Ministerstvo financií
Slovenskej republiky

EFQM
Recognised for excellence
5 star - 2012

Protokoly pre autentifikáciu a dohodnutie kľúča

Richard Ostertág

KPMG
cutting through complexity™

Protokoly pre autentifikáciu a dohodnutie kľúča

Richard Ostertág

Obsah

- základné pojmy, označenia
- Dolevov-Yaov model útočníka
- princípy robustnosti kryptografických protokolov
  - explicitnosť, podpisovanie a šifrovanie, príležitostné slová, ochrana predpovedateľných údajov, časové pečiatky, ...
- demonštrácia vybraných princípov na protokoloch
  - Denningovej-Saccov, Otwayov-Reesov, Needhamov-Schroederov, Diffieho-Hellmanov, ...
- demonštrácia rôznych útokov
  - útok opakovaním, útočník uprostred, ...
- niektoré v praxi používané protokoly
  - Challenge-Handshake Authentication
  - Secure Shell

Kryptografické protokoly

- sú definované ako postupnosť krokov a výmen správ medzi dvoma alebo viacerými účastníkmi, s cieľom naplniť zvolené bezpečnostné požiadavky
  - obyčajne 2 až 5 správ
- využívajú pritom rôzne kryptografické konštrukcie

- veľmi ľahko sa v nich spraví chyba
  - ťažko sa hľadá (odhalenie trvá aj niekoľko rokov)
  - niečo nezvyčajné pre tak krátky „program“

Označenia pri popise protokolov

- účastníci protokolu
  - A – Alice
  - B – Bob
  - S alebo T – dôveryhodný (trusted) server
    - pôsobí ako sprostredkovateľ pri autentifikácii a distribúcii kľúča, pričom sa predpokladá, že účastníci serveru dôverujú
- útočníka označujeme ako
  - M – malicious, middle man
  - E – enemy, eavesdropper, evil
  - I – intruder
- asymetrické kľúče účastníka A:
  - verejný kľúč K_A
  - súkromný kľúč K_A^{-1}
- symetrický kľúč na komunikáciu medzi A a B: K_{AB}
- príležitostné slovo vytvorené účastníkom A: N_A
  - nonce – number used only once
- časová pečiatka vytvorená účastníkom A: T_A
- certifikát obsahujúci verejný kľuč účastníka A: CA

Označenia pri popise protokolov

- $A \rightarrow B : m$
  - účastník A posiela účastníkovi B správu m
- $A \rightarrow M\ (B) : m$
  - účastník A chce poslať B správu m, tú však zachytí útočník M
  - účastník B správu m vôbec nedostane
- $M(A) \rightarrow B : m$
  - útočník M posiela správu m účastníkovi B, vydávajúc sa za A
  - účastník A o správe m vôbec nevie
- $\{m\}_K$
  - správa m šifrovaná kľúčom K (symetrickým alebo asymetrickým)
- pri popise protokolov abstrahujeme od konkrétnych algoritmov
  - použité konštrukcie považujeme za ideálne bezpečné
  - v praxi bezpečnosť protokolov závisí aj na **vlastnostiach** a **spôsobe použitia** konkrétnych kryptografických konštrukcií

Typické použitie krypto. protokolov v IS:

- autentifikácia komunikujúcich účastníkov
  - každý účastník si overí, že komunikuje so želaným partnerom
- dohodnutie a distribúcia kryptografických kľúčov
  - zvyčajne symetrické, ktoré sa v následnej komunikácii použijú na
    - šifrovanie
    - výpočet autentifikačných kódov
    - na zabezpečenie iných bezpečnostných atribútov pomocou vhodných kryptografických konštrukcií
- kľúče spojenia (angl. session keys)
  - kryptografické kľúče dohodnuté v rámci protokolu
  - význam:
    - efektívnosti symetrickej kryptografie
    - logické oddelenie jednotlivých spojení medzi rovnakými účastníkmi
    - menší objem dát šifrovaných rovnakým kľúčom
- najčastejšie prakticky používané sú práve protokoly napĺňajúce vyššie uvedené požiadavky
  - napríklad SSL/TLS, IPsec, Kerberos, SSH, ...

Dolevov-Yaov model

- pri analýze bezpečnosti protokolov predpokladáme, že útočník má úplnú kontrolu nad sieťou
- t.j. útočník môže každú prenášanú správu:
  - zachytiť
  - zmeniť
  - zopakovať
  - poslať ľubovoľnému adresátovi
  - neposlať vôbec
- nie každý útočník v praxi má tieto možnosti
  - avšak protokoly bezpečné v silnom modeli budú bezpečné aj v modeli so slabším útočníkom

Princípy robustnosti kryptografických protokolov

1. explicitnosť
2. predpoklady
3. identita účastníka
4. použitie šifrovania
5. podpisovanie a šifrovanie
6. príležitostné slová
7. ochrana predpovedateľných údajov
8. časové pečiatky
9. čerstvosť vs. použitie
10. jednoznačnosť
11. dôvera

1. princíp: Explicitnosť

- význam správy má závisieť len na správe samotnej
  - t.j. správa má obsahovať všetky údaje potrebné pre jej interpretáciu
    - odosielateľ, príjemca, ...
- inak sa môže stať, že sa správa dá použiť namiesto inej

- dôsledkom tohto princípu je napríklad 3. princíp

2. princíp: Predpoklady

- pre každú správu (na základe ktorej sa má vykonať nejaká akcia), uviesť všetky vyžadované predpoklady
  - pomôže pri analýze protokolu
  - na nič sa nezabudne ...

príklad

- ak sa predpokladá, že kľúče generuje dôveryhodný server
  - potom sa nemôže použiť protokol WMF
    - kľúč je kompletne generovaný jedným z účastníkov

3. princíp: Identita účastníka

- ak je identita účastníka podstatná pre význam správy, je potrebné túto identitu v správe priamo uviesť
  - Denningovej-Saccov protokol

Denningovej-Saccov protokol

- protokol na výmenu kľúča
- postavený na asymetrickom kryptosystéme

1. $A \rightarrow S: A, B$
2. $S \rightarrow A: CA, CB$
3. $A \rightarrow B: CA, CB, \left\{\{K_{AB}, T_A\}_{K_A^{-1}}\right\}_{K_B}$

- nikto okrem B nemôže získať K_{AB} zo správy z kroku 3
  - lebo je zašifrovaná verejným kľúčom K_B
- B vie, že správa je od A
  - lebo je podpísaná A (pomocou súkromného kľúča)
- B vie, že správa bola zamýšľaná pre B
  - lebo je šifrovaná K_B a iba B ju vie dešifrovať

Denningovej-Saccov protokol – útok

1. $A \rightarrow S: A, B$
2. $S \rightarrow A: CA, CB$
3. $A \rightarrow B: CA, CB, \left\{\{K_{AB}, T_A\}_{K_A^{-1}}\right\}_{K_B}$

- obsah poslednej správy B dešifruje
- od S si vypýta CC a použitím K_C ho zase zašifruje

1. $B \rightarrow S: B, C$
2. $S \rightarrow B: CB, CC$
3. $B(A) \rightarrow C: CA, CC, \left\{\{K_{AB}, T_A\}_{K_A^{-1}}\right\}_{K_C}$

- v tomto momente si C myslí, že si dohodlo kľúč s A

Denningovej-Saccov protokol – oprava

- význam správy 3 je:
  - v čase T_A účastník A považuje kľúč K_{AB} za dôveryhodný kľúč pre komunikáciu medzi A a B
- preto treba túto informáciu explicitne alebo implicitne uviesť, napríklad:

1. $A \rightarrow S: A, B$
2. $S \rightarrow A: CA, CB$
3. $A \rightarrow B: CA, CB, \left\{\{A, B, K_{AB}, T_A\}_{K_A^{-1}}\right\}_{K_B}$

- meno účastníka A je možné vynechať, keďže jeho prítomnosť vyplýva z jeho podpisu

4. princíp: Použitie šifrovania

- pri použití šifrovania je potrebné uviesť, aký význam má šifrovanie konkrétneho textu
  - významy šifrovania (aj kombinácia):
    - zachovanie dôvernosti,
      - iba zamýšľaní príjemcovia poznajú dešifrovací kľúč
    - zaručenie autentickosti
      - iba správny odosielateľ vlastní použitý šifrovací kľúč
      - príklad: účastník podpíše časovú pečiatku
    - previazanie časti správ dokopy
      - prijatie správy $\{X, Y\}_K$ je niečo iné, ako prijatie $\{X\}_K$ a $\{Y\}_K$
      - na previazanie netreba šifrovať, stačí aj podpis
    - vytvorenie „náhodného“ čísla

5. princíp: Podpisovanie a šifrovanie 1/2

- digitálny podpis šifrovanej správy nezaručuje, že podpisujúci účastník pozná otvorený text správy
  - preto treba najskôr podpísať a potom šifrovať podpísanú správu
- 1. príklad
  - A, B, C poznajú svoje verejné kľúče
  - $A \rightarrow B: A, \{T_A, B, \{X\}_{K_B}\}_{K_A^{-1}}$
  - B vie, že správu poslalo A, lebo je ním podpísaná
  - X je dôverné, lebo je zašifrované kľúčom pre B
  - ale nič nezaručuje, že odosielateľ pozná X
- $C \rightarrow B: C, \{T_C, C, \{X\}_{K_B}\}_{K_C^{-1}}$
  - B si myslí, že C vie X, ale to nie je pravda

5. princíp: Podpisovanie a šifrovanie 2/2

- v kontexte tohto pravidla je hašovanie považované za šifrovanie
- 2. príklad
  - X je tajné heslo
  - H je jednosmerná hašovacia funkcia
    - aby sa minimalizovalo asymetrické šifrovanie
  - $A \rightarrow B: \{X\}_{K_B}, \{H(X)\}_{K_A^{-1}}$
  - B vie, že správu poslalo A, lebo podpísalo $H(X)$
  - X je dôverné, lebo je zašifrované kľúčom pre B
  - ale nič nezaručuje, že odosielateľ pozná X
- $C \rightarrow B: \{X\}_{K_B}, \{H(X)\}_{K_C^{-1}}$
  - B si myslí, že C vie X, ale to nie je pravda

6. princíp: Príležitostné slová

- pre každé príležitostné slovo je potrebné uviesť
  - význam použitia, teda či je použité na zaručenie:
    - časovej závislosti (čerstvosti) správ
      - že správa nie je zopakovaním nejakej starej správy
        - pošle sa správa, ktorá vedie k odpovedi, ktorá ale mohla byť poslaná iba za znalosti prvej správy
    - asociácie
  - a očakávané vlastnosti
    - zaručene nové
    - náhodné
    - nepredpovedateľné

- ochrana pred útokom opakovaním

Otwayov-Reesov protokol

- cieľom je distribúcia kľúča spojenia K_{AB} medzi A a B
  - s pomocou servera T, s ktorým A a B zdieľajú kľúče K_{AT} a K_{BT}
  - pričom kľúč K_{AB} generuje dôveryhodný server T
- N_A a N_B sú príležitostné čísla generované A a B
- M je identifikátor spojenia generovaný A
  - vždy nanovo pre každý beh protokolu

1. $A \rightarrow B: M, A, B, \{N_A, M, A, B\}_{K_{AT}}$
2. $B \rightarrow T: M, A, B, \{N_A, M, A, B\}_{K_{AT}}, \{N_B, M, A, B\}_{K_{BT}}$
3. $T \rightarrow B: M, \{N_A, K_{AB}\}_{K_{AT}}, \{N_B, K_{AB}\}_{K_{BT}}$
4. $B \rightarrow A: M, \{N_A, K_{AB}\}_{K_{AT}}$

- ak N_B zabezpečuje len čerstvosť, tak by sme ho v kroku 2 nemuseli šifrovať
  - pretože prítomnosť N_B v $\{N_B, K_{AB}\}_{K_{BT}}$ zaručuje, že išlo o následnú odpoveď, ktorá nie je zopakovaná

Otwayov-Reesov protokol – „optimalizovaný“

1. $A \rightarrow B: M, A, B, \{N_A, M, A, B\}_{K_{AT}}$
2. $B \rightarrow T: M, A, B, \{N_A, M, A, B\}_{K_{AT}}, N_B, \{M, A, B\}_{K_{BT}}$
3. $T \rightarrow B: M, \{N_A, K_{AB}\}_{K_{AT}}, \{N_B, K_{AB}\}_{K_{BT}}$
4. $B \rightarrow A: M, \{N_A, K_{AB}\}_{K_{AT}}$

- útok za predpokladu, že útočník E
  - ovláda pripojenie B do siete
  - z predošlej komunikácie s B má odložený fragment $\{M', E, B\}_{K_{BT}}$
  1. $E(A) \rightarrow B: M, A, B, \{N_E, M', E, B\}_{K_{ET}}$
  2. $B \rightarrow E(T): M, A, B, \{N_E, M', E, B\}_{K_{ET}}, N_B, \{M, A, B\}_{K_{BT}}$
  3. $E(B) \rightarrow T: M', E, B, \{N_E, M', E, B\}_{K_{ET}}, N_B, \{M', E, B\}_{K_{BT}}$
  4. $T \rightarrow E(B): M', \{N_E, K_{EB}\}_{K_{ET}}, \{N_B, K_{EB}\}_{K_{BT}}$
  5. $E(T) \rightarrow B: M, \{N_E, K_{EB}\}_{K_{ET}}, \{N_B, K_{EB}\}_{K_{BT}}$
  6. $B \rightarrow E(A): M, \{N_E, K_{EB}\}_{K_{ET}}$
- B verí, že zdieľa kľúč K_{EB} s A, ale zdieľa ho s E
  - server T musí ignorovať zopakovanie M', čo je očakávateľné

Otwayov-Reesov protokol – optimalizovaný

- v pôvodnom protokole N_A a N_B sú spojené s menami A a B pomocou šifrovania
  - preto môžu byť použité namiesto nich
- nahradíme ich priamo menom, čím ušetríme dve šifrovania
  - N_A a N_B v protokole stále ostanú ako dôkaz čerstvosti

1. $A \rightarrow B: A, B, N_A$
2. $B \rightarrow T: A, B, N_A, N_B$
3. $T \rightarrow B: \{N_A, A, B, K_{AB}\}_{K_{AT}}, \{N_B, A, B, K_{AB}\}_{K_{BT}}$
4. $B \rightarrow A: \{N_A, A, B, K_{AB}\}_{K_{AT}}$

- v popisoch protokolov abstrahujeme od detailov
  - použitých šifier, módov, dĺžok jednotlivých častí
- v ďalšom si na troch útokoch ukážeme, ako bezpečnosť záleží aj na takýchto implementačných detailoch

Otwayov-Reesov protokol – 1. útok

- vráťme sa k pôvodnému protokolu

1. $A \rightarrow B: M, A, B, \{N_A, M, A, B\}_{K_{AT}}$
2. $B \rightarrow T: M, A, B, \{N_A, M, A, B\}_{K_{AT}}, \{N_B, M, A, B\}_{K_{BT}}$
3. $T \rightarrow B: M, \{N_A, K_{AB}\}_{K_{AT}}, \{N_B, K_{AB}\}_{K_{BT}}$
4. $B \rightarrow A: M, \{N_A, K_{AB}\}_{K_{AT}}$

- predpokladajme, že náhodou platí: $|K_{AB}| = |M, A, B|$

1. $A \rightarrow E(B): M, A, B, \{N_A, M, A, B\}_{K_{AT}}$
4. $E(B) \rightarrow A: M, \{N_A, M, A, B\}_{K_{AT}}$

- E úspešne vnútilo A kľúč $K_{AB} = (M, A, B)$ na domnelú komunikáciu s B
  - bez ohľadu na mód šifry alebo jej kvalitu
  - riešenie: napr. doplnenie identifikátora čísla kroku do šifrových textov tak, aby sa nedali zamieňať správy medzi rôznymi krokmi

Otwayov-Reesov protokol – 2. útok

- Otwayov-Reesov protokol:

1. $A \rightarrow B: M, A, B, \{N_A, M, A, B\}_{K_{AT}}$
2. $B \rightarrow T: M, A, B, \{N_A, M, A, B\}_{K_{AT}}, \{N_B, M, A, B\}_{K_{BT}}$
3. $T \rightarrow B: M, \{N_A, K_{AB}\}_{K_{AT}}, \{N_B, K_{AB}\}_{K_{BT}}$
4. $B \rightarrow A: M, \{N_A, K_{AB}\}_{K_{AT}}$

- predpokladajme, že sa použije šifra v ECB móde a dĺžka N_A je náhodou násobkom dĺžky bloku
- útočník odpočuje starú komunikáciu medzi T a B a získa staré hodnoty $\{N'_A, K'_{AB}\}_{K_{AT}}, \{N'_B, K'_{AB}\}_{K_{BT}}$ a časom získa prístup ku K'_{AB}

1. $A \rightarrow B: M, A, B, \{N_A, M, A, B\}_{K_{AT}}$
2. $B \rightarrow E(T): M, A, B, \{N_A, M, A, B\}_{K_{AT}}, \{N_B, M, A, B\}_{K_{BT}}$
3. $E(T) \rightarrow B: M, \{N_A, K'_{AB}\}_{K_{AT}}, \{N_B, K'_{AB}\}_{K_{BT}}$
4. $B \rightarrow A: M, \{N_A, K'_{AB}\}_{K_{AT}}$

- E úspešne vnútilo A a B starý kľúč K'_{AB}
- riešenie: napríklad použiť šifru v CBC móde

Otwayov-Reesov protokol – 3. útok

- Otwayov-Reesov protokol:

1. $A \rightarrow B: M, A, B, \{N_A, M, A, B\}_{K_{AT}}$
2. $B \rightarrow T: M, A, B, \{N_A, M, A, B\}_{K_{AT}}, \{N_B, M, A, B\}_{K_{BT}}$
3. $T \rightarrow B: M, \{N_A, K_{AB}\}_{K_{AT}}, \{N_B, K_{AB}\}_{K_{BT}}$
4. $B \rightarrow A: M, \{N_A, K_{AB}\}_{K_{AT}}$

- predstavme si, že T nekontroluje zhodu identifikátorov účastníkov v šifrovom a otvorenom texte v 2. kroku protokolu

1. $A \rightarrow B: M, A, B, \{N_A, M, A, B\}_{K_{AT}}$
2. $B \rightarrow E(T): M, A, B, \{N_A, M, A, B\}_{K_{AT}}, \{N_B, M, A, B\}_{K_{BT}}$
3. $E \rightarrow T: M, A, E, \{N_A, M, A, B\}_{K_{AT}}, \{N_E, M, A, E\}_{K_{ET}}$
4. $T \rightarrow E: M, \{N_A, K_{AE}\}_{K_{AT}}, \{N_E, K_{AE}\}_{K_{ET}}$
5. $E(B) \rightarrow A: M, \{N_A, K_{AE}\}_{K_{AT}}$

- E dokáže presvedčiť A o tom, že komunikuje s B a navyše získa aj príslušný kľúč spojenia K_{AE}

7. princíp: Ochrana predpovedateľných údajov

- predpovedateľné údaje použité na zabezpečenie čerstvosti správ musia byť v protokole chránené
  - napríklad počítadlo
- protokol na vyžiadanie aktuálneho času T_S
  1. $A \rightarrow S: A, N_A$
  2. $S \rightarrow A: \{T_S, N_A\}_{K_{AS}}$
- útok pri predpovedateľnosti N_A:
  1. $E(A) \rightarrow S: A, N_A$
  2. $S \rightarrow E(A): \{T_S, N_A\}_{K_{AS}}$
     - použiteľné v budúcnosti ako odpoveď pre A
- oprava:
  1. $A \rightarrow S: A, \{N_A\}_{K_{AS}}$
  2. $S \rightarrow A: \{T_S, \{N_A\}_{K_{AS}}\}_{K_{AS}}$
     - N_A nemusí byť utajené, šifrovanie slúži iba na vytvorenie hodnoty, ktorú vie predpovedať iba A a S z hodnoty, ktorú vie predpovedať každý

8. princíp: Časové pečiatky

- ak sú použité časové pečiatky na zabezpečenie čerstvosti, je potrebné synchronizovať lokálne časy
  - navyše, systém „správy času“ sa stáva kritickým komponentom protokolu

- účastník s pomalými hodinami
  - môže považovať staré časové pečiatky za aktuálne
- účastník s rýchlymi hodinami
  - ak vytvorí správu s časovou pečiatkou z budúcnosti, útočník ju môže zopakovať, keď tento čas skutočne nastane

9. princíp: Čerstvosť vs. použitie

- aktuálne použitie entity (napr. kľúča na šifrovanie) nie je to isté ako čerstvosť entity
  - Needhamov-Schroederov protokol so symetrickým kľúčom

Needhamov-Schroederov SK protokol

- využíva symetrické šifrovacie algoritmy
  - SK – symmetric key (protokol so symetrickým kľúčom)
- štruktúrou aj cieľom podobný protokolu Kerberos
  - tvorí jeho základ

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

} key distribution

4. $B \rightarrow A: \{N_B\}_{K_{AB}}$
5. $A \rightarrow B: \{N_B + 1\}_{K_{AB}}$

} handshake

- v 5. kroku A posiela B hodnotu $N_B + 1$ aby A potvrdilo B, že
  - je prítomné a pozná kľúč K_{AB}
  - nejde o zopakovanie starej správy $\{K_{AB}, A\}_{K_{BS}}$
- nemôže poslať späť iba N_B
  - to by bola rovnaká správa akú dostalo A, stačilo by ju zopakovať
  - postačuje však ľubovoľná modifikácia, ktorá dokazuje že A pozná N_B

NSSK protokol – útok

- predpokladajme, že útočník E odpočuje komunikáciu (stačí krok 3) a časom získa aj kľúč K_{AB}, napríklad
  - útokom na počítač účastníka A alebo B
  - kryptoanalýzou
- jeho cieľom je vnútiť B starý kľúč

3. $E(A) \rightarrow B: \{K_{AB}, A\}_{K_{BS}}$
4. $B \rightarrow E(A): \{N_B\}_{K_{AB}}$
5. $E(A) \rightarrow B: \{N_B + 1\}_{K_{AB}}$

- v 5. kroku
  - E dešifruje $\{N_B\}_{K_{AB}}$ pomocou K_{AB}, čím získa N_B
  - pošle B hodnotu $N_B + 1$ zašifrovanú pomocou K_{AB}

NSSK protokol – oprava

- pridaním časovej pečiatky
  - vyžaduje synchronizáciu času servera a účastníkov

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

- v 3. kroku B akceptuje K_{AB}, ak $|T - T_S| < \Delta t_1 + \Delta t_2$, kde
  - T je aktuálny čas účastníka B
  - Δt_1 je maximálny povolený časový nesúlad medzi B a S
  - Δt_2 je očakávané oneskorenie na sieti medzi S, A a B

10. princíp: Jednoznačnosť

- správa protokolu je jednoznačne dekódovateľná – účastník je schopný určiť príslušnosť správy do protokolu, behu protokolu ako aj poradie správy v protokole

- Needhamov-Schroederov protokol

4. $B \rightarrow A: \{N_B\}_{K_{AB}}$ ⇨ $B \rightarrow A: \{NS4, N_B\}_{K_{AB}}$
5. $A \rightarrow B: \{N_B + 1\}_{K_{AB}}$ ⇨ $A \rightarrow B: \{NS5, N_B\}_{K_{AB}}$

11. princíp: Dôvera

- je potrebné uviesť a zdôvodniť oprávnenosť (ako aj vhodnosť) všetkých predpokladov o dôvere, ktoré protokol vyžaduje
- vo WMF protokole sa dôveruje účastníkovi A pri tvorbe kľúča spojenia
  - takáto dôvera je obyčajne neakceptovateľná, lebo A samé musí zabezpečiť „kvalitu“ kľúča
    - neopakovateľnosť
    - nepredpovedateľnosť
    - ...
- obyčajne sa predpokladá, že účastníci čestne plnia protokol, to ale nemusí byť vždy pravda:
  - znepriatelení účastníci

Wide mouthed frog protokol

- cieľom je preniesť kľúč spojenia K_{AB} vytvorený účastníkom A účastníkovi B cez dôveryhodný server T
  - používa len dve správy
  - používa symetrické šifrovanie
  - používa časové pečiatky na zaručenie čerstvosti správ
  - každý účastník X má dohodnutý kľúč K_{XT} so serverom T

1. $A \rightarrow T: A, \{T_A, B, K_{AB}\}_{K_{AT}}$
2. $T \rightarrow B: \{T_T, A, K_{AB}\}_{K_{BT}}$

Wide mouthed frog protokol – útok

- útočník využije rovnakú štruktúru správ na ich opakovanie a postupné obnovovanie vnorených časových pečiatok

1. $A \rightarrow T: A, \{T_A, B, K_{AB}\}_{K_{AT}}$
2. $T \rightarrow B: \{T_T, A, K_{AB}\}_{K_{BT}}$
3. $E(B) \rightarrow T: B, \{T_T, A, K_{AB}\}_{K_{BT}}$
4. $T \rightarrow E(A): \{T_T^1, B, K_{AB}\}_{K_{AT}}$
5. $E(A) \rightarrow T: A, \{T_T^1, B, K_{AB}\}_{K_{AT}}$
6. $T \rightarrow E(B): \{T_T^2, A, K_{AB}\}_{K_{BT}}$
   - postupné obnovovanie časovej pečiatky pred vypršaním jej platnosti opakovaním krokov 3 až 6
     - útočník preposiela šifrové texty bez toho, že by ich vedel dešifrovať
   - po získaní kľúča K_{AB} ho dokáže E vnútiť A alebo $\boldsymbol{B}$
7. $E(A) \rightarrow T: A, \{T_T^n, B, K_{AB}\}_{K_{AT}}$
8. $T \rightarrow B: \{T_T^{n+1}, A, K_{AB}\}_{K_{BT}}$

Wide mouthed frog protokol – oprava

- ochrana pred útokom spočíva v porušení symetrie správ
- šifrový text z prvého kroku nesmie mať rovnakú štruktúru ako šifrový text z druhého kroku
- to možno dosiahnuť napr. pridaním identifikátora odosielateľa alebo čísla správy do prvej správy:

1. $A \rightarrow T: A, \{T_A, A, B, K_{AB}\}_{K_{AT}}$
2. $T \rightarrow B: \{T_T, A, K_{AB}\}_{K_{BT}}$

Odporučenia

- Odporučenie A. (použitie kľúča)
  - nepoužívať kľúč na dva rôzne účely
    - napríklad na podpisovanie a šifrovanie
      - lebo napríklad pri RSA je podpisovanie a dešifrovanie rovnaká operácia
- Odporučenie B. (použitie transformácií)
  - opatrne pristupovať k podpisovaniu a dešifrovaniu s použitím súkromného kľúča
    - aby sa účastník nestal orákulom pre útočníka
- Odporučenie C. (nič nepredpokladať)
  - nepredpokladať nič, o čom sa nemôžeme sami presvedčiť a nie je v predpokladoch protokolu
    - napríklad: účastník niečo pozná, správa je v určitom tvare, staré kľúče sú zachované v tajnosti, atď.

- príklad pre B a C: TMN protokol

TMN Protokol

- Tatebayashi-Matsuzaki-Newman
- protokol na dohodnutie kľúča, kde dôveryhodný server S urobí väčšinu práce (A a B sú napríklad smartkarty)
- A a B nemajú dlhodobé kľúče
- S pozná faktorizáciu N a preto vie vypočítať tretiu odmocninu

1. $A \rightarrow S: A, S, B, N_A^3 \bmod N$
2. $S \rightarrow B: S, B, A$
3. $B \rightarrow S: B, S, A, N_B^3 \bmod N$
4. $S \rightarrow A: S, A, B, X$, pričom $X = N_A \oplus N_B$

- Nakoniec A získa $N_B = N_A \oplus X$.

TMN Protokol – útok

- protokol:

1. $A \rightarrow S: A, S, B, N_A^3 \bmod N$
2. $S \rightarrow B: S, B, A$
3. $B \rightarrow S: B, S, A, N_B^3 \bmod N$
4. $S \rightarrow A: S, A, B, X$, pričom $X = N_A \oplus N_B$

- útok, ak C zistí N_D (napríklad je predpovedateľné):

1. $C \rightarrow S: C, S, D, N_B^3 N_C^3 \bmod N$
2. $S \rightarrow D: S, D, C$
3. $D \rightarrow S: D, S, C, N_D^3 \bmod N$
4. $S \rightarrow C: S, C, D, X$, pričom $X = N_B N_C \oplus N_D$

- Nakoniec C získa N_B: $X \oplus N_D = N_B N_C$, $N_B N_C N_C^{-1} = N_B$.

Diffieho-Hellmanov protokol

- slúži na dohodnutie kľúča
- v pôvodnej podobe je bez autentifikácie
  - bezpečný len pri pasívnom útočníkovi
- p je veľké prvočíslo
- g je vhodne zvolený generátor podgrupy
- p a g môžu byť
  - pevne zvolené
  - vybrané z množiny preddefinovaných grúp
  - náhodne vybrané

1. $A \rightarrow B: p, g, g^{N_A} \bmod p$
2. $B \rightarrow A: g^{N_B} \bmod p$

- Následne A vypočíta: $K = (g^{N_B})^{N_A} \bmod p = g^{N_A N_B} \bmod p$
- Nakoniec B vypočíta: $K = (g^{N_A})^{N_B} \bmod p = g^{N_A N_B} \bmod p$

Diffieho-Hellmanov protokol – útok

- pokiaľ môže útočník prenášané správy meniť (Dolevov-Yaov model), je možný MITM (man in the middle) útok

1. $A \rightarrow M(B): p, g, g^{N_A} \bmod p$
2. $M(A) \rightarrow B: p, g, g^{N'_M} \bmod p$
3. $B \rightarrow M(A): g^{N_B} \bmod p$
4. $M(B) \rightarrow A: g^{N''_M} \bmod p$

- A vypočíta: $K_A = \left(g^{N''_M}\right)^{N_A} \bmod p = g^{N_A N''_M} \bmod p$
- B vypočíta: $K_B = \left(g^{N'_M}\right)^{N_B} \bmod p = g^{N'_M N_B} \bmod p$
- S vysokou pravdepodobnosťou $K_A \neq K_B$, ale útočník vie vypočítať obe hodnoty a prešifrovávať posielané správy

Diffieho-Hellmanov protokol – varianty

- je základom pre výmenu kľúča v mnohých používaných protokoloch
- obvykle používajú rôzne varianty protokolu zamedzujúce MITM útoku

- napríklad protokol TLS 1.2 s dvoma účastníkmi, označovanými ako
  - klient (typicky webový prehliadač)
  - server (typicky web server)

  špecifikuje nasledujúce varianty:
  - DH_anon
    - anonymný DH protokol bez autentifikácie
    - možný MITM útok
  - DHE_RSA, DHE_DSS (E – Ephemeral – dočasný)
    - server generuje p, g a x pričom ich spolu s $g^x \bmod p$ podpíše použitím podpisovej RSA schémy, resp. použitím DSA algoritmu definovanom v štandarde DSS
    - server v takomto prípade disponuje certifikátom verejného kľúča, ktorý pošle klientovi, aby klient mohol podpis overiť
  - DH_RSA a DH_DSS
    - parametre DH protokolu sú raz pevne zvolené a sú súčasťou certifikátu servera

Needhamov-Schroederov PK protokol

- využíva asymetrické šifrovacie algoritmy
  - PK – public key (protokol s verejnými kľúčmi)
- účastníci protokolu si navzájom poznajú svoje verejné kľúče
- čerstvosť správ zaručená pomocou príležitostných slov
- cieľom protokolu je vzájomná autentifikácia účastníkov
  - N_A a N_B sa dajú použiť na odvodenie kľúčov (nebol pôvodný cieľ)

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

- trvalo 17 rokov, kým sa našla slabina

NSPK protokol – útok

- útok predpokladá, že A má záujem o NSPK s účastníkom E
  - E je zároveň útočníkom (napr. kompromitovaný účastník)
- útok prebieha ako dve paralelné inštancie NSPK
  - medzi A, E a medzi $E(A), B$ (obe inštancie skončia úspešne)

1. $A \rightarrow E: \{A, N_A\}_{K_E}$
2. $E(A) \rightarrow B: \{A, N_A\}_{K_B}$
3. $B \rightarrow E(A): \{N_A, N_B\}_{K_A}$
4. $E \rightarrow A: \{N_A, N_B\}_{K_A}$
5. $A \rightarrow E: \{N_B\}_{K_E}$

E použije A ako orákulum pre dešifrovanie

6. $E(A) \rightarrow B: \{N_B\}_{K_B}$

- útočník E sa autentifikuje pre B ako účastník A
  - navyše získa obe príležitostné slová N_A, aj N_B

NSPK protokol – oprava

- útoku zabránime doplnením identifikátora druhého účastníka do druhej správy
  - útočník potom nemôže túto správu úspešne preposlať v inej inštancii protokolu

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

Open-Source Fixed-Point Model Checker

- môže skrátiť 17 rokov na 17 ms
- http://www2.imm.dtu.dk/~samo/ofmc-2014.zip
  - BSD licencia (voľne použiteľné)
- Protokol treba zapísať pomocou „Alice and Bob notation“
  - súbory s príponou .AnB
- kontrolu spustíme:
  - ofmc-win-64bit NSPK.AnB

NSPK.AnB

```
Protocol: NSPK

Types:      Agent A, B;
            Number NA, NB;
            Function pk

Knowledge:  A: A, pk, inv(pk(A)), B;
            B: B, pk, inv(pk(B))

Actions:
A->B: { NA, A }pk(B)
B->A: { NA, NB }pk(A)
A->B: { NB }pk(B)

Goals:
B authenticates A on NA
A authenticates B on NB
NA secret between A,B
NB secret between A,B
```

Open-Source Fixed-Point Model Checker

ATTACK TRACE:

(x602,1) -> i: {NA(1),x602}_(pk(i))

i -> (x601,1): {NA(1),x602}_(pk(x601))

(x601,1) -> i: {NA(1),NB(2)}_(pk(x602))

i -> (x602,1): {NA(1),NB(2)}_(pk(x602))

(x602,1) -> i: {NB(2)}_(pk(i))

i -> (x601,1): {NB(2)}_(pk(x601))

- účastníka x602 nahradíme účastníkom A
- účastníka x601 nahradíme účastníkom B
- účastníka i nahradíme účastníkom E

Open-Source Fixed-Point Model Checker

ATTACK TRACE:

(A,1) -> E: {NA(1),A}_(pk(E))

E -> (B,1): {NA(1),A}_(pk(B))

(B,1) -> E: {NA(1),NB(2)}_(pk(A))

E -> (A,1): {NA(1),NB(2)}_(pk(A))

(A,1) -> E: {NB(2)}_(pk(E))

E -> (B,1): {NB(2)}_(pk(B))

- účastníka (A,1) z prvého spojenia nahradíme A
- účastníka (B,1) z prvého spojenia nahradíme B
- obdobne NA(1) a NB(2) nahradíme NA a NB

Open-Source Fixed-Point Model Checker

ATTACK TRACE:

A -> E: {NA,A}_(pk(E))

E -> B: {NA,A}_(pk(B))

B -> E: {NA,NB}_(pk(A))

E -> A: {NA,NB}_(pk(A))

A -> E: {NB}_(pk(E))

E -> B: {NB}_(pk(B))

- _(pk(X)) nahradíme _KX

Open-Source Fixed-Point Model Checker

ATTACK TRACE:

A -> E: {NA,A}_KE

E -> B: {NA,A}_KB

B -> E: {NA,NB}_KA

E -> A: {NA,NB}_KA

A -> E: {NB}_KE

E -> B: {NB}_KB

- už len upravíme formátovanie

Open-Source Fixed-Point Model Checker

ATTACK TRACE

1. $A \rightarrow E: \{N_A, A\}_{K_E}$
2. $E \rightarrow B: \{N_A, A\}_{K_B}$
3. $B \rightarrow E: \{N_A, N_B\}_{K_A}$
4. $E \rightarrow A: \{N_A, N_B\}_{K_A}$
5. $A \rightarrow E: \{N_B\}_{K_E}$
6. $E \rightarrow B: \{N_B\}_{K_B}$

Prezentovaný útok

1. $A \rightarrow E: \{A, N_A\}_{K_E}$
2. $E(A) \rightarrow B: \{A, N_A\}_{K_B}$
3. $B \rightarrow E(A): \{N_A, N_B\}_{K_A}$
4. $E \rightarrow A: \{N_A, N_B\}_{K_A}$
5. $A \rightarrow E: \{N_B\}_{K_E}$
6. $E(A) \rightarrow B: \{N_B\}_{K_B}$

Poučenie

- ilustrovali sme rôzne slabiny kryptografických protokolov
- ukázali sme, že navrhnúť (a implementovať) bezpečný protokol nie je jednoduché
  - navyše, aj keď odstránime identifikované slabiny, nemáme záruku, že výsledný protokol už je bezpečný
- v súčasnosti sa pri návrhu a analýze protokolov využívajú formálne metódy a automatizované dokazovanie bezpečnosti, resp. automatizované hľadanie slabín
- ako príklad sme uviedli nástroj OFMC
  - ale sú ďalšie, ako napríklad: ProVerif

Challenge-Handshake Authentication Protocol

- CHAP, definovaný v RFC 1994
- autentifikačný protokol medzi dvoma účastníkmi, ktorého cieľom je jednostranná autentifikácia jedného z nich pomocou hesla
  - heslo je známe v otvorenom tvare obom účastníkom
  - používa sa pri PPP, keď server overuje identitu pripájaného klienta
    - na začiatku aj počas spojenia
- pre zjednodušenie označme
  - klientom A účastníka, ktorý sa autentifikuje (v RFC je to „peer“)
  - serverom S účastníka, ktorý autentifikáciu požaduje („authenticator“)
- protokol prebieha nasledovne:

1. $S \rightarrow A: C, I, Ch$
   - I je identifikátor pre túto inštanciu protokolu
   - Ch je výzva generovaná serverom
2. $A \rightarrow S: R, I, Re$
   - Re je odpoveď na výzvu Ch: $Re = H(I, heslo, Ch)$
   - H je hašovacia funkcia, obvykle MD5
3. $S \rightarrow A: S, I$ alebo $S \rightarrow A: F, I$
   - server overí korektnosť odpovede Re
   - podľa toho oznámi úspech S alebo neúspech F

Sym.	Kód	Hod.
C	Challenge	1
R	Response	2
S	Success	3
F	Failure	4

Challenge-Handshake Authentication Protocol

- ak je potrebné pred autentifikáciou klienta informovať server o identite klienta (aby server vedel použiť správne heslo), je taká správa súčasťou sieťového protokolu, ktorý CHAP využíva
- výhoda CHAP
  - heslo nie je posielané v otvorenom tvare
    - nepochybne lepšia situácia ako v PAP (Password Authentication Protocol)
      - heslo posiela v otvorenom tvare
  - ochrana pre útokom opakovaním
- nevýhody CHAP :
  - poskytuje len jednostrannú autentifikáciu
  - neposkytuje žiadny spôsob pre dohodnutie kryptografických kľúčov
  - náchylný na slovníkový útok a útok úplným preberaním
    - zachytené správy protokolu umožňujú skúšať heslá off-line
    - v kontexte so sieťovým protokolom je prezradená identita klienta

Secure Shell protokol

- SSH-2, architektúra definovaná v RFC 4251
- primárne používaný pre vzdialený interaktívny prístup k príkazovému riadku serverov
  - umožňuje bezpečný prenos súborov
  - preposielanie a tunelovanie komunikácie na sieťovom porte
- SSH-2 je vylepšená verzia predchádzajúcej verzie Secure Shell protokolu, pričom odstránila aj jeho známe bezpečnostné slabiny
  - vloženie obsahu do šifrovaného SSH dátového toku
    - z dôvodu nedostatočnej ochrany integrity pomocou CRC-32
  - nedochádzalo k obmieňaniu kľúčov spojenia
    - SSH-2 obmieňa kľúče každý 1 GB alebo hodinu (čo príde skôr)
  - ...

Hlavné časti SSH-2

- Transport Layer Protocol (RFC 4253)
  - poskytuje jednostrannú autentifikáciu servera, distribúciu kľúča spojenia, šifrovanie, integritu, jedinečný identifikátor spojenia
  - voliteľne môže poskytovať aj kompresiu
  - obvykle prebieha nad TCP/IP
    - v princípe nad ľubovoľným spoľahlivým komunikačným kanálom
- User Authentication Protocol (RFC 4252)
  - zabezpečuje autentifikáciu klienta (používateľa)
  - prebieha nad Transport Layer protokolom
  - podporuje viaceré metódy autentifikácie
- Connection Protocol (RFC 4254)
  - poskytuje logické kanály pre služby (shell, preposielanie TCP/IP, atď.)
  - umožňuje multiplexovanie logických kanálov nad jedným spojením
  - prebieha nad Transport Layer a User Authentication protokolom
    - nesúvisí s kryptografiou

SSH-2 – Transport Layer Protocol

- zabezpečuje
  - distribúciu (odvodenie) kryptografických kľúčov
  - následný prenos údajov so zabezpečenou
    - dôvernosťou pomocou šifrovania
    - autentickosťou pomocou MAC
- umožňuje dohodnúť rôzne algoritmy pre rôzne smery komunikácie (klient → server, server → klient)
  - nie je to odporučené
- umožňuje dohodnúť rôzne kľúče pre rôzne smery komunikácie (klient → server, server → klient)
  - nutné

Kryptografické konštrukcie v SSH-2

Povinné

Konštrukcia	Algoritmus
šifrovanie	3des-cbc
integrita dát	hmac-sha1
výmena kľúča	diffie-hellman-group1-sha1 diffie-hellman-group14-sha1
digitálne podpisy	ssh-dss

Odporúčané

Konštrukcia	Algoritmus
šifrovanie	aes128-cbc
integrita dát	hmac-sha1-96
výmena kľúča	
digitálne podpisy	ssh-rsa

Pozor chyták:

- diffie-hellman-group1-sha1 používa Oakley Group 2 z RFC 2409
  - používa 1024 bitové prvočíslo
- diffie-hellman-group14-sha1 používa Oakley Group 14 z RFC 2409
  - používa 2048 bitové prvočíslo
- novšie RFC 4344 odporúča: aes128-ctr, aes192-ctr, aes256-ctr, 3des-ctr
  - OpenSSH od verzie 5.2 z roku 2009

Výmena kľúčov v SSH-2 – úvod

- využíva Diffieho-Hellmanov protokol
  - p,g,q sú hodnoty určené konkrétnou grupou pre DH protokol
- najprv si klient a server
  - pošlú identifikačné reťazce (SSH-ProtoVersion-SoftwareVersion comments)
  - dohodnú použité algoritmy (SSH_MSG_KEXINIT)
- označenie pre C – klient a S – server:
  - V_C: identifikačný reťazec klienta
    - SSH-protoversion-softwareversion comments
  - V_S: identifikačný reťazec servera
    - napríklad: SSH-2.0-OpenSSH_5.9p1 Debian-5ubuntu1.4
  - I_C: obsah klientovej správy SSH_MSG_KEXINIT
    - cookie (16 náhodných bajtov, príležitostné číslo), kex_algorithms, server_host_key_algorithms, encryption_algorithms_client_to_server, encryption_algorithms_server_to_client, ...
  - I_S: obsah serverovej správy SSH_MSG_KEXINIT
    - cookie (16 náhodných bajtov, príležitostné číslo), kex_algorithms, server_host_key_algorithms, encryption_algorithms_client_to_server, encryption_algorithms_server_to_client, ...
  - K_S: verejný kľúč servera
    - podľa zvoleného algoritmu z server_host_key_algorithms
  - K: zdieľané tajomstvo pre odvodenie kľúčov

Výmena kľúčov v SSH-2 – tajomstvo

- C si náhodne zvolí x, pričom $1 < x < q$
- C vypočíta $e = g^x \bmod p$
- S si náhodne zvolí y, pričom $1 < y < q$
- S vypočíta $f = g^y \bmod p$

1. $C \rightarrow S: e$ (SSH_MSG_KEXDH_INIT)

- S vypočíta $K = e^y \bmod p$
- S vypočíta $H = hash(V_C, V_S, I_C, I_S, K_S, e, f, K)$
- S vypočíta s, čo je podpis H jeho súkromným kľúčom

2. $S \rightarrow C: K_S, f, s$ (SSH_MSG_KEXDH_REPLY)

- C overí, že K_S je naozaj verejný kľúč servera S (napr. v lokálnej databáze)
  - C môže akceptovať certifikát aj bez overenia (potom ale pozor na MITM)
- C vypočíta $K = f^x \bmod p$
- C vypočíta $H = hash(V_C, V_S, I_C, I_S, K_S, e, f, K)$
- C overí, či s je korektný podpis servera S pre správu H

- pokiaľ všetky kroky prebehli úspešne
  - tak C má overenú identitu servera S
  - C aj S sa dohodli na spoločnej tajnej hodnote K
  - H z 1. výmeny sa použije ako $session\_id$ pre toto spojenie (už sa nemení)

Výmena kľúčov v SSH-2 – generovanie

- pokračuje sa odvodením konkrétnych kryptografických parametrov (rovnako postupuje aj server):

- iniciálny IV (klient → server): $hash(K, H, "A", session\_id)$
- iniciálny IV (server → klient): $hash(K, H, "B", session\_id)$
- šifrovací kľúč (klient → server): $hash(K, H, "C", session\_id)$
- šifrovací kľúč (server → klient): $hash(K, H, "D", session\_id)$
- MAC kľúč (klient → server): $hash(K, H, "E", session\_id)$
- MAC kľúč (server → klient): $hash(K, H, "\mathrm{F}", session\_id)$

- pokiaľ má odtlačok menej bitov ako kľúč, postupuje sa nasledovne:
  - $k_1 = hash(K, H, "\mathrm{C}", session\_id)$
  - $k_2 = hash(K, H, k_1)$
  - $k_3 = hash(K, H, k_1, k_2)$
  - ...
  - $k = k_1, k_2, k_3, \ldots$

SSH-2 – bezpečný kanál

- SSH-2 umožňuje kedykoľvek inicializovať opätovnú výmenu kľúča
- v prípade CTR módov sú IV použité na inicializáciu vnútorného počítadla
- po odoslaní resp. prijatí špecifických správ ukončujúcich výmenu kľúča sa klient a server prepnú do stavu, v ktorom je celá odosielaná resp. prijímaná komunikácia spracúvaná dohodnutými kryptografickými algoritmami a kľúčmi
- SSH-2 počíta autentifikačný kód z nešifrovaného paketu a následne šifruje prenášané dáta (MAC šifrovaný nie je)
  - pokaľ implementácie podporujú módy pre autentifikované šifrovanie (napr. mód GCM), situácia v zabezpečení paketu je iná

SSH-2 – User Authentication Protocol

- už predpokladá vytvorený bezpečný komunikačný kanál, zabezpečujúci dôvernosť a integritu prenášaných dát
- prebieha nad transportným protokolom
- SSH-2 definuje v RFC 4252 nasledujúce metódy autentifikácie:
  - publickey (povinná)
  - password (voliteľná, odporúčaná)
    - používateľ dokazuje svoju identitu prostredníctvom hesla
    - klient posiela používateľské meno a heslo (v otvorenom tvare)
      - mala by byť zakázaná, ak transportná vrstva neposkytuje dôvernosť
    - server môže požadovať aj zmenu hesla
      - mala by byť zakázaná, ak transportná vrstva neposkytuje dôvernosť a integritu
  - hostbased (voliteľná)
  - none (neodporúčaná)
    - server nepožaduje autentifikáciu používateľa

SSH-2 – publickey user authentication

- vlastníctvo súkromného kľúča slúži ako dôkaz identity
- klient si vyberie algoritmus na podpisovanie
  - nie je obmedzený tým, čo bolo vybrané pri výmene kľúčov
- klient podpíše nasledovné dáta:
  - $session\_id$, SSH_MSG_USERAUTH_REQUEST
  - používateľovo meno, meno služby, "publickey", TRUE
  - meno algoritmu pre podpisovanie, verejný kľúč
- server overí, či uvedený verejný kľúč je vhodný na autentifikáciu a či je podpis korektný

SSH-2 – hostbased user authentication

- dôkaz identity pomocou hostiteľovho súkromného kľúča a používateľovho mena
  - nie je vhodné pre prostredia s vyššími požiadavkami na bezpečnosť
- klient si vyberie algoritmus na podpisovanie
  - nie je obmedzený tým, čo bolo vybrané pri výmene kľúčov
- klient podpíše nasledovné dáta:
  - $session\_id$, SSH_MSG_USERAUTH_REQUEST
  - používateľovo meno, meno služby, "hostbased",
  - meno algoritmu pre podpisovanie,
  - verejný kľúč a certifikáty pre hostiteľa,
  - meno hostiteľa ako FQDN
  - používateľovo meno na hostiteľovi v UTF-8
- server overí, či uvedený verejný kľúč patrí hostiteľovi, či daný používateľ na danom hostiteľovi sa môže prihlásiť a či je podpis korektný

Ďakujem za pozornosť