

Základy moderní kryptologie - Symetrická kryptografie III.

operační módy blokových šifer a hašovací funkce

Vlastimil Klíma

verze 1.2

Abstrakt

Cílem třech přednášek Symetrická kryptografie I, II. a III je

- a) ukázat, že moderní kryptologie se zabývá mnohem širším okruhem věcí než jen utajováním zpráv a jejich luštěním,
- b) seznámit s novými myšlenkami,
- c) a věnovat se více jedné části moderní kryptologie, tzv. symetrickým schémátům.

Vzhledem k rozsahu těchto přednášek, které mají úvodní přehledový charakter, nebude možné postihnout ani klíčové, ani nejkrásnější myšlenky této vědy, ale jen některé nejpoužívanější. Následující texty vychází částečně z citované a doporučené literatury, jsou však nutně zatíženy subjektivním výkladem autora.

Doporučená literatura:

Základní příručka:

Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1996, dostupné celé on-line na <http://www.cacr.math.uwaterloo.ca/hac/>

Často doporučovaná alternativa:

Doug Stinson, *Cryptography: Theory and Practice*, CRC Press, 1995

Historie:

David Kahn, *The Codebreakers*, Scribner, 1996

Internetový portál:

<http://theory.lcs.mit.edu/~rivest/crypto-security.html>

Kontakt a osobní stránky:

v.klima@volny.cz, <http://cryptography.hyperlink.cz>

Poděkování: Autor vyjadřuje poděkování společnosti LEC s.r.o., <http://www.lec.cz>, za podporu při psaní textu přednášky.

Obsah

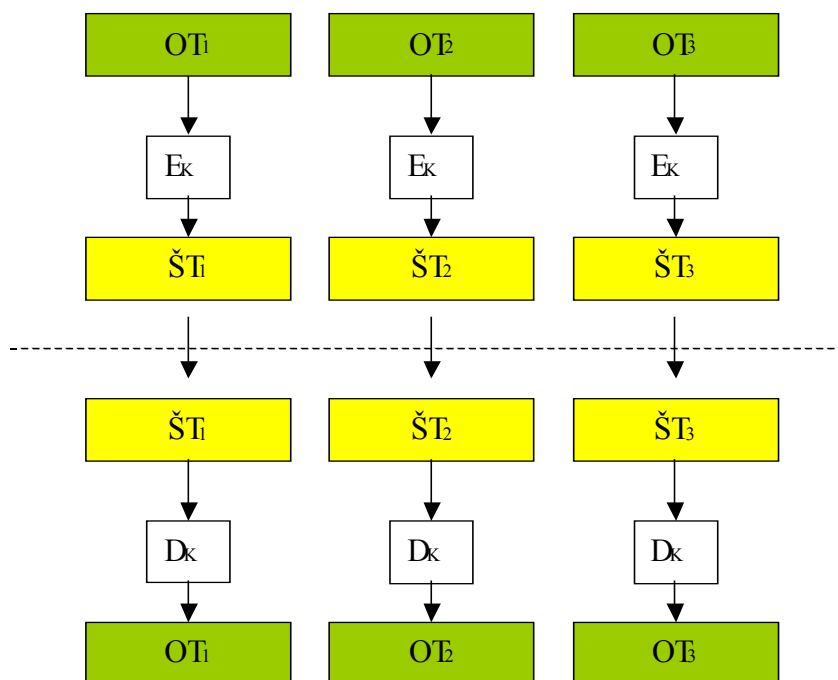
11.	Operační módy blokových šifer	3
11.1.	ECB (Electronic Codebook).....	3
11.1.1.	Informace, vyznačující ze šifrovaného textu v modu ECB.....	3
11.1.2.	Integrita a modus ECB	3
11.2.	CBC (Cipher Block Chaining).....	4
11.2.1.	Rozšíření difúze a konfúze z bloku na celý otevřený text.....	4
11.2.2.	Definice CBC	4
11.2.3.	Samosynchronizace modu CBC.....	5
11.3.	Mody CFB a OFB (Cipher Feedback, Output Feedback).....	5
11.3.1.	Samosynchronizace, neúplná zpětná vazba a délka periody.....	6
11.4.	Čítačový modus.....	7
11.5.	Metoda solení	8
11.6.	Útoky na synchronní proudové šifry a módy CFB, OFB a CTR	8
11.7.	MAC (Message Authentication Code).....	9
11.8.	Další módy	9
12.	Hašovací funkce	10
12.1.	Definice	10
12.2.	Konstrukce hašovacích funkcí	10
12.2.1.	Inicializace	11
12.2.2.	Zarovnání a doplnění vstupní zprávy	11
12.2.3.	Kompresce	11
12.3.	Příklad: SHA-1	12
12.3.1.	Doplnění	12
12.3.2.	Expanze bloku M(i) u SHA-1	12
12.3.3.	Hlavní smyčka.....	12
12.4.	Kolize	13
12.5.	Narozeninový paradox	13
12.6.	Kryptografické využití hašovacích funkcí	14
12.6.1.	Standardy a protokoly symetrické a asymetrické kryptografie.....	14
12.6.2.	Digitální otisk dat	14
12.6.3.	Kontrola integrity	14
12.6.4.	Porovnání rozsáhlých databází.....	14
12.6.5.	Ukládání přihlašovacích hesel.....	14
12.6.6.	Další příklady použití	15
12.7.	Nejpoužívanější hašovací funkce	15
12.8.	Hašovací funkce SHA-256, 384, 512 a 224.....	15
13.	Klíčovaný hašový autentizační kód - HMAC	16
13.1.	Obecná definice HMAC.....	16
13.1.1.	Nepadělatelný integritní kód	16
13.1.2.	Autentizace.....	16
14.	Literatura	17

11. Operační módy blokových šifer

Operační módy blokových šifer jsou způsoby použití blokových šifer v daném kryptosystému. kde otevřeným textem není jen jeden blok blokové šifry, ale obecně posloupnost znaků dané abecedy. U moderních blokových šifer chápeme jako znaky bajty, i když se délka bloku N uvádí v bitech. Obvykle $N = 64$ nebo 128 . Pomocí operačních módů můžeme získat nové zajímavé vlastnosti a využití blokových šifer. Seznámíme se s módy ECB, CFB, OFB, CBC, CTR a MAC.

11.1. ECB (Electronic Codebook)

Tento operační módus se nazývá elektronická kódová kniha a je základním módem. Posloupnost bloků otevřeného textu OT_1, OT_2, \dots, OT_n se šifruje tak, že každý blok je šifrován zvlášť, což lze vyjádřit vztahem $\mathring{S}T_i = E_K(OT_i)$, $i = 1, 2, \dots, n$.



Obr.: Modus ECB

11.1.1. Informace, vyzařující ze šifrového textu v modu ECB

Nevýhodou takového typu šifrování je, že **stejné bloky otevřeného textu mají vždy stejný šifrový obraz**. Pokud nalezneme několik shodných bloků šifrového textu, že může to v určitém kontextu dokonce rozkrývat i hodnotu otevřeného bloku (například prázdné sektory na disku jsou vyplněny hodnotou $0xFF$ apod).

11.1.2. Integrita a modus ECB

Ve zprávě, šifrované módem ECB **útočník může bloky šifrového textu vyměňovat, vkládat nebo vyjmát**, a tak snadno docílovat pro uživatele nežádoucích změn v otevřeném textu, zejména, pokud nějakou dvojici ($OT, \mathring{S}T$) už zná. Tím je opět ilustrován v praxi často opomíjený fakt, že integrita otevřeného textu se šifrováním nezajistí.

Na obrázku vidíme, že vložením šifrového bloku se docílí změna otevřeného textu, aniž je nutné znát šifrovací klíč.

.....	3tdszj34	j7čžuths	bgžc4rš7	rg43č7řz	
.....	převedte	1	0 0 0	, - Kč	
.....	3tdszj34	j7čžuths	bgžc4rš7	bgžc4rš7	rg43č7řz
.....	převedte	1	0 0 0	0 0 0	, - Kč

Obr.: Útok na modus ECB vložení bloku šifrového textu

11.2. CBC (Cipher Block Chaining)

11.2.1. Rozšíření difúze a konfúze z bloku na celý otevřený text

Synchronní proudové šifry mají nedostatečnou vlastnost difúze neboť pracují jen nad jednotlivými znaky abecedy. Moderní blokové šifry naproti tomu dosahují velmi dobrých vlastností jak difúze, tak konfúze.

Je tomu tak proto, že vznikaly už v době elektronických šifrátorů, kdy nebyl problém vytvářet iterativní součinnové šifry. Větší počet iterací a vhodné rundovní funkce pak mohly vlastnost difúze a konfúze zajistit s dostatečnou rezervou. Provést tolik operací v případě ručního šifrování, u mechanických nebo elektromechanických šifrátorů nebylo technicky možné. Difúze a konfúze při zpracování jednoho bloku otevřeného textu u moderních blokových šifer byla tedy dosažena. Je to ale málo, neboť otevřeným textem je obvykle velmi mnoho bloků.

Aby blokové šifry rozšířili difúzi na více bloků, byl definován modus řetězení šifrového textu (CBC). Každý blok otevřeného textu se v něm nejprve modifikuje předchozím blokem šifrového textu, a teprve poté se šifruje. To zajišťuje, že běžný šifrový blok závisí na celém předchozím otevřeném textu z důvodu řetězení této závislosti přes předchozí šifrový text.

11.2.2. Definice CBC

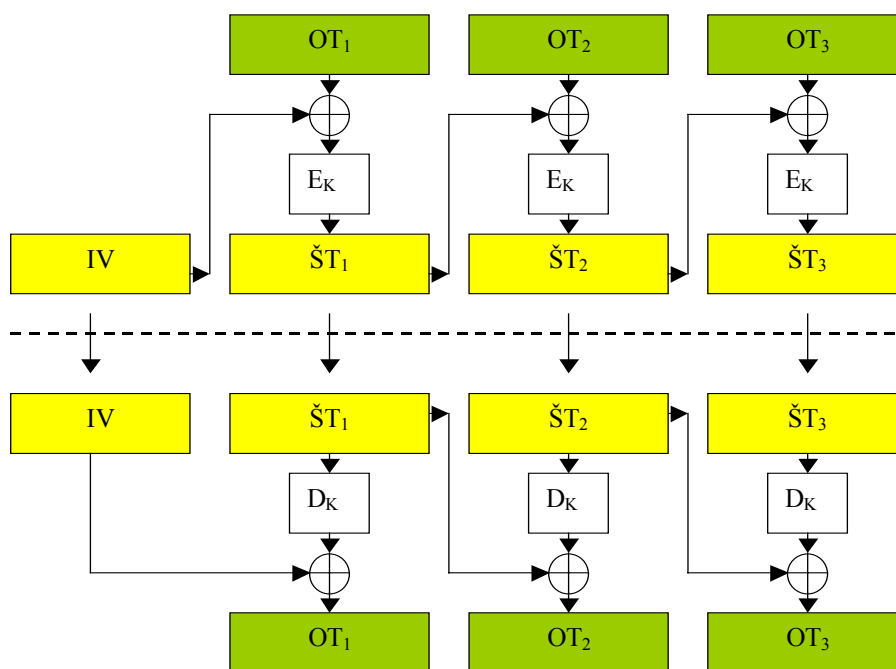
CBC je v současné době nejpoužívanějším operačním modem blokových šifer. Eliminuje některé slabosti modu ECB a zajišťuje difúzi celého předchozího otevřeného textu do daného bloku šifrového textu. První blok otevřeného textu je modifikován náhodnou, tzv. inicializační hodnotou (initializing value, IV), která je vysílána před vlastním šifrovým textem, podobně jako u proudových šifer. Šifrování se provádí podle vztahů

$$\begin{aligned} \check{S}T_0 &= IV, \\ \check{S}T_i &= E_K(OT_i \oplus \check{S}T_{i-1}), i = 1, 2, \dots, n. \end{aligned}$$

a odšifrování probíhá podle vztahů:

$$\begin{aligned} \check{S}T_0 &= IV, \\ OT_i &= \check{S}T_{i-1} \oplus D_K(\check{S}T_i), i = 1, 2, \dots, n. \end{aligned}$$

Náhodný inicializační vektor způsobí, že pokud bychom dvakrát šifrovali shodný první blok OT_1 , různé náhodné IV ho modifikují na různé náhodné vstupní bloky. Výsledný blok šifrového textu $\check{S}T_1$ je proto také náhodný a různý. Protože však tento blok přebírá úlohu inicializačního vektoru pro šifrování následujícího bloku otevřeného textu, efekt znáhodnění se postupně projeví i na dalších blocích. **Budeme-li šifrovat jeden a tentýž, byť i velmi dlouhý, otevřený soubor dat dvakrát, obdržíme naprosto odlišný šifrový text.**



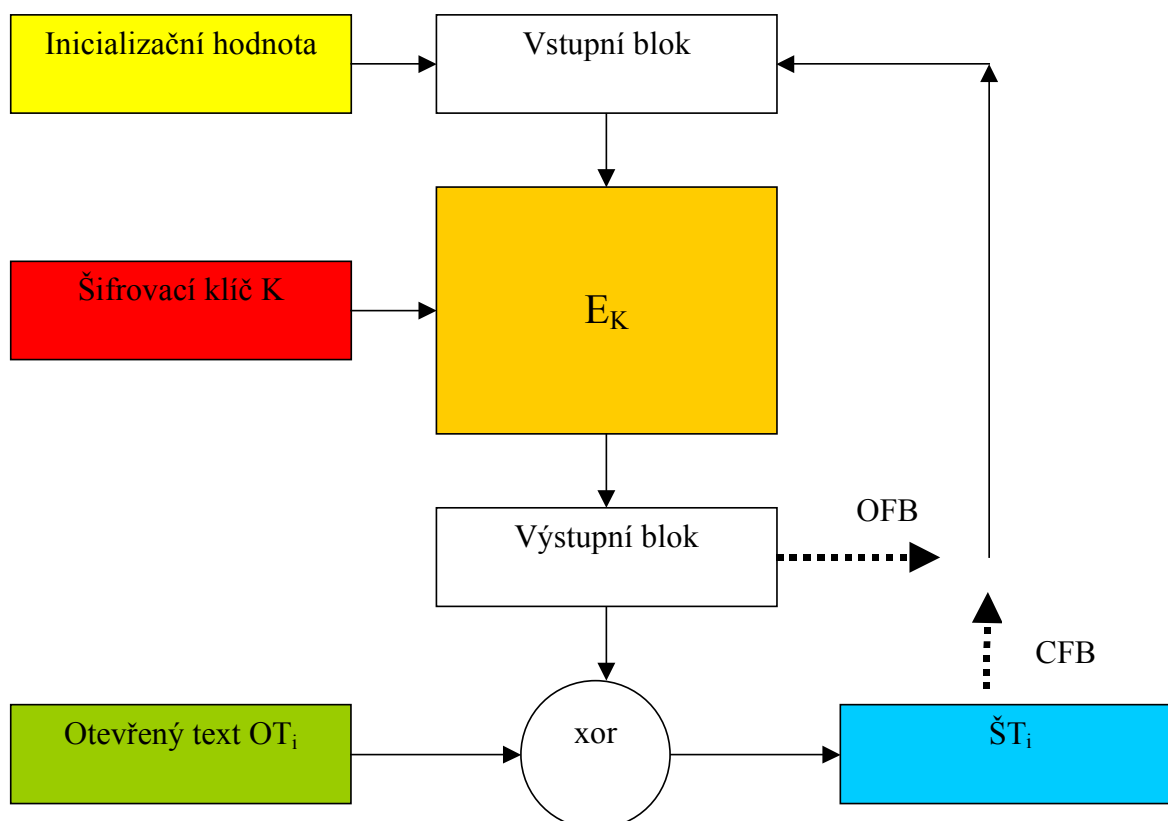
Obr.: Modus CBC

11.2.3. Samosynchronizace modu CBC

Řetězení mírně znesnadňuje útoky na modus ECB popsané výše. Z definice modu CBC však vyplývá tzv. vlastnost samosynchronizace. Proces odšifrování je schopen se zotavit a produkovat správný otevřený text i při výpadku nebo porušení několika bloků šifrového textu. K obnově otevřeného textu dojde už při dvou za sebou jdoucích správných blocích ŠT. Z rovnice pro odšifrování vidíme, že ke správnému odšifrování bloku OT_i potřebujeme pouze správné bloky šifrového textu $ŠT_{i-1}$ a $ŠT_i$.

11.3. Mody CFB a OFB (Cipher Feedback, Output Feedback)

Tyto operační mody převádí blokovou šifru na proudovou. Používají náhodnou inicializační hodnotu IV k nastavení odpovídajícího konečného automatu do náhodné polohy. Tento automat pak produkuje posloupnost hesla, které se jako u proudových šifer xoruje na otevřený text. První blok hesla se získá zašifrováním IV. Konečný automat pracuje tak, že vzniklé heslo (v modu OFB) nebo vzniklý šifrový text (v modu CFB) jsou vedeny na vstup blokové šifry a jejich zašifrováním je produkován následující blok hesla. OFB má vlastnost čisté (synchronní) proudové šifry, neboť heslo je generováno zcela autonomně bez vlivu otevřeného a šifrového textu. CFB je kombinací vlastností CBC a proudové šifry.



Obr.: Modus zpětné vazby z výstupu (OFB) a ze šifrového textu (CFB)

Předpis pro zašifrování v modu CFB:

$$\check{S}T_0 = IV,$$

$$\check{S}T_i = OT_i \oplus E_K(\check{S}T_{i-1}), i = 1, 2, \dots, n.$$

Předpis pro odšifrování v modu CFB:

$$\check{S}T_0 = IV,$$

$$OT_i = \check{S}T_i \oplus E_K(\check{S}T_{i-1}), i = 1, 2, \dots, n.$$

Předpis pro zašifrování v modu OFB:

$$H = IV = \check{S}T_0,$$

$$\text{pro } i = 1, 2, \dots, n: \{H = E_K(H), \check{S}T_i = OT_i \oplus H\}$$

Předpis pro odšifrování v modu OFB:

$$H = IV = \check{S}T_0,$$

$$\text{pro } i = 1, 2, \dots, n: \{H = E_K(H), OT_i = \check{S}T_i \oplus H\}$$

Všechny uvedené operační mody definují normy FIPS PUB 81, ANSI X3.106, ISO 8732 a ISO/IEC 10116. Povšimněte si, že v **modech OFB a CFB se bloková šifra používá jen jednosměrně** tj. jen transformace E_K . Transformaci D_K není použita. To je výhodné při hardwarové realizaci.

11.3.1. Samosynchronizace, neúplná zpětná vazba a délka periody

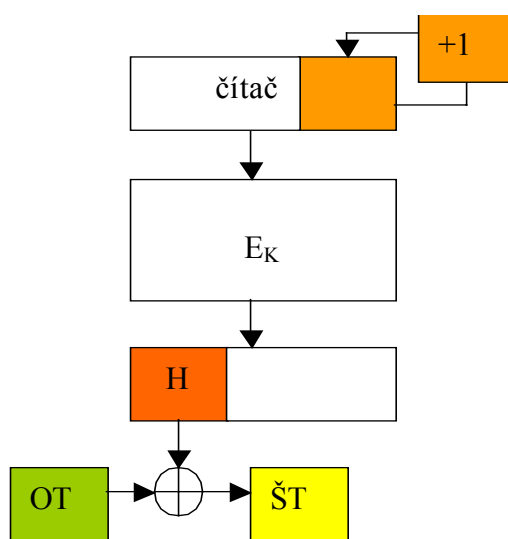
Poznamenejme, že z výstupu nemusíme použít celý blok hesla, ale jen jeho část, například b bitů. V tom případě se b bitů hesla (u OFB) nebo b bitů vzniklého šifrového textu (u CFB) vede zprava do vstupního registru, přičemž původní obsah vstupního registru se posune doleva o b bitů (b bitů nejvíce vlevo z registru vypadne).

Modus CFB má vlastnost samosynchronizace, a to podle délky zpětné vazby. Je-li b bitů, pak postačí dva nenarušené b -bitové bloky šifrového textu, aby se otevřený text sesynchronizoval.

Modus OFB poskytuje synchronní proudovou šifru. Heslo generuje konečným automatem, který má maximálně 2^N vnitřních stavů. Po tomto počtu kroků se produkce hesla musí nutně opakovat. **Délka periody hesla** je proto maximálně 2^N , její konkrétní délka je určena hodnotou IV a může se pohybovat náhodně v rozmezí od jedné do 2^N . Struktura hesla je značně závislá na tom, zda zpětná vazba je plná nebo nikoli. Pro $b < N$ je střední hodnota délky periody pouze cca $2^{N/2}$, zatímco pro $b = N$ je to 2^{N-1} .

11.4. Čítačový modus

Čítačový modus (**CTR, Counter Mode**) je v principu velmi podobný modu OFB a také převádí blokovou šifru na synchronní proudovou šifru. Odstraňuje problém s neznámou délkou periody hesla, neboť zde je délka periody hesla dána předem.



Obr.: Čítačový modus

Používá také inicializační hodnotu IV, která se načte do vstupního registru (čítače) T. Po jeho zašifrování vzniká první blok hesla. Poté dojde k aktualizaci čítače T, nejčastěji přičtením jedničky (odtud název modu) a ke generování dalšího bloku hesla. Heslo se může využít v plné šíři bloku nebo jen jeho $b < N$ bitů.

Způsob aktualizace čítače je definován poměrně volně, inkrementovat se může jen například dolních B bitů čítače T. Musí se však dodržet zásada, aby ani v jedné zprávě, ani v dalších zprávách šifrovaných tímtež klíčem, nedošlo k vygenerování stejného bloku hesla dvakrát, tj. musí se zabránit tomu, aby byl obsah čítače stejný. V takovém případě by došlo k tzv. dvojímu použití hesla a mohlo by dojít k rozluštění otevřeného textu dotčených zpráv. Podrobnější definice s příklady vytváření čítače jsou uvedeny v [25].

Předpis pro zašifrování v modu CTR:

$$CTR_i = (IV + i - 1) \bmod 2^B, H_i = E_K(CTR_i), ŠT_i = OT_i \oplus H_i, i = 1, 2, \dots$$

Předpis pro odšifrování v modu CTR:

$$CTR_i = (IV + i - 1) \bmod 2^B, H_i = E_K(CTR_i), OT_i = ŠT_i \oplus H_i, i = 1, 2, \dots$$

I když se jedná o nedávno standardizovaný modus, jeho myšlenku popsali v roce 1979 Diffie a Hellman [26]. Jedná se o čistou (synchronní) proudovou šifru, používající pouze

transformaci E_K i při dešifrování. Výstupní blok lze použít celý nebo jen jeho část. Smyslem modu je zaručit maximální periodu hesla, což je zaručeno periodou čítače.

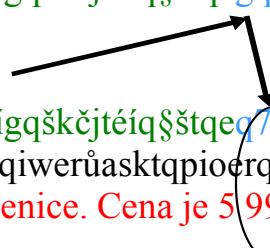
Další výhodou je, že heslo může být vypočítáno jen na základě pozice otevřeného textu a IV , nezávisle na ničem jiném. Tuto vlastnost má i modus OFB, ale než se u něj vypočítá heslo, příslušné například k miliontému bloku šifrového textu, je potřeba provést milion šifrovacích transformací E_K . Naproti tomu u modu CTR se vypočte hodnota čítače (zde counter = 1000000) a provede se jen jedna transformace E_K : $E_K(\text{counter})$. Možnost vypočítat heslo jen na základě pozice daného bloku v otevřeném nebo šifrovém textu je výhodná například v takových systémech, které poskytují jen danou část otevřeného nebo šifrového textu a nikoli jeho okolí. Modus CFB se liší tím, že potřebuje předchozí blok šifrového textu. CTR naproti tomu zase nemá vlastnost samosynchronizace.

11.5. Metoda solení

Poznamenejme, že u operačních modů CBC, OFB, CFB i CTR je též možné využívat metodu solení IV . Spočívá v tom, že komunikujícímu protějšku se sice předává hodnota IV , ale k šifrování se použije jiná hodnota IV' ("osolený IV "), která se na obou stranách vypočítá z IV a klíče K nějakým definovaným způsobem. Například to může být hašovací hodnota, vypočítaná ze zřetězení obou hodnot. Bezpečnostní výhodou je, že skutečně použitá inicializační hodnota IV' se nikde neobjevuje na komunikačním kanálu. Metodou solení (salting) se zabývá norma PKCS#5 [27].

11.6. Útoky na synchronní proudové šifry a mody CFB, OFB a CTR

Synchronní proudové šifry a blokové šifry v proudových modech CFB, OFB a CTR jsou náchylné na útok změnou šifrového textu. Změna v šifrovém textu ($\check{S}T \oplus d$) se promítá do otevřeného textu jako ($OT \oplus d$), tedy velmi přímočaře. Toho lze využít k různým útokům, princip ilustruje obrázek (poznamenejme, že šifrový text je smyšlený).

OT1: kontrakt na dodávku pšenice. Cena je 5 000 000,- USD. Splatn....
H: kasũfkaiqpoirksdaũirtpqiwerũasktqpioerqũkdjairqpiraskgaũirup....
ŠT : áílkjěšdgjkqwšpéitũ\$aeígqškcjtéiqšštqegq\$dlgšrlflũ\$leglladgwá...
(xor 999 na ŠT) 
ŠT : áílkjěšdgjkqwšpéitũ\$aeígqškcjtéiqšštqeg78dlgšrlflũ\$leglladgwá...
H: kasũfkaiqpoirksdaũirtpqiwerũasktqpioerqũkdjairqpiraskgaũirup....
OT2: kontrakt na dodávku pšenice. Cena je 5 999 000,- USD. Splatn....

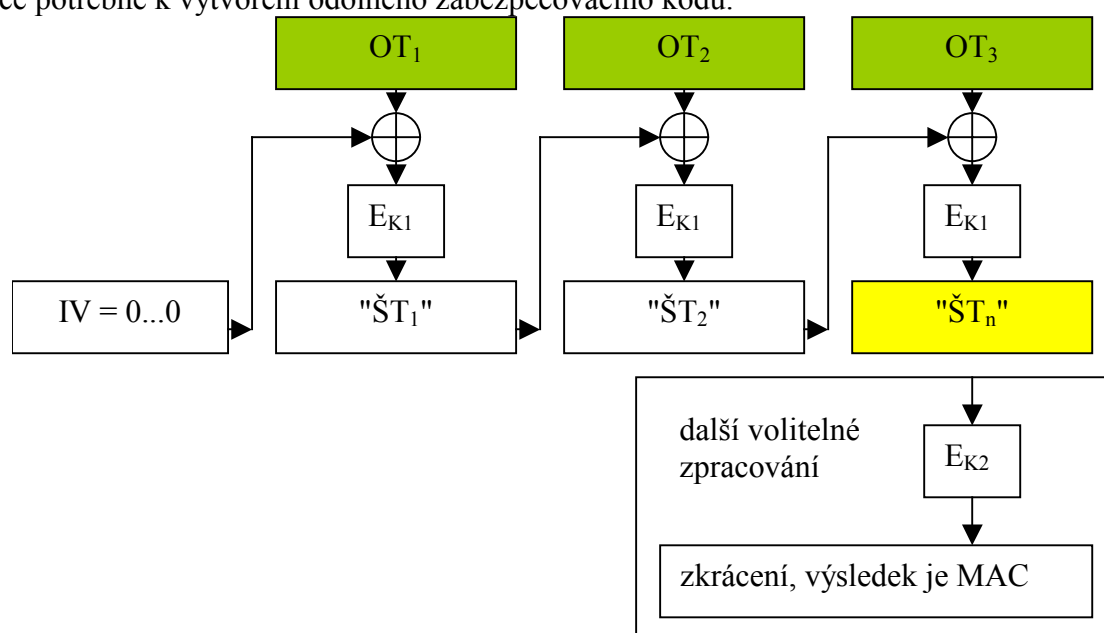
dešifrováním změněného šifrového textu ($\check{S}T1 \oplus d$) obdržíme
 $(\check{S}T1 \oplus d) \oplus H = (OT1 \oplus H \oplus d) \oplus H = OT1 \oplus d$

Obr.: Příklad útoku na proudovou šifru změnou šifrového textu

11.7. MAC (Message Authentication Code)

Jak jsme ukázali výše, proudové a blokové šifry zajišťují důvěrnost, ale nikoli **integritu** zpráv. Mody CBC a CFB sice způsobí mírnou propagaci chyby (chyba v jednom bloku šifrovaného textu naruší dva bloky otevřeného textu), ale v systémech, kde není ve vlastním otevřeném textu zajištěna nějaká redundance, nemusí být tato chyba pozorována a příslušným automatizovaným systémem mohou být zpracována chybná data. **Autentizační kód zprávy** (MAC) je dalším modem blokové šifry, který řeší právě zajištění neporušenosti dat. Tento zabezpečovací kód autentizuje původ zprávy a řeší obranu proti náhodným i úmyslným změnám nebo chybám na komunikačním kanálu. MAC je krátký kód, který vznikne zpracováním zprávy s tajným klíčem (K1). Klíč by se měl použít jiný, než k šifrování zprávy.

Výpočet MAC probíhá tak, že se zpráva jakoby šifruje v modu CBC s nulovým inicializačním vektorem, přičemž průběžný šifrový text se nikam neodesílá. MAC je pak tvořen až posledním blokem ŠT_n , přičemž je možné ještě jedno přídatné šifrování navíc, tj. $\text{MAC} = E_{K2}(\text{ŠT}_n)$. Z výsledného bloku se obvykle bere jen určitá část (často polovina bloku) o délce potřebné k vytvoření odolného zabezpečovacího kódu.



Obr.: Autentizační kód zprávy MAC

Pokud ke zprávě, ať otevřeně nebo šifrované, připojíme MAC, příjemce může ověřit, že data pochází od toho, kdo zná klíč K1 (ev. přídatný K2). MAC proto také zajišťuje službu **autentizace původu dat**. Protože je to symetrická technika, **nezaručuje nepopíratelnost**. Nezávislá třetí strana nemůže v případě sporu rozhodnout, zda data i s jejich zabezpečením vytvořil odesílatel nebo příjemce.

11.8. Další mody

Přestože se zdá, že operačních modů je dost, není tomu tak. Nové potřeby vyžadují nové mody, příkladem budiž nutnost šifrovat data a současně počítat jejich zabezpečovací kód MAC. Současná kombinace modů CBC a MAC může však být ve výpočetně omezených zařízeních pomalá nebo příliš náročná na systémové prostředky (zejména paměť). Proto NIST vyhlásil novou iniciativu na vytváření nových modů. Blíže viz web NIST [25]. Například v září r. 2003 byl publikován pracovní materiál [28], kde se definuje modus CCM. Používá AES v čítačovém modu k šifrování v kombinaci s technikou CBC-MAC pro autentizaci. Práce na modech jsou živé, takže se stále jedná o pracovní materiály, které nejsou uzavřeny.

12. Hašovací funkce

Hašovací funkce jsou silným nástrojem moderní kryptologie. Jsou jednou z klíčových myšlenek druhé poloviny dvacátého století a přinesly řadu nových použití. V jejich základu jsou pojmy jednocestnosti a bezkoliznosti.

12.1. Definice

Definice: jednosměrná (jednocestná) funkce

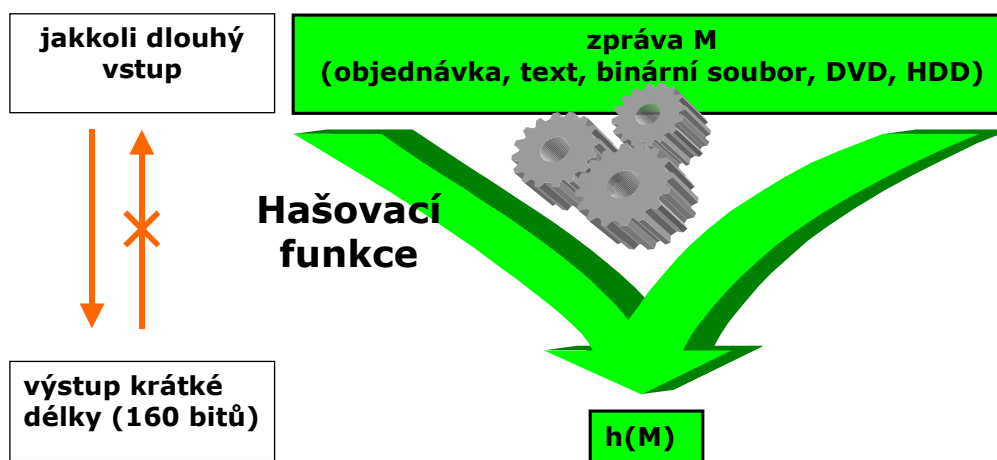
Funkci $f: X \rightarrow Y$ nazveme jednosměrnou (jednocestnou), jestliže z jakékoli hodnoty $x \in X$ je snadné vypočítat $y = f(x)$, ale pro náhodně vybranou hodnotu $y \in f(X)$ neumíme (je výpočetně nemožné) najít její vzor $x \in X$ tak, aby $y = f(x)$.

Definice: bezkolizní funkce

Funkci $f: X \rightarrow Y$ nazveme bezkolizní, jestliže je výpočetně nemožné nalézt různá $x, x' \in X$ tak, že $f(x) = f(x')$.

Definice: hašovací funkce

Mějme přirozená čísla L, n a necht' X je množina všech binárních řetězců délky 0 až L (prázdný řetězec je platným vstupem a má délku nula). Funkci $h: X \rightarrow \{0,1\}^n$ nazveme hašovací, jestliže je jednosměrná a bezkolizní. Říkáme, že každému binárnímu řetězci z množiny X přiřadí binární **hašový kód (hash, haš)** délky n .



Obr.: Hašovací funkce

Poznámky:

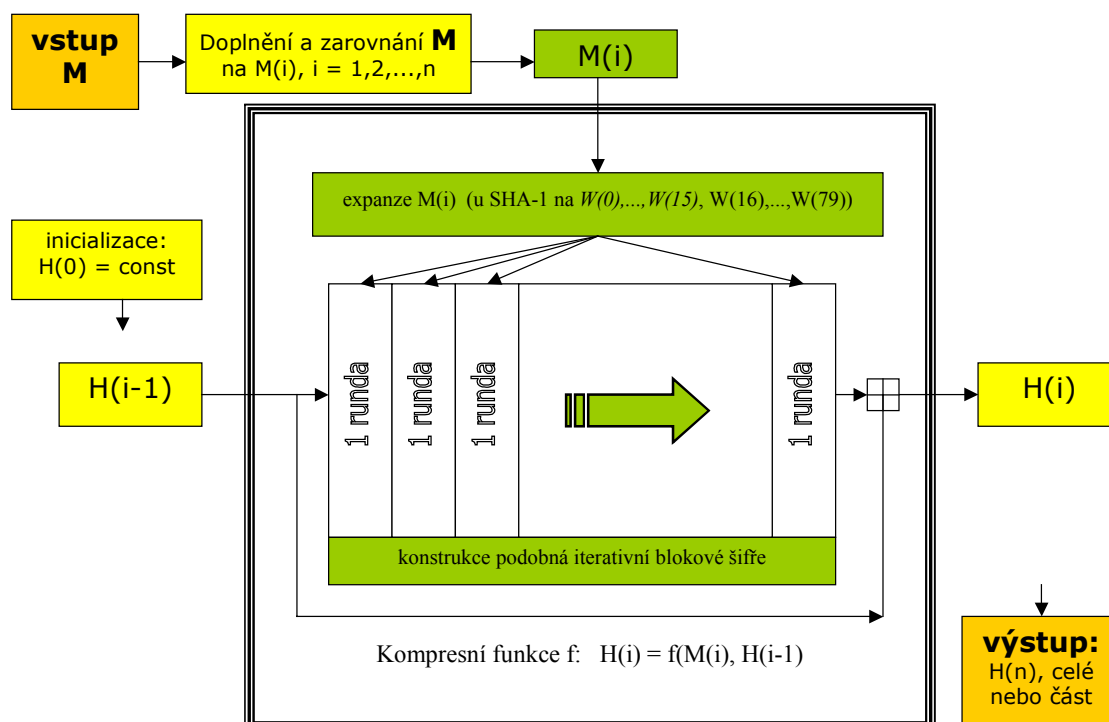
V praxi bývá $L = 2^{64} - 1$, $L = 2^{128} - 1$ apod., takže vstupem hašovací funkce je (z hlediska současných výpočetních aplikací) libovolný a prakticky neomezeně dlouhý binární řetězec M . Výstupem je naopak hašový kód $h(M)$ pevné, krátké a předem stanovené délky n bitů. Z vlastnosti jednocestnosti vyplývá, že pro dané M je jednoduché vypočítat hašový kód $h(M)$, ale obráceně to není výpočetně zvládnutelné.

12.2. Konstrukce hašovacích funkcí

Konstrukce řady hašovacích funkcí je založena na tzv. kompresní funkci f , která se použije v mnoha krocích - iteracích (i). Hašovací funkce si vytváří tzv. kontext, který si označíme H , a který se v každém kroku (i) mění. Kontextem bývá obvykle několik 16bitových nebo 32bitových proměnných (v počítačové praxi se nazývají "slova").

12.2.1. Inicializace

Na počátku je kontext naplněn inicializační hodnotou $H(0)$, což jsou nějaké veřejné konstanty, odvozené od známých čísel tak, aby bylo zřejmé, že nemají žádný jiný význam než jakékoliv konstanty.



Obr.: Hašování a kompresní funkce

12.2.2. Zarovnání a doplnění vstupní zprávy

Vstupní zpráva M může být libovolně dlouhá, například několik bitů nebo Gigabitů, a protože kompresní funkce pracuje s bloky pevné délky (B bitů), je nutno zprávu nějakým způsobem doplnit a zarovnat na celé bloky B bitů. Protože zprávu dorovnáme, je nutné také uložit někde původní délku zprávy, aby nedorovnaná a dorovnaná zpráva nebyly z hlediska hašování totožné.

Zpráva se proto většinou doplní nejprve jedním bitem s hodnotou 1, poté určitým počtem nulových bitů a poté binárním vyjádřením délky původního vstupu tak, že celkem dostaneme nejbližší celistvý násobek B -bitových bloků. Tyto bloky označme $M(1), \dots, M(n)$, přičemž B bývá 512 nebo 1024, viz obrázek dále.

12.2.3. Kompresce

Potom se v cyklu podle i od 1 do n zpracovává vždy běžný blok $M(i)$ současně s předchozím kontextem haše $H(i-1)$, tj.

$$H(i) = f(M(i), H(i-1)).$$

Po zpracování posledního bloku $M(n)$ se jako výstup celé hašovací funkce definuje kontext $H(n)$ nebo jeho část. Kompresní funkce připomíná blokovou funkci, přičemž části zprávy $M(i)$ působí na místě rundovních klíčů, zatímco zpracování kontextu připomíná modus řetězení šifrového textu. Kontext hašovací funkce můžeme chápat jako jakýsi akumulátor, kam se "přimíchávají" nové části zprávy. Aby zpracování každého bloku $M(i)$ bylo co nejsložitější, sama kompresní funkce pracuje v iteracích, například SHA-1 má 80 iterací. Samotný blok $M(i)$ o 512 bitech je u SHA-1 rozdělen na 16 32bitových slov $W(0)$ až $W(15)$, a ty jsou ještě dále expandovány na $W(16)$ až $W(79)$, tedy původní blok $M(i)$ je uměle "natažen" na pětinašobek. Do nových slov $W(i)$ se dále promítá závislost na původním bloku

$M(i)$. Nyní je vstup $H(i-1)$ zpracován v 80 rundách, přičemž rundovní funkce používají jako řídicí parametr právě odpovídající slovo $W(i)$. Rundovní funkce jsou stejné až na konstanty a drobné funkční odchylky. Po 80. rundě se ještě k výsledku aritmeticky přičte původní vstup $H(i-1)$, což vytváří ještě složitější vazby mezi vstupem a výstupem kompresní funkce. To je princip zpracování bloku kompresní funkcí u SHA-1. Podrobný popis SHA-1 je k dispozici ve standardu FIPS PUB 180-2 [29].

Složitost konstrukce kompresní funkce je základem pro vlastnost jednosměrnosti hašovací funkce.

12.3. Příklad: SHA-1

V tomto příkladu uvedeme pro ilustraci složitosti hašovací funkce výňatek popisu SHA-1[29].

12.3.1. Doplnění

Mějme zprávu M , skládající se ze tří písmen "abc". Nejprve provedeme její doplnění. Tato zpráva má délku 24 bitů, proto bude tvořit pouze jeden blok $M(1)$. Ten se bude skládat z bitové reprezentace znaků abc (24 bitů), následuje bit 1 a nulové bity až do délky $448 = 512 - 64$ a poté 64bitové vyjádření čísla $L = 24$, viz obrázek.

01100001	01100010	01100011	1	000000.....0000000000	00....00011000
a	b	c	1	nulové bity	číslo 24

Obr.: Doplnění zprávy "abc" na 512bitový blok

12.3.2. Expanze bloku $M(i)$ u SHA-1

512 bitů $M(i)$ tvoří 16 32bitových slov $W(0)$ až $W(15)$, které jsou expandovány na $W(16)$ až $W(79)$ pro $t = 16 \dots 79$ předpisem

$W(t) = (W(t-3) \text{ XOR } W(t-8) \text{ XOR } W(t-14) \text{ XOR } W(t-16)) \lll 1$, kde
 $(X \lll r)$ je cyklický posun 32bitového slova X o r bitů doleva

12.3.3. Hlavní smyčka

Hlavní smyčka kompresní funkce má 80 rund, v každé se použije jiná funkce f_t , $0 \leq t \leq 79$ používající logické bitové operace OR, AND, NOT a XOR, :

$$f_t(B,C,D) = (B \text{ AND } C) \text{ OR } ((\text{NOT } B) \text{ AND } D), \quad 0 \leq t \leq 19,$$

$$f_t(B,C,D) = B \text{ XOR } C \text{ XOR } D, \quad 20 \leq t \leq 39,$$

$$f_t(B,C,D) = (B \text{ AND } C) \text{ OR } (B \text{ AND } D) \text{ OR } (C \text{ AND } D), \quad 40 \leq t \leq 59,$$

$$f_t(B,C,D) = B \text{ XOR } C \text{ XOR } D, \quad 60 \leq t \leq 79.$$

Konstanty K_0 až K_{79} :

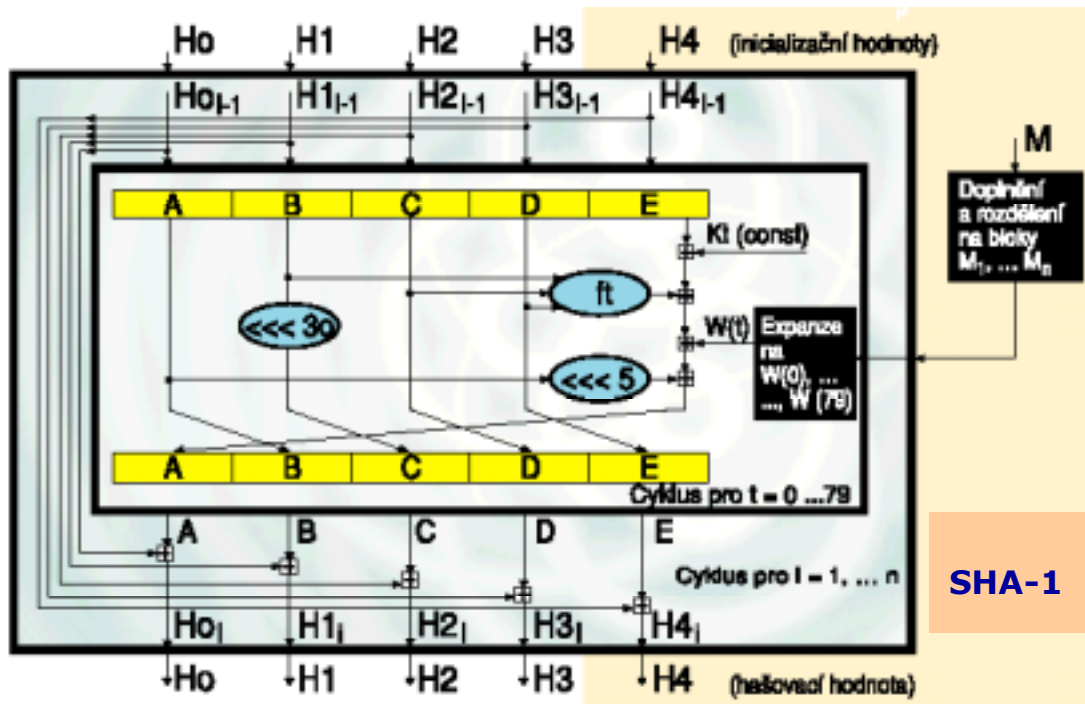
$$K_t = 5A827999, \quad 0 \leq t \leq 19,$$

$$K_t = 6ED9EBA1, \quad 20 \leq t \leq 39,$$

$$K_t = 8F1BBCDC, \quad 40 \leq t \leq 59,$$

$$K_t = CA62C1D6, \quad 60 \leq t \leq 79.$$

Před zpracováním bloku $M(1)$ se H_0 až H_4 nastaví na inicializační hodnoty $H_0 = 67452301$, $H_1 = \text{EFCDAB89}$, $H_2 = 98BADCFE$, $H_3 = 10325476$, $H_4 = \text{C3D2E1F0}$.



Obr.: SHA-1 [30]

12.4. Kolize

Vezměme si příklad hašovací funkce SHA-1. Vstupem může být libovolný řetězec délky až L bitů, $L = 2^{64} - 1$, tedy můžeme hašovat celkem $2^0 + 2^1 + \dots + 2^L = 2^{L+1} - 1$ zpráv. Naproti tomu je pouze 2^{160} hašových kódů. Při hašování dochází ke ztrátě informace - na jeden hašový kód v průměru připadá ohromná množina $(2^{L+1} - 1)/2^{160}$ zpráv. **Kolizí** máme na mysli právě situace, kdy dvě různé zprávy vedou na stejnou haš. Z principu definice hašovací funkce tak vyplývá existence neuvěřitelného množství kolizí, ale stejně tak z její definice vyplývá, že nalezení těchto kolizí je pro nás příliš těžká a výpočetně neproveditelná úloha.

12.5. Narozeninový paradox

Jestliže kolize zákonitě existují, položme si otázku, jak velká musí být množina náhodných zpráv, aby v ní existovaly s nemalou pravděpodobností dvě různé zprávy se stejnou haší, tj. aby nastala kolize. Tuto otázku řeší tzv. narozeninový paradox, od něhož se odvozuje bezpečnost hašovacích funkcí. Říká, kolik zpráv je nutno zhašovat, aby s cca 50% pravděpodobností nastala kolize v množině vzniklých hašovacích kódů. Pro 160bitový hašový kód bychom očekávali $1/2 * 2^{160}$ zpráv, paradoxně je to pouhých 2^{80} zpráv.

Tvrzení: Mějme množinu M n různých koulí a provedme výběr k koulí po jedné s vracením do množiny M . Potom pravděpodobnost, že se ve výběru objeví alespoň jedna koule více než jednou, je $P(n, k) = 1 - (n(n-1) \dots (n-k+1)/n^k)$. Pro $k = O(n^{1/2})$ a n jdoucí do nekonečna je $P(n, k)$ rovno přibližně $1 - \exp(-k^2/2n)$.

Důsledek: Pro n velké se ve výběru $k = (2n \ln 2)^{1/2}$ prvků z M s cca 50% pravděpodobností naleznou dva prvky shodné.

Důsledek pro kolize hašovacích funkcí. Obecně pro hašovací funkce s m bitovým hašovým kódem postačí zhašovat $2^{m/2}$ zpráv, abychom s přibližně 50% pravděpodobností našli kolizi.

Paradoxnost. Máme $P(365, 23) = 0.507$, $P(365, 30) = 0.706$. Pro čísla $n = 365$ a $k = 23$ interpretujeme tvrzení tak, že skupina náhodně vybraných 23 lidí postačí k tomu, aby se mezi

nimi s cca 50% pravděpodobností našla dvojice, slavící narozeniny tentýž den. U skupiny 30 lidí je pravděpodobnost už 70%. Tvrzení se zdá paradoxní protože, ač je vyřčeno jinak, obvykle ho vnímáme ve smyslu "kolik lidí je potřeba, aby se k danému člověku našel jiný, slavící narozeniny ve stejný den". V této podbízející se interpretaci hledáme jednu konkrétní narozeniny, nikoli "jakékoliv shodné" narozeniny.

12.6. Kryptografické využití hašovacích funkcí

12.6.1. Standardy a protokoly symetrické a asymetrické kryptografie

Hašovací funkce se využívají se jak v symetrické, tak asymetrické kryptografii a jsou součástí moderních standardů a protokolů, například v SSL/TLS, PKCS, v autentizačních schématech, zabezpečení integrity dat, v digitálních podpisech apod. Jejich odnoží jsou tzv. klíčované hašové autentizační kódy HMAC (Keyed-Hash Message Authentication Code), viz dále.

12.6.2. Digitální otisk dat

Hašovací funkce vytváří z jakkoliv velkých dat de facto jejich identifikátor, neboť hašový kód díky bezkoliznosti "jednoznačně" identifikuje tato vstupní data. Nejsme totiž schopni nalézt jinou zprávu (jiná data), která by měla stejnou haš. Můžeme proto hovořit o tom, že jde o **digitální otisk dat (digital fingerprint)** nebo **výtah zprávy (message digest)**. Jakákoli data lze tedy identifikovat digitálním otiskem mající řádově pár set bitů. V řadě zemí byly digitální otisky dat z hlediska identifikace dat legislativně postaveny de facto na roveň identifikace lidí pomocí otisků prstů. Je proto obvyklé, že v kryptosystémech digitálního podpisu se nepodepisují vlastní data, ale pouze jejich hašové kódy. To je výhodné, protože podepisované zprávy nebo soubory dat mohou být velmi dlouhé, zatímco na podpis haše postačí jedna podepisovací operace. Výjimkou jsou tzv. *kryptosystémy digitálního podpisu s obnovou zprávy*, kdy zpráva, která se podepisuje, bývá většinou velmi krátká, a systém ji podepisuje přímo.

12.6.3. Kontrola integrity

Hašovací kódy se mohou díky výše uvedené vlastnosti použít přímo ke **kontrole integrity** přenášených zpráv podobně jako se využívají zabezpečovací kódy CRC (Cyclic Redundancy Check). Pokud se totiž zpráva naruší, změní se i její hašovací kód. Neporušenost zprávy můžeme proto kontrolovat neporušeností jejího hašovacího kódu.

12.6.4. Porovnání rozsáhlých databází

Uveďme si příklad banky, která ukládá všechna data ze všech účtů klientů do databázového systému, který je on-line zálohován, takže se vyskytuje současně na třech geograficky vzdálených místech. V určitý okamžik je nutné zjistit, zda jsou tyto databáze opravdu totožné. Mohli bychom například ze všech tří databází číst jednotlivé záznamy a porovnávat je proti sobě. To by znamenalo přenos celých databází komunikačním systémem, což by nemuselo být prakticky vůbec realizovatelné. Místo toho stačí na všech třech místech vypočítat otisky databází nebo jejich částí přenést hodnoty jen několika set bitů do centra. Pokud jsou hodnoty otisků shodné, máme jistotu, že se databáze neliší ani o jeden bit.

12.6.5. Ukládání přihlašovacích hesel

Vlastnost jednocestnosti se využívá ke kontrole hesel v operačních systémech. Hesla se zde neukládají přímo, ale pouze jejich haše. Haše nijak neodhalují hesla, z nichž jsou vypočteny, ale přitom umožňují kontrolu jejich správnosti při přihlašování uživatelů do systému. Pokud útočník přečte haš uloženého hesla nějakého uživatele, není z této hodnoty díky jednocestnosti schopen určit uživatelské heslo. Aby se vyloučil slovníkový útok, kdy si

útočník předvypočítá haše pro často používaná slova a výrazy, používá se tzv. metoda solení. Při ní se vždy při ukládání otisku hesla vygeneruje i náhodný řetězec (sůl), který se poté hašuje dohromady s vlastním heslem. Do databáze se ukládá dvojice (sůl, hash(heslo, sůl)). Útočník by si pak musel předvypočítávat slovník teoreticky pro libovolnou hodnotu soli, což je výpočetně nemožné.

12.6.6. Další příklady použití

Dalšími příklady mohou být kontrola šifrovacích klíčů při dešifrování (haš od hodnoty správného klíče může být uvedena například v hlavičce zašifrovaného souboru), autentizace (prokázání znalosti tajného klíče nepřímo pomocí haše), prokazování autorství (zveřejní se otisk dokumentu, dokument až ve vhodný čas později) apod.

12.7. Nejpoužívanější hašovací funkce

Dříve nejpoužívanějšími hašovacími funkcemi byly MD2, MD4, MD5 [31], které mají 128bitový hašový kód. Od jejich používání se z bezpečnostních důvodů upouští a kde je to možné, přechází se na SHA-1 se 160bitovým hašovým kódem. Důvodem není jen to, že 128bitový hašový kód je náchylnější na nalezení kolize (složitost 2^{64}), ale zejména to, že u MD2 a MD5 byly nalezeny kolize jejich kompresních funkcí, tj.

$$f(H(i-1), \mathbf{M}(i)) = f(H(i-1), \mathbf{M}(i)')$$

a u MD4 byla nalezena kolize přímo (Dobbertin, 1996, [33]), viz obrázek.

CONTRACT

At the price of **\$176,495** Alf Blowfish
sells his house to Ann Bonidea.

CONTRACT

At the price of **\$276,495** Alf Blowfish
sells his house to Ann Bonidea.

Zprávy, které mají stejný hašový kód MD4

Obr.: Kolize u MD4 [33]

Nyní je nejpoužívanější hašovací funkcí SHA-1 se 160bitovým hašovým kódem, která je považována za bezpečnou a dosud u ní nebyly nalezeny žádné slabiny. K nalezení kolize je potřeba 2^{80} zpráv, což může být pro některé bezpečnostně náročné aplikace málo.

12.8. Hašovací funkce SHA-256, 384, 512 a 224

Z důvodu zvýšení odolnosti vůči kolizím je od 1. února 2003 k dispozici nová trojice hašovacích funkcí SHA-256, SHA-384 a SHA-512 [29] a od února 2004 SHA-224 (dodatek [29]). Tyto funkce přichází se zvýšením délky hašového kódu na 256, 384 a 512 bitů (SHA-224 má 224 bitový hašový kód), což odpovídá složitosti 2^{128} , 2^{192} a 2^{256} pro nalezení kolizí narozdíl od SHA-1, což odpovídá složitosti útoku hrubou silou na tři délky klíčů, které nabízí standard AES.

Pokud se týká konstrukce nových funkcí, jsou velmi podobné SHA-1 a používají stejné principy, pracují však se složitějšími funkcemi a širšími vstupy. Podrobnosti lze nalézt v uvedených standardech. Jejich cílem bylo poskytnout větší odolnost proti kolizi a nabídnout odpovídající bezpečnost jako klíče pro AES.

13. Klíčovaný hašový autentizační kód - HMAC

Klíčované hašové autentizační kódy zpráv HMAC zpracovávají hašováním nejen zprávu M, ale spolu s ní i nějaký tajný klíč K. Jsou proto podobné autentizačnímu kódu zprávy MAC, ale místo blokové šifry využívají hašovací funkci. Používají se jak k nepadělatelnému zabezpečení zpráv, tak k autentizaci (prokazováním znalosti tajného klíče K). Klíčovaný hašový autentizační kód je obecná konstrukce, která využívá obecnou hašovací funkci. Podle toho, jakou hašovací funkci používá konkrétně, označuje se výsledek, například HMAC-SHA-1(M, K) používá SHA-1, M je zpráva a K je tajný klíč.

13.1. Obecná definice HMAC

HMAC je definován ve standardu FIPS PUB 198 [32], kde je o něco obecněji popsán, než v RFC 2104 a ANSI X9.71. Jeho definice závisí na tom, kolik bajtů (B) má blok kompresní funkce. Například u SHA-1 je B = 64, u SHA-384 a SHA-512 je B = 128.

Definujeme konstantní řetězce *ipad* jako řetězec B bajtů s hodnotou 0x36 a *opad* jako řetězec B bajtů s hodnotou 0x5C. Klíč K doplníme bajty 0x00 do délky B a definujeme hodnotu HMAC jako

$$\text{HMAC-H}(M, K) = H((K \text{ xor } opad) \parallel H((K \text{ xor } ipad) \parallel M)),$$
 kde \parallel označuje zřetězení.

13.1.1. Nepadělatelný integritní kód

Zabezpečovací kód HMAC-SHA-1(M, K), pokud je připojen za zprávu M, detekuje neúmyslnou chybu při jejím přenosu. Případnému útočníkovi také zabraňuje změnit zprávu a současně změnit HMAC, protože bez znalosti klíče K nelze nový HMAC vypočítat. HMAC může být proto chápán jako nepadělatelný integritní kód, který samotná haš neposkytuje.

13.1.2. Autentizace

HMAC může být použit jako průkaz znalosti tajného sdíleného tajemství (K). Princip průkazu je tento. Dotazovatel odešle nějakou náhodnou výzvu (řetězec) *challenge* a od prokazovatele obdrží odpověď $response = \text{HMAC}(challenge, K)$. Nyní ví, že prokazovatel zná hodnotu tajného klíče K. Přitom případný útočník na komunikačním kanálu z hodnoty *response* klíč K nemůže odvodit.

14. Literatura

- [25] Special Publication SP800-38A: *Recommendation for Block Cipher Modes of Operation - Methods and Techniques*, NIST, <http://csrc.nist.gov/>
- [26] W. Diffie, M. Hellman, *Privacy and Authentication: An Introduction to cryptography*, Proceedings of the IEEE, 67 (1979), pp. 397 - 427
- [27] *PKCS#5 v2.0: Password-Based Cryptography Standard*, RSA Labs, March 25, 1999
- [28] Special Publication SP800-38C: *Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality*, <http://csrc.nist.gov/>
- [29] *FIPS PUB 180-2: Secure Hash Standard*, NIST, <http://csrc.nist.gov/>
- [30] V. Klíma, *Hašovací funkce a kódy: Výživná haše*, Chip 3/1999, str. 40 - 43, <http://cryptography.hyperlink.cz>
- [31] *MD2, MD4, MD5: RFC 1319, 1320, 1321*, <http://www.rfc-editor.org/>
- [32] *FIPS PUB 198: HMAC*, NIST, <http://csrc.nist.gov/CryptoToolkit/tkhash.html>, viz též RFC 2104, <http://www.rfc-editor.org/>
- [33] H. Dobbertin, *Cryptanalysis of MD4*, FSE 1996, pp. 53 - 69