** Yahalom Protocol Tamarin Script (without example)

The following is an explanation of the Yahalom protocol. Write this protocol as a Tamarin Prover script:
Yahalom is an authentication and secure key-sharing protocol designed for use on an insecure network such as the Internet. Yahalom uses a trusted arbitrator to distribute a shared key between two people. This protocol can be considered as an improved version of Wide Mouth Frog protocol (with additional protection against man-in-the-middle attack), but less secure than the Needham–Schroeder protocol.

Protocol description
If Alice (A) initiates the communication to Bob (B) with S is a server trusted by both parties, the protocol can be specified as follows using security protocol notation:

* A and B are identities of Alice and Bob respectively
* Kas is a symmetric key known only to A and S
* Kbs is a symmetric key known only to B and S
* Na and Nb are are nonces generated by A and B respectively
* Kab is a symmetric, generated key, which will be the session key of the session between A and B

A->B: A, Na
  Alice sends a message to Bob requesting communication.
B->S: {A, Na, Nb}Kbs
  Bob sends a message to the Server encrypted under Kbs
S->A: {B, Kab, Na, Nb}Kas, {A, Kab}Kbs
  The Server sends to Alice a message containing the generated session key Kab and a message to be forwarded to Bob.
A->B: {A, Kab}Kbs, {Nb}Kab
  Alice forwards the message to Bob and verifies Na has not changed. Bob will verify Nb has not changed when he receives the message.

メモリを更新しました

Below is a Tamarin Prover script for the Yahalom protocol. It follows the described structure and includes the necessary facts and actions for Alice (A), Bob (B), and the Server (S). I've ensured it aligns with your preferences for syntax and structure in Tamarin Prover.

```
tamarin


  // Yahalom Protocol in Tamarin Prover

  // Define Facts
```

```
free A, B, S : Agent   // Participants
free Kas, Kbs, Kab : Key // Symmetric keys
free Na, Nb : Nonce     // Nonces
free Msg : Message      // Generic message
builtins: symmetric-encryption

// Define the knowledge of the participants
Knowledge(A) := {A, Kas}.
Knowledge(B) := {B, Kbs}.
Knowledge(S) := {S, Kas, Kbs}.
Knowledge(Adv) := {A, B, S}.

rule Alice_Init:
  [ Fr(Na) ]
  --[ SendMessage('A_to_B', A, Na) ]->
  [ Out(<A, Na>),
    State_A(A, Na)
  ]

rule Bob_Receives_Alice:
  [ In(<A, Na>), Fr(Nb), LongtermKey(B, Kbs) ]
  --[ SendMessage('B_to_S', A, Na, Nb) ]->
  [ Out(senc(<A, Na, Nb>, Kbs)),
    State_B(B, A, Na, Nb)
  ]

rule Server_Responds:
  [ In(senc(<A, Na, Nb>, Kbs)), LongtermKey(A, Kas), LongtermKey(B, Kbs), Fr(Kab) ]
  --[ SendMessage('S_to_A', B, Kab, Na, Nb),
      SendMessage('S_to_B', A, Kab) ]->
  [ Out(< senc(<B, Kab, Na, Nb>, Kas),
         senc(<A, Kab>, Kbs) >),
    State_S(S, A, B, Na, Nb, Kab)
  ]

rule Alice_Receives_Server:
  [ In(<senc(<B, Kab, Na, Nb>, Kas), senc(<A, Kab>, Kbs)), State_A(A, Na), LongtermKey(A,
Kas) ]
  --[ SendMessage('A_to_B', A, Kab, Nb) ]->
  [ Out(senc(<A, Kab>, Kbs)),
    Out(senc(Nb, Kab)),
    State_A2(A, B, Kab, Nb)
  ]

rule Bob_Verifies_And_Completes:
  [ In(senc(<A, Kab>, Kbs), senc(Nb, Kab)), State_B(B, A, Na, Nb), LongtermKey(B, Kbs) ]
  --[ ReceiveMessage('B_Verified') ]->
  [ State_Complete(B, A, Kab)
  ]

// Security properties
lemma secrecy_of_Kab:
  "not (Ex A B Kab. Secret(A, B, Kab) & attacker(Kab))"

lemma authentication_A_to_B:
  "All A B Kab. (State_Complete(B, A, Kab) ==> Ex Na Nb. State_A2(A, B, Kab, Nb))"
```

Script Explanation:

1. **Roles and Messages**:
   - **`` `Alice_Init` ``**: Alice generates a nonce (**`` `Na` ``**) and sends it to Bob.
   - **`` `Bob_Receives_Alice` ``**: Bob receives Alice's message, generates his nonce (**`` `Nb` ``**), and forwards both nonces to the server (**`` `S` ``**).
   - **`` `Server_Responds` ``**: The server generates the session key (**`` `Kab` ``**), encrypts relevant parts for Alice and Bob, and sends them back.

    - `Alice_Receives_Server`: Alice receives the server's response and forwards the encrypted session key and `Nb` to Bob.
    - `Bob_Verifies_And_Completes`: Bob verifies the session key and the nonce `Nb` to ensure the integrity of the communication.

2. **Secrecy and Authentication**:
    - The `secrecy_of_Kab` lemma ensures the session key `Kab` is not compromised by the attacker.
    - The `authentication_A_to_B` lemma ensures that if Bob completes the protocol, Alice also participated correctly with matching session keys.

This script models the essential steps of the Yahalom protocol while adhering to Tamarin's logical framework. Let me know if you'd like to add or refine any details!

ChatGPT の回答は必ずしも正しいとは限りません。重要な情報は確認するようにしてください。