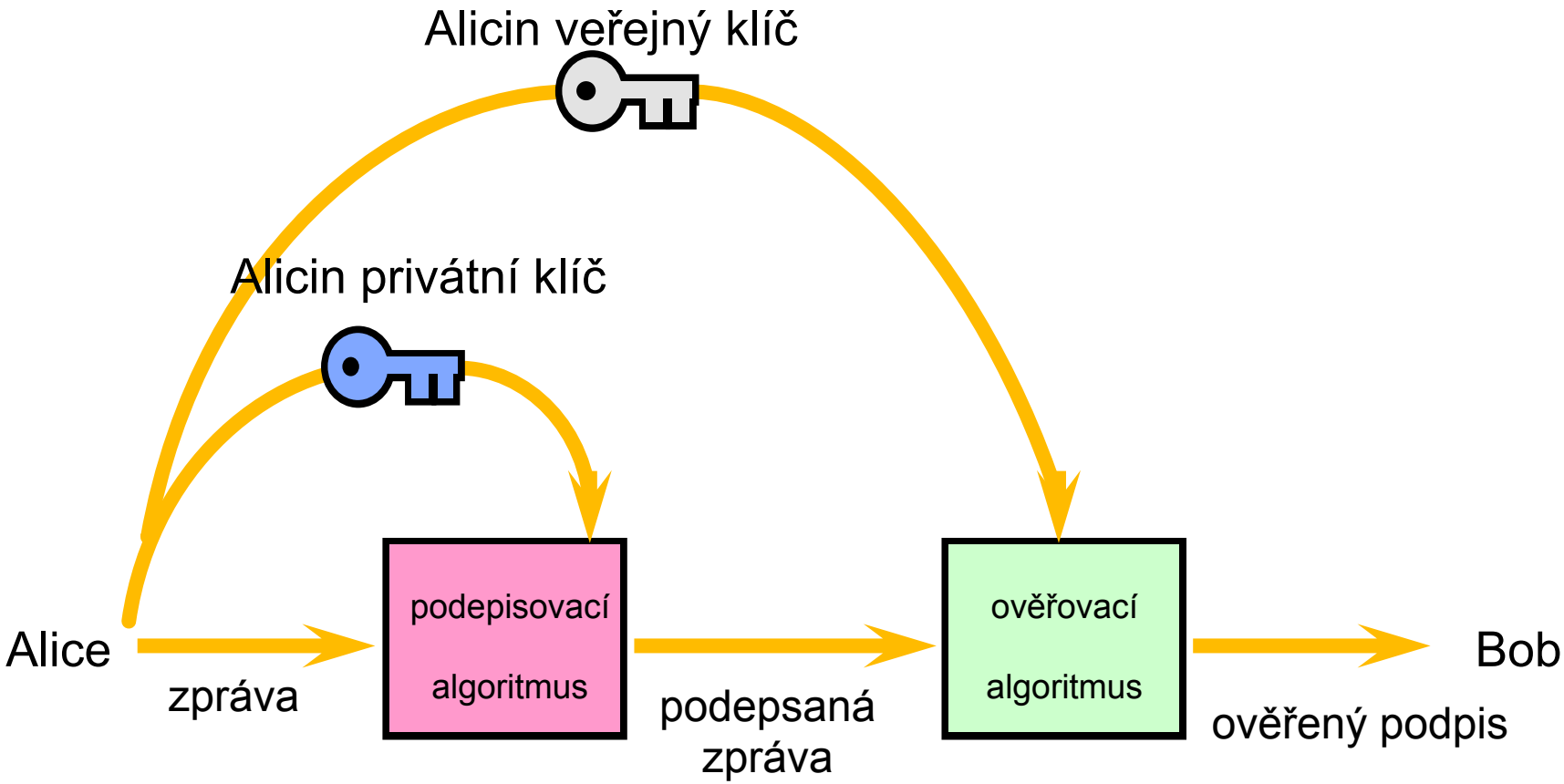
PV157 – Autentizace a řízení přístupu

Autentizace dat a zpráv (pokračování)

Schéma digitálního podpisu



Převzato z: *Network and Internetwork Security* (Stallings)

Digitální podpis – klíče

- Každý subjekt má 2 klíče:
  - privátní (soukromý) klíč pro vytváření podpisu;
  - veřejný klíč pro ověření podpisu.
- Správný digitální podpis může vytvořit jen ten, kdo má k dispozici soukromý klíč.
- Pro ověření podpisu je nutné mít veřejný klíč podepsaného subjektu.
- Digitální podpis nedává sám o sobě žádnou záruku o době jeho vytvoření.

Digitální podpis – co podepisujeme?

- Algoritmy digitálního podpisu založené na asymetrické kryptografii jsou relativně pomalé.
- V praxi nepodepisujeme celý dokument (velikosti v kB, MB i GB) ale pouze haš (fixní délky řádově stovky bitů).
- Není bezpečné rozdělit dokument na několik částí a ty podepsat separátně.
- Při kombinaci šifrování veřejným klíčem a podpisu je nutné dokument nejprve podepsat a pak teprve zašifrovat.

Digitální podpis – bezpečnost

- Pro bezpečnou funkčnost digitálního podpisu je nezbytně nutné:
  - udržovat privátní klíč v tajnosti – každý, kdo má přístup k privátnímu klíči má možnost vytvářet platné podpisy;
  - zajistit integritu veřejného klíče – pokud bychom ověřovali platnost podpisu pomocí nesprávného klíče můžeme dojít k nesprávným závěrům.

Digitální podpis – integrita VK

- Jak zajistit integritu veřejného klíče?
- Pomocí certifikátu veřejného klíče!
- Certifikát váže veřejný klíč k subjektu (spíše k nějakému identifikátoru subjektu).
- Tato vazba je digitálně podepsána důvěryhodnou třetí stranou (certifikační autoritou).
- Jak zajistit integritu veřejného klíče certifikační autority?

Digitální podpis – jak vypadá certifikát?

- Certifikát standardu X.509 verze 3

```
Certificate  ::=  SEQUENCE  {
     tbsCertificate       TBSCertificate,
     signatureAlgorithm   AlgorithmIdentifier,
     signature            BIT STRING  }

TBSCertificate  ::=  SEQUENCE  {
     version         [0]  Version DEFAULT v1,
     serialNumber         CertificateSerialNumber,
     signature            AlgorithmIdentifier,
     issuer               Name,
     validity             Validity,   -- notBefore, notAfter
     subject              Name,
     subjectPublicKeyInfo SubjectPublicKeyInfo,  -- algID, bits
     issuerUniqueID  [1]  IMPLICIT UniqueIdentifier OPTIONAL,
     subjectUniqueID [2]  IMPLICIT UniqueIdentifier OPTIONAL,
     extensions      [3]  Extensions OPTIONAL
         -- sequence of: extnID, crit, value }
```

Digitální podpis

- Jak udržovat důvěrnost privátního klíče?
- Key recovery – obnova klíčů!?
- Některé algoritmy asymetrické kryptografie umožňují používat jednu dvojici klíčů pro šifrování i podpis – toto však není příliš vhodná kombinace.
- Jak se brání zaměstnavatel vůči odchodu zaměstnance? $\rightarrow$ Rozdílný přístup k šifrovacímu a podepisovacímu klíči!

Digitální podpis – algoritmy

- První algoritmy asymetrické kryptografie na začátku 70. let 20. století:
  - Britská tajná služba GCHQ (Clifford Cocks).
  - Veřejné oznámení až koncem 20. století (1997).
  - Aplikaci algoritmů pro autentizaci – podpis – „objevila“ později až akademická komunita u svých veřejných algoritmů.
- První veřejné algoritmy koncem 70. let 20. století (W. Diffie a M. Hellman ovlivněni prací R. Mercla).
- Nejznámější algoritmus RSA (Rivest, Shamir, Adelman) publikován v roce 1977, patentován v roce 1983 (v současné době patent již vypršel).

Digital Signature Algorithm

- V roce 1994 proběhl v USA výběr Digital Signature Standard (DSS) – vyhrál DSA (Digital Signature Algorithm) modifikovaný algoritmus ElGamal, založený na diskrétním logaritmu v Z_p.
- Další algoritmy, mj. založeny i na eliptických křivkách.

Digitální podpis – délky klíčů

- Algoritmus RSA
  - Při jednom z popisů algoritmu (ve „Scientific American“ v roce 1977) autoři publikovali příklad kryptosystému (prvočísla byla 64- a 65bitová), o kterém věřili, že je bezpečný.
  - Tento kryptosystém byl rozlomen v roce 1994.
  - Koncem roku 1999 došlo k prolomení 512bitového modulo RSA (několik set rychlých počítačů pracovalo přes 4 měsíce).
  - V současné době se používá modulo o délkách 1024 nebo 2048 bitů.

Digitální podpis – časová náročnost

- Algoritmy asymetrické kryptografie jsou relativně časově náročné – příklady časů pro čipovou kartu

Operace	Modulus	Exponent	Čas
Podpis RSA	1024 bitů	1024 bitů	25,2 ms
Podpis RSA	2048 bitů	2048 bitů	0,17 s
Ověření RSA	1024 bitů	32 bitů	2,8 ms
Ověření RSA	2048 bitů		38 ms
Generování klíče RSA	1024 bitů		1,56 s
Generování klíče RSA	2048 bitů		14,4 s
Podpis EC DSA (GF(p))	160 bitů		24 ms
Ověření EC DSA (GF(p))	160 bitů	160 bitů	50 ms

Digitální podpis – RSA– matematika

- Násobení prvočísel snadné, ale faktorizace čísel výpočetně náročná.
- Velká prvočísla **p**, **q**, **n** = **p·q**,
  φ(n) = (p-1)(q-1).
- Zvolíme velké **d** takové, že **gcd(d, φ(n)) = 1.**
- Spočítáme $\mathbf{e = d^{-1} \pmod{\varphi(n)}}$.
- Veřejný klíč: **n, e.**
  Neveřejné parametry: **p, q, d.**
- Šifrování: $\mathbf{c = w^e \bmod n}$.
- Dešifrování: $\mathbf{w = c^d \bmod n}$.

Výpočetní bezpečnost

- Bezpečnost RSA je založena na nesnadnosti faktorizace čísel.
- Je zřejmé, že pouhým „vyzkoušením“ všech čísel do odmocniny z **n** se nám podaří **n** faktorizovat.
- Bezpečnost RSA je založena na tom, že faktorizovat velká čísla (tím v současné době myslíme čísla o tisících bitů) v rozumném čase neumíme.
- Pokrok v oblasti faktorizace čísel (například nalezení nového algoritmu) však může znamenat, že z veřejného klíče budeme schopni odvodit klíč privátní.
- Tento algoritmus je založen na tzv. „výpočetní bezpečnosti“ (nejen tento algoritmus, „výpočetní bezpečnost“ je běžně používaný přístup).

Digitální podpis – RSA– příklad

- Čísla jsou úmyslně příliš malá (takový systém není bezpečný).
- Parametry: p = 7927, q = 6997, n = pg = 55465219, $\varphi(n) = 7926 \times 6996 = 55450296$.
- Alice volí e = 5, řeší rovnici ed = 5d = 1 (mod 55450296), d = 44360237.
- Alicin veřejný klíč: (n = 55465219, e = 5), privátní klíč: d = 44360237.
- Podpis: m = 31229978; $s = 31229978^{44360237} \bmod 55465219 = 30729435$.
- Ověření podpisu: $m = 30729435^5 \bmod 55465219 = 31229978$.
- Podpis je ověřen, pokud je přijata zpráva m=31229978.

Hašovací funkce

- **Kryptografická** hašovací funkce.
- Vstup libovolné délky.
- Výstup pevné délky: n bitů (často 128 nebo 192).
- Funkce není prostá (vznikají kolize).
- Haš slouží jako kompaktní reprezentace vstupu (nazýváme též otisk, anglicky imprint, digital fingerprint nebo message digest).
- Hašovací funkce často používáme při zajišťování integrity dat. Spočítáme nejprve haš a pak pracujeme s tímto hašem (například jej podepíšeme).

Vlastnosti hašovacích funkcí

- Jednosměrnost
  - Pro libovolné x je snadné spočítat h(x).
  - V rozumném čase nejsme schopni pro pevně dané **y** najít takové **x**, že h(x) = y.
- Bezkoliznost
  - (slabá): pro dané x nejsme schopni v rozumném čase najít x’ (x ≠ x’) takové, že h(x) = h(x’).
  - (silná): v rozumném čase nejsme schopni najít libovolná x, x’ taková, že h(x) = h(x’).

Příklad hašovací funkce

- Uvažujme následující hašovací funkci:
  - Jednoduchý součet bajtů modulo 256 .
  - Fixní osmibitový výstup.
  - Pro text „ahoj“ získáme 97+104+111+106 mod 256 = 162.
- Tuto funkci je sice jednoduché spočítat, není to však dobrá kryptografická hašovací funkce, neboť nemá vlastnost bezkoliznosti.
- h(„ahoj“) = h(„QQ“) = h(„zdarFF“)

Běžné kryptografické hašovací funkce

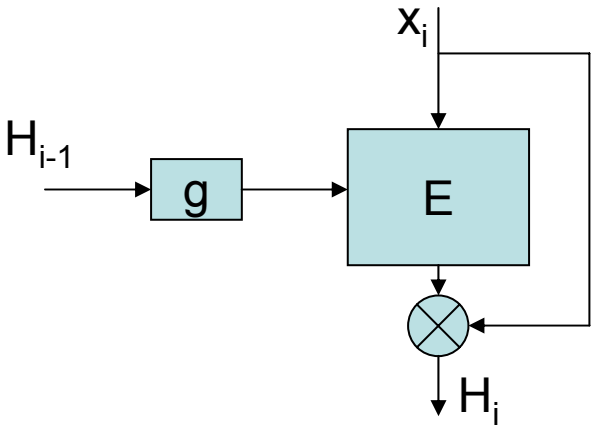
- MD4: výstup 128 bitů – dnes se již nepoužívá – byly nalezeny slabiny v algoritmu (umožňující nalezení kolizí, snižující efektivní výstup asi na 20 bitů).
- MD5: výstup 128 bitů – dnes ještě používána, ačkoliv byly nalezeny významné slabiny a algoritmus pro nalezení kolizí – 128 bitů se již nepovažuje za dostatečně bezpečnou délku!
- SHA-1 (Secure Hash Algorithm): výstup 160 bitů – NIST standard, používána v DSS (Digital Signature Standard), považována za bezpečnou pro nejbližší roky.
- **„SHA-2“ – SHA-256, SHA-384, SHA-512 (a dodána SHA-224) – doporučuje se používat především tyto funkce! Definovány ve standardu (NIST) FIPS 180-2**.

Hašovací funkce – příklady

- MD5
  - Vstup: „Autentizace“.
  - Výstup: 2445b187f42245830378885l1d5411c7 .
  - Výstupem je 128 bitů, zapisujeme hexadecimálně.
  - Vstup: „Cutentizace“.
  - Výstup: cd99abbba3306584e90270bf015b36a7.
  - Změna jednoho bitu vstupu → velká změna výstupu.
- SHA-1
  - Vstup: „Autentizace“.
  - Výstup: dfcee447d609529f0335e67016557c281fc6eb44 .

Hašovací funkce z šifrovacích algoritmů

- Hašovací funkci můžeme vytvořit z libovolné blokové symetrické šifry.
- Existují postupy pro vytvoření hašovací funkce s délkou haše o velikosti délky bloku či dvojnásobku délky bloku.



Matyas-Meyer-Oseas

Narozeninový paradox

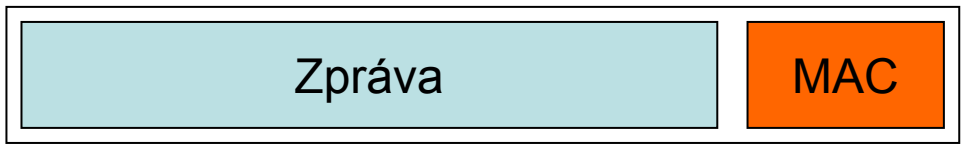
- Zajímavé útoky na hašovací funkce vycházejí z tzv. birthday (narozeninového) paradoxu.
- Pravděpodobnost, že 2 lidé v sále s 23 lidmi mají narozeniny ve stejný den je více než 50 %. Tato pravděpodobnost je překvapivě vysoká a s počtem lidí v sále dále rychle roste (při 30 lidech v sále je vyšší než 70 %).
- Tohoto můžeme využít při hledání kolizí hašovací funkce (tj. pokud nám jde o LIBOVOLNOU kolizi máme větší šanci, než při kolizi se specifickou hodnotou haše).

Integrita dat – CRC

- Detekce chyb (error detection)
  - Kontrolní součty – paritní bity, aritmetický součet modulo, exkluzivní součet (XOR).
  - CRC (cyclic redundancy check) – cyklický kontrolní součet. Např. 16-bitový $g(x) = 1+x^2+x^{15}+x^{16}$.
  - Slouží k detekci (neúmyslných) chyb při přenosu dat, uložení dat apod.
  - Je snadné spočítat kontrolní součet dat i vytvořit data odpovídající určitému kontrolnímu součtu, proto kontrolní součty neposkytují ochranu před úmyslnou modifikací dat.

Integrita dat – MAC

- Autentizační kódy (též digitální pečetě, message authentication code – MAC)
  - Slouží k zajištění integrity dat (ne důvěrnosti dat).
  - Původce a příjemce dat sdílí tajný klíč **k.**
  - Původce zprávy spočítá $h_k(x)$, které přidá ke zprávě x.
  - Příjemce zprávy spočítá $h_k(x')$ z přijaté zprávy x' a srovná s přijatým $h_k(x)$.



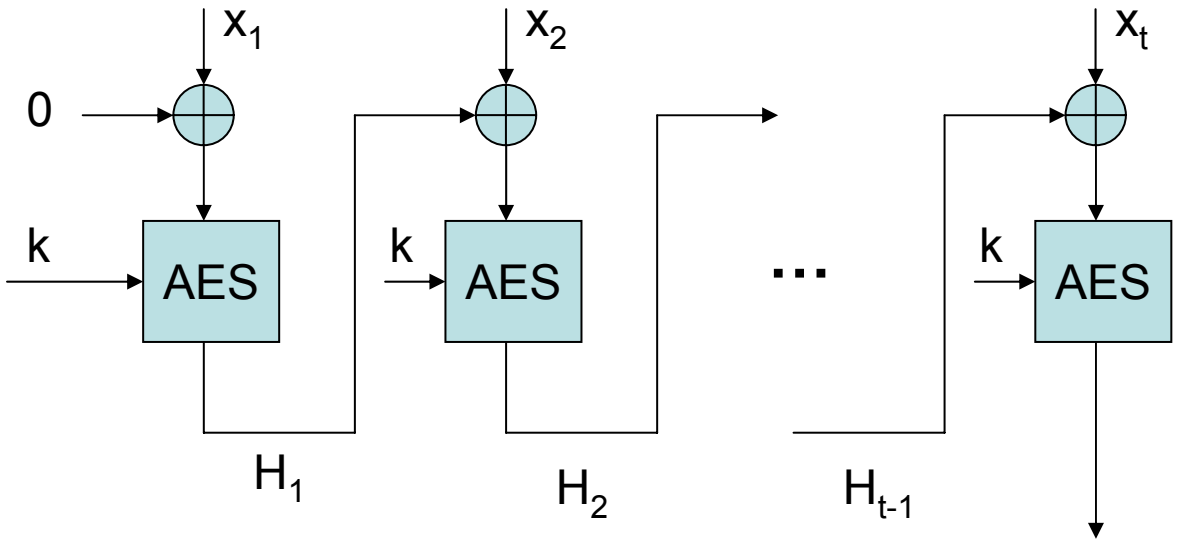
Jak získat funkci pro vytváření MAC?

- MAC můžeme získat z libovolné symetrické šifry v CBC (cipher block chaining) režimu (kdy zašifrovaná data závisí na všech předchozích datech).
- MAC můžeme získat i z hašovací funkce, např. tajný prefix, sufix.

  $HMAC(x) = h(\mathbf{k}||\mathbf{p}_1||h(\mathbf{k}||\mathbf{p}_2||\mathbf{x}))$
  k klíč, **p** padding (doplnění dat)

MAC

- Příklad: MAC založený na AES



„Chaffing and windowing“

- Metoda „chaffing and windowing“ (oddělení zrn od plev).
- Použití MAC pro zajištění důvěrnosti.
- Zprávu rozdělíme na jednotlivé bity.
- Pro každý bit vytvoříme 2 zprávy (obsahující 0 a 1), jednu se správným MAC a jednu s nesprávným MAC. Tyto 2 zprávy v náhodném pořadí odešleme příjemci.
- Příjemce dostane 2 zprávy, tu se správným MAC si ponechá a tu druhou zahodí.
- Nikdo, kdo nezná tajný sdílený klíč, není schopen odlišit zprávu se správným a nesprávným MAC.
- Existují i efektivnější metody než posílání po 1 bitu.

Praktická ukázka autentizace dat

- Autentizace spustitelných EXE souborů v prostředí Microsoft Windows
- Proč autentizujeme?
  - Chceme zajistit integritu programů.
  - Chceme znát autora programu.
  - Věříme například jen kódu od firmy Microsoft a chceme mít jistotu, že kód při přenosu někdo nemodifikoval, či nepodstrčil.

Microsoft Autenticode

- Jak autentizace funguje?
- EXE soubor je digitálně podepsaný.
- Digitální podpis je ověřen.
- Pokud je podpis platný, aplikace je automaticky spuštěna.
- Je-li podpis neplatný nebo není-li EXE soubor podepsán, je interaktivně dotázán uživatel.

Microsoft Autenticode

- Toto dialogové okno asi už znáte:

Microsoft Autenticode

- A tento program není podepsaný vůbec:

Microsoft Authenticode

- Autentizovaná data
  - Při bezchybné autentizaci dat víme, že data byla získána z uvedeného zdroje (např. od firmy Microsoft) a že nebyla modifikována.
  - Autentizace dat neznamená, že data jsou správná, programy bezchybné, bezpečné apod.

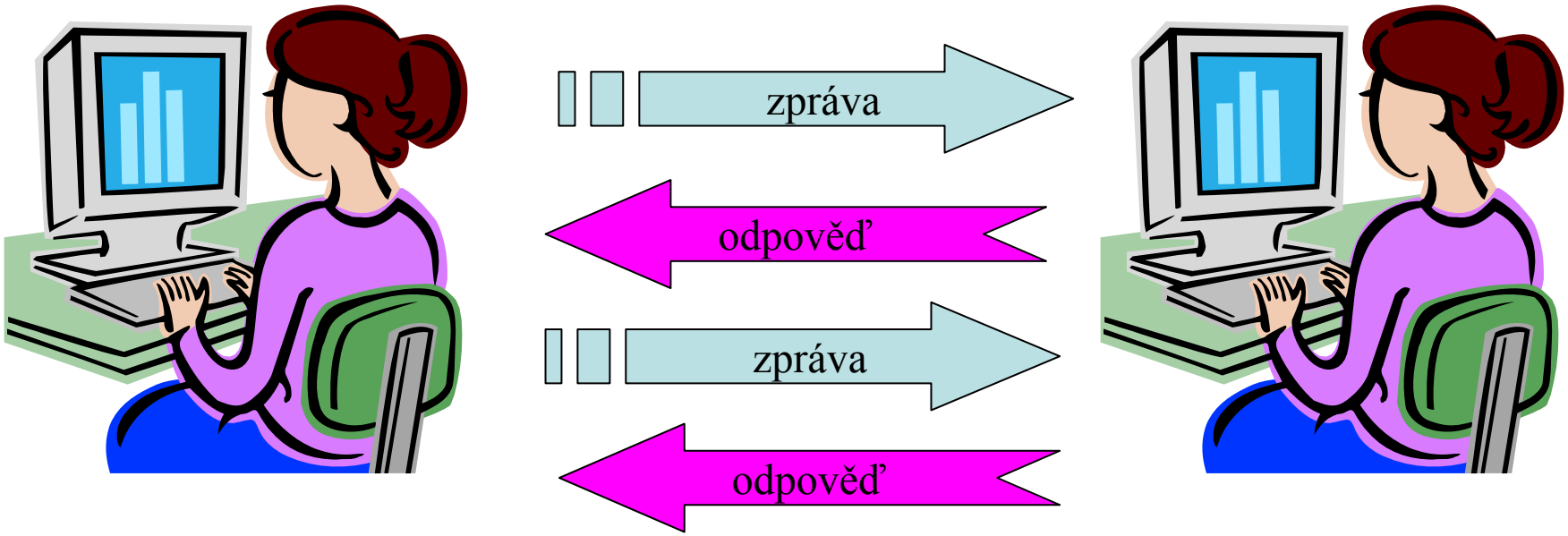
Microsoft Autenticode

- 100% bezpečnost neexistuje!
- V roce 2001 získal neznámý útočník 2 certifikáty veřejného klíče určené pro firmu Microsoft a podepsané firmou Verisign (obě firmy jsou klíčoví hráči ve svém oboru a určitě mají pečlivě propracované bezpečnostní předpisy).
- Útočník se úspěšně vydával za zaměstnance firmy Microsoft a získal certifikát podepsaný firmou Verisign včetně soukromého klíče.
- Jakýkoliv kód podepsaný takto certifikovaným klíčem bude ve Windows spuštěn bez úvodního varování.

Autentizační protokoly

Protokol

- Protokol je několikastranný algoritmus definovaný posloupností kroků, které specifikují akce prováděné dvěma a více stranami, pro dosažení určitého cíle



Kryptografické protokoly

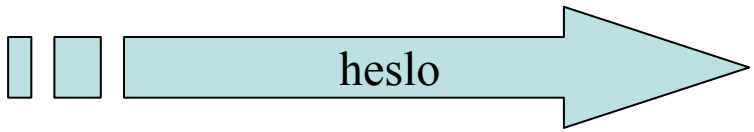
- **Autentizační protokol** – zajistí jedné straně určitou míru jistoty o identitě jiné strany (té, se kterou komunikuje)
- **Protokol pro ustavení klíče** (key establishment protocol) – ustaví sdílené tajemství (typicky klíč)
- **Autentizovaný protokol pro ustavení klíče** (authenticated key establishment protocol) – ustaví sdílené tajemství se stranou, jejíž identita byla potvrzena

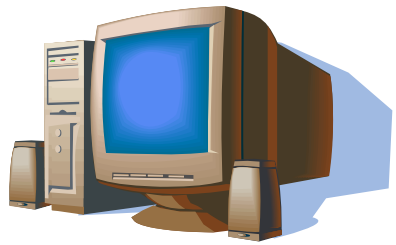
Autentizační protokoly

- Během protokolu autentizujeme:
  - Pouze jednu ze stran
  - Obě strany
  - Kontinuální autentizace
- Kdo koho autentizuje
  - Alice vyzývá Boba, aby se autentizoval
  - Bob se autentizuje rovnou sám bez výzvy

Autentizace heslem

- Alice se autentizuje Bobovi tak, že mu pošle své heslo
- Heslo je možné odposlechnout
- Bob po úspěšné Alicině autentizaci zná Alicino heslo a může se (např. vůči Cyrilovi) autentizovat jako Alice (pokud Alice používá stejné heslo pro autentizace vůči různým strana  ám)



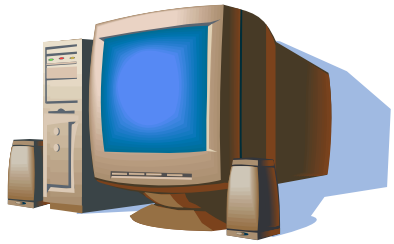


Útok impersonací (vydáváním se za jiného)

Alice

Jsem Alice

heslo



Bob

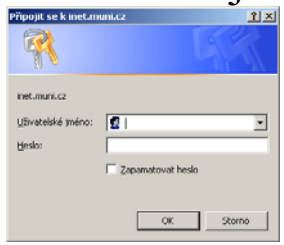
odposlech

Emil

Emil je pasivní útočník (odposlouchává komunikaci)

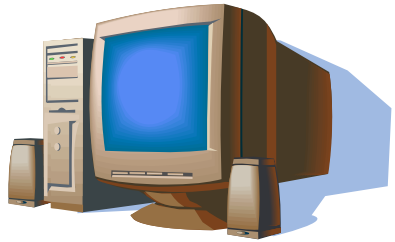
Emil zadá heslo do existující aplikace

Emil



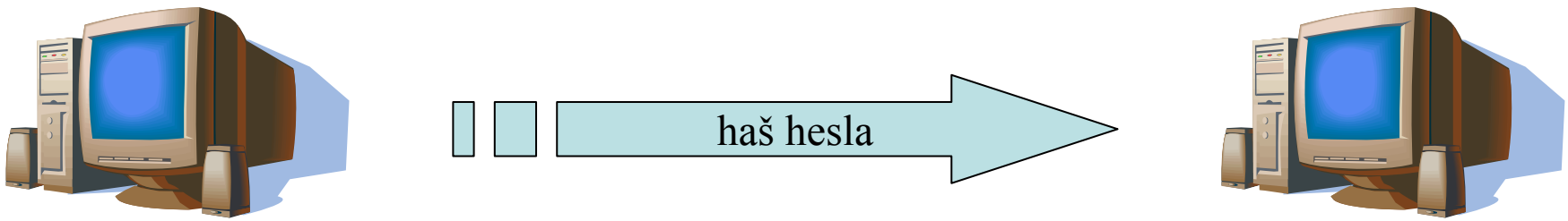
Jsem Alice

heslo



Hašované heslo

- Při autentizaci se neposílá heslo samotné, ale pouze haš hesla
- Kdo odposlechne haš nezíská automaticky heslo
- Haš však lze použít pro podvodnou autentizaci

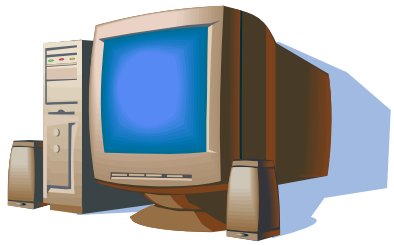


Útok přehráním

Alice

Jsem Alice

haš hesla



Bob

odposlech

Emil

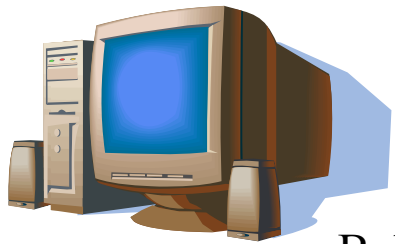
Emil je pasivní útočník (odposlouchává komunikaci)

Emil nezná heslo, ale pošle odposlechnutý haš hesla, pomocí své pomocné aplikace

Emil

Jsem Alice

haš hesla



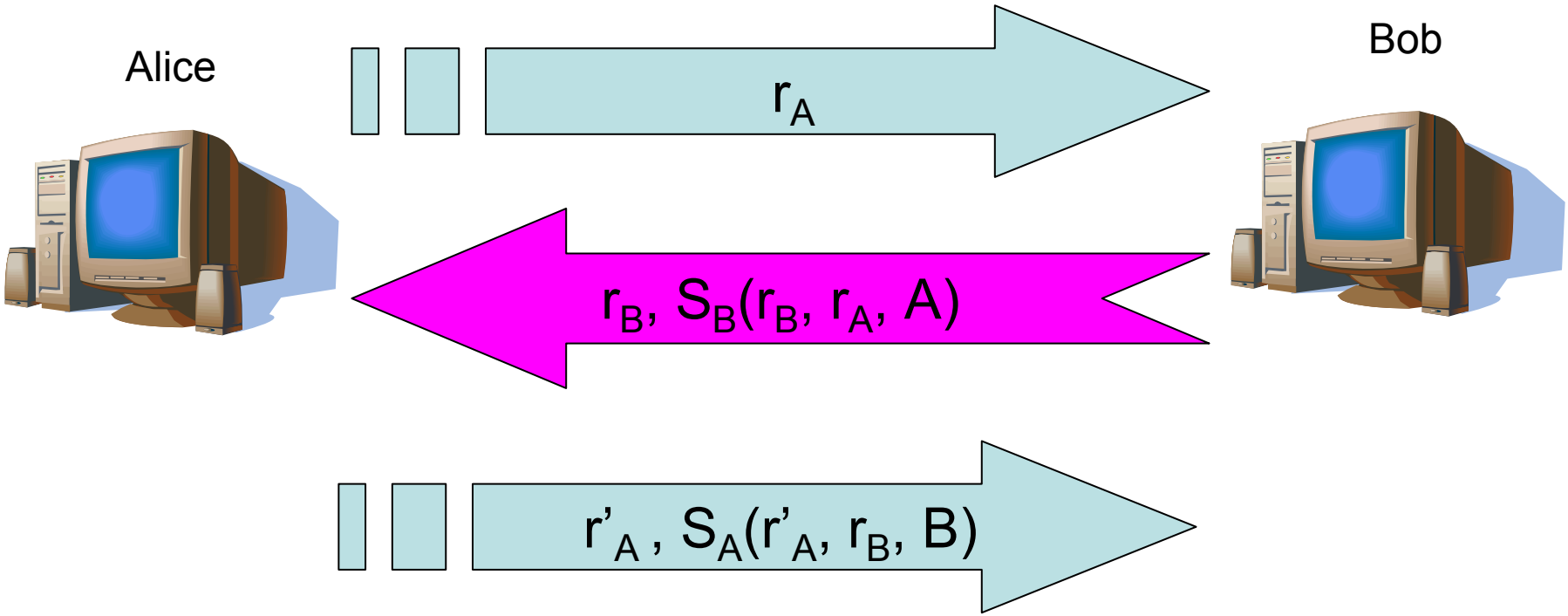
Bob

Další útoky na protokoly

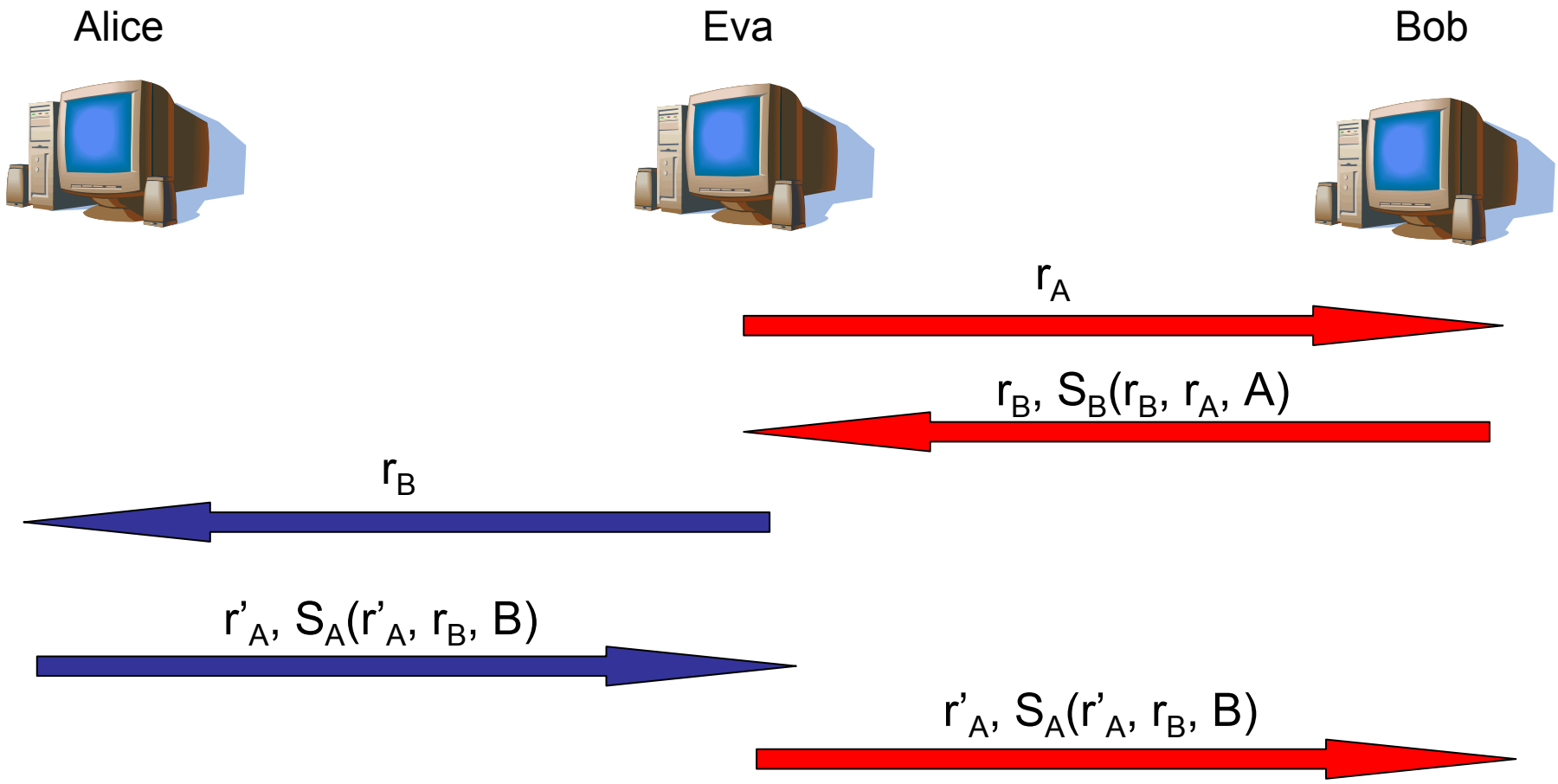
- Zmíněné útoky impersonací a přehráním
- Úplný výčet je nesnadný, ale zmínit je třeba
  - útoky **prolínáním** (interleaving) – kombinujeme zprávy z více průběhů – obvykle, ale ne nutně jen, stejného protokolu – ať již ukončených, nebo právě probíhajících (viz další slajd)
  - **slovníkové** útoky – na protokoly využívající hesla, diskutováno později u autentizace uživatelů
  - útoky **využitím známého klíče** (known-key) – obvyklé u protokolů pro ustanovení klíče, kde se klíč ustanoví na základě staršího/ch (útočníkovi známého/ch) klíče/ů

Útok prolínáním (1)

- Mějme autentizační protokol:



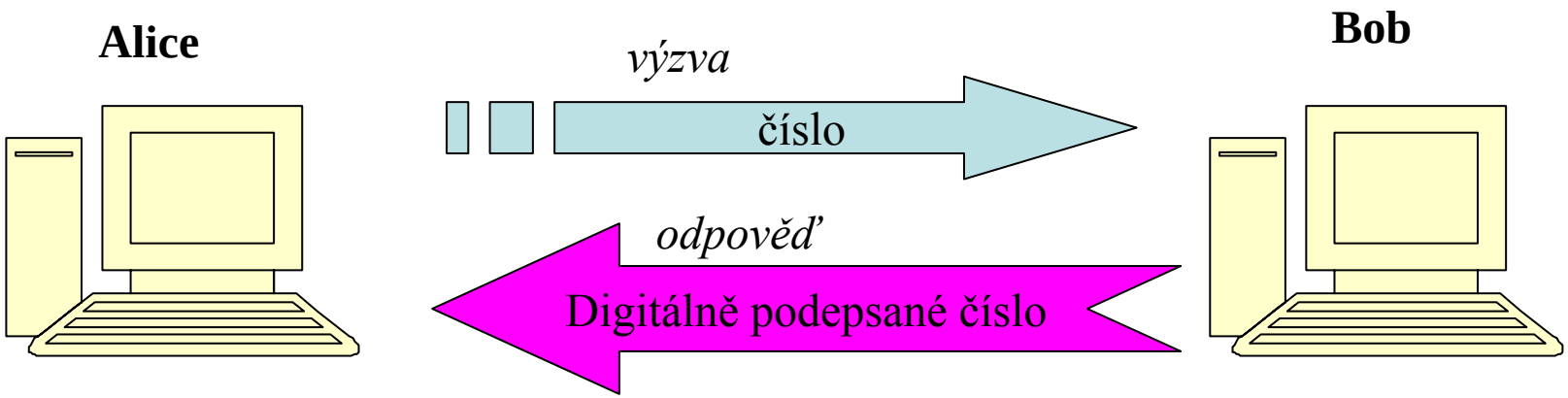
Útok prolínáním (2)



Evě se podařilo vydávat se vůči Bobovi za Alici

Protokoly výzva-odpověď

- Protokoly typu výzva-odpověď (challenge-response)
  - Odposlechem výzvy i odpovědi útočník moc nezíská
  - Bob se může přesvědčit o identitě Alice, bez získání jejího tajemství



Časově proměnné parametry

- **Náhodná čísla** (random numbers) – čísla, která jsou nepredikovatelná (v tomto kontextu zahrnujeme pod náhodná čísla i čísla pseudonáhodná). Použitím náhodných čísel zajišťujeme jedinečnost a „aktuálnost/čerstvost“. Získat skutečně náhodná čísla je netriviální (vyžaduje speciální HW zařízení). V praxi obvykle používáme pseudonáhodná čísla (které na základě tajného stavu - semínka (seed) generují sekvence čísel). Značíme **r**.
- **Sekvence** (sequence numbers) – monotonně rostoucí posloupnost čísel (obě strany musí dlouhodobě uchovávat informaci o poslední hodnotě). Jednoznačně identifikují zprávy a umožňují detekovat útoky přehráním předchozí komunikace. Značíme **n**.
- **Časová razítka** (timestamps) – obě strany musí synchronizovat a zabezpečit hodiny. Zajišťují jedinečnost a časovou přesnost. Značíme **t**.

Protokoly výzva-odpověď

- Založené na symetrických technikách
  - Symetrické šifrování
  - Jednosměrná funkce s klíčem
  - Generátory passcode
- Založené na asymetrických technikách
  - Dešifrování
  - Digitální podpis

Symetrické techniky

- Založené na symetrickém šifrování (Alice a Bob sdílí tajný symetrický klíč **K**)
- Standard ISO/IEC 9798-2
- Jednostranná autentizace (časové razítko)
  - $A \rightarrow B$: $E_K(t_A,\text{“B“})$
- Možné útoky
  - Útok přehráním: odposlechnu $E_K(t_A,\text{“B“})$ a pošlu jej rychle znovu (v době platnosti t_A)
  - Změna hodin: odposlechnu $E_K(t_A,\text{“B“})$, později změním hodiny B tak, aby odpovídaly času t_A a znovu pošlu $E_K(t_A,\text{“B“})$

Symetrické techniky

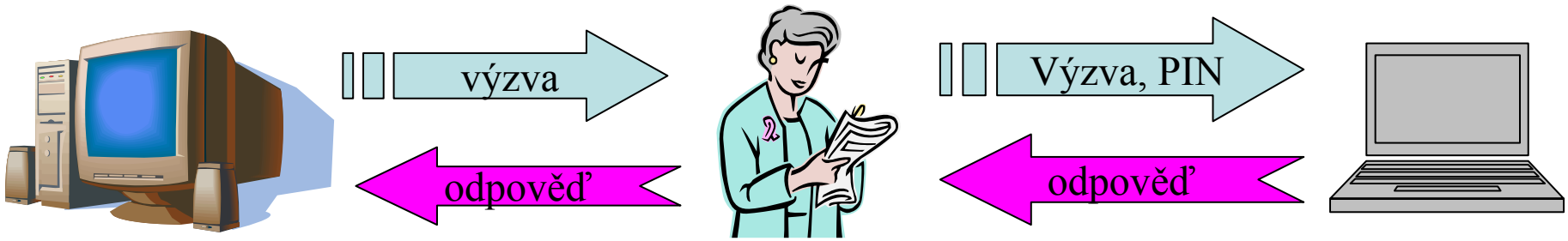
- Jednostranná autentizace (náhodné číslo)
  - $A \leftarrow B: r_B$
  - $A \rightarrow B: E_K(r_B, \text{“B“})$
- Možné útoky
  - Útočník odposlouchává a ukládá $[r_B, E_K(r_B, \text{“B“})]$, pokud se challenge r_B opakuje, pak je schopen poslat správný response. Případně se může aktivně snažit ovlivnit vytváření náhodných r_B (např. ovlivněním vstupu generátoru náhodných čísel Boba).
- Oboustranná autentizace (náhodná čísla)
  - $A \leftarrow B: r_B$
  - $A \rightarrow B: E_K(r_A, r_B, \text{“B“})$
  - $A \leftarrow B: E_K(r_B, r_A)$

Symetrické techniky

- Založené na klíčovaných jednosměrných funkcích (Alice a Bob sdílí tajný symetrický klíč **K**)
- Standard ISO/IEC 9798-4, protokoly SKID
- Oboustranná autentizace
  - $A \leftarrow B: r_B$
  - $A \rightarrow B: r_A, h_K(r_A, r_B, \text{"B"})$
  - $A \leftarrow B: h_K(r_B, r_A, \text{"A"})$
  - h_K je MAC algoritmus

Symetrické techniky

- Generátory passcode – hand-held (PDA, kapesní počítače) pro bezpečné uložení dlouhodobých klíčů doplněné zadáním PINu uživatele
- Subjekty A, B sdílí tajný klíč s_A a tajný PIN p_A
  - $A \leftarrow B: r_B$
  - subjekt A zadá do generátoru přijatou výzvu r_B a vloží svůj PIN
  - $A \rightarrow B: f(r_B, s_A, p_A)$



Asymetrické techniky

- Založené na dešifrování soukromým klíčem
- Jednostranná autentizace
  - A ← B: h(r), „B“, P_A(r,”B”)
  - A → B: r
- h – hašovací funkce
- h(r) slouží k prokázání znalosti r bez jeho odhalení

Asymetrické techniky

- Založené na digitálním podpisu
- Standard ISO/IEC 9798-3
- Jednostranná autentizace (časové razítko)
  - A → B: $cert_A$, t_A, ”B“, $S_A(t_A, \text{“B“})$
- Možné útoky
  - Útok přehráním: odposlechnu $S_A(t_A, \text{“B“})$ a pošlu jej rychle znovu (v době platnosti t_A)
  - Změna hodin: odposlechnu $S_A(t_A, \text{“B“})$, později změním hodiny B tak, aby odpovídaly času t_A a znovu pošlu $S_A(t_A, \text{“B“})$

Asymetrické techniky

- Jednostranná autentizace (náhodné číslo)
  - A ← B: r_B
  - A → B: $cert_A$, r_A, "B", $S_A(r_A,r_B,\text{"B"})$
  - r_A zde zabraňuje útokům s vybraným textem
- Možné útoky
  - Obdobné útoky na náhodné r_B jako v případě symetrických technik
- Oboustranná autentizace (náhodná čísla)
  - A ← B: r_B
  - A → B: $cert_A$, r_A, "B", $S_A(r_A,r_B,\text{"B"})$
  - A ← B: $cert_B$, "A", $S_B(r_B,r_A,\text{"A"})$

Protokoly pro mgmt klíčů

- Účel
  - Přenos klíče
  - Ustavení klíče
  - Aktualizace klíče (strany sdílí dlouhodobý klíč **K**)
  - Zároveň i autentizace jedné nebo obou stran
- Počet stran
  - Protokol pro dvě strany
  - Protokol s důvěryhodnou třetí stranou

Symetrické techniky přenosu klíče

- Aktualizace klíče založená na symetrické šifře (Alice a Bob sdílí tajný klíč K)
  - Přenos klíče (1 zpráva, časové razítko)
  - $A \rightarrow B: E_K(r_A, t_A, \text{“B“})$
  - Přenos klíče (výzva-odpověď, náhodné nebo sekvenční číslo)
  - $A \leftarrow B: n_B$
  - $A \rightarrow B: E_K(r_A, n_B, \text{“B“})$

Symetrické techniky přenosu klíče

- Přenos klíče odvozením
  - $A \rightarrow B: r_A$
  - Nový klíč $W=E_K(r_A)$
- Aktualizace klíče se vzájemnou autentizací
  - AKEP2 (Authenticated Key Exchange Protocol 2)
  - $A \rightarrow B: r_A$
  - $A \leftarrow B: (\text{"B"},\text{"A"},r_A,r_B), h_K(\text{"B"},\text{"A"},r_A,r_B)$
  - $A \rightarrow B: (\text{"A"},r_B), h_K(\text{"A"},r_B)$
  - Nový klíč $W=h'_{K'}(r_B)$
  - h_K je MAC algoritmus, h' je MAC algoritmus (odlišný od h), obě strany sdílí K, z K je odvozen K'

Symetrické techniky přenosu klíče

- Přenos klíče bez předchozího sdíleného tajemství
  - Shamirův protokol bez klíčů (Shamir's no-key protocol)
  - Komutativní symetrická šifra E
  - Každá strana má svůj symetrický klíč K_A,K_B
  - $A \rightarrow B: E_{K_A}(X)$
  - $A \leftarrow B: E_{K_B}(E_{K_A}(X))$
  - $A \rightarrow B: E_{K_B}(X)$
  - Nyní obě strany sdílí X; byly nutné 3 zprávy

Otázky?

Vítány!!!

Příští přednáška 6. 3. 2006 ve 14:00

matyas@fi.muni.cz
zriha@fi.muni.cz