Щелкунов Д.А., к.т.н.
Компания «Актив», www.aktiv-company.ru

О практическом применении White-Box криптографии

Рассказывается о практическом применении механизмов White-Box криптографии. Проводится исследование одной из White-Box реализаций алгоритма шифрования AES-128. Описаны принципиальные проблемы при создании такого рода реализаций. Показаны недостатки алгоритма блочного шифрования AES для создания стойких White-Box схем. Предложен оригинальный подход, позволяющий создавать более стойкие White-Box реализации.

В последнее время большой популярностью пользуются изыскания в области White-Box криптографии [1]-[5]. Как утверждают авторы ряда работ на эту тему, механизмы White-Box криптографии позволяют скрыть ключ симметричного алгоритма шифрования в самом алгоритме. Таким образом, симметричный шифр превращается в ассиметричный. Действительно, если восстановление ключа из White-Box реализации алгоритма шифрования будет трудной задачей, также как и создание алгоритма дешифрования по имеющемуся алгоритму шифрования, то пара алгоритмов (алгоритм шифрования – алгоритм дешифрования) становится фактически ключевой парой, где White-Box реализация алгоритма шифрования является открытой, а реализация алгоритма дешифрования недоступна аналитику. Такая схема может обладать целым рядом преимуществ по сравнению с классическими ассиметричными криптосистемами. Основным преимуществом является скорость работы открытой части.

За последние годы было представлено несколько White-Box реализаций известных алгоритмов шифрования. Основной особенностью этих реализаций является встраивание раундовых ключей в таблицы подстановок. Таким образом, получается, что ни сам ключ шифрования – ни раундовые ключи явным образом в программе не фигурируют. Однако, в подавляющем большинстве случаев их восстановление возможно. Более того, даже если достаточно трудно восстановить сам ключ, то существует возможность построить таблицы обратных подстановок, а, следовательно, возможность построить алгоритм дешифрования по алгоритму шифрования. Рассмотрим, например, White-Box реализацию алгоритма AES-128 [5].

Данная реализация отличается от классической реализации AES-128 тем, что ключ шифрования помещен в таблицы подстановок. Это стало возможным сделать, благодаря особенностям 32-битной реализации алгоритма блочного шифрования AES. Благодаря своей структуре, алгоритм AES достаточно легко реализуется при помощи набора таблиц подстановок или *T*-блоков, каждый из которых представляет собой преобразование одного байта в четыре. Рассмотрим это преобразование поподробнее:

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

$$T_0[a] = \begin{bmatrix} S[a] \cdot 02 \\ S[a] \\ S[a] \\ S[a] \cdot 03 \end{bmatrix};\ T_1[a] = \begin{bmatrix} S[a] \cdot 03 \\ S[a] \cdot 02 \\ S[a] \\ S[a] \end{bmatrix};\ T_2[a] = \begin{bmatrix} S[a] \\ S[a] \cdot 03 \\ S[a] \cdot 02 \\ S[a] \end{bmatrix};\ T_3[a] = \begin{bmatrix} S[a] \\ S[a] \\ S[a] \cdot 03 \\ S[a] \cdot 02 \end{bmatrix} \qquad (2)$$

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

Здесь e_{ij}-зашифрованный i-й байт состояния j, k_{ij}-байт раундового ключа, $S[a_{ij}]$-результат, полученный из таблицы подстановок для исходного i-го байта состояния j, a_{ij}-исходный i-й байт состояния j. $T_i[a]$-T-блок. Вот в такие T-блоки и был спрятан ключ AES. Действительно, это сделать достаточно просто. Достаточно, разложив предварительно раундовый подключ на 4 составляющих, сложить по модулю 2 значения каждого из T-блоков с соответсвующей составляющей. Однако, восстановить такой ключ будет несложно. Для этого надо будет всего-лишь выполнить последовательность операций *SubBytes*, *ShiftRows* и *MixColumns*, а затем полученный результат сложить по модулю 2 с результатом раунда такой White-Box реализации. Для того, чтобы взлом не был столь простым, автор ввел дополнительные межраундовые запутывающие преобразования.

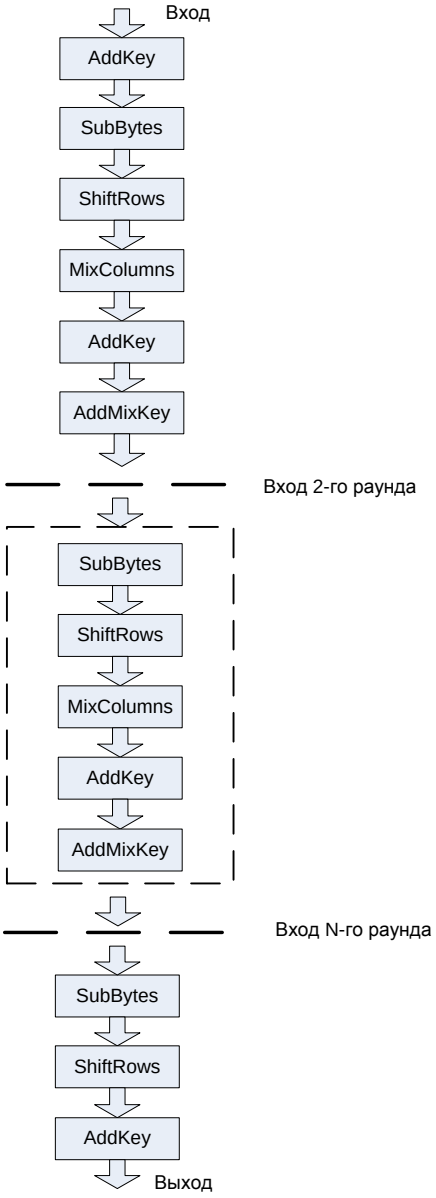

Рисунок 1. Структура White-Box реализации.

На рисунке 1 эти преобразования обозначены, как *AddMixKey*. По сути они означают сложение по модулю два результатата кажого раунда с некоторым значением. Для того, чтобы результат работы алгоритма не отличался от AES-128, блоки подстановок *SubBytes* тоже были модифицирован соответствующим образом. В итоге на каждый раунд получилось 16 различных семейств 256-и элементных *T*-блоков. Однако, метод взлома данной схемы тоже не сложен. Рассмотрим, например, 1-й раунд White-Box реализации. Применяя к результату этого раунда преобразование *InvMixColumns*, обратное преобразованию *MixColumns*, получим фактически набор из 16-и однобайтовых подстановок. Т.к. результат White-Box реализации должен совпадать с результатом AES-128, мы можем предположить, что используются те же таблицы подстановок. Таким образом мы получаем, что нам фактически для каждого из 16-и байт необходимо подобрать два однобайтовых числа val_1 и val_2 такие, что

$$S[val_1 \oplus a] \oplus val_2 = S^{'}[a] \qquad (4)$$

Здесь $S^{'}[a]$-результат для каждого входного байта после применения *InvMixColumns*. Таким образом, найдя 16 различных val_1, по одному для каждого из входных байтов мы автоматически получим встроенный ключ шифрования.

После успешного анализа вышеописанной схемы автор модифицировал ее, использовав другое конечное поле $GF(2^8)$, которое не является полем *Rijndael* и изменив полином в операциях *MixColumns* и *InvMixColumns*. Таким образом, полученный в результате шифр уже не является AES-128. Тем не менее, в результате анализа приведенных автором таблиц были построены таблицы обратной подстановки общим размером 40К. Это удалось сделать, благодаря тому, что линейная зависимость между элементами *T-box*-ов не была скрыта, что позволило найти полином, обратный используемому в операции *MixColumn* и, как следствие, инвертировать полученные в результате *S-box*-ы.

Рассмотрим следующую формулу:

$$((a \cdot b)(mod\ p_1) \cdot c)(mod\ p_2) \neq (a \cdot (b \cdot c)(mod\ p_2))(mod\ p_1) \qquad (5)$$

Здесь p_1, p_2 – неприводимые полиномы 8-й степени. Формула (5) говорит нам о том, что закон ассоциативности в данном случае не выполняется. Таким образом, если многократно умножать каждый элемент *T-box*-а на произвольно выбранные полиномы по модулю произвольно выбранных полиномов 8-й степени, то восстановление линейной зависимости между элементами *T-box*-ов будет сложной задачей. Формально это выглядит следующим образом:

$$T_i^{'}[a] = \begin{bmatrix} ((\dots((t_i^{(0)}(a) \cdot b_i^{(0,0)})(mod\ p_i^{(0,0)}) \cdot b_i^{(0,1)})(mod\ p_i^{(0,1)})\dots) \cdot b_i^{(0,k_0)}(mod\ p_i^{(0,k_0)}) \\ ((\dots((t_i^{(1)}(a) \cdot b_i^{(1,0)})(mod\ p_i^{(1,0)}) \cdot b_i^{(1,1)})(mod\ p_i^{(1,1)})\dots) \cdot b_i^{(1,k_1)}(mod\ p_i^{(1,k_1)}) \\ ((\dots((t_i^{(2)}(a) \cdot b_i^{(2,0)})(mod\ p_i^{(2,0)}) \cdot b_i^{(2,1)})(mod\ p_i^{(2,1)})\dots) \cdot b_i^{(2,k_2)}(mod\ p_i^{(2,k_2)}) \\ ((\dots((t_i^{(3)}(a) \cdot b_i^{(3,0)})(mod\ p_i^{(3,0)}) \cdot b_i^{(3,1)})(mod\ p_i^{(3,1)})\dots) \cdot b_i^{(3,k_3)}(mod\ p_i^{(3,k_3)}) \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\ ((\dots((t_i^{(n)}(a) \cdot b_i^{(n,0)})(mod\ p_i^{(n,0)}) \cdot b_i^{(n,1)})(mod\ p_i^{(n,1)})\dots) \cdot b_i^{(n,k_n)}(mod\ p_i^{(n,k_n)}) \end{bmatrix} \qquad (6)$$

Здесь t_i^j - элемент *T-box*-а до применения запутывающих преобразований, $b_i^{(j,u)}$ – случайно выбранный полином в $GF(2^8)$, $p_i^{(j,u)}$ – случайно выбранный неприводимый полином 8-й степени над $GF(2)$. При этом должно соблюдаться следующее условие:

$$p_i^{(0,v)} \neq p_i^{(1,v)} \neq \dots \neq p_i^{(n,v)} \qquad (7)$$

Для *n*, не превышающего 29, условие (7) вполне выполнимо. В этом случае отыскать линейную зависимость между элементами *T-box*-ов представляется затруднительным.

Однако если применять данный подход к классическому AES-128, то возможен взлом посредством построения таблиц обратной подстановки размером 154 Гб.

Для воспрепятствования этому необходимо создать новый *SPN*-шифр, в котором в операции *MixColumns* использовался бы полином по модулю $x^{16} \oplus 1$. В пилотном варианте, разработанном в компании «Актив», используются случайно сгенерированные *S-box*-ы и полиномы для операции *MixColumns*. Однако в будущем, если не будет найдено эффективного алгоритма инвертирования защищенных вышеописанным методом *T-box*-ов, а следовательно не будет найдено эффективного алгоритма построения алгоритма дешифрования, при наличии White-Box реализации алгоритма шифрования и наоборот, возможно добавление к разработанной схеме более стойких таблиц подстановки и полиномов с хорошими рассеивающими свойствами.

Использованные материалы:

1. S. Chow, P. Eisen, H. Johnson, P.C. van Oorschot, A White-Box DES Implementation for DRM Applications, 2002.
2. S. Chow, P. Eisen, H. Johnson, P.C. van Oorschot, White Box Cryptography and an AES Implementation, 2002.
3. Julien Bringer, Herve Chabanne, Emmanuelle Dottax, White Box Cryptography: Another Attempt, 2006.
4. Hamilton E. Link, William D. Neumann, Clarifying Obfuscation: Improving the Security of White-Box Encoding.
5. www.softwaresecurity.org - Ivan V. Petrov, Seculab JSC.