Protocoale de gestiune a cheilor

Prof. Dr. Adrian Atanasiu

Universitatea București

November 18, 2011

Sistemele bazate pe chei publice nu necesită un canal sigur pentru transmiterea unei chei private.

Acest avantaj este redus de faptul că un canal cu cheie publică este mult mai lent decât unul cu cheie simetrică, el necesitând protocoale de transmitere, scheme de autentificare etc.

Vom aborda problema de stabilire a unor modalități de a schimba mesaje între mai mulți utilizatori, folosind protocoale rapide și fiecare utilizator având propria sa cheie privată.

Protocoalele de gestiune a cheilor acoperă diverse variante. Există

1. Protocoalele de **distribuire a cheilor**:
   o autoritate generează o cheie K pe care o transmite mai departe altor entități (utilizatori).
2. Protocoalele de **punere de acord** (key agreement):
   participanții stabilesc o cheie K folosind scheme de partajare a secretelor.

Protocoalele de punere de acord pot fi:

- **implicite**:
  Alice este asigurată că nimeni înafară de *Bob* nu poate determina cheia K.
  Aceasta nu înseamnă că *Alice* este sigură că *Bob* are efectiv cheia K.
  Un astfel de protocol se numește *punere de acord autentificată* (AK).
- **explicite**:
  Alice este sigură că *Bob* a calculat efectiv cheia K.

  De obicei, aceste protocoale se obțin prin completarea punerilor de acord implicite cu protocoale de confirmare a cheii (de către *Bob*).
  Sunt numite AKC (protocoale de *punere de acord explicită cu confirmare*).

Baza de comunicare este o rețea (nesigură) care leagă n utilizatori.

Se presupune că există o autoritate T (*arbitru*) considerată onestă; ea verifică identitățile utilizatorilor, alegerea și transmiterea cheilor inițiale, precum și alte probleme curente.

Deoarece rețeaua nu este sigură, participanții trebuie să prevadă unele atacuri din partea lui *Oscar*.

Acesta poate fi pasiv (doar interceptează mesaje) sau activ.

Un adversar activ este capabil:

- Să modifice un mesaj în cursul transmisiei;
- Să transmită mesaje vechi;
- Să încerce să se prezinte drept unul din utilizatorii rețelei.

Obiectivul lui *Oscar* este:

- Să îi facă pe *Alice* și *Bob* să accepte o cheie *invalidă*;
- Să îi facă pe *Alice* și *Bob* să creadă că au schimbat chei între ei.

Proprietăți de securitate

- **Securitatea cheii**: Fiecare execuție a protocolului de generare a cheii trebuie să genereze o cheie secretă unică: *cheia de sesiune*.
- **Secretul cheilor ulterioare**: Compromiterea cheii secrete a unui utilizator să nu afecteze secretul cheilor de sesiune anterioare.
  Adesea se face distincție între situația când numai cheia secretă a lui *Alice* este compromisă, și cea când cheile lui *Alice* și *Bob* sunt compromise simultan.

Continuare

- **Impersonalizarea cheii compromise**: Dacă *Oscar* a descoperit cheia secretă a lui *Alice*, el îi poate lua locul în protocoalele de gestiune a cheii (o "impersonalizează" pe *Alice*).
  Acest lucru nu trebuie însă să-l impersonalizeze pe *Bob* față de *Alice*.
- **Anonimitatea componentelor cheii**: *Bob* nu poate fi obligat să împartă o cheie cu o entitate pe care nu o cunoaște.

Protocoale de distribuire a cheilor

Pentru orice pereche de utilizatori (A, B), arbitrul T generează aleator o cheie $K_{A,B} = K_{B,A}$ pe care o transmite lui A și B printr-un canal securizat (neutilizat pentru alte comunicații).

Această cheie va fi folosită ulterior de cei doi utilizatori pentru a interschimba mesaje.

Deci, în această fază, T generează C_n^2 chei pe care le distribuie celor n utilizatori.

Fiecare utilizator trebuie să păstreze cheia sa, precum și cele $n - 1$ chei de comunicare cu ceilalți utilizatori.

Probleme:

- Existența de canale securizate între T și fiecare din cei n participanți.
  Pentru o rețea cu 1000 utilizatori trebuie $C_{1000}^2 = 499.500$ canale sigure de comunicație.
  În plus, la venirea unui utilizator nou trebuie securizate alte 1000 canale.
- Obligația pentru fiecare participant să stocheze $n-1$ chei și să participe la C_n^2 transmisii de chei solicitate de server.
  Chiar pentru o rețea mică, acest lucru devine destul de dificil.

Protocolul Blom

Fie n ($n \geq 3$) utilizatori și p ($p \geq n$) un număr prim.
O cheie este un element din $\mathbb{Z}_p^*$.
Se alege un număr întreg $k \in \mathbb{Z}_{n-1}^*$ (k este gradul de protecție).

În procedeul Blom – definit inițial în 1984 pe baza codurilor separabile cu distanță maximă (MDS) și simplificat în 1993 – T transmite fiecărui utilizator, prin canale securizate, $k+1$ elemente din $\mathbb{Z}_p$ (în loc de $n-1$ chei în varianta generală).
Fiecare pereche de utilizatori (A, B) poate calcula o cheie de sesiune $K_{A,B} = K_{B,A}$.

Condiția de securitate: orice coaliție de k utilizatori – diferiți de A și B – nu poate obține informații despre $K_{A,B}$.

Protocolul Blom pentru cazul $k = 1$

1. T face publice: un număr prim p și – pentru fiecare utilizator A – un număr $r_A \in \mathbb{Z}_p$.
   Toate numerele r_A sunt distincte.
2. T generează aleator trei numere $a, b, c \in \mathbb{Z}_p$ și formează polinomul

$$f(x, y) = a + b(x + y) + cxy \pmod{p}$$

3. Pentru fiecare utilizator A, T determină polinomul

$$g_A(x) = f(x, r_A) \pmod{p}$$

   și transmite $g_A(x)$ lui A prin canal securizat.
4. Dacă A și B doresc să comunice, cheia lor secretă este

$$K_{A,B} = K_{B,A} = f(r_A, r_B)$$

- Aplicația $g_A(x)$ este o transformare afină, de forma

$$g_A(x) = a_A + b_A x$$

unde

$$a_A = a + br_A \ (mod\ p), \quad b_A = b + cr_A \ (mod\ p).$$

- La pasul 4, A poate calcula cheia secretă

$$K_{A,B} = f(r_A, r_B) = g_A(r_B)$$

iar B – în mod similar

$$K_{B,A} = f(r_A, r_B) = g_B(r_A)$$

Exemplu

Fie $p = 17$ și 3 utilizatori A, B, C – având cheile publice $r_A = 12,\ r_B = 7,\ r_C = 1$.
Presupunem că arbitrul alege $a = 8,\ b = 7,\ c = 2$. Atunci

$$f(x, y) = 8 + 7(x + y) + 2xy$$

Polinoamele g sunt

$$g_A(x) = 7 + 14x, \quad g_B(x) = 6 + 4x, \quad g_C(x) = 15 + 9x.$$

Cele trei chei private sunt

$$K_{A,B} = 3, \quad K_{A,C} = 4, \quad K_{B,C} = 10.$$

A poate calcula $K_{A,B}$ prin $g_A(r_B) = 7 + 14 \cdot 7 = 3 \pmod{17}$.
B poate calcula $K_{B,A}$ prin $g_B(r_A) = 6 + 4 \cdot 12 = 3 \pmod{17}$.

Teoremă

Protocolul Blom pentru $k = 1$ este necondiționat sigur contra oricărui atac individual.

Un protocol este considerat *necondiționat sigur* dacă securitatea sa nu depinde de ordinul de complexitate al calculelor necesare pentru spargerea sa.

În schimb, orice coaliție între doi participanți C, D poate conduce la aflarea cheii $K_{A,B}$.
Aceștia dispun de informațiile:

$$\begin{cases} a_C = a + br_C, & b_C = b + cr_C, \\ a_D = a + br_D, & b_D = b + cr_D \end{cases}$$

Avem un sistem de 4 ecuații cu 3 necunoscute, din care se poate afla soluția unică (a, b, c).
După ce s-au aflat aceste trei valori, se poate construi polinomul $f(x, y)$, din care se poate determina mai departe restul informației.

Protocolul Blom poate fi generalizat pentru a rezista la atacul unei alianțe de k utilizatori.

Singura modificare se face la pasul 2, unde arbitrul folosește polinomul

$$f(x,y) = \sum_{i=0}^{k}\sum_{j=0}^{k} a_{i,j}x^i y^j \pmod{p}$$

$a_{i,j} \in \mathbb{Z}_p$ $(0 \leq i \leq k,\ 0 \leq j \leq k)$ și $a_{i,j} = a_{j,i}$ pentru orice i și j.

Protocolul Needham - Schroeder

Un protocol de distribuție a cheilor, celebru prin implicațiile sale în stabilirea de chei de comunicație este definit în 1978 de Roger Needham și Michael Schroder.

Protocolul Needham - Schroeder (*NS*) a fost propus în două variante:

- Protocolul *NS* cu chei simetrice.
  A fost destinat stabilirii unei chei de sesiune între doi utilizatori conectați pe o rețea și a stat ulterior la baza construirii protocolului Kerberos.
- Protocolul *NS* cu chei publice.
  Oferă o autentificare reciprocă a celor doi utilizatori care comunică prin rețea

Protocolul *NS* cu chei simetrice

A inițiază un protocol de comunicație cu B, iar T este un server – considerat de ambii utilizatori ca fiind "de încredere".
Se mai folosesc:

- $K_{A,T}$ – o cheie simetrică, cunoscută de A și T;
- $K_{B,T}$ – o cheie simetrică, cunoscută de B și T;
- N_A, N_B – numere (nonces) generate aleator de A respectiv B.
- $K_{A,B}$ – cheia (simetrică) de sesiune între A și B.

Notație: $\{\alpha\}_K$ – criptarea mesajului α cu cheia K.
Varianta de bază a protocolului *NS* este:

1. A trimite lui T tripletul (A, B, N_A).
2. T trimite lui A mesajul

$$\{N_A, K_{A,B}, B, \{K_{A,B}, A\}_{K_{B,T}}\}_{K_{A,T}}$$

3. A decriptează mesajul și transmite mai departe lui B:

$$\{K_{A,B}, A\}_{K_{B,T}}$$

4. B decriptează și trimite spre A mesajul $\{N_B\}_{K_{A,B}}$.
5. A răspunde lui B cu $\{N_B - 1\}_{K_{A,B}}$.

1. A anunță serverul că dorește stabilirea unui canal de comunicație cu B.
2. Serverul generează cheia $K_{A,B}$ pe care o retrimite lui A în două copii: una pentru A, iar alta pe care A o va forwarda lui B. Deoarece A poate solicita chei pentru comunicarea cu mai mulți utilizatori, nonce-ul asigură faptul că serverul a răspuns la acest mesaj; de asemenea, includerea lui B îi spune lui A cu cine va partaja această cheie de sesiune.
3. A forwardează cheia lui B; acesta o poate decripta folosind cheia de comunicare cu serverul.
4. B arată că deține cheia de sesiune $K_{A,B}$.
5. A arată că este conectat și deține cheia de sesiune.

Protocolul este vulnerabil la un atac prin reluare.

Dacă *Oscar* deține o valoare compromisă (mai veche) pentru $K_{A,B}$, el poate returna lui B mesajul $\{K_{A,B}, A\}_{K_{B,T}}$.
Acesta îl va accepta, neputând să distingă dacă ce a primit este o cheie nouă sau nu.

Această slăbiciune va fi corectată de protocolul Kerberos prin introducerea unei ștampile de timp sau de nonce-uri.

Protocolul *NS* cu chei publice

A și B folosesc un server T care distribuie – la cerere – chei publice. Aceste chei sunt:

- (K_{PA}, K_{SA}) – componenta publică și secretă a cheii lui A.
- (K_{PB}, K_{SB}) – componenta publică și secretă a cheii lui B.
- (K_{PT}, K_{ST}) – componenta publică și secretă a cheii serverului T.

  K_{ST} este folosită pentru criptare, iar K_{PT} – pentru decriptare (deci T folosește cheia sa pentru autentificare).

1. A trimite lui T perechea (A, B) prin care solicită cheia publică a lui B.
2. T trimite lui A elementul $\{K_{PB}, B\}_{K_{ST}}$.
3. A generează nonce-ul N_A și-l trimite lui B mesajul $\{N_A, A\}_{K_{PB}}$
4. B trimite lui T cuplul (B, A) solicitând cheia publică a lui A.
5. T răspunde cu $\{K_{PA}, A\}_{K_{ST}}$.
6. B generează nonce-ul N_B și-l trimite lui A $\{N_A, N_B\}_{K_{PA}}$. Trimiterea lui N_A arată lui A că B deține de cheia de decriptare K_{SB}.
7. A confirmă că deține această cheie, trimițând lui B mesajul $\{N_B\}_{K_{PB}}$.

Securitatea protocolului

Nu rezistă atacului *man-in-the-middle*.
Dacă O îl convinge pe A să inițieze o sesiune cu el, poate retrimite mesajele lui B și să-l convingă că este în contact cu A.
Ignorând schimbul de informații cu serverul T, atacul este:

1. A trimite lui O mesajul $\{N_A, A\}_{K_{PO}}$.
2. O decriptează cu K_{SO} și trimite lui B mesajul $\{N_A, A\}_{K_{PB}}$.
3. B răspunde lui O (crezând că este A) cu mesajul $\{N_A, N_B\}_{K_{PA}}$.
4. O trimite acest mesaj mai departe, spre A.
5. A răspunde lui O cu $\{N_B\}_{K_{PO}}$.
6. O extrage de aici N_B și-l trimite lui B, criptat cu cheia publică a acestuia: $\{N_B\}_{K_{PB}}$.

În final, B este convins că comunică cu A și că N_A, N_B sunt cunoscute doar de el și de A.

Atacul a fost prezentat prima oară în 1995 de Gavin Lowe.

Acesta construiește o variantă numită protocolul *Needham - Schroeder - Lowe*.

Modificarea se referă la pasul 6 din protocol, unde mesajul $\{N_A, N_B\}_{K_{PA}}$ – trimis de B către A – este înlocuit cu

$$\{N_A, N_B, B\}_{K_{PA}}$$

Protocolul Kerberos

Este unul din cele mai răspândite sisteme de gestiune a cheilor – în special versiunile 4 și 5 (ultimul, adoptat ca standard Internet *RFC* 1510).
Protocolul se bazează parțial pe protocolul Needham - Schroeder, aducând unele îmbunătățiri legate de securitate.

Fiecare utilizator A împarte cu T o cheie *DES* notată K_A.
De asemenea, lui A i se asociază (în general sub controlul lui T) o secvență de informații care-l identifică în mod unic.

Această secvență se notează cu $ID(A)$.

La solicitarea lui A de a comunica cu B, arbitrul T efectuează următoarele operații:

- Generează o cheie K;
- Înregistrează ora H a cererii;
- Stabilește o durată L de validitate a lui K; deci cheia de sesiune este validă în intervalul de timp $[H, H + L]$.

Tichetul cu informațiile (K, H, L) sunt transmise de T lui A, apoi lui B.

1. T generează K, H și L.
2. T calculează mesajele

$$m_1 = \{K, ID(B), H, L\}_{K_A}, \quad m_2 = \{K, ID(A), H, L\}_{K_B}$$

pe care le trimite lui A.
3. A decriptează m_1 și află $K, H, L, ID(B)$. Pe urmă calculează

$$m_3 = \{ID(A), H\}_K$$

și trimite lui B mesajele m_2 și m_3.
4. B află $K, H, L, ID(A)$ din decriptarea lui m_2. Cu ajutorul lui K decriptează m_3 și află $H, ID(A)$.
Verifică dacă cele două valori pentru H și $ID(A)$ sunt identice.
5. B calculează $m_4 = \{H + 1\}_K$ pe care îl trimite lui A.
6. A decriptează m_4 și verifică dacă mesajul obținut este $H + 1$.

Fiecare din cele patru mesaje m_i transmise are rolul său bine determinat.

- m_1 și m_2 servesc la transmiterea confidențială a cheii K.
- m_3 și m_4 asigură o confirmare a cheii; după primirea ei, A și B sunt siguri că dispun de aceeași cheie de sesiune K.
- Rolul lui H și L este acela de protejare contra unui atac activ constând din înregistrarea unor mesaje vechi și – ulterior – retransmiterea lor.

Opțiuni de implementare a tichetului (K, H, L)

- *Tichet eliberat pe termen lung*.
  Există riscul de a permite atacuri asupra cheii.
- *Tichet postdatat*.
  Riscul este ca *Oscar* să afle tichetul înainte de utilizarea sa; de aceea multe servere au permisiunea de a le putea respinge.
- *Tichet proxy*.
  Un client dă permisiune serverului de a iniția anumite comunicații în numele său.

Una din slăbiciunile sistemului *Kerberos* constă în imposibilitatea unei bune sincronizări a ceasurilor utilizatorilor.

În practică se admit anumite decalaje, stabilite de comun acord.

În plus, spargerea sistemului *DES* a condus – în unele standarde – la înlocuirea sistemului *Kerberos*.

Schimbul de chei Diffie - Hellman

Dacă nu se acceptă un furnizor universal de chei, atunci va trebui stabilit un protocol de punere de acord.

Primul și cel mai cunoscut astfel de protocol este *protocolul de schimb de chei Diffie - Hellman*.

Definit în 1976 și folosit în diverse variante, el se bazează pe problema logaritmului discret.

Variante ale protocolului

Pentru inițializare:

- p – număr prim (de 1024 biți – în protocoalele standard).
- q – divizor prim al lui $p - 1$ (de 160 biți).
- $\alpha \in \mathbb{Z}_p^*$, element de ordin q.
- h – funcție de dispersie criptografică (de exemplu $SHA1$).
- (sig_T, ver_T) – algoritm de semnătură a arbitrului T.

Utilizatorul A dispune de cheia secretă $a \in \mathbb{Z}_q^*$ și cheia publică $Y_A = \alpha^a$.
Pe baza componentelor publice stocate în $ID(A)$ și la solicitarea lui A, arbitrul T eliberează un certificat

$$Cert(A) = (ID(A), Y_A, sig_T(ID(A), Y_A))$$

care este de asemenea public.
Orice utilizator B poate verifica autenticitatea certificatului $Cert(A)$ și poate obține de aici – printre altele – cheia publică Y_A.

Aceleași elemente

$$(b,\ Y_B = \alpha^b,\ ID(B),\ Cert(B))$$

sunt generate pentru utilizatorul B.

1. Arbitrul nu trebuie să cunoască cheia secretă "a" pentru a produce certificatul.
   Când A intră în rețea, se generează un astfel de certificat, care poate fi păstrat în baza de date sau poate fi comunicat chiar de A la fiecare utilizare.
2. Înainte de a elibera $Cert(A)$, T se convinge că A posedă cheia secretă "a" corespunzătoare cheii publice Y_A (cu un protocol "provocare - răspuns").
   Astfel se evită un atac prin care *Oscar* înregistrează Y_A drept cheia sa publică, îngreunând posibilitatea ca un alt utilizator B să intre în contact cu A.
3. De asemenea, T face o "validare" a cheii publice Y_A, verificând relațiile

$$1 < Y_A < p, \quad Y_A^q \equiv 1 \ (mod\ p)$$

Protocolul Diffie - Hellman de perioadă scurtă

1. A generează aleator $x \in \mathbb{Z}_q^*$ și trimite lui B valoarea
$$R_A = \alpha^x \pmod p$$
2. B generează aleator $y \in \mathbb{Z}_q^*$ și trimite lui A valoarea
$$R_B = \alpha^y \pmod p$$
3. A calculează
$$K_{A,B} = R_B^x = \alpha^{xy}$$
4. B calculează
$$K_{B,A} = R_A^y = \alpha^{xy}$$

Schimbul de chei din sistemul *SSH* (Secure SHell) – care asigură comunicarea pentru sistemele bazate pe UNIX are implementat protocolul Diffie - Hellman de perioadă scurtă.

Protocolul asigură o autentificare implicită a cheii de sesiune.

Avantaj: fiecare execuție a protocolului generează o cheie de sesiune nouă.

Dezavantaj: nu rezistă la un atac *man-in-the-middle*, neexistând nici o autentificare a utilizatorilor implicați.

Protocolul Diffie - Hellman de perioadă lungă

1. A trimite lui B certificatul $Cert(A)$.
2. B trimite lui A certificatul $Cert(B)$.
3. A extrage Y_B din $Cert(B)$ și calculează
$$K_{A,B} = Y_B^a = \alpha^{ab}$$
4. B extrage Y_A din $Cert(A)$ și calculează
$$K_{B,A} = Y_A^b = \alpha^{ab}$$

Avantaj: autentificarea reciprocă a partenerilor.
Dezavantaj: la fiecare execuție se obține aceeași cheie de sesiune; cheile se schimbă doar odată cu schimbarea certificatelor – după o perioadă de timp relativ lungă.

Exemplu

Să presupunem

$$p = 25307, \quad \alpha = 2$$

Dacă luăm $a = 3578$, vom avea $Y_A = 2^{3578} = 6113 \pmod{25307}$, valoare pusă în certificatul lui A.
Să presupunem că B alege $b = 19956$; atunci
$Y_B = 2^{19956} = 7984 \pmod{25307}$.
A poate calcula cheia comună

$$K_{A,B} = 7984^{3578} = 3694 \pmod{25307}$$

Aceiași cheie este calculată și de B:

$$K_{U,V} = 6113^{19956} = 3694 \pmod{25307}$$

Protocolul *MTI/C0*

Combină cele două variante Diffie - Hellman în una singură:

1. A generează aleator $x \in \mathbb{Z}_q^*$ și trimite lui B valoarea

$$T_A = Y_B^x \pmod p$$

2. B generează aleator $y \in \mathbb{Z}_q^*$ și trimite lui A valoarea

$$T_B = Y_A^y \pmod p$$

3. A calculează $K_{A,B} = (T_B)^{a^{-1}x} = \alpha^{xy}$
4. B calculează $K_{B,A} = (T_A)^{a^{-1}y} = \alpha^{xy}$

Nu asigură o autentificare implicită a cheii.

Astfel, *Oscar* poate (printr-un atac *man-in-the-middle*) să înlocuiască T_A și T_B cu numărul 1.

Atunci A și B vor obține cheia 1, cheie cunoscută și de *Oscar*.

Securitatea protocolului Diffie - Hellman

Semnătura lui T pe certificate împiedică producerea de informații publice false.

Rămân în discuție atacurile pasive, care se reduc la problema:

"*C poate determina* $K_{A,B}$ *dacă nu este A sau B ?*"

Sau:

"Fiind date α^a și α^b, ambele $(mod\ p)$, se poate calcula α^{ab} $(mod\ p)$?"

Formal (*problema Diffie - Hellman*):

Fie $I = (p, \alpha, \beta, \gamma)$ unde p este număr prim, $\alpha \in \mathbb{Z}_p$ este primitiv, iar $\beta, \gamma \in \mathbb{Z}_p^*$.
Se poate determina $\beta^{log_\alpha \gamma}$ $(mod\ p)$?

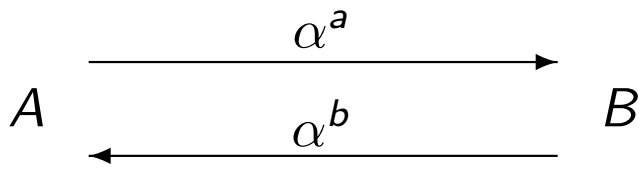
Securitatea protocolului Diffie - Hellman față de atacurile pasive este echivalentă cu dificultatea problemei Diffie - Hellman.

Dacă C poate calcula a (sau b) plecând de la Y_A (respectiv Y_B), el poate deduce $K_{A,B}$ așa cum face A (respectiv B).
Aceasta conduce la rezolvarea unei probleme de logaritm discret.

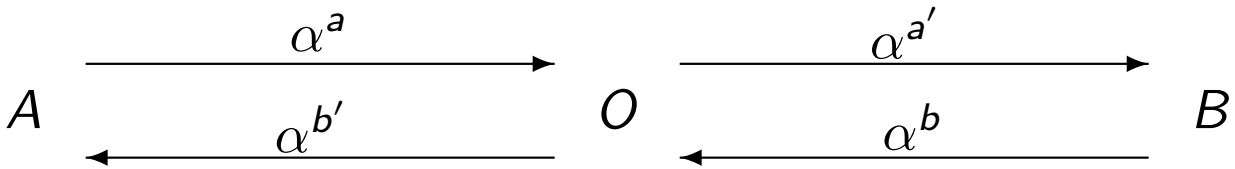
Deci, dacă problema logaritmului discret în $\mathbb{Z}_p$ este dificilă, atunci protocolul de punere de acord a cheii Diffie - Hellman nu poate fi atacat.

Conjectură: problema Diffie - Hellman este echivalentă cu problema logaritmului discret.

Protocolul de bază Diffie - Hellman nu rezistă unui atac *man-in-the-middle*.
Protocolul Diffie - Hellman se poate reprezenta schematic:



Într-un atac activ, O se interpune între A și B:



unde $a', b' \in \mathbb{Z}_q^*$ sunt generate aleator de O.

O interceptează mesajele lui A și B și le înlocuiește cu ale sale.

La sfârșitul protocolului, el a stabilit o cheie de comunicație $\alpha^{ab'}$ cu A și o cheie $\alpha^{a'b}$ cu B.

Când A dorește să trimită un mesaj lui B, el va utiliza cheia pe care o împarte cu O; acesta poate decripta mesajul și apoi să îl cripteze cu cheia comună cu B.

B primește mesajul, fără să realizeze că a fost citit de O.

Punerea de acord *MTI*

Îmbunătățirea lui $MTI/C0$ pentru a rezista unui atac *man-in-the-middle*.

1. A generează aleator $x \in \mathbb{Z}^*_{p-1}$, calculează $R_A = \alpha^x \pmod p$ și trimite lui B perechea $(Cert(A), R_A)$.
2. B generează aleator $y \in \mathbb{Z}^*_{p-1}$, calculează $R_B = \alpha^y \pmod p$ și trimite lui A perechea $(Cert(B), R_B)$.
3. A calculează $K_{A,B} = R_B^a \cdot Y_B^x \pmod p$ iar B calculează $K_{B,A} = R_A^b \cdot Y_A^y \pmod p$.

Cheia comună de sesiune este deci

$$K = K_{A,B} = K_{B,A} = \alpha^{ay+bx} \pmod p$$

Exemplu

Fie $p = 27803$ și $\alpha = 5$.
A alege $a = 21131$ și va calcula $Y_A = 21420\ (mod\ 27803)$, pe care și-l pune în certificat.
La fel, B alege $b = 17555$; deci $Y_B = 17100\ (mod\ 27803)$.
Presupunem că A ia $x = 169$; el va trimite lui B

$$R_A = 6268\ (mod\ 27803)$$

Dacă B alege $y = 23456$, el trimite lui A

$$R_B = 26759\ (mod\ 27803)$$

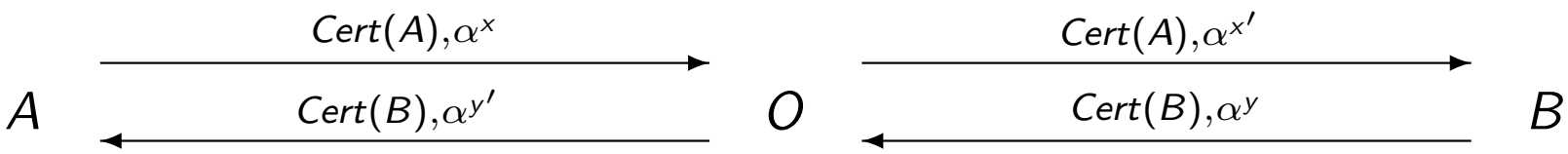
Acum se poate calcula

$$K_{A,B} = R_B^a \cdot Y_B^x\ (mod\ p) = 21600\ (mod\ 27803).$$

Aceeași cheie o obține și B.

Semnătura arbitrului elimină posibilitatea interpunerii lui *Oscar*.

Într-adevăr, dacă un intrus O se interpune între A și B, va avea loc scenariul



A și B vor avea chei diferite: A calculează $K = \alpha^{xb+y'a} \ (mod\ p)$, iar B calculează $K = \alpha^{x'b+ya} \ (mod\ p)$.
În plus, nici una din cheile lui A sau B nu poate fi calculată de O, pentru că aceasta ar necesita cunoașterea lui "a" respectiv "b".
Deci, deși A și B obțin chei distincte, niciuna din ele nu poate fi calculată de O.
Este o *autentificare implicită* a cheii de sesiune.

Protocoale *AK* înrudite cu protocolul Diffie - Helman

Sunt protocoale de punere de acord a cheilor cu autentificare (*AK*) utilizate frecvent în standarde criptografice.

Confirmarea cheii este făcută prin protocoale separate, incluse în standarde.

1. A și B obțin – din certificate – cheile publice Y_B respectiv Y_A.
2. A generează aleator $x \in \mathbb{Z}_q^*$ și trimite lui B valoarea $R_A = \alpha^x$.
3. B generează aleator $y \in \mathbb{Z}_q^*$ și trimite lui A valoarea $R_B = \alpha^y$.
4. A verifică: $1 < R_B < p, \qquad (R_B)^q \equiv 1 \pmod p$.
   Dacă relațiile sunt îndeplinite, calculează cheia
   $$K_{A,B} = (Y_B)^x + (R_B)^a \pmod p$$
5. B procedează similar, obținând – în caz de succes – cheia
   $$K_{B,A} = (Y_A)^y + (R_A)^b \pmod p$$
6. A și B calculează cheia de sesiune (de 80 biți)
   $$K = e(K_{A,B}) = e(K_{B,A}) = \alpha^{ay} + \alpha^{bx}$$
   unde e este funcția de criptare a sistemului simetric SKIPJACK.

Exemplu

Fie $p = 43$ și $q = 7$. Se alege $\alpha = 4$ (ordin 7 în $\mathbb{Z}_{43}^*$).
Să presupunem că $a = 5$ și $b = 2$; deci cheile publice sunt $Y_A = 4^5 = 35\ (mod\ 43)$ respectiv $Y_B = 4^2 = 16\ (mod\ 43)$.
Protocolul *KEA* va decurge astfel:

1. A și B obțin (din certificate) $Y_B = 16$ respectiv $Y_A = 35$.
2. A alege $x = 6$ și trimite lui B valoarea $R_A = 4^6 = 11$.
3. B alege $y = 3$ și trimite lui A valoarea $R_B = 4^3 = 21$.
4. A verifică condițiile $1 < R_B < 21 < 43$ și $R_B^q = 21^7 = 1$.

Cum ele sunt îndeplinite, A calculează cheia

$$K_{A,B} = 16^6 + 21^5 = 39\ (mod\ 43).$$

5. B verifică condițiile $1 < R_A < 11 < 43$ și $R_A^q = 11^7 = 1$.

Cum ele sunt îndeplinite, B calculează cheia

$$K_{B,A} = 35^3 + 11^2 = 39\ (mod\ 43).$$

A nu verifică $(R_B)^q \equiv 1 \ (mod \ p)$

Să presupunem că $p-1$ are un divizor s de lungime mică și fie $\beta \in \mathbb{Z}_p^*$ de ordin s.
Dacă B trimite $R_B = \beta$, atunci A calculează

$$K_{A,B} = \alpha^{bx} + \beta^a \ (mod \ p) \text{ și } K = e(K_{A,B}).$$

Să presupunem acum că A trimite lui B un mesaj criptat $c = e_K(m)$, unde m are o structură standard.
Pentru fiecare $d \in [0, s-1]$, B va calcula

$$K'_{B,A} = \alpha^{bx} + \beta^d \ (mod \ p) \text{ și } K' = e(K'_{B,A}),$$

după care va decripta $m' = e_{K'}^{-1}(c)$.

Dacă m' are structura standard, atunci B deduce că $d = a \ (mod \ s)$, deci obține informație parțială despre a.

A nu verifică $1 < Y_B < p, \quad 1 < R_B < p.$

Atunci *Oscar* certifică $Y_O = 1$ drept cheia sa publică, după care trimite lui B cheia publică temporară R_A a lui A, spunând că este a sa.

După ce B replică cu R_B, *Oscar* trimite $R'_B = 1$ lui A, spunând că vine de la B.
A calculează $K_{A,B} = \alpha^{bx} + 1$ și B calculează $K_{B,O} = \alpha^{bx} + 1$.

Deci B are – fără să știe – o cheie de sesiune comună cu A.

Argumente pentru construirea cheii de sesiune prin criptarea cu un sistem de criptare simetric a cheii comune:

1. Se amestecă biții "tari" cu eventuali biți "slabi" – care oferă informație despre cheia comună $K_{A,B}$, sau pot fi ghiciți relativ ușor.
2. Se distrug relațiile algebrice dintre cheia comună și cheile publice Y_A, Y_B, R_A, R_B.

   Astfel se previne *atacul triunghiular al lui Burmester*

Atacul Burmester

C este un utilizator cu perechea de chei (c, α^c).
El observă o execuție a protocolului între A și B, în care aceștia inter-schimbă cheile publice temporare $R_A = \alpha^x$, $R_B = \alpha^y$;
cheia comună rezultată este $K_{A,B} = K_{B,A} = \alpha^{ay} + \alpha^{bx}$

C inițiază cu A o execuție a protocolului, folosind drept cheie publică temporară $R_C = \alpha^y$.
Dacă A folosește acum $R_A = \alpha^{x'}$, va rezulta o cheie

$$K_{A,C} = \alpha^{ay} + \alpha^{cx'}$$

pe care o poate calcula numai A.
Similar, C inițiază cu B o execuție a protocolului, folosind $R_C = \alpha^x$;
rezultă o cheie

$$K_{B,C} = \alpha^{bx} + \alpha^{cy'}$$

care poate fi calculată numai de B.

Dacă C poate afla prin alte mijloace $K_{A,C}$ și $K_{B,C}$ (faza de "cheie cunoscută" a atacului), el va putea determina

$$K_{A,B} = K_{A,C} + K_{B,C} - \alpha^{cx'} - \alpha^{cy'}.$$

Utilizat în prima versiune ANSI $X9.42$, ANSI $X9.63$ și IEEE $P1363$.

1. A generează aleator $x \in \mathbb{Z}_q^*$; trimite lui B elementele $R_A = \alpha^x$ și $Cert(A)$.
2. B generează aleator $y \in \mathbb{Z}_q^*$; trimite lui A elementele $R_B = \alpha^y$ și $Cert(B)$.
3. A verifică relațiile
$$1 < R_B < p, \qquad (R_B)^q \equiv 1 \pmod{p}$$
Dacă ele sunt îndeplinite, A calculează cheia de sesiune
$$K = h((Y_B)^a \| (R_B)^x)$$
4. B procedează similar.
Cheia de sesiune este obținută de B cu formula
$$K = h((Y_A)^b \| (R_A)^y) = h(\alpha^{ab} \| \alpha^{xy})$$

Inclus în prima versiune ANSI *X*9.42, ANSI *X*9.63 și IEEE *P*1363. Diferă de *Modelul unificat* prin calculul cheii de sesiune.

1. A generează $x \in \mathbb{Z}_q^*$; trimite lui B $R_A = \alpha^x$ și $Cert(A)$.
2. B generează $y \in \mathbb{Z}_q^*$; trimite lui A $R_B = \alpha^y$ și $Cert(B)$.
3. A verifică relațiile

$$1 < R_B < p, \qquad (R_B)^q \equiv 1 \ (mod\ p)$$

Dacă ele sunt îndeplinite, A calculează

$$s_A = (x + a \cdot \overline{R}_A)\ (mod\ q), \quad K_A = \left(R_B \cdot (Y_B)^{\overline{R}_B}\right)^{s_A}$$

Dacă relațiile nu sunt verificate sau $K_A = 1$, atunci A oprește protocolul.

4 B procedează similar și calculează

$$s_B = (y + b \cdot \overline{R}_B)\ (mod\ q), \quad K_B = \left(R_A \cdot (Y_A)^{\overline{R}_A}\right)^{s_B}$$

5 Cheia comună de sesiune este

$$K = h(K_A) = h(K_B) = h(\alpha^{s_A s_B})$$

Raportul care definește protocolul folosește o notație suplimentară: Dacă $X \in \mathbb{Z}_p^*$, atunci

$$\overline{X} = (X \ (mod \ 2^{80})) + 2^{80}$$

Mai general,

$$\overline{X} = (X \ (mod \ 2^{\lceil f/2 \rceil})) + 2^{\lceil f/2 \rceil}$$

unde f este lungimea lui q, exprimată în biți. De remarcat că $\overline{X} \ (mod \ q) \neq 0$.
Utilizarea lui $\overline{R}_A$ folosește doar jumătate din biții lui R_A; în acest fel numărul de operații pentru calculul lui $(Y_A)^{\overline{R}_A}$ se înjumătățește fără a afecta securitatea protocolului.

În plus, definiția lui $\overline{R}_A$ asigură $\overline{R}_A \neq 0$, deci contribuția cheii private "a" la calculul lui s_A nu poate fi ignorată.

Asupra protocolului *MQV* poate fi lansat un atac *on-line*:
Să presupunem că C interceptează cheia R_A trimisă spre B; pe baza ei:

1. C calculează
   $R_C = R_A \cdot (Y_A)^{\overline{R}_A} \cdot \alpha^{-1}$,
   $c = \left(\overline{R}_C\right)^{-1}$, $Y_C = \alpha^c$.
2. C determină cheia Y_C și o certifică (posibil, deoarece C știe cheia secretă c).
3. C transmite lui B elementele R_C și $Cert(C)$.
4. B răspunde cu R_B, pe care C îl forwardează lui A.

În acest moment, A și B au o cheie de sesiune comună K, pe care B crede că o împarte cu C (nu cu A).

Variante ale protocoalelor *AK* într-o singură trecere

Principalul scop: punerea de acord a lui A și B asupra unei chei de sesiune necesare transmiterii unui singur mesaj – de la A la B; acest lucru presupune evident faptul că A are acces la $Cert(B)$.

Astfel de protocoale sunt utile în aplicații în care numai unul din utilizatori este on-line.

KEA, *Modelul unificat* și *MQV* pot fi convertite în protocoale cu o singură trecere, prin simpla înlocuire a cheii temporare R_B a lui B cu cheia publică Y_B.

Transformarea *MQV* într-un protocol cu o singură trecere

1. A generează $x \in \mathbb{Z}_q^*$; trimite lui B $R_A = \alpha^x$ și $Cert(A)$.
2. A calculează

$$s_A = (x + a \cdot \overline{R}_A) \, (mod \ q), \quad K_A = \left(Y_B \cdot (Y_B)^{\overline{Y}_B} \right)^{s_A}$$

   Dacă $K_A = 1$, atunci A oprește execuția (cu eșec).
3. B verifică $1 < R_A < p$ și $(R_A)^q \equiv 1 \pmod{p}$
   Dacă aceste condiții sunt îndeplinite, atunci B calculează

$$s_B = (b + b \cdot \overline{Y}_B) \, (mod \ q), \quad K_B = \left(R_A \cdot (Y_A)^{\overline{R}_A} \right)^{s_B}$$

   Dacă $K_B = 1$, atunci B oprește protocolul (cu eșec).
4. Cheia de sesiune este

$$K = h(K_A) = h(K_B) = h(\alpha^{s_A s_B})$$

Protocoale *AKC* derivate din protocoale *AK*

Un protocol în trei treceri derivat din *Modelul unificat*, la care se adaugă un *MAC* care autentifică numărul trecerii, utilizatorii și cheile temporare.

Se mai folosesc două funcții de dispersie h_1, h_2.

În practică, ele sunt definite

$$h_1(m) = h(10, m), \quad h_2(m) = h(01, m)$$

unde h este o funcție de dispersie criptografică.

1. A generează aleator $x \in \mathbb{Z}_q^*$; trimite lui B $R_A = \alpha^x$ și $Cert(A)$.
2.
   1. B verifică $1 < R_A < p$ și $(R_A)^q \equiv 1 \pmod p$
      Dacă nu sunt îndeplinite, B oprește execuția (cu eșec).
   2. B generează $y \in \mathbb{Z}_q^*$ și calculează:
      $k' = h_1\left((Y_A)^b \| (R_A)^y\right), \qquad K = h_2\left((Y_A)^b \| (R_A)^y\right),$
      $R_B = \alpha^y, \qquad m_B = MAC_{k'}(2, B, A, R_B, R_A).$
   3. B trimite lui A tripletul $(Cert(B),\ R_B,\ m_B)$.

3

1. A verifică $1 < R_B < p$ și $(R_B)^q \equiv 1 \,(mod\ p)$ Dacă nu sunt îndeplinite, B oprește execuția (cu eșec).
2. A calculează
   $k' = h_1\left((Y_B)^a \,\|\, (R_B)^x\right), \quad m'_B = MAC_{k'}\,(2, B, A, R_B, R_A)$
   și verifică egalitatea $m'_B = m_B$.
3. A calculează
   $m_A = MAC_{k'}\,(3, A, B, R_A, R_B), \quad K = h_2\left((Y_B)^a \,\|\, (R_B)^x\right)$
   și trimite lui B valoarea m_A.

4 B calculează $m'_A = MAC_{k'}\,(3, A, B, R_A, R_B)$ și verifică $m'_A = m_A$.

5 Cheia de sesiune este K.

În mod similar se pot construi protocoale *AKC* derivate din *KEA* și *MQV*.

Variantele *AKC* ale protocoalelor *MQV* și *Model Unificat* sunt incluse în standardul ANSI *X*9.63.

Protocol între stații (*STS*)

1. A generează aleator un număr $a \in \mathbb{Z}_{p-1}^*$; apoi calculează numărul $\alpha^a \ (mod\ p)$ pe care îl trimite lui B.
2. B generează aleator un număr $b \in \mathbb{Z}_{p-1}^*$.
3. B calculează $\alpha^b \ (mod\ p)$, apoi

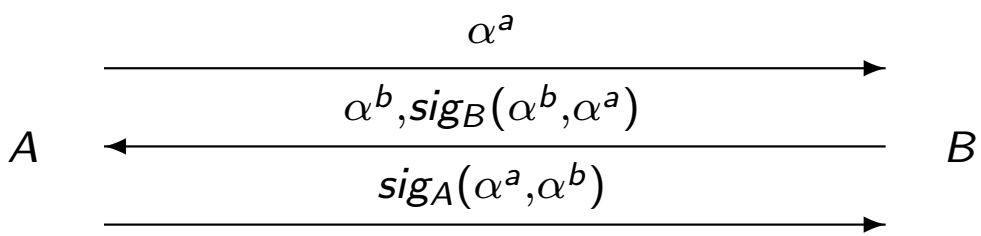
$$K = (\alpha^a)^b \ (mod\ p), \quad y_B = sig_B(\alpha^b, \alpha^a)$$

   Trimite lui A mesajul $(Cert(B), \alpha^b \ (mod\ p), y_B)$.
4. A calculează $K = \left(\alpha^b\right)^a \ (mod\ p)$ și verifică $Cert(V)$ cu ver_T, apoi y_B cu ver_B extras din $Cert(B)$.
5. A calculează $y_A = sig_A(\alpha^a, \alpha^b)$ și trimite lui B mesajul $(Cert(A), y_A)$.
6. B verifică $Cert(A)$ cu ver_T, apoi y_A cu ver_A extras din $Cert(A)$.

Numărul prim p și elementul primitiv $\alpha \in \mathbb{Z}_p$ sunt publice.
Fiecare utilizator A dispune de un protocol privat de semnătură sig_A și unul public de verificare ver_A.
Arbitrul T are de asemenea asociate funcțiile sig_T respectiv ver_T.
Certificatul lui A este

$$Cert(A) = (ID(A), ver_A, sig_T(ID(A), ver_A))$$

Informațiile schimbate în cadrul protocolului STS simplificat sunt schematizate:



Un intrus O nu se poate interpune între A și B.

Într-adevăr, dacă O interceptează α^a și-l înlocuiește cu $\alpha^{a'}$, el va trebui să înlocuiască de asemenea și $sig_B(\alpha^b, \alpha^{a'})$ cu $sig_B(\alpha^{b'}, \alpha^a)$, ceea ce nu poate decât în cazul $a = a'$ și $b = b'$ (pentru că nu cunoaște sig_B).

La fel, O nu poate înlocui $sig_A(\alpha^{a'}, \alpha^b)$ cu $sig_A(\alpha^a, \alpha^{a'})$, pentru că nu cunoaște sig_A.

Protocolul nu oferă o confirmare a cheii.

Pentru aceasta trebuie modificat

$$y_B = e_K(sig_B(\alpha^b, \alpha^a))$$

în pasul 3 și

$$y_A = e_K(sig_A(\alpha^a, \alpha^b))$$

în pasul 5.
În acest fel – ca la *Kerberos* – se obține o confirmare a cheii decriptând o parte cunoscută a cheii de sesiune.

Acesta este protocolul *STS* complet.

Chei auto-certificate

Valoarea cheii publice asigură o autentificare implicită.

Fie $n = pq$ unde $p = 2p_1 + 1,\ q = 2q_1 + 1$ sunt numere prime tari (p_1, q_1 sunt numere prime distincte) și $\alpha \in \mathbb{Z}_n^*$ de ordin $2p_1q_1$.

Factorizarea $n = pq$ este cunoscută numai de către arbitrul T.
Valorile n, α sunt publice, iar p, q (deci și p_1, q_1) sunt secrete.

T alege e – exponent de criptare *RSA* public.
Exponentul de decriptare $d = e^{-1}\ (mod\ \phi(n))$ este secret.

Fiecare utilizator A are un identificator $ID(A)$ și primește de la T o cheie publică auto-certificată R_A, conform cu:

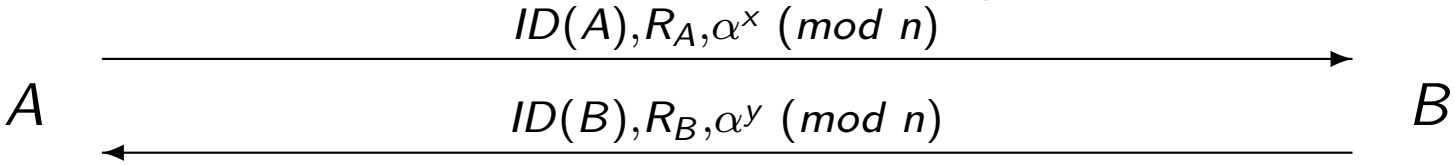
1. A alege un exponent secret a și calculează $Y_A = \alpha^a \ (mod\ n)$.
2. A trimite lui T valorile (a, Y_A).
3. T calculează $R_A = (Y_A - ID(A))^d \ (mod\ n)$, pe care îl trimite lui A.

De remarcat aportul arbitrului în calculul lui R_A.
Se observă că

$$Y_A = R_A^e + ID(A) \quad (mod\ n)$$

poate fi calculat folosind numai informațiile publice.

Schematizat, schimbul de informații din protocolul Girault este:



La sfârșitul acestui protocol, A și B dispun de aceeași cheie:

$$K = \alpha^{bx+ay} \pmod n$$

1. A generează x și calculează $S_A = \alpha^x \pmod n$; tripletul $(ID(A), R_A, S_A)$ este trimis lui B.
2. B alege y și calculează $S_B = \alpha^y \pmod n$; tripletul $(ID(B), R_B, S_B)$ este trimis lui A.
3. A calculează cheia

$$K_{A,B} = S_B^a \cdot (R_B^e + ID(B))^x \pmod n.$$

Cheia calculată de B este

$$K_{B,A} = S_A^b \cdot (R_A^e + ID(A))^y \pmod n$$

Exemplu

Fie $p = 839$ și $q = 863$. Atunci $n = 724057$ și $\phi(n) = 722356$.
Elementul $\alpha = 5$ are ordinul $2p_1q_1 = \phi(n)/2$.

Dacă T alege exponentul $e = 84453$, atunci $d = 125777$.
Dacă $ID(A) = 500021$ și $a = 111899$, avem $Y_A = 488889$ și $R_A = 650704$.
Similar, fie $ID(B) = 500022$ și $b = 123456$, deci $Y_B = 111692,\ R_B = 683556$.

Dacă A și B vor să stabilească o cheie de sesiune și A alege numărul $x = 56381$, iar B numărul $y = 356935$, vom avea

$$S_A = 171007,\ S_B = 320688.$$

După protocol, cei doi utilizatori vor dispune de cheia comună $K = 42869$.

Securitate

Cum valorile $Y_A, R_A, ID(A)$ nu sunt semnate de autoritatea T, nimeni nu poate verifica direct autenticitatea lor.

Să presupunem că ele provin de la O (fără ajutorul arbitrului), care vrea să se dea drept A.

Dacă O furnizează $ID(A)$ și dacă R_A conduce la un Y'_A greșit, nu se poate calcula exponentul a' asociat (dacă problema logaritmului discret este dificilă).
Fără a', O nu poate determina cheia.

O situație similară apare dacă O se interpune între A și B.
El poate împiedica pe A și B să obțină o cheie comună, dar nu poate efectua calculele lor.

De ce A trebuie să comunice arbitrului valoarea a

T poate determina R_A plecând de la Y_A, fără să cunoască a. Acest lucru se face pentru ca arbitrul să fie convins (înainte de a calcula R_A) că A posedă cheia secretă a.

Dacă T nu face această verificare înainte de calculul lui R_A, sistemul poate fi atacat.

Să presupunem că O alege un a' fals și determină $Y'_A = \alpha^{a'} \pmod n$.
Cu aceste elemente, el stabilește o cheie publică falsă

$$R'_A = (Y'_A - ID(A))^d \pmod n$$

în felul următor: O calculează

$$Y'_O = Y'_A - ID(A) + ID(O)$$

și trimite lui T perechea $(Y'_O, ID(O))$.

Presupunem că arbitrul calculează efectiv pentru O valoarea

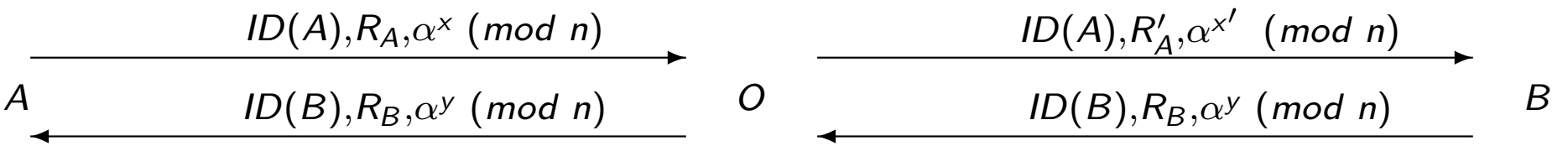
$$R'_O = (Y'_O - ID(O))^d \pmod n$$

Atunci, din

$$Y'_O - ID(O) \equiv Y'_A - ID(A) \pmod n$$

se obține imediat $R'_O = R'_A$.

Să presupunem acum că A și B efectuează protocolul, iar O se interpune conform schemei:



B calculează cheia $K' = \alpha^{x'b+ya'} \pmod n$, iar A calculează $K = \alpha^{xb+ya} \pmod n$.

O obține K' calculând $K' = S_B^{a'} \cdot (R_B^e + ID(B)))^{x'} \pmod n$.
O și B posedă aceeași cheie, iar B crede că o împarte cu A.

În acest moment, O poate decripta mesajele trimise de B pentru A.

Sfârșit