Diffie-Hellman with Forward Secrecy

AVISPA Formal Analysis

Marco Grisanti

Sommario

1. Problema

AVISPA (Automated Validation of Internet Security Protocols and Applications) è un software che permette di analizzare in modo semplice protocolli di sicurezza con lo scopo di trovare eventuali attacchi cosicché tali protocolli di sicurezza possano essere, se possibile, rafforzati per soddisfare le proprietà di sicurezza richieste.

AVISPA permette di scrivere un protocollo di sicurezza nel linguaggio HLPSL (High−Level Protocol Specification Language), il quale verrà tradotto grazie al "traduttore" HLPSL2IF nel linguaggio IF (Intermediate Format) che può essere dato in input ad uno dei back-end messi a dispozione per AVISPA:

- OFMC - On the Fly Model Checker
- ATSE – CL Based Attack Searcher
- SATMC – SAT Based Model Checker
- TA4SP - Tree Automata Based Protocol Analyzer

Ciascuno dei sopra elencati back-end di AVISPA fornisce una propria interpretazione del protocollo di sicurezza da analizzare del quale verrà restituito un breve sommario che indicherà se il protocollo è sicuro ("SAFE") oppure non sicuro ("UNSAFE") mostrando in quest'ultimo caso l'attacco trovato.

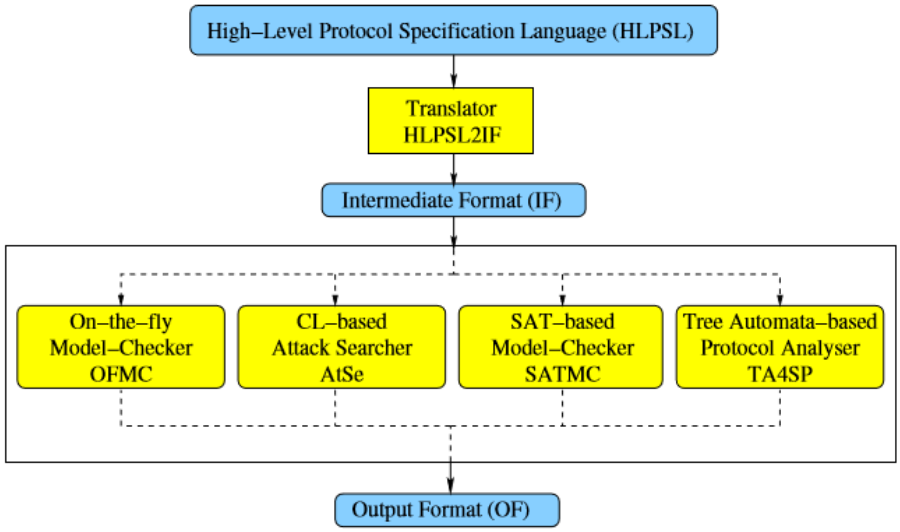

Immagine 1 - Architettura di AVISPA

Dunque, per introdurre AVISPA è sufficiente studiare il linguaggio HLPSL, i cui principi di base per analizzare qualche protocollo di sicurezza "giocattolo" verranno dati nelle prossime pagine di questa trattazione.

2. Lavoro Svolto

Il protocollo che si vuole analizzare è un protocollo simile a quello creato da Whitfield Diffie e Martin Hellman, il quale, a differenza dell'originale, garantisce la *forward secrecy*.

Un protocollo relativo allo scambio di una chiave di sessione garantisce forward secrecy se la compromissione delle chiavi a lungo termine dei partecipanti al protocollo di non compromette la chiave di sessione stabilita in una precedente esecuzione del protocollo dove tali partecipanti sono stati coinvolti.

Un protocollo relativo allo scambio di una chiave di sessione garantisce partial forward secrecy se la compromissione delle chiavi a lungo termine di uno o più partecipanti al protocollo non compromette la chiave di sessione stabilita in una precedente esecuzione del protocollo dove tali partecipanti sono stati coinvolti.

Di seguito è riportato nel dettaglio il protocollo Diffie-Hellman che garantisce la forward secrecy:

$$1. A \rightarrow S: A, B$$
$$2. A \rightarrow B: A, g^{N_A}$$
$$3. S \rightarrow B: \{A, B, K_{AB}\}_{K_{BS}}$$
$$4. S \rightarrow A: \{A, B, K_{AB}\}_{K_{AS}}$$
$$5. B \rightarrow A: A, g^{N_B}$$

Dove:

- A è l'agente che inizia il protocollo di sicurezza.
- B è l'agente con il quale A vuole scambiare una chiave di sessione.
- S è la terza parte fidata che condivide con A e B due chiavi a lungo termine, rispettivamente K_{AS} e K_{BS}.
- K_{AB} è una chiave che genera S e che viene consegnata sia ad A che a B.
- N_A ed N_B sono due nonce che generano rispettivamente A e B.
- g è un generatore di un qualche gruppo moltiplicativo.

La chiave di sessione K può essere calcolata sia da A che da B nel modo seguente:

$$K = (g^{N_B})^{N_A K_{AB}} \text{ per } A$$

$$K = (g^{N_A})^{N_B K_{AB}} \text{ per } B$$

Si noti che $(g^{N_B})^{N_A K_{AB}} = (g^{N_A})^{N_B K_{AB}}$.

Dunque, la forward secrecy è garantita nel caso in cui l'attaccante conosca K_{AS} e K_{BS} e non riesca a calcolare K.

Per iniziare, è necessario scaricare AVISPA, il quale verrà scaricato insieme a SPAN (Secure Protocol Animator) che permetterà di visualizzare il risultato di AVISPA in una forma più intuitiva grazie all'aiuto di un'interfaccia grafica che permette di simulare il comportamento del protocollo di sicurezza e di vedere in modo chiaro un eventuale attacco. Una macchina virtuale con Ubuntu 10.10 sul quale sono installati sia AVISPA che SPAN è possibile scaricarla dal link presente nella bibliografia di questa trattazione. Una volta riuscito a formalizzare il protocollo Diffie-Hellman che assicura la forward secrecy si verificherà tale proprietà in modo da mostrare il comportamento di AVISPA nel caso di un protocollo "SAFE" relativamente ad una data proprietà.

2.1 Il protocollo Diffie-Hellman con la Forward Secrecy in HLPSL

Si definisce ruolo un processo indipendente all'interno di un protocollo, ovvero, semplificando, un ruolo rappresenta ciò che deve fare un agente che partecipa al protocollo.

```
role roleName(Var1:typeOfVar1, Var2:typeOfVar2, ..., VarN:typeOfVarN)
played_by AgentName
def=
        local
                % Local Variables
        init
                % Local Variables To Initialize
        transition
                Transition1: Precondition =|> Postcondition
                Transition2: Precondition =|> Postcondition
                ...
                TransitionM: Precondition =|> Postcondition

end role
```

Immagine 2 - Struttura di un Ruolo

Le N variabili che vengono passate in input rappresentano la conoscenza di AgentName, ovvero dell'agente che interpreta questo ruolo. Successivamente AgentName crea, se necessario, nella sezione *local,* delle sue variabili le quali possono essere inizializzate nella sezione *init*. Infine, si scrivono le transizioni che rappresentano i passi del protocollo riguardanti AgentName nella sezione *transition*. Ogni transizione è formata da una precondizione e da una postcondizione.

Prima di passare a descrivere i ruoli del protocollo Diffie-Hellman con la forward secrecy, si noti che per dimostrare tale proprietà non si devono formalizzare tutti i passi del protocollo, ma si danno le informazioni (dalle quali si può ricavare K) ad A e B in modo implicito. Ovvero, per esempio, nella formalizzazione non si dirà che A invia a B il messaggio g^{N_A}, ma quest'ultimo sarà già conosciuto a priori da B. Ciò si deve realizzare per impedire all'attaccante che possa in qualche modo disturbare la comunicazione tra A e B grazie alla quale tali agenti possono calcolare K. Infatti, dalla definizione di forward secrecy, si evince in modo chiaro che la chiave di sessione deve essere stata stabilita da A e B in una precedente esecuzione del protocollo.

Dunque, il protocollo da formalizzare, per verificare la forward secrecy, diventa il seguente:

1. $A\ conosce\ \{A, B, N_A, N_B, g, g^{N_A}, g^{N_B}, K_{AB}, K_{AS}\}$
2. $B\ conosce\ \{A, B, N_A, N_B, g, g^{N_A}, g^{N_B}, K_{AB}, K_{BS}\}$
3. $L'attaccante\ conosce\ \{A, B, g\}$
4. $A\ calcola\ K = (g^{N_B})^{N_A K_{AB}}$
5. $B\ calcola\ K = (g^{N_A})^{N_B K_{AB}}$
6. $A\ invia\ nella\ rete: g^{N_A}, \{A, B, K_{AB}\}_{K_{AS}}, K_{AS}$
7. $B\ invia\ nella\ rete: g^{N_B}, \{A, B, K_{AB}\}_{K_{BS}}, K_{BS}$

Osservazioni:

- In realtà, A non conosce N_B così come B non conosce N_A, ma supporre ciò semplifica la formalizzazione del protocollo in questione e non è significante per un eventuale attacco relativo alla forward secrecy.

- A e B inviano nella rete K_{AS} e K_{BS} per simulare la perdita delle chiavi a lungo termine così come scritto nella definizione di forward secrecy. Inoltre, inviano nella rete anche gli altri messaggi che, nella implicita sessione del protocollo svolta in precedenza, l'attaccante può aver intercettato.

Adesso, è possibile cominciare a studiare quanto appena descritto nel linguaggio HLPSL.

Il ruolo di A si formalizza nel modo seguente:

```
role role_A(A:agent,
            B:agent,
            S:agent,
            G:nat,
            Na:nat,
            Nb:nat,
            Kab: symmetric_key,
            Kas: symmetric_key,
            SND:channel(dy),
            RCV:channel(dy))
played_by A
def=
        local
                State:nat,
                K:symmetric_key
        init
                State := 0
        transition
                1. State = 0
                   /\ RCV(start)
                   =|>
                   State' := 1
                   /\ K' := exp(exp(exp(G, Nb), Na), Kab)
                   /\ secret(K', secretAB, {A, B})
                   /\ SND({A.B.Kab}_Kas)
                   /\ SND(A.exp(G, Na))
                   /\ SND(Kas)

end role
```

Immagine 3 - Ruolo di A in HLPSL

Osservazioni:

- SND e RCV sono i canali di comunicazione che l'agente che interpreta il ruolo in questione usa per inviare e ricevere messaggi dalla rete. Il parametro dy indica che nel canale è presente un attaccante che rispecchia il modello d'attaccante Dolev-Yao.
- $State$ è una variabile locale del ruolo che serve a dare un ordine alle transizioni.
- $RCV(start)$ serve ad avviare il protocollo.
- L'apice serve per gli assegnamenti e per distinguere un valore nuovo o vecchio di una certa variabile x in una data transizione. Ovvero, per riferirsi al nuovo valore si scrive x', per riferirsi al vecchio valore si scrive x.

- $secret(K', secretAB, \{A, B\})$ significa che il valore del termine K' deve essere un segreto condiviso solo tra gli agenti A e B. $secretAB$ è il nome assegnato alla proprietà di sicurezza appena spiegata.
- Il punto è l'operatore di concatenazione dei messaggi.
- Nelle transizioni, ciò che viene messo come parametro in SND viene inviato nella rete.

In modo analogo si formalizza il ruolo di B:

```
role role_B(B:agent,
            A:agent,
            S:agent,
            G:nat,
            Na:nat,
            Nb:nat,
            Kab:symmetric_key,
            Kbs:symmetric_key,
            SND:channel(dy),
            RCV:channel(dy))
played_by B
def=
        local
                State:nat,
                K:symmetric_key
        init
                State := 0
        transition
                1. State = 0
                   /\ RCV(start)
                   =|>
                   State' := 1
                   /\ K' := exp(exp(exp(G, Na), Nb), Kab)
                   /\ secret(K', secretAB, {A, B})
                   /\ SND({A.B.Kab}_Kbs)
                   /\ SND(B.exp(G, Nb))
                   /\ SND(Kbs)

end role
```

Immagine 4 - Ruolo di B in HLPSL

Proseguendo, è necessario definire un ruolo particolare, chiamato $session$, il cui compito è quello di definire i vari parametri di sessione da passare ai ruoli di A e B e di creare una sessione del protocollo concatenando i suddetti ruoli.

```
role session(A:agent, B:agent, S:agent, G:nat, Kas:symmetric_key, Kbs:symmetric_key)
def=
        local
                SND1:channel(dy),
                RCV1:channel(dy),
                SND2:channel(dy),
                RCV2:channel(dy),
                Na:nat,
                Nb:nat,
                Kab:symmetric_key
        composition
                role_A(A, B, S, G, Na, Nb, Kab, Kas, SND1, RCV1)
                /\ role_B(B, A, S, G, Na, Nb, Kab, Kbs, SND2, RCV2)
end role
```

Immagine 5 - Ruolo "session" in HLPSL

L'ultimo ruolo da definire è $environment$ nel quale si definiscono le costanti (ovvero quei parametri che non variano al variare delle sessioni, come, ad esempio, le chiavi a lungo termine oppure i nomi degli agenti), la conoscenza iniziale dell'attaccante e le sessioni da avviare.

```
role environment()
def=
        const
                a:agent,
                b:agent,
                s:agent,
                g:nat,
                kas:symmetric_key,
                kbs:symmetric_key,
                secretAB:protocol_id
        intruder_knowledge = {a, b, g}
        composition
                session(a, b, s, g, kas, kbs)
end role
```

Immagine 6 - Ruolo "environment" in HLPSL

Si noti che $secretAB{:}\,protocol\_id$ serve semplicemente a specificare che $secretAB$ è un goal del protocollo.

Infine, si definiscono gli obiettivi di sicurezza (ovvero, i goal) del protocollo:

```
goal

        secrecy_of secretAB

end goal
```

Immagine 7 - Obiettivi di sicurezza del protocollo in HLPSL

Caricando il file HLPSL su AVISPA ed eseguendo il protocollo si ottiene il seguente risultato:

```
SUMMARY
 SAFE

DETAILS
 BOUNDED_NUMBER_OF_SESSIONS
 TYPED_MODEL

PROTOCOL
 /home/span/span/testsuite/results/DH with Forward Secrecy-.if

GOAL
 As Specified

BACKEND
 CL-AtSe
```

Immagine 8 - Output di AVISPA relativamente al protocollo Diffie-Hellman con Forward Secrecy

Come si vede, il protocollo risulta *SAFE* in base agli obiettivi di sicurezza specificati.

Per verificare che il protocollo sia stato formalizzato correttamente, si avvia lo "Intruder Simulator" grazie al quale, tra le tante features, è possibile vedere i messaggi che arrivano all'attaccante.

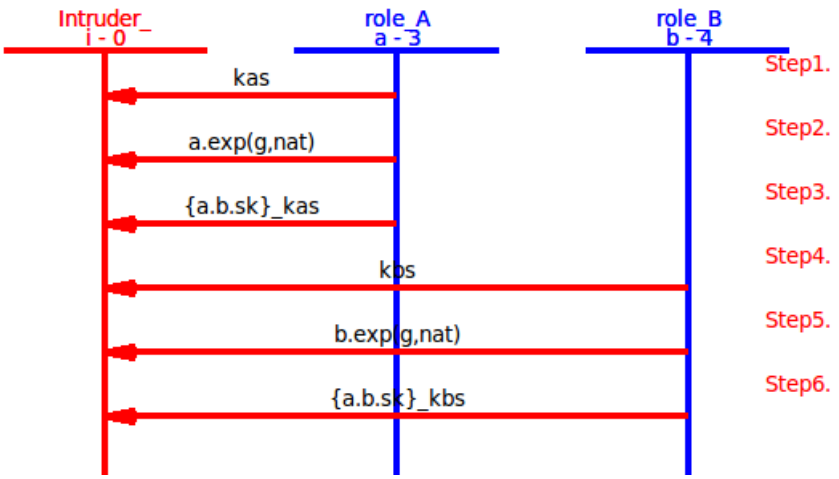

Immagine 9 - Intruder Simulation per il protocollo Diffie-Hellman con Forward Secrecy

Per fini didattici, si modifica leggermente il protocollo in questione per analizzare come viene mostrato un attacco all'utente. Si aggiunge $SND(N_A)$ al ruolo di A. In questo modo, l'attaccante conoscerà (per assurdo, poichè ciò è equivalente a calcolare il logaritmo discreto) anche N_A. Ripetendo i passaggi precedenti, si ottiene:

```
SUMMARY
  UNSAFE

DETAILS
  ATTACK_FOUND
  TYPED_MODEL

PROTOCOL
  /home/span/span/testsuite/results/DH with Forward Secrecy-.if

GOAL
  Secrecy attack on (exp(g,dummy_sk*dummy_nat*dummy_nat))

BACKEND
  CL-AtSe
```

Immagine 10 - Output di AVISPA relativamente al protocollo modificato Diffie-Hellman con Forward Secrecy

Come si vede, il protocollo risulta *UNSAFE* in base agli obiettivi di sicurezza specificati.

Cliccando su "Attack Simulation" si vede l'attacco:

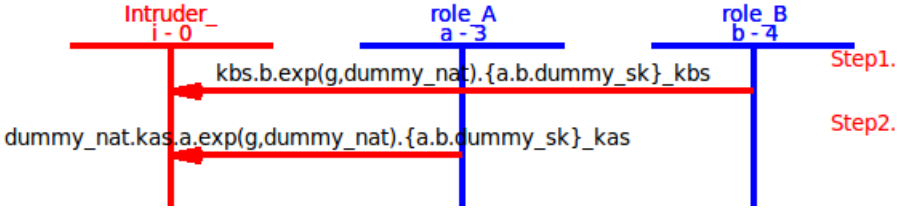

Immagine 11 - Attacco al protocollo modificato Diffie-Hellman con Forward Secrecy

Questa immagine mostra che l'attaccante riesce a calcolare K dopo che ottiene le varie informazioni dalla rete.

Cliccando su "Compose Knowledge" si può vedere nel dettaglio come l'attaccante costruisce K:

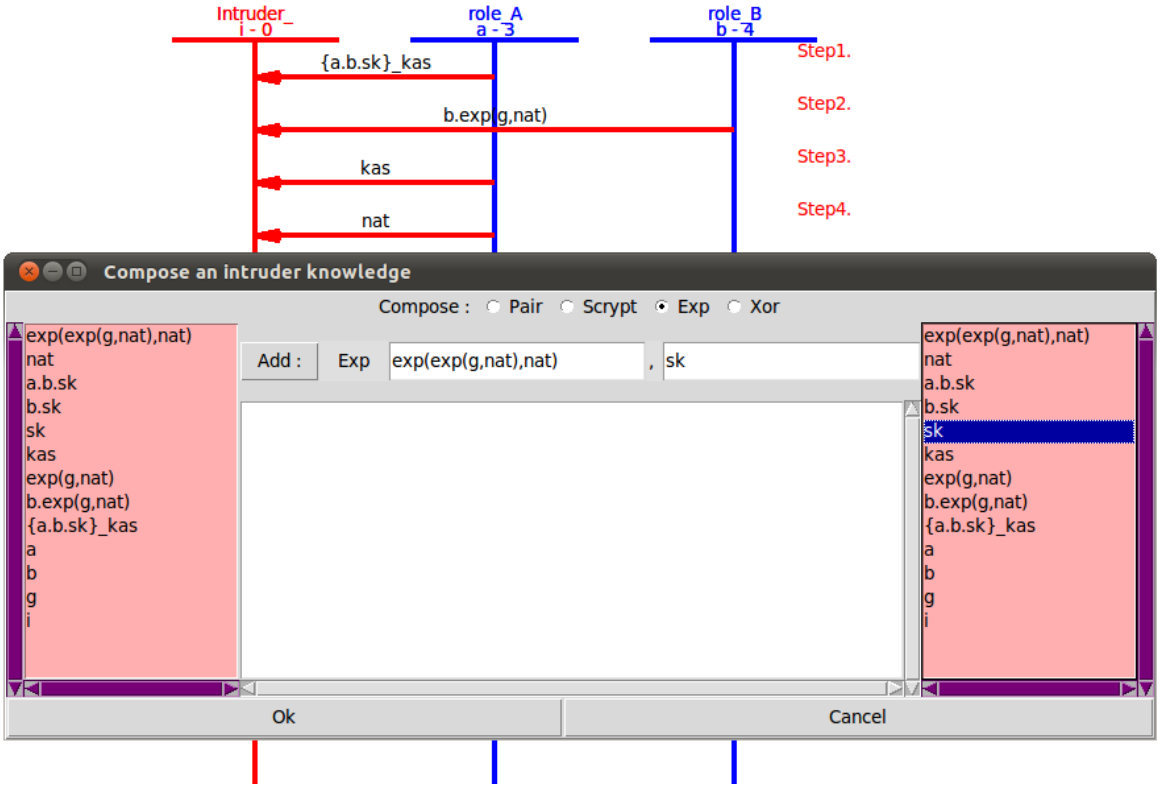

Immagine 12 - Composizione della conoscenza dell'attaccante nel protocollo modificato Diffie-Hellman con Forward Secrecy

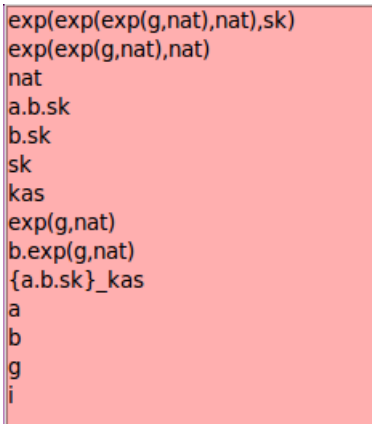

Immagine 13 - Conoscenza dell'attaccante

3. Conclusioni

Grazie all'utilizzo di AVISPA si è riusciti a dimostrare che il protocollo Diffie-Hellman che garantisce la forward secrecy è sicuro in senso a tale proprietà. Inoltre, si ha "giocato" un po' con SPAN per comprendere bene come è possibile per un eventuale attaccante, nel protocollo Diffie-Hellman modificato, violare la forward secrecy ricavando la chiave di sessione K dopo che sono state rilasciate le chiavi a lungo termine K_{AS} e K_{BS} che avevano concordato gli agenti A e B con la terza parte fidata S.

Un punto di forza di AVISPA è sicuramente la sua semplicità d'uso. Infatti, una volta imparata la sintassi HLPSL (il cui studio ha impiegato la maggior parte del tempo dedicato a questo progetto), è abbastanza semplice formalizzare i vari protocolli (perlomeno i più elementari).

Purtroppo, si è notato che AVISPA richiede grandi risorse computazionali rendendolo, forse, impossibile da usare per analizzare protocolli di sicurezza più avanzati con un comune computer.

4. Bibliografia

- Download della macchina virtuale con Ubuntu 10.10. contenente AVISPA e SPAN: http://people.irisa.fr/Thomas.Genet/span/span_on_ubuntu10.ova
- Pagina web ufficiale di AVISPA: http://www.avispa-project.org/
- Manuale per l'utente di AVISPA: http://www.avispa-project.org/package/user-manual.pdf