Guardant®

КФ МГТУ имени Н.Э. Баумана
Щелкунов Д.А.

White-Box криптография, обфускация и защита ПО. Основные направления развития

White-Box криптография

- Сокрытие ключа симметричного шифра в специальной его реализации
  - Очень быстрые асимметричные схемы, где пара алгоритмов шифрования-расшифрования является ключевой парой
- Изменение исходного симметричного шифра для работы с модифицированным ключом
  - Имеет смысл при выполнении на недоверенных платформах

White-Box криптография

- Пока не найдено подходов, позволяющих создать стойкие White-Box реализации известных симметричных шифров

Куда двигаться дальше?

White-Box криптография

- Что важнее — максимальная криптографическая стойкость шифра к классическим атакам или возможность использовать его на недоверенной платформе?
- Консенсус?

LRC-метод

- Сокрытие линейной зависимости между элементами конечного поля (Linear Relationship Concealing)

$$\begin{cases} x_1 = ((a \cdot b)(mod\ p_1) \cdot c)(mod\ p_2) \\ x_2 = (a \cdot (b \cdot c)(mod\ p_2))(mod\ p_1) \end{cases} \quad (1)$$

Утверждение 1:
В системе (1) *существуют такие полиномы* a, b, c, *что* $x_1 \neq x_2$.

LRC-метод

$$\begin{cases} y_1(x) = (s(x) \cdot a(mod\ p_1)) \cdot b(mod\ p_2) \\ y_2(x) = (s(x) \cdot c(mod\ p_1)) \cdot d(mod\ p_3) \end{cases} \quad (2)$$

p_1, p_2, p_3 – *неприводимые попарно неравные полиномы одинаковой степени над* $GF(2)$

x, a, b, c, d – *произвольные полиномы над* $GF(2)$

$s(x)$ – *нелинейная функция от* x

функции $y_1(x)$ *и* $y_2(x)$ *заданы таблично*

LRC-метод

$$\begin{cases} y_1(x) = (s(x) \cdot a(mod\ p_1)) \cdot b(mod\ p_2) \\ y_2(x) = (s(x) \cdot c(mod\ p_1)) \cdot d(mod\ p_3) \end{cases} \quad (2)$$

Задача нахождения линейной зависимости между $s(x) \cdot a(mod\ p_1)$ *и* $s(x) \cdot c(mod\ p_2)$ *при известных* $y_1(x)$ *и* $y_2(x)$ *имеет сложность не менее , чем* $2^{2n.}$

$n -$ *степень* p_1 *и* p_2

LRC-метод

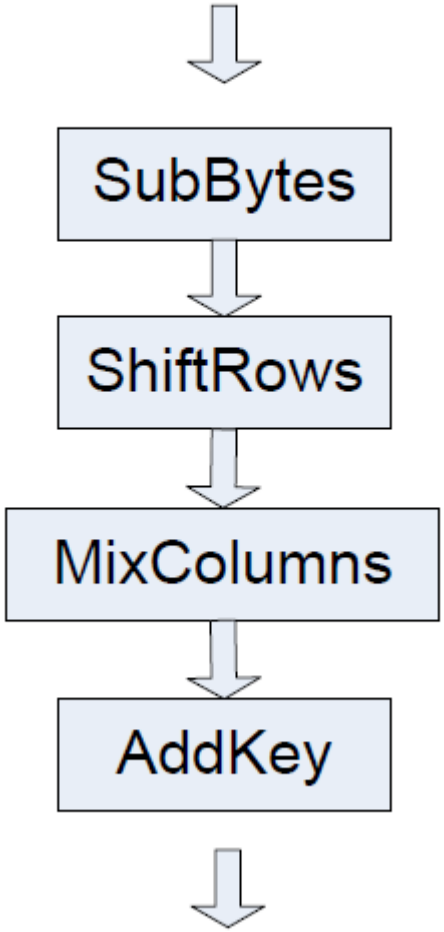
$$\begin{cases} y_1(x) = (\ldots(s(x)\cdot a(\mathrm{mod}\ p_1))\cdot b^{(0)}(\mathrm{mod}\ p_2^{(0)})\ldots)\cdot b^{(k)}(\mathrm{mod}\ p_u^{(k)}) \\ y_2(x) = (\ldots(s(x)\cdot c(\mathrm{mod}\ p_1))\cdot d^{(0)}(\mathrm{mod}\ p_3^{(0)})\ldots)\cdot d^{(k)}(\mathrm{mod}\ p_v^{(k)}) \end{cases} \quad (3)$$

$$p_i^{(\alpha)} \neq p_i^{(\beta)}$$

Сложность восстановления линейной зависимости между $s(x)\cdot a(\mathrm{mod}\ p_1)$ *и* $s(x)\cdot c(\mathrm{mod}\ p_2)$ *при известных* $y_1(x)$ *и* $y_2(x)$ *составит* $2^{2n(k+1)}$.

AES-128

Раунд AES-128



$$\begin{bmatrix} e_{0j} \\ e_{1j} \\ e_{2j} \\ e_{3j} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \bullet \begin{bmatrix} S[a_{0j}] \\ S[a_{1j-1}] \\ S[a_{2j-2}] \\ S[a_{3j-3}] \end{bmatrix} \oplus \begin{bmatrix} k_{0j} \\ k_{1j} \\ k_{2j} \\ k_{3j} \end{bmatrix}$$

$$T_0[a] = \begin{bmatrix} S[a] \bullet 02 \\ S[a] \\ S[a] \\ S[a] \bullet 03 \end{bmatrix}; T_1[a] = \begin{bmatrix} S[a] \bullet 03 \\ S[a] \bullet 02 \\ S[a] \\ S[a] \end{bmatrix};$$

$$T_2[a] = \begin{bmatrix} S[a] \\ S[a] \bullet 03 \\ S[a] \bullet 02 \\ S[a] \end{bmatrix}; T_3[a] = \begin{bmatrix} S[a] \\ S[a] \\ S[a] \bullet 03 \\ S[a] \bullet 02 \end{bmatrix}$$

AES-128

$$\begin{bmatrix} e_{0j} \\ e_{1j} \\ e_{2j} \\ e_{3j} \end{bmatrix} = T_0[a_{0j}] \oplus T_1[a_{1j-1}] \oplus T_2[a_{2j-2}] \oplus T_3[a_{3j-3}] \oplus \begin{bmatrix} k_{0j} \\ k_{1j} \\ k_{2j} \\ k_{3j} \end{bmatrix}$$

LRC-метод

$$T'_i[a] = \begin{bmatrix} ((\ldots(t_i^{(0)}(a)\cdot b_i^{(0,0)})(mod\, p_i^{(0,0)})\cdot b_i^{(0,1)}(mod\, p_i^{(0,1)})\ldots)\cdot b_i^{(0,k_0)}(mod\, p_i^{(0,k_0)}) \\ ((\ldots(t_i^{(1)}(a)\cdot b_i^{(1,0)})(mod\, p_i^{(1,0)})\cdot b_i^{(1,1)}(mod\, p_i^{(1,1)})\ldots)\cdot b_i^{(1,k_1)}(mod\, p_i^{(1,k_1)}) \\ ((\ldots(t_i^{(2)}(a)\cdot b_i^{(2,0)})(mod\, p_i^{(2,0)})\cdot b_i^{(2,1)}(mod\, p_i^{(2,1)})\ldots)\cdot b_i^{(2,k_2)}(mod\, p_i^{(2,k_2)}) \\ ((\ldots(t_i^{(3)}(a)\cdot b_i^{(3,0)})(mod\, p_i^{(3,0)})\cdot b_i^{(3,1)}(mod\, p_i^{(3,1)})\ldots)\cdot b_i^{(3,k_3)}(mod\, p_i^{(3,k_3)}) \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ ((\ldots(t_i^{(n)}(a)\cdot b_i^{(n,0)})(mod\, p_i^{(n,0)})\cdot b_i^{(n,1)}(mod\, p_i^{(n,1)})\ldots)\cdot b_i^{(n,k_n)}(mod\, p_i^{(n,k_n)}) \end{bmatrix} \tag{4}$$

$t_i^{(j)}$ – *элемент* T – *box до применения* LRC – *метода*

$b_i^{(j,u)}$ – *случайно выбранный полином в* $GF(2^8)$

$p_i^{(j,u)}$ – *неприводимый полином степени* 8 *над* $GF(2)$

LRC-метод

$$p_i^{(0,v)} \neq p_i^{(1,v\psi} \neq \ldots \neq p_i^{(n,v)}$$

$$Y_j = \begin{bmatrix} y_j^{(0)} \\ y_j^{(1)} \\ y_j^{(2)} \\ y_j^{(3)} \end{bmatrix} = \begin{bmatrix} mix_j^{(0)}(t_j^{(0,0)}) \\ mix_j^{(1)}(t_j^{(1,0)}) \\ mix_j^{(2)}(t_j^{(2,0)}) \\ mix_j^{(3)}(t_j^{(3,0)}) \end{bmatrix} \oplus \begin{bmatrix} mix_j^{(0)}(t_j^{(0,1)}) \\ mix_j^{(1)}(t_j^{(1,1)}) \\ mix_j^{(2)}(t_j^{(2,1)}) \\ mix_j^{(3)}(t_j^{(3,1)}) \end{bmatrix} \oplus \begin{bmatrix} mix_j^{(0)}(t_j^{(0,2)}) \\ mix_j^{(1)}(t_j^{(1,2)}) \\ mix_j^{(2)}(t_j^{(2,2)}) \\ mix_j^{(3)}(t_j^{(3,2)}) \end{bmatrix} \oplus \begin{bmatrix} mix_j^{(0)}(t_j^{(0,3)}) \\ mix_j^{(1)}(t_j^{(1,3)}) \\ mix_j^{(2)}(t_j^{(2,3)}) \\ mix_j^{(3)}(t_j^{(3,3)}) \end{bmatrix} \quad (5)$$

$$mix_j^{(i)}(t_j^{(i,k)}) = ((\ldots(t_j^{(i,k)} \cdot b_j^{(i,0)})(mod\ p_j^{(i,0)}) \cdot b_j^{(i,1)}(mod\ p_j^{(i,1)})\ldots) \cdot b_j^{(i,n)}(mod\ p_j^{(i,n)}) \quad (6)$$

LRC-метод

- Метод нестоек к CPA-атаке в следующих случаях:
  - Когда таблицы подстановок (S-box) известны
  - Когда известно преобразование MixColumns
  - Когда таблицы подстановок одинаковы для каждого байта в рамках одного раунда

LRC-метод

- Известны S-box таблицы и преобразование MixColumns

$$t_j^{(i,k)} = s[a] \cdot n \oplus key_j^{(i,k)} \qquad (7)$$

$$mix_j^{(i)}(t_j^{(i,k)}) = mix_j^{(i)}(s[a] \cdot n) \oplus mix_j^{(i)}(key_j^{(i,k)}) \qquad (8)$$

$$mix_j^{(i)}(t_j^{(i,k)}) \oplus mix_j^{(i)}(t'^{(i,k)}_j) = mix_j^{(i)}(n \cdot (s[a] \oplus s[a'])) \qquad (9)$$

$mix_j^{(i)} - находится$, как таблица подстановок

$Сложность - 2^8$

LRC-метод

- Известны S-box таблицы и неизвестно преобразование MixColumns

$$\begin{cases} t_j^{(0,0)} = s[a] \cdot n^{(0)} \oplus key_j^{(0,0)} \\ t_j^{(0,1)} = s[a] \cdot n^{(1)} \oplus key_j^{(0,1)} \\ t_j^{(0,2)} = s[a] \cdot n^{(2)} \oplus key_j^{(0,2)} \\ t_j^{(0,3)} = s[a] \cdot n^{(3)} \oplus key_j^{(0,3)} \end{cases} \quad (10)$$

LRC-метод

Пусть $n^{(0)} = alpha$. Тогда, следуя предыдущему алгоритму, найдем $mix_j^{(0)}$, а следовательно, $n^{(1)}, n^{(2)}, n^{(3)}$.

Если $n^{(0)} = \alpha$, то $n^{(0)}, n^{(1)}, n^{(2)}, n^{(3)}$ – коэффициенты многочлена в MixColumns, что легко проверить, применив InvMixColumns.

Сложность $2^{16.}$

LRC-метод

- Неизвестны S-box таблицы и неизвестно преобразование MixColumns, но S-box таблицы одинаковы для каждого байта в рамках раунда.

$$Пусть\ n^{(0)}=\alpha,\ s[a]=\beta,\ s[a']=\beta'.\ Действуем,\ как\ в\ предыдущем\ случае.$$

$$Сложность\ 2^{24}$$

LRC-метод

- Для Rijndael метод LRC не стоек к CPA
- Для успешного решения задачи создания асимметричной схемы с применением метода LRC необходимо автоматически генерировать SPN шифр со структурой, схожей с Rijndael
  - S-box таблицы случайны и отличаются для каждого входного байта каждого раунда
  - Преобразование MixColumns случайно и выполняется по модулю $x^{16}+1$

LRC-метод

- Для Rijndael метод LRC применяется для усложнения реверсирования алгоритма, т.к. позволяет создать схему, работающую с модифицированным сеансовым ключом
  - Возможно использовать для дополнительной защиты от атак по побочным каналам
- Необходимы дальнейшие исследования в направлении White-Box криптографии

Механизмы White-Box в обфускации

- Замена ряда инструкций промежуточного представления таблицами подстановок 6 x 4
  - Возможность производить вычисления над зашифрованными данными
  - Небольшой размер
  - Высокая скорость
  - Противодействие шаблонной дебфускации и декомпиляции ВМ
  - Используется в Guardant

Guardant 3a-code

- Фреймворк для упрощения создания алгоритмов обфускации-деобфускации
  - Позволяет упростить создание утилит автоматической защиты
  - Базируется на трехадресном представлении инструкций
  - Содержит базовые стратегии анализа кода и данных
  - Имеется возможность перевода из x86 в промежуточное представление

Guardant 3a-code

- Содержит компилятор в x86
- Легко интегрируется с псевдокодом Guardant и генератором полиморфного кода Guardant
- Повышает защищенность ПО
- Позволяет опробовать различные техники обфускации без серьезных затрат

Вопросы