AES a jeho implementace

Doc. Karel BURDA, CSc.

Vojenská akademie, Brno

Abstrakt

Příspěvek pojednává o kryptografickém algoritmu AES, který se pravděpodobně stane novým kryptografickým standardem USA. V článku je nejprve uveden popis a implementace elementárních operací AES. Druhá část článku popisuje šifrování a třetí část je věnována dešifrování v AES. V poslední části příspěvku jsou rozebrány možnosti implementace AES nejčastějšími typy procesorů.

1 Úvod

V roce 1997 americký NIST (National Institute of Standards and Technology - Národní institut standardů a technologií) inicioval proces výběru nového symetrického kryptografického algoritmu k ochraně citlivých informací ve státní sféře. Plánovaný standard byl označen zkratkou AES (Advanced Encryption Standard - zdokonalený šifrovací standard). Vyhlášeného výběru se celkově zúčastnilo 15 algoritmů, které byly veřejně posuzovány až do začátku roku 2001. Na základě zjištěných poznatků NIST navrhl jako nový kryptografický standard šifru Rijndael [1, 2], kterou navrhli pánové J. Daemen a V. Rijmen.

Ze všech přihlášených algoritmů šifra Rijndael nejlépe vyhověla požadavkům na bezpečnost, výkonnost, účinnost, flexibilitu a snadnost implementace. V závěrečné zprávě je vyzdvižena především výkonnost šifry v hardwarové i v softwarové implementaci, nízké paměťové nároky a snadná ochrana proti útokům parazitními kanály.

AES by měl být novým standardem symetrické blokové šifry pro státní instituce USA. Dá se předpokládat, že podobně jako jeho předchůdce DES (Data Encryption Standard) bude široce využíván i v komerční sféře.

AES je symetrická bloková šifra s délkou datového bloku 128 bitů a s volitelnou délkou klíče $D_k = 128$, 192 nebo 256 bitů. Kromě samotného šifrováni a dešifrování umožňuje i jiné využití - například autentizaci dat (MAC, hašování) nebo generování pseudonáhodných čísel.

2 Implementace elementárních operací AES

Základní datovou jednotkou algoritmu je osmice bitů (bajt) $a = (a_7, a_6, a_5, a_4, a_3, a_2, a_1, a_0)$, kde $a_i \in \{0, 1\}$. Kromě uvedeného vektorového vyjádření je výhodné reprezentovat bajt polynomem $a(x) = a_7 \cdot x^7 + a_6 \cdot x^6 + a_5 \cdot x^5 + a_4 \cdot x^4 + a_3 \cdot x^3 + a_2 \cdot x^2 + a_1 \cdot x + a_0$ (viz. např. [3]). Jednotlivé bajty pak lze považovat za prvky tzv. Galoisova tělesa $T = \mathrm{GF}(2^8)/m(x)$, kde $m(x) = (x^8 + x^4 + x^3 + x + 1)$.

V tělese T je definováno sčítání bajtů:

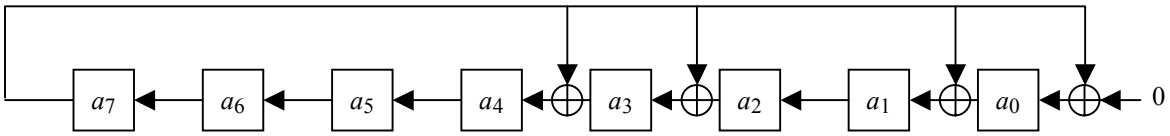
$$c = a \oplus b, \text{ kde } c_i = \begin{cases} 0, \text{ pokud } a_i = b_i \\ 1, \text{ pokud } a_i \neq b_i \end{cases} \qquad (1)$$

a násobení bajtů:

$$c = a \bullet b, \text{ kde } c_i \text{ je } i-\text{tý koeficient polynomu } c(x) = [a(x) \cdot b(x)] \bmod m(x) \qquad (2)$$

Operace sčítání je hardwarově i softwarově dostupná přímo - jedná se o sčítačky modulo 2, resp. o operaci EXOR. Implementace násobení je poněkud složitější. V návrhu standardu je definována

operace b = xtime(a), která odpovídá operaci $b(x) = [a(x) \cdot x] \bmod m(x)$. Hardwarově lze uvedenou operaci realizovat pomocí obvodu na obr. 1. Daný obvod sestává z osmibitového registru se zpětnou vazbou podle polynomu $m(x)$ a ze čtyř sčítaček modulo 2. Operace b = xtime(a) spočívá v naplnění registru bajtem a a následným posuvem tohoto bajtu o jednu pozici doleva. Nový obsah registru je výsledným bajtem b.



Obr. 1: Hardwarová implementace operace xtime.

Softwarová implementace je analogická k hardwarové:

1. Když $a_7 = 1$, tak nastav p = 00011011. Jinak p = 00000000.
2. Pro $i = 7$ až 1 nastav $a_i = a_{i-1}$. Dále nastav $a_0 = 0$.
3. $b = a \oplus p$.

V prvém kroku se nastavuje hodnota zpětné vazby, ve druhém kroku se provádí posuv bajtu o jednu pozici doleva a v posledním kroku se zpětná vazba realizuje.

Samotné násobení $c = (a \bullet b)$ se pomocí operace xtime softwarově implementuje podle následujícího předpisu:

1. Nastav c = 00000000 a $p = a$.
2. Pro $i = 0$ až 7 proveď:
   2.1. $c = c \oplus (b_i \cdot p)$.
   2.2. p = xtime(p).

Hardwarová implementace násobení vychází z uvedeného předpisu. K realizaci násobení je zapotřebí jeden obvod A pro operaci xtime a dva osmibitové registry B a C. Na počátku registr obvodu A naplníme bajtem a, registr B naplníme hodnotou bajtu b a registr C hodnotou c = 00000000. Postupně provedeme 8 kroků pro $i = 0$ až 7. V rámci každého kroku podmínečně přičítáme podle aktuální hodnoty b_i k obsahu registru C obsah registru v A. Taktéž v každém kroku provedeme v obvodu A operaci xtime. Výsledkem násobení je obsah registru C po skončení osmého kroku.

Další elementární operací v AES je bajtová substituce. Formálně se zapisuje b = SubBytes(a) a její realizace je následovná. Nejprve se k číslu reprezentovanému bajtem a nalezne v tělese T inverzní hodnota $x = a^{-1}$. Pro $a = 0$ je definováno, že $x = 0$. Bajtem x se podle následujícího vztahu vynásobí binární matice a k výsledku se přičte konstanta:

$$\begin{bmatrix} b_7 \\ b_6 \\ b_5 \\ b_4 \\ b_3 \\ b_2 \\ b_1 \\ b_0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x_7 \\ x_6 \\ x_5 \\ x_4 \\ x_3 \\ x_2 \\ x_1 \\ x_0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} \qquad (3)$$

Násobení i sčítání se v této operaci provádí po bitech modulo 2. Operace SubBytes je numericky poměrně náročná a proto se předpokládá její implementace ve formě předem vypočítané substituční tabulky o rozsahu 256 bajtů.

3 Šifrovací algoritmus AES

V předchozí části jsme si uvedli základní matematické operace ve standardu AES a nyní můžeme přejít k stručnému popisu samotného algoritmu. Šifrování a dešifrování se provádí v tzv. stavové matici formátu (4 x 4) bajty. Zápis do matice a výpis z matice se realizuje po sloupcích matice zleva doprava. Například pro vstupní blok bajtů A, B, C, ..., O, P bude stavová matice následující:

Jednotlivé bajty matice jsou indexovány od 0 po 3. Potom například prvek matice $a_{2,3}$ je v našem případě roven bajtu 0.

Stavová matice =

A	E	I	M
B	F	J	N
C	G	K	O
D	H	L	P

Bajtově se do tzv. klíčové matice o čtyřech řádcích zapisuje i klíč. Při délce klíče D_k = 128, 192 nebo 256 bitů (tj. 16, 24 nebo 32 bajtů) bude vyplněno k sloupců klíčové matice, kde $k = D_k / 32 = 4$, 6 nebo 8. Šifrování a dešifrování sestává z r rund, jejichž počet závisí na délce klíče. Pro AES je stanoveno, že $r = (k + 6)$. Počet rund je pak 10, 12 nebo 14.

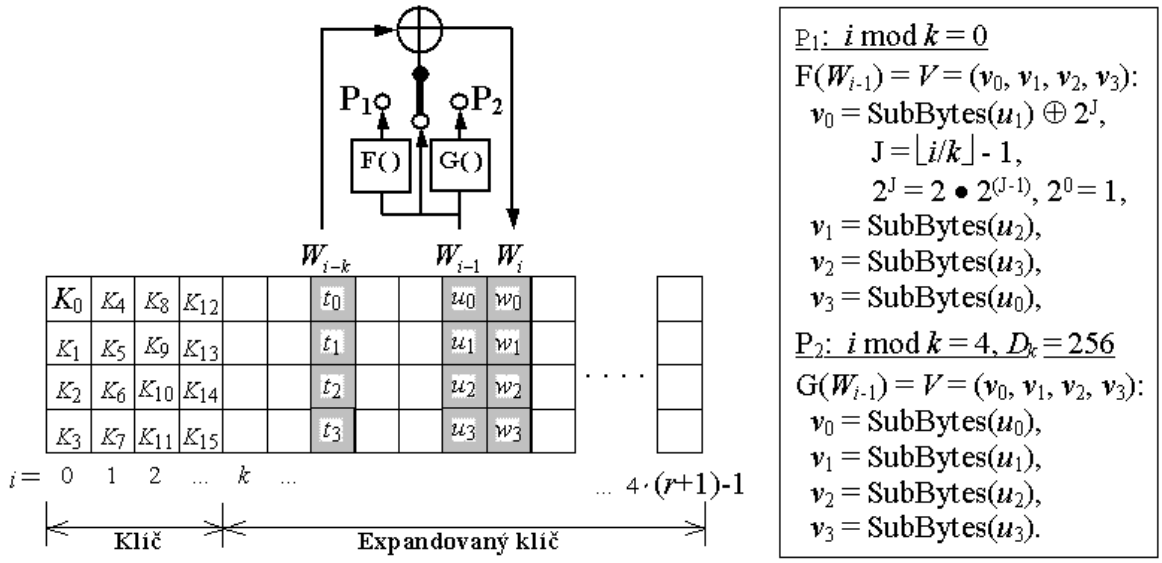
Obecný zápis algoritmu pro šifrování je následující:

1. Expanze klíče (KeyExpansion)
2. Přičtení klíče (AddRoundKey)
3. Provedení (r-1) rund:
   3.1. Bajtová substituce (SubBytes)
   3.2. Rotace v řádcích (ShiftRows)
   3.3. Násobení maticí (MixColumns)
   3.4. Přičtení klíče (AddRoundKey)
4. Provedení finální rundy:
   4.1. Bajtová substituce (SubBytes)
   4.2. Rotace v řádcích (ShiftRows)
   4.3. Přičtení klíče (AddRoundKey)

Ze zápisu šifrovacího algoritmu je zřejmé, že se (r + 1)-krát provádí procedura přičtení klíče (AddRoundKey). K tomu je zapotřebí (r + 1) rundovních klíčů ve formě matice rundovního klíče, která má formát (4 x 4) bajty. V prvém kroku algoritmu se proto provádí expanze klíče. Na obr. 2 je uveden příklad expanze klíče pro délku klíče D_k = 128 bitů. Šifrovací klíč se nejprve zapíše po jednotlivých bajtech do klíčové matice o 4 řádcích po sloupcích zleva doprava. Tím je v této matici k dispozici $k = D_k / 32 = 4$ sloupce klíče. Pro zašifrování datového bloku je celkově zapotřebí $[(r + 1) \cdot 4]$ sloupců. Každý další i-tý sloupec klíčové matice W_i se rekurentně odvozuje z předchozích sloupců W_{i-1} a W_{i-k} podle následujícího předpisu:

$$W_i = W_{i-k} \oplus V, \quad \text{kde } V = \begin{cases} \mathrm{F}(W_{i-1}), \text{ pokud } i \bmod k = 0, \\ \mathrm{G}(W_{i-1}), \text{ pokud } i \bmod k = 4 \text{ a } D_k = 256 \text{ bitů}, \\ W_{i-1}, \text{ jinak}, \end{cases} \qquad (4)$$

přičemž funkce F( ) a G( ) jsou uvedeny v legendě k obr.2.



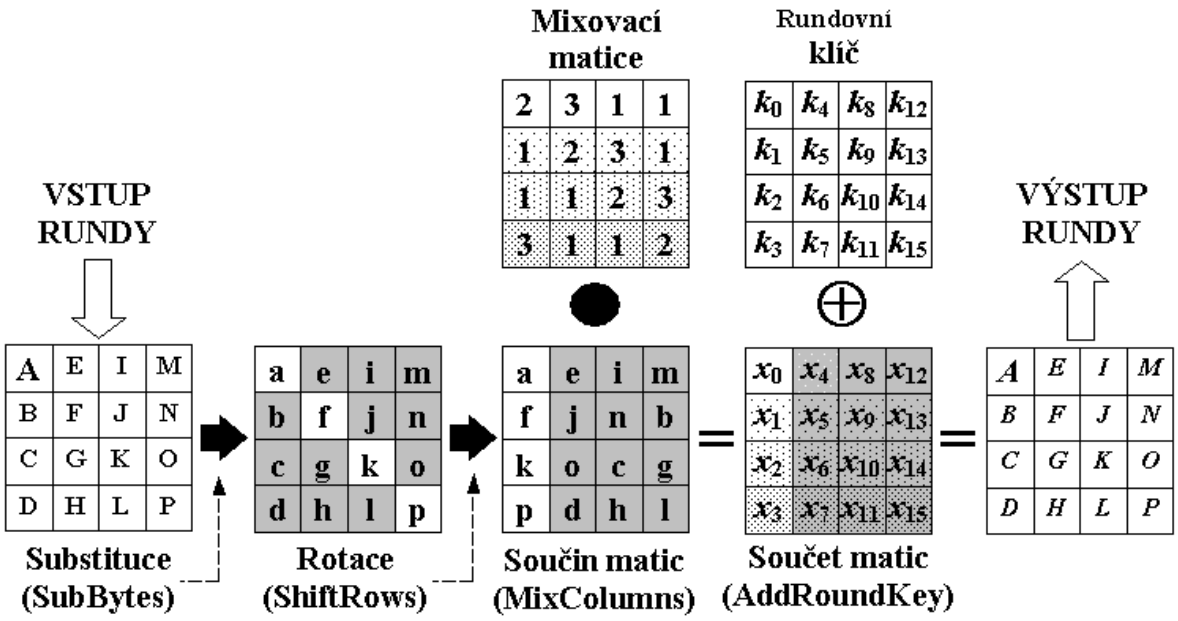
Obr.2: Expanze klíče.

Druhým krokem šifrování je přičtení klíče – AddRoundKey. Prvními čtyřmi sloupci klíčové matice se naplní matice rundovního klíče $K = [k_{i,j}]$, kde $i, j = 0, 1, 2, 3$. Nové hodnoty stavové matice $A = [a_{i,j}]$ se získají sečtením bajtů rundovního klíče a aktuálních bajtů stavové matice:

$$a_{i,j} = a_{i,j} \oplus k_{i,j}, \quad i, j = 0,1,2,3. \tag{5}$$

Po úvodním přičtení klíče následuje provedení (r–1) rund. Celkové schéma jednotlivé rundy je uvedeno na obr. 3. Se všemi jednotlivými bajty vstupní stavové matice se nejprve provede bajtová substituce (SubBytes). S takto vzniklou stavovou maticí se provede cyklická rotace vlevo po řádcích (ShiftRows). První řádek zůstává, druhý řádek se cyklicky posune vlevo o jednu pozici, třetí řádek o dvě pozice a poslední řádek se posune o tři pozice. Po této operaci se ze stavové matice berou jednotlivé sloupce a násobí se jimi mixovací matice (MixColumns). Výsledkem každého takového součinu je sloupcový vektor, který je vložen na příslušnou pozici stavové matice. Poté se naplní matice rundovního klíče prvními čtyřmi doposud nepoužitými sloupci klíčové matice. Toto matice se přičte bajt po bajtu do stavové matice. Výsledná stavová matice je výstupem dané rundy a zároveň vstupem pro rundu následující.

Finální runda se od běžné rundy odlišuje tím, že neobsahuje operaci MixColumns. Později uvidíme, že důvodem uvedené odlišnosti je požadavek, aby šifrování a dešifrování bylo možné realizovat hardwarově stejným způsobem.



Obr. 3: Celkové schéma jednotlivé rundy.

4 Dešifrovací algoritmus AES

Jednotlivé kroky šifrovacího algoritmu realizují prosté zobrazení vstupní stavové matice na výstupní stavovou matici. To znamená, že ke všem operacím šifrovacího algoritmu existují inverzní operace. Dešifrovací algoritmus je potom postupností odpovídajících inverzních operací uspořádaných v opačném pořadí než při šifrování.

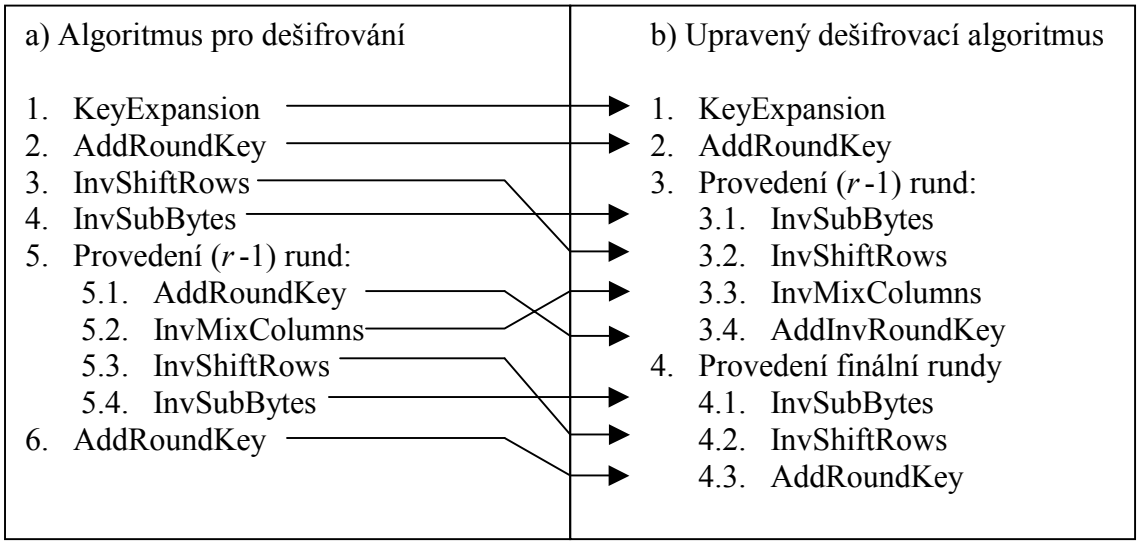
Inverzní operací k bajtové substituci je inverzní bajtová substituce - InvSubBytes. Na základě (3) ji lze teoreticky realizovat přičtením substituční konstanty, vynásobením inverzní substituční maticí a následným výpočtem inverzní hodnoty získaného výsledku v tělese T. Mnohem jednodušší je však implementace uvedené operace inverzní substituční tabulkou v rozsahu 256 bajtů.

Inverzní operací k rotaci řádků je opačná rotace řádků - InvShiftRows. V tomto případě se provádí cyklický posun řádků doprava. Inverzní operací k násobení mixovací maticí je násobení inverzní mixovací maticí - InvMixColumns. První řádek této matice v dekadickém vyjádření je vektor (14, 11, 13, 9). Ostatní řádky vzniknou cyklickou rotací uvedeného vektoru o jednu pozici doprava oproti předchozímu řádku. Hardwarová, resp. softwarová implementace všech inverzních operací je stejná jako u šifrovacích operací.

Zápis dešifrovacího algoritmu je uveden na obr. 4a. Tento dešifrovací algoritmus vznikne inverzním seřazením inverzních operací. Pro hardwarovou implementaci je výhodné, aby šifrovací i dešifrovací algoritmus měly stejnou strukturu. V tomto případě se využívá následujících skutečností:

1. Pořadí operací InvShiftRows a InvSubBytes lze prohodit, protože operace InvShiftRows nemění hodnoty bajtů a operace InvSubBytes mění hodnotu konkrétního bajtu nezávisle na jeho pozici ve stavové matici.
2. Postupnost operací AddRoundKey $\rightarrow$ InvMixColumns lze nahradit postupností InvMixColumns $\rightarrow$ AddInvRoundKey. Vychází se zde z rovnosti $M^{-1} \times (A \oplus K) = (M^{-1} \times A) \oplus (M^{-1} \times K)$, kde M^{-1} je inverzní mixovací matice, A je stavová matice a K je matice rundovního klíče. První sčítanec pravé části rovnice je operace InvMixColumns. Druhý sčítanec se získá vynásobením inverzní mixovací matice maticí rundovního klíče. Takto vytvořená matice se přičte jako rundovní klíč (operace AddInvRoundKey). Inverzní rundovní klíče se vypočítávají v průběhu KeyExpansion.

Důsledkem úprav algoritmu z obr. 4a. podle výše uvedených záměn je algoritmus na obr. 4b. Porovnáním se zápisem šifrovacího algoritmu vidíme, že struktura šifrovacího a dešifrovacího algoritmu je nyní stejná.



Obr. 4: Algoritmus pro dešifrování - a) základní, b) upravený.

5 Implementace AES pro různé procesory

Doposud popsané možnosti implementace operací SubBytes, RowShifts a AddRoundKey jsou obecné pro všechny typy používaných procesorů. Pro 8-bitové procesory je možné specificky řešit operaci MixColumns, kde se čtyřikrát provádí násobení sloupce bajtů stavové matice $W = (a_0, a_1, a_2, a_3)$ mixovací maticí M v tělese T. V návrhu AES se doporučuje následující implementace:

1. $p = a_0 \oplus a_1 \oplus a_2 \oplus a_3$
2. Pro $i = 0$ až 3 proveď: $a_i = p \oplus a_i \oplus \text{xtime}(a_i \oplus a_j)$, kde $j = (i + 1) \bmod 4$.

Po provedení uvedeného postupu pak nový obsah sloupce W odpovídá hodnotě $M \bullet W$. Uvedené řešení však lze použít pouze pro šifrování. Inverzní mixovací matice obsahuje čísla, která se musí násobit klasickým způsobem. Proto je dešifrování pomalejší.

Dalším častým omezením pro 8-bitové procesory je malý objem paměti. Toto omezení může být kritické pro operaci expanze klíče. Uvedený problém lze řešit průběžnou expanzí klíče vždy před jednotlivými rundami. V tomto případě postačí pro expanzi klíče paměť $(4 \cdot k)$ bajtů.

U 32-bitových procesorů lze při implementaci AES využít toho, že mohou současně pracovat se 4 bajty (tj. s jedním sloupcem stavové matice) a že zpravidla disponují dostatečně velkou pamětí. V návrhu AES je uvedena rychlá implementace jednotlivé rundy. Výpočet nové hodnoty j-tého sloupce stavové matice $W_j = (a_{0,j}, a_{1,j}, a_{2,j}, a_{3,j})$ lze pro šifrovací rundu formálně vyjádřit následovně:

$$\begin{bmatrix} a_{0,j} \\ a_{1,j} \\ a_{2,j} \\ a_{3,j} \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \bullet \begin{bmatrix} \mathrm{S}(a_{0,j}) \\ \mathrm{S}(a_{1,j+1}) \\ \mathrm{S}(a_{2,j+2}) \\ \mathrm{S}(a_{3,j+3}) \end{bmatrix} \oplus \begin{bmatrix} k_{0,j} \\ k_{1,j} \\ k_{2,j} \\ k_{3,j} \end{bmatrix}, \qquad (6)$$

kde S( ) je operace SubBytes, sloupcový index substituovaných bajtů (j+h) je v modulu 4 a vektor $K_j = (k_{0,j}, k_{1,j}, k_{2,j}, k_{3,j})$ je j-tý sloupec matice rundovního klíče. Zavedeme-li čtyři tabulky T_0 až T_3:

$$T_0(a) = \begin{bmatrix} 2 \bullet S(a) \\ S(a) \\ S(a) \\ 3 \bullet S(a) \end{bmatrix}, T_1(a) = \begin{bmatrix} 3 \bullet S(a) \\ 2 \bullet S(a) \\ S(a) \\ S(a) \end{bmatrix}, T_2(a) = \begin{bmatrix} S(a) \\ 3 \bullet S(a) \\ 2 \bullet S(a) \\ S(a) \end{bmatrix}, T_3(a) = \begin{bmatrix} S(a) \\ S(a) \\ 3 \bullet S(a) \\ 2 \bullet S(a) \end{bmatrix}, \qquad (7)$$

pak můžeme výpočet j-tého sloupce stavové matice v jedné rundě vyjádřit:

$$W_j = T_0(a_{0,j}) \oplus T_1(a_{1,j+1}) \oplus T_2(a_{2,j+2}) \oplus T_3(a_{3,j+3}) \oplus K_j . \qquad (8)$$

Vidíme, že při tomto způsobu implementace potřebujeme 4 tabulky T_0 až T_3 s 256 hodnotami po čtyřech bajtech (tj. celkově 4 kbajty) a čtyři sloupcová sčítání. Variantou uvedeného postupu, která má nižší paměťové nároky, ale větší zpoždění je varianta založená na funkci cyklické rotace bajtů ve sloupci $R(W_j) = (a_{1,j}, a_{2,j}, a_{3,j}, a_{0,j})$. Z definice tabulek je zřejmé, že $R[T_{i+1}(a)] = T_i(a)$. Potom k provedení rundy postačí jediná tabulka. Například při volbě tabulky $T = T_3$ platí:

$$W_j = R\Big(R\Big(R\big(T(a_{0,j})\big) \oplus T(a_{1,j+1})\Big) \oplus T(a_{2,j+2})\Big) \oplus T(a_{3,j+3}) \oplus K_j . \qquad (9)$$

Pro každou rundu se uvedený vztah použije k výpočtu všech čtyř sloupců. Finální runda se musí realizovat klasicky.

6 Závěr

Z doposud uvedeného popisu lze celkově konstatovat, že AES je iterovaná bloková šifra se substitučně-lineární strukturou. Sestává z lineární postupnosti jednoduchých lineárních operací prokládaných nelineární substitucí. Lineární operace AddKey zajišťuje závislost kryptogramu na klíči a lineární operace ShiftRows a MixColumns zajišťují rozptýlení redundance zprávy (tzv. difúzi - anglicky "diffusion"). Nelineární operace SubBytes zabezpečuje zakrytí závislostí mezi zprávou a kryptogramem (anglicky "confusion").

Přínosem AES pro vývoj blokových šifer je důsledná bajtová architektura, která dovoluje jednoduchou implementaci šifry na nejrůznějších hardwarových a softwarových platformách. Algoritmus je vhodný k implementaci na různých typech procesorů (od procesorů na čipových kartách až po digitální signálové procesory), na programovatelných hradlových polích, na specializovaných integrovaných obvodech atd. Dalším přínosem AES je maticová forma bloků zpracovávaných dat, která zajišťuje rychlejší difúzi. Tím se snižuje potřebný počet rund a zvyšuje se tak rychlost šifrování (v současné době už přes 1 Gb/s).

Nevýhodou AES je pomalejší dešifrování oproti šifrování. Pro mnohé aplikace (např. výpočet MAC nebo šifrovaný přenos v módu CFB, či OFB) však tato skutečnost není kritická, protože dešifrování se neprovádí. V případě softwarové implementace s využitím tabulek je nevýhodou nutnost použití jiných tabulek pro šifrování resp. dešifrování. V případě hardwarové implementace je nevýhodou skutečnost, že k dešifrování je zapotřebí určitý přídavný hardware pro výpočet inverzních rundovních klíčů.

Operace použité v AES jsou vůči doposud známým typům útoků prostřednictvím parazitních kanálů většinou imunní. Výjimku v tomto případě tvoří operace xtime. Bezpečné implementaci uvedené operace je nutné věnovat značnou pozornost.

Literatura

[1] DAEMEN, J. - RIJMEN, V.: AES Proposal: Rijndael. Dostupné na internetu: < http://csrc.nist.gov/encryption/aes/rijndael/Rijndael.pdf >.

[2] DAEMEN, J. - RIJMEN, V.: Draft FIPS for the AES. Dostupné na internetu: < http://csrc.nist.gov/publications/drafts/dfips-AES.pdf >.

[3] BURDA, K. - HALOUZKA, K.: Základy kryptologie. Vojenská akademie, Brno 2001.

Informace o autorovi

Doc. Karel BURDA, CSc. se narodil ve Frýdlantě v roce 1958. Vysokoškolské vzdělání získal na Vysoké vojenské technické škole (VVTŠ) v Liptovském Mikuláši v roce 1981. Poté sloužil u spojovacího útvaru v Písku. V roce 1985 nastoupil ve VVTŠ vědeckou přípravu, kterou ukončil v roce 1988 získáním vědecké hodnosti CSc.. Na škole v Liptovském Mikuláši působil jako učitel do roku 1993. Ve stejném roce byl přijat jako vysokoškolský učitel na Vojenskou akademii v Brně, kde v roce 2001 získal pedagogický titul Doc. Na Vojenské akademii působí dodnes. Specializuje se na teorii informace, kryptologii a modelování komunikačních systémů.