Attacks On An ISO/IEC 11770-2 Key Establishment Protocol

Zhaohui Cheng*and Richard Comley

September 23, 2004

School of Computing Science
Middlesex University
White Hart Lane, London N17 8HR, UK.
{m.z.cheng, r.comley}@mdx.ac.uk

Abstract

Two possible types of attack (a replay attack and a type attack) on a key establishment protocol (mechanism 12) standardised in ISO/IEC 11770-2 are described and two solutions are proposed.

1 Introduction

In this paper we demonstrate that a server-based protocol standardised as key establishment mechanism 12 in ISO/IEC 11770-2 [7] is vulnerable to a replay attack and also, depending on the implementation, to a type attack. We propose two possible solutions to fix these problems.

Mechanism 12 in [7] is specified as Protocol 1, where all the optional text fields are ignored (the notation used throughout the paper is summarised in Table 1). Protocol 1 is a two-party key transport

X	The distinguish identifier of entity X
S	A trusted server
$\{M\}_K$	The encryption of message M with key K to provide both confidentiality and integrity
TVP_X	A time variant parameter, such as a random number, a timestamp or a sequence number generated by entity X
K_{XY}	The key intended to be shared by entity X and Y
R_X	A random number chosen by entity X
T_X	A timestamp issued by entity X
N_X	A sequence number issued by entity X
T_X/N_X	A timestamp or a sequence number issued by entity X
C_X	An attacker C impersonating entity X
$[M]$	An optional message M
X, Y	The result of the concatenation of data strings X and Y

Table 1: Notation

protocol with a trusted server (often called a three-party protocol in the literature). Basically the protocol assumes that all parties, including the server in the system have loosely synchronised clocks (or

*Associated with Olym-Tech Inc. as an external researcher.

1.	$A \to S$	:	$\{TVP_A, B, K_{AB}\}_{K_{AS}}$
2.	$S \to A$	:	$\{TVP_A, B\}_{K_{AS}}, \{T_S/N_S, K_{AB}, A\}_{K_{BS}}$
3.	$A \to B$	:	$\{T_S/N_S, K_{AB}, A\}_{K_{BS}}, [\{T_A/N_A, B\}_{K_{AB}}]$
4.	$B \to A$	:	$[\{T_B/N_B, A\}_{K_{AB}}]$

Protocol 1: ISO/IEC 11770-2 Key Establishment Mechanism 12

synchronised sequence numbers) and each user shares a long-term secret key with the trusted server. The initiator (the user who sends the first message in a run of a protocol) controls session key generation and wishes to transport the chosen session key securely to the responder (the user with whom the initiator wants to securely communicate later) with the help of the trusted server. An optional handshake in the protocol is designed to provide entity authentication and key confirmation.

The standard allows the initiator to use one of three different types of time variant parameter (TVP) to secure the freshness of an established session key. However, we show that the protocol is seriously flawed if a random number or a timestampe is used for the TVP.

2 Attacks On The Protocol And Fixing Variants

2.1 A Replay Attack

If the TVP field contains a random number, Protocol 1 suffers from a replay attack[1]. Note that the two parts of message (2) use two different freshness mechanisms, but these two mechanisms are not securely associated. More precisely, the initiator uses a random number to check the freshness of the first part of message (2) and the responder uses a timestamp (or a sequence number) to verify the freshness of message (3), which is equal to the second part of message (2) as generated by the server S. However S has no means of verifying that message (1) is fresh.

1.	$A \to S$	:	$\{R_A, B, K_{AB}\}_{K_{AS}}$
2.	$S \to A$	:	$\{R_A, B\}_{K_{AS}}, \{T_S/N_S, K_{AB}, A\}_{K_{BS}}$
3.	$A \to B$	:	$\{T_S/N_S, K_{AB}, A\}_{K_{BS}}$
$1'$.	$C_A \to S$	:	$\{R_A, B, K_{AB}\}_{K_{AS}}$
$2'$.	$S \to C_A$	:	$\{R_A, B\}_{K_{AS}}, \{T'_S/N'_S, K_{AB}, A\}_{K_{BS}}$
$3'$.	$C_A \to B$	:	$\{T'_S/N'_S, K_{AB}, A\}_{K_{BS}}$

Attack 1: Replay Attack On Protocol 1 Using Random Numbers

The attack (**Attack 1**) using a compromised session key [6] proceeds as follows. Suppose that attacker C has recorded one valid run of the protocol between A and B and has, by some means, successfully recovered the session key K_{AB} established in that run. C launches an attack by impersonating A to initiate a new run of the protocol by replaying the first recorded message (1) as message ($1'$). The attacker intercepts message ($2'$) sent as a reply to A by S and sends the second part of message ($2'$) to B as message ($3'$). Because message ($3'$) includes the new timestamp T'_S (or the new sequence number N'_S) issued by the trusted server S, B will accept K_{AB} as the valid session key to be used with A. Clearly the protocol's goal to achieve key freshness is thwarted. Even with the key confirmation option the protocol is still vulnerable to the attack. Note that when a sequence number is used in the optional messages, C needs to know A's current sequence number N'_A which is normally assumed to be predictable.

[1]In Annex A (properties of key establishment mechanisms) of the standard [7], it is stated that if the replay detection is required then a timestamp or a sequence number should be used for the TVP

One way of preventing this attack would be to use a timestamp or a sequence number for TVP_A, and to require the server to check the freshness of TVP_A[2]. However, as we show below, another attack still applies in this case.

2.2 A Type Attack

Even if an implementer of Protocol 1 uses a timestamp for TVP_A, a type attack (see, for example, [4], [5]) may become feasible, depending on the details of the implementation.

The type attack (**Attack 2**) works as follows (for simplicity we suppose that the identifier field and the key field are of the same size. More sophisticated examples are demonstrated in Appendix 1). First attacker C chooses a victim user V and a user B whom it wants to impersonate to V. Then C initiates a run of the protocol in which it pretends to intend to send user B a session key equal to the identifier V. Whether or not this will be possible in practice depends on whether the identifier V would be accepted as a valid key value by the server S, which is why we describe the attack as being implementation-dependent. After receiving message (2) from S, C replays part 2 of message (2) to S as the first message ($1'$) of a new run of the protocol to impersonate B. After receiving message ($1'$), S will treat the identifier C as the chosen session key of this run between B and V. The freshness check based on timestamp T_S will succeed. C intercepts message ($2'$) and forwards part 2 of the message to the victim V. After checking the validity of message ($3'$), V will treat the identifier C as the session key for use with B.

1.	$C \to S$	:	$\{T_C, B, V\}_{K_{CS}}$
2.	$S \to C$	:	$\{T_C, B\}_{K_{CS}}, \{T_S, V, C\}_{K_{BS}}$
$1'$.	$C_B \to S$	:	$\{T_S, V, C\}_{K_{BS}}$
$2'$.	$S \to C_B$	:	$\{T_S, V\}_{K_{BS}}, \{T'_S, C, B\}_{K_{VS}}$
$3'$.	$C_B \to V$	:	$\{T'_S, C, B\}_{K_{VS}}$

Attack 2: Type Attack On Protocol 1 Using Timestamps

Similarly, if we suppose that the server cannot differentiate a valid random number from a timestamp (or a sequence number) (e.g. both use a 32-bit integer), the aforementioned type attack is also feasible to the protocol using a random number for the TVP, e.g. **Attack 3**. Note that the attack also applies in mechanism 11 of the standard [7] (refer to Appendix 2 for how the type attack works on mechanism 11).

1.	$C \to S$	:	$\{R_C, B, V\}_{K_{CS}}$
2.	$S \to C$	:	$\{R_C, B\}_{K_{CS}}, \{T_S/N_S, V, C\}_{K_{BS}}$
$1'$.	$C_B \to S$	:	$\{T_S/N_S, V, C\}_{K_{BS}}$
$2'$.	$S \to C_B$	:	$\{T_S/N_S, V\}_{K_{BS}}, \{T'_S/N'_S, C, B\}_{K_{VS}}$
$3'$.	$C_B \to V$	:	$\{T'_S/N'_S, C, B\}_{K_{VS}}$

Attack 3: Type Attack On Protocol 1 Using Random Numbers

2.3 Fixing Variants

Possible means to fix the identified vulnerabilities are now described. In fact, the three types of TVP have very different security properties. We would like to treat the protocol using different TVP as a

[2]Note that regardless of the type of TVP in use the specification [7] does not clearly require the server to check the freshness of the TVP field of message (1))

different protocol. The protocol using a sequence number for the TVP is the most robust one among three. Please refer to Appendix 3 for some security related issues using sequence numbers in the protocol.

When the protocol uses a random number in the TVP, it is faced with two types of attack, a replay attack and a type attack. ISO indicates in Annex A of [7] that if the replay detection is required, the protocol should not be used in this way. However, we find that the protocol can be modified to prevent the replay attack. Although the server cannot check the freshness of message (1) when the TVP uses a random number, we propose a simple strategy to counter the replay attack, i.e. the server S secretly transmits the content that will be used as message (3) to the initiator in message (2). The variant is specified as **Protocol 2**. Without the long-term secret key K_{AS}, attacker C cannot recover the part

1.	$A \rightarrow S$	:	$\{R_A, B, K_{AB}\}_{K_{AS}}$
2.	$S \rightarrow A$	:	$\{R_A, B, \{T_S/N_S, K_{AB}, A\}_{K_{BS}}\}_{K_{AS}}$
3.	$A \rightarrow B$	:	$\{T_S/N_S, K_{AB}, A\}_{K_{BS}}, [\{T_A/N_A, B\}_{K_{AB}}]$
4.	$B \rightarrow A$	:	$[\{T_B/N_B, A\}_{K_{AB}}]$

Protocol 2: Fixed Protocol Using Random Numbers

$\{T_S, K_{AB}, A\}_{K_{BS}}$ in message (2), and so cannot generate a message (3) to deceive B into accepting an old session key. The replay attack is thus prevented. Note that the type attack is essentially a replay attack. Hence Protocol 2 also defeats the aforementioned type attack.

When the protocol merely uses a timestamp and the server checks the correctness of the TVP field, we must deal with the type attack. To prevent the attack, we can tweak Protocol 1 by switching the positions of the identifier A and the session key in message (2) and (3) respectively. The tweaked version (**Protocol** $1'$) is essentially similar to the wide-mouthed-frog protocol [3]. Unfortunately, just as the

1.	$A \rightarrow S$	:	$\{T_A, B, K_{AB}\}_{K_{AS}}$
2.	$S \rightarrow A$	:	$\{T_A, B\}_{K_{AS}}, \{T_S, A, K_{AB}\}_{K_{BS}}$
3.	$A \rightarrow B$	:	$\{T_S, A, K_{AB}\}_{K_{BS}}$

Protocol $1'$**:** Tweaked Version Of Protocol 1 Using Timestamps

wide-mouthed-frog protocol, Protocol $1'$ is vulnerable to a reflection attack (**Attack 4**) [9][1][5]. In the attack, attacker C can keep the server generating an up-to-date message containing the key K_{AB} until it compromises the key.

1.	$A \rightarrow S$	:	$\{T_A, B, K_{AB}\}_{K_{AS}}$
2.	$S \rightarrow A$	:	$\{T_A, B\}_{K_{AS}}, \{T_S, A, K_{AB}\}_{K_{BS}}$
3.	$A \rightarrow B$	:	$\{T_S, A, K_{AB}\}_{K_{BS}}$
$1'$.	$C_B \rightarrow S$	:	$\{T_S, A, K_{AB}\}_{K_{BS}}$
$2'$.	$S \rightarrow C_B$	:	$\{T_S, A\}_{K_{BS}}, \{T'_S, B, K_{AB}\}_{K_{AS}}$
$1''$.	$C_A \rightarrow S$	:	$\{T'_S, B, K_{AB}\}_{K_{AS}}$
$2''$.	$S \rightarrow C_A$	:	$\{T'_S, B\}_{K_{AS}}, \{T''_S, A, K_{AB}\}_{K_{BS}}$

Attack 4: Reflection Attack On Protocol $1'$

However, by introducing asymmetry between message (1) and (3), another variant (**Protocol** 3) which uses timestamps, can prevent the described attack, if the implementation processes every byte of a received message[3]. More systematic strategies to counter replay attacks can be found in [2]. The

[3] A poor implementation not processing a received message completely will suffer from the reflection attack

same strategy as adopted by Protocol 2 also works here. Hence we suggest a fixed universal protocol (**Protocol** 4) suitable for all the three different types of time variant parameter. Moreover, now it seems that the positions of the identifier A and the session key in message (2) and (3) are immaterial to the security of Protocol 4.

1.	$A \rightarrow S$	:	$\{T_A, B, K_{AB}\}_{K_{AS}}$
2.	$S \rightarrow A$	:	$\{T_A, B\}_{K_{AS}}, \{T_S, B, K_{AB}, A\}_{K_{BS}}$
3.	$A \rightarrow B$	:	$\{T_S, B, K_{AB}, A\}_{K_{BS}}, [\{T_A, B\}_{K_{AB}}]$
4.	$B \rightarrow A$	:	$[\{T_B, A\}_{K_{AB}}]$

Protocol 3: Fixed Protocol Using Timestamps

1.	$A \rightarrow S$	:	$\{TVP_A, B, K_{AB}\}_{K_{AS}}$
2.	$S \rightarrow A$	:	$\{TVP_A, B, \{T_S/N_S, K_{AB}, A\}_{K_{BS}}\}_{K_{AS}}$
3.	$A \rightarrow B$	:	$\{T_S/N_S, K_{AB}, A\}_{K_{BS}}, [\{T_A/N_A, B\}_{K_{AB}}]$
4.	$B \rightarrow A$	:	$[\{T_B/N_B, A\}_{K_{AB}}]$

Protocol 4: The Fixed Universal Protocol

3 Conclusion

Serious vulnerabilities in a server-based key establishment protocol standardised by ISO/IEC has been identified. Possible solutions to prevent these attacks are also proposed. One alternative to use of a modified version of the flawed protocol is to adopt an alternative (but sound) standardised protocol, such as the four-pass authentication protocol in [8], although in which the server controls the key generation.

4 Acknowledgement

We would like to thank Chris Mitchell for his comments and constructive criticism.

References

[1] Ross Anderson and Roger Needham. Programming satan's computer. *Computer Science Today: Recent Trends and Developments*, pages 426–440, 1995.

[2] Tuomas Aura. Strategies against replay attacks. *10th IEEE Computer Security Foundations Workshop*, 1997.

[3] Michael Burrows, Martín Abadi, and Roger Needham. A logic of authentication. *Proceedings of the Royal Society of London*, A426:233–271, 1989.

[4] Ulf Carlsen. Cryptographic protocol flaws. *In Proceedings 7th IEEE Computer Security Foundations Workshop*, pages 192–200, 1994.

[5] John Clark and Jeremy Jacob. Attacking authentication protocols. *High Integrity Systems*, 1(5):465–473, August 1996.

[6] Dorothy E. Denning and Giovanni Maria Sacco. Timestamps in key distribution protocols. *Communications of the ACM*, 24(8):533–536, August 1981.

[7] ISO. Information technology-security techniques-key management-part 2: Mechanisms using symmetric techniques. *ISO/IEC 11770-2*, 1996.

[8] ISO. Information technology-security techniques-entity authentication-part 2: Mechanisms using symmetric encipherment algorithms. *ISO/IEC 9798-2*, the second edition, 1998.

[9] Chris J. Mitchell. Limitations of challenge-response entity authentication. *Electronics Letters*, 25:1195–1196, 1989.

Appendix 1. Implementation Examples Vulnerable to The Type Attack

Suppose in a system with a large number of users, each user is identified by a 32-bit integer and a session key is a 64-bit random number (key size is immaterial in the following attack. We use short session keys merely for better paper format). An implementation of Protocol 1 (mechanism 12 in [7]) whose message formats (we call this type of message format a *typeless message format*) are defined using C language as follows is vulnerable to the type attack presented in the paper.

```
/*A-->S: message 1*/
struct msg1{
    unsigned int timestampe;
    unsigned int id;
    unsigned int key[2];
};


/*S-->A: part (a) of message 2*/
struct msg2a{
    unsigned int timestampe;
    unsigned int id;
};


/*S-->A: part (b) of message 2*/
struct msg2b{
    unsigned int timestampe;
    unsigned int key[2];
    unsigned int id;
};


/*A-->B: part (a) of message 3*/
struct msg3a{
    unsigned int timestampe;
    unsigned int key[2];
    unsigned int id;
};

/* option part of message 3 and 4*/
struct option{
    unsigned int timestampe;
    unsigned int id;
};
```

Suppose the current timestamp is 0x00000001 and C's (B's and V's, resp.) identifier is 0x0000000C (0x0000000B and 0x0000000F, resp.). X denotes any 4-bit number. The messages of the type attack can be detailed as follows.

$$
\begin{array}{rll}
C \to S: & \text{Message 1.} & \{\underbrace{00000001}_{T_C}, \underbrace{0000000B}_{B}, \underbrace{0000000FXXXXXXXX}_{K}\}_{K_{CS}} \\
S \to C: & \text{Message 2a.} & \{\underbrace{00000001}_{T_C}, \underbrace{0000000B}_{B}\}_{K_{CS}} \\
S \to C: & \text{Message 2b.} & \{\underbrace{00000001}_{T_S}, \underbrace{0000000FXXXXXXXX}_{K}, \underbrace{0000000C}_{C}\}_{K_{BS}} \\
C_B \to S: & \text{Message 1}'. & \{\underbrace{00000001}_{T_S}, \underbrace{0000000F}_{V}, \underbrace{XXXXXXXX0000000C}_{K'}\}_{K_{BS}} \\
S \to C_B: & \text{Message 2}'\text{a.} & \{\underbrace{00000001}_{T_S}, \underbrace{0000000F}_{V}\}_{K_{BS}} \\
S \to C_B: & \text{Message 2}'\text{b.} & \{\underbrace{00000002}_{T'_S}, \underbrace{XXXXXXXX0000000C}_{K'}, \underbrace{0000000B}_{B}\}_{K_{VS}} \\
C_B \to V: & \text{Message 3}'\text{a.} & \{\underbrace{00000002}_{T'_S}, \underbrace{XXXXXXXX0000000C}_{K'}, \underbrace{0000000B}_{B}\}_{K_{VS}} \\
C_B \to V: & \text{Message 3}'\text{b.} & \{\underbrace{00000002}_{T'_C}, \underbrace{0000000F}_{V}\}_{K'} \\
V \to C_B: & \text{Message 4}'. & \{\underbrace{00000002}_{T'_V}, \underbrace{0000000B}_{B}\}_{K'}
\end{array}
$$

Table 2: Message Details Of Type Attack On Protocol 1

In fact, it is not necessary that the identifier field is an integer. The attack also works in some implementations where users are identified by character strings. Furthermore, we stress that the positions of the elements in the messages is crucial to the security of Protocol 3. We show an example to demonstrate these points. The following protocol (**Protocol 5**) is our first attempt to fix the protocol using a timestamp for TVP_A. The only difference between Protocol 5 and Protocol 3 is the position of responder's identifier in message (2) and (3). But an implementation of Protocol 5 using a typeless message format is still possibly vulnerable to the type attack, even if it processes every byte of a received message. Suppose in a system, each user is identified by a character string and a session key is a 128-bit

1.	$A \to S$	:	$\{T_A, B, K_{AB}\}_{K_{AS}}$
2.	$S \to A$	:	$\{T_A, B\}_{K_{AS}}, \{T_S, K_{AB}, A, B\}_{K_{BS}}$
3.	$A \to B$	:	$\{T_S, K_{AB}, A, B\}_{K_{BS}}, [\{T_A, B\}_{K_{AB}}]$
4.	$B \to A$	:	$[\{T_B, A\}_{K_{AB}}]$

Protocol 5: Partially-fixed Protocol Using Timestamps

random number. In an implementation, each party uses a special symbol to indicate the end of a string (in fact even using an extra length field for each string, the attack is still possible), for example, in C language, the commonly used symbol is '\0' and a character is represented by 8 bits. Assume the current timestamp is 0x00000001 and attacker C's identifier is "michael" and user B and V's identifier is "richard" and "c.chris@msn.com" respectively. The messages of the attack are as follows.

$$
\begin{array}{rll}
C \to S: & \text{Message 1.} & \{\underbrace{00000001}_{T_C}, \underbrace{richard\backslash 0}_{B}, \underbrace{c.chris@msn.com\backslash 0}_{K}\}_{K_{CS}} \\
S \to C: & \text{Message 2a.} & \{\underbrace{00000001}_{T_C}, \underbrace{richard\backslash 0}_{B}\}_{K_{CS}} \\
S \to C: & \text{Message 2b.} & \{\underbrace{00000001}_{T_S}, \underbrace{c.chris@msn.com\backslash 0}_{K}, \underbrace{michael\backslash 0}_{C}, \underbrace{richard\backslash 0}_{B}\}_{K_{BS}} \\
C_B \to S: & \text{Message } 1'. & \{\underbrace{00000001}_{T_S}, \underbrace{c.chris@msn.com\backslash 0}_{V}, \underbrace{michael\backslash 0richard\backslash 0}_{K'}\}_{K_{BS}} \\
S \to C_B: & \text{Message } 2'\text{a.} & \{\underbrace{00000001}_{T_S}, \underbrace{c.chris@msn.com\backslash 0}_{V}\}_{K_{BS}} \\
S \to C_B: & \text{Message } 2'\text{b.} & \{\underbrace{00000002}_{T'_S}, \underbrace{michael\backslash 0richard\backslash 0}_{K'}, \underbrace{richard\backslash 0}_{B}, \underbrace{c.chris@msn.com\backslash 0}_{V}\}_{K_{VS}} \\
C_B \to V: & \text{Message } 3'\text{a.} & \{\underbrace{00000002}_{T'_S}, \underbrace{michael\backslash 0richard\backslash 0}_{K'}, \underbrace{richard\backslash 0}_{B}, \underbrace{c.chris@msn.com\backslash 0}_{V}\}_{K_{VS}} \\
C_B \to V: & \text{Message } 3'\text{b.} & \{\underbrace{00000002}_{T'_C}, \underbrace{c.chris@msn.com\backslash 0}_{V}\}_{K'} \\
V \to C_B: & \text{Message } 4'. & \{\underbrace{00000002}_{T'_V}, \underbrace{richard\backslash 0}_{B}\}_{K'}
\end{array}
$$

Table 3: Message Details Of Type Attack On Protocol 5

Appendix 2. A Type Attack On Mechanism 11

The key establishment mechanism 11 in [7] is a simplified version of mechanism 12 which does not provide key freshness.

1.	$A \to S:$	$\{B, K_{AB}\}_{K_{AS}}$
2.	$S \to A:$	$\{K_{AB}, A\}_{K_{BS}}$
3.	$A \to B:$	$\{K_{AB}, A\}_{K_{BS}}$

Protocol 6: ISO/IEC 11770-2 Key Establishment Mechanism 11

The protocol is also vulnerable to the type attack.

1.	$C \to S$	:	$\{B, V\}_{K_{CS}}$
2.	$S \to C$	:	$\{V, C\}_{K_{BS}}$
$1'$.	$C_B \to S$	:	$\{V, C\}_{K_{BS}}$
$2'$.	$S \to C_B$	:	$\{C, B\}_{K_{VS}}$
$3'$.	$C_B \to V$	:	$\{C, B\}_{K_{VS}}$

Attack 5: Type Attack On Protocol 6

Appendix 3. Using Sequence Numbers In The Protocol

Now let us investigate the feasibility of applying the identified attacks on the protocol using a sequence number for the TVP. In the system, each user has a synchronised sequence number with the server. The standard does not specify how to synchronize the sequence numbers between entities. However, in an unreliable network with parallel running sessions it is not an easy job to synchronize these numbers. In the standard, the server S is required to include its own sequence number N_S in the second part of message (2), which means S has only one sequence number shared with all the users and will update its sequence number after each run of the protocol. This method introduces a problem in practice when parallel sessions are allowed. For example, both A and B initiate a run of the protocol with the same responder C. S first generates the reply to A. Hence part 2 of message (2) replied to A includes a smaller sequence number than the one included in message (2) to B. However, it is possible that B's message (3) first reaches C. Upon receiving message (3) from B, C will accept the message because the sequence number in the message is greater than its local copy of N_S and then updates its N_S. Unfortunately, message (3) from A will be rejected later although it is a valid message generated by S. Even worse, C has no means to securely inform A this error because C does not share a secret with A. The same problem happens even if S maintains a separate sequence number for each user to send messages to that user (that is $N_{XS} \neq N_{SX}$). Here N_{XY} denotes the current sequence number of entity X for sending messages to Y.

The above problem can be solved if N_{XS} is used in part 2 of message (2) (i.e. $N_{XS} = N_{SX}$). That is the server will use user X's current sequence number in all the messages sent to X and will update N_{XS} only if it receives a valid message (1) from X which indicates that X has initiated a new run of the protocol. Hence N_{XS} is the counter of the runs of the protocol initiated by X. Although it works in many situations, the mechanism introduces serious flaws in this protocol. The replay attack and the type attack become feasible. Suppose attacker C has recorded a valid run of the protocol between A (the initiator) and B (the responder) and has successfully recovered the session key K_{AB}. If B has not initiated a new run of the protocol after K_{AB} was compromised, C can replay message (3) of the recorded run to deceive B into accepting K_{AB}. The type attack is described as follows. Hence using a single N_S for message (2) to every user is a relatively better (secure) solution, although it does not always work.

1.	$C \rightarrow S$	:	$\{N_{CS}, B, V\}_{K_{CS}}$
2.	$S \rightarrow C$	:	$\{N_{CS}, B\}_{K_{CS}}, \{N_{BS}, V, C\}_{K_{BS}}$
$1'$.	$C_B \rightarrow S$	:	$\{N_{BS}, V, C\}_{K_{BS}}$
$2'$.	$S \rightarrow C_B$	:	$\{N_{BS}, V\}_{K_{BS}}, \{N_{VS}, C, B\}_{K_{VS}}$
$3'$.	$C_B \rightarrow V$	:	$\{N_{VS}, C, B\}_{K_{VS}}$

Attack 6: Type Attack On Protocol 1 Using Sequence Numbers

Another security issue is related with message retransmission. For example, when user B sends message (1) with its latest sequence number to the server S, S checks if the sequence number in message (1) equals to its local copy of the sequence number of B and if the check succeeds, S updates its local copy by increasing one and replies with message (2). However it is possible that message (2) is lost during the transmission. It is a common practice that B will retransmit message (1). Now S has two strategies to process this retransmitted message, either accepting the message whose sequence number is only one less than the valid one or securely sending an error message to inform the sender. If the former strategy is used, the replay attack becomes feasible. However the threat of the replay attack can be reduced if the server only accepts the retransmission in a short period immediately after the sequence number is updated.