Е. Ю. Родионов

ПРЕОБРАЗОВАНИЯ ДЛЯ ЗАПУТЫВАНИЯ СИММЕТРИЧНЫХ БЛОЧНЫХ ШИФРОВ

В статье [1] описывается метод построения асимметричных криптосистем посредством применения запутывающих преобразований к симметричным криптосистемам. На основании этого симметричный алгоритм шифрования может быть обращен в асимметричный, а алгоритм вычисления MAC (код аутентификации сообщения) может быть обращен в алгоритм электронной цифровой подписи. Для решения первой задачи предлагается запутать алгоритм шифрования, вследствие чего определить используемый запутанным алгоритмом ключ шифрования не предоставляется возможным за «полиномиальное» время. В результате полученную конструкцию можно использовать для построения асимметричных криптографических алгоритмов. Таким образом, мы можем получить асимметричный алгоритм шифрования, временная сложность которого сопоставима с временной сложностью симметричного алгоритма.

В статье [2] авторы предлагают метод для запутывания алгоритма DES. В настоящей статье описываются преобразования и элементы математического аппарата, позволяющие адаптировать данный метод для алгоритма ГОСТ 28147-89. Принципиальное различие в методах запутывания обусловлено тем, что цикловой ключ в алгоритме ГОСТ 28147-89 «подмешивается» посредством операции сложения по модулю 2^{32}, а в случае алгоритма DES — побитовым сложением по модулю 2.

Как известно, симметричные алгоритмы шифрования, имеющие в основе своей конструкции шифр Фейстеля, как правило, состоят из следующих основных операций:

- перестановка (биективное преобразование конечного множества);
- преобразования таблиц замен (S-бокс);
- операция подмешивания циклового ключа (побитовое сложение по модулю 2 для алгоритма DES, в случае алгоритма ГОСТ 28147-89 — сложение по модулю 2^{32}).

Основной и наиболее часто используемой конструкцией в процессе запутывания является затемняющее преобразование, которое позволяет скрыть поток информации между раундами алгоритма, а также информацию в таблицах замен. Предположим, X — отображение множества m-битовых векторов во множество n-битовых. Выберем биективное преобразование m-битовых векторов F и биективное преобразование n-битовых векторов G. Назовем преобразование (1) «затемненным» вариантом преобразования X, где $\cdot$ — операция суперпозиции отображений, F — входное затемняющее преобразование, G — выходное затемняющее преобразование:

$$X' = G \cdot X \cdot F^{-1}. \quad (1)$$

Во многих случаях, когда отображение X получает на вход достаточно длинную строку, затемняющие преобразования получаются довольно громоздкими, например, в случае преобразования $X:\{0,1\}^{64} \rightarrow \{0,1\}^{64}$, отображающего множество 64-битных векторов само в себя, преобразования F и G будут представлять матрицы размерностью 64×64. Для того чтобы избежать такой громоздкости, можно представить затемняющие преобразования в виде конкатенации биективных преобразований битовых векторов меньшей длины (2):

$$F(x) = F_1 \| F_2 \| F3 \| \ldots \| F_k(x), \quad (2)$$

где для любого n-битного вектора b:

$$F(b) = F_1(b_1,\ldots,b_{n1}) \| F_2(b_{n1+1},\ldots,b_{n1+n2}) \| \ldots \| F_k(b_{n1+\ldots+n(k-1)+1},\ldots,b_{nk}). \quad (3)$$

Очевидно, что если все F_i биективны, то и F также биективно. Назовем преобразование F конкатенацией «затемняющих» преобразований. С помощью конкатенации можно добиться того, что «затемняющие» преобразования будут занимать существенно меньше места в памяти.

Рассмотрим суперпозицию двух отображений $X{\cdot}Y$. Назовем отображение (4) «затемненной» суперпозицией преобразований X и Y:

$$X'{\cdot}Y' = (H{\cdot}X{\cdot}G^{-1}) \cdot (G{\cdot}Y{\cdot}F^{-1}) = H{\cdot}X{\cdot}Y{\cdot}F^{-1}. \quad (4)$$

Таким образом, для сокрытия информации, передаваемой между S-боксами, необходимо последовательно применять затемняющие преобразования для S-боксов и операций сложения по модулю 2. Затемняющие преобразования удобно задавать в виде таблицы, которые в настоящей статье также именуются специальными таблицами замены.

Далее рассмотрим преобразования, которые применяются для запутывания алгоритма шифрования ГОСТ 28147–89.

– Частичные предвычисления. Если нам известна часть входных данных во время запутывания алгоритма, то целесообразно их вычислить и использовать во время запутывания. В нашем случае известен ключ, следовательно, этот факт можно использовать для замены обычных S-боксов на специальные S-боксы, именуемые T-боксами, зависящие от используемого ключа.

– Перемешивающие подстановки. Перемешивающая подстановка — это биективное преобразование множества двоичных векторов, такое, что каждый бит в векторе, полученном на выходе преобразования, зависит от большого числа битов во входном векторе. В алгоритме ГОСТ 28147-89, например, операция циклического сдвига представляется как умножение вектора на матрицу, которая довольно сильно разреженна, в том смысле, что она состоит практически из нулей и лишь в некоторых местах имеет единичные элементы. Этот факт существенно ослабляет стойкость запутанной конструкции. Для того чтобы избежать этого, операцию циклического сдвига ρ можно представить в виде суперпозиции двух преобразований $J{\cdot}K$, матрицы которых менее разреженны. Достигается это следующим образом: перемешивающая подстановка K выбирается случайно, а J определяется как $J = \rho{\cdot}K^{-1}$.

– Затемнение таблиц замен. Случайное преобразование ${}^{n}_{m}P$, где n и m — большие числа, не может быть запутано при использовании «затемняющих» преобразований. Так как при использовании специальных таблиц замен для «затемняющих» преобразований потребуется много памяти из-за экспоненциальной зависимости их размера от длины двоичных векторов. Однако этого можно избежать, применяя метод конкатенации «затемняющих» преобразований. Разобьем вход ρ на j блоков длиной a, а выход — на k блоков длиной b. Выберем для каждого входного и выходного блока «затемняющие» преобразования ${}^{a}_{a}F_1, {}^{a}_{a}F_2, \ldots, {}^{a}_{a}Fj$ и ${}^{b}_{b}G_1, {}^{b}_{b}G_2, \ldots, {}^{b}_{b}G_k$, затем определим отображения (5) и (6):

$$F_\rho = (F_1 \| F_2 \| F3 \| \ldots \| F_j), \quad (5)$$

$$G_\rho = (G_1 | G_2 \| G3 \| \ldots \| G_k) \text{ и } \rho' = G_\rho{\cdot}\rho{\cdot}F_\rho^{-1}. \quad (6)$$

– Затемнение комбинации функций. Рассмотрим две функции ρ и Q, которые вычисляются параллельно $\rho \| Q$, и затемняющие преобразования для $\rho \| Q$: $G{\cdot}(\rho \| Q){\cdot}F^{-1}$. Таким образом, мы объединили две функции в одну и запутали конструкцию, применив «затемняющие» преобразования. Затемнение функции $\rho \| Q$ перемешивает входные и выходные данные функции ρ с входными и выходными данными функции Q и делает трудной задачу их разделения.

– Введение избыточности. Для того чтобы отображения было сложнее анализировать, вводится избыточность. Например, рассмотрим отображение ${}^{n}_{m}P$, расширим входные и выходные вектора на a и b бит соответственно, в результате получим отображение ${}^{n+b}_{m+a}P$, которое можно запутать следующим образом:

$${}^{n+b}_{m+a}P' = G{\cdot}({}^{n}_{m}P \,\|\, {}^{b}_{a}E){\cdot}F^{-1}. \quad (7)$$

${}^{b}_{a}E$ называется избыточной компонентой ${}^{n+b}_{m+a}P'$. Можно представить любое отображение ${}^{n}_{m}P$ как специальную таблицу замены: в виде массива из 2^m элементов, где каждый элемент

массива имеет размер n бит. Для того чтобы вычислить $P(x)$, мы находим элемент массива с индексом x и возвращаем его вызвавшей программе. Однако размер такой таблицы замены экспоненциально растет с ростом размера входных данных, что накладывает определенные ограничения на использование данного метода.

В итоге с помощью описанных преобразований можно построить запутанный вариант алгоритма ГОСТ 28147-89 с фиксированным ключом шифрования. Полученный алгоритм шифрования состоит целиком из таблиц замен, запутанных посредством «затемняющих» преобразований. С одной стороны, данные преобразования позволяют скрыть ключ шифрования, с другой стороны, данные преобразования увеличивают размер реализации алгоритма и его временную сложность. В настоящий момент производятся оценка их эффективности и улучшение характеристик производительности.

СПИСОК ЛИТЕРАТУРЫ:

1. *Dennis Hofheinz*. Obfuscation for Cryptographic Purposes. URL: http://eprint.iacr.org/2006/463.pdf.
2. *Chow S.* A White-Box DES Implementation. URL: http://crypto.stanford.edu/DRM2002/whitebox.pdf.