

IPsec – IKE

Stefan Dziembowski

IPsec

IPsec to zestaw protokołów zabezpieczających komunikację IP.

Składa się z:

- protokołów zabezpieczających ruch pakietów, do wyboru:
 - Encapsulating Security Payload (ESP) — szyfrowanie i autentykacja
 - Authentication Header (AH) — autentykacja (bez szyfrowania)
rzadko używany
- protokołów uzgadniania klucza, obecnie dostępne tylko:
 - Internet Key Exchange (IKE) — **tym się dzisiaj zajmiemy!**

Historia IPsec

Autorem IPsec jest specjalna grupa robocza organizacji **The Internet Engineering Task Force** [link].

Pierwsza wersja standardu powstała w 1998.

Protokół jest pod silnym wpływem wcześniejszych protokołów:

- Oakley [RFC2412],
- SKEME [Krawczyk 1995]

IPsec — wady



Częste zarzut wobec IPsec

- Jest zbyt skomplikowany.
- Trudno zrozumieć dokumentację.

Morał [Ferguson i Schneier]

Protokoły kryptograficzne nie powinny być tworzone przez komitety.

SSL/TLS vs. IPsec

- Inna warstwa:

Aplikacji
SSL
Transportu (np. TCP)
Sieci (np. IP)
Łącza

Aplikacji
Transportu (np. TCP)
Sieci (np. IPsec)
Łącza

- IPsec jest znacznie bardziej skomplikowany:
 - ma więcej opcji,
 - dba o więcej rzeczy (Perfect Forward Security, bardzo silna anonimowość).
- IPsec ma z reguły autentykację w dwie strony
- IPsec jest rozwijany przez komitet.

Perfect Forward Security

Interesuje nas **Perfect Forward Security (PFS)**.

Co to jest ?

Chodzi o maksymalną minimalizację konsekwencji wycieku klucza.

Założmy, że klucz K został w pewnym momencie użyty do generacji klucza sesyjnego K_i . Jeśli potem K wycieknie to przeciwnik nie powinien uzyskać żadnej informacji na temat K_i . Nawet jeśli wyciekną też inne klucze sesyjne.

Przykład kiedy nie ma PFS

E_B — szyfrowanie z kluczem publicznym Boba

Alicja

Bob

losuje klucz K $\xrightarrow{E_B(K)}$ odszyfrowuje klucz K
(warto by jeszcze jakoś zapewnić świeżość klucza...)

Jeśli klucz prywatny Boba wycieknie, to koniec...

Czy PFS jest potrzebne

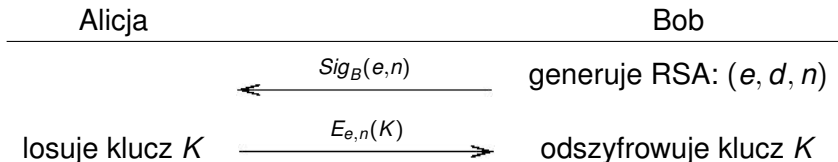
Jeśli zależy nam na wydajności, możemy pominąć wymaganie PFS.

Jest tak zwłaszcza w przypadkach kiedy:

- interesuje nas tylko **autentykacja** a nie tajność
- przekazywane dane nie muszą pozostawać zbyt długo tajne.

Jak zapewnić PFS?

Dodajemy jeszcze więcej losowości:



Przydałoby się jeszcze dodać autentykację.

Problem

To rozwiązanie jest bardzo niewydajne.
(generacja kluczy RSA obejmuje znajdowanie liczb pierwszych,
etc..)

Lepsze rozwiązanie: Diffie-Hellman

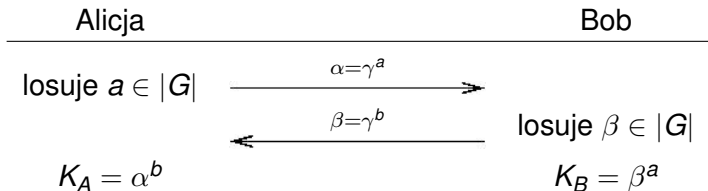
Użyjemy protokołu Diffiego-Hellmana.

G — multiplikatywna grupa cykliczna

γ — generator G

Logarytm dyskretny $\log_{\gamma} x = a$ jeśli $\gamma^a = x$

Zakładamy, że wyznaczenie tego logarytmu dyskretnego jest obliczeniowo trudne.



Powinno wyjść: $K_A = K_B$.

Przeciwnik widzi tylko $\alpha = \gamma^a$ i $\beta = \gamma^b$

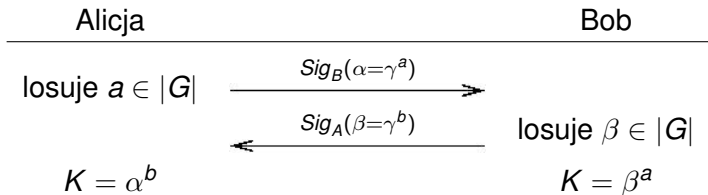
Jak wybrać G

Najczęściej są to:

- grupy Z_p^* , gdzie p — pierwsza (często na dodatek: $p = 2q + 1$).
- grupy oparte na na krzywych eliptycznych.

Autentykowany protokół Diffiego-Hellmana

Trzeba jeszcze dodać autentykację (od razu: w dwie strony)



Przy okazji dostaliśmy świeżość klucza (i to z punktu widzenia zarówno Alicji jaki i Boba)!

Ataki DoS

Co z atakami Denial of Service na protokół Diffiego-Hellmana?

Atak DoS na Boba (w DH)

(Złośliwa) Alicja wysyła po prostu losowe $\alpha \in G$ (a nie: $\alpha = \gamma^G$).

Kosztuje ją to niewiele wysiłku.

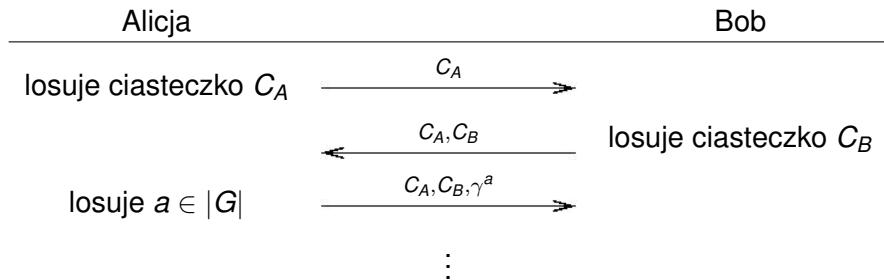
(Uczciwy Bob) wykouje następujące czynności

- 1 oblicza $\beta \in |G|$,
- 2 odsyła β Alicji,
- 3 oblicza $K = \beta^a$.

Kosztuje to Boba znacznie więcej wysiłku!

Częściowe remedium na DoS

Weryfikujemy poprawność adresu IP nadawcy (Alicji) za pomocą **ciasteczek**.



W ten sposób przynajmniej znamy adres IP z którego pochodzą te ataki (chyba, że przecinik może monitorować nasz ruch wychodzący).

Działanie w różnych trybach

Skąd pochodzą klucze początkowe?

Wymagamy, by system działał w

- 1 Modelu z kluczem publicznym
- 2 Modelu z dzielonym kluczem symetrycznym (uzyskanym za pomocą metod typu Kerberos albo wpisanym ręcznie).

Negocjowanie parametrów systemu

Na początku komunikacji obie strony muszą uzgodnić parametry (rodzaj szyfrów, schematów autentykacji, grupa G , czy interesuje nas PFS, częstotliwość odświeżania kluczy etc).

Nie można tego zrobić bez autentykacji (bo man-in-the-middle mógłby ustalić te parametry za Alicję i Boba).

Ale jak zrobić autentykację zanim jeszcze ustaliliśmy parametry kryptograficzne?

Zrobimy tak:

- 1 najpierw Alicja i Bob ustalą wszystko bez autentykacji,
- 2 potem Alicja i Bob sprawdzą czy ustalili to samo.

Anonimowość

Pełna anonimowość jest trudna do osiągnięcia.
Czasami (w przypadku dynamicznych numerów IP) jest
możliwa częściowa anonimowość nadawcy.

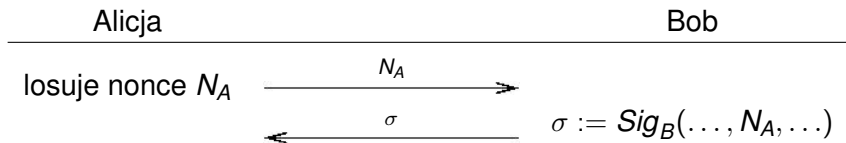
Nie chcemy by nasz protokół zniszczył tę anonimowość.

Poza tym:

Nie chcemy by zapis konwersacji mógł posłużyć jako dowód, że
się z kimś komunikowaliśmy.

Jak taki dowód Alicja może uzyskać?

Założmy, że protokół wymaga by Bob podpisał jakąś wiadomość zawierającą *nonce* wybrany przez Alicję



Jeśli (złośliwa) Alicja wybierze $N_A :=$ „Jestem Alicją”, to potem σ może być dowodem, że Alicja kontaktowała się z Bobem.

Rozwiązanie

Rozwiązanie tego problemu jest takie [Krawczyk 95]

Zamiast podpisu użyjemy szyfrowania/desyfrowania.

Bob chce udowodnić swoją tożsamość Alicji

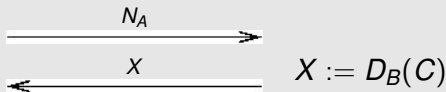
(E_B, D_B — szyfrowanie i odszyfrowywanie kluczami Boba)

Alicja

Bob

losuje nonce N_A , zapisuje
je w specjalnym formacie i
szyfruje $C := E_B(N_A)$

$$X \stackrel{?}{=} N_A$$



Periodyczne odświeżanie klucza

Protokół ustalania klucza jest złożony i wymaga sporo zasobów.

Nie opłaca się go wykonywać za każdym razem gdy chcemy odświeżyć klucz sesyjny.

Dlatego zdecydowano się na rozwiązanie dwufazowe

- **Faza I** jest wykonywana najpierw. Jest bardziej kosztowna, ale oferuje większe możliwości.
Może zacząć się od wpisanych kluczy symetrycznych, albo kluczy publicznych.
- **Faza II** jest wykonywana potem i może być wykonywana np. co parę minut. Jest tania i oferuje mniej możliwości (zmiany parametrów). Punktem wyjścia są klucze powstałe w Fazie II.

Najbardziej podstawowe terminy

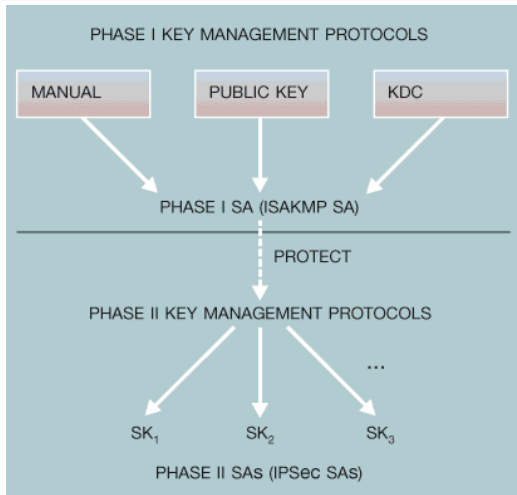
- **Security Domain of Interpretation (DOI)** — określa szczegółową składnię komunikatów.
- **Internet Security Association and Key Management Protocol (ISAKMP)**
- **Internet Key Exchange Protocol (IKE).**

Oprócz tego ważnym pojęciem jest:

Security association (SA)

- meta-parametry (algorytmy kryptograficzne, zasady odswieżania, identyfikator, etc.),
- tożsamości uczestników,
- uzgodnione klucze.

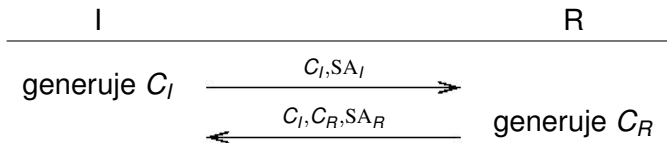
Ogólny schemat



ISAKMP SA —
Security Association
wyprodukowane przez
Fazę I („pośrednie”)

IPsec SA — **Security Association**
wyprodukowane przez
Fazę II („do użytku
zewnętrznego”)

Faza I (część 1)



SA_I jest listą propozycji szyfrów i metod autentykacji.
 SA_R jest wskazaniem która propozycja została wybrana.

SA_I

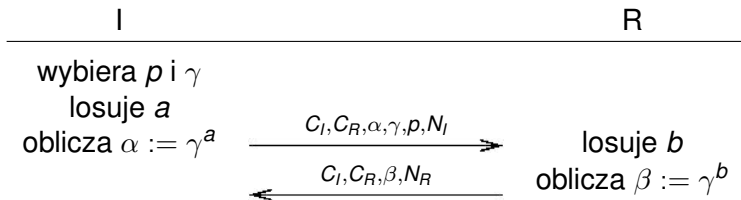
Proposal: ENC = DES or 3DES, AUTH = MD5

Proposal: ENC = IDEA, AUTH = SHA-1

SA_R

Proposal: ENC = 3DES, AUTH = MD5

Faza I (część 2)



N_I, N_R — nonce

Co to znaczy „wybiera p i γ ?

To znaczy: wybiera identyfikator grupy ze standardu.

w standardzie są grupy **MODP** — zwykłe Z_p

i grupy **EC2N** — grupy oparte na krzywych eliptycznych

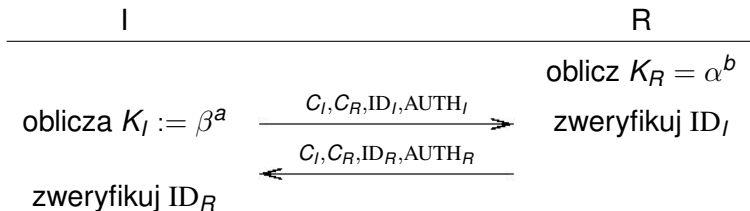
Przykład: grupa MODP o numerze 14

$p =$

```
FFFFFFFF FFFFFFFF C90FDAA2 2168C234 C4C6628B 80DC1CD1
29024E08 8A67CC74 020BBEA6 3B139B22 514A0879 8E3404DD
EF9519B3 CD3A431B 302B0A6D F25F1437 4FE1356D 6D51C245
E485B576 625E7EC6 F44C42E9 A637ED6B 0BFF5CB6 F406B7ED
EE386BFB 5A899FA5 AE9F2411 7C4B1FE6 49286651 ECE45B3D
C2007CB8 A163BF05 98DA4836 1C55D39A 69163FA8 FD24CF5F
83655D23 DCA3AD96 1C62F356 208552BB 9ED52907 7096966D
670C354E 4ABC9804 F1746C08 CA18217C 32905E46 2E36CE3B
E39E772C 180E8603 9B2783A2 EC07A28F B5C55DF0 6F4C52C9
DE2BCBF6 95581718 3995497C EA956AE5 15D22618 98FA0510
15728E5A 8AACAA68 FFFFFFFF FFFFFFFF
```

$\gamma = 2$

Faza I (część 3)



Te wiadomości są zaszyfrowane (jakim kluczem — o tym za chwilę)

O tym co to jest AUTH i co to znaczy „zweryfikuj ID” — za chwilę

Zauważmy, że ID są zaszyfrowane.

Tryby

Wyróżniamy takie tryby autentykacji:

- z kluczem dzielonym,
- z podpisami,
- z szyfrowaniem (2 tryby).

Każdy z nich może być w wersji zwykłej albo **agresywnej**.

Derywacja kluczy

$\text{PRF}(\text{klucz}, \text{dane})$ — funkcja haszująca z kluczem (inaczej: MAC, albo PRF)

$K = K_I = K_R$ — klucz uzgodniony za pomocą DH.

SKEYID — określimy później.

$\text{SKEYID}_d = \text{PRF}(\text{SKEYID}, K|C_I|C_R, 0)$ — do użytku późniejszego

$\text{SKEYID}_a = \text{PRF}(\text{SKEYID}, \text{SKEYID}_d|C_I|C_R, 1)$ — do autentykacji

$\text{SKEYID}_e = \text{PRF}(\text{SKEYID}, \text{SKEYID}_a|K|C_I|C_R, 2)$ — do szyfrowania

$\text{HASH}_I = \text{PRF}(\text{SKEYID}_a, \alpha|\beta|C_I|C_R|SA_I|ID_I)$

$\text{HASH}_R = \text{PRF}(\text{SKEYID}_a, \beta|\alpha|C_I|C_R|SA_I|ID_R)$

Tryb z kluczem dzielonym

PSKEY — klucz dzielony

$$\text{SKEYID} := \text{PRF}(\text{PSKEY}, N_I | N_R)$$

$$\text{AUTH}_I := \text{HASH}_I$$

$$\text{AUTH}_R := \text{HASH}_R$$

Wady

- Trzeba znać ID rozmówcy z góry.
- Kiepsko się skaluje.

Tryb z podpisami

$$\text{SKEYID} := \text{PRF}(N_I | N_R, K)$$

$$\text{AUTH}_I := \text{CERT}_I, \text{SIG}_I$$

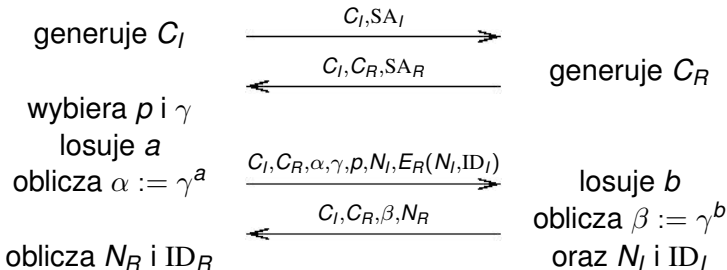
$$\text{AUTH}_R := \text{CERT}_I, \text{SIG}_R,$$

Gdzie SIG_I i SIG_R są podpisami na HASH_I i HASH_R , odpowiednio.

Wady

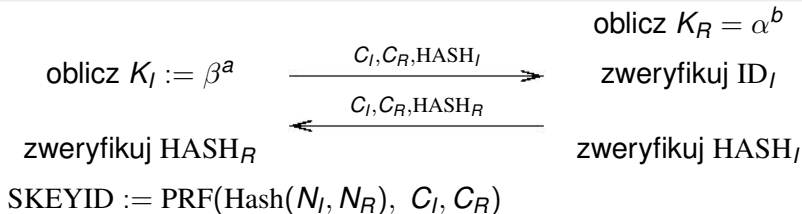
Brak pełnej anonimowości.

Tryb z szyfrowaniem [1/2]



Nie wysyłamy certyfikatów, bo trzeba by to zrobić tekstem jawnym

Tryb z szyfrowaniem [2/2]



Wady

- Może nie być wiadomo którego klucza użyć (jak ma się kilka)
- Aż 4 szyfrowania z kluczem publicznym!

Istnieje wersja ulepszona (tylko 2 szyfrowania).

Wersja agresywna [1/2]

Dla przykładu rozważymy **wersję agresywną trybu z kluczem dzielonym**.

Z czego rezygnujemy?

- Nie ma utajnienia tożsamości uczestników.
- Nie ma możliwości negocjacji grupu do protokołu Diffiego-Hellmana.

Wersja agresywna [2/2]

zaloguj g i p
wygeneruj C_I, x

oblicz $K_I = \beta^a$ $\xrightarrow{C_I, C_R, \text{HASH}_I}$ oblicz $K_R = \alpha^b$

Potem $K (= K_I = K_R)$ jest używane do wyprodukowania kluczy (tak jak w wersji nieagresywnej).

Tryb szybki (wersja bez PFS) [1/2]

Komunikacja szyfrowowana kluczami ustalonymi w poprzedniej fazie.

M_{ID} — Identyfikator wiadomości.

$\text{HASH}(1) := \text{PRF}(\text{SKEYID}_a, M_{ID}|SA_I|N_I)$

$\text{HASH}(2) := \text{PRF}(\text{SKEYID}_a, M_{ID}|SA_R|N_I|N_R)$

$\text{HASH}(3) := \text{PRF}(\text{SKEYID}_a, 0|M_{ID}|N_I|N_R)$

oblicz HASH(1)	$\xrightarrow{C_I, C_R, \text{HASH}(1), SA_I, N_I}$	sprawdź HASH(1)
sprawdź HASH(2)	$\xleftarrow{C_I, C_R, \text{HASH}(2), SA_R, N_R}$	
	$\xrightarrow{C_I, C_R, \text{HASH}(3)}$	sprawdź HASH(3).

Tryb szybki (wersja bez PFS) [2/2]

Uzgodnionym kluczem jest

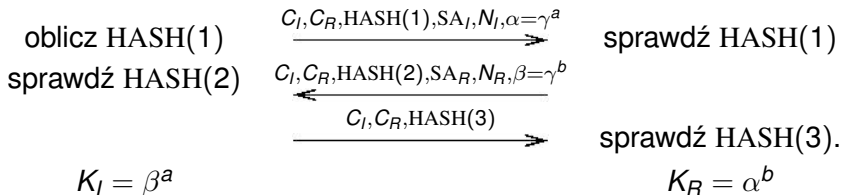
$$\text{KEYMAT} := \text{PRF}(\text{SKEYID}_d, \text{protocol}|\text{SPI}|\text{N}_I|\text{N}_R)$$

protocol — identyfikator protokołu
SPI — security parameter index

Jak uzyskać PFS?

Dodamy protokół Diffiego-Hellmana,

Tryb szybki(z PFS)



$(K = K_I = K_R)$

Wartości HASH też trzeba zmodyfikować.

Teraz:

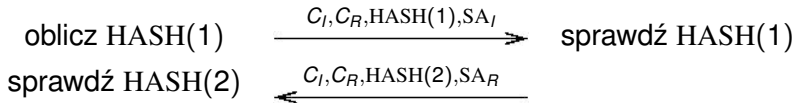
$\text{KEYMAT} := \text{PRF}(\text{SKEYID}_d, K | \text{protocol} | \text{SPI} | N_I | N_R)$

Tryb wyboru nowej grupy

Służy tylko zmianie grupy na potrzeby przyszłej wymiany kluczy (w trybie szybkim).

$\text{HASH}(1) := \text{PRF}(\text{SKEYID}_a, M_{\text{ID}} | \text{SA}_I)$

$\text{HASH}(2) := \text{PRF}(\text{SKEYID}_a, M_{\text{ID}} | \text{SA}_R)$



Reszta literatury

- Michael S. Borella. Methods and Protocols for Secure Key Negotiation Using IKE
- P.-C. Cheng. An architecture for the Internet Key Exchange Protocol