bob. Then Bob can decrypt the message by raising c to the power of his secret exponent d_B. Thus he gets

$$c^{d_B} \equiv (m^{e_B})^{d_B} \equiv m^{e_B d_B} \equiv m^{1+l(\varphi(n_B))} \equiv m(m^{\varphi(n_B)})^l$$

By applying the following Euler's theorem:

Theorem 2.1. *Let a and n integers, and a coprime to n. Then $a^{\varphi(n)} \equiv 1 (mod\ n)$*

to the last equation, the original message sent by Alice, $m(\text{mod } n_B)$, can be obtained.

From the explanation above, we can summarize the RSA system as follows:

public	e_U and n_U
secret	d_U
property	$e_U d_U \equiv 1(\text{mod } \varphi(n_U))$
message to Bob	m with $0 < m < n_B$
encryption by Alice	$c \equiv m^{e_B}(\text{mod } n_B)$
decryption by Bob	$c^{d_B} \equiv m(\text{mod } n_B)$

A two-way RSA key agreement protocol is derived from the RSA cryptosystem. Suppose that user A and B are the parties who want to generate a secret key by applying the protocol. Based on [3], the protocol works as follows:

1. User A chooses two large numbers p_A and q_A and let $n_A = p_A q_A$.
2. User B chooses two large numbers p_B and q_B and let $n_B = p_B q_B$.
3. User A chooses an integer e_A with $1 < e_A < \varphi(n_A)$ and $\gcd(e_A, \varphi(n_A)) = 1$ and compute d_A which satisfy $e_A d_A \equiv 1(\text{mod } \varphi(n_A))$.
4. User B chooses an integer e_B with $1 < e_B < \varphi(n_B)$ and $\gcd(e_B, \varphi(n_B)) = 1$ and compute d_B which satisfy $e_B d_B \equiv 1(\text{mod } \varphi(n_B))$.
5. Both A and B make e_A, n_A and e_B, n_B respectively public.
6. A chooses a random number m_A, $0 < m_A < n_B$ and B chooses a random number m_B, $0 < m_B < n_A$.
7. A computes $r_A \equiv m_A^{e_B}(\text{mod } n_B)$ and sends it to B.
8. B computes $r_B \equiv m_B^{e_A}(\text{mod } n_A)$ and sends it to A.
9. A computes $m_B \equiv r_B^{d_A}(\text{mod } n_A)$.
10. B computes $m_A \equiv r_A^{d_B}(\text{mod } n_B)$.
11. Both users compute the secret key $SK = m_A + m_B$.

The protocol is also shown in Figure 1 below.

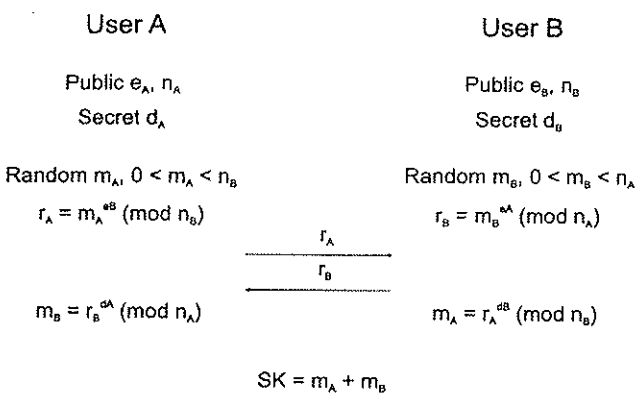

Figure 1: A two-way RSA key agreement protocol

All parties involved in the key agreement protocol should somehow inform each other of their identities. This is usually done by exchanging identities and may be done in advance, which means before the key agreement protocol begins. But the exchanges could be of the first key agreement messages, for example in our case A can send his identity a together with r_A to B and B can do the same.

3 The attacks

This section introduces the main idea of a triangle attack, an unknown key-share attack, and a small subgroup attack. How the attacks are applied on the two-way RSA key agreement protocol is also explained.

3.1 Triangle attack

In triangle attack, the adversary follows the protocol to generate the secret key by communicating with honest parties. Suppose that there are three parties involved in the protocol, namely A, B, and C. Say that the honest parties are B and C, and so that A be the adversary. Then, he eavesdrops the conversations between B and C. By using the known secret exponent and the result from eavesdropping, he can compute the share secret key between B and C.

The triangle attack on the two-way RSA key agreement protocol works as follows:

1. A gets r_B and r_C by eavesdropping conversation between B and C. Let r_B and r_C be the calls exchanged and let $SK_{BC} = r_B^{d_C} + r_C^{d_B}$ be the shared secret key. A wants to compute this key.

2. A does a conversation with B and exchanges key by using r_C as his key and let $SK_{BA} = r_B'^{d_A} + r_C^{d_B}$ be the shared secret key of B.

3. A does a conversation with C and exchanges key by using r_B as his key and let $SK_{CA} = r_C'^{d_A} + r_B^{d_C}$ be the shared secret key of C.

4. Suppose that SK_{BA} and SK_{CA} are revealed to A. Thus A can compute $r_C^{d_B} = SK_{BA} - r_B' d_A$ and $r_B^{d_c} = SK_{CA} - r_C'^{d_A}$. Hence the shared secret key of B and C can be obtained $SK_{BC} = r_B^{d_C} + r_C^{d_B}$.

Note that in Step 2 and Step 3, A cannot compute SK_{BA} and SK_{BC} himself since he does not know d_C and d_B, but he need to retrieve them in some way. One way to do that is by determining d_C and d_B with brute force method based on the knowledge he has. However, this is not possible since finding d_C and d_B means breaking the RSA cryptosystem which is extremely unlikely. Another way is by breaking the place where the shared secret keys are stored. This method is possible to do since usually a shared secret key used to encrypt a file or message is stored in a place which is supposed to be safe.

The figures below describe the attack more clear.

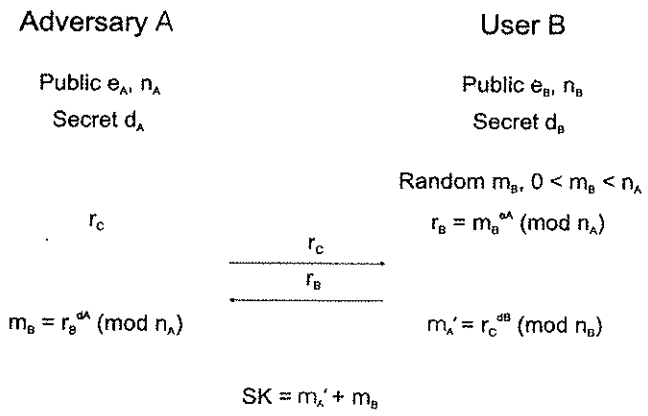

Figure 2: A triangle attack: conversations between adversary A and user B

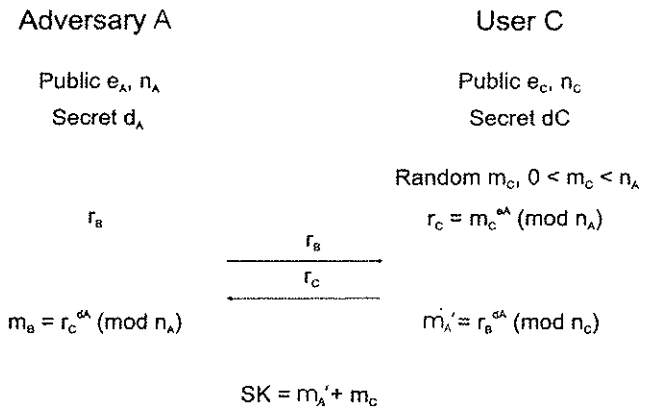

Figure 3: A triangle attack: conversations between adversary A and user C

3.2 Unknown key-share attack

In an unknown key-share attack, an adversary which is a man in the middle, makes the honest parties confuse about with whom they are communicating with. Here, at least one of the honest party does not know that he communicates with the other. For example, in our case, say that Alice and Bob follow a protocol to generate a secret key. There is Eve, which is an adversary, makes Bob believe that he shares a secret with Eve, while he actually shares it with Alice.

The adversary does not learn anything about the secret key. Thus, she cannot decrypt or modify the message exchanged by the honest parties. However, the adversary may take an advantage from the parties' false assumption about with whom they communicate. For example, say that Alice sends a message with Bob with her secret key. Since she uses the secret key which is shared with Bob, then she assumes that Bob knows that the message is from her. But when Eve does the unknown key-share attack, Bob will think that the message is from Eve.

The following is the classical example of an unknown key-share attack application in daily life (from [2]): Bob is a manager and sends Eve an order (like the apt, "You're fired!"), integrity-protected with the shared secret key. Believing that the key is shared with Bob, Alice assumes the order is for her and follows it.

Let Eve be the adversary who makes public e_E and n_E and has a secret d_E. The unknown key-share attack on the two-way RSA key agreement protocol works as follows:

1. When A sends r_A together with his identity a to B, Eve replaces A's identity by hers, e, and sends it to B.

2. B receives the message and thinks that it is from Eve, then he calculates his contribution under Eve's public key; $r_B = m_B^{e_E} (\text{mod } n_E)$ and sends it to Eve.

3. Eve decrypts r_B by using her secret key thus she gets m_B. Next she she calculates a new $r_B = m_B^{e_A} (\text{mod } n_A)$ and sends it to A. She can do this since e_A and n_A are public.

4. A calculates $m_B = r_B^{d_A} (\text{mod } n_A)$ and B calculates $m_A = r_A^{d_B} (\text{mod } n_B)$.

5. A and B calculate a secret key which is $SK = m_A + m_B$.

Here, Eve does not obtain the shared secret key, but makes A think that she communicates with B, while B thinks that he communicates with Eve.

The attack is also shown in Figure 4 below.

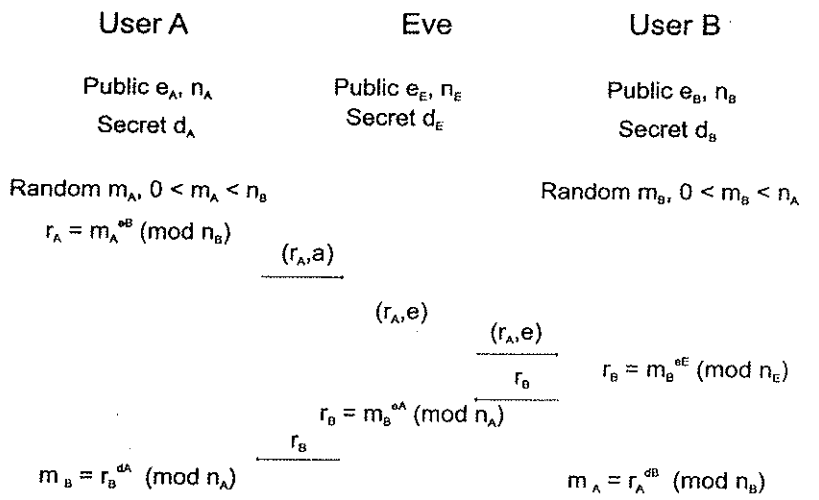

Figure 4: An unknown key-share attack on the two-way RSA key agreement protocol

3.3 Small subgroup attack

A small subgroup attack is an attack that operates in a large finite group where an adversary attempts to break a protocol by forcing a key to be part of an unexpectedly small subgroup of the desired group. For example, say that A and B follow a protocol to generate a share secret key and the protocol operates in a large finite group. An adversary Eve, can try to deduce some partial information about the shared secret key s by inducing either A or B to raise a group element α chosen by Eve to the power s. Then, if α is an element in a subgroup of small order l, then if Eve learns α^s, thus she can determine $s(\text{mod } l)$.

In this paper, we will prove that the small subgroup attack cannot be applied on the two-way RSA key agreement protocol using the fact that the RSA system is secure. We will prove it by first assuming that the small subgroup attack is applicable on the two-way RSA key agreement protocol and then come up with a contradiction.

The proof is as follows: assume that an attacker C is able to successfully apply the small group

attack to the two-way RSA key agreement protocol. This means that C knows e_A, n_A, e_B, n_B, $m_A^{e_B} \bmod n_B$ (random) and $m_B^{e_A} \bmod n_A$ (random). Thus he is able to generate in a deterministic and efficient way an $m_A'^{e_B} \bmod n_A$ and $m_B'^{e_A} \bmod n_B$ such that:

$$SK = m_A \bmod n_B \text{ xor } m'_B \bmod n_A = m'_A \bmod n_B \text{ xor } m_B \bmod n_A$$

is an element of a small known subset S of the set of integers, with $|S|$ is very small compared to maximum of $\{n_A, n_B\}$.

From this, it follows that C has a deterministic and efficient method to break the RSA system. The method goes like this:

1. First, C intercepts an encrypted message $m_A^{e_B} \bmod n_B$ that is sent from a party, say A, to another party, say B.

2. Then C applies the small group attack to the two way RSA key agreement protocol between A and B in which $r_A = m_A^{e_B} \bmod n_B$ is the first message sent by A and in which $r_B = m_B^{e_A} \bmod n_A$ (randomly made up C) is the second message. Note that C can easily make up such a second RSA key agreement protocol message since he only needs to know A's public RSA key for this.

3. After having applied the small group attack to the above instance of the two-way RSA key agreement protocol, C knows the values of $m_A'^{e_B} \bmod n_A$ and $m_B'^{e_A} \bmod n_B$ such that:

   $$SK = m_A \bmod n_B \text{ xor } m'_B \bmod n_A = m'_A \bmod n_B \text{ xor } m_B \bmod n_A$$

   is an element of a small known subset S of the set of integers, with $|S|$ is very small compared to maximum of n_A, n_B. Hence, C knows:

   $$m_A \bmod n_B = SK \text{ xor } m'_B \bmod n_A \quad \text{with } SK \in S, \ |S| \text{ very small.}$$

4. Now remember that C him generated the second two-way RSA key agreement protocol message $m_B^{e_A} \bmod n_A$. So C knows $m_B \bmod n_A$. Because the attack is deterministic this implies that C also knows the value of $m'_B \bmod n_A$. As a result C can easily determine $m_A \bmod n_B$ by exhaustively checking all possible values of S, which means breaking the RSA system.

Therefore, C can use the above method to break the RSA system. However, since the RSA system is assumed secure, this is extremely unlikely.

From the proof above, we can conclude that the small subgroup attack is not applicable on the two-way RSA key agreement protocol.

4 Countermeasures

Since small subgroup attack is not applicable on the two-way RSA key agreement protocol, we only consider the measures to prevent triangle attack and unknown key-share attack.

First, we discuss the measures to prevent the triangle attack. In the earlier section, it is shown that the attack can successfully be applied on the two-way RSA key agreement protocol since the shared secret keys are revealed to the adversary. Thus, the first thing that can be done is never reveal the shared secret keys to any parties. To make it more secure, the shared secret keys can also be destroyed immediately after the sessions are ended. Hence there is no way for the shared secret keys to be revealed. Another way to avoid the triangle attack is by preventing parties from knowing their own shared secret key. This method can be implemented by encapsulating the key distribution algorithm in a tamper resistant device. And to make it more secure, the encryption algorithm should also be encapsulated.

Another way to avoid triangle attack is modifying the key distribution system that enable other parties to get key confirmation. This method can be implemented by having each party send an additional message which contains the generated session key, for example through encryption. Then A cannot complete Step 2 and Step 3 of the attack on Section 3.1 and therefore B and C will destroy the shared secret keys.

The third method to avoid the triangle attack is by hashing the shared secret key into $h(SK)$ where $h(.)$ is a suitable hash function or the hash of the concatenation of the partial shared keys.

The last method is by strengthening the algorithm. Here, the users U_i, U_j are required to prove each other that they know the discrete logarithm of their calls r_i, r_j by using an interactive zero-knowledge proof. This will prevent the fraudulent runs; in our case it means B and C will not compute their session key unless they are convinced that A know the discrete logarithms of her calls. Furthermore, no knowledge about the session keys leaked to the adversary A.

Next, we discuss the measures to prevent the unknown key-share attack. The first solution is by requiring that messages should identify the sender and the recipient. This method is useful since if the identity is not integrated with the message itself, an adversary may intercept the message and changes the sender's identity with his identity. Another way is including a key confirmation step by which the intended participants can verify one anothers possession of the secret key.

In [2], there is another countermeasure that might be considered, namely a session key commitment: the parties exchange one-way hashes of their session public keys before exchanging the shared secret keys. The method of the exchange are important, as each party needs to be assured that the other party has received the commitment before it "decommits" and sends the public key. This assurance can be obtained by appropriate sequencing; for example, Alice sends her commitment, Bob receives it, and sends his commitment, Alice receives Bobs commitment and sends her key; then Bob receives Alices key and sends his key. Without appropriate sequencing, for instance,

if Alice and Bob exchange commitments in parallel, the protocol will still be vulnerable to attack.

The last method to prevent unknown key-share attack is by implementing some kind of "delay detection", which is not a cryptographic solution. The method requires a party terminates a run of the protocol if the other party takes too long time to reply.

5 Concluding remarks

This paper discusses the two-way RSA key agreement protocol, which is derived from an RSA cryptosystem, and the attacks that can be applied on it. The attacks discussed are a triangle attack, an unknown key-share attack and a small subgroup attack.

We can conclude that the triangle attack and the unknown key-share attack can be applied on the two-way RSA key agreement protocol, while this is not the case for the small subgroup attack.

Several countermeasures to prevent the protocol from the triangle attack and the unknown key-share attack are also discussed. To avoid the triangle attack, it is really important to not reveal or expose the shared secret keys to any parties. Key confirmation might be considered to be a good solution also. Another way is to use a hash function. Lastly, all parties involved should prove each other that they know the discrete logarithm of their calls using zero knowledge protocol. To avoid the unknown key-share attack, the first solution is by requiring that messages that identify the sender and recipient. Another way is including a key confirmation step by which the intended participants can verify one anothers possession of the secret key. Session commitment also can be applied to avoid unknown key-share attack. A delay detection, which is not a cryptography solution, may be useful in preventing the protocol from unknown key-share attack.

However, the protocol still need to be strengthened since some of the countermeasures are highly cost consuming to be realized. Moreover, the skill of the attackers become better and better. Once we make a new system, their skills will quickly improve to break the system.

References

[1] B. de Weger, Lecture note of Mathematics for Information Security course, Eindhoven University of Technology, 2004

[2] B. S. Kaliski, JR, *An unknown key-share attack on the MQV key agreement protocol*, Journal ACM Transactions on Information and System Security (TISSEC), Volume 4 Issue 3, August 2001

[3] C. van Pul, Lecture slide of Kaleidoscope in Information Security Technology course, Eindhoven University Technology, 2005

[4] D. Brown and A. Menezes, *A small subgroup attack on Arazi's key agreement protocol*, Bulletin of the Institute of Combinatorial Mathematics and its Applications, Volume 37, January 2003, pp. 45-50

[5] H. van Tilborg, *Fundamentals of Cryptology: A professional preference and interactive tutorial*, Kluwer academic publishers, 3rd edition, 2003

[6] M. Burmester, *On the risk of opening distributed keys*, CRYPTO '94 Proceedings of the 14th Annual International Cryptology, 1994

[7] L. Law et al., *An efficient protocol for authenticated key agreement*, Journal Designs, Codes and Cryptography, Volume 28 Issue 2, March 2003

[8] S. Hirose and S. Yoshida, *An authenticated Diffie-Hellman key agreement protocol*, PKC '98 Proceedings of the First International Workshop on Practice and Theory in Public Key Cryptography: Public Key Cryptography, 1998