Digitální podpisy založené na asymetrické kryptografii

* předpokládejme, že algoritmus pro šifrování a dešifrování má následující vlastnost: P = D(E(P)) = E(D(P))
 - pak může Alice podepsat p pomocí svého tajného dešifrovacího klíče D_A a poslat Bobovi: D_A(P)
 - Bob může zprávu ověřit pomocí veřejného šifrovacího klíče Alice E_A: P = E_A(P)

* de-facto standard pro digitální podpisy je RSA
* NIST standardizoval DSS (Digital Signature Standard), používá se méně protože je 10 až 40x pomalejší než RSA

Certifikáty veřejných klíčů

* potřebujeme zajistit, aby veřejné klíče mohly být bezpečně přenášeny po nezabezpečeném médiu

* mějme důvěryhodnou třetí stranu, nazýváme jí certifikační autorita (CA)
 - CA vytvoří vlastní dvojici klíčů, veřejný předá každému uživateli a tajný utají
 - pokud uživatel chce zveřejnit vlastní veřejný klíč, dodá ho CA
  . CA ověří fyzickou identitu předkladatele klíče
  . CA připojí řetězec identifikující tvůrce klíče a další data (dobu platnosti apod.)
  . CA data podepíše (data = klíč + identita předkladatele atd.)
 - tj. důvěru kterou máme v CA můžeme přenést i na klíč jím podepsaný
 - možnost hierarchie certifikačních autorit
 - podepsané klíče můžeme zveřejnit na serverech veřejných klíčů

* standardní formáty: X.509 (ISO 9594-8), PKCS#6
* X.509 je součást standardu ISO 9594-1 (X.500) pro adresářové služby v ISO/OSI sítích, adresář = databáze informací, stromová struktura položek

Jednosměrné hashovací fce a jejich použití

* šifrování s veřejným klíčem je pomalé, bezpečnost závisí na obsahu P
* proto je lepší použít jednosměrnou hashovací fci (nazývána message digest, MD) a posílat zprávu (P, D_A(MD(P))) místo D_A(P) (De Jonge & Chaum 1987)

Hashovací fce musí mít 3 základní vlastnosti:

1. z P je snadné spočíst MD(P)
2. k MD(P) je výpočetně neproveditelné najít P
3. najít dvě zprávy které dávají stejné MD(x) je neproveditelné

* kolik operací je zapotřebí pro zfalšování m-bitového hashe?
 - je to 2^{m/2} za pomocí útoku nazvaného ”birthday attack” (Yuval 1979)
 - název vychází z otázky: kolik studentů musíte mít, aby pravděpodobnost že budete mít dva lidi s narozeninami ve stejný den byla > 1/2?
 - ve skutečnosti 23 (s 23 lidmi můžeme vytvořit (23*22)/2 = 253 párů, každý z nich má pravděpodobnost 1/365)
 - obecněji: pokud máme N vstupů, existuje N*(N-1)/2 vstupních párů; je K možných výstupů, každý z nich má pravděpodobnost 1/K, tj. pravděpodobnost dvou různých výstupů se stejným vstupem > 1/2 pokud N > sqrt(K)
 - tj. pokud máme hash 64 bitů, stačí nám vygenerovat cca 2^32 zpráv a dívat se po zprávách se stejným výstupem

Např. budeme mít podepsanou zprávu a chceme vygenerovat jinou zprávu se stejným MD - vytvoříme zprávu s 32 možnostmi:

  Vážený [pane | investore],

  v [tomto dopise | této zprávě] bych Vám rád [sdělil | popsal] [svůj

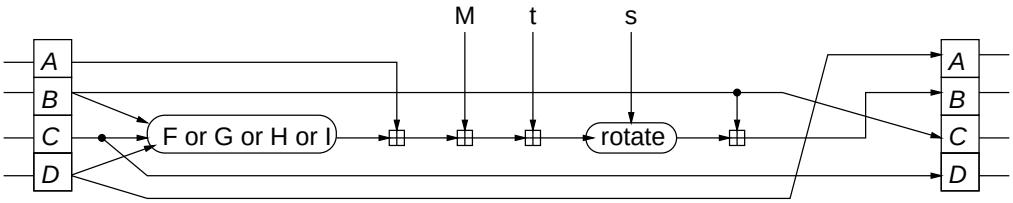
názor | své mínění] o zamýšlené [transakci | investici] do
Nigérijského ropného [průmyslu | zařízení].

* spočteme 2^32 hashů, je možné že některý bude odpovídat původnímu
* pokud ne, můžeme vyzkoušet přidat další možnosti
* existují i varianty nevyžadující tolik paměti

* to znamená, že pokud chceme používat hash ve spojení např. s digitálními
  podpisy, měl by výsledek být nejméně 128 bitů dlouhý, raději delší
 - první varianty založené na DESu už v 1977
 - hledaly se fce s delším výstupem - MDC2 a MDC4 128 bitů - pomalé a další
   problémy
 - MD4 (Rivest 1991, RFC1320), z MD4 vychází většina novějších
  . MD5 (Rivest 1992 v RFC1321), RIPEMD (den Boer 1992) - 128 bitů, pokusy o
    delší hash protože 2^64 operací začalo být považováno za realizovatelné
  . HAVAL (Zheng & al. 1993) - do 256 bitů
  . SHA, SHA-1 (NIST 1994 a 1995 (FIPS 180-1)) - 160 bitů
   . SHA-1 je revize SHA, ve které NSA našla problémy
  . RIPEMD-160 (Dobbertin 1996) - 160 bitů
  . SHA-192, SHA-224, SHA-256 (NIST) - do 256 bitů

Algoritmus MD5
....

* je složitější, popíšu pouze základní princip; podrobnosti RFC1321
 - používá čtyři nelineární fce, v jazyce C zapsané:
    F(X,Y,Z) = (X & Y) | ((~X) & Z) // if X then Y else Z (po bitech)
    G(X,Y,Z) = (X & Z) | (Y & (~Z)) // if Z then X else Y (po bitech)
    H(X,Y,Z) = X^Y^Z   // parita
    I(X,Y,Z) = Y^(X | (~Z))
 - pro každý 512 bitový blok 4 iterace, v každé iteraci 64 kroků
   používajících jednu z fcí F, G, H a I



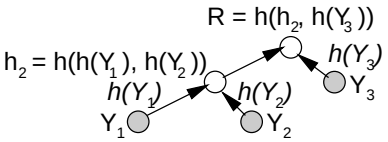
 - zpráva se před zpracováním doplní řetězcem bitů 1000...0 a 64 bitovou
   délkou zprávy na násobek 512 bitů

* algoritmus MD5 se používá nejčastěji, např. v Linuxu program md5sum(1)
* existuje modifikace MD5 pro použití jako klíčované hashovací fce
  (viz RFC1828 nebo HAC, alg. 9.69 na str. 358)

Časová razítka
....

* Alice potřebuje mít potvrzení, že nějaký dokument existoval k určitému datu
* jak zajistit, aby mohla prokázat že dokument existoval, např. pokud bude
  žádat o udělení patentu?

1. Alice vytvoří jednosměrný hash dokumentu, výsledek h(D) pošle důvěryhodné
   třetí straně (timestamping service)
2. třetí strana vytvoří: S(h(D), t1) kde S = podpis, t1 = datum příchodu zprávy
   a pošle výsledek zpět
3. jednou za časovou jednotku (den, týden) třetí strana vytvoří z časových
   razítek tzv. autentizační strom, kde společný uzel má hodnotu h(h1, h2)
 - Alici pošle cestu od jejího časového razítka je kořeni
 - kořenovou hodnotu publikuje na na veřejném místě, např. v novinách



Autentizace v UNIXových systémech
.....

* některé starší systémy uchovávají hesla v OT, chráněné přístupovými právy
 - nechrání před privilegovanými uživateli
 - problém v případě zálohování systému

* lepší - místo hesla X uchovat h(X)
 - uživatel zadá heslo X, systém spočte h(X) a ověří zda je h(X) v souboru s hesly
 - problém - slovníkové útoky - oponent vytvoří slovník pravděpodobných hesel, provede h(X) a výsledek porovná s existujícími ”zašifrovanými” hesly
  . zpomalení mapování, např. místo jednoduché fce se použije 25x
  . snížení efektivity oponenta - salting

* salting (”osolení”)
 - vytvoříme t-bitovou náhodnou hodnotu y, tzv. salt
 - spočteme H=h(y, X), výsledek H spolu se hodnotou y uložíme do souboru s hesly
 - např. UNIX používá 12 bitový salt odvozený ze systémového času v době vytvoření hesla
 - Oskar může provádět slovníkové útoky na jednotlivá hesla, ale zvýší se obtížnost slovníkových útoků na velké množiny hesel

Jednorázová hesla
.....

* v některých kritických aplikacích (banky) dostane uživatel sešit jednorázových hesel, problém ztráty sešitu

* Lamportovo schéma pro bezpečné přihlašování po nezabezpečené síti (Lamport 1981)
 - uživatel zvolí tajné heslo X a počet jednorázových hesel N (např. 4)
  . první heslo bude P1=h(h(h(h(X))), druhé P2=h(h(h(X)), třetí P3=h(X), P4=X
  . Oskar je z odposlechu schopen spočíst předcházející, ale nikoli následující
 - server bude mít P=h(P1) a počet iterací I=1
  . při prvním přihlášení ověří h(P1) a zvýší I o 1
  . při druhém přihlášení ověří h(h(P2) atd.

Standardy PKCS
---

* de facto standardy vytvořené RSA Laboratories, viz http://www.rsa.com
* název ”The Public-Key Cryptography Standards” (PKCS), často se prakticky používají

* PKCS#1 - šifrování a podepisování pomocí RSA (podpis viz HAC str. 445)
* PKCS#3 - Diffie-Hellmanovo dohadování klíče
* PKCS#5 - symetrické šifrování klíčem odvozeným z hesla
* PKCS#6 - syntaxe rozšířených certifikátů (nadmnožina X.509)
* PKCS#7 - syntaxe šifrovaných zpráv
* PKCS#8 - syntaxe soukromých klíčů
* PKCS#9 - vybrané typy atributů
* PKCS#10 - syntaxe požadavku na certifikaci
* PKCS#11 - rozhraní pro práci s kryptografickým ”tokenem” (CRYPTOKI), dovoluje vykonávat kryptografické operace na např. kartě

Závěr
---

* nebezpečí kryptografie - dá se ”téměř měřit” (délka klíčů apod.)
* problémem je, že mohou být lepší útoky:
 - PGP zašifruje text, ale původní OT zůstane na disku; pokud zrušíme ”rm/DEL”, zůstanou bloky; na některých FS utility jako ”wipe” (jak smazat soubor viz http://www.cs.auckland.ac.nz/~pgut001/secure_del.html)
 - problémy se SW; je složitý, mnoho šancí na únik citlivých informací
 - problémy s přístupovou cestou k SW; pokud přenášíte soubory nebo zadáváte heslo po otevřené síti (telnet)
 - nevhodná rozhraní - pokud to dá práci, nebudou to uživatelé používat
 - pokud jsou klíče generovány pomocí slabého RNG nebo PRNG, nemají předpokládanou entropii; podobně slabá hesla
 - není místo pro bezpečné uchování klíče; šifrování chrání ”velká” tajemství pomocí ”malých” klíčů, ale klíče jsou většinou uchovány v paměti apod.
 - útoky na protokoly, viz dříve uvedené příklady
 - útoky založené na jiných než datových vlastnostech: timing attacks, přeslechy, elektromagnetické vyzařování apod.

* největší praktický problém - lidé vyžadují funkčnost a bezpečnost je nezajímá; kolik lidí používá PGP?
 - proto systémy nepodporují bezpečnost jako základní vlastnost

* doporučení
 - používat publikované algoritmy a protokoly, sledovat literaturu
 - kriticky přemýšlet o tom, co je chráněno a jak mohou vypadat útoky

❇