INFORMATIKA

Téměř dokonalá šifra

PETR VOBORNÍK

Přírodovědecká fakulta, Univerzita Hradec Králové

Princip dokonalé šifry je znám již téměř celé století. Podmínky, které musí být při jejím používání dodrženy, však praktické nasazení stále komplikují. Šifra je tedy vhodná tam, kde se vyžaduje extrémně vysoké utajení a kde nevadí mimořádně vysoké náklady spojené s výrobou a distribucí klíčů. [1]

V tomto článku shrneme princip dokonalé šifry a na jeho základě se s využitím moderních hashovacích algoritmů pokusíme navrhnout jednu z možných modifikací práce s klíči tak, aby byly odstraněny zásadní faktory znemožňující praktické používání šifry.

Základní pojmy

V následujícím textu budou používány známé výrazy, které ovšem mohou mít v různých kontextech různé významy. Upřesněme si tedy, co je pod jejich označením míněno zde. Ostatní pojmy budou vysvětleny přímo v článku.

Data – zdrojová data, která jsou třeba ochránit šifrováním před jejich přečtením a zneužitím třetími osobami.

Zpráva – je soubor informací zasílaných mezi dvěma komunikujícími stranami. Skládá se z dat (zašifrovaných) a dalších údajů, jako jsou identifikátor odesílatele, určení příjemce, datum a čas odeslání, apod.

Klíč – tajná sada dat, s jejichž pomocí budou výpočetní operací zdrojová data šifrována a následně dešifrována.

Heslo – slovo, fráze nebo kombinace znaků, které umožňuje zašifrovaná data dešifrovat a naopak. Tzn. heslo se buď použije přímo jako **klíč**, nebo se klíč vygeneruje na jeho základě, což bude i případ tohoto článku.

Dokonalá šifra

Roku 1917 *Američan Gilbert Sandford Vernam* (1890–1960) zkonstruoval zajímavý šifrovací systém. Ten vycházel z Vigenerovy šifry z roku 1586 [2], ovšem obsahoval několik zásadních změn údajně inspirovaných německým kryptologem Hermannem z roku 1892 [3]. Systém byl založený na použití jednorázového náhodně vygenerovaného hesla, které má stejnou délku jako zpráva sama. Fakt, že je Vernamova šifra, zvaná též „one-time pad“ (jednorázová tabulka), absolutně bezpečným (neprolomitelným) šifrovacím systémem matematicky prokázal v roce 1949 *Claude Elwood Shannon* (1916–2001) [4]. Této dokonalé nerozluštitelnosti šifry lze dosáhnout ovšem pouze při dodržení tří přísných podmínek spolehlivosti:

1. Klíč musí být stejně dlouhý jako přenášená zpráva.
2. Klíč musí být dokonale náhodný.
3. Klíč nesmí být použit opakovaně. [5]

Jsou-li podmínky spolehlivosti dodrženy, je tato šifra zcela bezpečná proti jakémukoli pokusu o prolomení, včetně útoku hrubou silou. Není-li totiž znám správný klíč, neexistuje způsob proveditelný ani v libovolně dlouhém časovém horizontu, jak zprávu rozluštit. Je sice možné najít takový klíč, který by dokázal zašifrovaná data převést na srozumitelný text téže délky, ovšem takovýchto klíčů lze nalézt tolik, že tato data mohou v podstatě dávat smysl libovolný, přičemž nelze odhadnout, která z takovýchto zpráv byla tou odesílanou.

Vernam ve svém následném patentu [6] navrhl i jednoduchý přístroj, který pracoval již s 31 znaky – 26 písmen, mezera, znaky návrat vozíku (CR) a posun o řádek (LF) a signály „následují číslice“ a „následují písmena“, což pokrývalo 5 bitů, jež přístroj šifroval náhodným klíčem pomocí operace XOR.

Operace XOR

Logická operace exkluzivní disjunkce, v originále „exkluzive or“, zkráceně XOR se značí takto: $\oplus$. Výsledkem je 0, pokud jsou obě vstupní hodnoty shodné, a 1, jsou-li rozdílné (viz tab. 1).

$\mathtt{A} \oplus \mathtt{B} = \mathtt{C}$

A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

Tab. 1: Stavová tabulka hodnot operace XOR

XOR je komutativní operace, tj. nezáleží na pořadí jednotlivých hodnot s nimiž se operace XOR provádí.

$$(\mathtt{A} \oplus \mathtt{B} = \mathtt{C}) \Leftrightarrow (\mathtt{B} \oplus \mathtt{A} = \mathtt{C})$$

Poznámka: Místo operace XOR lze také použít funkci modulo 2 ze součtu obou hodnot:

$$(\mathtt{A} \oplus \mathtt{B} = \mathtt{C}) \Leftrightarrow ((\mathtt{A} + \mathtt{B}) \bmod 2 = \mathtt{C})$$

Z výsledku operace XOR lze následně další operací XOR s jednou z výchozích hodnot dopočítat tu druhou.

$$(\mathtt{A} \oplus \mathtt{B} = \mathtt{C}) \Leftrightarrow (\mathtt{C} \oplus \mathtt{A} = \mathtt{B}) \Leftrightarrow (\mathtt{C} \oplus \mathtt{B} = \mathtt{A})$$

Pokud tedy i-tý datový bit $\mathtt{D}_i$ zašifrujeme operací XOR s i-tým bitem klíče $\mathtt{K}_i$ do zašifrovaného znění $\mathtt{C}_i$, pak z $\mathtt{C}_i$ zpětně dešifrujeme výchozí datový bit $\mathtt{D}_i$ opětovným provedením operace XOR s i-tým bitem klíče $\mathtt{K}_i$.

$$\textit{Zašifrování: } \mathtt{D}_i \oplus \mathtt{K}_i = \mathtt{C}_i \qquad \textit{Dešifrování: } \mathtt{C}_i \oplus \mathtt{K}_i = \mathtt{D}_i$$

Je-li zároveň klíč náhodný, pak pravděpodobnost, že $\mathtt{K}_i = 0$, je stejná jako pravděpodobnost, že $\mathtt{K}_i = 1$, je rovna $1/2$ (tj. 2^{-1}), čili 50 %. Stejně tak je tomu i u zašifrované hodnoty $\mathtt{C}_i$. Není-li tedy znám klíč $\mathtt{K}_i$, je pravděpodobnost „uhodnutí“ správné hodnoty přesně poloviční. Pro celý znak skládající se z 8 bitů (1 byte) je pak za těchto okolností pravděpodobnost určení správného znaku již jen 2^{-8} (tj. $1/256$), čili cca 0,4 %.

Princip Vernamovy šifry

Těmto písmenům A až Z se přiřadila čísla 0 až 25 a pro i-tý znak utajovaných dat $\mathtt{D}_i$ s klíčem $\mathtt{K}_i$ (klíč se také skládal pouze ze znaků této abecedy) se znak šifrovaných dat $\mathtt{C}_i$ určil následujícím způsobem [2]:

$$\mathtt{C}_i = (\mathtt{D}_i + \mathtt{K}_i) \bmod 26$$

Dešifrování pak proběhlo opačnou operací:

$$\mathtt{D}_i = (26 + \mathtt{C}_i - \mathtt{K}_i) \bmod 26$$

Například slovo „AHOJ“ by se s použitím náhodného klíče „SMHZ“ šifrovalo jako je tomu v tab. 2.

	Znaky				Čísla			
Data	A	H	O	J	0	7	14	9
Klíč	S	M	H	Z	18	12	7	25
Zašifrovaná data	S	T	V	I	18	19	21	8
Dešifrovaná data	A	H	O	J	0	7	14	9

Tab. 2: Ukázka postupu šifrování znaků původní Vernamovou širou

Vylepšená verze této šifry pracuje s binární reprezentací dat. Jednotlivé bity dat převedených do binární podoby jsou šifrovány operací XOR s jednotlivými bity klíče. Výhodou byla možnost strojového zpracování této šifry. Vernam ve své době používal vlastní převodní tabulku pro znaky základní abecedy do binární formy, kterou místo hodnot 0 a 1 označoval znaky + a −. [6] V současnosti, kdy jsou data uchovávána a přenášena v elektronické formě bitů standardně, je situace pro podobný styl šifrování ještě daleko jednodušší.

Stejná data jako v předchozím případě by při binárním kódování (pro převod znaků na bity je použita standardní ASCII tabulka znaků) vypadala tak, jak ukazuje tab. 3.

	Znaky				Bity			
Data	A	H	O	J	01000001	01001000	01001111	01001010
Klíč	S	M	H	Z	01010011	01001101	01001000	01011100
Zašifrovaná data	18	5	7	22	00010010	00000101	00000111	00010110
Dešifrovaná data	A	H	O	J	01000001	01001000	01001111	01001010

Tab. 3: Ukázka šifrování dat Vernamovou šifrou na binární úrovni

Zašifrovaná data v příkladu jsou uvedena ve formě indexu znaku v ASCII tabulce, jelikož tyto jsou v textovém formátu nezobrazitelné. V tomto případě by zároveň náhodný klíč neměl využívat pouze byty znaků písmen, ale vybírat z celé tabulky všech 256 znaků, resp. generátor klíče by měl pracovat na úrovni bitů (náhodně volit sekvence 0 a 1), a na výsledné znaky vůbec nehledět.

Důsledky porušení podmínek spolehlivosti

Porušení podmínek spolehlivosti vede k nedostatečné bezpečnosti šifry a umožňuje její prolomení. Konkrétně nedodržení každé jednotlivé podmínky má následující důsledky.

V případě *opakovaného použití klíče* je možné tento klíč snadno určit pouze ze znalosti dvou zachycených zpráv šifrovaných týmž klíčem. Platí totiž následující vztah:

$$\mathtt{D1}_i \oplus \mathtt{K}_i = \mathtt{C1}_i \qquad \mathtt{D2}_i \oplus \mathtt{K}_i = \mathtt{D2}_i \qquad \mathtt{C1}_i \oplus \mathtt{C2}_i = \mathtt{D1}_i \oplus \mathtt{D2}_i$$

kde $\mathtt{D1}_i$ je i-tý znak 1. dat, $\mathtt{D2}_i$ je i-tý znak 2. dat, $\mathtt{K}_i$ je i-tý znak klíče, $\mathtt{C1}_i$ je i-tý znak zašifrovaného znění 1. zprávy a $\mathtt{C2}_i$ je i-tý znak zašifrovaného znění 2. zprávy. Výsledkem operace XOR dvou zašifrovaných dat je tedy XOR dvou původních dat. Tím dojde k odstranění veškeré náhodnosti klíče a z výsledku lze jednoduchou statistickou kryptoanalýzou získat oboje původní data a tím pádem i klíč. [7]

$$\mathtt{K}_i = \mathtt{C1}_i \oplus \mathtt{D1}_i = \mathtt{C2}_i \oplus \mathtt{D2}_i$$

Díky tomu lze následně každou další zprávu zašifrovanou týmž klíčem dešifrovat již v reálném čase, bez nutnosti dalších kryptoanalýz. Po prvním použití každého klíče je tedy třeba jej celý bezpečně „zničit“, jak na straně příjemce, tak na straně odesílatele.

Pokud by klíč *nebyl stejně dlouhý jako přenášená zpráva*, muselo by dojít k jeho opakování pro šifrování částí dat, které nepokryl. To by mělo za následek týž efekt jako opakované použití klíče. V případě že by útočník znal některou z částí dat, získal by tak zpětným provedením operace XOR část nebo dokonce celý klíč a mohl jej použít na zbylé části dat, jež nezná.

Znalost části dat útočníkem je celkem běžný fakt. V dopisech bývá na začátku obvykle uvedeno „Dobrý den“, „Ahoj“ apod., na konci zase podpis odesílatele. Při posílání binárních dat je situace ještě jednodušší, protože většina formátů souborů má vlastní hlavičku, která je vždy shodná (JPEG, ZIP, WAV, DOC, ...) nebo je alespoň z konečné množiny možností. Struktura dokumentů textových editorů (RTF, XML, HTML, ...) pak navíc opakovaně obsahuje známé formátovací sekvence znaků, které se dají frekvenční analýzou snadno detekovat.

Předpoklad *dokonalé náhodnosti celého klíče* stejně jako jeho dostatečná délka zaručuje, že každý jednotlivý znak (bit) dat je zašifrován zcela nezávisle na ostatních znacích. Znalost jakékoli části dat tedy útočníkovi z výpočetního hlediska neprozradí nic o kterémkoli jiném jemu neznámém znaku dat ani klíče.

Pro dokonalou bezpečnost šifry nelze použít ani pseudonáhodné hodnoty. Ty jsou totiž generovány dle určitého algoritmu a jejich vygenerování je tak při dodržení stejných podmínek zopakovatelné. Data jsou pak rozluštitelná v konečném čase, resp. lze nalézt takový klíč, který převede zašifrovanou zprávu na srozumitelná data a zároveň u něho bude možné prokázat vztah mezi jeho jednotlivými částmi nebo k nějaké výchozí hodnotě (seedu[2]) a tak identifikovat, která z možných rozluštění zpráv je ta pravá. Pro generování klíče je nejvhodnější užití fyzikálních metod, např. radioaktivity, o níž je prokázáno, že její charakter je skutečně náhodný [1].

Kvantová kryptografie

Kvantová kryptografie využívá bezpečného komunikačního kanálu (optického vlákna) mezi dvěma komunikujícími stranami. Pro přenos jednotlivých bitů jsou použity ve smluveném směru polarizované fotony. Ty totiž nelze odposlouchávat jako klasický elektrický signál, jehož intenzitu je možné změřit, aniž by byl tok dat narušen. Foton je dále nedělitelná a neklonovatelná částice a jakákoli interakce s ním jej zásadně ovlivní. Případný odposlech lze tedy snadno odhalit [8].

Aby se předešlo zachycení dat případným útočníkem odposlouchávajícím komunikaci (tzv. *Man in the middle*), je nejprve poslán tímto kanálem klíč, který splňuje požadavky spolehlivosti na jeho délku a náhodnost. Pokud přenos klíče proběhne v pořádku (nedošlo k jeho odposlechu), jsou teprve pak odeslána data zašifrovaná tímto klíčem. V opačném případě je klíč „zapomenut“ a zkusí se poslat jiný. Přenos dat již poté ani nemusí probíhat přes zabezpečený kanál, jelikož ta jsou bez klíče, při dodržení požadavků spolehlivosti, absolutně nedešifrovatelná [5]. Tento princip přináší možnost zcela bezpečné komunikace, ovšem vyžaduje přímé spojení nepřerušovaným optickým kabelem mezi oběma stranami, což je podmínka splnitelná jen v některých výjimečných případech. Při komunikaci prostřednictvím veřejné sítě internet tedy globálně použít nelze.

Jinou možností bezpečného přenosu klíče je osobní předání datového média (např. CD) obsahujícího data klíče pro budoucí použití. Podobným způsobem je například zabezpečena horká linka spojující prezidenty Ruska a USA [1].

[2]Seed je výchozí hodnota generátoru pseudonáhodných hodnot, v němž je každá následující hodnota odvozena od hodnoty předchozího kroku. Při zadání téhož seedu lze tudíž zopakovat vygenerování stejné sady pseudonáhodných hodnot.

Dlouhý a náhodný klíč

Podmínky spolehlivosti zaručují šifře nerozluštitelnost, zároveň však také komplikují její užívání. Konkrétně požadavek dokonalé náhodnosti klíče znesnadňuje jeho automatické softwarové generování. Také nutnost, aby jeho délka byla shodná s délkou šifrovaných dat, přináší (pomineme-li kvantovou kryptografii) týž problém, jako přenos dat samotných, na čemž se podílí i jednorázovost každého klíče. Může tedy být žádoucí, byť za cenu ztráty absolutní neprolomitelnosti šifry, aby klíč resp. heslo mohlo být kratší, ne zcela náhodné (zapamatovatelné) a opakovaně použitelné.

Jak již bylo uvedeno, klíč složený z pseudonáhodných hodnot nezabrání v rozluštění zašifrovaných dat v konečném čase. Bude-li však tento klíč „kvalitně náhodný“ a zároveň splňovat ostatní podmínky spolehlivosti, zůstane vyloučena možnost použití jakýchkoli výpočetních a statistických kryptoanalýz, s výjimkou útoku hrubou silou. Ten může útočník například použít na určení klíčů, které zašifrovaným datům (nebo jejich částem) dávají smysl, a následně k hledání souvztažnosti mezi jednotlivými částmi klíče. Účinnější formou útoku hrubou silou by pak bylo, v případě znalosti algoritmu generátoru pseudonáhodných hodnot pro klíče, určení výchozí hodnoty generátoru – seedu, resp. hesla. Bude-li toto dobře zvoleno, lze data rozluštit pouze „uhodnutím hesla“ s použitím „hrubé síly“ a onen konečný čas rozluštění dat tak může být nereálně dlouhý.

V případě, že budou pro účely šifrování přínosné výše zmíněné výhody ohledně hesla (krátké, nenáhodné a opakovatelné) a zároveň nebudou nepřekonatelnou překážkou uvedená omezení dokonalosti šifry (při „uhodnutí“ hesla budou data dešifrovatelná), pak již zbývá jen vytvořit algoritmus, který z krátkého hesla dokáže opakovaně vytvořit libovolně dlouhý klíč, jenž bude statisticky prokazatelně náhodný v rovnoměrném rozdělení. To znamená, aby všechny hodnoty v rozsahu bytu (rozsah bytu je 0–255, tedy 256 (2^8) možných hodnot) byly generovány se stejnou pravděpodobností, resp. generovaná sekvence bitů byla sama o sobě náhodná. Popsané vlastnosti přímo zapadají do definice hash funkce a při jejím vhodném užití lze s její pomocí docílit veškerých požadovaných vlastností klíče.

Hash

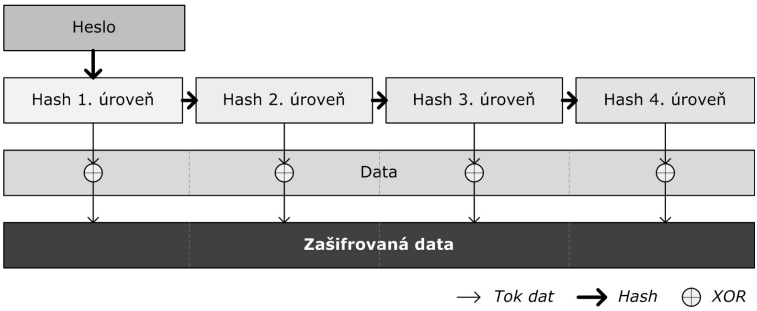
Hash je jednosměrná (ireverzibilní) výpočetně efektivní funkce mapující binární řetězce libovolné délky na řetězce pevné délky, tzv. hash-hodnoty. Základní myšlenkou je, že hash-hodnota slouží jako kompaktní zástupce vstupního řetězce. Při kryptografickém použití, je hash funkce `H`

zvolena tak, že je výpočetně nemožné nalézt dva různé vstupy, jejichž hash-hodnota by byla shodná, (tj. nelze nalézt `X` a `Y` takové, aby platilo $(\mathtt{H(X)} = \mathtt{H(Y)}) \wedge (\mathtt{X} \neq \mathtt{Y})$), a zároveň je také výpočetně nemožné určit vstup `X` pro danou hash-hodnotu `Y` (tj. $\mathtt{H(X)} = \mathtt{Y}$). Pravděpodobnost, že n-bitová hash-hodnota (např. $n = 128$ nebo 160) náhodně vybraného řetězce bude mít konkrétní n-bitovou hash-hodnotu je tedy 2^{-n} [7].

Statistické testy náhodnosti hash-kódu generovaného užitím algoritmu SHA-1[3] dle [9], [10] prokázaly, že jím generovaná sekvence bitů vyhovuje ze statistického hlediska podmínkám rovnoměrného náhodného rozdělení. Jiné hashovaní algoritmy (např. MD5, SHA-256, SHA-512, atd. nebo připravovaný SHA-3 [11]) by samozřejmě dle své definice měly mít tytéž vlastnosti, což je možné ověřit například dle postupů uvedených v [7] pomocí software popsaného v [9].

Heslo a klíč, délka klíče

Pomocí hash algoritmu lze tedy z libovolného hesla vytvořit klíč vyhovující podmínce náhodnosti a neumožňující zpětný výpočet hesla. Jeho délka je ovšem předem určena na konstantní počet bitů dle konkrétního užitého algoritmu. Zapotřebí je však klíč mnohem delší, než je hash-kód. Jednou z možností je použít jako klíč víceúrovňový hash. V tomto případě by byl hash původního hesla použit na zašifrování prvního bloku dat a následně posloužit jako vstupní hodnota pro vygenerování nového hash-kódu (hash 2. úrovně). Jím by se opět zašifroval další blok dat a znovu by z něho byl vygenerován hash 3. úrovně jako klíč pro další blok dat a tak dále, až by byla pokryta celá datová zpráva (viz obr. 1).

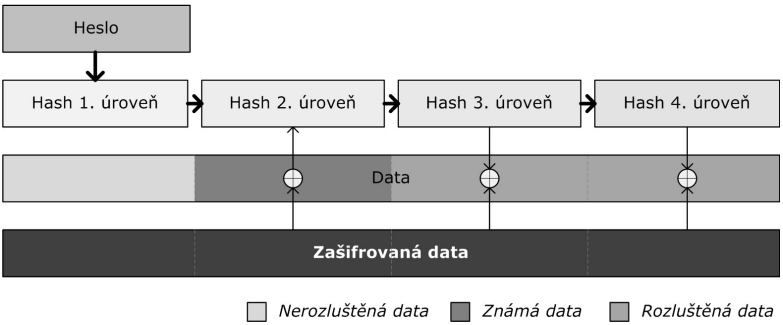


Obr. 1: Schéma šifrování dat operací XOR, kde klíč tvoří prostý víceúrovňový hash hesla

[3]SHA-1 – Secure Hash Algorithm, vracející hash-kód o délce 160 bitů, který byl navržen institutem NIST pro americké vládní aplikace [7].

Zamezení útoku při znalosti části dat

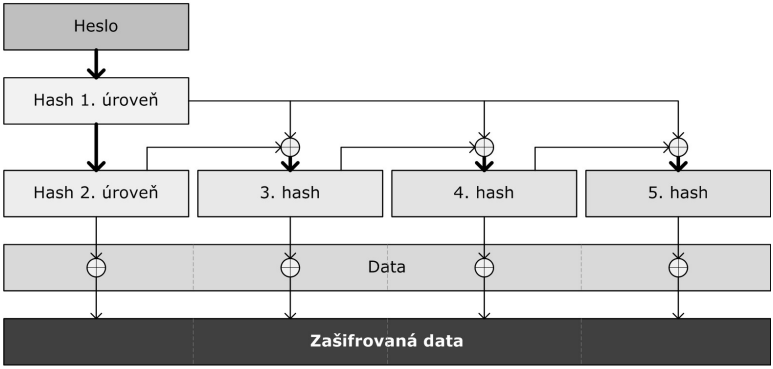
V uvedeném případě je ovšem útočník, který zná část dat, schopen rozluštit i jejich další neznámé části. Pokud by například znal data zašifrovaná hashem 2. úrovně, stačilo by mu provést operaci XOR mezi těmito a zašifrovanými daty a získal by část klíče (hash 2. úrovně). Z něho by sice nedokázal dopočítat hash 1. úrovně ani heslo, ovšem mohl by vygenerovat hash 3. úrovně, z něho pak 4. úrovně atd. Díky tomu by byl schopen dešifrovat data od bloku, jehož obsah znal, či např. slovníkovým útokem odhalil (viz obr. 2).



Obr. 2: Schéma rozluštění části dat zašifrovaných pomocí klíče z prostého víceúrovňového hashe hesla

Jednou z možností, jak takovémuto útoku zabránit, je modifikovat před generováním hash-kódu každé další úrovně vstup hash funkce (předchozí úroveň hashe) způsobem, který útočník nedokáže zopakovat, ovšem dešifrovací proces znalý správného prvotního hesla ano. Tato modifikace tedy musí přímo vycházet a záviset na tomto heslu. Lze například k hash-kódu hesla každé úrovně před generováním hashe další úrovně přičíst heslo samotné, avšak mnohem účinnější, bezpečnější a výpočetně efektnější je hash zkombinovat s jiným hashem. Oba totiž obsahují pseudonáhodné znaky z celé škály bytového rozsahu a také mají stejnou délku. Díky tomu lze opět využít operaci XOR.

Vzniklý blok bytů poslouží pouze jako vstup pro vytvoření hashe následující úrovně a sám o sobě nebude nikde použit. Pro tento účel ideálně poslouží hash hesla 1. úrovně, který by v tomto případě neměl být sám o sobě použit pro šifrování žádného z bloků dat a sloužil pouze pro kombinování s hashi vyšších úrovní (viz obr. 3).



Obr. 3: Schéma šifrování dat pomocí klíče z kombinovaného víceúrovňového hashe hesla

Pokud tedy útočník bude znát určitou část dat, dokáže sice rozluštit klíč, kterým byla tato část zašifrována, ale již nedokáže určit klíč pro následující (a samozřejmě ani předchozí) blok dat. K tomu by potřeboval znát buď hash hesla 1. úrovně nebo zdroj hashe pro klíč následujícího bloku dat. Oba požadavky by znamenaly určení reverze hashe, což je již dle základní definice této funkce výpočetně nemožné.

Jediný způsob, jak neznámé části dat určit, je „uhodnout“ heslo, popřípadě jeho hash 1. úrovně. Zde již závisí hlavně na „síle“ zvoleného hesla, tj. jak dokáže odolat před slovníkovým útokem a útokem hrubou silou. Pro volbu snadno zapamatovatelných hesel odolávajících těmto druhům útoku existuje řada postupů (např. viz [12]).

Heslo delší než hash-kód nemá u jednorázového použití smysl, jelikož v takovém případě se útočníkovi vyplatí spíše určit 1. hash-kód tohoto hesla, neboť heslo samotné k rozluštění dat nepotřebuje. Pokud by však heslo mělo být používáno opakovaně, útočníkovi se vyplatí hledat spíše heslo než jeho hash-kód, i když jeho určení bude o něco náročnější.

Jedinečný klíč pro každou zprávu

Posledním úskalím je požadavek na jedinečnost klíče pro každou komunikaci (každá data). Pro dosažení tohoto požadavku existuje několik možností. Je-li například souběžně s rychlým datovým potencionálně odposlouchávaným kanálem soustavně otevřen i další zabezpečený, byť třeba pomalý kanál, může být před zasláním každé datové zprávy tímto kanálem zasláno i nové heslo.

Druhou možností je, v případě souvislé komunikace mezi dvěma účastníky, používat stále další a další úrovně původního hesla a nezačínat je generovat vždy znovu od začátku. Klíčem pro šifru pak bude neustále jiný klíč. Nevýhodou ovšem je nezbytnost pamatovat si úroveň hashe, na které komunikace skončila a nemožnost zapojení více účastníků (při architektuře klient–server) do komunikace, aniž by neustále museli zbytečně dopočítávat příslušnou úroveň hashe dosaženou ostatními.

Jinou možností je využití faktu, že na kompletní změnu celého klíče sestávajícího z moha úrovní hashe stačí změna jediného bitu v heslu či hashe 1. úrovně. Při tomto druhu změny může být její popis součástí zprávy obsahující i zašifrovaná data. Může se jednat například o dodatečný textový řetězec, který byl k heslu přičten před výpočtem hashe první úrovně. Takovýto přídavek hesla se nazývá „salt“ a je zároveň dobrou pomůckou proti slovníkovým útokům postaveném na předgenerovaných hashových slovnících. Jeho zveřejnění přitom nijak nesnižuje obtížnost dešifrování dat, neboť pro výpočet klíče je stále nezbytné znát i původní část hesla.

Díky saltu je klíč pro data pokaždé kompletně jiný a ani případné určení jeho části, nebo i klíče celého, v některém z minulých datových přenosů, nesnižuje zabezpečení přenosu dat budoucích, aniž by muselo dojít ke změně hesla. Je pouze třeba zabezpečit, aby byl salt pokaždé jiný, čehož lze například dosáhnout pomocí generátoru tzv. GUID hodnot (*Globally Unique IDentifier*. Náhodně vygenerovaná hodnota se zanedbatelně malou pravděpodobností, že by někde někdy byly vygenerovány dvě stejné hodnoty).

Způsobů jak heslo a salt zkombinovat je nespočet. Například, pokud heslo bude `heslo` a salt `SALT` lze použít tyto způsoby:

- `hesloSALT`
- `SALTheslo`
- `hSeAsLlTo`
- `HASH`(`heslo`) $\oplus$ `HASH`(`SALT`)
- ...

Dlouhodobé uchovávání saltu

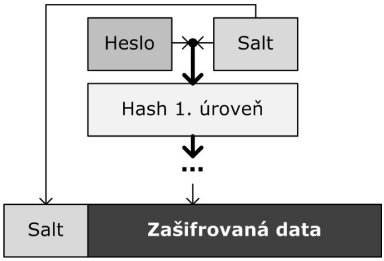
Pokud není zaručeno, že pro každá šifrovaná data bude použito jiné heslo, je pro originalitu každého klíče nezbytné použít salt. Při komunikaci dvou stran může být salt jedním z předávaných parametrů zprávy. V případě použití šifry na dlouhodobě samostatně uchovávané soubory je

třeba salt uložit tak, aby byl při potřebě soubor dešifrovat kdykoli dohledatelný a konkrétnímu souboru přiřaditelný, tedy nejlépe přímo do tohoto souboru.

Ani pozice saltu uloženého v souboru šifrovaných dat nemusí být vždy stejná, tím se také dále znesnadňuje prolomení šifry. Salt totiž nemusí být přidán pouze na začátek nebo konec zašifrovaných dat, ale i na libovolnou pozici. Je také možné salt rozdělit na jednotlivé byty a ty různě mezi data rozmístit. Jejich pozice, případně saltu jako celku, i způsobu rozmístění by pak měla být jednoznačně určitelná na základě hesla, aby ho bylo možné při dešifrování zpětně dohledat a oddělit od dat. Jelikož by salt měl být zcela náhodný, stejně jako zašifrovaná data, nemělo by dojít k jeho identifikaci a pokusu o dopočítávání hesla.

Každopádně nic nebrání použití zpětně nevypočitatelného postupu (rozmísťování saltu na základě hodnot hash-kódu hesla). Také je možné salt před uložením k datům zašifrovat pouze pomocí hashe hesla (náhodné bity zašifrované náhodnými bity bez klíče dešifrovat nelze). V případě uchování saltu u dat ovšem již nejde o šifru $1:1$, kdy jeden bit zdrojových dat je zašifrován právě do jednoho bitu zašifrovaných dat, ale o šifru $1:(1+\texttt{délka saltu})$.

Tento postup také komplikuje blokové zpracování dat. Dešifrovací algoritmus musí nejprve přečíst salt a až po té s jeho pomocí může postupně dešifrovat data. Aby tedy nebylo nezbytné dvojí zpracování dat, je nejvýhodnější salt uložit hned na jejich začátek (viz obr. 4).



Obr. 4: Schéma šifrování dat se zapojením saltu

Při takto uloženém saltu může šifrování i dešifrování dat probíhat obvyklým blokovým i proudovým způsobem. Začátek dat, kde je uložen salt, ovšem musí být přečten vždy a nelze tak dešifrovat pouze určité úseky dat nezávisle na pořadí.

Rychlost

Navržený šifrovací algoritmus tedy přímo staví na specifickém klíči, jenž tvoří víceúrovňový hash, který je navíc v každé úrovni znovu kombinován s hashem 1. úrovně. Výpočet hashe není samozřejmě triviální výpočetní operací, ale komplikovaným algoritmem, který zabírá určitý výpočetní čas. Je tedy zřejmé, že na rychlosti, či spíše pomalosti šifry, bude mít nevětší podíl právě výpočet tohoto klíče. Jelikož šifra dokáže pracovat s libovolným hashovacím algoritmem, pokusili jsme se pro ni zvolit ten nejrychlejší.

Za tímto účelem, bylo provedeno následující srovnání (tab. 4). V něm byly pomocí jednotlivých hashovacích algoritmů (řádky tabulky) výše uvedenou metodou vypočteny víceúrovňové klíče daných délek (sloupce tabulky). Pokus byl opakován vždy třikrát a výsledný průměr (buňky tabulky) je čas potřebný na výpočet každého klíče vyjádřený v milisekundách. Základní vlastnost hash-kódu (ireverzibilita a náhodnost) byla již brána jako dále netestovaná samozřejmost.

Měření bylo prováděno za identických podmínek, tj. na stejném počítači při týchž běžících procesech. Parametry testovacího stoje byly tyto: CPU 2,5 GHz, RAM 4GB, HDD 7 200 RPM, OS Windows 7 64bit. Pro výpočet klíče byly použity algoritmy integrované v programovacím prostředí Microsoft.NET Framework 4.0, jazyk C#, ve kterém byl implementován i následně testovaný šifrovací algoritmus.

	1 MB	2 MB	3 MB	5 MB	10 MB	20 MB	30 MB	50 MB	100 MB
MD5	519	1 034	1 539	2 532	5 060	10 084	15 105	25 206	50 313
SHA-1	430	842	1 272	2 098	4 216	8 410	12 613	21 016	42 006
SHA-256	291	560	890	1 406	2 820	5 682	8 648	14 168	28 523
SHA-384	295	560	862	1 368	2 757	5 549	8 225	13 734	27 379
SHA-512	211	420	655	1 046	2 083	4 221	6 309	10 496	20 874

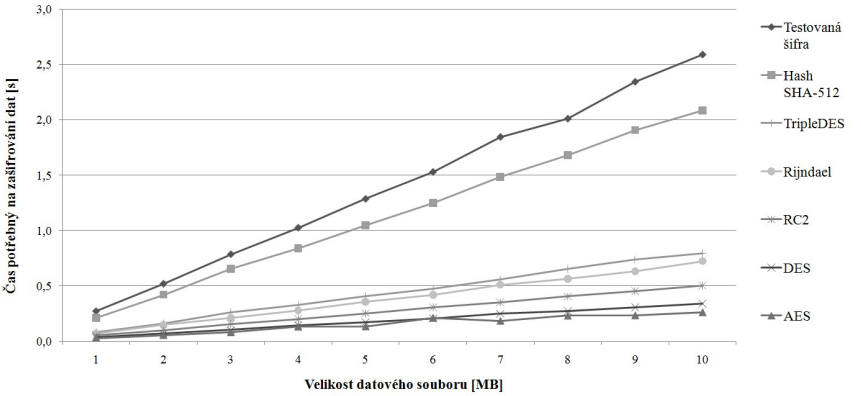
Tab. 4: Porovnání rychlostí výpočtů [ms] víceúrovňového hashe dané délky jednotlivými algoritmy

Z porovnání v tab. 4 plyne, že nejrychlejším z testovaných hashovacích algoritmů je SHA-512. Ten byl tedy následně použit i pro test srovnání rychlostí tohoto a již existujících šifrovacích algoritmů (tab. 5). Porovnání bylo provedeno podobným testem a za stejných podmínek jako srovnání rychlostí hashovacích algoritmů. Tentokrát ovšem již nešlo pouze o výpočet hodnot v rámci paměti počítače, ale zdrojová data byla čtena ze souboru na pevném disku a výsledná zašifrovaná data na disk znovu ukládána. Data byla načítána, zpracována a ukládána proudově, po blocích o velikosti 5 kB.

	1 MB	2 MB	3 MB	5 MB	10 MB	20 MB	30 MB	50 MB	100 MB
AES	28	54	79	133	264	531	818	1 263	2 797
DES	35	68	101	172	340	703	1 030	1 709	3 476
RC2	52	101	153	252	501	1 012	1 523	2 547	5 114
Rijndael	71	148	211	357	723	1 452	2 147	3 581	7 557
TripleDES	82	160	264	406	795	1 620	2 403	4 013	8 300
Testovaná šifra	270	518	785	1 285	2 590	5 113	7 703	12 764	25 825

Tab. 5: Porovnání rychlostí šifrování [ms] datových souborů dané velikosti jednotlivými algoritmy

Porovnání v tab. 5 a na obr. 5 ukazuje, že navržený šifrovací algoritmus je oproti ostatním výrazně pomalejší. Zhruba 81 % tohoto času ovšem zabírá výpočet klíče, byť nejrychlejším z testovaných hashovacích algoritmů SHA-512. Použitím rychlejšího hashovacího algoritmu by tedy mohlo dojít i k výraznému zrychlení této šifry. Tuto vlastnost by mohl přinést připravovaný hashovaní algoritmus SHA-3 (Soutěž o SHA-3 viz `http://csrc.nist.gov/groups/ST/hash/sha-3/`).



Obr. 5: Graf porovnání rychlostí šifrování datových souborů dané velikosti jednotlivými algoritmy. Zahrnuta je i rychlost generování klíče pomocí hashovacího algoritmu SHA-512.

Rychlost šifrování tedy limituje použití při klasickém šifrování v reálném čase, například při on-line komunikaci dvou stran, kde je rychlost spojení jedním z hlavních parametrů. Její využití by tak mohlo být spíše v případech, kdy má úroveň zabezpečení vyšší prioritu, nežli čas potřebný na zašifrování.

Závěr

V článku byl shrnut původní princip Vernamovy dokonalé šifry a na jejím základě navržena modifikace práce s klíči, jejichž správa zatím velmi komplikuje její praktické využití. Při kombinaci s moderními hashovými algoritmy lze šifrovat data pomocí matematicky prokazatelně zpětně nevypočitatelných postupů a přitom i opakovaně používat „jednoduchá“ hesla. Síla šifry je pak vždy přímo úměrná síle zvoleného hesla.

Implementace uvedeného postupu je přitom programově velmi snadná. Rychlost šifrování a dešifrování dat nejvíce závisí na rychlosti výpočtu hash-kódu, tedy na zvolené hash funkci. Připravovaná funkce SHA-3 přitom slibuje mnohem rychlejší výpočet a zároveň i vyšší bezpečnost než ty stávající, byť dosud neprolomené [11].

Navrženou šifru lze, díky své stávající nižší rychlosti, používat pro ochranu přenosu dat přes veřejnou síť internet zatím pouze v případech, kdy nevadí zpomalení potřebné pro šifrování dat. V případě šifrování archivů a souborů pro dlouhodobou úschovu, kde je obvykle přednější jejich bezpečnost před časem potřebným na zašifrování, může tato šifra nalézt své uplatnění již nyní.

Literatura

[1] *Singh, S.*: Kniha kódů a šifer. Dokořán a Argo, Praha, 2009.

[2] *Piper, F., Murphy, S.*: Kryptografie: Průvodce pro každého (překlad). P. Mondschein – Dokořán, Praha, 2006.

[3] *Janeček, J.*: Gentlemani nečtou cizí dopisy. Books, Brno, 1998.

[4] *Shannon, C. E.*: Communication Theory of Secrecy Systems. Bell System Technical Journal, 1949.

[5] *Hála, V.*: Kvantová kryptografie. Aldebaran Bulletin 4 (2005), č. 14. Dostupné z www [http://aldebaran.cz/bulletin/2005_14_kry.php].

[6] *Vernam, G. S.*: Secret signaling system. 1310719 U.S. Patent, 22. červenec 1919.

[7] *Menezes, A. J., Oorschot van, P. C., Vanstone, S. A.*: Handbook of Applied Cryptography. CRC Press, Boca Raton, 1996.

[8] *Dušek, M.*: Kvantová kryptografie [online]. Koncepční otázky kvantové teorie [cit. 23. 8. 2010]. Dostupné z www [http://muj.optol.cz/dusek/predn/kokt/krypt.htm].

[9] *Andrew, R. a kol.*: A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications [online]. NIST Special Publications (800 Series) [cit. 21. 8. 2010]. SP 800-22 Rev 1a.
Dostupné z www [http://csrc.nist.gov/publications/nistpubs/800-22-rev1a/SP800-22rev1a.pdf].

[10] *Pierre, L., Richard, S.*: TestU01: A C Library for Empirical Testing of Random Number Generators. Université de Montréal: ACM Trans. Math. Softw. 33, 4, Article 22, 2007. DOI=10.1145/1268776.
[11] *Klíma, V.*: Blue Midnight Wish, kandidát na SHA-3 aneb poněkud privátně o tom, jak jsem k BMW přišel. Crypto-World. roč. 11 (2009), č. 2.
[12] *Musílek, M., Hubálovský, Š.*: Počítačová bezpečnost ve výuce informatiky (Tvorba hesel a steganografie).
[13] *Musílek, M., Hubálovský, Š.*: Počítačová bezpečnost ve výuce informatiky. Jednoduchá záměna (monoalfabetické šifry). MFI roč. 20 (2010/11), č. 6.
[14] *Musílek, M., Hubálovský, Š.*: Počítačová bezpečnost ve výuce informatiky (Luštění jednoduché záměny, frekvenční analýza). MFI roč. 20 (2010/11), č. 9.

Simulace elektronických obvodů programem Multisim a možnosti využití jeho speciálních funkcí vhodných pro výuku

PETR MICHALÍK – PAVEL BENAJTR

Fakulta pedagogická, Západočeská univerzita, Plzeň

Simulační programy patří mezi moderní výukové prostředky. Jedním z jejich představitelů je program Multisim, který vytvořila mezinárodní společnost National Instruments. Společnost je zaměřena na výzkum i vývoj měřicí a řídící techniky. Na rozdíl od velké většiny jiných výrobců není zaměřena pouze na firemní oblast. Zajišťuje podporu pro školy v podobě seminářů i jiných aktivit, např. soutěže a veletrhy. Uvedený simulační program je určen pro simulaci a návrh elektronických obvodů. Využívají jej nejen profesionální firmy, ale také velmi často odborné školy a univerzity, např. na Pedagogické fakultě ZČU v Plzni. Vzhledem k několikaletým zkušenostem s uvedeným simulačním programem na Pedagogické fakultě, lze zhodnotit výsledky simulací jako velmi dobré v porovnání s reálným měřením v laboratoři.