Алгоритм ANUBIS

Пузевич Роман, Б01-906

Ноябрь 2022

1 Введение

ANUBIS - симметричный блочный криптоалгоритм, разработанный Винсентом Риджменом и Пауло Баррето. ANUBIS является вариантом алгоритма Rijndael. Их объединяет структура типа "квадрат"и набор выполняемых преорбазований. Алгоритм шифрует данные блоками по 16 байтов с ключом размером от 128 до 320 бит с шагом в 32 бита. Анубис был египетским богом бальзамирования и погребения, следовательно, богом "шифрования". Поэтому, по мнению авторов алгоритма, это имя кажется подходящим для шифра.

1.1 Симметричные криптосистемы

Прежде чем перейти к алгоритму, рассмотрим симметричные криптосистемы. Симметричная криптосистема - это способ шифрования, в котором, в отличие от асимметричных алгоритмов, для шифрования и расшифрования используется один и тот же ключ. Благодаря этому факту, в сравнении с асимметричными системами, достигается большая простота, большая скорость, меньшая требуемая длина ключа, более простая реализация. Но так же у подобных алгоритмов есть существенный недостаток: необходим защищенный канал для передачи ключа.

Рис. 1: Симметричная криптосистема

2 Алгоритм

Теперь рассмотрим сам алгоритм. Алгоритм представляет блок данных в виде 16-байтового массива $M_{4x4}[GF(2^8)]$ (все вычисления производятся в конечном поле Галуа):

В каждой итерации алгоритма выполняются действия:

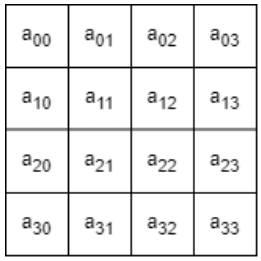

Рис. 2: Представление данных в алгоритме

2.1 Первая операция

Табличная замена γ – отображение $M_{4x4}[GF(2^8)] \to M_{4x4}[GF(2^8)]$ (или $M_{Nx4}[GF(2^8)] \to M_{Nx4}[GF(2^8)]$ в общем случае):
$b_{ij} = S(a_{ij})$

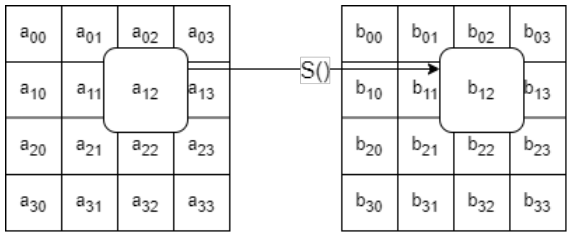

Рис. 3: Первое преобразование

Псевдослучайные значения таблицы выбраны так, чтобы они удовлетворяли следующему свойству: $S(S(x)) = x, \forall x \in GF(2^8)$

2.2 Вторая операция

Байтовая перестановка τ – отображение $M_{4x4}[GF(2^8)] \to M_{4x4}[GF(2^8)]$:
$c_{ij} = b_{ji}$, где b_{ij} результат выполнения предыдущей операции.

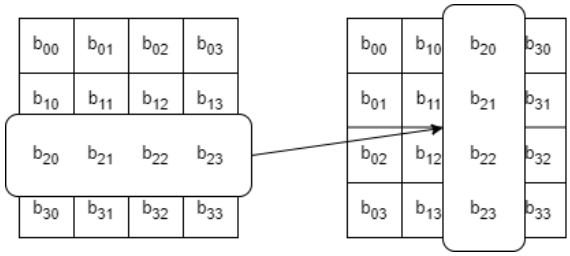

Рис. 4: Второе преобразование

Данную операцию можно воспринимать как транспонирование исходного блока данных.

2.3 Третья операция

Матричное умножение θ – отображение $M_{4x4}[GF(2^8)] \to M_{4x4}[GF(2^8)]$ (или $M_{Nx4}[GF(2^8)] \to M_{Nx4}[GF(2^8$ в общем случае):

$D = CH$
В этой операции мы умножаем результат предыдущих операций на постоянную матрицу H (умножение производится в конечном поле $GF(2^8)$):

$$H = \begin{pmatrix} 1 & 2 & 4 & 6 \\ 2 & 1 & 6 & 4 \\ 4 & 6 & 1 & 2 \\ 6 & 4 & 2 & 1 \end{pmatrix}$$

2.4 Последняя операция

Нахожение ключа σ - отображение $M_{4x4}[GF(2^8)] \to M_{4x4}[GF(2^8)]$ (или $M_{Nx4}[GF(2^8)] \to M_{Nx4}[GF(2^8)]$ в общем случае):
$e_{ij} = d_{ij} \oplus K_{ij}^r$

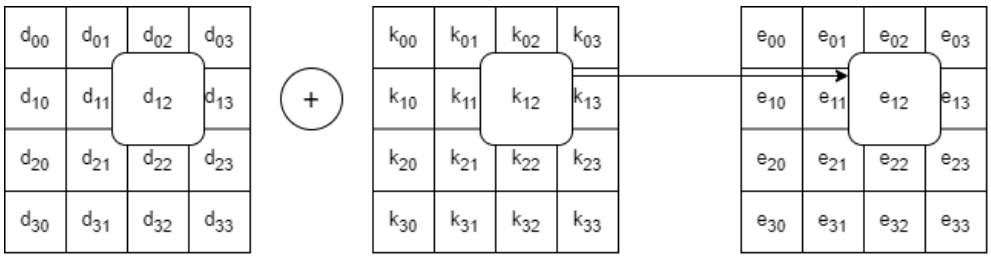

Рис. 5: Последнее преобразование

В данной операции мы побитово накладываем ключ r-го раунда K^r на каждый из байтов матрицы, получившейся после предыдущих операций.

Перечисленные операции выполняются в заданной последовательности $(\gamma, \tau, \theta, \sigma)$ во всех раундах, кроме последнего. В последнем раунде не выполняется операция θ. Более того, перед первым раундом выполняется "входное отбеливание"данных путем наложения нулевого ключа с помощью операции XOR. Перечисленные операции обратимы, то есть расшифрование происходит в обратном порядке шифрования. Количество раундов шифрования R определяется размером ключа и вычисляется следующим образом: $R = 8 + N$, где N размер ключа в 4-байтовых фрагментах.

3 Генерация ключа

В данном разделе мы затронем то, как мы получим ключ, с помощью которого будем шифровать данные, из исходного ключа. Ключ у нас может быть больше массива данных, поэтому необходимо привести его к такому же виду, как и шифруемый блок данных. Стоит упомянуть то, как мы будем представлять наш ключ в процессе шифрования. Ключ представляется в виде $4N$-байтовго массива $M_{Nx4}[GF(2^8)]$

3.1 Вычисление вспомогательных ключей

В самом начале вычислим вспомогательные ключи, которые понадобятся для нахождения основных ключей шифрования. Количество вспомогательных ключей зависит от того, сколько раундов

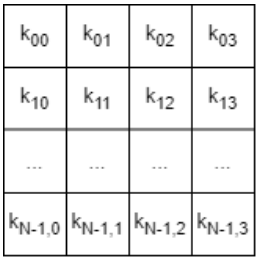

Рис. 6: Представление ключа в алгоритме

шифрования у нас будет. Процедура их нахождения рекурсивная, поэтому определим нулевой член рекурсии:

1. Нулевым вспомогательным ключом будет являться исходный ключ шифрования $K \in GF(2^8)^{4N}$, $3 < \mathrm{N} < 11$:

$$k^0 = K$$

2. Рекуррентная формула для нахождения вспомогательных ключей:

$$k^r = (\sigma(c^r) \cdot \theta \cdot \pi \cdot \gamma)(k^{r-1}), r > 0$$

Рассмотрим операции, которые выполняются для генерации вспомогательных ключей.

1. Операция γ аналогична той, которую мы рассматривали в пункте 2.1: $B = \gamma(A)$
2. Операция π является циклическим сдвигом. В данной операции столбец матрицы циклически сдвигается вниз в зависимости от номера столбца, j-ый столбец циклически сдвигается вниз на j позиций: $\pi(B) = C \leftrightarrow c_{ij} = b_{(i-j)modN,j}, 0 \le i \le N-1, 0 \le j \le 3$

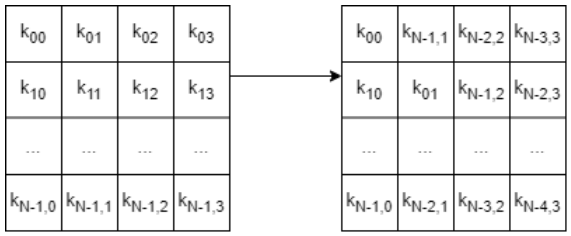

Рис. 7: Операция π

3. Операция θ аналогична той, которую мы рассматривали в пункте 2.3: $D = \theta(C)$
4. Операция γ аналогична той, которую мы рассматривали в пункте 2.4, но вместо ключа используется матрица c^r. Матрица $c^r \in M_{Nx4}[GF(2^8)], 4 \le N \le 10$ определяется следующим образом:

$$c^r_{0,j} = S[4(r-1)+j], 0 \le j \le 3$$
$$c^r_{i,j} = 0, 1 \le i < N, 0 \le j \le 3$$

, где S - табличная замена из пункта 2.1, а r - раунд шифрования

3.2 Основной ключ

Теперь рассмотрим, как генерируется основной ключ из вспомогательного. Алгоритм выглядит следующим образом:

$$K^r = (\tau \cdot \omega \cdot \gamma)(k^r), 0 \leq r \leq R;$$

Рассмотрим операции, которые выполняются для генерации основного ключа:

1. Операция γ аналогична той, которую мы рассматривали в пункте 2.1: $B = \gamma(A)$
2. Операция ω - операция извлечения ключа. Она представляет собой линейное отображение основанное на матрице $V \in M_{4xN}[GF(2^8)]$:

$$V = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & 2 & 2^2 & \dots & 2^{N-1} \\ 1 & 6 & 6^2 & \dots & 6^{N-1} \\ 1 & 8 & 8^2 & \dots & 8^{N-1} \end{pmatrix})$$

3. Операция τ аналогична той, которую мы рассматривали в пункте 2.2: $D = \tau(C)$

На выходе мы получаем ключ размером 4x4 байта.

4 Криптостойкость

Перед тем, как говорить о безопасности алгоритма, введем понятия, связанные с безопасностью.

4.1 K-безопасность

Блочный шифр является K-безопасным, если все возможные типы атак на него имеют то же ожидаемое время и требования к хранению, что и для большинства возможных блочных шифров с одинаковыми размерами. Это должно иметь место для всех возможных режимов доступа для криптоаналитика и для любого априорного распределения ключей. K-безопасность – это, по сути, относительная мера. Вполне возможно построить K-безопасный блочный шифр с 5-битными блоком данных и длиной ключа. Недостаток безопасности, обеспечиваемый такой схемой, обусловлен ее небольшими размерами, а не тем фактом, что схема не соответствует требованиям, предъявляемым этими размерами. Очевидно, что чем длиннее ключ, тем выше требования к безопасности.

4.2 Герметичный блочный шифр

Можно представить себе шифры, которые имеют определенные слабые места и все еще являются K-безопасными. По этим причинам введем еще одно понятие безопасности. Блочный шифр является герметичным, если у него нет слабых мест, которых нет у большинства блочных шифров с одинаковой длиной блока и ключа. Другими словами, блочный шифр является герметичным, если его внутренняя структура не может быть использована при какой-либо атаке.

Для всех разрешенных длин ключей ANUBIS является:
- K-безопасным;
- Герметичным.

Ожидается, что для всех определенных длин ключей Anubis будет вести себя так же, как и другой

блочный шифр с заданными длинами блока данных и ключа (в смысле K-безопасности и Герметичности). Это подразумевает, среди прочего, следующее: наиболее эффективной атакой по восстановлению ключей для ANUBIS является атака полным перебором. Получение информации из заданных пар открытый текст/зашифрованный текст о других парах открытый текст/зашифрованный текст не может быть выполнено более эффективно, чем путем определения ключа путем полного перебора. Ожидаемая сложность полного перебора ключа зависит от длины ключа шифрования и равна 2^{m-1} для m-битного ключа.

5 Эффективность алгоритма

5.1 Генерация ключа

В таблице приведена скорость генерации ключа. Генерация ключа реализована с помощью "модифицированной"таблицы S (данная таблица используется в операции γ). Модификация заключается в том, что после операции γ идет операция ω, поэтому таблицу S можно немного модифицировать с учетом умножения на матрицу V. Объем памяти, необходимый для хранения "модифицированной"таблицы S: 2 Мбайта.

Размер ключа, бит	Циклы (шифрование)	Циклы (расшифрование)
128	3352	4527
160	4445	5709
192	6644	8008
224	8129	9576
256	9697	11264
288	11385	12931
320	13475	15169

Более дорогая стоимость генерация ключей для расшифрования обусловлена операцией θ для ключей $R-1$ раунда

5.2 Шифрование и расшифрование

Поскольку ANUBIS имеет инволюционную структуру, то шифрование и расшифрование одинаково эффективны для любого количества раундов. Кроме этого для процедур шифрования и расшифрования также используется "модифицированная"таблица S, только уже с учетом матрицы H. В таблице приведена скорость на процессоре Intel Core i7-12700K:

Размер ключа, бит	Циклы/байт	Циклы/блок	Мбит/с
128	36.8	589	793
160	39.3	628	744
192	41.6	665	703
224	43.8	701	667
256	46.3	740	631
288	48.5	776	602
320	50.8	813	575

6 Достоинства и недостатки

Рассмотрим достоинства алгоритма:

1. Аппаратное. ANUBIS довольно быстрый алгоритм. При этом он не требует большого объема памяти (как для кода, так и для вспомогательных таблиц). Кроме того, есть возможность параллельного исполнения (можно вычислять ключи параллельно с шифротекстом/открытым текстом).

2. Инволюционная структура. Тот факт, что все компоненты алгоритма ANUBIS являются инволюциями, уменьшает размер кода, который осуществляет шифрование/расшифрование.

3. Сложная генерация ключей. Преимуществом является повышенная устойчивость к атакам на ключи.

Рассмотрим недостатки алгоритма:

1. Медленная генерация ключей. За устойчивость к атакам приходится платить скоростью. Но несмотря на это алгоритм все еще остается довольно быстрым.

2. ANUBIS схож с другим блочным криптоалгоритмом Rijndael (он является стандартом шифрования в США). Но несмотря на его схожесть, он не имеет серьезных преимуществ перед Rijndael. Поэтому он не используется как стандарт.

7 Заключение

В работе рассмотрен алгоритм ANUBIS. Данный алгоритм обеспечивает быстрое и надежное шифрование. На конкурсе NESSIE он был признан одним из самых быстрых и криптостойких алгоритмов. Если бы не большое сходство с Rinjdael, он мог бы быть стандартом шифрования в США.

8 Ссылки

1. https://web.archive.org/web/20160606112246/http://www.larc.usp.br/ pbarreto/AnubisPage.html

2. https://www.phantastike.com/encryption/algoritmy_shifrovaniya/djvu/view/

3. https://ru.wikipedia.org/wiki/

4. https://ru.wikipedia.org/wiki/