باسمه تعالی

رمز پیشرفته

تمرین سری 5 #

پوریا دادخواه

۴۰۱۲۰۱۳۸۱

1-

$d(x) = a(x)\, b(x) = 1 \mod x^4+1$ معلوم

برای درست آمدن ضرایب $d(x)= d_3x^3+d_2x^2+d_1x+d_0$ داریم:

$$\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} a_0 & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & a_1 & a_0 & a_3 \\ a_3 & a_2 & a_1 & a_0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} 0E \\ 09 \\ 0D \\ 0B \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

که این شکل همان ماتریسی است که در MixColumn دیده‌ایم.

$(\{0E\}\cdot\{02\}) \oplus \{0B\} \oplus \{0D\} \oplus (\{09\}\cdot\{03\}) = \{01\}$ ✓

$(\{09\}\cdot\{02\}) \oplus \{0E\} \oplus \{0B\} \oplus (\{0D\}\cdot\{03\}) = \{00\}$

$(\{0D\}\cdot\{02\}) \oplus \{09\} \oplus \{0E\} \oplus (\{0B\}\cdot\{03\}) = \{00\}$

$(\{0B\}\cdot\{02\}) \oplus \{0D\} \oplus \{09\} \oplus (\{0E\}\cdot\{03\}) = \{00\}$

با فرض این که همه شبیه هستند مثلاً مورد دوم را داریم:
و از شیفت معادل ضرب در {02} می‌توان استفاده کرد.

$\{09\}\cdot\{02\} = 00010010$

$\{0D\}\cdot\{03\} = \{0D\} \oplus \{0D\}\cdot\{02\}$

$= 00001101 \oplus 00011010 = 00010111$

$\Rightarrow 00010010 \oplus 00001110 \oplus 00001011 \oplus 00010111 = 00000001$ ✓

معادله سوم:

$\{0D\}\cdot\{02\} = 00011010 \quad , \quad \{0B\}\cdot\{03\} = \{0B\}\cdot\{02\} \oplus \{0B\}$

$= 00010110 \oplus 00001011 = 00011101$

$\Rightarrow 00011010 \oplus 00001001 \oplus 00001110 \oplus 00011101 = 00000000$ ✓

معادله چهارم :

$\{0B\}\cdot\{02\} = 00010110$, $\{0E\}\cdot\{03\} = \{0E\}\cdot\{02\} \oplus \{0E\}$

$= 00001110 \oplus 00011100 = 00010010$

$FD = 00010110 \oplus 00001101 \oplus 00001001 \oplus 00010010 = 00000000$ ✓

و نهایتاً معادله اول :

$\{0E\}\cdot\{02\} = 00011100$, $\{09\}\cdot\{03\} = \{09\}\cdot\{02\} + \{09\}$

$= 00010010 \oplus 00001001 = 00011011$

$= 00011100 \oplus 00001011 \oplus 00001101 \oplus 00011011 = 00000001$ ✓✓

۲_

کلید اصلی

1	2	3	4	5	6	7	8
FF	FF	FF	FF	E8	17	E8	17
FF	FF	FF	FF	E9	16	E9	16
FF	FF	FF	FF	E9	16	E9	16
FF	FF	FF	FF	E9	16	E9	16

Subword = [16 16 16 16]

RCon = [01 00 00 00]

خانه 5 به بعد را محاسبه می‌کنیم

خانه پنجم) $\begin{bmatrix}16\\16\\16\\16\end{bmatrix} \oplus \begin{bmatrix}01\\00\\00\\00\end{bmatrix} \oplus \begin{bmatrix}FF\\FF\\FF\\FF\end{bmatrix} = \begin{bmatrix}E8\\E9\\E9\\E9\end{bmatrix}$

خانه ششم) و خانه هشتم) $\begin{bmatrix}E8\\E9\\E9\\E9\end{bmatrix} \oplus \begin{bmatrix}FF\\FF\\FF\\FF\end{bmatrix} = \begin{bmatrix}17\\16\\16\\16\end{bmatrix}$,

خانه هفتم) $\begin{bmatrix}17\\16\\16\\16\end{bmatrix} \oplus \begin{bmatrix}FF\\FF\\FF\\FF\end{bmatrix} = \begin{bmatrix}E8\\E9\\E9\\E9\end{bmatrix}$

۳ـ

الف) $x^i \bmod (x^4+1) = x^{i \bmod 4}$

ابتدا واضح است که $x^4 = 1 \bmod (x^4+1)$ چرا که

$$x^4 = [1 \times (x^4+1)] + 1$$

$$x^8 = (x^4)_{\bmod (x^4+1)} \times (x^4)_{\bmod (x^4+1)} = 1 \times 1 = 1$$

بنابراین برای همه مضرب مثبت a خواهیم داشت: $x^{4a} \bmod (x^4+1) = 1$. الان برای هر عدد صحیح

دلخواه $i = 4a + r$ داریم، $0 \le r < 4$

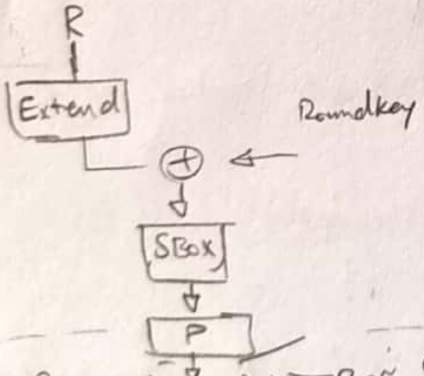
$$x^i \bmod (x^4+1) = (x^{4a} \times x^r)_{\bmod x^4+1}$$

$$= (x^{4a})_{\bmod (x^4+1)} \times (x^r)_{\bmod (x^4+1)} = 1 \times x^r = x^r \checkmark$$

که r همان $i \bmod 4$ است.

۴ـ الف) XOR کردن مواد زیر کلید: ورودی تابع f؛ در DES در هر دور نصف متن ورودی را از تابع f عبور داده

و نصف دیگر مستقیم رد می‌شود؛ در تابع f هم به این صورت زیر با کلید تولید شده در آن راند، ورودی را xor می‌کنیم:



در AES در هر دور تقریباً داریم را: AddRoundkey:

کلید دور را XOR می‌کنیم.

ب) XOR کردن خروجی تابع f با نیمه چپ: در DES خروجی F با نیمه چپ بعدی ترکیب XOR می‌شود

که برای اعمال بخش در fiestel است! در AES برای ایجاد بخش MixColumn را داریم که

هر بیت خانه در ۴ خانه دیگر اثر ندارد و با ترکیب با Shiftrow در کل ماتریس این اثر پخش می‌شود.

ج) تابع F، همان طور که ساختار f را نیز داریم، بخشی از الگوریتم fiestel است که با اعمال مراحل فوق Confusion Diffusion

را اعمال می‌کند. در AES: اعمال Sub-byte - Shiftrow - MixColumn این دو عمل انجام می‌پذیرد و شبیه به f معمولی

است. هم چنین تاثیر تزریق کلید در AddRoundkey صورت می‌پذیرد.

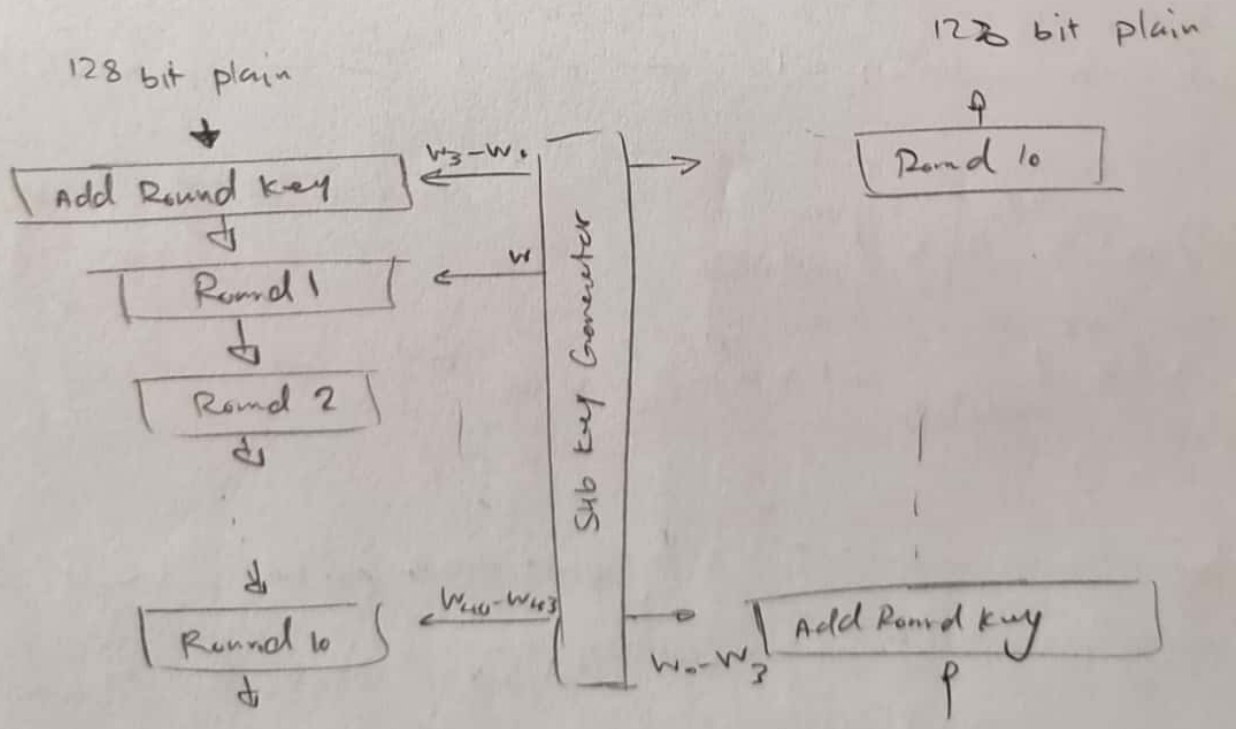
د) تابع جایگشت P : این تابع عمل پخش را انجام می‌دهد که معادل Shiftrow و MixColumn می‌باشد

هـ) جابجا کردن دو نیمه بلوک : این عمل به دلیل ساختار fiestel بودن انجام می‌گیرد ولی در AES نیازی نیست چرا که با MixColumn و Shiftrow روی کل بیت‌های ورودی انجام می‌شود و کل مراحل جانشینی و جایگشت روی کل داده صورت می‌پذیرد به اندازه کافی.

- ب

راند ۱۰ ام از رمزنگاری شامل MixColumn&Rkp نمی‌شود ، بطور مشابه راند ۱۰ از رمزگشایی شامل Inverse MixColumn Skp نیست .

و زیرکلیدهای مورد استفاده در رمزنگاری و رمزگشایی به ترتیب $\{W_{40}-W_{43}\}$ و $\{W_0-W_3\}$ هستند



(۶)

از آنجایی که $MC(m) = MC \cdot x$ و اعمال MixColumn همان ضرب ماتریس است.

با مقدار $MC \cdot MC \cdot MC = A$ را حساب می‌کنیم و نشان می‌دهیم $A = MC^{-1}$

محاسبه:

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \cdot \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} = \begin{bmatrix} 05 & 00 & 04 & 00 \\ 00 & 05 & 00 & 04 \\ 04 & 00 & 05 & 00 \\ 00 & 04 & 00 & 05 \end{bmatrix} = MC \cdot MC$$

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \cdot \begin{bmatrix} 05 & 00 & 04 & 00 \\ 00 & 05 & 00 & 04 \\ 04 & 00 & 05 & 00 \\ 00 & 04 & 00 & 05 \end{bmatrix} = \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} = MC^{-1}$$

$MC \cdot MC \cdot MC =$

۷- می‌دانیم در الگوریتم تعمیم یافته اقلیدس برای دو عدد (چندجمله‌ای) a و b که دارای $gcd = d$ هستند

می‌توان علاوه بر یافتن gcd، ضرایب x و y که $ax + by = d$ می‌شوند را نیز پیدا کنیم

(در روند الگوریتم به جای الگوریتم عادی اقلیدس که صرفاً باقی‌مانده‌ها نگه داشته می‌شوند، در این روند خارج قسمت‌های

تقسیم را نیز نگه می‌داریم که در نهایت x و y در $ax + by = d$ نوشته شوند.)

حال در این میدان از آن جایی که چندجمله‌ای $m(x)$ در میدان $GF(2^8)$ ساده نشدنی هست

نسبت به همه چندجمله‌ای‌های این میدان مثل $b(x)$ که ضرایب در $GF(2)$ و $deg(b(x)) < 8$

اول است، و داریم $gcd(m(x), b(x)) = 1$ بنابراین چندجمله‌ای $c(x)$، $a(x)$ وجود دارند و می‌توان

طی روند الگوریتم تعمیم یافته که $a(x) \cdot b(x) + m(x)c(x) = 1$ شوند. بنابراین

$a(x)b(x) = 1 \pmod{m(x)}$ (چرا که طبق رابطه $m(x) \mid a(x)b(x) - 1$) و به عبارت دیگر

$a(x)$ همان وارون ضربی $b(x)$ به پیمانه $m(x)$ می‌باشد. ■

Scanned by CamScanner

$(A, B, 0, 0, 0, 0, 0, 0) \xrightarrow{16r} (0, 80, 0, 0, 0, 0, 0, 0)$ — 8

$(A, B) \in \{ (e_1, 80), (e_0, 80), (e_1, 0) \}$ e_0, e_1

$(x, 80, 0, 0, 0, 0, 0, 0) \to$ (I

$(0, 80, 0, 0, 0, 0, 0, 0)$

	Δx_7	Δx_6	Δx_5	Δx_4	Δx_3	Δx_2	Δx_1	Δx_0
J	x	80	0	0	0	0	0	0
J+1	80	0	0	0	0	0	0	x
+2	0	0	0	0	0	x	x	80
+3	0	0	0	x	x	x	80	0
+4	0	x	x	x	x	80	0	0
+5	x	x	x	x	80	0	0	x
+6	x	x	x	80	0	x	x	x
+7	x	x	80+?	x	x	x	x	x
+8	x	80+?	?	x	x	x	x	x
+9	e_0	80	?	?	x	x	x	x
+10	0	0	?	?	x	x	e_1	80
+11	0	0	?	?	x	e_1	80	0
+12	0	0	?	?	e_1	80	0	0
+13	0	0	?	e_3	80	0	0	0
+14	0	0	e_3	80	0	0	0	0
+15	0	0	80	0	0	0	0	0
+16	0	80	0	0	0	0	0	0

(rows +8, +9) ⟹ به تناقض

80+? , 80

رسیدیم.

II) $(e_1,0,0,0,0,0,0,0) + (0,80,0,0,0,0,0,0)$

δx_7 δx_6 δx_5 δx_4 δx_3 δx_2 δx_1 δx_0

e_1 0 0 0 0 0 0 0

0 0 0 0 0 0 0 e_1

0 0 0 0 0 ? e_1 0

0 0 0 ? ? e_1 0 0

0 ? ? * e_1 0 0 0

? * * e_1 0 0 0 ?

* * e_1 0 0 ? ? *

* e_1 0 ? * * * *

e_1 ? * * * * * *

e_0 80 * * * * * * *

80 0 * * * * * e_1

0 0 * * * * e_1 80

0 0 * * * e_1 80 0

0 0 * * e_1 80 0 0

0 0 * e_4 80 0 0 0

0 0 e_4 80 0 0 0 0

0 0 80 0 0 0 0 0

0 80 0 0 0 0 0 0

$\Rightarrow$ نستنتج:

$(e_0,80) + (e_1,?)$

۹- همان‌طور که می‌دانیم در هر ساختار، بیت‌های غیرفعال یک مقدار ثابت گرفته و مشخص اند و پس از بررسی یک ساختار می‌توان با تغییر بیت‌های غیرفعال روندها به را طی کرد و شروط لازمی بر کلید به دست آورد که انتخاب‌های محدودتری داشته باشد. در همین راستا تعداد ساختارهایی که باید بررسی کرد و روند بازیابی کلید و تناقض را روی آن پیاده کرد باید تعدادی باشد که مجموع هزینه‌های صرف این تحلیل و جستجوی جامع بعد از آن (روی مجموعه کلید محدود باقی مانده برای سایر دورهای خارج از حمله) کمتر از هزینه حمله‌های مشابه شود.

به همین جهت اگر n ساختار را بررسی کنیم، بنابه روند حمله گفته شده روی Hight،

همه متن‌های plain در یک ساختار $lsb_7(P_0)$ یکسان داشته و بقیه بیت‌ها دلخواه اند و هر دو متن plain در یک ساختار دارای تفاوت (0, *, *, *, *, *, ?, ?) و یا (0x80, *, *, *, *, *, ?, ?) اند.

اگر $u = msb(P_0)$ و $v = lsb(P_1)$ باشد، هر تفاضل دلخواهی را در نظر بگیرید هر تفاضل از آن‌هایی که (0,0) و (u,v). آن‌هایی را نگه می‌داریم که تفاضل cipher برابر (0, 0, 0x80, *, *, *, ?, ?)

بنابه حمله روند تولید تعداد کلیدهای باقی مانده برابر $2^d = 2^n \times \frac{3}{4} \times 2^{57} \times 2^{56} \times 2^{-25}$ می‌شود.

با مقایسه با سایر حملات که در جدول ۱ از مقاله داده شده آمده است، اگر $n = 2.3$ قرار بگیرد

$2^d \approx 2^{89.89}$ شده و پیچیدگی داده برابر $2^{59.3}$ می‌شود که نسبت به سایر روش‌ها پیچیدگی داده و زمان

بهتری دست می‌یابیم.

الف)

۱۰ در این حمله به این ترتیب حمله میکنیم:

* ابتدا تعداد دور حمله را مشخص میکنیم (مثلاً ۲ دور از n دور کل الگوریتم)
* یک حالت ناممکن دو دوسو از مبدا الگوریتم پیدا میکنیم. مثلاً اگر j دور تناقض داشته باشیم، با داشتن تفاضل ورودی A در ابتدای j دور، امکان وقوع تفاضل B در پایان j دور وجود ندارد.
* در مرحله بازیابی کلید از دوسو تناقض به دوسو حمله گسترش می‌دهیم از بالا و پایین و کلیدهایی که منجر به تناقض می‌شوند را شناسایی میکنیم
* تعداد کافی جفت (plain, cipher) را که منجر به تناقض می‌شوند را انتخاب میکنیم و حالات مختلف راست می‌کنیم.
* کلیدهایی که برای جفت‌های فوق منجر به تناقض می‌شوند را از حالت‌های ممکن کلید حذف میکنیم.
* برای کلیدهای باقی مانده در حالیکه دورهای الگوریتم، جستجوی جامع انجام می‌دهیم

ب) * تعداد دور مربوط به تناقض باید MAX شود چرا اگر تعداد دور حمله کل ثابت باشد، با آوردن دور تناقض، دورهای بازیابی کلید از دوسو تناقض کمتر می‌شود و محاسبات کمتری را باید انجام دهیم و پیچیدگی کاهش می‌یابد

* تعداد بیت‌های کلید که در مرحله استخراج کلید وجود دارند باید MAX شوند (یا همان k_3 در جزوه دکتر افدلی)

چرا که با ثابت بودن اندازه کل کلید k، با آوردن k_3، k_1+k_2 کاهش می‌یابد که یعنی حالات ممکن کمتری برای بازیابی کلید باقی می‌ماند و پیچیدگی جستجو کاهش می‌یابد

* باید جمع بیت‌های متقابل در متن اصلی و رمز را Min کرد چرا که همان طور که در جزوه محاسبه شده، تعداد بیت‌های جفت متقابل برابر $2^{lp+lc-1}$ است با کاهش آن تعداد جفت‌های لازم برای امکان حمله نیز کاهش می‌یابد و هزینه زمان حمله نسبت به متغیرها جمع کمتر می‌شود و حمله آسان‌تر صورت می‌گیرد.

ج)

plain : بیت‌ها l_p فعال $\Rightarrow$ $b-l_p$ بیت مقدار ۵ ثابت دارند.

cipher : " " l_c " $\Rightarrow$ $b-l_c$ " " "

در یک ساختار یک زوج (p_1, p_2) با (c_1, c_2) داریم که تمام در بیت‌های فعال تفاوت دارند و سایر بیت‌ها صفت مشخصی از حالات ممکن را اتخاذ کرده‌اند. $\Leftarrow$

plain : 2^{b-l_p} ساختار

cipher : 2^{b-l_c} ساختار

plain $\rightarrow$ $\binom{2^{l_p}}{2} \times 2^{-(b-l_c)} = \frac{1}{2} 2^{l_p}(2^{l_p}-1) \times 2^{-(b-l_c)} \simeq 2^{2l_p+l_c-b-1}$

p_2, p_1 انتخاب دو جفت $\hookrightarrow$ با فرض توزیع یکنواخت احتمال وقوع تفاضل مورد نیاز $\rightsquigarrow$ وقوع تفاضل مورد نیاز در سمت cipher را حساب کنیم.

cipher $\rightarrow$ $\binom{2^{l_c}}{2} \times 2^{-(b-l_p)} \simeq 2^{2l_c+l_p-b-1}$

$\Rightarrow$ تعداد جفت‌ها:

plain : # structure $\times$ # pair در هر ساختار $= 2^{b-l_p} \times 2^{2l_p+l_c-b-1}$

cipher = روند مشابه $\rightsquigarrow$ $= 2^{l_p+l_c-1}$

$b=128$ $\quad l_p = 8\times 7 = 56$, $l_c = 6\times 8 = 48$

د)

structure $\begin{cases} \text{plain} = 2^{b-l_p} = 2^{72} \\ \text{cipher} = 2^{b-l_c} = 2^{80} \end{cases}$

جفت مورد نیاز $= 2^{72} \times 2^{31} = 2^{103}$

برای حمله $\rightsquigarrow$

تعداد اعضای هر ساختار $= \begin{cases} \text{plain}: 2^{2l_p+l_c-b-1} = 2^{31} \\ \text{cipher} = 2^{23} \end{cases}$

plain $\uparrow$

$P_0 = 2^{120}$ $\quad P_0 \times 2^{-(128-48)} = 2^{40}$

$P_1 = 2^{112}$ $\quad P_1 \times 2^{-(128-56)} = 2^{40}$

cipher $\downarrow$

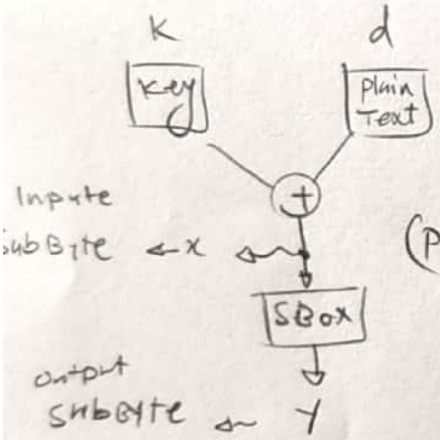
$2^{80+23} = 2^{72+30} = 2^{103}$ $\Rightarrow$ همانطور که در قسمت قبل دیدیم. ه)

2^{104} جفت نیاز داریم پس 2 جفت لازم است

Scanned by CamScanner

(۱۱) الف) کلیت این حمله بدین صورت است که به ازای تعداد زیادی متن اولیه، خروجی sbox حاصل از اعمال بر روی xor کلید و متن اولیه را حساب می‌کنیم و یک تابع از رد توان خروجی را برای تحلیل بررسی می‌کنیم (مثلاً MSB، فاصله همینگ، وزن همینگ، ...) پس تحلیل (فاصله) نمونه‌های مختلف خروجی را به ازای همه متن اولیه حساب می‌کنیم (برای تمام کلیدهای احتمالی j) طبیعتاً اگر کلید درست بوده باشد، برای همه این متون اولیه اختلاف خواهیم داشت و این میانگین اختلاف MAX خود را اختیار خواهد کرد ولی برای کلیدهای اشتباه، این خروجی شبیه تصادفی تولید شده و در طول آزمایش این تعداد مقدار اولیه حول صفر خواهد ماند.

برای دیدن جزئیات دقیق‌تر این الگوریتم، آن را برای تابع MSB مثال می‌زنیم:



← به ازای $0 \le i \le 999$، خروجی $y_{d,k} = SBOX(d \oplus k)$ را برای ۱۰۰۰ متن اولیه d حساب می‌کنیم و برای هر کدام رد توان (Power trace) P_i را ذخیره می‌کنیم.

به عنوان تابع بررسی، $v_{i,j} = MSB(y_{d_i,j}) = MSB(SBOX(d_i \oplus j))$ را محاسبه می‌کنیم. (یعنی برای هر i، همه کلیدهای احتمالی j را در نظر گرفته و $v_{i,j}$ را حساب می‌کنیم) $0 \le j \le 255$

میانگین اختلاف توان خروجی را برای هر کلید روی متون اولیه حساب می‌کنیم:

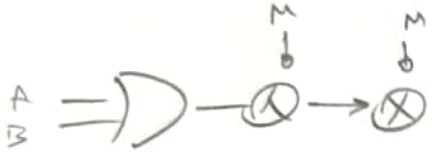
$$DP_j^{avg} = \left| mean(P_i \mid v_{i,j}=0) - mean(P_i \mid v_{i,j}=1) \right|$$

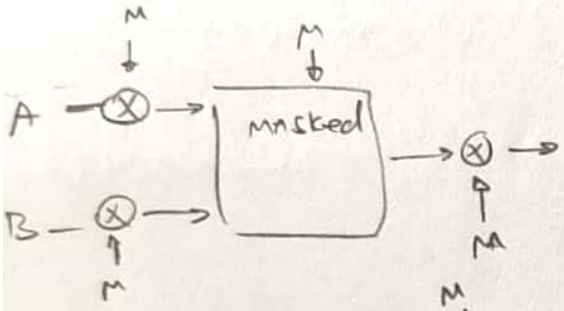
همان‌طور که گفته شد، این مقدار برای کلیدهای نادرست توزیع یکنواخت خواهد داشت. ولی برای کلید اصلی تفاوت قابل قبول در طول همه ۱۰۰۰ متن ورودی دارد پس

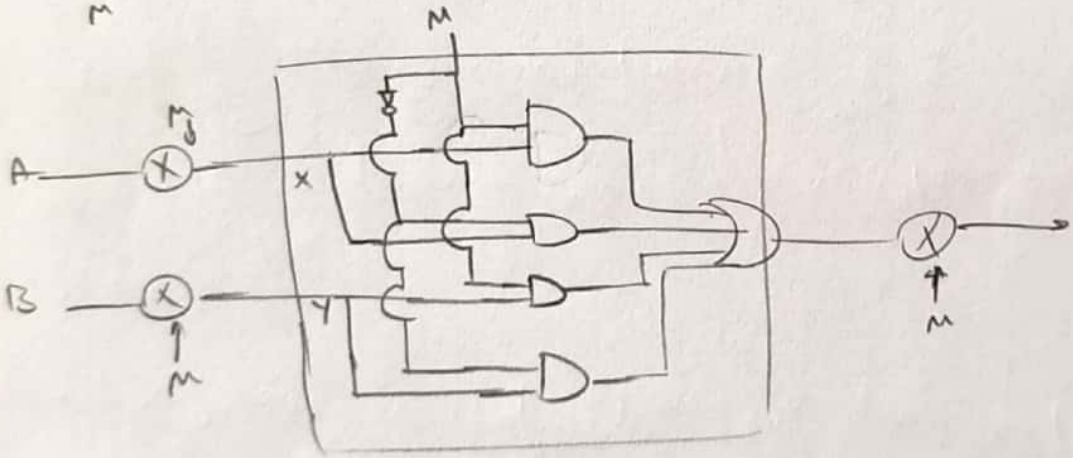
$$K = \arg\max_j (DP_j^{avg})$$

ب) دلیل اصلی این حمله اختلاف توان مصرفی مدار برای تولید بیت کم صفر و یک در خروجی است و از روی تعداد یک های موجود در یک byte و یا همان وزن هیب آن توان مصرفی را (power trace) را داشته است.

۱۲-







برای بررسی صحت به ورودی داده شده مستقل از M باشد

$$(x\,M) + (x\bar{M}) + (yM) + (y\bar{M}) = x + y \rightarrow \text{OR}$$

دیفی دوست است.