Věci, které jsou levné, nebývají příliš kvalitní. Když ale jde o algoritmus, který se má stát šifrovacím standardem, je situace úplně jiná. Musí být superkvalitní a zcela zadarmo. Na jeden takový se nyní podíváme podrobně.

Šifrovací standard zdarma

V polovině devadesátých let nebyla situace na poli šifrovacích algoritmů růžová. Mnozí lidé pochopili, že DES už má svá nejlepší léta dávno za sebou, ale neměli místo něj příliš na vybranou. Existovalo totiž jen velmi málo alternativ. Mnohé z nich byly pouze za peníze – při komerčním použití se za ně muselo platit. To byl zejména případ švýcarské šifry IDEA, amerických šifer RC2, RC4, RC5 a dalších. V této situaci přišla kanadská společnost Entrust Technologies s nápadem poskytnout šifrovací algoritmus, který by byl volně šiřitelný a použitelný zdarma pro osobní i komerční účely. Proto sponzorovala vývoj nového algoritmu, který by byl dostatečně variabilní a silný, použitelný bez přehnaných nároků na složitost a také dostatečně svižný. Výsledek byl nazván ***CAST*** podle iniciál svých kanadských tvůrců (C. Adams, S. Tavares) a na internetu byl v květnu roku 1997 publikován pod označením *RFC 2144* .

Americký, nebo kanadský?

CAST tak přinesl alternativu k algoritmu *Blowfish*, který ze stejných důvodů nabídl jako freeware Američan B. Schneier o dva roky dříve. Tím vznikla konečně i v oblasti freewaru určitá konkurence a nabídka. CAST si brzo získal příznivce na celém světě, stejně tak jako Blowfish, a oba se díky bezplatnému použití brzy objevily v řadě komerčních produktů.

Výhodou CAST se staly dvě podstatné skutečnosti, které jej upřednostňovaly před algoritmem Blowfish. Jednak jej ve svých produktech začal používat Microsoft, jednak získal certifikát kvality od oficiálního kanadského úřadu CSE (Communication Security Establishment), a dá se tak použít pro ochranu citlivých dat v kanadských vládních organizacích a institucích (určení pro stupeň “designated information” podle kanadského systému klasifikace). Za zmínku stojí i jeho včlenění do programu PGP (Pretty Good Privacy), kde nahradil pro komerční účely licenčními poplatky zatížený švýcarský algoritmus IDEA.

Je CAST podobný DES?

Ano, v mnoha směrech je. Používá ovšem řadu nových kryptografických myšlenek a je nesrovnatelně kvalitnější. S DES má společný tzv. *Feistelův princip*. Algoritmus Feistelova typu pracuje v cyklech totožných operací, které se nazývají *rundy* (rounds). Každá runda opakuje stejné operace se vstupem a výstup z jedné rundy je zároveň vstupem do rundy následující, ale v různých rundách jsou

použity odlišné tzv. rundovní klíče (viz též připojená schémata). Rundovní klíče se odvozují v inicializační fázi algoritmu z vlastního šifrovacího klíče různě složitým procesem (DES například oproti CAST používá velmi jednoduchý postup, který může vytvořit i tzv. slabé klíče).

Feistelův princip umožňuje pro odšifrování použít stejný postup jako pro zašifrování, jen se obrátí pořadí výběru rundovních klíčů (zkuste si tento geniální princip rozmyslet podle obrázku). Tato vlastnost je výhodná zejména pro hardware a šifrovací čipy. Další podobnost s DES je opět na úrovni obecného principu, kterým jsou tzv. *substitučně-permutační sítě*. DES i CAST používají substituce, a to ve formě pevných substitučních tabulek zvaných *S-boxy*. DES má osm S-boxů, které zobrazují šest bitů na čtyři, u CAST to jsou čtyři mohutnější S-boxy S1 až S4, které převádějí osm bitů na 32. Toto "rozšíření" je jedním z nových prvků, které se objevily u několika šifer devadesátých let.

Dále DES používá permutaci bitů (přeházení pořadí bitů), která je skutečným "zabijákem" pro softwarovou realizaci, zatímco CAST jako permutaci používá cyklickou rotaci bitů. Tu lze, jak známo, softwarově realizovat velmi jednoduše. Kromě toho CAST používá aritmetické operace sčítání a odčítání v celé šíři 32bitového slova, tj. v modulu 2^{32}. Uvedené operace jsou kryptologicky silnější a při realizaci současně rychlejší. Výsledkem je proto šifra kvalitnější a rychlejší než DES. Autoři uvádějí, že rychlost šifrování dat v operační paměti PC je na 150MHz Pentiu asi 3 MB/s.

CAST je bezpečný

Díky výše uvedeným zesílením oproti DES si CAST získal velkou důvěru. Přispělo k tomu jistě i zveřejnění všech kryptografických detailů a úvah o bezpečnosti algoritmu ve studii "Constructing Symmetric Ciphers Using the CAST Design Procedure" (viz infotipy).

CAST výhodně využívá skutečnosti, že Feistelův princip je už velmi dlouho používaný a prozkoumaný, a přitom zavádí další kvalitní kryptografické operace. Výsledkem jsou velmi dobré vlastnosti algoritmu, jako je konfuze, difuze, lavinovitost, nezávislost bitů atd., což dosvědčuje i skutečnost, že dosud nebyla odhalena žádná jeho slabina.

CAST se samozřejmě vyhýbá všem notoricky známým nešvarům DES, jako je vlastnost komplementárnosti nebo existence slabých a poloslabých klíčů. Dalším důležitým předpokladem bezpečnosti je délka klíče. Zde dává CAST uživateli volnou ruku a umožňuje volit délku klíče od 40 až do 128 bitů (po osmi bitech). Posledním – a nepochybně zásadním – potvrzením jeho kvality se stalo už zmíněné oficiální stanovisko kanadského úřadu CSE.

Popis algoritmu

Popis lze shrnout do čtyř základních kroků. Označme bity otevřeného textu (vstupu algoritmu) jako $m_1 ... m_{64}$, bity šifrovacího klíče K jako $k_1 ... k_{128}$ (pokud je klíč kratší, je do délky 128 bitů doplněn nulami) a šifrový text (tj. výstup algoritmu) jako $c_1 ... c_{64}$. Algoritmus má 12 nebo 16 rund, podle toho, zda původní klíč má délku do 80 bitů (včetně) nebo více než 80 bitů. Rundy nemají zcela totožnou míchací funkci **f** (jak je znázorněno na obrázku Feistelova principu), ale funkce **f** jsou tři. Liší se v různém pořadí výběru operací **+**, **–**, **xor**, které jsou v obrázku znázorňujícím míchací funkci označeny jako operace **a**, **b**, **c**, **d**. Proto rozeznáváme typ1 (f1), typ2 (f2) a typ3 (f3) této funkce. Konkrétní naplnění operací u funkcí f1 až f3 je uvedeno dále v odstavci o míchací funkci **f**.

Protože se popis algoritmu mění v závislosti na délce klíče **K**, zohledňuje se to také v označení

algoritmu. Algoritmus se pak označuje jako ***CAST5-x***, přičemž číslo za pomlčkou udává délku klíče v bitech (12 možností). Za základní verze se považují CAST5-40, CAST5-80 a CAST5-128. K nim jsou také definovány kontrolní příklady (viz infotipy).

Zašifrování

1. krok:

Příprava rundovních klíčů. Z klíče **K** se vypočte 16 párů rundovních klíčů $\{Km_i, Kr_i\}$, i = 0 ... 15, kde Km_i jsou 32bitová slova (maskovací klíče) a Kr_i pětibitová slova (rotační klíče).

2. krok:

Otevřený text se rozdělí na levé (**L**) a pravé (**R**) 32bitové slovo:

$L_0 = m_1 ... m_{32}, \quad R_0 = m_{33} ... m_{64}$.

3. krok:

Nyní následuje 16 (případně 12) rund (i = 0 ... 15), v nichž se vypočítávají L_{i+1} a R_{i+1} z L_i a R_i podle vztahů:

$L_{i+1} = R_i$

$R_{i+1} = L_i \text{ xor } f(R_i, Km_i, Kr_i)$

Přitom funkce **f** je vybíraná postupně jako f1, f2, f3 a cyklicky dále.

4. krok:

Na závěr proběhne výměna pořadí L_{16} a R_{16}, tj. šifrový text je definován jako $c_1 ... c_{64} = (R_{16}, L_{16})$ – nikoliv, jak by se dalo očekávat, jako (L_{16}, R_{16}).

Odšifrování

Jak už bylo řečeno, odšifrování probíhá stejným postupem jako zašifrování. Proto byl také učiněn 4. krok v uvedeném popisu. Jediný rozdíl při odšifrování oproti zašifrování spočívá v obrácení pořadí výběru párů rundovních klíčů. Jako první se tedy použije ten pár rundovních klíčů, který byl při zašifrování použit jako poslední, atd.

Substituční boxy

CAST používá 8 konstantních substitučních boxů S1 až S8, které zobrazují bajt na čtyřbajtové slovo. Každý S-box tedy zabírá v paměti 1 KB. Při vlastním šifrování dat ale postačí vyhradit paměť 4 KB na S-boxy S1 až S4, neboť S5 až S8 se používají pouze k přípravě párů rundovních klíčů.

Míchací funkce f

V míchací funkci jsou použity běžné operace intelských procesorů, jako je sčítání (**+**) a odčítání (**–**) 32bitových slov v modulu 2^{32} (tj. případné přetečení se zanedbává), operace **xor** a cyklická rotace 32bitového slova o n bitů doleva (**<<< n**). Pro snadnější popis označme **D** vstupní 32bitové slovo funkce **f** a **I** 32bitové slovo, které vznikne jako mezivýsledek. Jednotlivé bajty slova **I** označme od nejvýznamnějšího **Ia** až po nejméně významný **Id**. Zápis míchací funkce **f** je pak jednoduchý:

Typ 1:

$I = ((Km_i + D) <<< Kr_i)$

f = ((S1[Ia] xor S2[Ib]) - S3[Ic]) + S4[Id]

Typ 2:

$I = ((Km_i \text{ xor } D) <<< Kr_i)$

f = ((S1[Ia] - S2[Ib]) + S3[Ic]) xor S4[Id]

Typ 3:

$I = ((Km_i - D) <<< Kr_i)$

f = ((S1[Ia] + S2[Ib]) xor S3[Ic]) - S4[Id]

Tyto rovnice pak definují konkrétní instance operací **a** až **d** v obrázku míchací funkce.

Příprava rundovních klíčů

Zbývá popsat, jak se vytvářejí rundovní klíče z $k_1 ... k_{128}$. Postup je dost podobný vlastnímu procesu šifrování, ale jeho popis by vyžadoval poměrně hodně místa. Zájemce proto odkazujeme na popis uvedený v RFC 2144 nebo na programovou realizaci (viz infotipy). Poznamenejme, že zde, a právě jen zde, se využívají S-boxy S5 až S8, které jsou také pevně definované. Výstupem je nakonec 12 nebo 16 párů rundovních klíčů $\{Km_i, Kr_i\}$.

Závěr

CAST představuje zdarma standard, který je široce akceptován, je dostatečně rychlý a bezpečný a má možnost silné i slabé verze podle délky klíče. Co si přát více? Není divu, že vyplnil vakuum, které existuje od “pádu” DES, a potrvá ještě jeden dva roky, než bude vybrán nový americký šifrovací standard AES.

Vlastimil Klíma (vklima@decros.cz)

Infotipy

http://ds.internic.net/frc/

rfc2144 .txt

(Obsahuje definici a základní dokument – RFC 2144: CAST-128 Encryption Algorithms, C. Adams, Entrust Technologies, May 1997.)

http://www.entrust.com/library.htm

(Obsahuje studii “Constructing Symmetric Ciphers Using the CAST Design Procedure” o kryptograficko-bezpečnostních aspektech algoritmu.)

http://adonis.ee.queensu.ca:8000/cast/cast.html

(Obsahuje dokument “CAST Encryption Algorithm Related Publications” s dalšími informacemi k algoritmu.)

http://www.mhv.net/~mgraffam/ce/cryptography.html

(Obsahuje volně šiřitelnou implementaci CAST v jazyce C.)