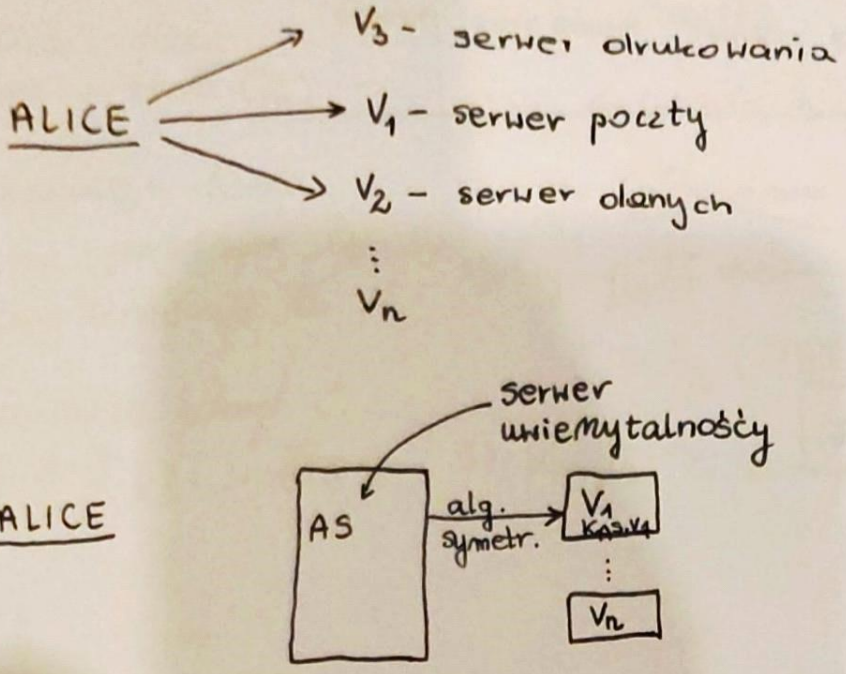
KERBEROS (wersja 4) — Alg. symetryczny!



LEGENDA!!!

- P_A - hasło ALICE
- K_{A^S,V_1} - klucz tajny do algorytmu symetrycznego
- ID_V - identyfikator serwera V
- ID_A - identyfikator Alice
- T - ticket (bilet)
- AD_A - Adres sieciowy Alice

Prosty protokół uwierzytelniania

Faza uwierzytelnienie

ALICE

1) Wysyła ID_A, P_A, ID_V

AS

2) Uwierzytelnia na podstawie P_A (porównanie)

3) Tworzy $T = E_{K_{AS,V}}(ID_A, AD_A, ID_V)$

4) Wysyła T do Alice

5) Wysyła ID_A, T do V

V

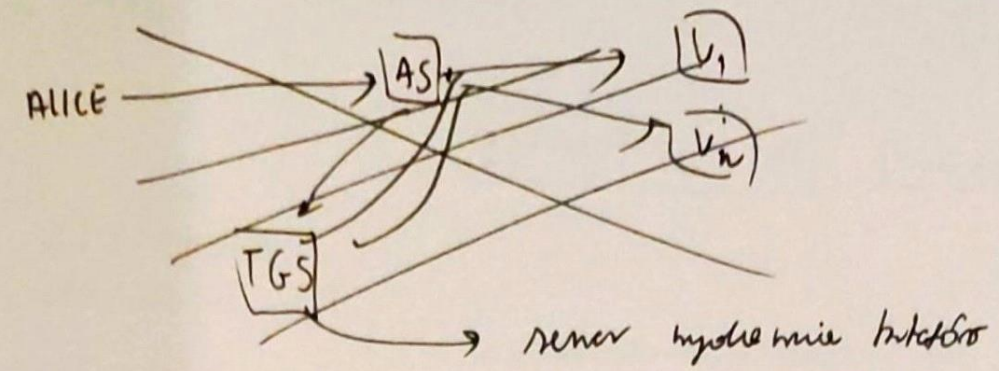
6) Deszyfruje T i wten sposób Alice : ID_A, AD_A, ID_V

6)

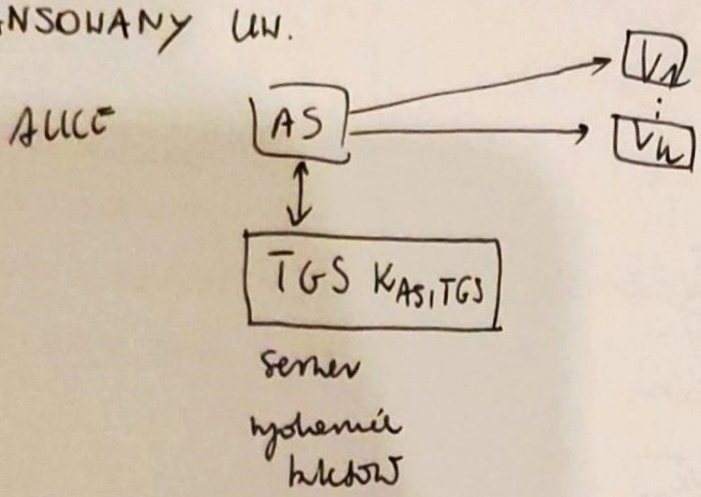
- porównuje A_{DA} z A_{DA} (czy wysłany z wł. adresu)

7) Sprawdza czy ID_V jest poprawny (czy są dobre zdeszyfr.)

- porównuje ID_A z ID_A (czy bilet został wydany IDA)



ZAAWANSOWANY UW.



IDEA

I Alice ——— jednokrotnej ———→ AS

←——— T(bilet do TGS)

II Alice ——— bilet do TGS ———→ TGS

←——— bilet do zasobów V (Tv)

III ALICE ——— T_v ———→ V

FAZA I

ALICE

1) Wysyła ID_A, ID_{TGS}

6) Klient Alice prosi o hasło (P_A)

7) Alice podaje hasło i na podstawie P_A Klient Generuje K

8) Alice może odszyfr. C tzn $D_K(C) = \overline{TGS}$

AS

2) Tworzy bilet

$T_{TGS} = E_{K_{AS,TGS}}(ID_A, AD_A, ID_{TGS}, TS_1, LT_1)$ (life of time)

(znacznik czasu, długość czasu trwania)

3) Na podstawie P_A generuje klucz sesyjny do algorytmu symetrycznego.

4) Szyfruje K bilet T_{TGS}

$C = E_K(TTGS)$

5) Wysyła C do Alice

FAZA II

ALICE

1) Wysyła ID_A, ID_V, T_{TGS} ⟶

TGS

2) Deszyfruje T_{TGS}

$D_{K_{AS,TGS}}(TTGS) = [ID_A, AD_A, ID_{TGS}, TS_1, LT_1]$

* Sprawdza
* porównuje $AD_A = AD_A$
* -11- $ID_A = ID_A$
* czy bilet został wydany Alicji
* Bilet jest ważny (aktualny) TS_1, LT_1
* Sprawdzenie uprawnień do V z Alice.

3) Tworzy $T_V = E_{K_{TGS,V}}(ID_A, AD_A, ID_V, TS_2, LT_2)$

4) Wysyła T_V Alice.

FAZA III

ALICE

V

1) ID_A, T_V →

2) Wykonuje te same kroki co TGS w fazie II
* deszyfruje $D_{K_{TGS,V}}(T_V)$...

UWAGI

1) Bilet zostanie wydany jeśli wiadomo, że one to one?
2) Będzie wiadomo czy connect with right serwer.

Jeśli wszystko zakończy się poprawnie to Alice dostaje

FINAL KERBEROS IV

I ALICE

AS

1) wysyła ID_A, ID_{TGS}, TS_1 →

2) ~~Tworzy bilet~~
Generuje $K_{A,TGS}$ – klucz do alg. symetr.

3) Tworzy $T_{TGS} = E_{K_{AS,TGS}}[K_{A,TGS}, ID_A, AD_A, ID_{TGS}, TS_2, LT_2]$

4) Na podstawie hasła generuje K

5) Szyfruje $C = E_K(K_{A,TGS}, ID_{TGS}, TS_2, LT_2, TGS)$

6) Wysyła C do ALICE

7) Alice podaje hasło P_A i w ten sposób jako wyjście uzyskuje dostęp do $K_{A,TGS}$, T_{TGS}

ALICE

1) Tworzy AUTH $= E_{K_{A,TGS}}(ID_A, AD_A, TS_5)$

2) Wysyła do TGS

$ID_V, T_{TGS}, AUTH$ →

TGS

3) ~~Odbiera~~ Deszyfruje $D_{K_{AS,TGS}}(T_{TGS}) = [K_{A,TGS}, \dots]$

4) Deszyf. AUTH

$D_{K_{A,TGS}}(AUTH) = [ID_A, AD_A, TS_3]$

5) Sprawdza dane po zdeszyfrowaniu T_{TGS} i AUTH

6) Generuje $K_{A,V}$ - klucz do algor. symetr.

7) Tworzy $T_V = E_{K_{TGS,V}}(K_{A,V}, ID_A, AD_A, ID_V, TS_4)$ ~~TV~~

8) Tworzy szyfruje C

$C = E_{K_{A,S}}(K_{A,V}, ID_V, T_V)$

i wysyła do Alic

ALICE

1) Tworzy $AUTH = E_{K_{A,V}}(ID_A, AD_A, TS_5)$

2) Wysyła do V

AUTH, V →

V

J.W (jak TGS)

3) szyfruje

$C = E_{K_{A,V}}(TS_{5+1})$ ←

4) Jeśli wynik ok TO AUTH

Wie(, że komunikuje) z dobrym serwerem.

BPS1 - wykład 12/12/2019

1. Protokół SSL

	WARSTWA
KERBEROS / SSH	aplikacji
SSL	
TCP	
IP	

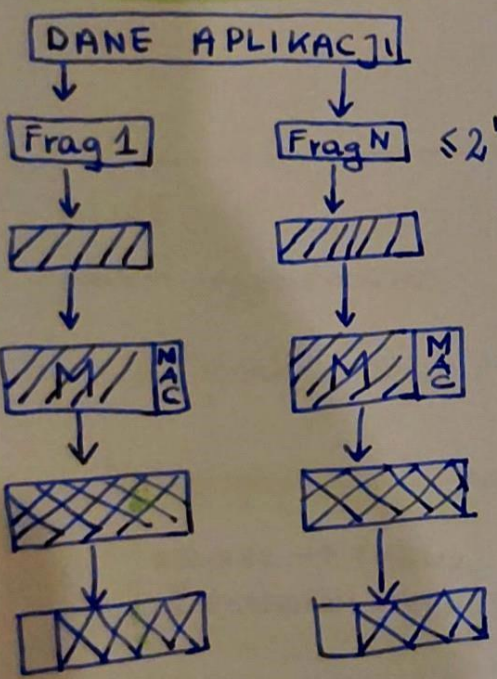
PROT. - Protocol

2. Struktura SSL

SSL HAND-SHAKE	SSL ALERT PROT.	SSL CHANGE CIPHER SPECYFICATION
SSL RECORD PROT.		

Klucze są już uzgodnione (zapisane).

3. SSL record protocol



Fragmentacja

Kompresja (opcjonalna) bezstrona

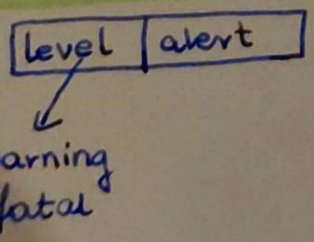
Tworzenie MAC (message authentication code)

K - tajne (sesyjne)
H(K, M)

Szyfrowanie z uzgodnionym alg. symetrycznym E.
(K, E, D)

Dodanie nagłówka SSL

4. SSL alert protokol



4. Cel Handshake:

1) uzgodnienie wersji SSL $\in \{1, 2, 3\}$

2) wybranie | uzgodnienie algorytmów kryptograficznych (np. H, (E,D), tryby itp.)

3) „Wzajemne" uwierzytelnienie — NAJRZADZIEJ!

CLIENT SERVER

a) ← (serwer do klienta)

b) → ← (klient do serwera, serwer do klienta)

c) —

4) Uzgodnienie „Master Key"

$K_S = (e, n)$

S - podpis

K_{CA} - klucz pub. CA (do weryfikacji $CERT_{SE}$)

3a

ALICE

SERVER

$CERT_{Se} = [K_{Se}, ID_{Se}, S]$

① CLIENT_HELLO →

1	WERSJA SSL	C_{RAND}	SESSION ID	CIPHER SUITE	COMPRESS SUITE

C_{RAND} — losowa liczba

SESSION ID — domyślnie „0" na początku sesji

CIPHER SUITE — np. 3DES_CBC_SHA3

← SERVER HELLO ②

2	3	S_{RAD}	SESSION ID	USTALA E,D,H itp	...

3 — wybiera wersję bezpieczeństwa

S_{RAD} — losowa wartość

SESSION ID — nadaje wartości

← CERTIFICATE ③

CLIENT ← SERWER W TYM MOMENCIE!

3	$CERT_S$

← SERVER_HELLO DONE ④

ALICE

D-H edition

$CERT_{SE} = [p, q, y_S, ID_S, S]$

$pmk = y_S^{x_S} \bmod p$

$y_A = g^{x_S} \bmod p$

server | client key exchanged

NOR { CKE → $c = RSA_k(pms)$ ←

D-H { $y_{CA} \equiv g^{x_{CA}} \bmod p$

⑤ Weryfikuje $CERT_{SE}$

(... zabierz mnie stąd ... - dodaj wyjątek bezpieczeństwa (brak K_{CA}))

W tym celu musi znać K_{CA}

⑥ Generuje PMS (pre-master sekret) (prawd. < n)

⑦ Szyfruje $c = RSA_{K_{SE}}(pms)$

⑧ Wysyła komunikat client-key-exchange

c →

⑨ Wysyła komunikat change cipher-spec →

← change cipher-spec ⑩

---

dodatkowe opisywanie ⑦,⑧

ALICE

pmk - wygenerowane

SERVER

pms - po odszyfrowaniu

$H(pmk, C_{RAND}, S_{RAND}, + \dots)$ $H(\dots)$ = Master Key (obie strony go posiadają)

Deterministyczna metoda

deterministycznie ustawione

---

⑪ FINISHED ($H(MASTERKEY + \dots)$ obie strony go wysyłają i porównują.)

(1)

KLEPTOGRAFIA — kryptografia w kryptografii

ALICE

$CERT_A = [K, ID_A, S_{CA}]$

↑ klucz pub do weryfikacji podpisu

K - klucz tajny do podpisu

$SIG_K(M) = S$

(3b)

CLIENT HELLO →

← SERV_HELLO

← CERT

← CERT REQUEST

M - wiadomość, którą ma podpisać ALICE

CERT
CERT$_A$ →

(X) CERT VERFICATION →
[M, S]

CLIENT EXCHAGE KEY
C = RSA$_E$ (pms)

po otrzymaniu cert (weryfikuję) oraz po otrzymaniu X

← SERV. DONE

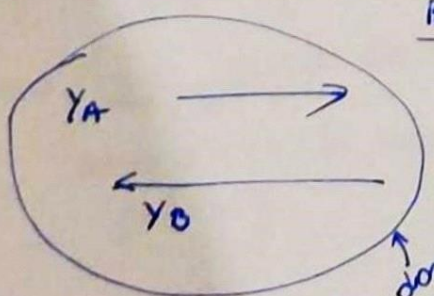
KLEPTOGRAFICZNY D-H

p - pierwsza
ρ - generator

MALLET
X - klucz tajny
$Y \equiv g^X \bmod p$
H - f. hash.
W, a, b - stałe

ALICE

BOB



S_{AB}

S_{AB}

dostępne dla każdego, osoba, która posiada klucz z S_{AB}

„SKAŻONY DH"

- PIERWSZE URUCHOMIENIE DH (ALICE)

1) Losuje c_1, $1 < c_1 < p$

2) Oblicza $m_1 \equiv g^{c_1} \bmod p$

3) Zapisuje c_1 w pamięci komputera

4) return m_1 $\longmapsto$ m_1 (zostało wysłane)

- DRUGIE URCHOMIENIE DH (i kolejne)

1) Losuje $t \in \{0,1\}$

2) Oblicza $z = g^{c_1 - wt} \cdot Y^{-a(c_1-wt)-b} \pmod p$ — tutaj zostało zdemaskowane nie bezpośrednio

3) Oblicza $c_2 = H(z)$

4) Oblicza $m_2 = g^{c_2} \bmod p$

5) return m_2 $\longmapsto$ m_2 (zostało wysłane)

„ODZYSKANIE c_2"

Dane:
m_1, m_2, w, a, b, X

1) Oblicza $r \equiv m_1^{\,a} g^{b} \pmod p$

2) Oblicza $z_1 \equiv m_1 (r^x)^{-1} \pmod p$

3) Jeśli $m_2 \equiv g^{H(z_1)} \pmod p$, to return $H(z_1)$

4) Oblicz $z_2 = z_1 (g^w)^{-1} \pmod p$

5) Jeśli $m_2 \equiv g^{H(z_2)} \pmod p$ to return $H(z_2)$

ANALIZA

1) $t = 0$

$$r = m_1^{\,a} g^b \pmod p = g^{c_1 a} g^b \equiv g^{c_1 a + b} \pmod p$$

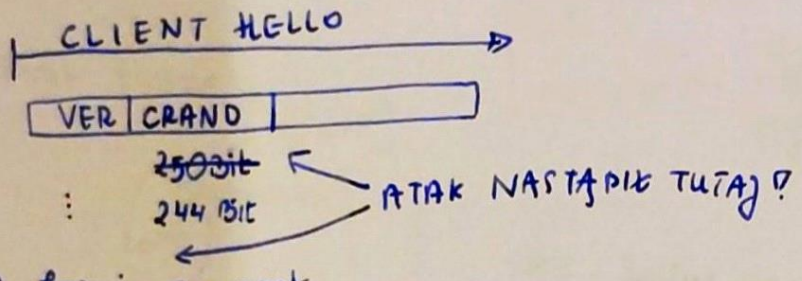
Wtedy:

$$z_1 = m_1 \cdot (r^x)^{-1} \equiv g^{(c_1 a + b)x} \cdot g^{c_1 \cdot x} \pmod p =$$

$$= g^{c_1 Y^{-ac_1 - b}} \pmod p = z \pmod p$$

$$z_2 = z_1 g^{-W} = g^{q - W} Y^{-ac_1 - b} \equiv z \pmod p \quad ; t = 1$$

SKAŻONY HAND SHAKE



1) Losuj $c = prnlc$

$$x_c \cdot y_c \equiv g^{x_c} \pmod p$$

„Losuj"

G - generator

$G(seed) = b_1 \dots b_n = C$

Jeśli były RSA IDH mod p
Mamy, że p nie może być
244 bit musi mieć
więcej.

Po kilkunastym razem
2245 x 10 ⇒ otrzymamy
p lub n to wtedy
dostajemy nasz random

Dodatkowo musimy wiedzieć
jaka „technologia" została
użyta.

- PIERWSZE WYKONANIE - Ktoś ma skażony SSL uruchamia go po raz pierwszy

1) Losuje k; $kg \in \underbrace{E(\mathbb{F}_p)}_{\text{KRZYWE ELIPTYCZNE}}$ / generator ~~INF~~ ; $k \in \mathbb{N}$ i zapisuje k na dysku

$c = g^k$ w grupie g

$C_{RAND} = (C$ Li pakuje do CLIENT_HELLO $\pmod{225})$

Jeśli $C_{RAND} > 244$ to wynika ile napierw dwieli to na pakiety

MALLET
X - klucz tajny
$Y \equiv g^X \bmod p$
H - funkcja hash.

DRUGIE WYKONANIE 1) $i = 1, \dots$

2) $seed = H(Y^k, i)$ — pełni rolę „zmieniana" do generatora dodat. i nie jest ukryte.

Y^k — klucz publiczny

3) $i = i + 1$

zapisuje i.

4) Oblicza $G(seed) = x_c$

$x_c \cdot (y_c) \equiv g^{x_k}$

x_c połączone do y_c i y_c następnie wysłane.

MALLET

$C \equiv g^k$ w grupie $\boxed{y = Y}$

Oblicza $c^x = g^{kx} = y^k$ w g

Oblicza seed w zależności od i

$seed_i = H(Y^k, i)$

W ten sposób po i możemy zatrzeć SSL.

RSA

p, q — sekretne

$n = pq$

$e: \quad (e, \varphi(n)) = 1$

$d: \quad ed = 1 \pmod{\varphi(n)}$

$c \equiv m^E \pmod N$

$(e, \varphi(n)) = 1$

Jak posiada p to faktoryzuje „n".

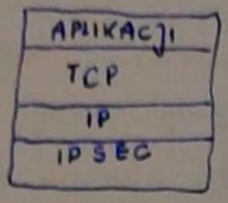
MALLET / RSA

$k = (D, N)$

$K = (E, N)$

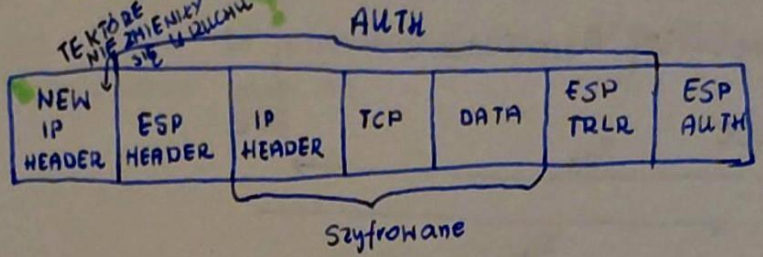
9.01.2020

DSPI - wykład

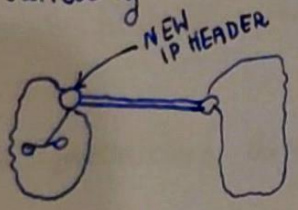


IPSEC:

* IKE - Internet Key Exchange
** ESP - Encapsulting Security Protocol
*** AH - Authentication Header Protocol

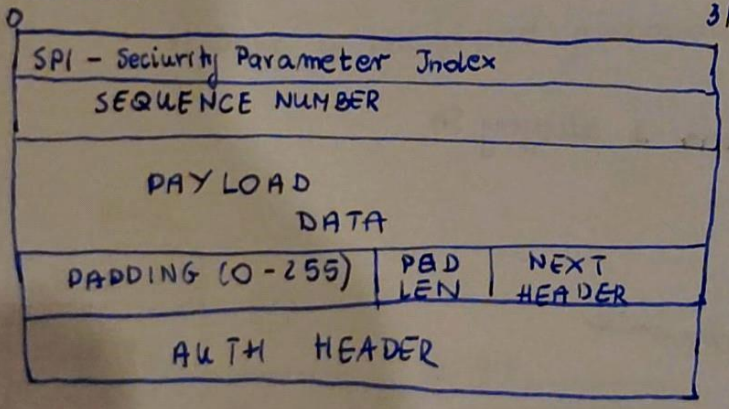


Tryb tunelowy:



Tryb transportowy:

ESP HEADER

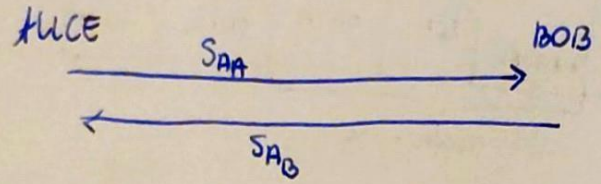


IKE :

- negocjacja parametrów
- wymiana (ustalenie) klucy (DH)* — rozbudowany (DH)
- uwierzytelnianie stron
- zarządzanie kluczami

zbudowane z :
→ SKEME - metody uwierzytelnienie
→ OAKLEY - mechanizm opanty trybach do wymiany kluczy
→ ISAKMP - dostarsza architektury paimetrów

SA - Security Associations



~~SA~~

SA jest wyznaczone przez:

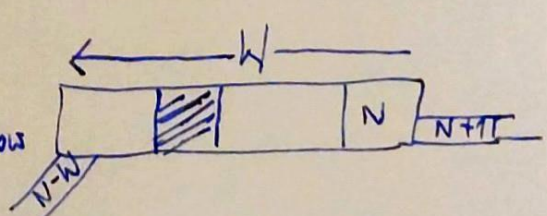
- SPI
- IP destination address
- Identyf. Prot. bezpieczeństwa tz AH lub ESP.

SAD - SA DATA BASE:

- przechowuje parametry związane z aktywny SA

np: * jaki tryb
* jaki protokół
* metody uwierzytelnienie
* Grupa DH
* LOCALNE..

+ długość życia... , ANTY replay window



SPD : (SECURITY POLICY DATABASE)

- INbound
  - * jakie przyjmować
  - * odrzuć, omiń, zastosuj
- outbound

-II-

„RUCH IPSEC"

1) Pakiet dociera do IP
2) Przeszukanie SPD, JEŚLI IPSEC to przeszukanie bazy ~~SPD~~ SAD i znalezienie akt. SA
3) Jeśli nie ma SA to protokół IKE.

IKE jest protokołem dwufazowym:

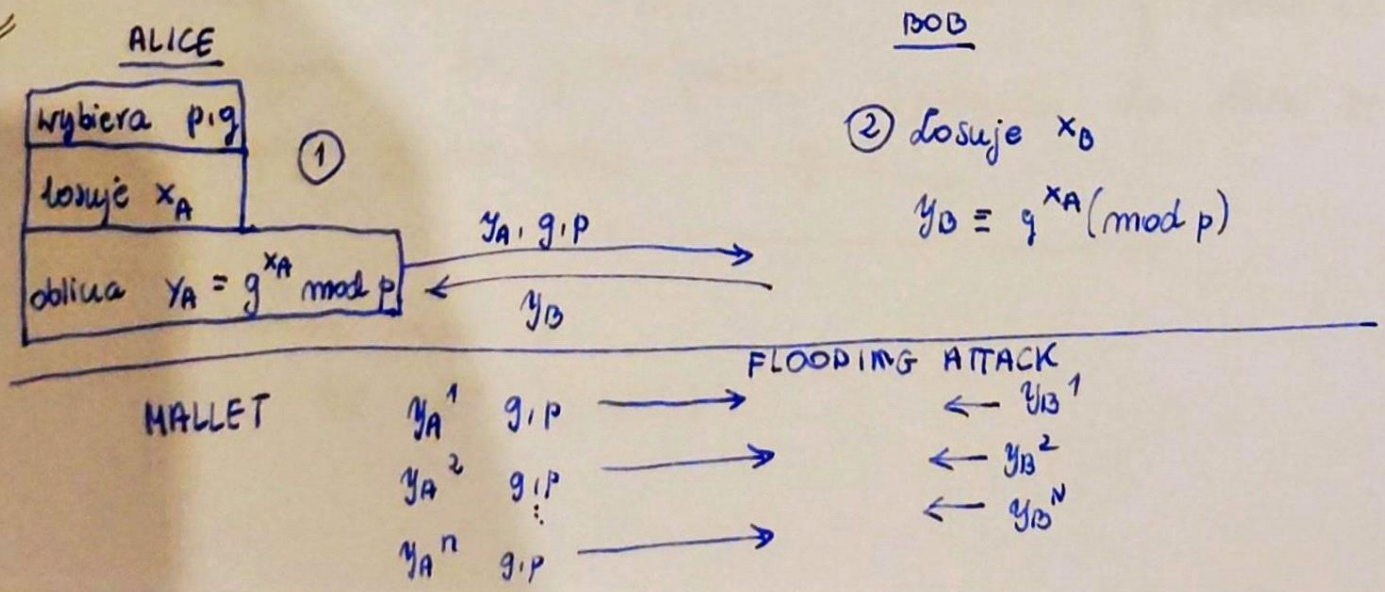
1) Ruch generowany przez jedną stronę

I 2) Startuje IKE tryb Main lub Agresive

UZGODNIENIE IKE SA

3) Startup IKE (quick mode) Uzgodnienie IPSEC SA

II 4) Dane są przesyłane protokołem AH i/lub ESD



DH. odporny na FLOODING

ALICE

Generuje C_A = cookie
$C_A = H(ip\ local, ip\ out, \dots)$

BOB

Generuje C_B = cookie

C_A →

← C_A, C_B

C_A, C_B, Y_A, Pig →

← $C_A, C_B, Y_B,$

1. mechanizm do DH na wodpornienie go

cd #

$K = y_B^{x_A} \pmod p$ $\qquad K' = y_A^{x_B} \pmod p$

$SIG_{k_A}(ID_A, K) = S_A$ $\qquad SIG_{k_B}(ID_B, K) = S_B$

S_A →

← S_B

OBIE STRONY MUSZĄ DIĘ PRZEKONAĆ ŻE NORMALNIE ZE SOBĄ

2. mechanizm do DH na wodpornienie go

IKE - Internet Key Exchange

FAZA I → ISAKMP

Main Mode, 6 trybów | Agressive Mode 3 tryb

FAZA II → Quick Mode

IPsec SA

ALICE — cookie — zestaw propozycji: — BOB

kryptografia asymetryczna

np. Proposal: Podpis cyfrowy
ENC = DES or 3DES
AUTH = MD5

Dwie strony mają K_{AB}

Proposal:
ENC = IDEA, AUTH = MD5

1) C_A, ISA_A →
2) ← C_B, C_A, ISA_B
3) wybiera się p, g, x_A; $y_A = g^{x_A} \pmod p$ → C_A, C_B, y_A, [p, g], N_A
4) ← C_A, C_B, y_B, N_B

N_A - nonce - no once
od 64 - 2048 bitów

te 4 tryby są kompletnie niezaszyfrowane → $SKEYID_e$

ID_A - identyfikator Alice
np. IPaddress, certyfikat, ...

5)* C_A, C_B, ID_A, $AUTH_A$ →
6)* ← C_A, C_B, ID_B, $AUTH_B$

sprawdzenie ID_B — sprawdzenie ID_A

$AUTH_A$ - metoda uwierzytelnienia
HMAC(Ckey, ...)
podany z kilku kluczy, który będzie do szyfrowania

FAZA II Quick Mode (wszystko szyfrowane kluczami)

KLUCZE

1) SKEYID - sposób generowania zależy od mechanizmu uwierzytelnienia

2) $SKEYID_d = H(SKEYID, t, C_A, C_B, 0)$ (0 zero)
3) $SKEYID_a = H(SKEYID, SKEYID_d, t, C_A, C_B, 1)$
4) $SKEYID_e = H(SKEYID, SKEYID_a, t, C_A, C_B, 2)$

$t = H(g_B^{x_A})$

1)* C_A, C_B, HASH(1), SA_A, N_A, y_A, [p,g], ID_A, ID_B →
2)* ← C_A, C_B, HASH(2), SA_B, N_B, y_B, [ID_A, ID_B]

$t = y_B^{x_A}$ — $t = y_A^{x_B}$

3)* C_A, C_B, HASH(3) →

KONIEC (z powodzeniem)

Preshared:
$SKEYID = H(K_{AB}, N_A, N_B)$

$AUTH_A$ to samo co $HASH_A = H(SKEYID_a, g_A, g_B, C_A, C_B, ISA_A, ID_A)$

$HASH_B = H([\perp \ldots], ID_B)$

$HASH(1) = H(SKEYID_a, SA_A, N_A, y_A, ID_A, ID_B)$
$HASH(2) = H(SKEYID_a, SA_B, N_A, N_B, y_B, ID_A, ID_B)$
$HASH(3) = H(SKEYID_a, 0(zero), N_A^{nowe}, N_B^{nowe})$

$NEWKEY = H(SKEYID_d, t, SPI, \underbrace{\text{Security Parameter Index}}_{\text{losowa liczba}}, \text{protokół (wynegocjowany z SA)}, N_A, N_B)$

z I FAZY

WYBORY ELEKTRONICZNE

1) Mogą głosować uprawnieni wyborcy
2) Nikt nie może głosować więcej niż 1 raz
3) Nikt nie może ustalić na kogo głosował inna osoba
4) Nikt nie może zmienić niczyjego głosu bez wyłączenia tego faktu
5) Wszyscy wyborcy mogą się upewnić, że ich głosy zostały policzone
6) Każdy wie o tym kto głosował, a kto nie

BSPI - wykład 30.01.2020

①

1) Głosować może uprawniony wyborca ?
2) Nikt nie może głosować, więcej niż jeden raz. +
3) — " — ustalić, jaki inny głosował +
4) — " — zmienić głos, a jeśli zmieni to zostanie to wykryte. (oddany zmienniti)
5) Wiadomo wyborca wie czy jego głos został podliczony. (głos został policzony)
6) Wiadomo widać kto głosował, a kto nie. — (NIE MAMY)

Głosowanie z wykorzystaniem ślepych podpisów.

Ślepy podpis

Alicja (chciała podpisać) M - wiad.

$K_B = (e, n)$ klucz publiczny

$k_B = (d, n)$ klucz do podpisu

Bob (wykonuje podpis)

1) Otrzymuje od Boba $K_B = (e, n)$
2) Losuje k, $(k, n) \equiv 1$, $k < n$
3) Zakrywa M tw. $y = M k^e \pmod n$
4) Wysyła y do Boba $\xrightarrow{\quad y \quad}$
5) Ślepo podpisuje y tw. $z = y^d \pmod n$
6) Wysyła do Alicji z $\xleftarrow{\quad z \quad}$
7) Odkrywa wiad. z

$S \equiv z k^{-1} \pmod n$

ZAŁ: można głosować tak/nie
T - tak
N - nic

KW, ?! BUDZI KONTROWERSJE

(1) Przygotowanie głosu

ALICE

a) losuje R_1, R_2 ← numery seryjne głosu

b) tworzy $M_1 = R_1 T$ ← konkatenacja

$M_2 = R_2 N$ ← !? $R_1 = R_2$?!

b) Zakrywa głosy M_1, M_2 w ten sposób do pkt. 3 poprzedniego protokołu dostaje y_1, y_2 (zakrycia głosów)

c) wysyła y_1, y_2 do KW — ID_A, y_1, y_2 →
dodatkowo wysyła np. PESEL (ID_A) a KW wie, że on/ona nie kiedy ona/on zagłosował

czy mogą sprawdzić czy ta osoba już zagłosowała

(2) Sprawdza bazę uprawnionych

ID	OTRZYMAŁA
ID_A	V
⋮	
ID_N	
⋮	

(3) Ślepo podpisuje głosy y_1, y_2 i odsyła Alice (otrzymuje Z_1, Z_2)

(4) Wysyła do Alice. (z_1, z_2)

← Z_1, Z_2

d) Alice zdejmuje zaciemnienie $(y_i^d \bmod n)$ oraz je wie który głos jest na TAK, który na nie. (S_1, S_2)

II Głosowanie

Alice (S_1, S_2) LE(TAK, NIE) = (1, 2)

1) Alice wybiera S_i

2) Szyfruje S_i do kluczem K_{KW} publicznym (może być innym)

$C_i = E_{K_{KW}}(S_i)$

3) Wysyła szyfrogram do komisji wyb.

KW

4) deszyfr. C_i

5) Sprawdza podpisy pod

6) Sprawdza czy numer seryjny się nie powtarza, jeśli to dodaje go do bazy

7) Publikuje wyniki (R, T).

Głosowanie z dwoma komisjami wyborczymi.

KWK
(Komisja wydająca Karty)

KW - Komisja zliczająca

I Wydanie kart do głosowania

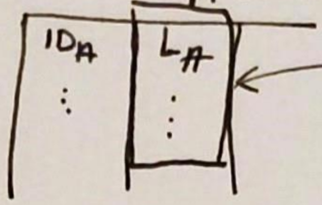
ALICE

1) Alice wysyła prośbę o (ID_A) o numer rejestracyjny

4) Alice zarejestrowała się

KWK

2) (Sprawdza czy jest w bazie jak nie to ją dodaje) Losuje L_A



po zamknięciu 5) rejestracji L_i wysyła do KW.

3) Wysyła L_A do Alice

II Dzień wyborów (Głosowanie)

Alice

1) Losuje I_A (jej losowa liczba podczas głosu własnej wyniku aby może sprawdzić swój głos)

2) Tworzy M (głos)

$M = L_A M_A I_A T$

3) Szyfruje M kluczem publicznym KW

$C = E_{K_{KW}}(M)$

4) Wysyła do KW „C".

KW

5) Deszyfruje C i w ten sposób dostaje $M = L_A I_A T$

sprawdza w bazie; dodaje do kolejnych głos

6) PUBLIKUJE WYNIKI

I_A T