Symbolická analýza kryptografických protokolov

Martin Stanek

2025

KI FMFI UK Bratislava

Úvod

- bezpečnostné ciele kryptografických protokolov
  - dôvernosť správ, kľúčov spojenia
  - autentifikácia účastníkov
  - dopredná bezpečnosť (*forward secrecy*)
  - čerstvosť správ,
  - neodlíšiteľnosť behov (anonymita), …
- intuícia pre bezpečnosť nestačí (NSPK)
- formálna analýza zvyšuje dôveru v protokol
  - verifikácia bezpečnosti v danom modeli
- automatizácia analýzy: rýchlosť a menej chýb

Cieľ: jemný úvod do symbolickej analýzy protokolov

NSPK

1. $A \to B: \{A, N_A\}_{K_B}$
2. $B \to A: \{N_A, N_B\}_{K_A}$
3. $A \to B: \{N_B\}_{K_B}$

- vzájomná autentizácia účastníkov A a B
- šifrovanie s verejnými kľúčmi K_A a K_B
- príležitostné slová N_A a N_B
- známy problém začína $A \to E: \{A, N_A\}_{K_E}$ …

Symbolická analýza

- prakticky používaná pri návrhu a analýze protokolov: TLS, Wireguard, Signal, WPA2, …
- odlišnosti nástrojov
  - jazyk pre popis a modelovanie protokolu
  - možnosť modelovať globálny stav
  - aké vlastnosti je možné dokazovať
  - obmedzený/neobmedzený počet inštancií
  - modelovanie rovností (grupy, xor, …)
  - rýchlosť a zaručený dobeh
  - úplnosť: bez falošných útokov
- najznámejšie nástroje: Tamarin, ProVerif
  - iné: Verifpal, Deepsec, Scyther, OFMC

Symbolický model

- kryptografické algoritmy (primitíva) sú „čierna skrinka“
  - symetrické šifrovanie, MAC, …
- správy sú termy nad primitívami
- predpoklad perfektnej kryptografie
  - útočník obmedzený len na definované vlastnosti primitív
- vysokoúrovňová analýza

Iný prístup: výpočtový model

- CryptoVerif, Squirrel prover

Verifpal

Hlavné ciele

- intuitívny jazyk pre popis protokolu
  - účastníci, predpoklady, správy, vlastnosti
- prevencia používateľských chýb
  - nemožnosť vytvárať vlastné primitíva
- ľahko interpretovateľný výstup analýzy

„Menej dôležité“

- flexibilita a expresivita jazyka
- verifikačné techniky

dostupné: https://verifpal.com/

Vybrané primitíva

- symetrické šifrovanie/dešifrovanie
  - `ENC(key, p), DEC(key, c)`
  - `AEAD_ENC(key, p, ad)`
- asymetrické šifrovanie/dešifrovanie
  - `PKE_ENC(G^key, p)`
  - `PKE_DEC(key, p)`
- podpisovanie a overovanie podpisu
  - `SIGN(key, m)`
  - `SIGNVERIF(G^key, m, sig)`
- skladanie a rozoberanie správ
  - `CONCAT(a, b, ...)`
  - `SPLIT(con)`

Deklarácie: útočník a účastníci

Útočník

- aktívny: `attacker[active]`
- pasívny: `attacker[passive]`

Účastník

- niečo pozná vopred (`knows`)
  - súkromné: napr. súkromný kľúč, správa
  - verejné: známe komukoľvek, aj útočníkovi
- generované hodnoty (`generates`)
  - garantuje čerstvosť
- priradenia/výpočty
- testovanie zhody, overovanie podpisov a pod.

DH protokol (začiatok)

```
attacker[active]

principal Alice [
  generates a
  ga = G^a
]

Alice -> Bob: ga
```

- zoznam prenášaných konštánt (bez výrazov)
- chránené konštanty `[x]`
  - aktívny útočník dokáže čítať ale nie meniť
  - modeluje autentizáciu napr. verejného kľúča, uskutočnenú vopred

DH protokol (pokračovanie)

```
principal Bob [
  generates b
  gb = G^b
  gab = ga^b
]

Bob -> Alice: gb

principal Alice [
  gba = gb^a
]
```

Vlastnosti

- otázky na bezpečnostné vlastnosti (overované ciele protokolu)
- dôvernosť (`confidentiality? m`)
  - útočník sa nedostane k `m`
- autentickosť (`authentication? A -> B: c`)
  - `B` si môže byť istý, že `c` je od `A`
- čerstvosť (`freshness? h`)
  - útočník nedokáže vynútiť rovnaké `h` v rôznych behoch protokolu
- nelinkovateľnosť, ekvivalencia, …

DH protokol (záver)

```
queries [
  confidentiality? gab
  confidentiality? gba
]
```

Výsledok verifikácie (skrátené)

– žiadny cieľ nie je naplnený

```
Verifpal • Summary of failed queries will follow.
Result • confidentiality? gab — When:
    gb → G^nil ← mutated by Attacker (originally G^b)
    gab → G^a^b
    gba → G^nil^a

    G^nil^b is obtained:
    ga → G^nil ← mutated by Attacker (originally G^a)
    gab → G^nil^b ← obtained by Attacker
    gba → G^b^a
    gab (G^nil^b) is obtained by Attacker.
```

Modelovanie

- distribúcia verejných kľúčov ≈ chránené správy
- identifikátory účastníkov ≈ verejné kľúče
- čerstvosť ≈ rôzna hodnota v rôznych behoch
- NSPK:
  - nemožné overiť autentickosť na základe `PKE_DEC`
  - problematické modelovanie a verifikácia

Príklad

Challenge-Response protokol
(výmena podpísaných výziev)

1. $C \rightarrow S: N_C$
2. $S \rightarrow C: \mathrm{sig}_S(N_C), N_S$
3. $C \rightarrow S: \mathrm{sig}_C(N_S)$

Ako sa verifikuje

- prehľadávanie priestoru behov protokolu
- modelovanie stavov účastníkov a útočníka
- hľadanie sporu s deklarovanými vlastnosťami (`queries`)
- prevencia „explózie“ stavu (napr. nekonečné vnáranie primitív)

5 fáz (pokiaľ útočník získava nové hodnoty)

1. zber hodnôt (všetko poslané)
2. vložiť hodnoty do stavu útočníka
3. aplikovať transformácie
   - rezolúcia, rekonštrukcia, dekonštrukcia, stotožnenie hodnôt
4. mutácie pre ďalšie spojenie
   - nahradenie hodnôt inými útočníkovi známymi hodnotami
5. iterácia cez jednotlivé mutácie

Úlohy – experimenty s DH a Verifpal

1. Overte, že Verifpal nenájde útok ak
   - označíme správy v DH protokole ako chránené pomocou `[]`
   - zmeníme útočníka na pasívneho
2. Doplňte do pôvodného protokolu správu `m`, ktorú `Alice` bude chcieť poslať šifrovanú pomocou symetrickej AEAD šifry, kde kľúč je `HASH` výstupu DH protokolu.
   - sformulujte ciele protokolu ako dôvernosť `m` a autentickosť šifrového textu
   - analyzujte protokol a pozrite popis identifikovaných problémov
3. Implementujte a analyzujte úpravu DH protokolu z predchádzajúcej úlohy, kde `Bob` podpíše svoju DH správu. `Alice` pozná/získa zodpovedajúci verejný kľúč vopred.
   - preskúmajte výsledky analýzy

ProVerif

ProVerif

- neobmedzený počet inštancií protokolu (aj paralelných) a neobmedzený priestor správ
  - vo všeobecnosti je utajenie (*secrecy*) v protokoloch nerozhodnuteľný problém
- nie vždy dobehne (obvykle je však analýza rýchla)
- potenciálne nájde falošné útoky (zriedka)
- ak vlastnosť overí, tak naozaj platí
- vlastnosti, ktoré ProVerif overuje:
  - utajenie, autentizácia, ekvivalencie, …
- π-kalkulus (formalizmus pre popis súbežných výpočtov) + kryptografické primitíva
  - preklad do Hornovych klauzúl, odvodzovanie vlastností

$P, Q ::=$	processes
0	null process
$P \mid Q$	parallel composition
$!P$	replication
$\mathbf{new}\ n : t;\ P$	name restriction
$\mathbf{in}(M, x : t);\ P$	message input
$\mathbf{out}(M, N);\ P$	message output
$\mathbf{if}\ M\ \mathbf{then}\ P\ \mathbf{else}\ Q$	conditional
$\mathbf{let}\ x = M\ \mathbf{in}\ P\ \mathbf{else}\ Q$	term evaluation
$R(M_1, \ldots, M_k)$	macro usage

https://bblanche.gitlabpages.inria.fr/proverif/manual.pdf

Konštruktory, deštruktory a rovnosti

```
(* symetrická kryptografia; konštruktor a deštruktor *)
type key.
fun enc(key, bitstring): bitstring.
reduc forall x: key, y: bitstring ; dec(x, enc(x,y)) = y.

(* funkcie pre DH výpočet a súvisiaca rovnosť *)
type privkey.
type pubkey.
fun dh(privkey, pubkey): key.
fun pk(privkey): pubkey.
equation forall x: privkey, y: privkey ; dh(x,pk(y)) = dh(y,pk(x)).
```

Deklarácie a cieľ

```
free c : channel.                (* kanál *)
free s : bitstring [private].    (* tajná správa s *)

(* pasívny útočník *)
(* set attacker = passive. *)

(* Dozvie sa útočník s? *)
query attacker(s).
```

Príklad: špecifikácia procesu v DH protokole

```
let A = new a: privkey;
        out(c,pk(a));
        in(c,x1: pubkey);
        let k = dh(a,x1) in out(c,enc(k,s)).

let B = new b: privkey;
        out(c,pk(b));
        in(c,x0: pubkey);
        let k = dh(b,x0) in in(c,m: bitstring);
        let s2 = dec(k,m) in 0.

process A | B
```

Výstup analýzy (skrátené)

```
Starting query not attacker(s[])
goal reachable: attacker(s[])

Derivation:

1. The attacker has some term x.
attacker(x).

2. By 1, the attacker may know x.
Using the function pk the attacker may obtain pk(x).
attacker(pk(x)).
```

… (vynechané ďalšie kroky útoku a jeho sumarizácia)

```
A trace has been found.
RESULT not attacker(s[]) is false.
```