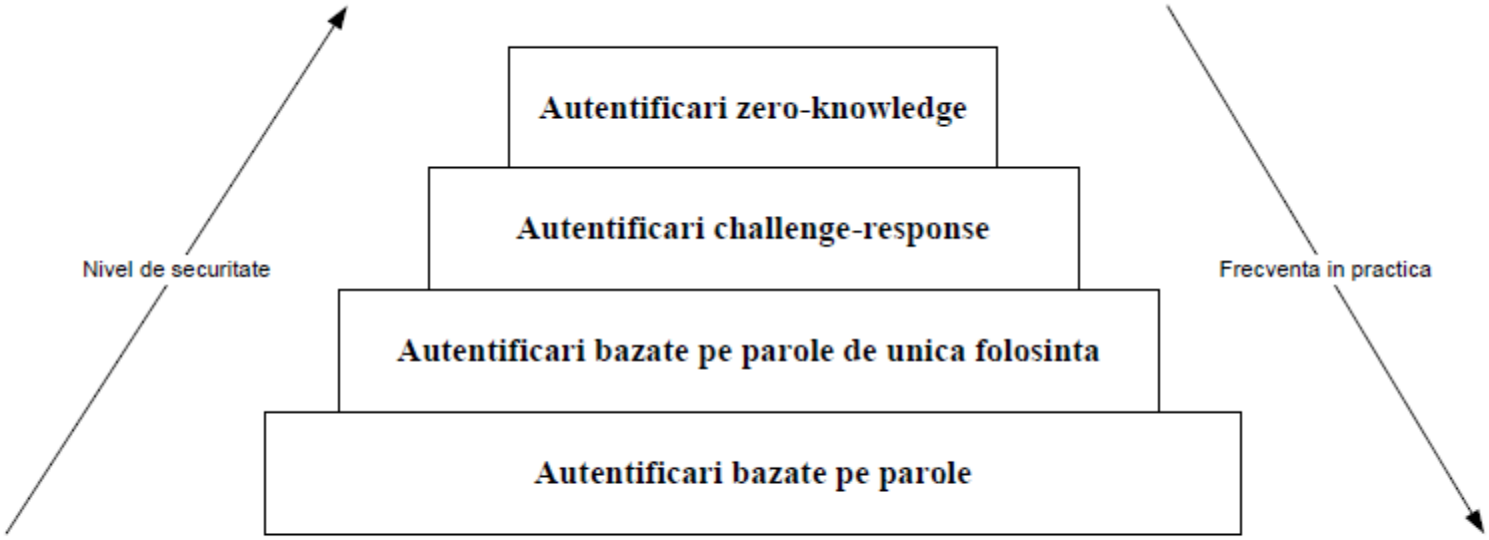
Cap. 4.1. Protocoale de autentificare. Autentificarea informatiei, autentificarea entitatilor si schimburi autentificate de cheie secreta. Principii constructive: password based authentication, one-time passwords, challenge-response, zero-knowledge.

Protocoale de autentificare



Obiectivul Autentificarii

- **Importanta**:

1) **cel mai frecvent intalnit/impus** obiectiv de securitate – folosim protocoale de autentificare aproape zilnic: citirea unui email, plata cu o carte de credit, simple convorbiri telefonice
2) **adauga valoare informatiei** – informatia nu are valoare atata timp cat nu exista o garantie asupra sursei sale de provenienta

Obiective distincte in autentificare

- Obiectivul autentificarii poate fi descompus in trei obiective distincte:

1) **Autentificarea Informatiei**
2) **Autentificarea Entitatilor (Identificare)**
3) **Schimb autentic de cheie secreta** (doi participanti intra in posesia unei chei secrete cu garanti asupra sursei de la care provine cheia)

Tipuri de atacurilor asupra protocoalelor de autentificare

- **Retransmisie** a mesajului (**replay**) – comun in autentificarile bazate pe parole,
- **Pre-play**, varianta atacului replay – de exemplu asupra autentificarii S/Key
- **Man-in-the-middle** – adversarul (exemplu clasic protocolul de schimb de cheie Diffie-Hellman)
- **Sesiuni paralele** – doua sau mai multe sesiuni sunt rulate in paralel, adversarul extragand din una din ele raspunsuri pentru alta
- Atacuri prin **reflexie** – vezi slide protocoale challenger-response cu chei simetrice
- Atacuri de **suprapunere** (**interleaving**) – tot prin dezvoltarea de sesiuni paralele
- Atacuri datorate **tipurilor de date gresite** – confuzii intre identitati de participant si nonce etc.
- Atacuri datorate **omisiei numelor participantilor** – prin omisia numelor participantilor, mesaje din sesiunea cu un participant pot fi folosite in sesiuni cu alt participant
- Atacuri datorare **folosirii gresite a primitivelor criptografice** – se datoreaza folosirii unei primitive criptografice cu prezumtia ca asigura si alt obiective decat cel care il asigura (de exemplu multi cred ca functiile de criptare asigura si integritate)

Autentificarea Informatiei

- **Autentificarea Informatiei** (Autentificarea sursei informatiei) se refera la o **garantie asupra sursei de provenienta a informatiei**.
- Este in stransa legatura cu obiectivul integritatii informatiei pe care il implica. Exista diferente semnificative:
  - autentificarea obliga la o garantie asupra sursei de unde provine informatia iar integritatea nu
  - autentificarea implica o comunicare iar integritatea nu (date care nu se afla in tranzit pot fi garantate ca integritate dar nu ca autenticitate pentru ca puteau fi stocate de catre oricine),
  - autenticitatea implica o prospetimea informatiei (orice informatie care poate fi sursa a unui replay are integritate dar nu autenticitate).
- In urma unui schimb autentic de informatie trebuie sa poata fi stabilite: identitatea celui care a trimis informatia, integritatea informatiei, actualitatea informatiei respective.

Functii criptografice utilizate in asigurarea autenticitatii informatiei

- **Coduri de autentificare a mesajelor** MAC (Message Authentication Code) - $MAC_k(m)$ - cod de autentificare a mesajului m calculat cu cheia k :
- **Garanteaza autenticitatea** unui mesaj
- Se **construiesc peste o functie hash** (in general MD5 sau SHA1, cu toate ca ambele au un nivel de securitate destul de scazut) in conjunctie cu o cheie secreta
- In practica se foloseste **HMAC** si **NMAC** propuse de Mihir Bellare, Ran Canetti, Hugo Krawczyk in lucrarea http://www-cse.ucsd.edu/users/mihir/papers/kmd5.pdf
- **Semnaturi Digitale** – $Sig_A(M)$ – semnatura digitala a participantului A asupra mesajului M:
- Pot fi vazute ca **functionalitate aditionala a criptosistemelor cu cheie publica**
- **Garanteaza non-repudierea**, oferind astfel si o garantie asupra sursei mesajului (autenticitate)
- Se construiesc pe sisteme criptografice asimetrice prin **“inversarea” rolului cheii publice si private**, astfel se foloseste cheia privata pentru a semna mesaje iar cheia publica pentru a verifica semnatura
- **Pot fi construite si pe functii criptografice simetrice** (semnaturi digitale one-time)
- Observatie: aceste functii criptografice garanteaza autenticitatea informatiei fara a aduce garantii de timp (asupra actualitatii informatiei), in momentul in care aceste primitive sunt incluse in protocoale de autentificare trebuie asigurata si actualitatea informatiei prin parametrii varianti in timp (contor, nonce, timestamp)

Coduri MAC vs. Semnaturi Digitale

- Codurile MAC nu pot fi vazute ca si substituent pentru Semnaturi Digitale si nici invers deoarece nu raspund aceluiasi obiectiv de securitate
- Totusi, deoarece ambele pot fi utilizate pentru garantarea autenticitatii, o comparatie este utila

- Coduri MAC:
- (+) Usor de calculat, implica doar operatii simple (doar putin mai costistisitoare ca o functie hash)
- (+) Dimensiune redusa (la nivelul unei functii hash)
- (-) Utilizeaza chei secrete
- (-) Pentru comunicarea multiaterala intre n entitati numarul de chei devine n(n-1)/2

- Semnaturi Digitale:
- (-) Greu de calculat, implica operatii aritmetice complexe (in grupuri de intregi)
- (-) Dimensiune ridicata (100, 1000 de biti)
- (+) Utilizeaza chei publice
- (+) Este suficienta o singura cheie publica pentru o entitate iar cu aceasta un mesaj semnat poate fi verificat de oricare alta entitate

O apreciere cantitativa asupra cerintelor computationale

Cryptographic function		CPU		
		Intel Centrino 1.7 GHz	Intel Dual Core 1.6 GHz	Intel Core Duo 6600 2.4 GHz
Modular exponentiation, basic operation for a digital signature (module and exponent size in right column)	512	7.8×10^{-3} s	6.1×10^{-3} s	3.1×10^{-3} s
	1024	48.4×10^{-3} s	44.6×10^{-3} s	20.3×10^{-3} s
	2048	359.4×10^{-3} s	323.8×10^{-3} s	153.2×10^{-3} s
Mac with SHA1	160	0.00859×10^{-3} s	0.00812×10^{-3} s	0.00406×10^{-3} s
Mac with MD5	128	0.00579×10^{-3} s	0.00354×10^{-3} s	0.00219×10^{-3} s
Sha-1	160	0.00281×10^{-3} s	0.00212×10^{-3} s	0.00109×10^{-3} s
Sha-256	256	0.0086×10^{-3} s	0.00592×10^{-3} s	0.00282×10^{-3} s
Sha -384	384	0.01359×10^{-3} s	0.01234×10^{-3} s	0.00579×10^{-3} s
Sha-512	512	0.02625×10^{-3} s	0.02324×10^{-3} s	0.01141×10^{-3} s
MD5	128	0.00156×10^{-3} s	$9.5E\text{-}4 \times 10^{-3}$ s	$4.6E\text{-}4 \times 10^{-3}$ s

Autentificarea Entitatilor (Identificare)

- **Autentificarea Entitatilor** (Identificare, Verificarea Identitatii) se refera la o **garantie asupra identitatii unui participant** la comunicare (poate fi vazuta ca un caz de autentificare a informatiei in care identitatea unei entitati este informatia ce trebuie autentificata)

Baza autentificarii entitatilor

- Autentificarea unei entitati se poate baza pe:

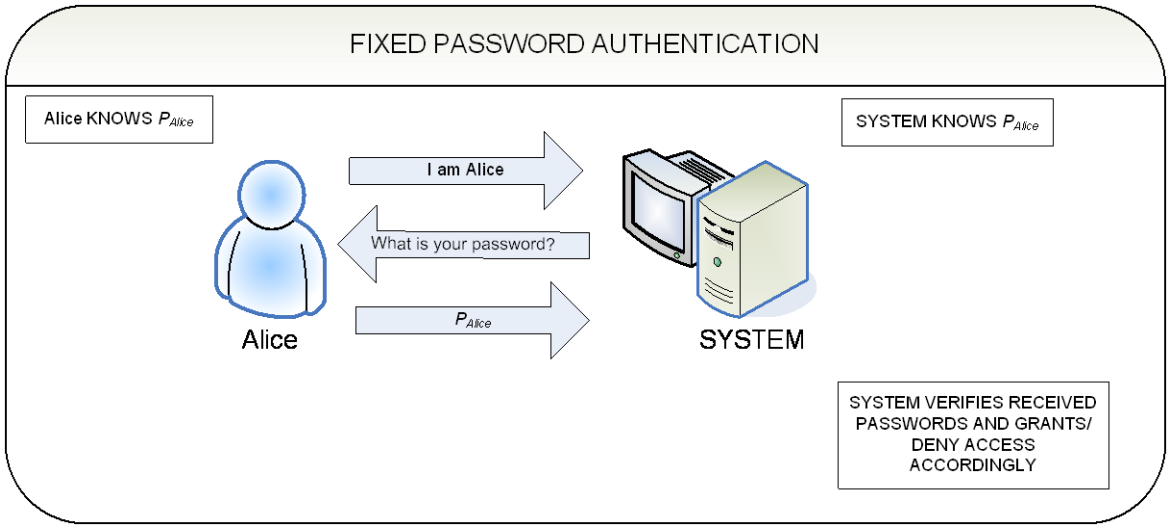
1) **Ceva cunoscut** – in general un secret, de exemplu: o parola, un cod PIN
2) **Ceva imanent (care tine de entitate)** – o trasatura biometrica: amprenta, retina, modul in care un individ tasteaza
3) **Ceva detinut** – un accesoriu fizic, de exemplu o cheie, un card magnetic, un smart-card

Caracteristici ale unui protocol de identificare ([Menezes et. al., p. 387])

- **Reciprocitate** – identificarea poate fi unilaterala, sau bilaterala (mutuala)
- **Eficienta Computationala** – intensitatea computationala a operatiilor necesare protocolului (operatiile cu primitive simetrice si fara cheie sunt ieftine iar cele cu primitive asimetrice de 10, 100 sau 1000 de ori mai scumpe)
- **Eficienta in Comunicare** – numarul de runde de comunicare si dimensiunea mesajelor vehiculate (cata informatie este necesara pentru a demonstra identitatea cuiva)
- **Implicarea real-time a unei parti terte** – uneori este necesara interventia unei parti terte pentru a putea efectua o identificare (de exemplu in cazul scenariilor bazate pe functii simetrice, pentru comunicarea intre n entitati sunt necesare n(n-1)/2 chei distincte, dar daca exista o parte de incredere o singura cheie partajata cu aceasta poate fi suficienta, in acest caz numarul de chei devine n pentru partea de incredere si 1 pentru participanti)
- **Natura implicarii partii terte** – in partea terta trebuie avuta incredere ca pastrator al confidentialitatii (securitate absoluta) sau doar asupra autenticitatii furnizata (securitate functionala)
- **Natura garantiilor de securitate** – pe ce se bazeaza securitatea?
- **Stocarea secretelor** – unde trebuie stocate secretele: pe un dispozitiv fizic, memorate etc.

Autentificarea bazata pe parole (Autentificare slaba)

- **Cel mai frecvent intalnite** si ofera si **cel mai scazut nivel de securitate** (totusi suficient pentru o buna parte din aplicatii)
- **Principiu**: autentificarea se face prin **dezvaluirea unui secret (numit parola)** pentru a demonstra identitatea unei entitati



Modul de stocare al parolelor

- **In fisiere de parole** – presupune stocarea parolei in plaintext intr-un fisier de parole. Fisierul este protejat la citire si scriere. Dezavataj: nu prezinta securitate in fata utilizatorilor privilegiati pentru a citi fisierul
- **In fisiere "criptate"** – presupune stocarea parolei intr-un fisier "criptat", in acest caz fisierul trebuie protejat doar la scriere.

- **Observatie**: termentul de fisier "criptat" pentru memorarea parolelor ("encrypted" password files) este pastrat in limbaj din ratiuni istorice, in realitate nu se foloseste o functie de criptare pentru stocarea parolelor, ci o functie one-way (de exemplu hash) pentru a face ca parola sa nu fie recuperabila din datele aflate in fisier, dar sa fie verificabila pe baza acestora

Parole fixe - Avantaje

- **Usor de implementat** (ca protocol de autentificare)
- Sunt **usor de memorat** pentru oameni

Parole fixe - Dezavantaje

- Pot fi **furate (observate)** si utilizate de un adversar
- Sunt **stocate atat de partea utilizatorului cat si a sistemului**
- Utilizatorii aleg in general parole cu **entropie scazuta**

Atacuri asupra autentificarilor bazate pe parola

- **Retransmisia (replay)** – presupune interceptarea (observarea) parolei si retransmiterea ulterioara a acesteia – **Aparare: parole one-time.**
- **Cautari exhaustive** - deoarece utilizatorii aleg parole de entropie scazuta, un potential adversar poate incerca toate parolele posibile. Cautarile exhaustive pot fi **on-line** (incearca diverse parole la login) sau **off-line** (analiza asupra unui fisier "criptat" de parole) – **Aparare: parole de entropie ridicata, politici de securitate.**
- **Atacuri de tip dictionar precalculat** – deoarece utilizatorii au obiceiul de a utiliza ca parole cuvinte (dintr-un dictionar), un adversar poate calcula off-line parole corespunzatoare tututor cuvintelor din dictionar ca apoi sa verifice daca in fisierul de parole apare o corespondenta. **Aparare: salting**.

Proceduri de intarire a parolelor

- **Politici de securitate** - obligarea la:

1) **Pentru a creste entropia**: o dimensiune minima a parolei (de ex. 10 caractere), utilizarea a minim 3 caractere din fiecare set (alfa-numerice, non-alfa-numerice, upper/lower-case)
2) **Pentru a reduce riscul unui atac**: schimbarea unei parole la intervale fixe de timp (de ex. 1 luna), limitarea numarului de incercari in cazul introducerii unei parole gresite

- **Cresterea intensitatii functiei de verificare a parolelor** – pentru a ridica necesitatile de calcul la o cautare exhaustiva poate fi crescuta intesitatea functiei de “criptare” a parolei (de exemplu iterarea functiei de n ori)
- **Adaugarea unor valori arbitrare (salt)** – pentru a opri atacuri de tip dictionar parola poate fi concatenata cu k biti aleatori, numiti salt, in acest caz dimensiunea dictionarului creste de 2^k ori, valoarea de salt se pastreaza alaturi de parola (Atentie! Aceasta metoda nu creste entropia parolei)
- **Utilizarea frazelor in locul unor simple cuvinte** – un caracter din limba curenta are cam 1-1.5 biti de entropie, utilizarea unei fraze suficient de lungi poate duce la o parola destul de rezistenta.

Autentificare cu parole de unica folosinta (one-time passwords)

- Pentru **a preveni** atacurile de tip retransmisie o solutie este utilizarea parolelor de unica folosinta (**one-time passwords**)
- Exista **doua solutii rudimentare:**

1) **Utilizarea unei liste de parole** – se foloseste o lista de *t* parole, fiecare valabila la o singura utilizare (parolele pot fi utilizate secvential sau nu)

2) **Utilizarea unor parole actualizate secvential** – la fiecare autentficare, utilizatorul schimba parola veche cu cea noua

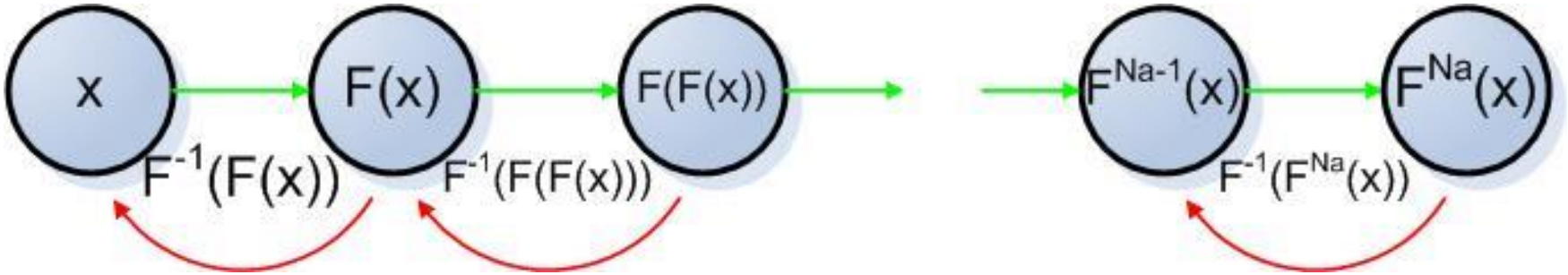
Schema Lamport – schema de generat parole one-time

- Prima solutie avansata de generare a parolelor one-time
- **Avantaje** ale schemei Lamport:
  1) **Nu se stocheaza secrete de partea sistemului**
  2) Fiecare parola este **valabila doar la o singura utilizare**
  3) Parolele au **entropie ridicata**
  4) Poate fi **utilizata pe canale nesigure**

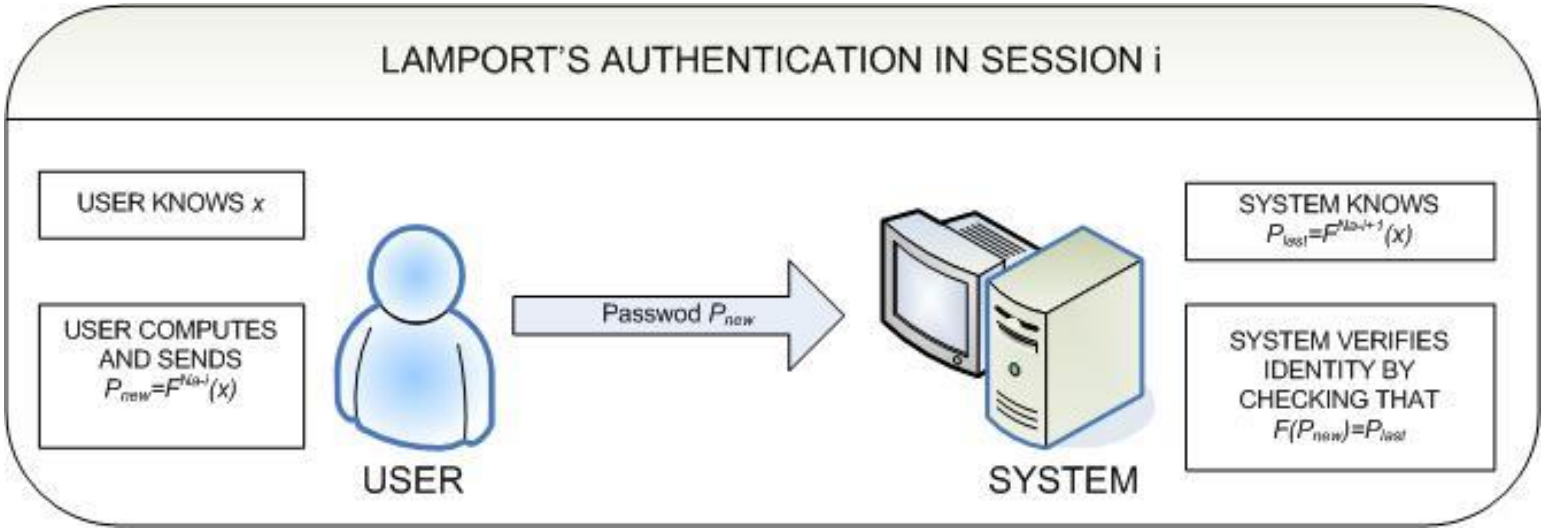
- **Principiu de functionare** al schemei Lamport:

- Fie *x* o valoare arbitrara aleasa de utilizator, *Na* numarul de autentificari necesare (de ex. 10000), *F* o functie one-way (de ex. o functie hash)
- Utilizatorul calculeaza de partea sa secventa $\{x, F(x), F(x), \ldots, F^{Na-1}(x), F^{Na}(x)\}$
- $F^{Na}(x)$ este trimisa sistemului intr-un stadiu off-line pentru a garanta autenticitatea acestei valori
- Cand utilizatorul se autentifica prima oara prezinta $F^{Na-1}(x)$ iar sistemul verifica daca $F(F^{Na-1}(x))=F^{Na}(x)$
- La modul general pentru a *i*-a autentificare parola este $F^{Na-i}(x)$

- Compozitia succesiva a functiei conduce la un lant one-way



- Fiecare element din lant joaca rol de parola



Schema Lamport in practica

- Exista o singura propunere de utilizare – sistemul S-Key [Haller, 1995]
- Poate fi spart printr-un atac de tip pre-play – un adversar poate impersona sistemul si poate captura parole inca nefolosite trimise de utilizator
- **Dezavantajul** schemei Lamport – **nu ofera autentificare mutuala**

Autentificare provocare-raspuns (challenge-response) (Autentificare puternica)

- **Principiu: Demonstreaza cunoasterea secretui** in baza caruia se demonstreaza identitatea **fara a dezvalui secretul**
- Se realizeaza prin calcularea unei valori numita raspuns pe baza altei valori numita challenge

1. A -> B : challenge

2. B -> A : response

Parametrii varianti in timp

- Unul dintre obiectivele autentificarilor challenge-response este **eliminarea toatala a atacurilor de tip retransmisie replay**
- Parametrii varianti in timp sunt necesari pentru a preveni atacuri de tip replay prin garantarea unicitatii unei instante a unui protocol respectiv a cronologiei in unele cazuri (timeliness)
- Parametrii varianti in timp se numesc **Nonce** (a value used no more than once)
- Exista trei tipuri de parametrii varianti in timp

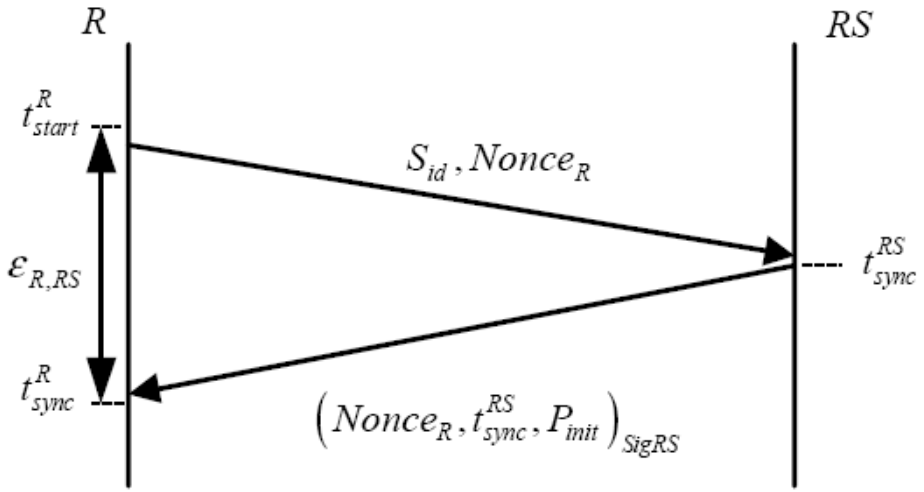
1) **Numere aleatoare** (cel mai comun utilizate) – numere generate aleator in fiecare instanta a protocolului (in general previn si atacuri de tip CPA si asigura unicitatea)

2) **Numere secventiale** – in general o valoare care se incremeneteaza la fiecare instanta a protocolului (utilizate pentru a detecta retransmisii)

3) **Amprente temporale** (timestamp) – asigura unicitate si timeline, previn atacuri de tip retransmisie – **necesita sincronizare temporala slaba**

Sincronizarea temporala slaba

- Element esential in multe protocoale de autentificare (in special in regim broadcast, vezi TESLA, curs 12)
- Principiul unei sincronizari temporale slabe: vezi desen, explicat la tabla



Challenge response cu functii criptografice simetrice

Identificare unilaterala bazata pe numere aleatoare

- **Principiu: utilizeaza un numar aleator ca si challenge**

$$A \rightarrow B: challenge$$
$$B \rightarrow A: response$$

Varianta 1 (poate fi atacata)

- A si B cunosc o cheie secreta k (r_A reprezinta un numar aleator generat de A)

$$A \rightarrow B: r_A$$
$$B \rightarrow A: E_k(r_A)$$

- A verifica identitatea lui B testand ca $E''_k(r_B) = E_k(r_B)$ ($E''_k(r_B)$ denota valoarea primita de A ca raspuns la challenge si care putea fi modificata de un adversar)

Atac prin reflexie asupra Variantei 1

$1.\ A \rightarrow Adv(B):\ r_A$

$1''.\ Adv(B) \rightarrow A:\ r_A$

$2''.\ A \rightarrow Adv(B):\ E_k(r_A)$

$2.\ Adv(B) \rightarrow A:\ E_k(r_A)$

- A a trimis un challenge catre adversarul Adv care pretinde ca este B iar intr-o noua instanta a protocolului Adv pretinde ca este B si ii cere lui A sa se autentifice folosind valoarea de autentificare a lui A ca autentificare pentru B in prima instanta a protocolului
- Rezultat A crede ca comunica cu B dar de fapt comunica cu Adv

Varianta 2 (Varianta 1 reparata in fata atacului prin reflexie)

- A si B cunosc o cheie secreta k (id_A reprezinta identitatea celui care a solicitat autentificarea)

$$A \rightarrow B: r_A$$
$$B \rightarrow A: E_k(r_A, id_A)$$

- A verifica identitatea lui B testand ca $E''_k(r_A) = E_k(r_A)$ si testand si ca identificatorul celui care a trimis challenge-ul este al sau

Atacul prin reflexie nu mai functioneaza asupra Variantei 2

$1.\ A \rightarrow Adv(B):\ r_A$

$1''.\ Adv(B) \rightarrow A:\ r_A$

$2''.\ A \rightarrow Adv(B):\ E_k(r_A, id_A)$

$2.\ Adv(B) \rightarrow A:\ E_k(r_A, id_A)$

- Atacul nu functioneaza pentru ca mesajul nu contine identificatorul lui A; adica $E_k(r_A, id_B) != E_k(r_A, id_A)$

Identificare unilaterala bazata pe time-stamp

- **Principiu: utilizeaza valoarea timpului** (cunoscuta de ambii participanti) **ca challenge**

$$A \rightarrow B: E_k(t_A, id_B)$$

- A verifica identitatea lui B testand ca $E''_k(t_A, id_B) = E_k(t_A, id_B)$ si testand ca timpul este timpul corect din sistem (mai exact se verifica ca timpul apartine unui interval de timp rezonabil, de ex. mesajul de autentificare provine din ultimele 5 secunde)

Identificare mutuala cu numere aleatoare

- **Observatie (legata de autentificarea mutuala):** Simpla dublare a unei autentificari unilaterale nu este in general o solutie deoarece nu exista nici o legatura intre cele doua instante ale protocolului de autentificare unilateral (sursa de atac)
- Autentificare mutuala

$$A \rightarrow B: r_A$$
$$B \rightarrow A: E_k(r_A, r_B, id_A)$$
$$A \rightarrow B: E_k(r_A, r_B)$$

- Se fac aceleasi verificari ca in cazurile precedente, in plus faptul ca valorile aleatoare sunt legate (fiind criptate impreuna) garanteaza caracterul mutual al autentificarii
- **Observatie (valabila pentru toate autentificarile challenge-response):** In orice autentificare challenge –response este recomandabil ca fiecare entitate ce executa o criptare cu o cheie secreta sa aplice functia de criptare nu doar pe valoarea de challenge ci sa mai adauge si o valoare aleatoare proprie (pentru a evita atacuri CPA). O alternativa este si utilizarea unei functii ireversibile cu cheie, de exemplu schimba $E_k(r_A, r_B, id_A)$ cu $MAC_k(r_A, r_B, id_A)$

Challenge-response cu tehnici asimetrice

- Aceleasi principiu al folosirii unei valori de challenge ca in cazul utilizarii tehnicilor simetrice
- Sistemele asimetrice pot fi utilizate in 2 abordari distincte:

a) **Decripteaza** o valoare criptata cu cheia s-a publica

b) **Semneaza digital** valoarea de challenge

Utilizarea criptarii asimetrice in challenge-response

- Varianta 1 (incorecta) – B isi demonstreaza identitatea decriptand mesaje la alegerea lui A

$$A \rightarrow B: E_{PbB}(r_A)$$
$$B \rightarrow A: r_A$$

- **Deficienta**: sursa perfecta de atacuri CCA2 asupra criptosistemului

Utilizarea criptarii asimetrice in challenge-response

- **Autentificare unilaterala prin utilizarea functiei de decriptare si a unei dovezi** (martor ,witness) al faptului ca A cunostea plaintextul: B isi demonstreaza identitatea decriptand mesajul trimis de A si verificand ca $h(r_A)$ corespunde hash-ului mesajului decriptat (deci A cunostea mesajul)

  - $A \rightarrow B: h(r_A), A, E_{PbB}(r_A, A)$
  - $B \rightarrow A: r_A$

- **Autentificare mutuala prin utilizarea functiei de decriptare**

$$A \rightarrow B: E_{PbB}(r_A, A)$$
$$B \rightarrow A: E_{PbA}(r_A, r_B)$$
$$A \rightarrow B: r_B$$

Utilizarea semnaturilor digitale in challenge-response

- Varianta 1 (incorecta)

$$A \rightarrow B: r_A$$
$$B \rightarrow A: Sig_B(r_A)$$

Exploatare asupra variantei 1

- A “pregateste” urmatoarea valoare de challenge pentru B
  - r_A = *h(“Eu sunt B si doresc sa transfer 1.000.000 USD din contul meu in contul lui A”)*
- A trimite valoarea de challenge pentru B

$$A \rightarrow B: r_A$$

$$B \rightarrow A: Sig_B(r_A)$$

- B semneaza deoarece pentru el valoarea hash-ului r_A este un numar aleator fara nici o semnificatie
- A prezinta semnatura digitala a lui B $Sig_B(r_A)$ alaturi de mesaj *m=“Eu sunt B si doresc sa transfer 1.000.000 USD din contul meu in contul lui A”* autoritatii fiscale si obtine 1.000.000 USD
- **Morala**: nu semna mesaje alese integral de alt participant sau nu folosi aceeasi cheie si pentru semnaturi digitale si pentru autentificare (poti sa o folosesti pentru ambele, dar cu precautie)

Utilizarea semnaturilor digitale in challenge-response

- **Autentificare unilaterala cu time-stamp** (se verifica certificatului lui A $cert_A$ la autoritatea competenta (distribuitorul de certificate), validitatea semnaturii si faptul ca timpul la care a fost facuta semnatura este acceptabil)

$$A \rightarrow B: cert_A, t_A, B, Sig_A(t_A, B)$$

- **Autentificare unilaterala cu valori aleatoare** (se verifica certificatului lui A $cert_A$ la autoritatea competenta (distribuitorul de certificate), validitatea semnaturii si faptul ca a fost facuta pe valoarea de challenge)

$$A \rightarrow B: r_A$$
$$B \rightarrow A: cert_B, r_A, A, Sig_B(r_A, r_B, B)$$

- **Autentificare mutuala cu valori aleatoare** (aceleasi verificari ca in cazul precendent de partea ambilor participanti)

$$A \rightarrow B: r_A$$
$$B \rightarrow A: cert_B, r_B, A, Sig_B(r_A, r_B, B)$$
$$A \rightarrow B: cert_A, B, Sig_A(r_B, r_A, A)$$

Autentificare zero-knowledge

- **Principiu**: **demonstreaza o identitate fara a spune absolut nimic cu privire la secretul** in baza careia se face aceasta demonstratie
- Observatie: Protocoalele challenge-response au acest potential dezavantaj, un adversar poate selecta valori de challenge in mod strategic pentru a afla ceva despre secret
- Explicatie pentru copii - Pestera lui Quisquater et. al. Jean-Jacques Quisquater, Louis C. Guillou, Thomas A. Berson. **How to Explain Zero-Knowledge Protocols to Your Children**, *Advances in Cryptology - CRYPTO '89: Proceedings*, v.435, p.628-631, 1990.
- Structura generala a unui protocol z-k consta intr-o sesiune de 3 runde (deoarece argumentul fiecarei sesiuni este probabilistic devine necesara repetarea succesiva a acestei sesiuni cu noi valori)

$$A \rightarrow B: \ witness$$

$$B \rightarrow A: \ challenge$$

$$A \rightarrow B: \ response$$

Autentificarea z-k Fiat-Shamir

- **Obiectiv:** A isi demonstreaza identitatea catre B
- **Setarea cheii de autentificare:**

a) O parte de incredere T publica un modul RSA $n=pq$

b) A alege un numar s din Z_n si calculeaza $v=s^2 \bmod n$

c) v este cheia publica a lui A iar s este cea secreta (v se foloseste la verificare, s pentru generare raspunsului la challenge)

- **Descrierea unei runde:**

a) **A** alege un numar r (commitment) din Z_n si calculeaza si **trimite** $\boldsymbol{x=r^2 \bmod n}$ **(witness)**

b) **B alege o valoare de challenge** $e=1$ sau $e=0$ si trimite e lui A

c) **A calculeaza raspunsul ca** $\boldsymbol{y= r*s^e \bmod n}$

d) **B verifica daca** $\boldsymbol{y^2 = xv^e \bmod n}$

- **Concluzia la final de runda: exista o probabilitate de exact ½ ca Adv(A) sa fii fraudat runda de autentificare** (explicatie pe slide-ul urmator)
- **Concluzia la finalul a** k **runde: exista o probabilitate de exact** $(½)^k$ **ca Adv(A) sa fii fraudat autentificarea**

Cum putea Adv(A) sa fraudeze o runda de autentificare cu probabilitate 1/2

- Cazul 1: **Adv(A)** poate selecta r, **calculeaza** $x=r^2/v$ si **raspunde cu** $y=r$ – acesta este un raspuns **corect pentru cazul in care** $e=1$ dar daca $e=0$ atunci Adv(A) trebuie sa calculeze radacina patrata a lui x, ceea ce este nefezabil
- Cazul 2: **Adv(A) poate raspunde corect in cazul in care** $e=0$ **daca calculeaza** $x=r^2 \bmod n$ dar in acest caz nu mai poate raspunde la $e=1$ pentru ca din nou trebuie sa poata calcula o radacina patrata

- Argument intuitiv pentru faptul ca este autentificare z-k: orice adversar poate simula valori corecte raspuns, dar nu poate poate sa raspunda in fata unei provocari real-time decat cu probabilitate $(½)^k$

Encrypted Key Exchange (EKE)

- Propus de Bellovin si Merritt in 1992, exista in diverse variante (folosind Diffie-Hellman, RSA, 3 participanti etc.)
- Primul protocol de schimb de cheie autentificat bazat pe parole **rezistent la cautari exhaustive** (ale parolei)

$$1.A \rightarrow B : A, E_{pwAB}\left(pk\right)$$

$$2.B \rightarrow A : E_{pk}\left(k\right)$$

$$3.A \rightarrow B : E_{k}\left(N_A\right)$$

$$4.B \rightarrow A : E_{k}\left(N_A, N_B\right)$$

$$5.A \rightarrow B : E_{k}\left(N_B\right)$$

- Protocolul se incheie cu succes daca nonce-urile trimise de A si B sunt verificate in raspunsurile primite
- Atacuri de cautare exhaustiva asupra parolei nu poti fi facute (doar daca cheia publica pk are o structura speciala)

Cap. 4.2. Protocoale de autentificare in sisteme de operare si sisteme bancare

Autentificarea in Windows

- Bazata pe parole, stocate criptat in SAM (Security Accounts Manager)

$$c:\windows\system32\config\SAM$$

- Diverse softuri intitulate pwdump pentru a extrage parola criptata (vezi http://en.wikipedia.org/wiki/Pwdump)

- Parolele sunt criptate sub forma

$$DES_{KD1(password)}(\text{"KGS!@\#\$\%"}) \| DES_{KD2(password)}(\text{"KGS!@\#\$\%"})$$

Problema 1: Utilizarea DES

$$DES_{KD1(password)}(KGS!@\#\$\%\ )||DES_{KD2(password)}(KGS!@\#\$\%\ )$$

- DES este inlocuit ca standard inca din 2001
- Hardware criptografic capabil sa sparga DES in cateva zile (Copacobana in medie 3.5 zile)
- DES are o cheie de 56 biti, in timp ce recomandarile curente NIST sunt

Bits	AES	RSA	ECC	LifeTime
80	x	1024	160-223	Until 2010
112	x	2048	224-255	Until 2030
128	128	3072	256-383	After 2030
192	192	7680	384-511	x
256	256	15360	512+	x

Problema 2: Fara salt

$$DES_{KD1(password)}(KGS!@\#\$\%\ )||DES_{KD2(password)}(KGS!@\#\$\%\ )$$

- Salt este o procedura elementara de intarire a parolelor
- Utilizata in toate distributiile UNIX
- Absenta saltului face posibila atacuri de tip dictionar precalculat
- Exemplu: tabele rainbow pentru a sparge toate parolele de 1 - 14 charactere lower/upper/numbere si caractere speciale se pot cumpara la http://www.rainbowtables.net/products.php#LM

Problema 3: Divizarea parolei !?

$$DES_{KD1(password)}(\text{"KGS!@\#\$\%"}) \,||\, DES_{KD2(password)}(\text{"KGS!@\#\$\%"})$$

- Daca parola este de 14 charactere sau mai scurta este divizata in 2 parole care sunt folosite ca si chei pentru cele doua criptari DES

- Ambele parti ale parolei pot fi atacate independent

- Efecte:
  - a sparge 14 caractere e aproape la fel de usor cu a sparge 7
  - o parola de 8, 9, 10, 11 (discutabil 12 si 13) e mai usor de spart decat una de 7

Solutii

- Nota Microsoft pentru dezactivarea LM Hash
  http://support.microsoft.com/kb/299656

- Parole de peste 14 caractere sunt ascunse cu MD5 in loc de DES (ceva mai sigur)

Bibliografie suplimentara

- Discutie despre autentificare in Windows

B. Groza, A. Alexandroni, I. Silea , V. Patriciu, On the security of some authentication mechanisms from Windows, Buletinul Stiintific al Universitatii Politehnica din Timisoara, Seria Automatica si Calculatoare, ISSN 1224-600X, 2008.

- si cod sursa detaliat pentru toate protocoalele de autentificare din Windows:

E. Glass, “The NTLM Authentication Protocol and Security Support Provider”, http://davenport.sourceforge.net/ntlm.html

Parole in UNIX

- Din nou fisiere criptate, dar de data asta cu salt si o functie mai intensa
- Stocate traditional in */etc/passwd*
- Mutate mai nou in */etc/shadow* (accesibil doar de root)
- Criptate folosind o modificare a MD5 (anterior de folosea DES) aplicata pe parola si salt (comanda *crypt*, vezi *man crypt*)
- Exemplu, fisier *passwd*

  x:x:501:501:x:/home/x:/bin/bash
  Alice:x:502:502:Alice:/home/Alice:/bin/bash
  Bob:x:503:503:Bob:/home/Bob:/bin/bash

- Exemplu, fisier *shadow*

  x:$1$7lSmIrJ4$nFh23Pb8xK8xW7VnHjOGm1:13338:0:99999:7:::
  Alice:1VSxlZUhg$bGH7FRq.5jhcCsDPS.Zdl1:13338:0:99999:7:::
  Bob:1ZRw9H/6L$0SJjGvTxJ2UrBrR2bQ7gA/:13338:0:99999:7:::

Securitate in sisteme bancare (bancomate)

- ATM (Automatic Teller Machines) sunt cel mai larg raspandit si solicitat dispozitiv al comertului
- Aceeasi tehnologie se foloseste si in POS (EFTPOS – Electronic Funds Transfer at the Point of Sales)
- Modul de functionare uzual este: un cod PIN este folosit pentru a cripta contul, valoare este transformata in zecimal si trunchiata rezultat denumit PIN natural. Un offset se poate adauga si se obtine PIN-ul utilizatorului

Account number N (on the mag stripe):	8807012345691715
PIN key KP:	FEFEFEFEFEFEFEFE
Result of DES $\{N\}_{KP}$:	A2CE126C69AEC82D
$\{N\}_{KP}$ decimalized:	0224126269042823
Natural PIN:	0224
Offset:	6565
Customer PIN:	6789

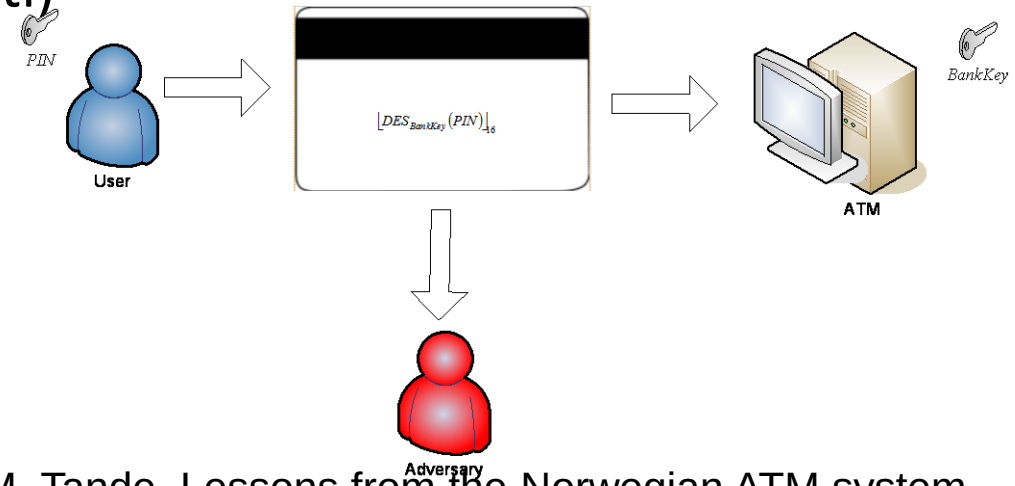
poza din: Ross Anderson, Security Engineering: A guide to Building Dependable Distributed Systems, John Wiley & Sons, 2001

Bancomate Modificate

- Discutie pe baza cazului din imagini
- Vulnerabilitati principale: autentificare bazata pe parole (fara challenge response), autentificare unilaterala (nu bilaterala)
- Solutie: autentificare challenge-response, renuntare la benzi magnetice si utilizare de smart-carduri

Bancomat Norvegian

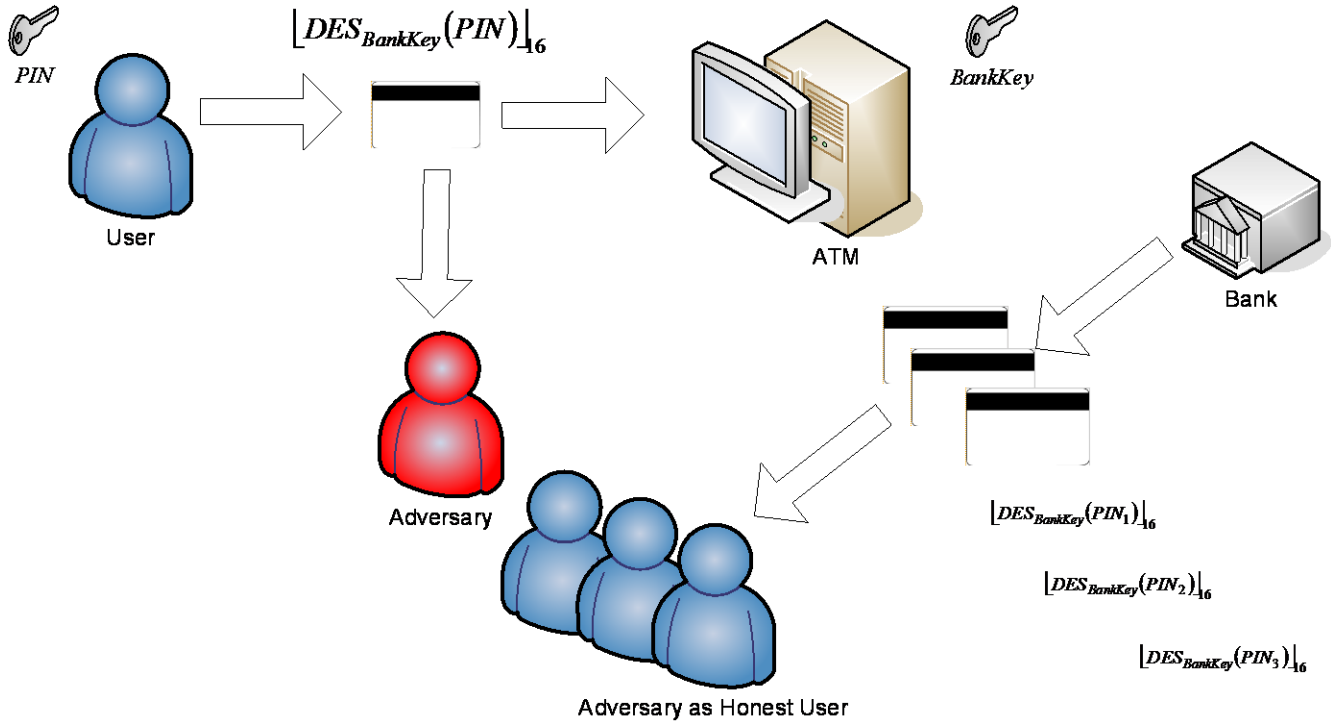
- Un adversar fura un card bancar si doreste sa sparga PIN-ul,
- Mecanismul de criptare are PIN-ului pe card este: $[DES_{BankKey}(PIN)]_{16}$ ($[]_{16}$ denota trunchiere la 16 biti)
- Argumentul de securitate: PIN-ul nu poate fi recuperat deoarece la fiecare cheie DES exista mai multe PIN-uri care se potrivesc



J. Hole, V. Moen, A. N. Klingsheim, and K. M. Tande. Lessons from the Norwegian ATM system. IEEE Security and Privacy, 5(6):2531, 2007.

B. Groza, M. Minea, A formal approach for automatic detection of off-line and undetectable on-line guessing, (to appear in) Proceedings of 14-th Financial Cryptography (FC'10), LNCS, Springer-Verlag, 2010.

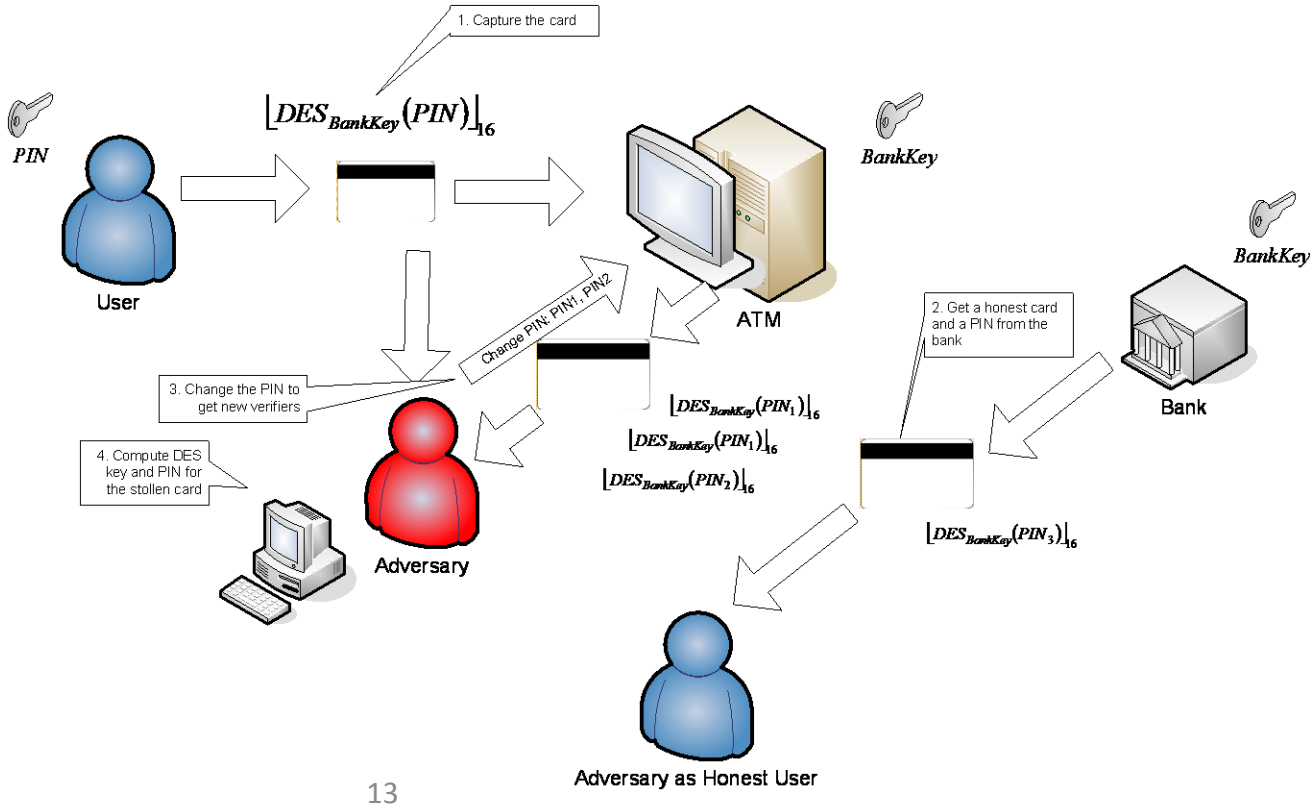
Atac asupra ATM Norvegian



- Atac in doi pasi:
  - Adevrsarul joaca rolul unui participat cinstit si obtine 3 carduri (in medie) din care extrage *BankKey*
  - Folosind *BankKey* adversarul calculeaza PIN (unic de aceasta data)

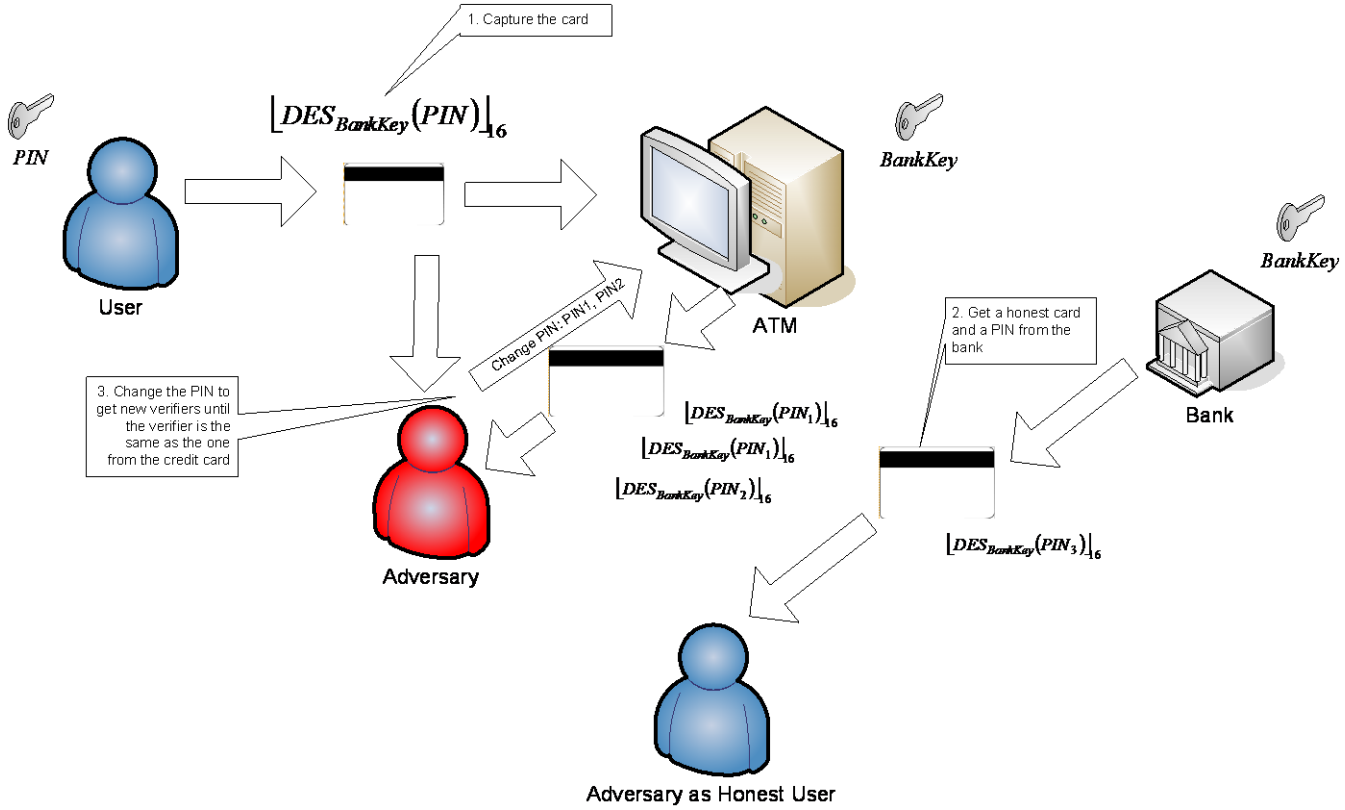
Un atac mai simplu

- Adversarul poate sa evite obtinerea de 3 carduri de la banca daca isi poate schimba PIN-ul pe cardul propriu



Si mai simplu

- Daca PIN-ul este criptat pe card fara alte informatii ale utilizatorului atunci ATM-ul poate fi folosit pentru a extrage PIN-uri



Cap. 4.3. Protocoale de autentificare in retele de calculatoare (EKE, STS, IPSec, SSL/TLS, SSH, Kerkeros).

Encrypted Key Exchange (EKE)

- Propus de Bellovin si Merritt in 1992, exista in diverse variante (folosind Diffie-Hellman, RSA, 3 participanti etc.)
- Primul protocol de schimb de cheie autentificat bazat pe parole **rezistent la cautari exhaustive** (ale parolei)

$$1.A \rightarrow B : A, E_{pwAB}\left(pk\right)$$

$$2.B \rightarrow A : E_{pk}\left(k\right)$$

$$3.A \rightarrow B : E_{k}\left(N_A\right)$$

$$4.B \rightarrow A : E_{k}\left(N_A, N_B\right)$$

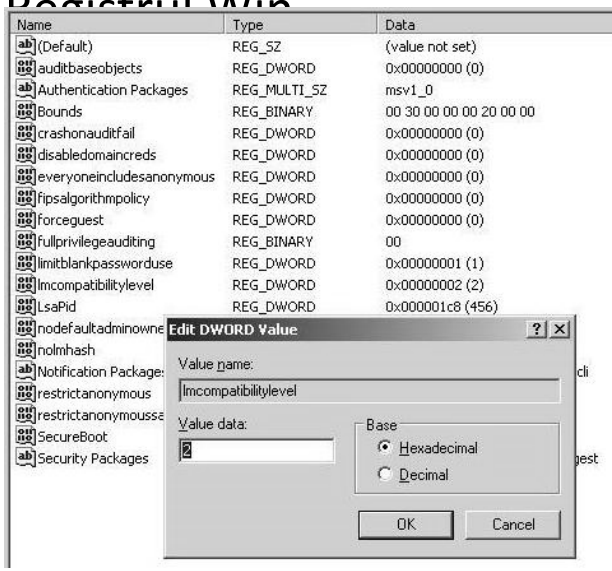
$$5.A \rightarrow B : E_{k}\left(N_B\right)$$

- Protocolul se incheie cu succes daca nonce-urile trimise de A si B sunt verificate in raspunsurile primite
- Atacuri de cautare exhaustiva asupra parolei nu poti fi facute (doar daca cheia publica pk are o structura speciala)

MS-CHAP si NTLM

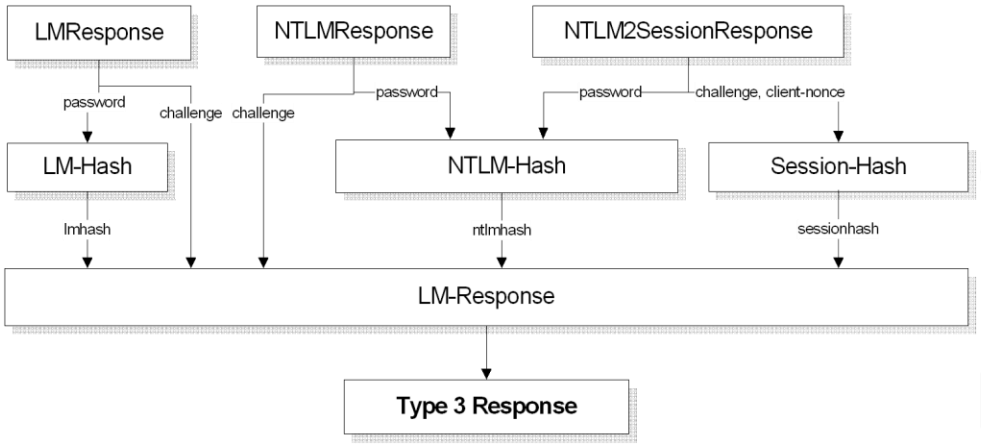
- MS-CHAP (Challenge-handshake authentication protocol) si NTLM (NT LAN Manager) sunt standarde *de facto* in sistemele de operare Microsoft
- Utilizate pentru autentificare utilizatorilor la acces remote catre fisiere, imprimante etc.
- MS-CHAP are 2 variante si NTLM 5 (inclusiv una bazata pe LM-Hash, i.e. DES)
- Toate cele 5 variante de NTLM au la baza un challenge-response in 3 pasi
- Alegerea variantei depinde e valoarea *lmcompatibility* din Registrul Win

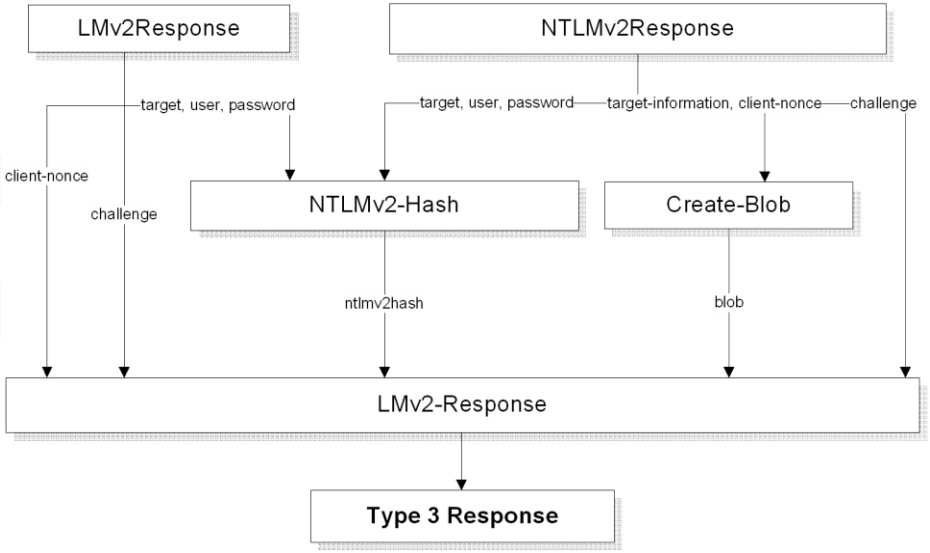
1. *Client → Server : Type 1 Message*
2. *Server → Client : Type 2 Message (includes the 64 bit challenge from the server)*
3. *Client → Server : Type 3 Message (includes the response from the client)*



Variante NTLM

- Trei variante bazate pe DES
- Doua variante bazate pe MD5





$$Client \rightarrow Server:$$
$$DES_{K1}(challenge) \,||\, DES_{K2}(challenge) \,||\, DES_{K3}(challenge)$$

$$Client \rightarrow Server:$$
$$HMAC-MD5_{HMAC-MD5_{MD4(password)}(user||target)}(challenge \,||\, clientNonce)$$

MS-CHAP v2 si NTLM v2 Session

- Ambele (ilustrate in figura dar si toate celelalte variante) sunt vulnerabile la cautari off-line exhaustive ale parolelor
- Pentru detalii vezi: B. Groza, A. Alexandroni, I. Silea , V. Patriciu, On the security of some authentication mechanisms from Windows, Buletinul Stiintific al Universitatii Politehnica din Timisoara, Seria Automatica si Calculatoare, ISSN 1224-600X, 2008.

MS-CHAP v2

1. $A \rightarrow B : A$
2. $B \rightarrow A : N_B$
3. $A \rightarrow B : N_A, H(k_{AB}, N_A, N_B, A)$
4. $B \rightarrow A : H(k_{AB}, N_A)$

NTLMv2-Session

1. $B \rightarrow A : N_B$
2. $A \rightarrow B : N_A, H(k_{AB}), H'(N_A, N_B))$
3. $B \rightarrow A : H(k_{AB}, H'(N_A, N_B)), H'(N_A, N_B)$

ISO/IEC 9798-3 Three-Pass Mutual Authentication

- Relizeaza autentificare mutuala folosind primitive asimetrice

$$1. B \rightarrow A : N_B$$

$$2. A \rightarrow B : Cert_A, TokenAB = N_A \,||\, N_B \,||\, B \,||\, Sig_A\left(N_A \,||\, N_B \,||\, B\right)$$

$$2. B \rightarrow A : Cert_B, TokenBA = N_B \,||\, N_A \,||\, A \,||\, Sig_B\left(N_B \,||\, N_A \,||\, A\right)$$

- Varianta initiala (se observa semnarea de catre B a unui nonce nou, masura care se credea ca imbunatateste securitatea deoarece A nu stie apriori ce mesaj va semna B)

$$1. B \rightarrow A : N_B$$

$$2. A \rightarrow B : Cert_A, TokenAB = N_A \,||\, N_B \,||\, B \,||\, Sig_A\left(N_A \,||\, N_B \,||\, B\right)$$

$$2. B \rightarrow A : Cert_B, TokenBA = N_B' \,||\, N_A \,||\, A \,||\, Sig_B\left(N_B' \,||\, N_A \,||\, A\right)$$

Atacul lui Wiener (Atacul Canadian)

- Descoperit de Wiener, membru al partii canadiene din comitetul ISO

- Pasul 1: Adv pretinde ca este B si incepe o conversatie cu A

$$1.Adv(B) \rightarrow A : N_{adv}$$

$$2.A \rightarrow Adv(B) : Cert_A, TokenAB = N_A \,||\, N_{adv} \,||\, B \,||\, Sig_A\left(N_A \,||\, N_{adv} \,||\, B\right)$$

- Pasul 2: Adv pretinde ca este A si incepe o conversatie cu B

$$1'.Adv(A) \rightarrow B : N_A$$

$$2'.B \rightarrow Adv(A) : Cert_B, TokenBA = N_B \,||\, N_A \,||\, A \,||\, Sig_B\left(N_B \,||\, N_A \,||\, A\right)$$

- Pasul 3: Adv incheie cu succes conversatia inceputa cu A in pasul 1 folosind mesajul primit de la B in pasul 2

$$3.Adv(B) \rightarrow A : Cert_B, TokenBA = N_B \,||\, N_A \,||\, A \,||\, Sig_B\left(N_B \,||\, N_A \,||\, A\right)$$

- Final: A crede ca B a inceput cu el o conversatie si a incheiat-o cu succes, in timp ce B crede ca A a inceput cu el conversatia si asteapta in continuare mesajul 3

Kerberos

- Dezvoltat la MIT ca parte a proiectului Athena, folosit in special in Windows, dar si pe Unix
- Ideea de baza: utilizarea unei parti de incredere pentru distributie de chei de sesiune ce pot fi utilizate intre utilizator si diverse servere
- Compus din 3 sub-protocoale:
  - Authentication Service Exchange (AS) – ruleaza intre C si AS
  - Ticket-Granting Service Exchange (TGS) – ruleaza intre C si TGS
  - Client/Server Authentication Application Exchange (AP) – ruleaza intre C si AP
- Diviziunea intre AS si TGS este utila pentru cazul in care serverele de aplicatie apartin la domenii diferite (deci U poate fi deservit de un singur AS dar de mai multe TGS)

Participanti la protocol

- U – utilizator al serviciului de single-sign-on (SSO)
- C – aplicatie client invocata de user
- S – server de aplicatie (care detine resursele necesare aplicatiei)
- KDC – Key Distribution Center format din 2 componente:
  - AS – Authentication Server – serverul la care C se autentifica (in faza intiala folosind o parola fixata prin alt mijloc decat Kerberos)
  - TGS – Ticket Grantig Server – elibereaza “bilete” pe care C le foloseste pentru a se autentifica la S

- Diviziunea intre AS si TGS este utila pentru cazul in care serverele de aplicatie apartin la domenii diferite (deci U poate fi deservit de un singur AS dar de mai multe TGS)

Protocolul Kerberos

$$1.C \rightarrow AS : U, TGS, LifeTime, N_1$$

$$2.AS \rightarrow C : U, T_{C,TGS} = Enc_{K_{AS,TGS}} \{U, C, TGS, K_{C,TGS}, TimeStart, TimeExp\},$$

$$TGT_C = Enc_{K_U} \{TGS, K_{C,TGS}, TimeStart, TimeExp, N_1\}$$

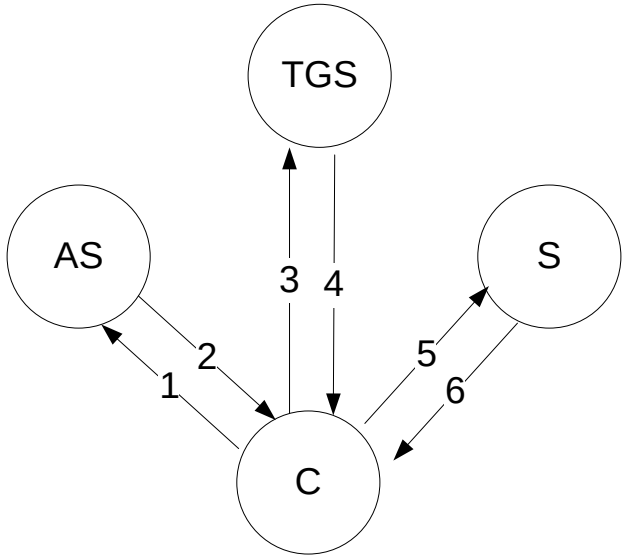
$$3.C \rightarrow TGS : S, LifeTime, N_2, T_{C,TGS}, A_{C,TGS} = \{C, ClientTime\}_{K_{C,TGS}}$$

$$4.TGS \rightarrow C : U, T_{C,S} = E_{K_{S,TGS}} \{U, C, S, K_{C,S}, TimeStart, TimeExp\},$$

$$TKT_C = E_{K_{C,TGS}} \{S, K_{C,S}, TimeStart, TimeExp, N_2\}$$

$$5.C \rightarrow S : T_{C,S}, A_{C,S} = E_{K_{C,S}} \{C, ClientTime\}$$

$$6.S \rightarrow C : A_{S,C} = E_{K_{C,S}} \{ClientTime\}$$



Station to station protocol (STS)

- Propus de Diffie, Oorschot si Wiener in 1992 (nume grele)
- Si are cinci proprietati grele de securitate:
  - Autentificare mutuala
  - Schimb mutual de cheie
  - Confirmare mutuala a cheii
  - Perfect forward secrecy – chiar daca o cheie privata este compromisa, nu pot fi aflate chei anterior folosite
  - Anonimitate – un observator din retea nu poate demonstra ca e vorba de comunicare intre 2 participanti anume

$$1.A \to B : \alpha^x$$

$$2.B \to A : \alpha^y, Cert_B, E_k\left(sig_B\left(\alpha^y, \alpha^x\right)\right)$$

$$3.A \to B : Cert_A, E_k\left(sig_B\left(\alpha^x, \alpha^y\right)\right)$$

Propus alaturi de varianta STS “Authentication only”

- Autorii sustin ca este in esenta ISO Public Key Three-Pass Mutual Authentication

$$1.A \rightarrow B : N_A$$
$$2.B \rightarrow A : Cert_B, N_B, sig_B\left(N_B, N_A\right)$$
$$3.A \rightarrow B : Cert_A, sig_A\left(N_A, N_B\right)$$

- Din pacate are o scapare esentiala: identitatile participantilor nu sunt semnate
- In atacul de mai jos: A vorbeste cu Adv iar B crede ca vorbeste cu A dar de fapt vorbeste tot cu Adv

$$1.A \rightarrow Adv : N_A$$
$$2.Adv\left(A\right) \rightarrow B : N_A$$
$$3.B \rightarrow Adv\left(A\right) : Cert_B, N_B, sig_B\left(N_A, N_B\right)$$
$$4.Adv \rightarrow A : Cert_B, N_B, sig_B\left(N_A, N_B\right)$$
$$5.A \rightarrow Adv : Cert_A, sig_A\left(N_A, N_B\right)$$
$$6.Adv\left(A\right) \rightarrow B : Cert_A, sig_A\left(N_A, N_B\right)$$

Atacul lui Lowe'94 asupra STS

- Considerat un atac minor, are urmatorul rezultat A crede ca vorbeste B, B crede ca vorbeste cu Adv intr-o sesiune incompleta
- Numit si atac DoS deoarece pentru A comunicatia post-autentificare nu continua (Adv tot nu stie cheia secreta) cu toate ca protocolul se incheie corect

$$1.Adv(B) \rightarrow A : start$$

$$2.A \rightarrow Adv(B) : \alpha^x$$

$$3.Adv \rightarrow B : \alpha^x$$

$$4.B \rightarrow Adv : \alpha^y, Cert_B, E_K\left(sig_B\left(\alpha^y, \alpha^x\right)\right)$$

$$5.Adv(B) \rightarrow A : \alpha^y, Cert_B, E_K\left(sig_B\left(\alpha^y, \alpha^x\right)\right)$$

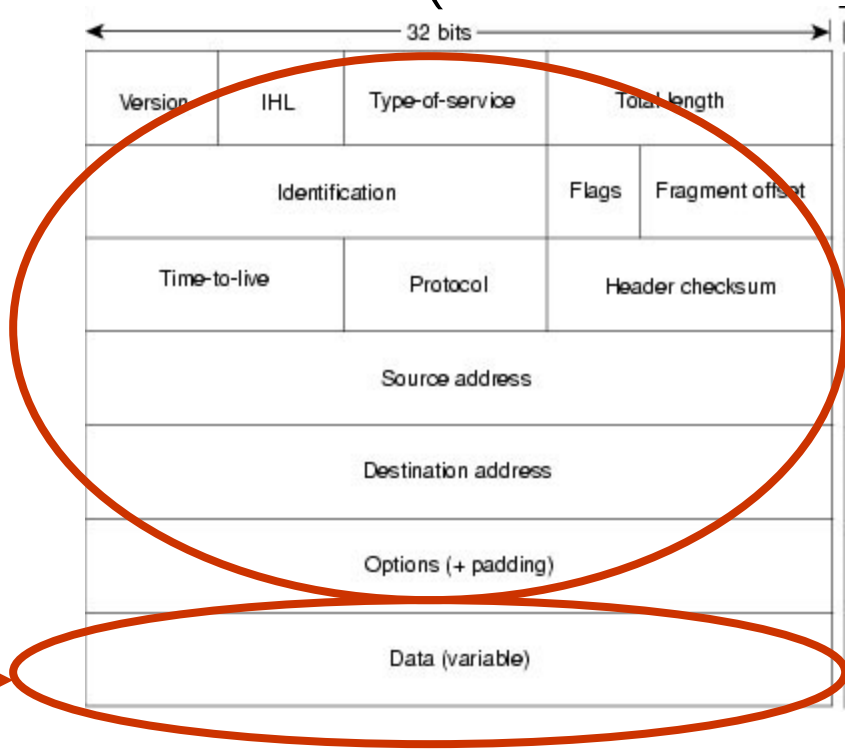
$$6.A \rightarrow Adv(B) : Cert_A, E_K\left(sig_A\left(\alpha^x, \alpha^y\right)\right)$$

Internet Security - IPSec

- Obiectiv: **obligatoriu autentificare** si **optional confidetialitate**
- Se doreste asigurarea protectiei pentru un header IP
- Motivatie: In absenta protectiei sunt posibile diverse atacuri (oricare din atacurile mentionate, de. ex. reflectie)

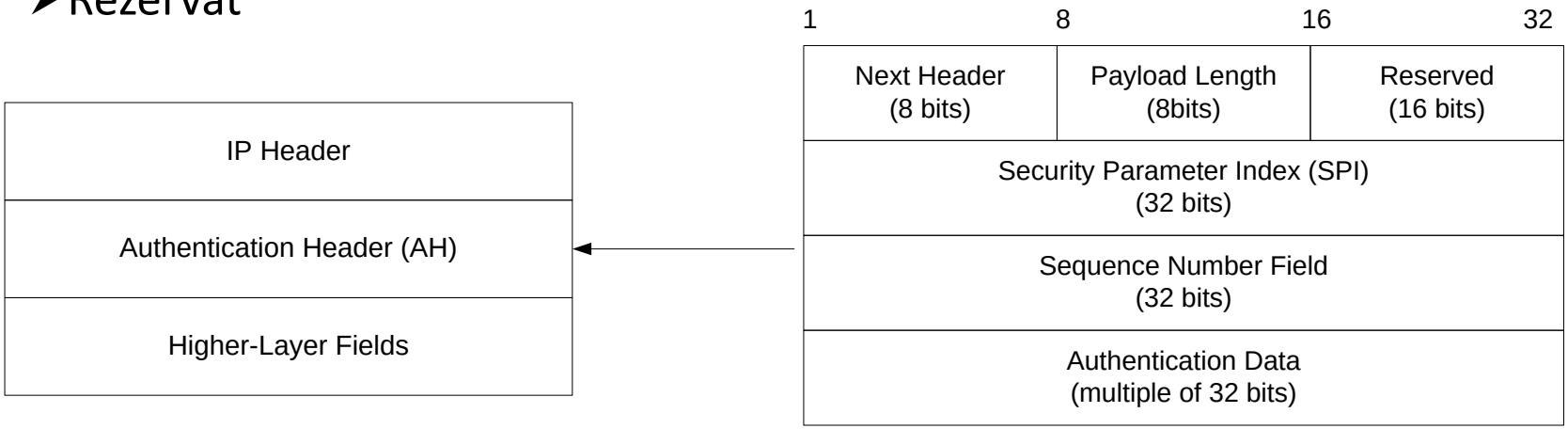
header IP

Campuri pentru layerul de nivel inalt (7)



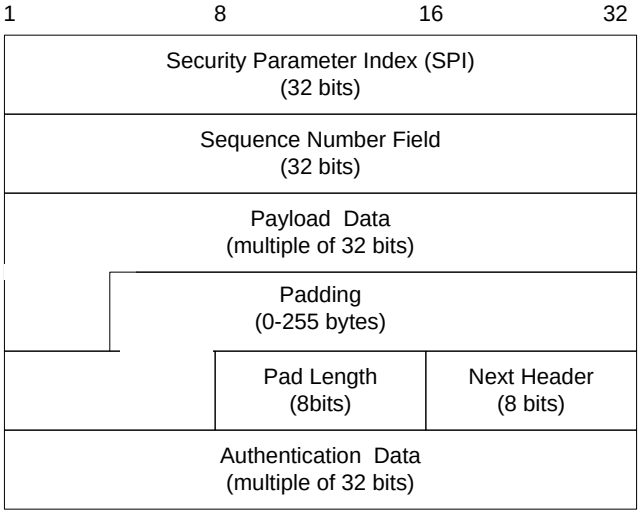
Asigurarea autenticitatii in IPSec

- Introducerea unui Authentication Header (AH)
- Campuri pentru securitate:
  - Security Parameter Index (SPI) – specifica modul de functionare (tunel/transport) si algoritmii utilizati (HMAC-SHA1, 3DES, AES)
  - Sequence Number – contor pentru a evita atacuri de tip replay
- Campuri fara semnificatie in securitate:
  - Next Header – tipul urmatorului header
  - Payload Legth – lungimea headerului
  - Rezervat



Asigurarea confidentialitatii in IPSec

- Blocuri de 32 de biti numite ESP (Encapsulating Security Payload) este alocata dupa un AH
- Campurile au aceeasi semnificatie
- Payload Data sunt datele criptate si padding este folosit pentru ca lungimea sa fie multiplu de 32 biti
- Authentication Data este acum AH si este optional (greseala fatala!)



Internet Key Exchange (IKE)

- Necesar pentru a stabili cheia in IPSec (pentru a stabili relatia de Security Association (SA) intre doua noduri)
- IKE este o suita de protocoale de schimb de cheie autentificat, majoritatea bazate pe Diffie-Hellman
- IKE este compus din 2 faze (Phase1 + Phase2)
- Faza 1 are 8 variante este un schimb de cheie cu autentificare mutuala
- Faza 2 (denumita si quick mode) poate avea loc de mai multe ori dupa faza 1si poate fi folosita pentru a stabili mai multe tipuri de conexiuni cu diferiti parametrii de securitate: integrity only, encryption only etc.

IKE – Faza 1

- Exista 8 moduri de faza 1 deoarece sunt 4 tipuri de chei (pre-shared symmetric, old-style public key, new-style public key si public signature verification key) si 2 tipuri de schimb (main mode si aggressive mode)
- Modul main schimba 6 mesaje (3+3) si este obligatoriu
- Modul aggressive schimba 3 mesaje si este optional

Signature-based IKE Phase 1 Main Mode

$1. A \rightarrow B : HDR_A, SA_A$

$2. B \rightarrow A : HDR_B, SA_B$

$3. A \rightarrow B : HDR_A, g^x, SA_A$

$4. B \rightarrow A : HDR_B, g^y, SA_B$

$5. A \rightarrow B : HDR_A, E_{g^{xy}}\left(ID_A, Cert_A, Sig_A\right)$

$6. B \rightarrow A : HDR_B, E_{g^{xy}}\left(ID_B, Cert_B, Sig_B\right)$

- Inrudit cu protocolul STS, difera prin criptarea certificatelor (utilizata pentru anonimitate) si semnarea cheii stabilite

$$Sig_A = Sig_A\left(M_1\right)$$

$$M_1 = prf_1\left(prf_2\left(N_A \| N_B \| g^{xy}\right) \| g^x \| g^y \| C_A \| C_B \| SA_A \| ID_A\right)$$

$$Sig_B = Sig_B\left(M_2\right)$$

$$M_2 = prf_1\left(prf_2\left(N_A \| N_B \| g^{xy}\right) \| g^x \| g^y \| C_B \| C_A \| SA_B \| ID_B\right)$$

HDR – headere care contin cookie C

SA – security association care negociaza parametrii de securitate: algoritmi de criptare, semnare, hash etc. (se pot propune mai multe variante si se raspunde cu alegerea uneia)

ID – identitatile participantilor

Atac similar atacului lui Lowe

- Atacul functioneaza in primul rand pentru ca mesajul semnat contine doar identitatea celui care semneaza, nu si cea a presupusului partener
- In urma atacului B crede ca vorbeste cu A, in timp ce A crede ca vorbeste cu Adv intr-o sesiune neterminata

$$1.A \rightarrow Adv : HDR_A, SA_A$$

$$1'.Adv(A) \rightarrow B : HDR_A, SA_A$$

$$2'.B \rightarrow Adv(A) : HDR_B, SA_B$$

$$2.Adv \rightarrow A : HDR_B, SA_B$$

$$3.A \rightarrow Adv : HDR_A, g^x, N_A$$

$$3'.Adv(A) \rightarrow B : HDR_A, g^x, N_A$$

$$4'.Adv(B) \rightarrow A : HDR_B, g^y, N_B$$

$$4.Adv \rightarrow A : HDR_B, g^y, SA_B$$

$$5.A \rightarrow Adv : HDR_A, E_{g^{xy}}(ID_A, Cert_A, Sig_A)$$

$$5'.Adv(A) \rightarrow B : HDR_A, E_{g^{xy}}(ID_A, Cert_A, Sig_A)$$

$$6.'B \rightarrow Adv(A) : HDR_B, E_{g^{xy}}(ID_B, Cert_B, Sig_B)$$

$$6.Abandon$$

IKE – Aggressive Mode

- Simplificare a modului de baza, prin reducere de la 6 la 3 mesaje
- Un atac similar cu cel al lui Lowe este fezabil

$$1.A \rightarrow B : HDR_A, SA_A, g^x, N_A, ID_A$$
$$2.B \rightarrow A : HDR_B, SA_B, g^y, N_B, ID_B, Cert_B, Sig_B$$
$$3.A \rightarrow B : HDR_A, Cert_A, Sig_A$$

Probleme ale IPSec si IKE

- Permite atacuri DoS prin faptul ca un participant poate fi determinat sa desfasoare operatii criptografice intense fara a avea vreo garantie cu privire la partenerul de comunicare
- Complexitatea ridicata a protocolului (multe variante si flexibilitate) nu este un factor pozitiv de securitate
- Schneier si Ferguson critica complexitatea ca efect de comitet
- IKE v2 repara parte din probleme lui IKE
- IKE v2 ofera anonimitate (plausible deniability), obiectiv dorit in diverse servicii IP

Secure Shell (SSH) remote login protocol

- Secure Shell (SSH) protocol de autentificare bazat pe cheie publica intre un server si un client
- Standard pe sistemele UNIX, exista si pentru Win
- Obiectiv: crearea unui canal sigur (autentic si confidential) pentru executia de comenzi, mutarea de fisiere etc.
- Trei componente majore:
  - SSH Transport Layer – autentifica serverul catre client si creeaza un canal sigur intre client si server (exista 2 strategii posibile: clientul are o baza de date cu chei publice ale serverelor de incredere sau clientul cunoaste autoritatea care a semnat certificatul serverului)
  - SSH User Authentication Protocol – autentifica clientul catre server (folosind chei publice sau mai uzual parole)
  - SSH Connection Protocol – este un tunel criptat prin care se trimit comenzile

Schimbul de cheie SSH

- Semnificatii deja cunoscute ale valorilor
- V_C, V_S – versiunea protocolului
- I_C, I_S – valori intiale (schimbate inainte de a se ajunge la pasul curent)

$$1.C \rightarrow S : e = g^x \bmod p$$

$$2.S \rightarrow C : K_S \parallel f = g^y \bmod p \parallel s = Sig_S\left(H\left(V_C \parallel V_S \parallel I_C \parallel I_S \parallel K_S \parallel e \parallel f \parallel K\right)\right)$$

$$3.C \rightarrow S : verificare : K = f^x, Ver_S\left(s\right)$$

- Daca verificare de la pasul 3 se incheie cu succes, se trece la autentificarea clientului (uzual folosind parole) pe un canal sigur (criptat)

SSL si TLS

- Protocol de autentificare unilaterala (de cele mai multe ori) sau mutuala destinat in special pentru WorldWideWeb
- Destinat socketurilor de aici numele de SecureSocketLayer
- Dezvoltat initial de Netscape, si ulterior acceptat ca standard de toti dezvoltatorii inclusiv Microsoft
- Evolueaza in Transport Layer Security (TLS) sub corpul de standardizare Internet Engineering Task Force (IETF)
- Ruleaza sub nivelul aplicatie (HTTP, IMAP etc.) si deasupra nivelelor retea TCP/IP
- Este format din 2 componente
  - TLS Record Protocol – calculeaza MAC-uri (integritate/autenticitate), cripteaza (confidentialitate), optional compreseaza, si la receptie decripteaza/verifica
  - TLS Handshake Protocol – componenta de autentificare, negociere a algoritmilor si a cheii

Structura unui handshake TLS

- Mesajele Hello au ca scop stabilirea valorilor: versiune protocol, random, id sesiune, algoritmi de criptare, compresie
- Certificatele sunt in formatul X.509
- KeyExchange contine partea de cheie Diffie-Hellman (dar functioneaza si cu schimb de cheie dinspre client folosind RSA)
- Valorile marcate cu ^{1} sunt optionale

$$1.C \rightarrow S : ClientHello$$

$$2.S \rightarrow C : ServerHello, ServerCertificate^1, ServerKeyExachange^1, CertificateRequest^1, ServerHelloDone$$

$$3.C \rightarrow S : ClientCertificate^1, ClientKeyExchange, CertificateVerify^1, ClientFinished$$

$$4.S \rightarrow C : ServerFinished$$

- Uzual clientul ramane anonim si nu trimite certificat

Securitatea SSH si SSL/TLS

- Ofera un nivel suficient de bun de securitate pentru majoritatea utilizatorilor
- Nu au vulnerabilitati semnificative (variantele noi si corect implementate)
- Sunt susceptibile la atacuri prin temporizare (side-channel)