Rozszerzone protokoły Diffiego-Hellmana

Aleksandra Kamińska, na podstawie:

1. “An authenticated Diffie-Hellman Key Agreement Protocol“ Shouichi Hirose, Susumu Yoshida
2. “Enhanced of Key Agreement Protocols Resistant to a Denial-of-Service Attack”, Min-Shiang Hwang, Jung-Wen Lo, Chia-Hsin Liu

Rozszerzone protokoły Diffiego-Hellmana

Tytułem wstępu:

Rozszerzone protokoły Diffiego-Hellmana to określenie na protokoły uzgadniania klucza oparte o schemat Diffiego-Hellmana. Protokół Diffiego-Hellmana nie jest odporny na atak "z człowiekiem pośrodku". Ten referat ma na celu zapoznanie słuchaczy z wybranymi protokołami opartymi o schemat DH, odpornymi na takie ataki, jak również z protokołami odpornymi na atak denial-of-service.

Zagadnienia referatu:

- protokół Diffiego-Hellmana
- protokół odporny na aktywne i pasywne ataki – KAP
- protokół Hiros-Matsura
- protokoły odporne na ataki DoS – KAP1 i KAP2

Podstawowy protokół Diffiego-Hellmana

(za "*Handbook of Applied Cryptograpy*")

Protocol Diffie-Hellman key agreement (basic version)

SUMMARY: A and B each send the other one message over an open channel.
RESULT: shared secret K known to both parties A and B.

1. *One-time setup*. An appropriate prime p and generator α of $\mathbb{Z}_p^*$ ($2 \leq \alpha \leq p-2$) are selected and published.
2. *Protocol messages*.

$$A \rightarrow B : \alpha^x \bmod p \quad (1)$$
$$A \leftarrow B : \alpha^y \bmod p \quad (2)$$

3. *Protocol actions*. Perform the following steps each time a shared key is required.
   (a) A chooses a random secret x, $1 \leq x \leq p-2$, and sends B message (1).
   (b) B chooses a random secret y, $1 \leq y \leq p-2$, and sends A message (2).
   (c) B receives α^x and computes the shared key as $K = (\alpha^x)^y \bmod p$.
   (d) A receives α^y and computes the shared key as $K = (\alpha^y)^x \bmod p$.

Podstawowy protokół Diffiego-Hellmana

Cechy:

- Pierwsze praktyczne rozwiązanie problemu dystrybucji klucza
- Odporny na ataki pasywne
- Nieodporny na ataki aktywne – brak autentykacji kluczy/ przesyłanych informacji
- Bezpieczeństwo protokołu opiera się na złożoności obliczeniowej problemu Diffiego-Hellmana, a co za tym idzie – na złożoności problemu logarytmu dyskretnego.

Protokół KAP

Autorzy: Schouichi Hirose, Susumu Yoshida

Obserwacja:

W protokole Diffiego-Hellmana, jeżeli informacja, którą dostaje strona A gwarantuje, że

- B otrzymał $u_A=\alpha^x$
- B wysłał $u_B=\alpha^y$ w odpowiedzi na $u_A=\alpha^x$
- B zna x – logarytm dyskretny u_B,

to B rzeczywiście jest w stanie obliczyć $K_B=K_A=K$.

Protokół KAP

Założenia:

p, q – duże liczby pierwsze, t.że $q \mid p-1$;

g – element grupy cyklicznej rzędu p (czyli GF(p)), o rzędzie qk, dla pewnego k naturalnego.

h – bezkolizyjna funkcja haszująca, z przeciwdziedziną Z_q

s_i – element Z_q, klucz prywatny strony i

$v_i = g^{-s_i} \bmod p$ – klucz publiczny uzytkownika i.

I_i – reprezentuje identyfikator użytkownika

p,q,g,h,v_i są znane publiczne, i w szczególności przez wszystkich użytkowników

Protokół KAP

0. (Precomputation)
   The initiator A randomly selects $k_A, r_A \in \mathbf{Z}_q$ and computes $u_A = g^{-k_A} \bmod p$ and $x_A = g^{r_A} \bmod p$.
   The responder B randomly selects $k_B, r_B \in \mathbf{Z}_q$ and computes $u_B = g^{-k_B} \bmod p$ and $x_B = g^{r_B} \bmod p$.

1. A sends u_A to B.

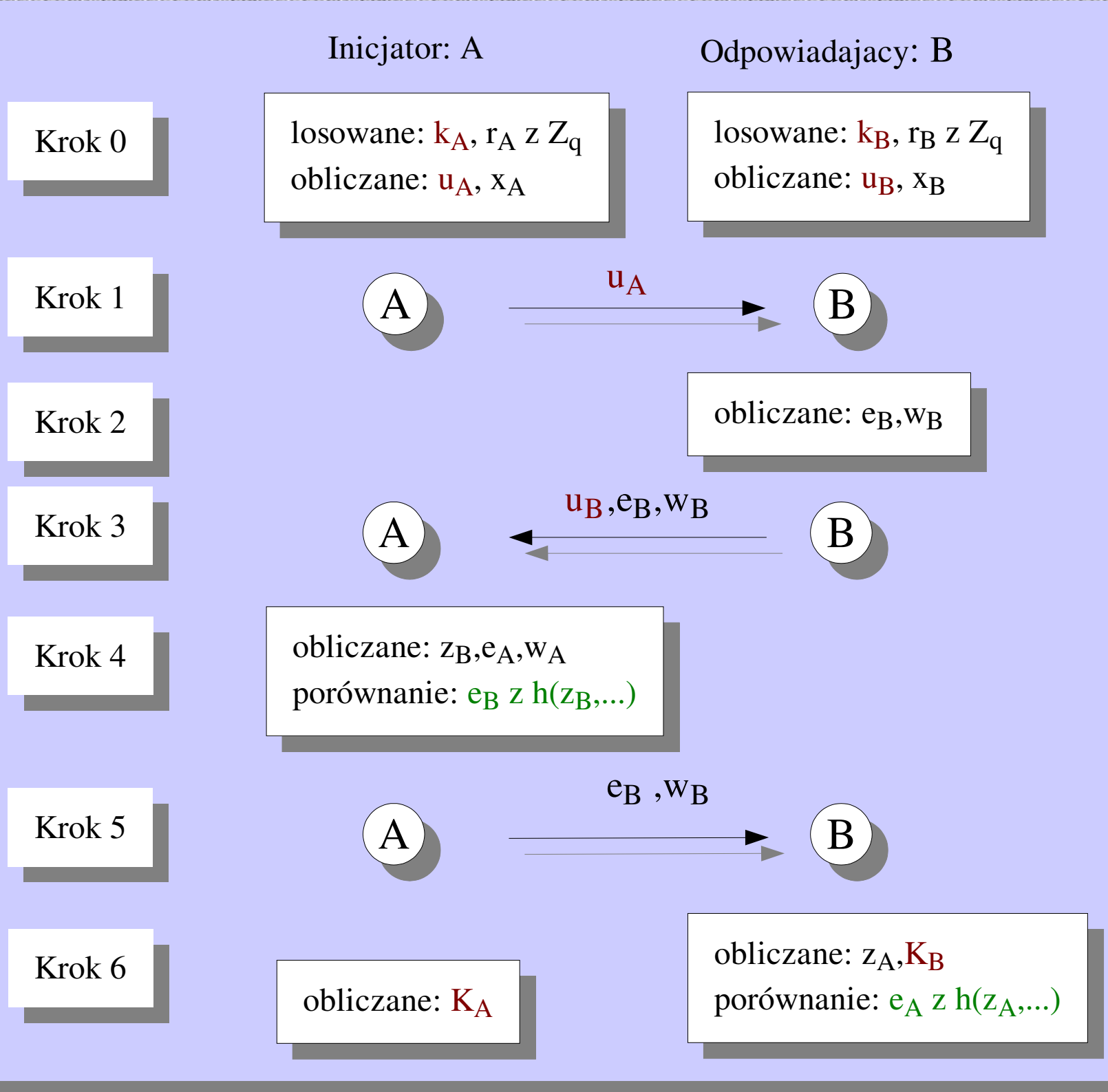
2. B computes $e_B = h(x_B, u_B, u_A, I_B, I_A)$ and $w_B = r_B + e_B k_B + {e_B}^2 s_B \bmod q$.

3. B sends u_B, e_B, w_B to A.

4. A computes $z_B = g^{w_B} {u_B}^{e_B} {v_B}^{{e_B}^2} \bmod p$ and checks if $e_B = h(z_B, u_B, u_A, I_B, I_A)$. If it does not hold, then A terminates the execution. Otherwise, A computes $e_A = h(x_A, u_A, u_B, I_A, I_B)$ and $w_A = r_A + e_A k_A + {e_A}^2 s_A \bmod q$.

5. A sends e_A, w_A to B.

6. A computes $K_A = {u_B}^{-k_A} \bmod p$.
   B computes $z_A = g^{w_A} {u_A}^{e_A} {v_A}^{{e_A}^2} \bmod p$ and checks if $e_A = h(z_A, u_A, u_B, I_A, I_B)$. If it does not hold, then B terminates the execution. Otherwise, B computes $K_B = {u_A}^{-k_B} \bmod p$.


Inicjator: A
Odpowiadajacy: B
Krok 0
losowane: k_A, r_A z Z_q
obliczane: u_A, x_A
losowane: k_B, r_B z Z_q
obliczane: u_B, x_B
Krok 1
A
u_A
B
Krok 2
obliczane: e_B,w_B
Krok 3
A
u_B,e_B,w_B
B
Krok 4
obliczane: z_B,e_A,w_A
porównanie: e_B z $h(z_B,...)$
Krok 5
A
e_B ,w_B
B
Krok 6
obliczane: K_A
obliczane: z_A,K_B
porównanie: e_A z $h(z_A,...)$

Idea stojąca za protokołem KAP

Wiadomość (u_B,e_B,w_B) można traktować jako podpis (sygnaturę) dla informacji (u_A,I_B,I_A). Ta sygnatura zawiera informację gwarantującą, że strona B jest w stanie obliczyć wspólny klucz ze stroną A – zgodnie z obserwacją dotyczącą protokołu Diffiego-Hellmana, zawiera informację że strona B dostała wiadomość u_A, że u_B jest odpowiedzią strony B na u_A, oraz że strona B zna $(-k_B)$, czyli logarytm dyskretny u_B.

$e_B = h(x_B,u_B,u_A,I_B,I_A)$

$w_B = r_B + e_B k_B + e_B^2 s_B \bmod q,$

z własności funkcji h e_B identyfikuje jednoznacznie (u_A,I_B,I_A), natomiast wykorzystanie w w_B klucza prywatnego strony B zapewnia że wiadomość (u_B,e_B,w_B) pochodzi od strony B.

Warto zwrócić uwagę, że klucze K_A i K_B są potwierdzane bez wyliczania tychże kluczy.

Zagadnienia złożone obliczeniowo,
istotne dla protokołu KAP

Problem logarytmu dyskretnego (DLP)

Mając daną liczbę pierwszą p, generator α grupy Z_p^*, oraz element β tej grupy, znaleźć liczbę całkowitą x, $0 \le x \le p-2$, t.że $\alpha^x \equiv \beta \pmod p$

Uogólniony problem logarytmu dyskretnego (GDLP)

Mając daną skończoną cykliczną grupę G rzędu n, generator α grupy G, oraz element β tej grupy, znaleźć liczbę całkowitą x, $0 \le x \le n-1$, t.że $\alpha^x = \beta$

Problem Diffiego-Hellmana

Mając daną liczbę pierwszą p, generator g grupy Z_p^*, oraz elementy ($g^a \bmod p$) i ($g^b \bmod p$), znaleźć ($g^{ab} \bmod p$)

Dzielenie klucza bez komunikacji

Korzystając ze schematu redundantnego podpisu (czyli schematu używanego w protokole KAP), można doprowadzić do dzielenia klucza bez komunikacji między stronami. W poniższym schemacie korzysta się jedynie z publicznie dostępnych informacji.

First, each user i follows the next procedure.

1. He selects $k_i \in \mathbf{Z}_q$ at random and computes $u_i = g^{-k_i} \bmod p$. He also determines $ExpDate_i$, the expiration date of u_i.

2. He selects $r_i \in \mathbf{Z}_q$ at random and computes $x_i = g^{r_i} \bmod p$. Then, he computes $e_i = h(x_i, u_i, I_i, ExpDate_i)$ and $w_i = r_i + e_i k_i + e_i{}^2 s_i \bmod q$.

3. He writes $(I_i, ExpDate_i, u_i, e_i, w_i)$ in publicly accessible board.

When A would like to share a session-key with B, first A checks the validity of $(I_B, ExpDate_B, u_B, e_B, w_B)$. If it is valid, then A computes $K = {u_B}^{-k_A} \bmod p$.

Jednoczesne uzgadnianie wielu kluczy

Protokół KAP można również, wprowadzając drobne modyfikacje, stosować do uzgadniania wielu kluczy (MKAP). MKAP można traktować jako ciekawostkę – autorzy twierdzą że nie znaleźli jeszcze dobrego zastosowania dla tej wersji protokołu.

0. (Precomputation)
   A randomly selects $k_{A1}, k_{A2}, \ldots, k_{An} \in \mathbf{Z}_q$ and computes $u_{Ai} = g^{-k_{Ai}} \bmod p$ for $i = 1, 2, \ldots, n$. A also selects $r_A \in \mathbf{Z}_q$ at random and computes $x_A = g^{r_A} \bmod p$.

   B randomly selects $k_{B1}, k_{B2}, \ldots, k_{Bn} \in \mathbf{Z}_q$ and computes $u_{Bj} = g^{-k_{Bj}} \bmod p$ for $j = 1, 2, \ldots, n$. B also selects $r_B \in \mathbf{Z}_q$ at random and computes $x_B = g^{r_B} \bmod p$.

1. A sends $u_{A1}, u_{A2}, \ldots, u_{An}$ to B.

2. B computes $e_B = h(x_B, u_{B1}, \ldots, u_{Bn}, u_{A1}, \ldots, u_{An}, I_B, I_A)$ and $w_B = r_B + \sum_{j=1}^{n} {e_B}^j k_{Bj} + {e_B}^{n+1} s_B \bmod q$.

Jednoczesne uzgadnianie wielu kluczy

3. B sends $u_{B1}, u_{B2}, \ldots, u_{Bn}, e_B, w_B$ to A.

4. A computes $z_B = g^{w_B} \left(\prod_{j=1}^{n} u_{Bj}^{e_B{}^j} \right) v_B{}^{e_B{}^{n+1}} \bmod p$ and checks if $e_B = h(z_B, u_{B1}, \ldots, u_{Bn}, u_{A1}, \ldots, u_{An}, I_B, I_A)$. If it does not hold, then A terminates the execution. Otherwise, A computes $e_A = h(x_A, u_{A1}, \ldots, u_{An}, u_{B1}, \ldots, u_{Bn}, I_A, I_B)$ and $w_A = r_A + \sum_{i=1}^{n} e_A{}^i k_{Ai} + e_A{}^{n+1} s_A \bmod q$.

5. A sends e_A, w_A to B.

6. A computes $K_{Ai} = u_{Bi}{}^{-k_{Ai}} \bmod p$ for $i = 1, \ldots, n$.

   B computes $z_A = g^{w_A} \left(\prod_{i=1}^{n} u_{Ai}{}^{e_A{}^i} \right) v_A{}^{e_A{}^{n+1}} \bmod p$ and checks if $e_A = h(z_A, u_{A1}, \ldots, u_{An}, u_{B1}, \ldots, u_{Bn}, I_A, I_B)$. If it does not hold, then B terminates the execution. Otherwise, B computes $K_{Bj} = u_{Aj}{}^{-k_{Bj}} \bmod p$ for $j = 1, \ldots, n$.

KAP odporny na atak denial-of-service

Atak denial-of-service (DoS) ma na celu wyczerpanie zasobów serwera oraz “zapchanie” łącza do serwera, w celu uniemożliwienia dostępu do serwera uczciwym użytkownikom.

Użytkownik przeprowadzający atak zgłasza jak najwięcej żądań uzgodnienia klucza, które następnie pozostawia bez odpowiedzi. W ten sposób złośliwy użytkownik chce wyczerpać zasoby obliczeniowe (wiele kosztownych obliczeń) lub pamięciowe (przechowywanie stanu operacji w trakcie uzgadniania) serwera.

Do schematów, które można stosować w celu zapewnienia odporności na atak DoS należą m.in.: korzystanie z ciasteczek (cookies), bezstanowe połączenia (informacje niezbędne w każdym kroku są pozyskiwane z informacji przesyłanych między stronami), obciążanie klienta wstępnymi kosztami (w celu nawiązania połączenia klient musi wykonać pewne – stosunkowo kosztowne – obliczenia; serwer może sprawdzić poprawność obliczeń).

Serwer może również wysyłać klientowi informacje, z użyciem których klient musi wykonać obliczenia i odpowiedzieć, a następnie łatwo weryfikować poprawność odpowiedzi.

KAP odporny na atak denial-of-service

W protokole KAP najbardziej kosztowną (ze względu na czas) operacją jest potęgowanie modulo p. Dlatego ważne jest, aby odkładać wyliczanie wyrażeń z potęgowaniem (z_A) do momentu, gdy serwer jest pewien tożsamości klienta.

DoS-resistant KAP Steps 0 through 4 are same as those of KAP. Step 5 and 6 of this protocol is as follows.

5. A sends z_B, e_A, w_A to B.
6. A computes $K_A = {u_B}^{-k_A} \bmod p$.

   B checks if $z_B = x_B$. If it does not hold, then B terminates the execution. Otherwise, B computes $z_A = g^{w_A} {u_A}^{e_A} {v_A}^{{e_A}^2} \bmod p$ and checks if $e_A = h(z_A, u_A, u_B, I_A, I_B)$. If it does not hold, then B terminates the execution. Otherwise, B computes $K_B = {u_A}^{-k_B} \bmod p$.

Bezpieczeństwo protokołu KAP

Można udowodnić odporność protokołu KAP na aktywne i pasywne ataki w modelu z wyrocznią (losową).

Odporoność na ataki pasywne oparta jest na trudności problemu Diffiego-Hellmana.

Przy dowodzie odporności na ataki aktywne korzysta się z trudności problemu logarytmu dyskretnego.

Ataki rozważane dla KAP:

- bierne podłuchiwanie
- “zakłócanie” - atak aktywny
- współdzielenie nieznanego klucza – także atak aktywny

Podsłuchiwanie KAP

Problem pasywnego podsłuchiwania (PEP)

input: $p, q, g, I_A, I_B, (s_A, v_A), (s_B, v_B), (u_A; u_B, e_B, w_B; e_A, w_A)$, where

- (s_A, v_A) and (s_B, v_B) are randomly selected keys of A and B, respectively,
- $(u_A; u_B, e_B, w_B; e_A, w_A)$ is messages exchanged in (A, B).

output: $u_A^{\log_g u_B \bmod p} \bmod p$.

Niech t_{ver} – ilość kroków przy weryfikacji podpisu. Istnieje twierdzenie:

Jeśli istnieje probabilistyczna maszyna Turinga M, która rozwiązuje problem PEP w co najwyżej t krokach z prawdopodobieństwem co najmniej ε, to istnieje probabilistyczna maszyna Turinga M rozwiązująca DHP w co najwyżej $(t+2t_{ver})$ krokach z prawdopodobieństwem co najmniej ε.

Podsłuchiwanie KAP

Dowód:

Dla instancji problemu DHP (p,q,g,a,b) maszyna M wykonuje następujące kroki:

1. Przyjmuje $u_A=g^a$ oraz $u_B=g^b$.
2. Losuje pary (s_A,v_A) oraz (s_B,v_B).
3. Losuje e_B,w_B (z Z_q), oblicza $x_B=g^{w_B}u_B^{e_B}v_B^{e_B{}^2}$, określa $h(x_B,u_B,u_A,I_B,I_A)=x_B$
4. Wykonuje punkt czwarty dla danych z indeksami A.
5. Symuluje M' z wejściem $p,q,g,I_A,I_B,(s_A,v_A),(s_B,v_B),(u_A;u_B,e_B,w_B;e_A,w_A)$.
6. Podaje na wyjściu wynik maszyny M'.

Zakłócanie KAP

Zakłócanie (interference) to aktywny atak, w czasie którego zmieniane są wiadomości wymieniane przez uczestników. Ma na celu uzyskanie informacji o kluczach prywatnych, albo doprowadzenie do nieuzgodnienia klucza przez strony.

Zdefiniujemy "Problem fałszowania sygnatury(SFP)", który okazuje się być bardziej złożony niż problem logarytmu dyskretnego:

wejście: p,q,g,v, gdzie v jest losowo wybrane spośród $\{g^0,...,g^{q-1}\}$

wyjście: Msg,u,e,w, gdzie Msg jest wiadomością, a (u,e,w) jest prawidłową sygnaturą dla Msg, ze względu na klucz publiczny v.

Lemma 1 If there exists some probabilistic Turing machine M' that solves SFP with at most t steps, with at most α legitimate message-signature pairs, with at most β queries to the random oracle and with probability at least ϵ, then there exists some probabilistic Turing machine M that solves DLP with at most $\alpha t_{sig} + \lceil 6\beta/\epsilon \rceil (t + t_{ver}) + O(l^3)$ steps and with probability at least $\epsilon/(4\beta^2)$.

Zakłócanie KAP c.d.

Następnie zdefiniujmy "problem interferencji (IP)":

wejście: p,q,g,v, gdzie v jest losowo wybrane spośród $\{g^0,...,g^{q-1}\}$

wyjście: I,I',u',u,e,w, gdzie

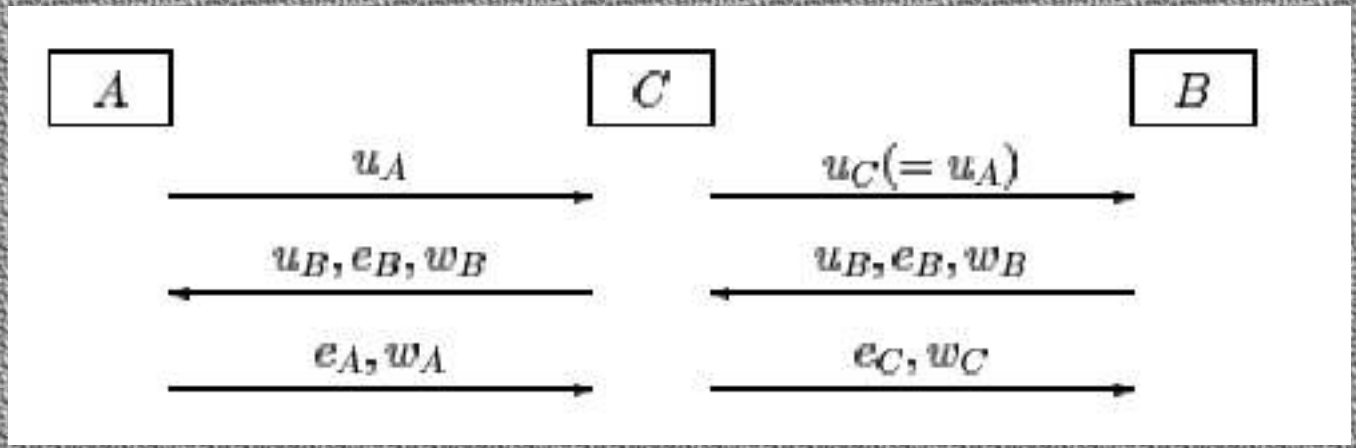
- $u' \in \{g^0,...,g^{q-1}\}$
- I,I' to ciągi $\{0,1\}$-kowe
- (u,e,w) jest prawidłową sygnaturą dla (u',I,I')

IP zakłada, że atakujący może sam wybrać wiadomość. Zauważmy że zakłócanie jest co najmniej tak trudne jak problem IP – w IP I,I',u' mogą być dowolnie wybrane, natomiast przy zakłócaniu I' oraz u' są dane.s

Lemma 2 If there exists some probabilistic Turing machine M' that solves IP with at most t steps, with at most α legitimate message-signature pairs, with at most β queries to the random oracle and with probability at least ϵ, then there exists some probabilistic Turing machine M that solves SFP with at most t steps, with at most α legitimate message-signature pairs, with at most β queries to the random oracle and with probability at least ϵ.

Współdzielenie nieznanego klucza

Przykład ataku “unknown key share” - C próbuje przekonać B, że klucz jest uzgodniony z C, mimo że naprawdę jest uzgodniony z A.



I ponownie bezpieczeństwo protokołu KAP na atak “unknown key share” opiera się na trudności logarytmu dyskretnego, z analogicznym lematem i dowodem jak dla zakłócania.

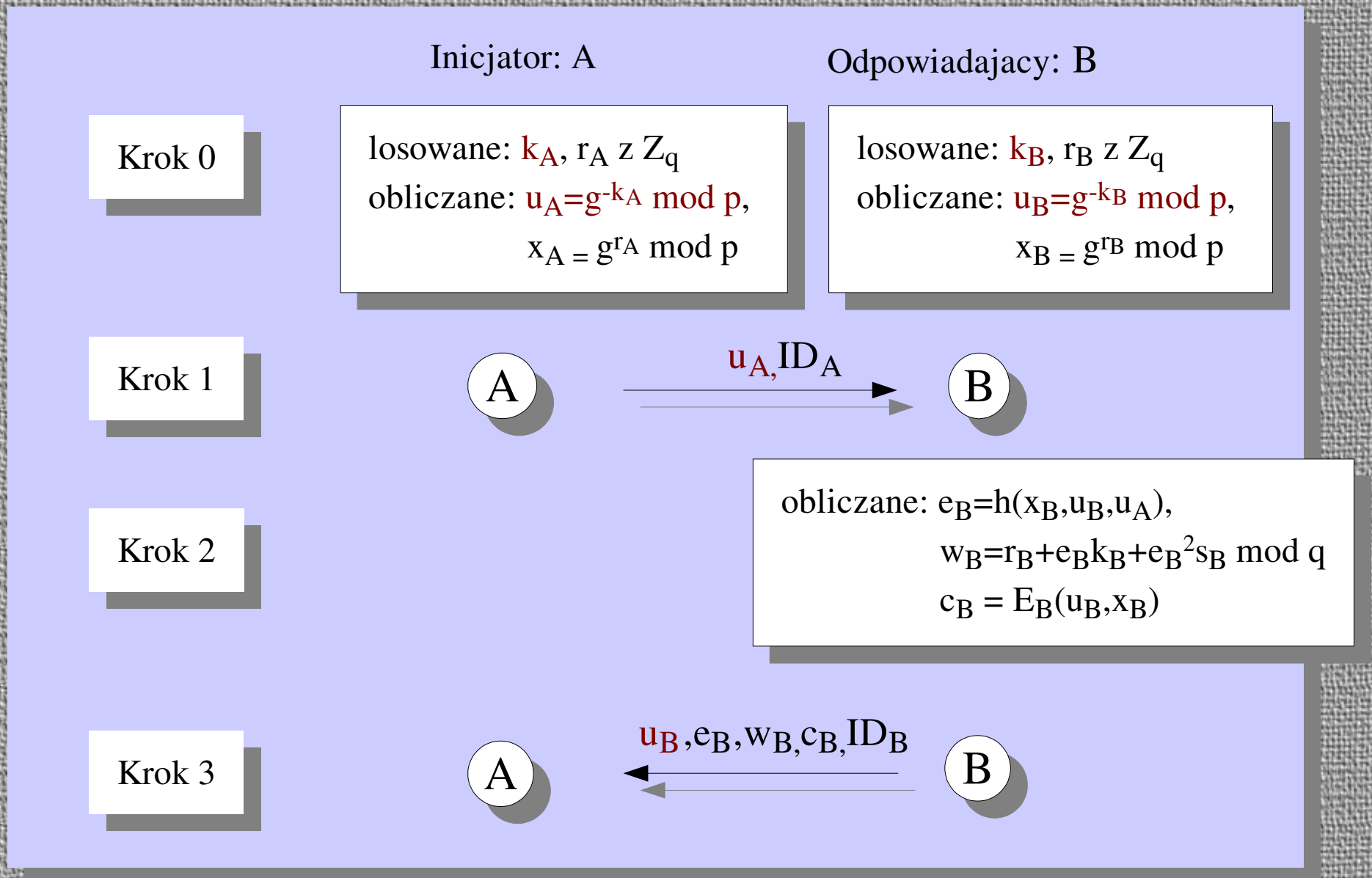
Podsumowanie KAP

- odporny na ataki pasywne i aktywne
- intuicyjny sposób uzgadniania kluczy (jak w protokole Diffiego-Hellmana)
- stosowanie schematu podpisu przesyłanych danych (rozszerzenie schamatu podpisu Schnorra)
- przy każdym wykonaniu każdy uczestnik musi wygenerować dwie losowe liczby z Z_q
- w normalnych warunkach (bez ataku) wymaga nieco więcej potęgowań niż inne protokoły z potwierdzaniem klucza

Protokół Hirosa-Matsury

- oparty o schemat Diffiego-Hellmana uzgadniania klucza
- odporny na atak DoS (zarówno obliczeniowy jak i pamięciowy)
- korzysta z idei bezstanowych połączeń
- dokonuje walidacji klucza bez jego wyliczania
- korzysta z szyfrowania symetrycznego do “przechowywania” prywatnych informacji
- wykorzystuje funkcje jednokierunkowe do unikania ataku obliczeniowego
- bezstanowość połączeń zwiększa ilość przesyłanych każdorazowo danych

Protokół Hiros'a-Matsur'y – schemat



Protokół Hiros'a-Matsur'y - schemat

Krok 4

obliczenie: $z_B=g^{w_B}u_B^{e_B}v_B^{e_B{}^2} \bmod p$

sprawdzenie: $e_B?=h(z_B,u_B,u_A)$,

obliczenie: $a_A=h(z_B,u_A,u_B)$

$e_A=h(x_A,u_B,u_A,z_B)$

$w_A=r_A+e_Ak_A+e_A{}^2s_A \bmod q$

Krok 5

A

$u_A,e_A,w_A,a_A,u_B,e_B,c_B,ID_A$

B

obliczenie: $(u_B,x_B)=D_B(c_B)$

sprawdzenie: $e_B?=h(z_B,u_B,u_A)$,

$a_A?=h(z_B,u_A,u_B)$

obliczenie: $z_A=g^{w_A}u_A^{e_A}v_A^{e_A{}^2} \bmod p$

sprawdzenie: $e_A?=h(z_A,u_B,u_A,x_B)$

obliczenie: $K_B=u_A^{-k_B} \bmod p$

Krok 6

$K_A=u_B^{-k_A} \bmod p$

Ulepszenie protokołu Hirosa-Matsury - KAP-1

- obniża koszt walidacji klucza oraz zmniejsza liczbę operacji wykładniczych
- zakłada, że obie strony korzystają z tej samej jednokierunkowej funkcji oraz synchronicznych stempli czasowych
- nie korzysta z dodatkowego kryptosystemu
- korzysta z bezstanowych połączeń
- korzysta z wartości sekretów zależnych od aktualnego czasu ($h(s_B,t)$)
  - uzgadnianie klucza nie może się przeciągać

Ulepszenie protokołu Hirosa-Matsury - KAP-1

Zmiany w stosunku do protokołu Hirosa-Matsury:

(1) W kroku nr 2 zamiast

$w_B = r_B + e_B k_B + e_B^2 s_B \bmod q$

$c_B = E_B(u_B, x_B)$

obliczane są

$w_B = r_B + e_B k_B + s_B h(T) \bmod q$

$c_B = x_B \oplus h(s_B, t)$

(2) W kroku nr 3 w wiadomości od B do A przesyłany jest dodatkowo stempel czasowy T:

$(u_B, e_B, w_B, c_B, ID_B, T)$

(3) W kroku nr 4:

- nie jest wyliczana wartość a_A

Ulepszenie protokołu Hiros'a-Matsur'y – KAP-1 c.d.

- zamiast

$$z_B = g^{w_B} u_B^{e_B} v_B^{e_B^2} \bmod p$$

$$w_A = r_A + e_A k_A + e_A^2 s_A \bmod q$$

obliczane są

$$z_B = g^{w_B} u_B^{e_B} v_B^{h(T)} \bmod p$$

$$w_A = r_A + k_A + s_A \bmod q$$

(4) W kroku nr 5 – nie jest przesyłane a_A

$$u_A, e_A, w_A, u_B, e_B, c_B, ID_A$$

(5) W kroku nr 6

- obliczane jest $x_B = c_B \oplus h(s_B, t)$
- zamiast $e_B ? = h(z_B, u_B, u_A)$ sprawdzane jest $e_B ? = h(x_B, u_B, u_A)$;
- zamiast $z_A = g^{w_A} u_A^{e_A} v_A^{e_A^2} \bmod p$ jest

$$z_A = g^{w_A} u_A v_A \bmod p$$

Ulepszenie protokołu Hiros'a-Matsur'y - KAP-2

- Protokół KAP-1, korzystając z połączeń bezstanowych, przekazuje stosunkowo dużo informacji, co przy serwerach z niską przepustowością łączy może również ograniczyć zdolność serwera do współpracy z klientami
- Protokół KAP-2 stara się sprawdzić tożsamość klienta przed akceptacją żądania – sprawdzenie tożsamości jest pierwszą operacja wykonywaną przez serwer.

Ulepszenie protokołu Hiros'a-Matsur'y - KAP-1

Zmiany w stosunku do protokołu KAP-2:

(1) W kroku nr 0 obliczana jest wartość $f_A = g^{h(T)+s_A} v_B \bmod p$, która w kroku nr 1 przekazywana jest wraz ze stemplem czasowym w wiadomości stronie B:

$$(u_A, ID_A, T, f_A)$$

(2) W kroku nr 2 pierwszą akcją jest sprawdzenie czy

$$g^{h(T)} = f_A v_A g^{s_B} \bmod p$$

i zamiast

$$w_B = r_B + e_B k_B + s_B h(T) \bmod q$$

obliczane są

$$w_B = r_B + e_B k_B + s_B h(T') \bmod q$$

Bezpieczeństwo protokołów KAP-1 i KAP-2

Bezpieczeństwo wobec ataków pasywnych (aktywnych też) wynika w prosty sposób z trudności problemu logarytmu dyskretnego.

Ponadto korzystanie ze stempli czasowych i wartości aktualnego czasu zapobiega atakom "replay attacks".