Bezpečnost v informačních technologiích (KIV/BIT)

7-8. Hashovací funkce, integrita dat, elektronický (digitální) podpis

Ing. Pavel Král, Ph.D.

Katedra informatiky a výpočetní techniky
Západočeská Univerzita

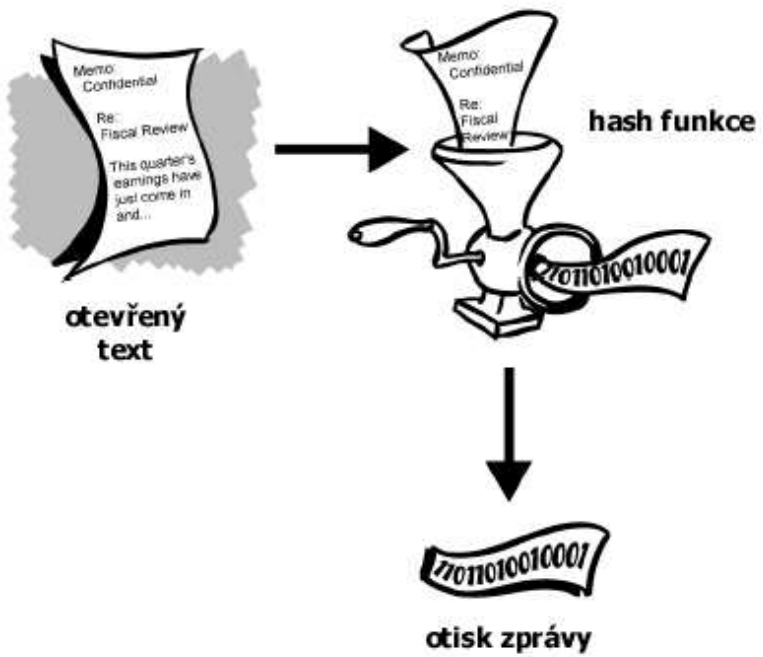
8. dubna 2015

Obsah

Kontrolní součet (Hash) (připomenutí)

=jednocestná funkce, která z libovolně dlouhého textu vyrobí krátký řetězec konst. délky

- Př: 16B (MD5), 20B (SHA-1)
- použití: otisk prstu dat (fingerprint), bezpečné ukládání hesel (linux - MD5), el. podpis, atd.
- naprosto stejné dva dokumenty → shodný hash (otisk)



Motivace
Proč používat hashovací funkce?

- "rychlost" asymetrických šifer
- závislost bezpečnosti na volbě klíče a obsahu zprávy (ovlivnitelnost člověkem)

$\longrightarrow$

- výhodnost použití hash funkce H
- místo $D_{SK_A}(P)$ poslat $P, D_{SK_A}(H(P))$ (pro autentizaci A)

Vlastnosti

základní:

1. $P \rightarrow$ jednoduchost výpočtu $H(P)$
2. $H(P) \rightarrow$ nemožnost nalezení P
3. platí: $\forall P_1, \forall P_2$; pokud $H(P_1) = H(P_2) \Rightarrow P_1 = P_2$

další:

- libovolné množství vstupních dat (délka zprávy) $\rightarrow$ otisk (hash) **konstantní** délky
- drobná změna vstupních dat $\rightarrow$ **velká** změnu otisku (patrné na první pohled)

Princip hash funkce

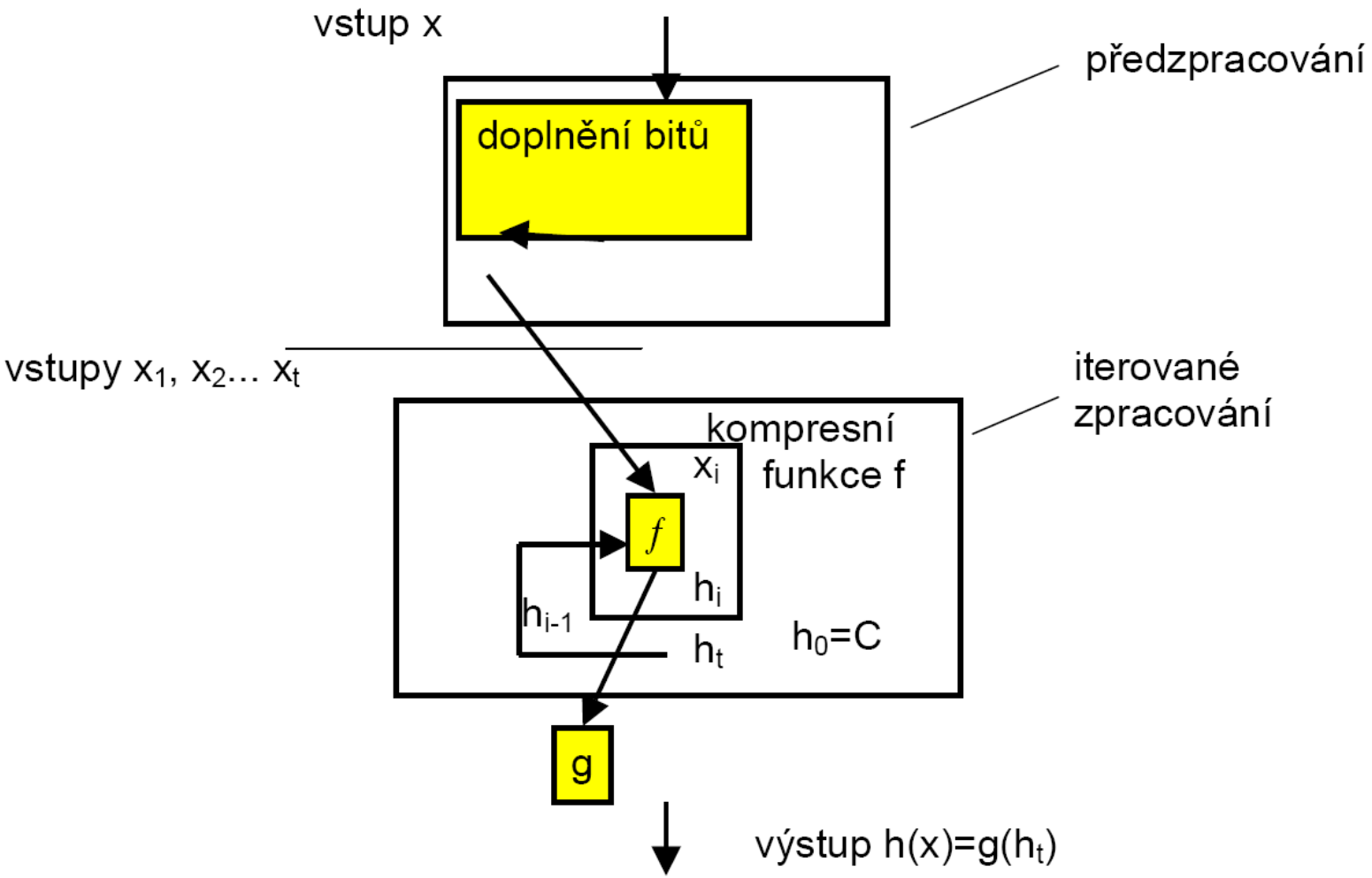
- $h_0 = C$
- $h_i = f(x_i, h_{i-1}) \quad i = 1, .., t$
- $h(x) = g(h_t)$

- doplnění zprávy: $l = t \times b$
  - l – délka zprávy
  - t – počet bloků x_i
  - b – délka bloku
- C – inicializační konst.

- f – kompresní funkce
  - pevná délka vstupu x_i
  - shodné zpracování bloků
  - činnost v iteracích
- g – výstupní zobrazení (většinou identické)

Poznámka: platí pro většinu hash fcí

Schéma (kompresní) hash funkce



Základní rozdělení hash funkcí

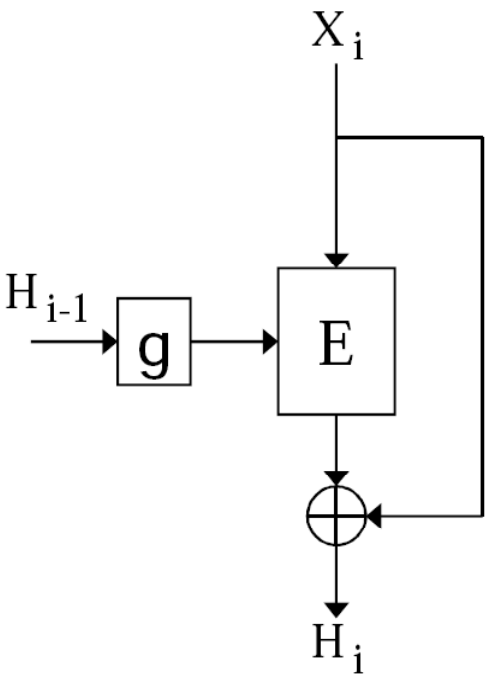
- bez klíče
  - MDC (Modification Detection Codes) - kódy pro detekci manipulací → zajištění integrity dat
  - Ostatní
- s klíčem
  - MAC (Message Authentication Codes)
  - Ostatní

Použití

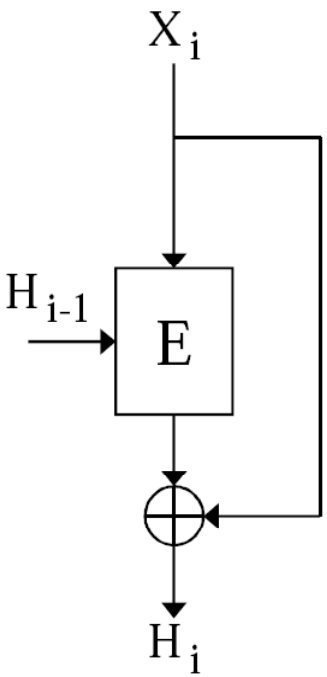
- kontrola integrity dat
- digitální podpis
- ukládání hesel
- porovnání shodnosti obsahu kopií dat
- generování pseudonáhodných posloupností

MDC Hash funkce založené na blokových šifrách

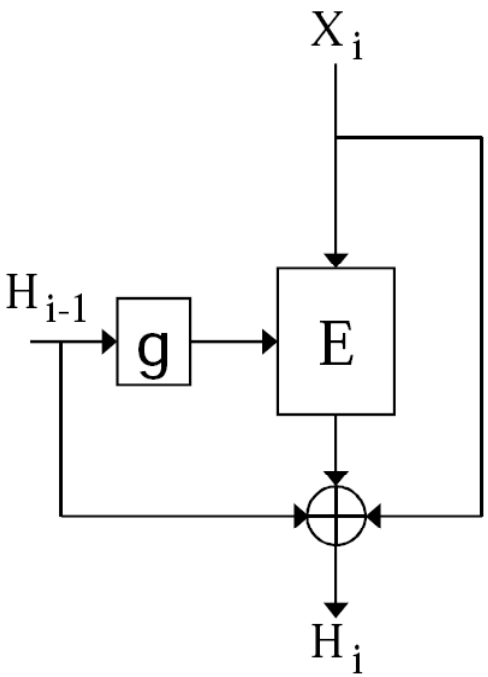
- Matyas - Meyer - Oseas



Davies - Meyer



Miyaguchi - Preneel



- g - úprava hash z předch. kroku (doplnění, konverze) $\rightarrow$ použití jako heslo blokové šifry
- inicializace náhodným vektorem *IV*

Hlavní MDC hash funkce

- 1977 - 1. varianty hash fce - základ alg. DES
- Ronald Rivest ("R" z RSA) [1]
  - 1989 - *MD2* - pomalé, kolize, délka 128b
  - 1990 - *MD4* - $-||-\times$ základ pro většinu používaných *MD*
  - 1991 - **MD5** - délka - 128b
  - 2008 - **MD6** - délka - 1-512b, do soutěže NIST SHA-3 (neúspěch-sám autor upozorňuje na nedostatky a vyjímá jej ze soutěže)
- 1993 - Zheng & al. - **HAVAL** - délka do 256b
- 1993-5 - NIST - **SHA** (Secure Hash Algorithm), **SHA-1** - délka 160b (SHA-1 = SHA s opravenými chybami)
- 1996 - Dobbertin - **RIPEMD** - délka 128-320b, od 160b považována za bezpečnou

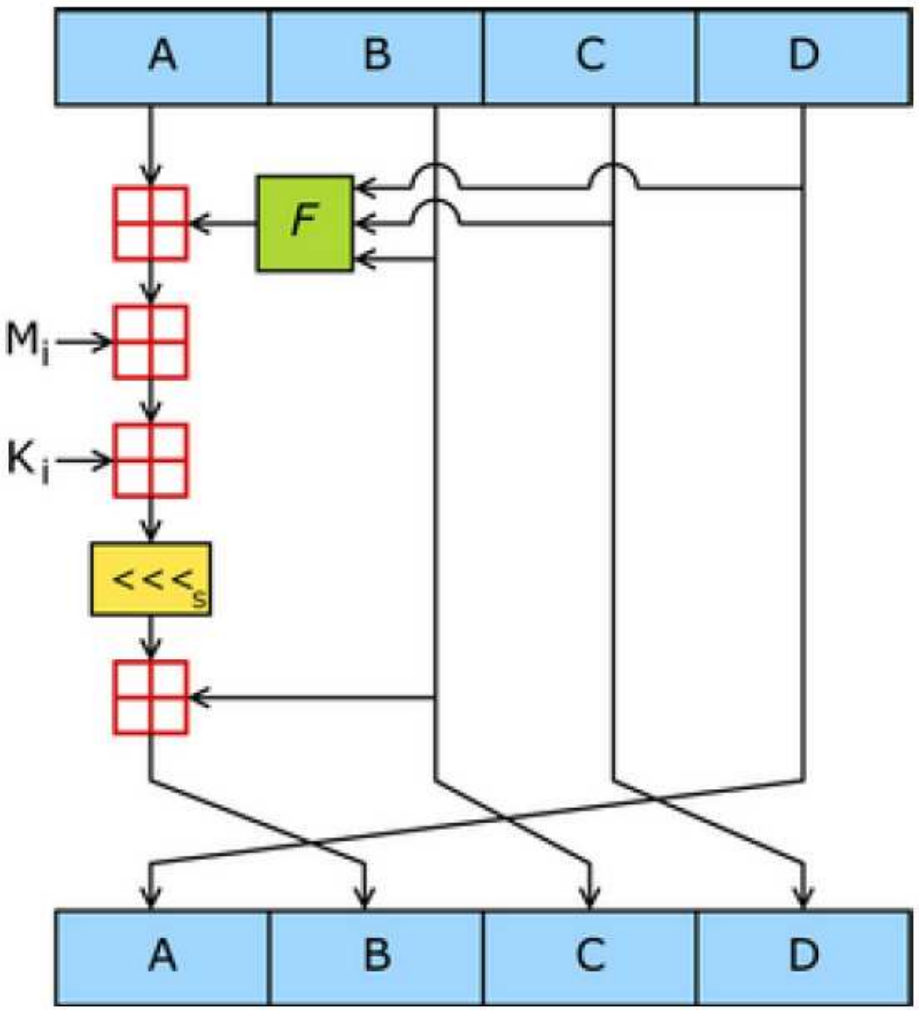
Hash funkce MD5

- 1991 - Ronald Rivest
- základ MD4, optimalizace pro 32 bit CPU
- více rezistentní proti kolizím (proti MD4)

- zpracování *vstupu* sekvence 512b (doplnění zprávy na násobky)
- blok 128 bitů, rozdělení na 4 části délky 32b
- 4 kola o 16 krocích $\longrightarrow$ 64 iterací
- *výstup*: 128b - A, B, C, D

- nejčastěji použitá hash fce
  - Př. md5sum (linux)
- 2004 - zveřejněn postup nalezení specifických kolizí; stále používaná, ale již není považována za bezpečnou

Hash funkce MD5 - princip



- $F_1(B, C, D) = (B \wedge C) \vee (\neg B \wedge D)$
- $F_2(B, C, D) = (B \wedge D) \vee (C \wedge \neg D)$
- $F_3(B, C, D) = B \oplus C \oplus D$
- $F_4(B, C, D) = C \oplus (B \vee \neg D)$

Operace

- nelin. fce f
- součet $mod\ 2^{32}$
- levá rotace
- M_i - zpráva - 32b
- K_i - konst - 32b

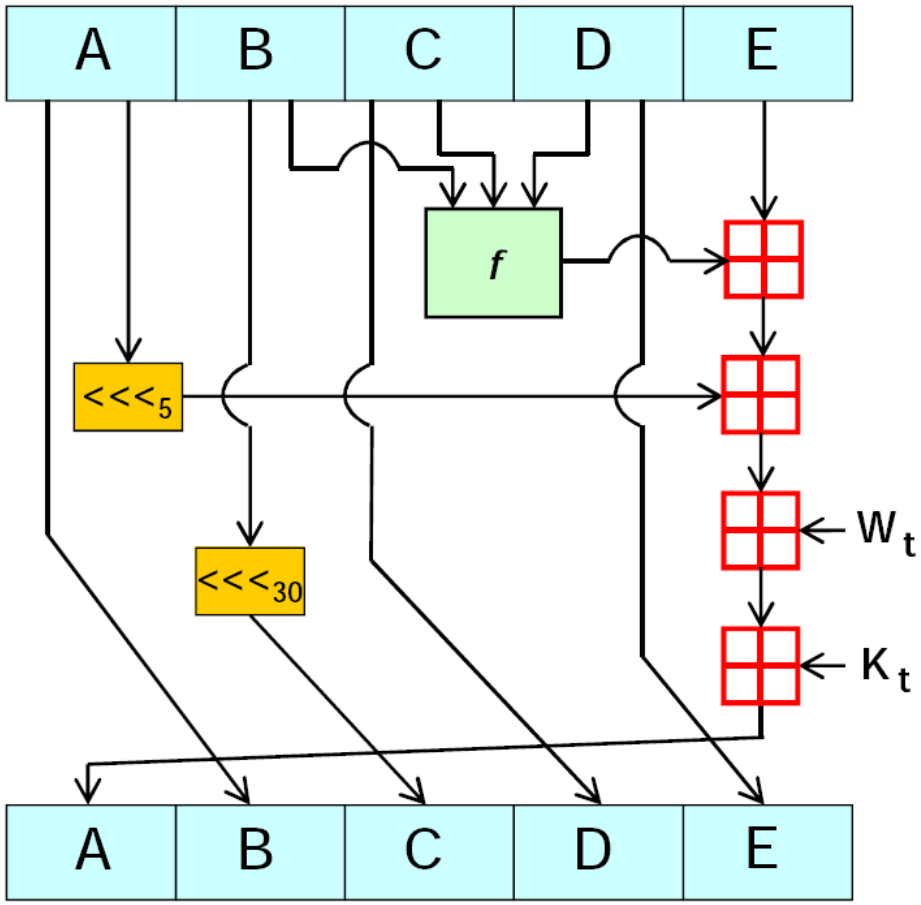
Hash funkce SHA-0, SHA-1

- 1993 - NIST - SHA-0 (Secure Hash Standard)
  - založena na MD4
  - kompresní fce 1 kolo navíc
  - kolo 20 kroků (místo 16)
  - jiné hodnoty *IV*
  - ...
- 1995 - pokyn NSA $\longrightarrow$ stažení těsně před schválením
- drobná modifikace $\longrightarrow$ SHA-1
  - přidání rotace vlevo - každá iterace kompresní fce

Hash funkce SHA-1

- zpracování *vstupu* různé délky (délka $< 2^{64} - 1$b):
- 4 rundy, 20 kroků v každé $\longrightarrow$ 80 iterací
- zpracování vstupu po sekvencích 512b
- doplnění zprávy na násobky 512b
- výstup 160b

Hash funkce SHA-1



- $f_1(B, C, D) = (B \wedge C) \vee (\neg B \wedge D)$
- $f_2(B, C, D) = B \oplus C \oplus D$
- $f_3(B, C, D) = (B \wedge C) \vee (B \wedge D) \vee (C \wedge D)$
- $f_4(B, C, D) = B \oplus C \oplus D$

Operace

- nelin. fce f, různá v každé rundě
- součet *mod* 2^{32}
- levá rotace
- W_t - zpráva - 32b
- K_t - konst - 32b

Bezpečnost SHA-1

- není garantována bezpečnost po r. 2010
- 2005 - návrh HW $SHA-1$ Cracker
  - 303 PC, každé 16 desek, každá 32 jader; cena: 1M $
  - doba prolomení 2 dny
- $\rightarrow$ NIST doporučuje její používání ukončit
- zatím používat $SHA-2$
- od r. 2012 - použití nové *AHS* (Advanced Hash Standard)
  - 2005 - NIST - soutěž o novou *bezpečnou* hashovací fci
  - ekvivalent k AES

Hash funkce typu MAC
Message Authentication Codes

Konstrukce z MDC

- přidání klíče do vstupní zprávy
- velmi jednoduché
- nevhodné

Vlastní návrh - platí

- $h = MAC_K(P)$, kde h - otisk, P - zpráva, K - klíč
- implementace
  - použití blokových šifrovacích algoritmů - př. CBC-MAC
  - použití hashovací funkce + klíče, tzv. HMAC (Hash MAC)

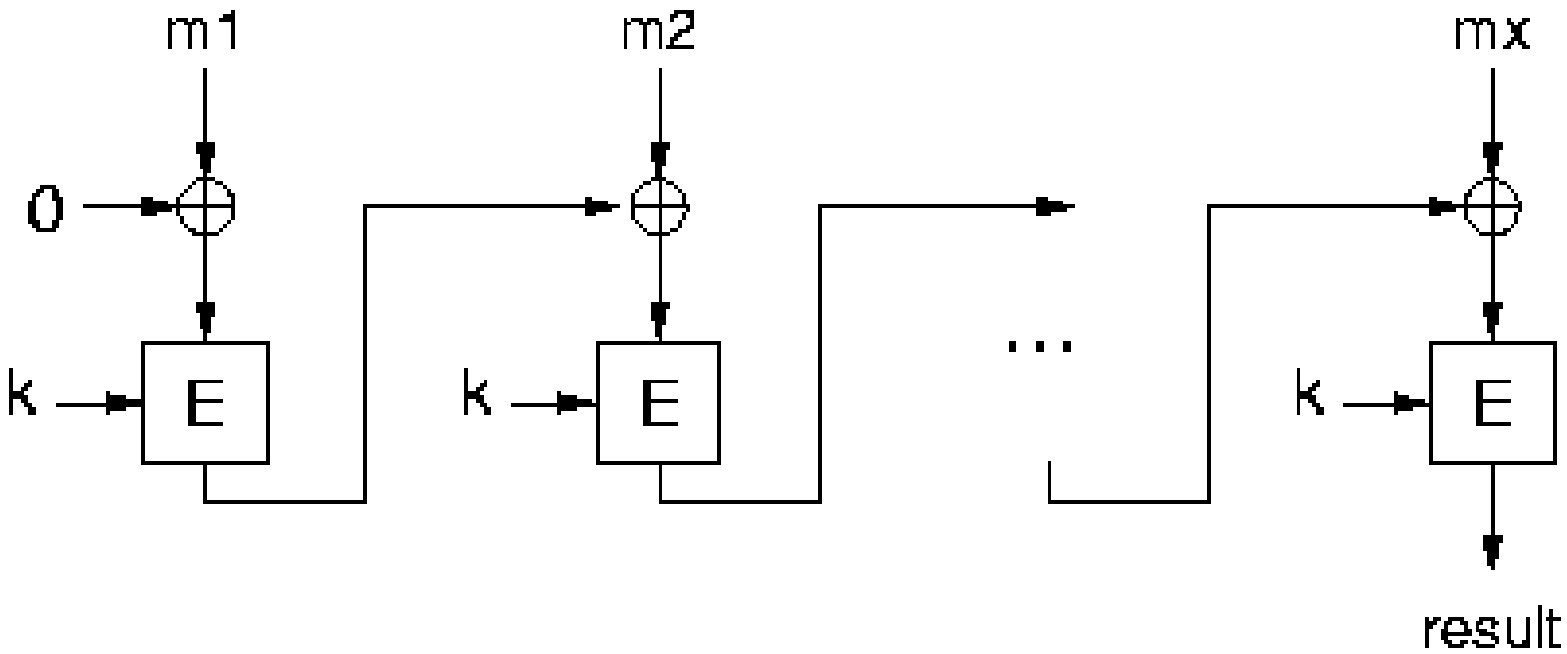
Dotaz:

- Použití klíčované hash fce dříve???

Použití blokových šifrovacích algoritmů

CBC - MAC - schéma

- rozložení zprávy na bloky: $m_1, .., m_x$
- inicializační vektor O

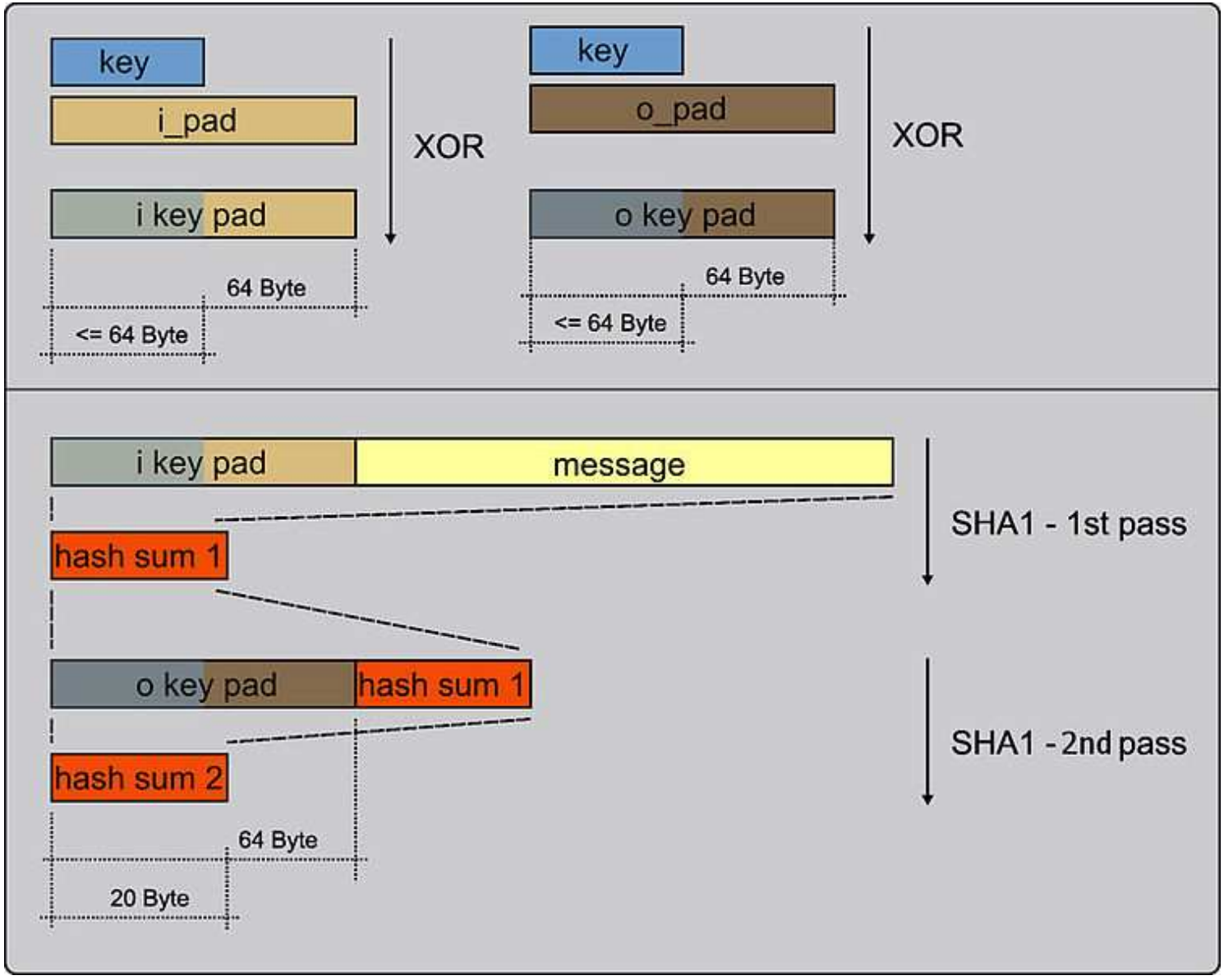


Použití hashovací funkce + klíče

Hash MAC

- možnost použití libovolné hashovací funkce typu MD
  - např. MD5 či SHA-1
  - $\rightarrow$ označení HMAC-MD5, příp. HMAC-SHA-1
- $\rightarrow$ bezpečnost HMAC - závislost na
  - použité hash fci.
  - velikosti a kvalitě klíče
  - délce výstupu hash funkce
- $\mathrm{HMAC}_K(P) = h\left((K \oplus \mathrm{opad}) \| h\left((K \oplus \mathrm{ipad}) \| P\right)\right)$
  - *opad* - konst., tzv. vnější zarovnání (outer padding)
  - *ipad* - konst., tzv. vnitřní zarovnání (inner padding)
- použití v protokolech IPsec a SSL/TLS

Hash MAC-SHA-1 - schéma



Útoky proti hash funkcím

Hledání kolizí

- P - existující zpráva
- $h(P)$ - hash zprávy P

Cíl útoku

- najít P', kde $h(P) = h(P')$
- $\longrightarrow$ hash. fce již **není bezpečná**
- např. digitální podpis, již nezaručuje *autenticitu* subjektu
  - v principu si jej může opatřit kdokoli

Útoky proti hash funkcím

Slovníkový útok

- znalost výsledku hash fce $\rightarrow$ snaha vyhledání původního řetězce
  - podmínka (krátký řetězec), tj. délka $<$ cca 448b
- *použití*: hledání běžných hesel
  - tj. prohledání dvojic hash - původní řetězec (vypočítáno dříve)
  - výsledky do několika vteřin
- nároky na úložný prostor

Rainbow tables

- ukládání pouze částí výpočtu ze všech možných výsledků hash fce $\rightarrow$ snížení nároků na prostor

Útoky proti hash funkcím

Srovnání

	Slovníkový útok	Útok hrubou silou	Rainbow table
Prostor klíčů	23 109	$\sim$ 8 miliard	$\sim$ 8 miliard
Příprava	1,05s	96h (odhad)	20h
Rychlost vyhledávání	$<$ 1s	dle alg.	$<$ 2,6s
Objem dat	$\sim$ 947KB	300GB	$\sim$ 611MB

Útoky proti hash funkcím
Metoda solení (tzv. salting)

$=$ *obrana proti útokům*

- vytvoření n-bitové náhodné hodnoty y
- výpočet $h = H(y, P)$
- uložení h, y místo h (použití - zejména při ukládání hesel)
  - př. Unix: použití 12 bitový salt - odvození ze systém. času
- výrazné zvýšení složitosti provádění slovníkových útoků a útoků Rainbow tables
- $\leftarrow$ jedna hodnota $\times$ nutnost určení tří informací P, y a algoritmu solení (kombinace P a y)

Generování náhodných čísel

- měření doby mezi stiskem kláves (Linux)
- pohyb myší (Windows)
- spouštění programů + vytvoření hashe jejich výstupu (Linux)
  - vmstat (monitorování zátěže) (př. vmstat | md5sum)
  - netstat -s (statistiky sítě) (př. netstat -s | sha1sum)
  - uptime (čas posl. rebootu + další info.) (př. uptime | sha512sum)

Použití

- náhodná hodnota pro salting
- generování dvojice VK, SK
- atd. ???

Integrita dat (z pohledu bezpečnosti)

Def:

- = zajištění, aby data nebyla úmyslně nebo neúmyslně změněna neoprávněným uživatelem, např. pozměněním, vložením nebo smazáním části dat, případně jejich opakováním ve zprávě

Př:

- Potvrzuji, že dlužím Frantovi 1 000 Kč, Alice.
- Potvrzuji, že dlužím Frantovi 1 000 000 Kč, Alice.

Porušení integrity dat

- záměrným pozměněním (útok)
- náhodným pozměněním (chyby HW & SW)

Způsoby zajištění integrity dat

- hash funkce (viz dříve)
- kontrolní součty
- žurnálování - Databázové Technologie (DT)
- integritní omezení - DT (entitní, referenční, doménové)
- synchronizace dat, propagace změn a maskování nekonzistentních mezistavů dat - distribuované systémy (více kopií dat)
- kompletnost dat - telekomunikace (zda se nic neztratilo při přenosu)

Kontrolní součet

- doplňková informace, předána spolu s daty - ověření úplnosti, správnosti dat
- příjemce: vypočtený kontrolní součet $\neq$ předaný kontrolní součet $\longrightarrow$ poškození zprávy nebo poškození kontrolního součtu

Metody kontrolního součtu

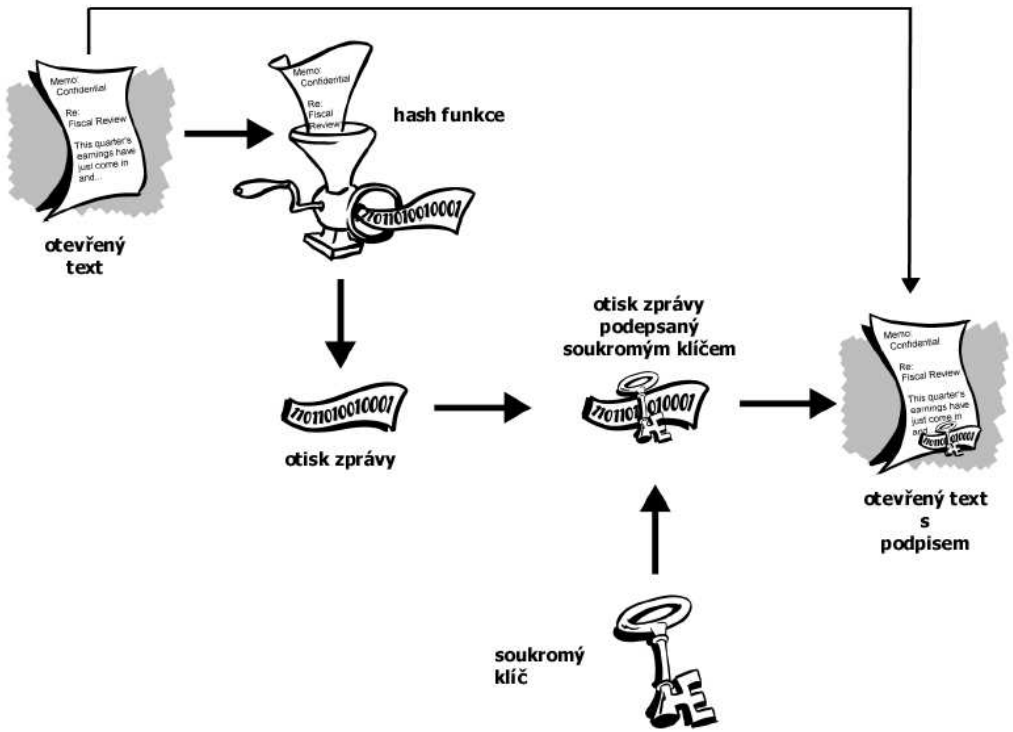
- kopie zprávy - bp. redundance dat $\longrightarrow$ nepoužíváno
- krátká dodatková informace (běžně používáno)
  1. ???
  2. ???
  3. ???
  4. ???

Metody kontrolního součtu

- kopie zprávy - bp. redundance dat $\longrightarrow$ nepoužíváno
- krátká dodatková informace (běžně používáno)
  1. Parita
  2. Modulo
  3. Hammingův kód (samoopravný)
  4. Cyklický Redundantní Součet (CRC) (samoopravný)

Elektronický (digitální) podpis

- analogie klasického podpisu v el. komunikaci
- typicky založen na *kontrolním součtu* a vlastnostech *asymetrické kryptografie*



Příjemce:

- ověření podpisu (rozšifrování hashe) pomocí veřejného klíče autora

Vlastnosti elektronického (digitálního) podpisu

- příjemce
  - možnost ověření identity odesílatele zprávy
  - nemožnost změny obsahu zprávy
- odesílatel
  - nemožnost pozdějšího odmítnutí obsahu dig. podepsané zprávy

Elektronický podpis založený na symetrické kryptografii

- existence důvěryhodné centrální autority (Server (S), obvykle ozn. Velký bratr, angl. Big Brother)
- každý uživatel - sdílení tajného klíče se S (Alice - K_{AS}, Bob K_{BS})

Alice - zaslání podepsané zprávy P Bobovi

1. $A \rightarrow S \ \{B, N_A, T, P\}_{K_{AS}}$
2. $S \rightarrow B \ \{A, N_A, T, P, \{A, T, P\}_{K_S}\}_{K_{BS}}$

Oskar - pokus o přehrání zpráv

- staré zprávy - odmítnutí díky T
- nové zprávy - odmítnutí na základě duplicitního N_A
- – důvěra S, přístup ke všem zprávám

Certifikáty veřejných klíčů

Cíl

- zajištění přenosu veřejných klíčů po nezabezpečeném kanálu
- nechť CA = Certifikační Autorita (důvěryhodnost)
  - vytvoření vlastní dvojice klíčů (veřejný zveřejněn, soukromý utajen)
  - uživatel - zveřejnění svého VK - poslání $\longrightarrow$ CA
    - ověření fyzické identity předkládajícího subjektu
    - připojení řetězce - identifikace tvůrce klíče a další data (např. doba platnosti)
    - podpis dat (data = klíč + identita předkladatele)

 "Rfc 1319, rfc 1320, rfc 1321," http://www.cert.dfn.de/eng/resource/rfc/rfc-tit.html.