Blokové šifry

Jan Přikryl

16. prosince 2013

Toto je vývojová verze dokumentu. Obsahuje druhou kryptologickou kapitolu rozepsaných skript pro předmět 11KZK ve formě, v jaké se nacházela k datu, uvedenému nahoře.

Historie úprav:

1.1.2011	jprk	Zrestaurovaná první verze po vykradení auta na základě jakéhosi PDF, které náhodou přežilo.
5.1.2013	jprk	Doplněno AES informacemi z knihy [DR02].
15.12.2013	jprk	Doplněno vysvětlení GF(2^8) u AES.

Obsah

V současné době používané blokové symetrické šifry jsou – pokud je nám známo – vždy založeny na kombinaci permutačních (transpozičních) šifer s komplikovanějšími substitučními šiframi. Jako zástupce těchto šifrovacích algoritmů blíže rozebereme již překonaný algoritmus DES (*Data Encryption Standard*) a srovnáme jej s konstrukcí dnes široce používané šifry AES (*Advanced Encryption Standard*).

Doplňující informace a detailnější rozbor vlastností diskutovaných algoritmů nalezne čtenář například v publikaci R. Mollina [Mol01, Mol07] či v textu pánů Paara a Pelzla [PP10]. Informace o AES pocházejí z knihy Joana Daemena a Vincenta Rijmena [DR02].

1 Dílčí transformace v blokových šifrách

Jak uvidíme dále, v symetrických šifrách (a tedy i v šifrách blokových) dosahujeme kryptografické bezpečnosti (tedy Shannonem definovaného zmatení a rozptýlení informace) iterativním opakováním nějakých dílčích transformací, jež jsou zjednodušeně řečeno substitučními a permutačními kroky. Pro tyto transformace se obecně používají jednoduché operace, jež lze snadno implementovat v hardwaru (sčítání bez přenosu, tedy XOR, posuny doleva a doprava, operace ve vhodně zkonstruovaných binárních tělesech, či vyhledávací tabulky).

1.1 Substituční transformace (S-box)

Úkolem substituční transformace v blokových šifrách je co nejvíce matematicky zkomplikovat vztah mezi šifrovacím klíčem a vlastním kryptogramem. V S-boxu tedy dochází k Shannonovskému *zmatení informace*.

S-box je substituční transformace, nahrazující malý blok bitů (vstup S-boxu) jiným blokem bitů (výstup S-boxu). V některých případech postup dešifrování krypytogramu vyžaduje, aby funkce S-boxu byla invertovatelná, a proto musí být tato náhrada *bijekcí* (tedy „jedna k jedné“). V takovém případě musí být délka výstupní sekvence S-boxu stejná, jako délka vstupu. To pro obecné neinvertovatelné S-boxy neplatí, běžný je například S-box transformující čtyři vstupní bity na šest výstupních, používaný v šifře DES. Dobrý S-box má tu vlastnost, že změna jednoho bitu vstupu změní asi polovinu výstupních bitů (tak zvaný *lavinový efekt*, jenž si popíšeme později v této kapitole). Důsledkem takové konstrukce S-boxu je, že informace z jednotlivých vstupních bitů je rovnoměrně rozptýlená do celkého výstupního slova a hodnota každého výstupního bit S-boxu závisí na hodnotě všech vstupních bitů. V případě, že by S-box byl proměnnou transformací, závislou na klíči, nebude jednoduché tuto vlastnost zaručit pro všechny možné kombinace vstupních data a klíčů. I to je důvod, proč je S-box většinou pevnou transformací, která na hodnotě klíče nezávisí (i když lze nalézt šifry, kde S-boxy na šifrovacím klíči nějakým způsobem závislé jsou – například Merkelova šifra Khufu [Mer91] nebo Schneierova Blowfish [Sch94]).

Příklad 1 (Čtyřbitový S-box)**.** *Pokud budeme uvažovat čtyřbitové vstupní hodnoty a čtyřbitový výstup, může výsledný S-box vypadat například takto:*

	00	*01*	*10*	*11*
00	*0000*	*1100*	*1101*	*1010*
01	*1111*	*1001*	*0001*	*0111*
10	*1011*	*0110*	*0010*	*0101*
11	*1000*	*0011*	*0100*	*1110*

Tento S-box je invertovatelný, jedná se o S-box S_3 v symetrické šifře Serpent.

V praxi je realizace kompletního lavinového efektu jedním S-boxem když ne nemožná, tak alespoň natolik obtížná, že dnešní šifry používají méně dokonalé S-boxy v kombinaci s permutační transformací, zavedenou níže, v několika iteracích. Po několika kolech opakování potom takovéto šifrovací schéma splňuje podmínky, uvedené v minulém odstavci.

1.2 Permutační transformace (P-box)

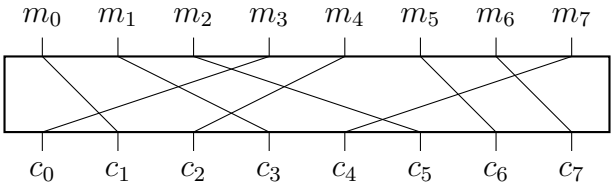
Druhou základní transformací v symetrických šifrách je permutace, implementovaná pomocí tak zvaného P-boxu. Ten reprezentuje transformaci, jež mezi sebou permutuje všechny bity vstupního slova. P-box sám o sobě nemá žádnou výraznou kryptografickou sílu, jeho hlavní role je v distribuci informace při iterativním používání stejné skupiny S-boxů několikrát po sobě. Kvalitní P-box pro použití s určitým počtem S-boxů má kromě snahy maximálního rozptýlení informace totiž i tu vlastnost, že výstupní bity předchozích S-boxů, předcházejících použitému P-boxu, jsou rozdistribuovány na tak mnoho následujících S-boxů, jak je jenom možné.

V P-boxu tedy dochází k Shannonovskému *rozptýlení informace.*

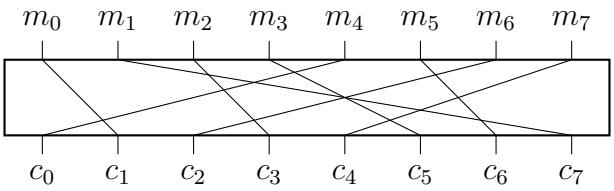
■ Prozkoumat diffusion vs. dispersion ■

■ Prozkoumat AES D-box ■

Příklad 2 (Osmibitový P-box)**.** *Osmibitový P-box by mohl vypadat například takto:*



Pokud ale tomuto P-boxu předcházejí dva čtyřbitové S-boxy S_1 a S_2 a stejné dva S-boxy mohou po nějakých dílčích operacích (blíže si to vysvětlíme za chvíli) následovat, bude informace, vystupující z S_1, mířit v převážné míře opět do S_1, neboť v prvním čtyřbitovém nibble jsou tři bity P-boxem pouze promíchány a jenom jeden pochází z S_2 (a pro S_2 platí zcela to samé). Kryptograficky vhodnější je proto v tomto konkrétním případě například P-box tohoto tvaru:



1.3 Zmatení a rozptýlení informace podruhé

Jak jsme se zmínili v předchozích odstavcích, jeden samostatně použitý S-box nebo P-box nemá sám o sobě téměř žádnou kryptografickou sílu: S-box reprezentuje jednoduše prolomitelnou substituční šifru, zatímco P-box implementuje jednoduchou transpoziční šifru. Nicméně, pokud šifrovanou informaci transformujeme posloupností *vhodně navržených dílčích substitucí a transpozicí*, popsaných S-boxy a P-boxy, a postupně ji zkombinujeme se šifrovacím klíčem, výsledek bude splňovat Shannonem definovaná kritéria pro zmatení a rozptýlení informace, zmíněná v minulé přednášce.

Podívejme se nejprve na kritérium *rozptýlení informace*. Pokud změníme jediný bit v reprezentaci šifrované zprávy, tento bit je transformován S-boxem na několik změněných bitů (ideálně okolo 50 % délky bloku, zpracovávaného S-boxem ■ **odkaz na SAC** ■). Po substituční transformaci bude v naší posloupnosti následovat transpozice bloku P-boxem, jehož účel je dvojí: používáme-li více, než jeden S-box, provedená permutace rozmístí změněné bity rovnoměrně do celého bloku dílčího kryptogramu, ■ **a co v případě jediného S-boxu? to asi fungovat nebude** ■. V dalším kroku posloupnosti opět přijde na řadu substituce S-boxy, jež znovu promíchají již změněné bity, a tak dále. Konečný výsledek celé této posloupnosti operací bude takový, že se pseudonáhodným způsobem změní všechny bity v šifrovaném bloku – v ideálním případě je splněna podmínka *silného lavinového efektu* (angl. *strict avalanche criterion*), tedy, že jeden změněný bit v bloku otevřeného textu vyvolá na výstupu změnu přibližně poloviny bitů v celém bloku kryptogramu.■ **Rozpracovat důvody pro** 50 % **změnu bitů – jde o maximalizaci entropie; SAC by mělo mít vlastní odstavec** ■.

Důvod pro splnění podmínky *zmatení informace* je obdobný, jako u rozptylu: budeme-li mezi jednotlivými kroky substituce a permutace přidávat k mezivýsledku dílčí klíč, odvozený vhodným způsobem od šifrovacího klíče, bude informace, obsažená v šifrovacím klíči, postupně velmi složitým pseudonáhodným postupem rozptýlena do celého kryptogramu a klíč nebude možno ze zprávy v rozumném čase odvodit. Toto platí i naopak: Pokud bychom změnili jediný bit kryptogramu a chtěli dešifrovat původní zprávu, bude to znamenat, že šifrovací klíč bude pseudonáhodnou permutací bitů původního klíče.

Pokud z výše uvedeného popisu nabýváte dojmu, že ideálním způsobem, jak implementovat zpracování otevřeného textu posloupností S-boxů a P-boxů, je provádět tuto dvojici transformací v předem daném počtu cyklů a výsledný kryptogram počítat iterativně, máte pravdu. Blíže si celý postup popíšeme v následujícím odstavci.

2 Iterované šifry

Z důvodů, uvedených v předchozím odstavci, se substituční a permutační transformace používají v moderních šifrách iterovaně. Přiblížíme si dvě základní schémata, jež se dnes při konstrukci šifer využívají.

2.1 Feistelovo schéma

Feistelovo schéma označuje symetrickou konstrukci stavby blokové šifry, pojmenovanou po Německém fyziku a kryptografovi Horstu Feistelovi. Feistel je jedním z průkopníků moderní symetrické kryptografie, během své práce pro IBM navrhl společně s Donem Coppersmithem šifru Lucifer, která se stala základem DES. Celý postup konstrukce blokové šifry se často označuje také jako Feistelova šifra nebo Feistelovská síť.

Feistelovo schéma má tu výhodu, že šifrovací a dešifrovací operace jsou velmi podobné, v některých případech dokonce totožné (v takovém případě vyžadují pouze obrácení seznamu dílčích klíčů). Proto je velikost kódu nebo obvodů, potřebných k provádění této šifry, téměř poloviční oproti jiným postupům.

Feistelovská síť je iterovaná bloková šifra, využívající v každé iteraci (rundě, angl. *round*) vnitřní *rundové funkce* $c = f(m, k)$, jež blok otevřeného textu m a klíč k zkombinuje do kryptogramu c. Tato blíže nespecifikovaná funkce se někdy označuje jako *Feistelova funkce*.

Schematicky je postup šifrování i dešifrování znázorněn na Obrázku 1. Matematicky lze postup šifrování Feistelovým schématem popsat následovně:

Vstupem algoritmu je trojice (m, r, k). Prvním parametrem je blok otevřeného textu $m = (L_0, P_0)$ délky $2b$ bitů, kde L_0 a P_0 označují levou a pravou polovinu bloku. Jak P_0, tak i P_0 mají tedy délku vždy b bitů. Druhým parametrem je počet iterací (rund) $r \in \mathbb{N}$. Třetím parametrem je šifrovací klíč k, z nějž si před začátkem výpočtu blíže nespecifikovaným způsobem odvodíme r podklíčů (dílčích klíčů, rundových klíčů) $k_1, k_2, \ldots, k_r$.

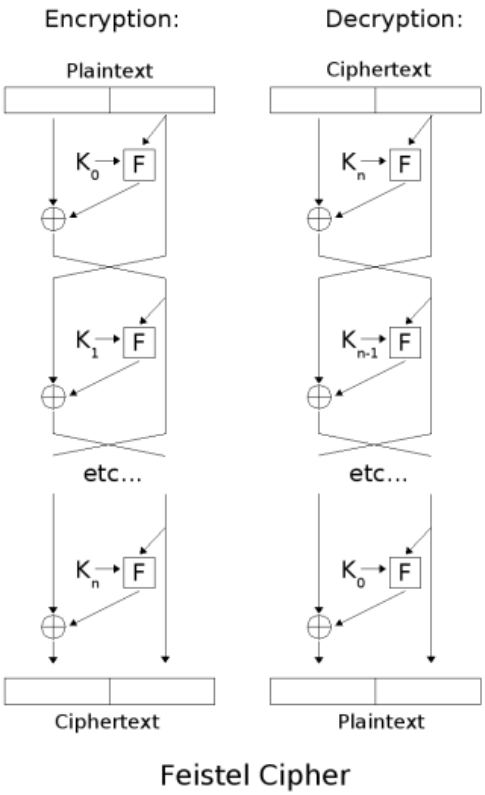
V i-té rundě potom proběhne transformace páru (L_{i-1}, P_{i-1}) na pár (L_i, P_i), přičemž platí (připomínáme, že symbol $\oplus$ označuje binární sčítání bez přenosu, neboli binární XOR)

$$L_i = P_{i-1}, \tag{1}$$
$$P_i = L_{i-1} \oplus f(P_{i-1}, k_i). \tag{2}$$

V tomto popisu označuje $f(P_{i-1}, k_i)$ výše uvedenou Feistelovu funkci, jež v každé rundě používá k šifrování mezivýsledku jiný podklíč.

V každé rundě Feistelovské sítě dojde k záměně levé poloviny šifrovaného textu za pravou a ke složení nové pravé poloviny textu z původní levé poloviny a z transformace pravé poloviny textu rundovou funkcí $f()$. Všimněme si, že Feistelovská síť v každé iteraci změní pouze polovinu šifrovaného bloku, druhá polovina zůstane beze změny.

Výhodou Feistelovy sítě je fakt, že postup šifrování a dešifrování je zcela totožný, liší se pouze pořadím použitých dílčích klíčů: při šifrování používáme sekvenci



Obrázek 1: Feistelova šifra (převzato z Wikipedie).

$\{k_1, k_2, \ldots, k_r\}$, zatímco při dešifrování sekvenci otočíme a použijeme $\{k_r, k_{r-1}, \ldots, k_1\}$. V i-té rundě dešifrování potom proběhne transformace páru (L_i, P_i) na pár (L_{i-1}, P_{i-1}) pomocí

$$P_{i-1} = L_i, \tag{3}$$
$$L_{i-1} = P_i \oplus f(P_{i-1}, k_i). \tag{4}$$

Všimněte si, že Feistelovu funkci $f(P_{i-1}, k_i)$ nebylo třeba pro dešifrování invertovat (ani by to nebylo správně). Znamená to, že v Feistelovské šifře lze použít i takové velmi složité rundové funkce, jejichž inverze neexistuje.

■ **balanced Feistel network (BFN), unbalanced Feistel network (UFN), generalised Feistel network (GFN)** ■

2.2 Substitučně-permutační (SP) síť

Alternativní iterační schéma pro blokové šifry představuje přímý sled operací zmatení a rozptýlení informace v bloku, nazývaný *substitučně-permutační síť*. SP síť je schematicky znázorněná na Obrázku 2. ■ **Zjistit historii a odkud se SP vzalo.** ■

Vstupem tohoto algoritmu je stejně jako u Feistelovy sítě trojice (m, r, k). Prvním parametrem je blok otevřeného textu $m = T_0$ délky b bitů, jenž se na rozdíl od Feistelovské sítě nedělí na dvě poloviny. Druhým a třetím parametrem je opět počet iterací (rund) $r \in \mathbb{N}$ a šifrovací klíč k. Ze šifrovacího klíče si před začátkem výpočtu obdobně jako u předchozího postupu blíže nespecifikovaným způsobem odvodíme r podklíčů $k_1, k_, \ldots, k_r$.

Algoritmus poté v každé iteraci transformuje vstupní blok holého textu T_{i-1} pomocí dílčího klíče k_i a substituční a permutační transformace $c = s(m, k)$ a $c = p(m)$ na blok T_i. Substituční transformace přitom s výhodou využívá S-boxů, jež mohou, ale nemusí záviset na dílčím klíči k_i, a permutační transformaci obstarává nějaký P-box. Jednu iteraci lze potom zapsat jako

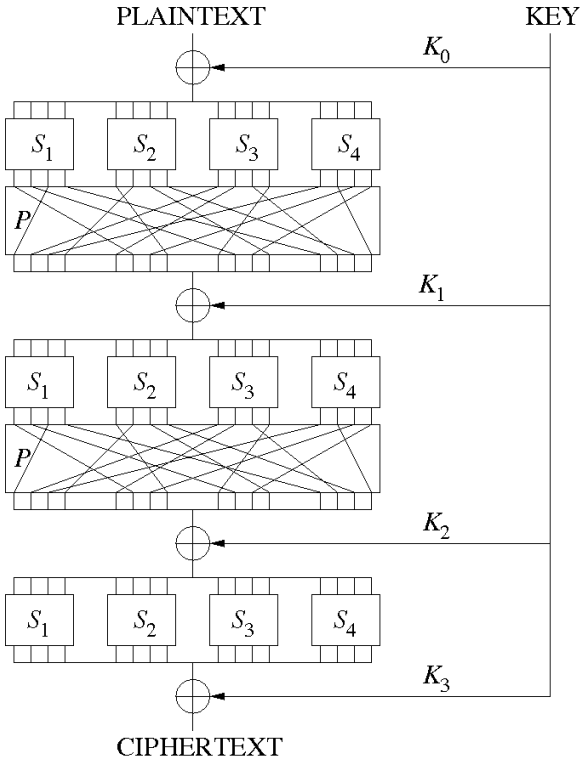
$$T_i = p(s(T_{i-1}, k_i)).$$

V každé rundě se k blok T_i kombinuje s dílčím klíčem k_i pomocí nějaké grupové operace – typicky jde o XOR, takže

$$T_i \leftarrow T_i \oplus k_i.$$

Dílčí klíč se získává ze šifrovacího klíče nejčastěji pomocí jednoduché transformace, například opět pomocí S-boxů a P-boxů).

Proces dešifrování je u substitučně-permutační sítě o něco složitější, než u Feistelovské sítě. V tomto případě je nutné obrátit celý proces šifrování, tedy získat funkce $m = s^{-1}(c, k)$ a $m = p^{-1}(c)$ pomocí inverze S-boxů a P-boxu a poté samozřejmě, stejně jako u Feistelovské sítě, aplikovat dílčí klíče v obráceném pořadí, tedy jako sekvenci $\{k_r, k_{r-1}, \ldots, k_1\}$. Zatímco u Feistelovské šifry by v rundové funkci bylo možno použít jakýkoliv vhodně navržený S-box, pro S-boxy v tomto šifrovacím schématu platí, že musí mít stejný počet bitů na vstupu i na výstupu a



Obrázek 2: SP síť s třemi rundami, šifrující šestnáctibitový otevřený text na šestnáctibitový šifrovaný text. Tato síť používá v každé rundě čtyři substituční boxy S_1, S_2, S_3, S_4 a jeden permutační box P, nezávislé na klíči. (převzato z Wikipedie).

jejich funkce $c = s(m, k)$ musí být bijekcí a tedy invertovatelná. Dešifrovací iterace potom počítá

$$T_{i-1} = s^{-1}(p^{-1}(T_i), k_i).$$

2.3 Srovnání algoritmů

Feistelova šifra s rundovou funkcí $f()$ využívající (neivertovatelných) S-boxů a P-boxu, je poměrně podobná SP síti – jak uvidíme dále při studiu šifry DES, rundová funkce této šifry je v podstatě jednou iterací neinvertovatelné SP sítě. V obou přístupech přesto nalezneme určité rozdíly, díky nimž lze pro určité aplikace výrazněji doporučit jednu či druhou variantu:

1. Jak uvádí [PRB98], pro určitou cílovou entropii signálu má substitučně-permutační síť více „vnitřní podobnosti“, a na současných CPU s velkým počtem výpočetních jader může být počítána výrazně rychleji, než Feistelovo schéma, zatímco jednodušší procesory – například ty, které jsou osazeny na čipových kartách – tohoto paralelismu nemohou využít a je pro ně vhodnější šifra, založená na Feistelovské síti.

2. Feistelovská síť je implementačně jednodušší, obsahuje stejné obvody či kód pro šifrování i dešifrování.

3. Substitučně-permutační síť má při stejném počtu iterací lepší rozptylující vlastnosti, neboť zpracovává celý blok textu najednou.

3 Data Encryption Standard (DES)

Na svou dobu byl DES velmi kvalitní šifrovací mechanismus. Roku 1976 byla tato šifra vybrána ve Spojených státech jako oficiální federální standard zpracování informací (angl. *Federal Information Processing Standard, FIPS*), odkud se později rozšířila do celého světa. Na počátky bylo přijetí DES poměrně vlažné, panovaly obavy z vlastností utajených částí šifry, relativně krátké délky klíče (56 bitů) a přijetí nenapomohlo ani podezření, že Národní agentura pro bezpečnost (angl. *National Security Agency, NSA*) si v programu nechala zadní vrátka, s jejichž pomocí lze snadno dešifrovat jakoukoliv zprávu. Tato podezření měla za následek, že vlastnosti DES se staly objektem velmi pečlivého zkoumání v akademické sféře. Standardizace DES tak nepřímo motivovala výzkum nových metod konstrukce (nejen) blokových šifer a jejich pečlivou kryptoanalýzu.

V současné době se tato šifra pro většinu aplikací již nepovažuje za bezpečnou. Hlavním důvodem je výše zmíněná malá délka klíče – 56 bitů se ukázalo jako opravdu nedostačující a při praktických experimentech byly některé klíče prolomeny za méně, než 24 hodin. Kromě těchto útoků na klíče existují i další analýzy šifry, ukazující na potenciální slabiny celého mechanismu, i když reálné využití těchto slabin se ukazuje jako nepraktické. V současné době se jako bezpečný algoritmus šifrování používá tak zvané 3DES (angl. *Triple DES*, TDES, trojnásobné

DES, teoreticky také úspěšně atakovatelné různými metodami), které je stále více vytlačováno nástupcem DES, jímž je angl. *Advanced Encryption Standard (AES)*.

3.1 Popis činnosti DES

DES je bloková symetrická šifra. Délka bloku je v tomto případě 64 bitů, stejně tak, jako zdánlivá délka klíče – pro klíč se běžně používá osm osmibitových znaků. Ve skutečnosti ovšem DES předpokládá, že v každém bajtu je jeden bit vyhrazený jako kontrola parity, a tyto bity ignoruje. Skutečná délka klíče je tedy pouze 56 bitů.

3.1.1 Struktura algoritmu

Celková struktura algoritmu šifry DES je znázorněna na obrázku 3. Šifrování bloku probíhá v 16 identických iteracích (rundách, angl. *rounds*), lišících se pouze použitým klíčem. Kromě nich je v algoritmu ještě zařazena počáteční permutace (angl. *initial permutation, IP*) a koncová permutace (angl. *final permutation, FP*), přičemž IP je inverzí FP a platí tedy

$$x = \mathrm{FP}(\mathrm{IP}(k)) \quad \text{respektive} \quad m = \mathrm{IP}(\mathrm{FP}(m)).$$

Obě tyto permutace nemají v podstatě žádný reálný kryptografický význam, dostupné prameny usuzují, že jejich hlavní rolí bylo zjednodušit nahrávání bloků dat do tehdejšího hardware. Před hlavními šestnácti rundami je 64-bitový blok textu rozdělen na dvě 32-bitové poloviny a tyto poloviny jsou poté zpracovány Feistelovým schématem pomocí rundové funkce, nazývané DES-funkce.

Feistelovská struktura šifry zajišťuje, že proces šifrování i dešifrování je velmi podobný. Jediným rozdílem je to, že dílčí klíče se při dešifrování aplikují v opačném pořadí, zbytek algoritmu je identický. Tato volba do značné míry zjednodušuje implementaci, zvláště pak v případech, kdy je algoritmus realizován přímo v hardwaru.

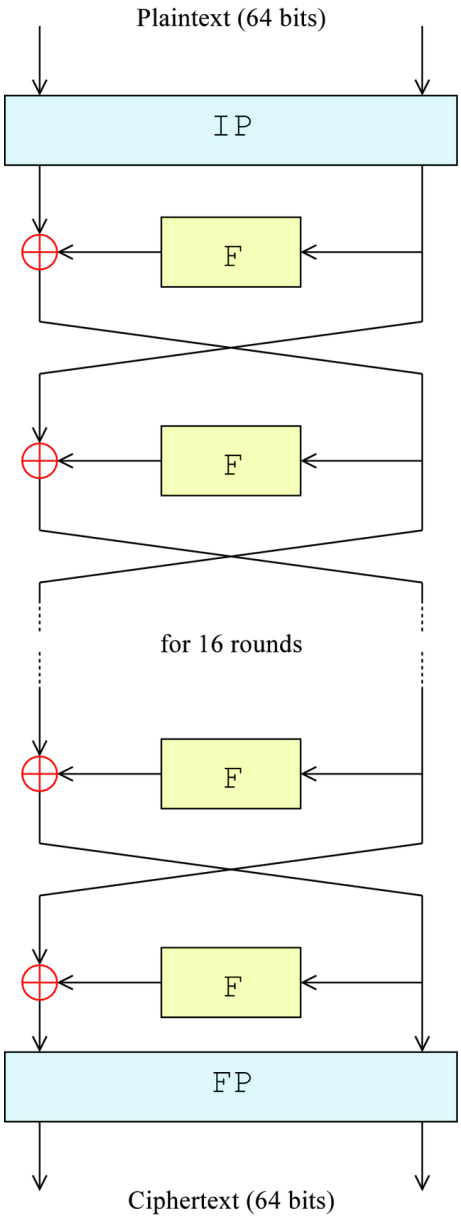
3.1.2 Feistelova funkce $f(x, y)$

Jak jsme již zmínili výše, jádrem šifry DES je tak zvaná rundová nebo Feistelova funkce f ve tvaru

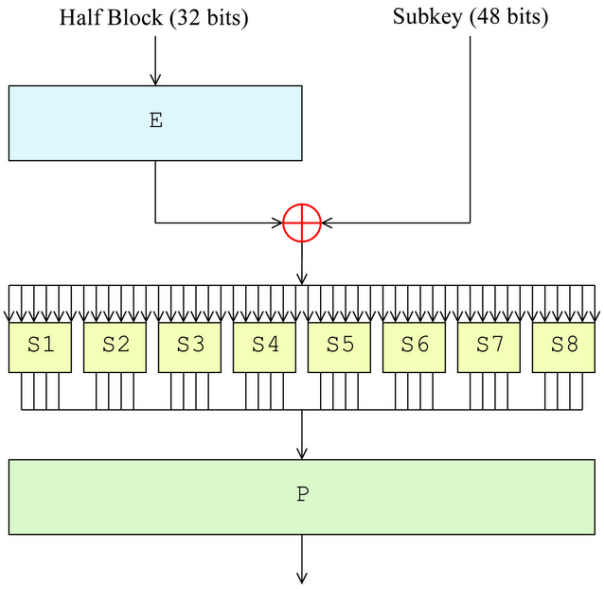
$$\boldsymbol{c} = f(\boldsymbol{m}, \boldsymbol{k}) : (\mathbb{Z}/2\mathbb{Z})^{32} \times (\mathbb{Z}/2\mathbb{Z})^{48} \mapsto (\mathbb{Z}/2\mathbb{Z})^{32}.$$

Tato funkce zpracovává vždy polovinu šifrovaného bloku (tedy jedno 32 bitové slovo $\boldsymbol{m}$) a míchá ji s dílčím rundovým klíčem $\boldsymbol{k}$ postupem, zobrazeným na Obrázku 4, jehož čtyři kroky si nyní popíšeme:

1. *Expanze.* Nejprve se každý 32bitový půlblok $\vec{m}$ rozšíří na délku 48 bitů pomocí expanzní permutace $\boldsymbol{y} = e(\boldsymbol{x})$, dané permutační tabulkou, nazývanou také *E-box*. Touto permutací se zdvojí výskyty některých bitů původního půlbloku.



Obrázek 3: Přehled struktury funkčních bloků algoritmu DES. Převzato z Wikipedie.

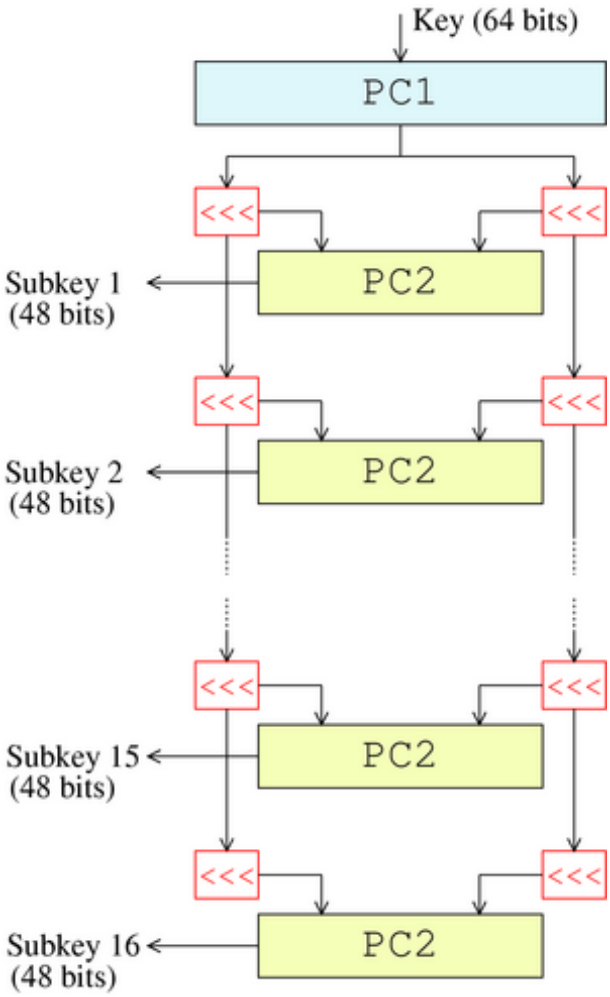


Obrázek 4: Schéma vyhodnocení Feistelovy funkce v algoritmu DES. Převzato z Wikipedie.

2. *Míchání klíčů.* Výsledek expanzního kroku se zkombinuje s *dílčím klíčem* $\boldsymbol{k}$ prostým přičtením bez přenosu, které je v binární aritmetice rovno exkluzivnímu OR (XOR). Smíchaný blok $\boldsymbol{b} \in (\mathbb{Z}/2\mathbb{Z})^{48}$ je nyni tvořen osmi šestibitovými bloky $\boldsymbol{b}_1$, $\boldsymbol{b}_2$, $\boldsymbol{b}_3$, $\boldsymbol{b}_4$, $\boldsymbol{b}_5$, $\boldsymbol{b}_6$, $\boldsymbol{b}_7$, $\boldsymbol{b}_8$, přičemž $\boldsymbol{b}_i \in (\mathbb{Z}/2\mathbb{Z})^6$, a odpovídá transformaci
$$\boldsymbol{b} = e(\boldsymbol{m}) \oplus \boldsymbol{k}$$
■ Podklíčů je šestnáct, pro každé kolo jeden, mají délku 48 bitů a jsou odvozeny od hlavního klíče algoritmem popsaným níže. ■

3. *Substituce.* Každý z šestibitových bloků $\vec{b}_i$ je zpracován nezávislou neinvertovatelnou substituční transformací pomocí odpovídajícího S-boxu. Každý z osmi S-boxů s_i nahradí vstupní šestibitové číslo $\vec{b}_i$ předem zvolenou čtyřbitovou hodnotou
$$\boldsymbol{u}_i = s_i(\boldsymbol{b}_i).$$
Transformace v S-boxu je nelineární, uskutečňuje se pomocí vyhledávacích tabulek. Tato část je jádrem DES, bez ní by šlo o poměrně triviálně zlomitelnou afinní šifru.

4. *Permutace.* Na závěr se dílčí výsledky substituce, tedy čtyřbitová slova $\boldsymbol{u}_1$ až $\boldsymbol{u}_8$, spojí a výsledný 32 bitový výstup S-boxů se přerovná pevně danou permutací hodnot,
$$\boldsymbol{c} = p(\boldsymbol{u}).$$
Tato permutace je dána pevně danou permutační tabulkou, P-boxem.



Obrázek 5: Generování dílčích klíčů v algoritmu DES. Převzato z Wikipedie.

■ **Toto se již zmiňovalo v popisu Feisetlovské sítě i SPN** ■ The alternation of substitution from the S-boxes, and permutation of bits from the P-box and E-expansion provides so-called "confusion and diffusion"respectively, a concept identified by Claude Shannon in the 1940s as a necessary condition for a secure yet practical cipher.

3.1.3 Generování dílčích klíčů

V každé rundě šifry DES probíhá výpočet Feistelovy funkce $f()$ nejenom nad jiným slovem $\boldsymbol{m}$, používá se také jiný dílčí klíč $\boldsymbol{k}_i$. Na Obrázku 5 je znázorněn postup, jímž se tyto dílčí rundové klíče odvozují z původního šifrovacího klíče $\boldsymbol{k}$. V angličtině se tento postup označuje jako *rozvrh klíčů* (angl. *key schedule*).

Jak jsme již zmínili v úvodu, šifrovací klíč $\boldsymbol{k}$ má u DES transformace délku 56 bitů. Tento klíč vznikne z osmi bajtů uživatelského šifrovacího klíče $\boldsymbol{k}_{64}$ vynecháním paritních bitů a permutací bitů ostatních v P-boxu, označovaném jako PC-1 (angl. .Permuted Choice 1). 56bitový výstup z PC-1 je vzápětí rozdělen na dvě 28bitová

slova, z nichž každé je v dalších krocích zpracováváno samostatně. V jednotlivých rundách jsou tato slova rotována doleva o jeden až dva bity a z každého zrotovaného 28bitového slova je vybráno 24 bitů transformací v P-boxu, označovaném jako PC-2 (angl. .Permuted Choice 2). Výsledkem je 48bitový dílčí klíč $\boldsymbol{k}_i$. Rotace doleva, v diagramu na Obrázku 5 označené jako „<<<“, způsobí, že v každém dílčím klíči $\boldsymbol{k}_i$ je použita jiná podmnožina bitů původního klíče, přičemž každý bit původního klíče se vyskytuje na nějaké pozici průměrně v 7/8 všech klíčů.

Dílčí klíče pro dešifrování je třeba generovat v opačném pořadí – opět se přitom využívá symetrie algoritmu, tentokrát jde o symetrii v rozvrhu klíčů: Z původního šifrovacího klíče $\boldsymbol{k}$ se dílčí klíče generují téměř identickým způsobem, jako při šifrování, jediný rozdíl je v tom, že místo operace levé rotace se použije rotace pravá.

3.2 Teoretické slabiny

Jedno ze slabých míst šifry, které ji podle některých názorů v dnešní době činí nepoužitelnou a sráží ji pod úroveň dnešních standardních šifer, jsou tak zvané *slabé klíče*, což jsou takové klíče, pro něž platí

$$\mathfrak{E}_{\boldsymbol{k}}(\mathfrak{E}_{\boldsymbol{k}}(\boldsymbol{m})) = \boldsymbol{m}$$

pro všechna 64bitová slova $\boldsymbol{m} \in \{0,1\}^{64}$. Tyto klíče má DES čtyři ve tvaru

$$\boldsymbol{k} \in \{(0^{28}, 0^{28}), (1^{28}, 1^{28}), (0^{28}, 1^{28}), (1^{28}, 0^{28}) \in \mathbb{Z}^{28} \times \mathbb{Z}^{28}\},$$

přičemž exponent udává počet opakování mocněné binární číslice. Použijeme-li tyto klíče, šifrovací i dešifrovací transformace jsou identické, je tedy jedno, jestli šifrujeme nebo dešifrujeme. Tyto klíče nesmí být v reálném provozu použity.

Další skupinu nevhodných klíčů tvoří *poloslabé klíče*. Jde o páry klíčů $(\boldsymbol{k}_1, \boldsymbol{k}_2)$, jež, ač rozdílné, dešifrují text, zašifrovaný druhým klíčem v páru, tedy

$$\mathfrak{E}_{\boldsymbol{k}_1}(\mathfrak{E}_{\boldsymbol{k}_2}(\boldsymbol{m})) = \boldsymbol{m}$$

Těchto párů klíčů je celkem šest:

$$\begin{aligned}(\boldsymbol{k}_1, \boldsymbol{k}_2) \in \Big\{ &\Big((01^{14}, 01^{14}), (10^{14}, 10^{14})\Big), \Big((01^{14}, 10^{14}), (10^{14}, 01^{14})\Big), \\ &\Big((01^{14}, 0^{28}), (10^{14}, 0^{28})\Big), \Big((01^{14}, 1^{28}), (10^{14}, 1^{28})\Big), \\ &\Big((0^{28}, 01^{14}), (0^{28}, 10^{14})\Big), \Big((1^{28}, 01^{14}), (1^{28}, 10^{14})\Big)\Big\}\end{aligned} \tag{5}$$

Postupným použitím každého z těchto párů klíčů obdržíme z otevřeného textu kryptogram, jehož znění je totožné se zněním šifrovaného textu. Tyto páry generují jenom dva rozdílné rundové klíče, z nichž každý je potom použit celkem osmkrát v jednotlivých iteracích DES transformace. I těmto klíčům je třeba se v reálném provozu vyhnout.

Další relativní slabinou DES šifry je její *komplementarita*. Ke slovu $\boldsymbol{x}$ vytvoříme jeho bitový doplněk $\overline{\boldsymbol{x}}$ (komplement, bitovou inverzi) tak, že všechny původní nuly

nahradíme jedničkami a naopak. Bude tedy platit $\overline{\boldsymbol{x}} \oplus \boldsymbol{x} = 0$. DES šifra potom splňuje rovnici

$$\overline{\mathfrak{E}_{\boldsymbol{k}}(\boldsymbol{m})} = \mathfrak{E}_{\overline{\boldsymbol{k}}}(\overline{\boldsymbol{m}})\,.$$

Pokud zašifrujeme bitový doplněk původního 64bitového bloku bitovým doplňkem klíče, obdržíme bitový doplněk kryptogramu, jenž by vznikl použitím původního klíče a textu. Znamená to, že inverze šifrované informace vytvoří inverzi šifrového textu a že v případě útoku na šifru pomocí voleného otevřeného textu (angl. *chosen plaintext attack, CPA*) stačí namísto prozkoumání celého klíčového prostoru 2^{56} klíčů zkoumat pouze jeho polovinu, tedy 2^{55} klíčů.

3.3 Alternativy DES

Jak jsme již zmínili v úvodu, DES je sice z dnešního hlediska slabou šifrou, je ale možné ji pozměnit a použít znovu jako základ bezpečnějších šifrovacích schémat. Uživatelé, kteří dříve používali DES, dnes často používají algoritmus 3DES (*Triple DES*), spočívající v třech cyklech běžného DES s různými klíči. 3DES v současnosti považujeme za přiměřeně bezpečnou, ovšem z hlediska rychlosti šifrování poměrně pomalou šifru.

Výpočetně méně náročnou a přitom také značně bezpečnější variantou je DESX, jejímž autorem je Ron Rivest (tedy „R“ z asymetrické šifry RSA). Tato šifra zvyšuje délku klíče doplněním přídavných klíčů, kombinovaných s otevřeným textem před a s kryptogramem po DES šifře pomocí logického exkluzivního OR (XOR). Pro tři 64bitové klíče $(\boldsymbol{k}_1, \boldsymbol{k}_2, \boldsymbol{k}_3)$ šifrujeme v DESX tak, že k otevřeném textu $\boldsymbol{m}$ ještě před šifrováním DES šifrou přičteme klíč $\boldsymbol{k}_3$, poté celé 64bitové slovo zašifrujeme klíčem $\boldsymbol{k}_2$ a nakonec ke kryptogramu přičteme klíč $\boldsymbol{k}_1$. Šifrovací transformace má tedy tvar

$$\mathfrak{E}_{(\boldsymbol{k}_1,\boldsymbol{k}_2,\boldsymbol{k}_3)} = \boldsymbol{k}_1 \oplus \mathfrak{E}_{\mathrm{DES},\boldsymbol{k}_2}(\boldsymbol{k}_3 \oplus \boldsymbol{m}).$$

Dalšími možnými náhradami DES, jež původní šifru přímo nevyužívají, jsou například RC5, Blowfish, nebo IDEA.

4 Advanced Encryption Standard (AES)

■ Doplnit něco o historii AES ■

Šifru vyvinuli dva belgičtí kryptologové, Joan Daemen a Vincent Rijmen, a do výběrového řízení na AES ji poslali pod názvem „Rijndael“, složeninou z příjmení obou autorů.

AES je, na rozdíl od svého předchůdce (DES) využívajícího Feistelovské schéma, substitučně-permutační šífra. AES je velice rychlé i v čistě softwarové verzi, je relativně jednoduše implementovatelné a má nízké nároky na paměť. Vzhledem k tomu, že celá specifikace AES je veřejně dostupná, jde v současné době jde o velmi oblíbenou a široce používanou šifru.

4.1 Popis činnosti AES

Zcela přesně vzato není AES přesně Rijndael, neboť původní návrh šifry počítal s volitelnou délkou bloku a větší množinou délky přípustných klíčů: zatímco Rijndael může být použit s délkou bloku i klíče rovnou násobku 32 bitů, ale nejméně 128 a nejvýše 256 bitů[1], AES je konzervativnější – má pevnou délku bloku 128 bitů a klíč o délce 128, 192 nebo 256 bitů. V praxi se ale oba názvy většinou zaměňují.

4.1.1 Stav

Vstupem a výstupem AES je vektor bajtů, rundová transformace substitučně-permutační sítí operuje s mezivýsledkem, jejž nazýváme *stav*. Stav $\boldsymbol{S}$ může být znázorněn jako matice bajtů o rozměrech 4 řádky a n_{bb} sloupce, přičemž n_{bb} je počet 32bitových slov v bloku. Pro AES je délka bloku pevně dána jako 128 bitů, je proto $n_{\mathrm{bb}} = 4$ a stav je reprezentován maticí 4×4 bajty. V případě šifry Rijndael mohou do stavu přibýt další sloupce, velikost stavové matice je potom proměnná od 4×4 do 4×8 bajtů. Stav je konstruován a uložen po sloupcích (v angličtině se tomu říká angl. *column-major format*), šestnáctibajtové slovo $\boldsymbol{s} = (\boldsymbol{s}_0, \boldsymbol{s}_1, \ldots, \boldsymbol{s}_{15})$ je tedy v matici stavu uloženo jako šestnáct osmibitových slov $\boldsymbol{s}_0$, $\boldsymbol{s}_1$ až $\boldsymbol{s}_{15}$ ve tvaru

$$\boldsymbol{S} = \begin{pmatrix} \boldsymbol{s}_0 & \boldsymbol{s}_4 & \boldsymbol{s}_8 & \boldsymbol{s}_{12} \\ \boldsymbol{s}_1 & \boldsymbol{s}_5 & \boldsymbol{s}_9 & \boldsymbol{s}_{13} \\ \boldsymbol{s}_2 & \boldsymbol{s}_6 & \boldsymbol{s}_{10} & \boldsymbol{s}_{14} \\ \boldsymbol{s}_3 & \boldsymbol{s}_7 & \boldsymbol{s}_{11} & \boldsymbol{s}_{15} \end{pmatrix}.$$

Hodnoty $\boldsymbol{s}_0$, $\boldsymbol{s}_1$ až $\boldsymbol{s}_{15}$ chápeme a označujeme stále jako vektory délky 8 bitů.

4.1.2 Průběh šifrování

Veškeré aritmetické operace, popsané níže, probíhají, pokud není řečeno jinak, v *Rijndaelovském konečném tělese*, což je Galoisovské těleso $\mathbf{GF}(2^8)$ s generujícím polynomem $g(x) = x^8+x^4+x^3+x+1$. Důvod pro použití zrovna tohoto generujícího polynomu není nikde ani pány Daemenem či Rijmenem přesně uveden (existuje celkem 30 generujících polynomů v $\mathbf{GF}(2^8)$, z toho 16 je ireducibilních). Panuje všeobecné přesvědčení, že by bylo možné zvolit jakýkoliv jiný ireducibilní polynom a po vhodné úpravě některých stavebních bloků by šifra měla stále stejné vlastnosti. Z implementačního hlediska je vhodné, aby polynom měl nízkou Hammingovu váhu (obvykle se udává 3 nebo 5) a co nejnižší mocniny x. Bližší informace lze nalézt v [Nyb94] a [DR99, strana 26].

Šifrování otevřeného textu na kryptogram šifrou AES začíná vložením otevřeného textu do stavové matice $\boldsymbol{S}$, následovaném operací *AddRoundKey*, tedy přimícháním dílčího rundovního klíče ke stavu. Poté proběhne celkem $N_{\mathrm{r}} - 1$ rund iterací, tvořených následujícími operacemi:

[1] Jak autoři šifry uvádějí v [DR02], bylo by možné zvětšit velokost bloku i délku klíče nad 256 bitů, ale v době návrhu tak dlouhé klíče a bloky nebyly potřeba (a patrně nejsou třeba ani v době, kdy vzniká tento text – vzhledem k zatím marným útokům i na zjednodušené varianty AES).

1. *SubBytes* – Nejprve jsou všechny bajty stavové matice transformovány nelineární substituční transformaci invertovatelným S-boxem. V celé šifře se používá jenom jediný S-box, implementovaný většinou jako vyhledávací tabulka ve tvaru matice 16×16 prvků, indexované pomocí dolního a horního nibble transformovaného bajtu. Tento krok je jedinou nelinearitou v celé šifře.

   Na rozdíl od DES je v tomto případě substituční tabulka odvozena algebraicky z inverze v Rijndaelovském konečném tělese kombinovaném s invertovatelnou afinní transformací[2], což má zabránit jednoduchým útokům, které by využívaly vlastností použitého konečného tělesa. Substituční transformace nemá žádné pevné body (neexistuje tedy $x = s(x)$, jedná se o tak zvanou ■ **derangement – dismutaci?** ■) a nemá ani tak zvané opačné pevné body, tedy body kde $\overline{x} = s(x)$.

   Krok *SubBytes* nahradí ve stavové matici každou 8bitovou hodnotu $\boldsymbol{s}_i = (s_{i,0}, s_{i,1}, \ldots, s_{i,7})$ hodnotou $\boldsymbol{b}_i = \mathrm{sbox}(\boldsymbol{s}_i)$, přičemž

$$\mathrm{sbox}(\boldsymbol{s}_i) = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \cdot \boldsymbol{s}_i^{-1} \oplus \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

   Hodnota $\boldsymbol{x}^{-1}$ označuje multiplikativní inverzi prvku $\boldsymbol{x}$ v použitém konečném tělese. Vzhledem k tomu, že možných hodnot prvků je pouze 255 (nulový prvek inverzi nemá, autoři mu proto jako inverzi přiřadili sebe sama), lze celý výsledek předpočítat do tabulky 16×16 prvků, zmíněné výše. ■ **vysvětlit, jak se to váže ke generujícímu polynomu a že offset je 0x63. možná algoritmus** ■

2. *ShiftRows* – V tomto kroku probíhá cyklická bajtová transpozice jednotlivých řádkových vektorů stavové matice. Vzhledem k tomu, že stavová matice AES má jenom čtyři sloupce, je nasnadě, že použité rotace budou $0, 1, 2$ a 3 bajty. V tomto pořadí se také posuny aplikují na jednotlivé řádky matice stavu: první řádek zůstává nezměněn, druhý řádek je zrotován doleva o jeden bajt, druhý řádek o dva a třetí o tři bajty. Budeme-li uvažovat popis matice stavu, uvedený výše, lze celou opraci zapsat jako

$$r(\boldsymbol{S}) = \begin{pmatrix} \boldsymbol{s}_0 & \boldsymbol{s}_4 & \boldsymbol{s}_8 & \boldsymbol{s}_{12} \\ \boldsymbol{s}_5 & \boldsymbol{s}_9 & \boldsymbol{s}_{13} & \boldsymbol{s}_1 \\ \boldsymbol{s}_{10} & \boldsymbol{s}_{14} & \boldsymbol{s}_2 & \boldsymbol{s}_6 \\ \boldsymbol{s}_{15} & \boldsymbol{s}_3 & \boldsymbol{s}_7 & \boldsymbol{s}_{11} \end{pmatrix}.$$

[2] Tedy transformací ve formě $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} \oplus \boldsymbol{b}$.

3. *MixColumns* – Tento krok zajišťuje další rozptýlení informace pomocí *kostkové permutace* (angl. *bricklayer permutation*) matice stavu po jednotlivých sloupcích. ■ **překlad? pro vysvětlení je třeba napřed definovat bricklayer function a S-box a D-box** ■ Sloupce stavu jsou považovány opět za polynomy, jejichž koeficienty jsou prvky Rijndaelovského tělesa $\mathbf{GF}(2^8)$ ■ **podle knihy ale jsou to polynomy v GF(), to nedává smysl, zřejmě tisková chyba** ■ a násobeny v aritmetice modulo $x^4 + 1$ s pevně daným polynomem $c(x)$. Vzhledem k tomu, že jedním z požadavků při návrhu tohoto kroku bylo, aby nebyl příliš výpočetně náročný pro 8bitové procesory, koeficienty tohoto polynomu mohou nabývat pouze hodnot $0, 1, \ldots, 3$. Násobení koeficientů polynomu hodnotami 0 a 1 má nulové výpočetní nároky, násobení hodnotou 2 (tedy x) je v Rijndaelovském konečném tělese realizovatelné jednoduchou rutinou, nejčastěji opět pomocí vyhledávací tabulky, a násobení hodnotou 3 (tedy $x + 1$) odpovídá násobení 2 a přičtení původního operandu pomocí XOR. Konkrétní hodnoty koeficientů v polynomu $c(x)$ jsou výsledkem poměrně komplikovaného procesu analýzy rozptylu informace, popsaného detailně v [DR02]. Výsledný polynom má tvar

$$c(x) = 3\,x^3 + x^2 + x + 2.$$

Každý ze čtyř sloupců matice stavu potom projde transformací

$$d(x) \equiv b(x) \cdot c(x) \pmod{x^4 + 1},$$

kterou lze zapsat možná přehledněji v maticovém zápisu

$$\boldsymbol{d}_j = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 01 \end{pmatrix} \cdot \boldsymbol{b}_j \quad \forall j \in \{0, 1, 2, 3\}.$$

4. *AddRoundKey* – posledním krokem každé iterace je přičtení rundovního klíče.

4.1.3 Generování dílčích klíčů

Proces tvorby dílčích klíčů sestává ze dvou na sebe navazujících kroků: Nejprve je původní klíč k expandován tak, aby výsledný expandovaný klíč k_{exp} obsahoval dostatečný počet bitů pro celý proces šifrování. Délka expandovaného klíče je $8N_{\text{b}}(N_{\text{r}}+1)$, neboť počet použitých rundovních klíčů je o jedničku vyšší, než počet rund šifry. Poté je z expandovaného klíče vybrán v každé iteraci odpovídající dílčí klíč k_i.

Podobně jako při operaci *MixColumns*, i při procesu generování dílčích klíčů kladli autoři důraz na to, aby celé schéma přípravy klíčů bylo implementovatelné i na jednoduchých 8bitových platofrmách. Z toho důvodu je expanzní schéma navrženo po bajtech.

Během expanze je původní klíč, vložený po sloupcích do matice 4×4 (pro 128bitový klíč) až 4×8 (pro 256bitový klíč) bajtů expandován do matice $4 \times$

$N_\mathrm{b}(N_\mathrm{r}+1)$. Rundovní klíč $\boldsymbol{K}_i$ je potom tvořen sloupci $N_\mathrm{b}\cdot i$ až $N_\mathrm{b}\cdot(i+1)-1$ této matice ■ **obrázek** ■.

Vlastní expanzní funkce závisí na hodnotě N_k. V každém případě je prvních N_k sloupců matice $\boldsymbol{K}_\mathrm{exp}$ tvořeno jednotlivými bajty původního klíče $\boldsymbol{k}$. Následující sloupce j jsou potom rekurzivně odvozeny z prvních N_k sloupců použitím hodnot ve sloupcích $j-1$, $j-N_\mathrm{k}$ a rundovní konstanty c_m. Rekurze přitom závisí na pozici sloupce:

$$k_j = \begin{cases} k_{j-N_\mathrm{k}} \oplus k_{j-1} & \text{pokud } j \neq m\cdot N_\mathrm{k}\text{respektive } N_\mathrm{k}>6 \wedge j \neq m\cdot N_\mathrm{k}+4 \\ k_{j-N_\mathrm{k}} \oplus f_\mathrm{exp}(k_{j-1}) & \text{jinak.} \end{cases}$$

Nelineární transformační funkce $f_\mathrm{exp}()$ spočívá v aplikaci Rijndaelovského S-boxu na prvky argumentu, cyklické rotaci bajtů celého sloupce o jednu pozici a přičtení konstanty. Rundovní konstanta nezávisí na velikosti klíče a je v Rijndaelovském tělese definována jako

$$c_m = x^{m-1},$$

tedy $c_0 = x^0 = 01$, $c_1 = x^1 = 02$ a tak dále.

Bližší popis včetně pseudokódu operací lze najít v knize pánů Daemena a Rijmena [DR02].

5 Operační módy blokových šifer

5.1 Mód elektronické kódové knihy (ECB)

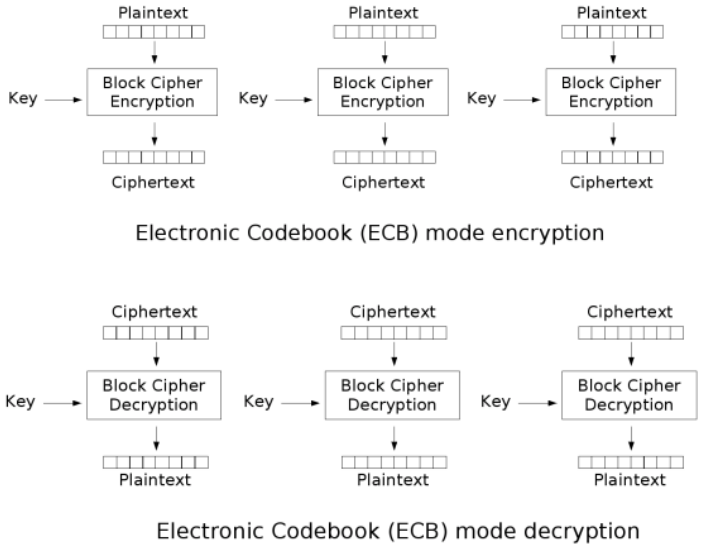
Nejjednodušší operační mód. Jak je znázorněno na Obrázku 6, každý otevřený text $\boldsymbol{m}_j$ je zašifrován stejným klíčem $\boldsymbol{k}$ na kryptogram $\boldsymbol{c}_j$. Platí

$$\mathfrak{E}_{\boldsymbol{k}}(\boldsymbol{m}_j) = \boldsymbol{c}_j, \quad \mathfrak{D}_{\boldsymbol{k}}(\boldsymbol{c}_j) = \boldsymbol{m}_j.$$

Hlavní výhodou ECB módu je jeho odolnost vůči chybám při přenosu dat: Synchronizace mezi odesilatelem a příjemcem není nutná, pokud přijemce nepřijme některý z vysalných bloků, stále je schopen dešifrovat bloky následující. Podobné je to s bitovými chybami – pokud se vlivem šumu některé bity vysílané zprávy při příjmu dekódují špatně, tato chyba ovlivňuje pouze přijatý blok. Šifrování lze efektivně paralelizovat, což u většiny dalších operačních módu není možné.

Základní problém tohoto postupu je fakt, že ECB mód šifruje vysoce deterministickým způsobem, kdy stejný text bude vždy zašifrován na stejný kryptogram, pokud nezměníme během vysílání šifrovací klíč. Jedná se tedy opravdu o jakousi obrovskou kódovou knihu, předepisující, jaký kryptogram odpovídá zadanému textu. To vede ke dvěma velmi nepříznivým důsledkům, jež v podstatě znemožňují jakékoliv použití tohoto přístupu pro šifrování běžné komunikace. Prvním důsledkem je, že protivník z šifrovaného provozu pozná, jestli byla nějaká část zprávy poslána jednou či vícekrát a může tak například sledovat změny ve struktuře zpráv. Kromě toho celý postup umožňuje i přesouvat kousky kryptogramu či doplňovat kusy do původní zprávy kusy nové bez toho, aby příjemce poznal, že zpráva byla modifikována.

Tento mód je bezpečný pouze pro velmi krátké zprávy do délky jednoho bloku.



Obrázek 6: Mód elektronické kódové knihy (ECB). Převzato z Wikipedie

5.2 Módy zřetězení šifrového textu (CBC,PCBC)

Nevýhodu módu ECB lze odstranit tak, že budeme šifrovanou zprávu nějakým způsobem modifikovat tak, aby stejný otevřený text nevedl na stejný kryptogram. Jednou z možností je přičítat k textu ještě předešlý kryptogram, tedy

$$\boldsymbol{c}_j = \mathfrak{E}_{\boldsymbol{k}}(\boldsymbol{c}_{j-1} \oplus \boldsymbol{m}_j), \quad \boldsymbol{m}_j = \mathfrak{D}_{\boldsymbol{k}}(\boldsymbol{c}_j) \oplus \boldsymbol{c}_{j-1}.$$

Tento postup, kdy „zřetězíme“ postup šifrování bloků otevřeného textu, dostatečným způsobem zakrývá vazbu mezi otevřeným textem a kryptogramem i pro dlouhoé zprávy a podstaným způsobem redukuje data, která může kryptoanalytik využít k případnému útoku na šifru. Pro fungování tohoto schématu je ovšem nutné, aby obě strany šifrovaného spojení měly na začátku k dispozici nějakou náhodnou hodnotu *inicializačního vektoru* $\boldsymbol{i} = \boldsymbol{c}_0$.

Problém volby a distribuce inicializačního vektoru $\boldsymbol{i}$ není jednoduše řešitelný. Nejčastěji navrhovaným řešením je ukládat $\boldsymbol{i}$ spolu s klíčem, tedy efektivně prodloužit délku klíče o $\boldsymbol{i}$. Toto ale není zcela optimální řešení. ■ **doplnit inicializaci klíče** ■

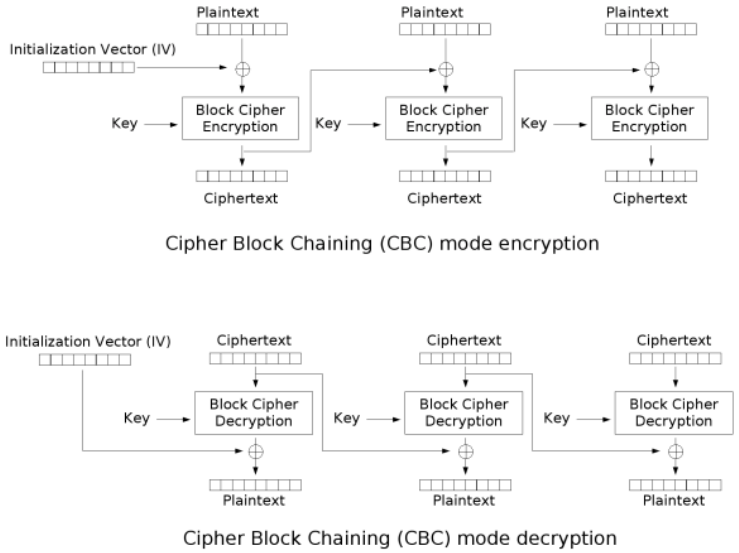
■ **zmínit Klímovy slajdy - útok na CBC** ■

5.3 Zpětnovazební módy (CFB, OFB)

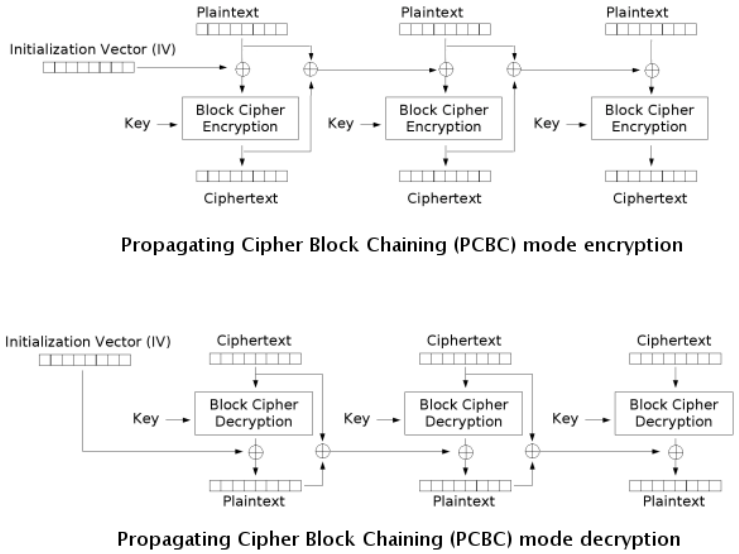
V operačním módu zpětné vazby ze šifrového textu CFB nastavíme $\boldsymbol{c}_0 = \boldsymbol{i}$ a potom

$$\boldsymbol{c}_j = \boldsymbol{m}_j \oplus \mathfrak{E}_{\boldsymbol{k}}(\boldsymbol{c}_{j-1}), \quad \boldsymbol{m}_j = \boldsymbol{c}_j \oplus \mathfrak{E}_{\boldsymbol{k}}(\boldsymbol{c}_{j-1}).$$

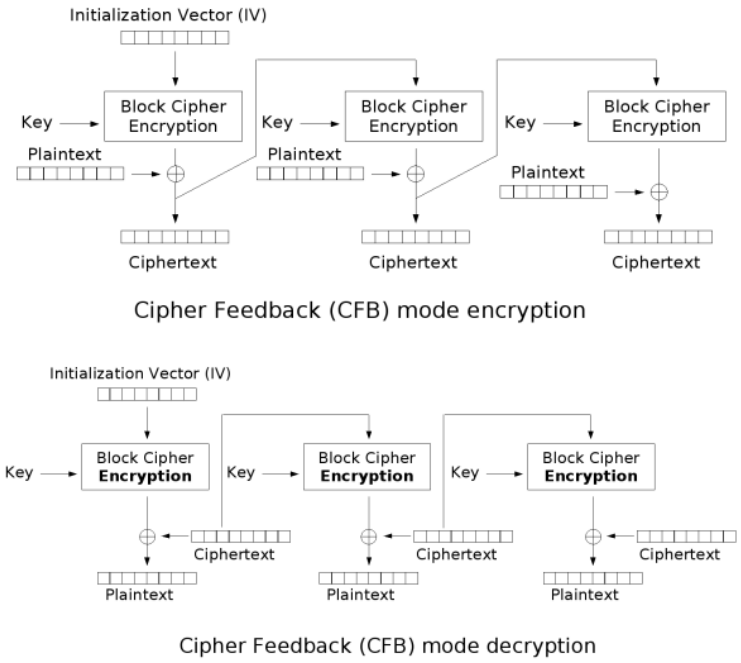
Princip tohoto módu ukazuje Obrázek 9.



Obrázek 7: Mód zřetězení šifrového textu (CBC). Převzato z Wikipedie.



Obrázek 8: Mód zmnoženého řetězení šifrovaného textu (PCBC). Převzato z Wikipedie.



Obrázek 9: Mód zpětné vazby ze šifrového textu (CFB). Převzato z Wikipedie.

Změna jednoho vstupního bloku $\boldsymbol{m}_j$ vede pouze na změnu $\boldsymbol{c}_j, \boldsymbol{c}_{j+1}, \ldots$, šíří se dále obdobně, jako v případě CBC. Oba zmíněné módy lze použít jako MAC. ■ **doplnit českou terminologii a bližší informace** ■

Mód zpětné vazby z výstupu (OFB) nastaví nejprve z inicializačního vektoru $\boldsymbol{k}_0 = \boldsymbol{i}$ a počítá postupně

$$\boldsymbol{k}_j = \mathfrak{E}_{\boldsymbol{k}}(\boldsymbol{k}_{j-1}), \quad \boldsymbol{c}_j = \boldsymbol{m}_j \oplus \boldsymbol{k}_j, \quad \boldsymbol{m}_j = \boldsymbol{c}_j \oplus \boldsymbol{k}_j.$$

Princip tohoto módu ukazuje Obrázek 10.

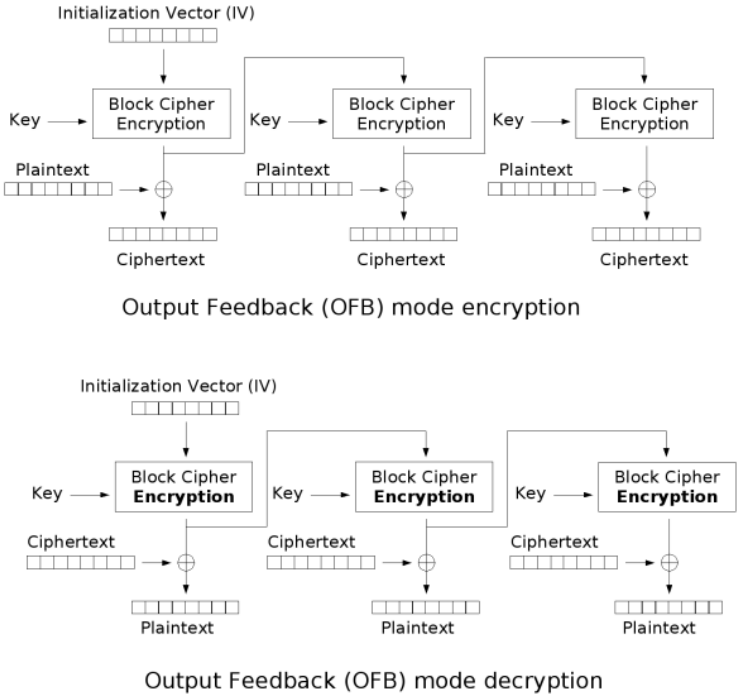
V OFB módu generuje šífra *pseudonáhodný proud klíčů* (angl. *keystream*). Inicializační vektor musí být nághodný nebo generovaný jako nonce stejně, jako v CBC módu.

Změna jednoho vstupního bloku $\boldsymbol{m}_j$ vede pouze na změnu $\boldsymbol{c}_j$, nešíří se dále. ■ **doplnit českou terminologii a bližší informace** ■

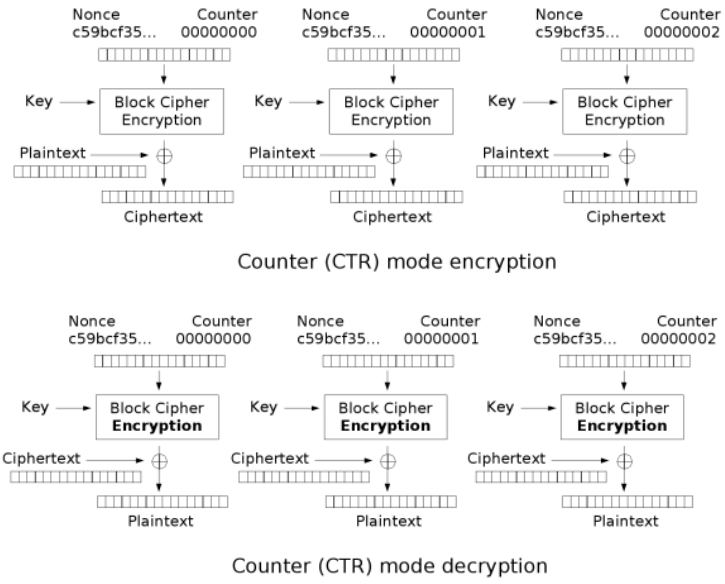
5.4 Čítačový mód (CTR)

Tento mód je dalším, jenž přetváří blokovou šifru v šifru proudovou. Obdobně jako u OFB a CFB, i zde je proud klíče generován po blocích. Vstupem blokové šifry je čítač, jehož hodnota se mění s každým nově zašifrovaným blokem. Princip tohoto módu ukazuje Obrázek 11.

CTR má podobné vlastnosti, jako OFB, navíc ovšem umožňuje dešifrování zvolené části kryptogramu bez nutnosti dešifrovat přescházející bloky. CTR je proto



Obrázek 10: Mód zpětné vazby z výstupu (OFB). Převzato z Wikipedie.



Obrázek 11: Čítačový mód (CTR). Převzato z Wikipedie.

velmi vhodný pro vícejádrové či víceprocesorové počítače, na nichž lze nezávislé bloky šifrovat a dešifrovat paralelně.

V tomto módu je třeba velmi obezřetně inicializovat vstup do blokové šifry, abcyhom zabránili použití stejné hodnoty čítače dvakrát. Pokud by k tomu došlo a protivník by měl k dispozici jeden ze dvou otevřených textů, zašifrovaných stejným klíčem, mohl by získat zpět celý blok proudového klíče a dešifrovat i druhou zprávu. Jeden z často používaných způsobů, naznačených i na Obrázku 11, je tento:

1. Předpokládejme, že bloková šifra pracuje s blokem velikosti b bitů (například 128).

2. Zvolíme si $\boldsymbol{i}$, který je nonce délky b_1 (například 96) bitů.

3. Zbylých b_2 bitů (v našem případě 32) použijeme pro čítač, jenž inicializujeme na nulu.

4. Se zvoleným $\boldsymbol{i}$ a tímto čítačem jsme schopni zašifrovat celkem 2^{32} osmibajtových bloků, tedy $8 \cdot 2^{32}$, přibližně 32 GiB dat. Poté je třeba vygenerovat nové $\boldsymbol{i}$.

Poznamejeme ještě, že počáteční hodnotu čítače pro CTR mód nemusíme držet v tajnosti.

Reference

[DR99] Joan Daemen and Vincent Rijmen. AES submission document on Rijndael, version 2, amended. [online], sep 1999.

[DR02] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES – The Advanced Encryption Standard*. Springer, 2002.

[Mer91] Ralph C Merkle. Fast software encryption functions. In *Advances in Cryptology-CRYPT0'90*, pages 477–501. Springer, 1991.

[Mol01] Richard A. Mollin. *An Introduction to Cryptography*. Chapman & Hall/CRC, 2001.

[Mol07] Richard A. Mollin. *An Introduction to Cryptography*. Taylor & Francis, 2nd edition, 2007.

[Nyb94] Kaisa Nyberg. Differentially uniform mappings for cryptography. In *Advances in cryptology—Eurocrypt'93*, pages 55–64. Springer, jan 1994.

[PP10] Christof Paar and Jan Pelzl. *Understanding Cryptography. A Textbook for Students and Practitioners*. Springer, 2010.

[PRB98] Bart Preneel, Vincent Rijmen, and Antoon Bosselaers. Recent developments in the design of conventional cryptographic algorithms. In *State of the Art in Applied Cryptography*, pages 105–130. Springer, 1998.

[Sch94] Bruce Schneier. Description of a new variable-length key, 64-bit block cipher (blowfish). In *Fast Software Encryption*, pages 191–204. Springer, 1994.