Upgrading the S-NCI Key Establishment Protocol Scheme to be Secure and Applicable

Setiyo Budiyanto[1], Galih Bangun Santosa[2] and Fajar Rahayu Ikhwannul Mariati[3]

[1,3] Department of Electrical Engineering, Mercu Buana University, Jakarta, Indonesia
[2] National Crypto Institute, Bogor, Indonesia

E-mail: sbudiyanto@mercubuana.ac.id, genio.genio444@gmail.com

Abstract. There are currently a number of key establishment protocols that have been developed with server-based as trusted third parties. However, there are still some attacks that may occur in the protocols are man-in-the-middle attack, replay attack, typing attack, and modification attack. Following this, S-NCI's key establishment protocol has been developed which claims to be immune to man-in-the-middle attacks, replay attacks, typing attacks, and modification attacks. However, the protocol has not been through formal analysis. Then, based on experiments that have been done in this study, S-NCI has not met the formal analysis criteria of Alive and Weakaggree. In addition, S-NCI also has not provided an applicable supporting procedure for the utilization of Key Translation Center that is realized until now no one has applied. The research method used in this research is a research library supported by experimenting formal analysis of protocols using Scyther Tool. This research has produced a procedure supporting the application of S-NCI protocol along with the result of modification of the protocol so that it can fulfill Alive and Weakagree criteria.

1. Introduction

Key establishment is any process whereby a shared secret key becomes available to two or more parties, for subsequent cryptographic use [1]. The key establishment technique is divided into Key Transport and Key Agreement as illustrated in Figure 1.

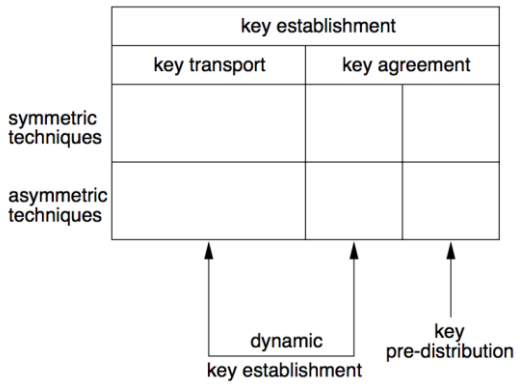

Figure 1. Simplified classification of key establishment techniques [1].

In practice, key establishment protocols can involve trusted third-parties as initial system setup and online actions [1][2][3][4][5][6][7]. Meanwhile, there are still some possible attacks on the current key establishment protocols that are man-in-the-middle attacks [1][2], replay attacks [1][2][3][4][5], typing attacks [1][2][4], and modification attacks [1][2][6][7]. Based on these conditions, it has now developed a protocol key establishment called S-NCI [8], which claims to have been immune to man-in-the-middle attack, replay attack, typing attack and modification attack. However, the protocol has not been through formal analysis that should be done [2]. Following this matter, then in this research will be conducted formal analysis of S-NCI protocol to be able to know weakness owned by protocol. In addition, the S-NCI protocol also does not provide an applicable implementation procedure for the utilization of Key Translation Center [1] which is realized that until now there has been no key establishment protocol to implement it. So this research aims to produce procedure of applying S-NCI protocol to be more applicable to modification of the protocol so that proven can fulfill all criteria of formal analysis using Scyther Tool [9][10][11][12].

2. Research Method

The research method used is a research library supported by conducting protocol security test experiment. Then, step research flowchart is described in Figure 2.

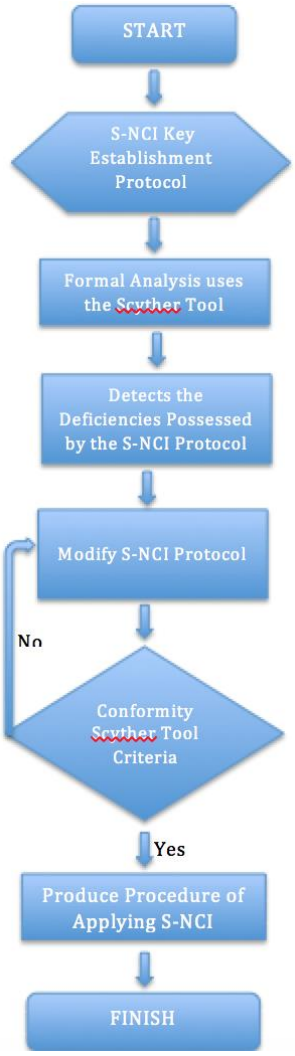

Figure 2. Step research flowchart

Formal analysis was conducted to support the results of informal analysis that had previously been done on [8]. Formal analysis of the S-NCI protocol was performed using the Scyther Tool, and it was detected that the Alive and Weakagree criteria were not met. So, then modify the existing S-NCI protocol to meet the Alive and Weakaggree criteria by adding the ID_T (identity T) and N_T (Nonce from T) attributes and one additional step as Step 5. After all criteria on formal analysis are met, then proceed with making supporting procedures in applying the S-NCI protocol to make it easier to apply.

3. Results and Formal Analysis

3.1. S-NCI Key Establishment Protocol

The notation used in the S-NCI key establishment protocol is described in Table 1. While the stages are described in Table 2.

Table 1. Notations and definitions of the S-NCI protocol

Notation	Definition
T	Trusted Third party as Key Translation Center (KTC)
A	Entity A
B	Entity B
E	The encryption process
K_{AT}	Encryption key between A and T
K_{BT}	Encryption key between B and T
ID_A	A identity
ID_B	B identity
K_S	Session Key
H_K	MAC hash function (key uses K_S)
H	Hash function
t	Timestamp
N	Nonce
||	Concate

Table 2. Stages in S-NCI protocol

$A \rightarrow T : E_{KAT}(K_s \| ID_A \| ID_B \| t_1) \| H(K_s \| ID_A \| ID_B \| t_1)$	Step 1
$T \rightarrow B : E_{KBT}(K_s \| ID_A \| t_2) \| H(K_s \| ID_A \| t_2)$	Step 2
$A \leftarrow B : E_{KS}(N_B \| t_3)$	Step 3
$A \rightarrow B : H_{KS}(N_B)$	Step 4

The explanation of Table 2 is as follows:

Step 1:

A generates a session key (K_s) and then sends the session key to T along with ID_A, ID_B, and t_1 encrypted using $_{KAT}$. Besides that A also sends the hash value from $K_s \| ID_A \| ID_B \| t_1$. T receives messages from A which are then decrypted using $_{KAT}$ and proceeds with hashing of received messages to check the integrity of received messages.

Step 2:

T encrypt messages containing $K_s||ID_A||t_2$ using $_{KBT}$, and doing hashing against $K_s||ID_A||t_2$, then the message and hash value are sent to B.

Step 3:

B decrypts the message received from T to get the value of the session key (K_s) and calculates the hash value to check the integrity of the message. Then B generates random numbers (N_B) and timestamp values (t_3). B will also calculate the MAC value of N_B. In addition, N_B and t_3 are then encrypted using K_s and sent to A.

Step 4:

A decrypt E_{KS} ($N_B||t_3$) by using K_s. Then A will calculate the MAC value with the K_s key from N_B and send it to B. When B receives the MAC value sent by A, B has previously calculated the MAC value of N_B. Thus, B will compare the MAC value to the one received in A. If the two hash values are the same, the protocol is successfully executed and A and B have a valid session key, K_s.

3.2. Formal Analysis uses Scyther Tool

Formal analysis of S-NCI key establishment protocols are performed using the Scyther Tool. The explanation of the source code and the results are described in sequence in Table 3 and Figure 3.

Table 3. Source code formal analysis of S-NCI key establishment protocol

Source Code : 01_KE_SNCI_beforeUpgrade.spdl

```
const Fresh: Function;
usertype String, SessionKey;
usertype TimeStamp;
usertype Nonce;
hashfunction h;
protocol SNCI(A,B,T)
{
        role A
        {
        fresh T1: TimeStamp;
        fresh Ks: SessionKey;
        var Nb: Nonce;
        var T3: TimeStamp;
        send_1(A,T, {Ks,A,B,T1}k(A,T),h(Ks,A,B,T1));
        recv_3(B,A,{Nb,T3}Ks);
        send_4(A,B,h(Nb));
        }
        role B
        {
        var T2: TimeStamp;
        var Ks: SessionKey;
        fresh Nb: Nonce;
        fresh T3: TimeStamp;
        recv_2(T,B,{Ks,A,T2}k(T,B),h(Ks,A,T2));
        send_3(B,A,{Nb,T3}Ks);
        recv_4(A,B,h(Nb));
        }
        role T
        {
        var T1: TimeStamp;
        var Ks: SessionKey;
        fresh T2: TimeStamp;
        recv_1(A,T, {Ks,A,B,T1}k(A,T),h(Ks,A,B,T1));
```

```
send_2(T,B,{Ks,A,T2}k(T,B),h(Ks,A,T2));
}}
```

SNCI	A	SNCI,A1	Secret Ks	Ok	Verified	No attacks.
		SNCI,A2	Secret T1	Ok	Verified	No attacks.
		SNCI,A3	Secret T3	Ok	Verified	No attacks.
		SNCI,A4	Secret Nb	Ok	Verified	No attacks.
		SNCI,A5	Alive	Ok	Verified	No attacks.
		SNCI,A6	Weakagree	Ok	Verified	No attacks.
		SNCI,A7	Niagree	Ok	Verified	No attacks.
		SNCI,A8	Nisynch	Ok	Verified	No attacks.
	T	SNCI,T4	Secret T2	Ok	Verified	No attacks.
		SNCI,T5	Secret Ks	Ok	Verified	No attacks.
		SNCI,T6	Secret T1	Ok	Verified	No attacks.
		SNCI,T7	Alive	Fail	Falsified	At least 1 attack.
		SNCI,T8	Weakagree	Fail	Falsified	At least 1 attack.
		SNCI,T9	Niagree	Ok	Verified	No attacks.
		SNCI,T10	Nisynch	Ok	Verified	No attacks.
	B	SNCI,B1	Secret T3	Ok	Verified	No attacks.
		SNCI,B2	Secret Nb	Ok	Verified	No attacks.
		SNCI,B3	Secret Ks	Ok	Verified	No attacks.

Figure 3. Scyther result: verify (before upgrading).

3.3. Upgrading the S-NCI Key Establishment Protocol

After performing a formal analysis of the S-NCI protocol using the Scyther Tool, it was found that the security characteristics of Alive and Weakagree on entity T are not owned by the protocol. So it is necessary to modify the S-NCI protocol by adding ID_T (identity T) and N_T (Nonce from T), and adding a step as a Step 5. Before and after modification of the S-NCI protocol is described in Table 4. Efforts to utilize and modify related protocols in various fields are also implemented on research [13][14][15].

Table 4. Stages of the S-NCI protocol before and after modification

Before Modification	
$A \rightarrow T : E_{KAT}(K_s \Vert ID_A \Vert ID_B \Vert t_1) \Vert H(K_s \Vert ID_A \Vert ID_B \Vert t_1)$	Step 1
$T \rightarrow B : E_{KBT}(K_s \Vert ID_A \Vert t_2) \Vert H(K_s \Vert ID_A \Vert t_2)$	Step 2
$A \leftarrow B : E_{KS}(N_B \Vert t_3)$	Step 3
$A \rightarrow B : H_{KS}(N_B)$	Step 4
After Modification	
$A \rightarrow T : E_{KAT}(K_s \Vert ID_A \Vert ID_B \Vert ID_T \Vert t_1) \Vert H(K_s \Vert ID_A \Vert ID_B \Vert ID_T \Vert t_1)$	Step 1
$T \rightarrow B : E_{KBT}(K_s \Vert ID_A \Vert ID_T \Vert N_T \Vert t_2) \Vert H(K_s \Vert ID_A \Vert ID_T \Vert N_T \Vert t_2)$	Step 2

$A \leftarrow B : E_{Ks}(N_B \| ID_T \| t_3)$	Step 3
$A \rightarrow B : H_{Ks}(N_B \| ID_T)$	Step 4
$B \rightarrow T : H_{Ks}(N_T)$	Step 5

The explanation of Table 4 (after modification) is as follows:
Step 1:
A generates a session key (K_s) and then sends the session key to T along with ID_A, ID_B, ID_T and t_1 encrypted using $_{KAT}$. Besides that A also sends the hash value from $K_s \| ID_A \| ID_B \| ID_T \| t_1$. T receives messages from A which are then decrypted using $_{KAT}$ and proceeds with hashing of received messages to check the integrity of received messages.
Step 2:
T encrypt messages containing $K_s \| ID_A \| ID_T \| N_T \| t_2$ using $_{KBT}$, and doing hashing against $K_s \| ID_A \| ID_T \| N_T \| t_2$, then the message and hash value are sent to B. T will also calculate the MAC value of N_T.
Step 3:
B decrypts the message received from T to get the value of the session key (K_s) and calculates the hash value to check the integrity of the message. Then B generates random numbers (N_B) and timestamp values (t_3). B will also calculate the MAC value of $N_B \| ID_T$. In addition, N_B ,ID_T and t_3 are then encrypted using K_s and sent to A.
Step 4:
A decrypt $E_{Ks}(N_B \| ID_T \| t_3)$ by using K_s. Then A will calculate the MAC value with the K_s key from $N_B \| ID_T$ and send it to B. When B receives the MAC value sent by A, B has previously calculated the MAC value of $N_B \| ID_T$. Thus, B will compare the MAC value to the one received in A. If the two hash values are the same, the protocol is successfully executed and A and B have a valid session key, K_s.
Step 5:
Next, B will calculate the MAC value with the key K_s to N_T and send it to T. When T receives the MAC value sent by B, T has previously calculated the MAC value of N_T. Then, T will compare the MAC value, so if both are equal, then B and T have succeeded in authenticating each other.

3.4. Formal Analysis after Modified
Formal analysis of the modified S-NCI key establishment protocol, source code and results are described in sequence in Table 5 and Figure 4.

Table 5. Source code formal analysis of the modified S-NCI key establishment protocol

Source Code : 02_KE_SNCI_upgrade_send3_idT_dan_send5_Nt.spdl

```
const Fresh: Function;
usertype String, SessionKey;
usertype TimeStamp;
usertype Nonce;
hashfunction h;
protocol SNCI(A,B,T)
{
        role A
      {
      fresh T1: TimeStamp;
      fresh Ks: SessionKey;
      var Nb: Nonce;
      var T3: TimeStamp;
```

```
        send_1(A,T, {Ks,A,B,T,T1}k(A,T),h(Ks,A,B,T,T1));
        recv_3(B,A,{Nb,T,T3}Ks);
        send_4(A,B,h(Nb,T));
        }
        role T
        {
        var T1: TimeStamp;
        var Ks: SessionKey;
        fresh T2: TimeStamp;
        fresh Nt: Nonce;
        recv_1(A,T, {Ks,A,B,T,T1}k(A,T),h(Ks,A,B,T,T1));
        send_2(T,B,{Nt,Ks,A,T,T2}k(T,B),h(Nt,Ks,A,T,T2));
        recv_5(B,T,h(Nt));
        }
        role B
        {
        var T2: TimeStamp;
        var Ks: SessionKey;
        fresh Nb: Nonce;
        fresh T3: TimeStamp;
        var Nt: Nonce;
        recv_2(T,B,{Nt,Ks,A,T,T2}k(T,B),h(Nt,Ks,A,T,T2));
        send_3(B,A,{Nb,T,T3}Ks);
        recv_4(A,B,h(Nb,T));
        send_5(B,T,h(Nt));
        }
}
```

SNCI	A	SNCI,A1	Secret Ks	Ok	Verified	No attacks.
		SNCI,A2	Secret T1	Ok	Verified	No attacks.
		SNCI,A3	Secret T3	Ok	Verified	No attacks.
		SNCI,A4	Secret Nb	Ok	Verified	No attacks.
		SNCI,A5	Alive	Ok	Verified	No attacks.
		SNCI,A6	Weakagree	Ok	Verified	No attacks.
		SNCI,A7	Niagree	Ok	Verified	No attacks.
		SNCI,A8	Nisynch	Ok	Verified	No attacks.
	T	SNCI,T4	Secret Nt	Ok	Verified	No attacks.
		SNCI,T5	Secret T2	Ok	Verified	No attacks.
		SNCI,T6	Secret Ks	Ok	Verified	No attacks.
		SNCI,T7	Secret T1	Ok	Verified	No attacks.
		SNCI,T8	Alive	Ok	Verified	No attacks.
		SNCI,T9	Weakagree	Ok	Verified	No attacks.
		SNCI,T10	Niagree	Ok	Verified	No attacks.
		SNCI,T11	Nisynch	Ok	Verified	No attacks.
	B	SNCI,B1	Secret T3	Ok	Verified	No attacks.
		SNCI,B2	Secret Nb	Ok	Verified	No attacks.

Figure 4. Scyther result: verify (after upgrading).

Based on the results of a formal analysis carried out on the modified S-NCI protocol, it is evident that Alive and Weakagree criteria have been met, so that the modification has succeeded in eliminating the weaknesses of the previous S-NCI protocol.

3.5. Procedure of Applying S-NCI Protocol

The supporting procedures that need to be applied in the application of the S-NCI key establishment protocol is as follows:

a. KTC can be as a cloud service provider or a Cloud application provider specified. KTC is recognized as a trusted entity to provide remote data storage and remote counting services, which in this case is more specifically functioned as a party that translates inter-party keys with an agreed KTC, such as the key between A with KTC and B with KTC.
b. KTC specifies the random number ($Rand_T$). After that will be determined the ID of each entity involved, in this case entities A and B, in the following way:
   - $ID_T = H(Rand_T)$
   - $ID_A = H(Rand_T \,||\, Rand_A)$
   - $ID_B = H(Rand_T \,||\, Rand_B)$
c. Once the ID of each entity is obtained, the S-NCI key supply protocol is ready to be applied.

4. Conclusion

This study has the following conclusions :

a. This protocol can be used applicable and securely for the various functions of the key establishment protocol that is the function of providing session keys, challenge response, mutual authentication, data integrity, and data encryption.
b. This protocol is proven to meet the Alive and Weakagree security criteria for previously unachieved entity T, after adding a new step as a fifth step by adding ID_T and N_T notation.
c. The procedure of applying KTC in the S-NCI protocol requires the identities (ID) of each entity involved, which can be generated from the hash random number (Rand) between the entities involved.

References

[1] A. J. Menezes, P. C. Van Oorschot, and S. a. Vanstone, *Handbook of Applied Cryptography*, vol. 106. 1997.
[2] B. Colin and M. Anish, *Information Security and Cryptography Texts and Monographs Springer-Verlag Berlin Heidelberg GmbH*. 2003.
[3] R. M. Needham and M. D. Schroeder, “Using encryption for authentication in large networks of computers,” *Commun. ACM*, vol. 21, no. 12, pp. 993–999, 1978.
[4] Z. Cheng and R. Comley, “Attacks on an ISO/IEC 11770-2 key establishment protocol,” *Int. J. Netw. Secur.*, vol. 3, no. 3, pp. 290–295, 2006.
[5] D. E. Denning and G. M. Sacco, “Timestamps in key distribution protocols,” *Commun. ACM*, vol. 24, no. 8, pp. 533–536, 1981.
[6] S. G. Stubblebine and V. D. Gligor, “On message integrity in cryptographic protocols,” *Proc. 1992 IEEE Comput. Soc. Symp. Res. Secur. Priv.*, pp. 85–104, 1992.
[7] W. Mao and C. Boyd, “On The Use of Encryption in Cryptographic Protocols,” *Proc. 4th IMA Conf. Cryptogr. Coding*, 1995.
[8] M. A. L. I. Sadikin and S. Windarta, “S-NCI : DESAIN PROTOKOL KEY ESTABLISHMENT,” *Proc. - Semin. Nas. Mat. Univ. Indones.*, no. Universitas Indonesia, pp. 1–10, 2017.
[9] C. J. F. Cremers, “Scyther : Unbounded Verification of Security Protocols,” no. 572, pp. 1–18, 2006.

[10] O. Pavel, "Analysis of authentication protocols with scyter: Case study," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 6771 LNCS, no. PART 1, pp. 359–365, 2011.
[11] P. Activities, P. Models, T. S. Tool, and T. Nguyen, "Cas Cremers." [Online]. Available: https://www.cs.ox.ac.uk/people/cas.cremers/scyther/. [Accessed: 02-Jan-2018].
[12] N. Dalal, J. Shah, K. Hisaria, and D. Jinwala, "A Comparative Analysis of Tools for Verification of Security Protocols," *Int. J. Commun. Netw. Syst. Sci.*, vol. 3, no. 10, pp. 779–787, 2010.
[13] S. Budiyanto, M. Asvial, and D. Gunawan, "Performance Analysis of Genetic Zone Routing Protocol Combined With Vertical Handover Algorithm for 3G-WiFi Offload," *J. ICT Res. Appl.*, vol. 8, no. 1, pp. 49–63, 2014.
[14] M. Asvial, S. Budiyanto, and D. Gunawan, "An intelligent load balancing and offloading in 3G - WiFi offload network using hybrid and distance vector algorithm," *IEEE Symp. Wirel. Technol. Appl. ISWTA*, pp. 36–40, 2014.
[15] S. Budiyanto and A. Nugroho, "A New Model of Genetic Zone Routing Protocol (GZRP): The Process of Load Balancing and Offloading on The," vol. 15, no. 2, pp. 598–605, 2017.