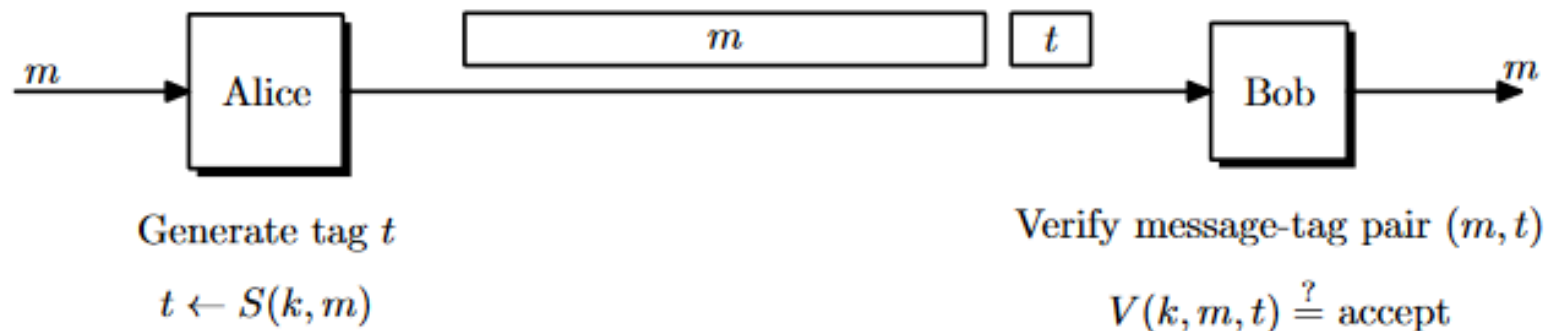


# Прикладная Криптография: Симметричные криптосистемы MAC: схемы

Макаров Артём  
МИФИ 2020

# Целостность сообщений

- Задача – обеспечить целостность сообщений  $m$  при передаче
- Обеспечиваем только **целостность**, сообщения предполагаются открытыми
- Основная идея – создать небольшую по длине величину  $t$  (tag, метка) на основе сообщения, и передать данную величину вместе с сообщением:  $(m, t)$ . На стороне получателя величина  $t'$  вычисляется для полученного сообщения  $m'$  и производится сравнение  $t = t'$ . В случае равенства полагается, что целостность сообщения не нарушена.



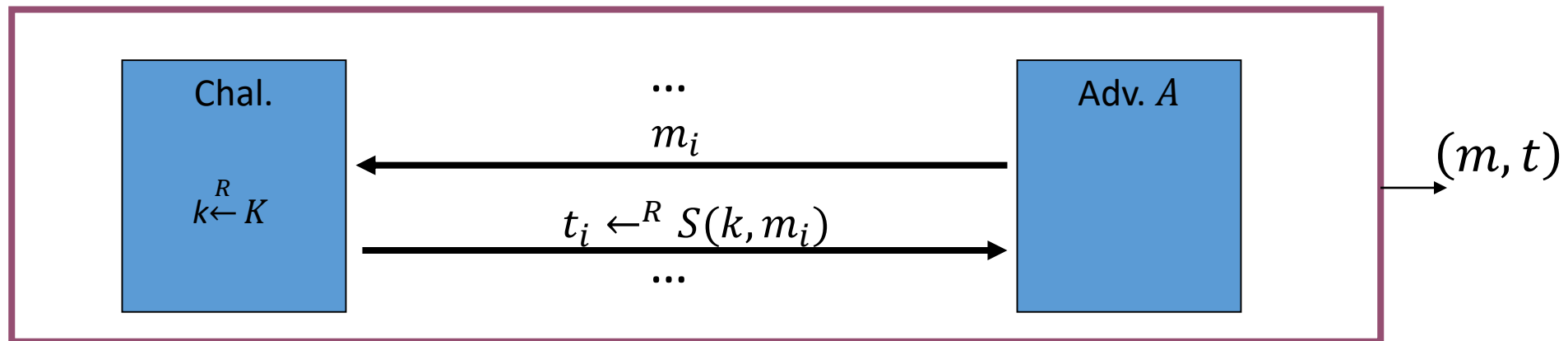
# Определение MAC

MAC на  $(K, M, T)$  называется пара эффективных алгоритмов  $I = (S, V)$ .  $S$  – алгоритм выработки MAC,  $V$  – алгоритм проверки MAC. Пусть  $M$  – множество сообщений,  $K$  – множество ключей,  $T$  – множество кодов аутентичности (меток). Тогда для  $m \in M, t \in T, k \in K$

- $S: K \times M \rightarrow T$  - вероятностный алгоритм, вычисляющий  $t \leftarrow^R S(k, m)$
- $V: K \times M \times T \rightarrow \{0,1\}$  – детерминированный алгоритм, вычисляющий результат проверки  $r \leftarrow V(k, m, t)$ .
- Свойство корректности -  $\Pr[V(k, m, S(k, m)) = 1] = 1$

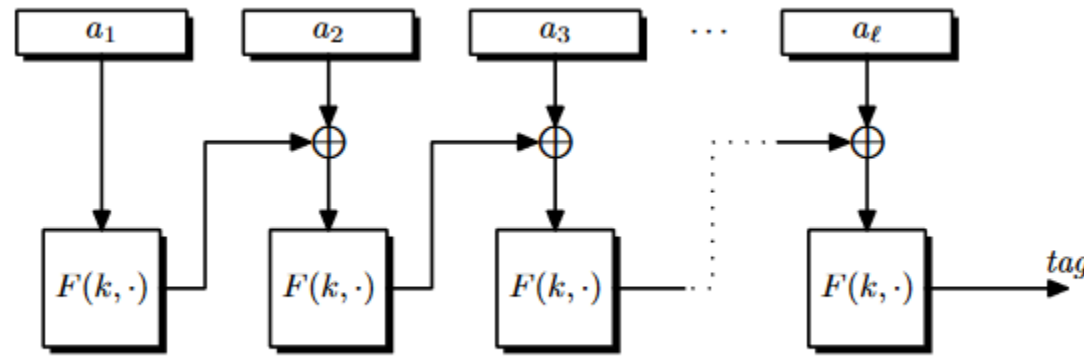
# Игра на стойкость MAC (chosen message attack)

- Противник побеждает в игре, если пара  $(m, t)$  – верная пара сообщение – MAC, т.е.  $V(k, m, t) = 1$ .
- Преимуществом противника  $A$  в игре против MAC  $I = (S, V)$  называется величина  $MAC_{adv}[A, I] = \Pr[V(k, m, t) = 1]$ .
- MAC  $I = (S, V)$  называется стойким MAC, если  $\forall A \text{ } MAC_{adv}[A, I] \leq \epsilon$ ,  $\epsilon$  – пренебрежимо малая величина.



# Беспрификсные PRF

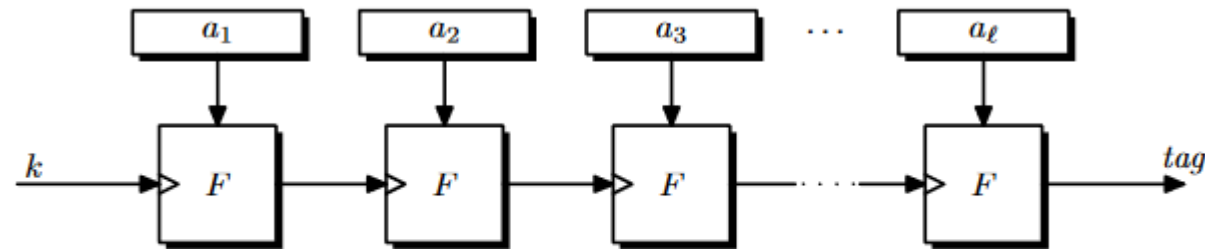
PRF  $F_{CBC}(k, m)$  – цепочка CBC с использованием PRF. В качестве значения используется последний элемент цепочки.



(a) The CBC construction  $F_{CBC}(k, m)$

# Беспрификсные PRF

PRF  $F^*(k, m)$  – каскадная конструкция. Выход каждой итерации PRF используется в качестве ключа в следующей итерации PRF.



(b) The cascade construction  $F^*(k, m)$

**Figure 6.3:** Two prefix-free secure PRFs

# Беспрификсные PRF

**Теорема 9.3.** Пусть  $F$  – стойкая PRF на  $(K, X, X)$ ,  $X = \{0,1\}^n$ . Для полиномиально ограниченной величины  $l$  PRF  $F_{CBC}: K \times X^{\leq l} \rightarrow X$  является стойкой беспрификсной PRF, причём для любого беспрификсного противника  $A$ , делающего не более  $Q$  запросов существует противник в игре на PRF, причём

$$PRF^{pf}[A, F_{CBC}] \leq PRF_{adv}[B, F] + (Ql)^2 / 2|X|$$

**Теорема 9.4.** Пусть  $F$  – стойкая PRF на  $(K, X, K)$ . Для полиномиально ограниченной величины  $l$  PRF  $F^*: K \times X^{\leq l} \rightarrow K$  является стойкой беспрификсной PRF, причём для любого беспрификсного противника  $A$ , делающего не более  $Q$  запросов существует противник в игре на PRF, причём

$$PRF^{pf}[A, F_{CBC}] \leq Ql * PRF_{adv}[B, F]$$

▷ без доказательства◁

# Построение стойкий PRF на основе беспрификсных PRF

Рассмотрим 3 способа построения PRF на основе беспрификсных PRF:

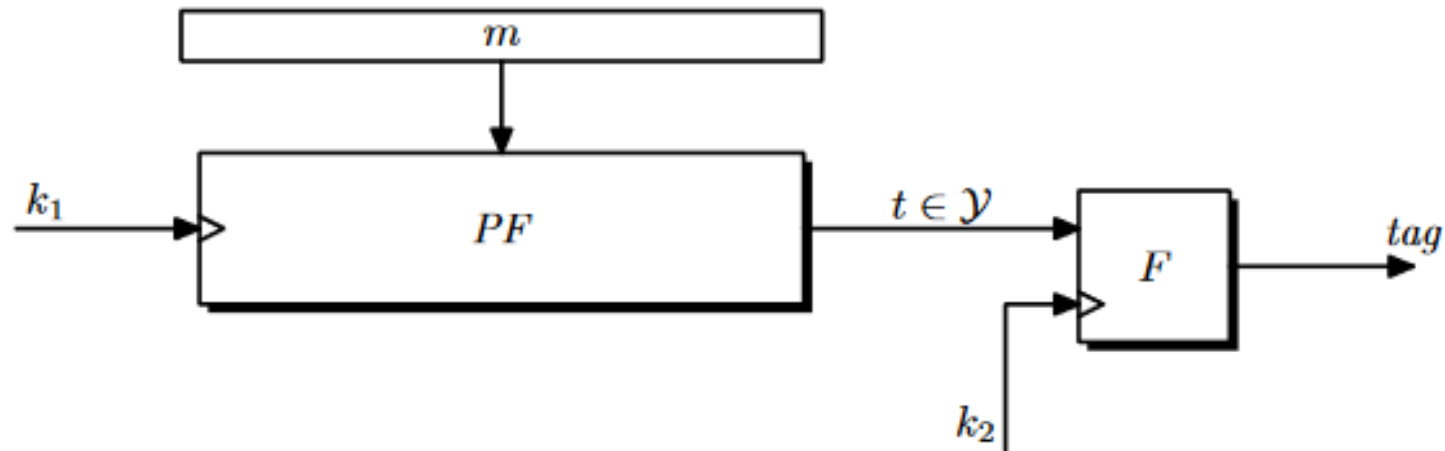
- Зашифрование выхода беспрификсной PRF: зашифрование выхода беспрификсной PRF с использованием другой PRF
- Беспрификсное кодирование: преобразовать входные данные так, чтобы все они были беспрификсными
- Беспрификсное кодирование с рандомизацией: CMAC



# Зашифрование выхода беспрификсной PRF

Пусть  $PF - \text{PRF}: K_1 \times X^{\leq l} \rightarrow Y$ ,  $F - \text{PRF}: K_2 \times Y \rightarrow T$ .

Определим  $EF((k_1, k_2), m) = F(k_2, PF(k_1, m))$ ,  $k_1 \in K_1, k_2 \in K_2, m \in X^{\leq l}$



# Зашифрование выхода беспрификсной PRF

Пусть  $PF$  – PRF на  $(K, X^{\leq l}, Y)$ .  $PF$  является **расширяемой PRF**, если  $\forall k \in K, x, y \in X^{\leq l-1}, a \in X$ :

$$PF(k, x) = PF(k, y) \Rightarrow PF(k, x||a) = PF(k, y||a).$$

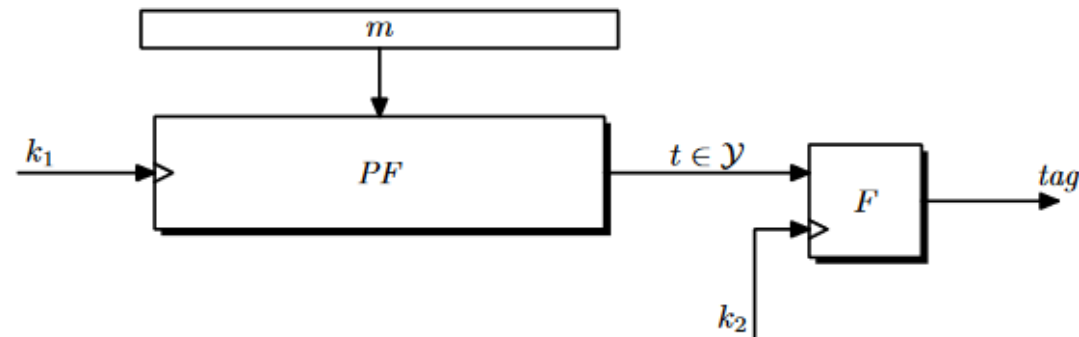
PRF CBC и каскадной конструкции являются расширяемыми.

# Зашифрование выхода беспрификсной PRF

Если  $PF$  функция CBC или каскадная конструкция то PRF  $EF$  – стойкая PRF.

**Теорема 10.1.** Пусть  $PF$  – расширяемая беспрификсная PRF на  $(K_1, X^{\leq l+1}, Y)$ ,  $|Y|$  - сверх полиномиальная,  $l$  – полиномиально ограниченная. Пусть  $F$  – стойкая PRF на  $(K_2, Y, T)$ . Тогда  $EF$  определённая ранее – стойкая PRF на  $(K_1 \times K_2, X^{\leq l}, T)$ :

$$PRF_{adv}[A, EF] \leq PRF_{adv}[B_1, F] + PRF_{adv}^{pf}[B_2, PF] + Q^2/2|Y|$$

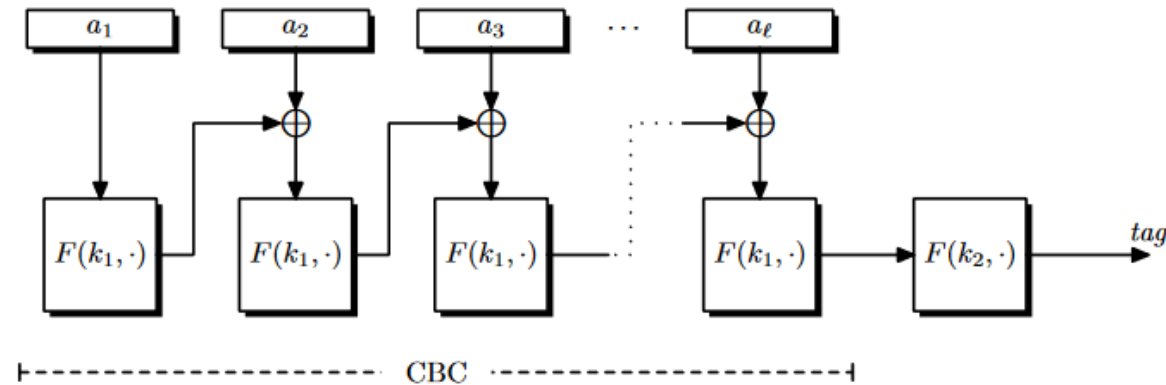


# Зашифрование выхода беспрификсной PRF

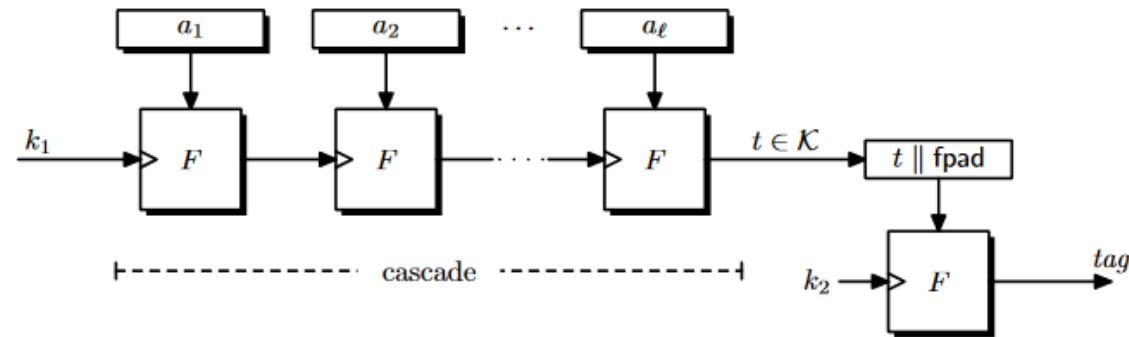
▷ Рассмотрим идею доказательства. Самая неочевидная часть результирующей формулы  $PRF_{adv}[A, EF] \leq PRF_{adv}[B_1, F] + PRF_{adv}^{pf}[B_2, PF] + Q^2/2|Y|$  это слагаемое  $Q^2/2|Y|$ . Рассмотрим причину его появления.

Пусть противник запрашивает у оракула (претендента)  $Q$  кодов аутентичности для  $Q$  различных сообщений. Так как размер области значений  $PF$  есть  $|Y|$ , то используя парадокс дней рождений за  $Q \sim \sqrt{|Y|}$  произойдёт коллизия, и итоговое значение MAC тоже даст коллизию. Т.е. мы нашли пару  $x_i, x_j: PF(k_1, x_i) = PF(k_1, x_j)$ . Так как  $PF$  расширяемая, то противник имея MAC  $t$  для сообщения  $x_i || a$  фактически имеет MAC для сообщения  $x_j || a$ . ◁

# Зашифрование выхода беспрификсной PRF



(a) The ECBC construction  $ECBC(k, m)$  (encrypted CBC)



(b) The NMAC construction  $NMAC(k, m)$  (encrypted cascade)

# ECBC MAC

**Теорема 10.2.** Зашфированный CBC MAC *ECBC*, зашифрованный с использованием PRF  $F$  на  $(K, X, X)$  ( $|X|$  - сверх полиномиальная,  $l$  – полиномиально ограниченная) – стойкая PRF на  $(K^2, X^{\leq l}, X)$ :

$$PRF_{adv}[A, ECBC] \leq PRF_{adv}[B_1, F] + PRF_{adv}[B_2, F] + \frac{(Q(l+1))^2 + Q^2}{2|X|}$$

▷ следствие **Теоремы 10.1.** ◁

# NMAC

- PRF  $F$  на  $(K, M, K)$ ,  $K = \{0,1\}^k$ ,  $X = \{0,1\}^n$ ,  $k \leq X$
- $g(t) = t || \text{fpad}$ ,  $\text{fpad}$  – фиксированное дополнение, длины  $n - k$  бит (например все 0).

**Теорема 10.3.** NMAC, использующая PRF  $F$  стойкая PRF на  $(K^2, X^{\leq l}, K)$ :

$$PRF_{adv}[A, NMAC] \leq (Q(l + 1)) * PRF_{adv}[B_1, F] + PRF_{adv}[B_2, F] + \frac{Q^2}{2|K|}$$

▷ следствие **Теоремы 10.1.** ◁

# NMAC и ECBC MAC

- Рассмотренные конструкции являются стойкими PRF и следовательно стойкими MAC
- Нет необходимости знать длину сообщения заранее, можно обновлять полученное значение MAC при получении новых блоков сообщения, не дожидаясь получения сообщения целиком
- Можно использовать для сообщений произвольной длины, **кратной размеру блока** PRF (чаще всего – блочного шифра)



# Беспрификсное кодирование

Цель – закодировать «префиксные строки» в непрификсные, для использования в непрификсных PRF для получения MAC

Пусть  $X_{>0}^{\leq l}$  - множество непустых строк, длины не более  $l$  элементов в  $X$ .

Функция  $pf: M \rightarrow X_{>0}^{\leq l}$  называется непрификсным кодированием, если  $pf$  – инъективна и множество элементов из образа  $pf$  – непрификсное множество.

**Теорема 10.4.** Пусть  $pf$  – непрификсное кодирование,  $PF$  – непрификсная PRF на  $(K, X^{\leq l}, Y)$ . Тогда PRF  $F(k, m) = PF(k, pf(m))$  – стойкая PRF на  $(K, M, Y)$

▷Очевидно следует из определения непрификсной PRF◁

# Беспрификсное кодирование

- Метод 1. Добавление длины

$$X = \{0,1\}^n, M = X^{\leq l-1}, m = (a_1, \dots, a_v) \in M$$
$$pf(m) = (< v >, a_1, \dots, a_v) \in X_{>0}^{\leq l}$$

- Метод 2. «Остановочные биты»

$$X = \{0,1\}^n, \tilde{X} = \{0,1\}^{n-1}, M = \tilde{X}_{>0}^{\leq l}, m = (a_1, \dots, a_v) \in M$$
$$pf(m) = ((a_1 || 0), (a_2 || 0), \dots, (a_{v-1} || 0), (a_v || 1)) \in X_{>0}^{\leq l}$$

▷Очевидна инъективность и беспрификсность образа◁

# Беспрификсное кодирование

- Позволяет использовать непрификсные PRF в качестве MAC
- Добавление длины сообщения увеличивает длину сообщений как входа для непрификсной PRF, так как непрификсное кодирование – избыточно.
- Добавление длины к сообщению не позволяет использовать MAC в поточном режиме (когда сообщение передаётся по частям), так как длина сообщения заранее не известна
- Так как в основном используются блочные шифры – добавление данных непрификсным кодирование означает добавление лишних блоков
- Использование «остановочных битов» также увеличивает длину сообщения

# Беспрификсное кодирование с рандомизацией

Пусть  $x, y \in X^{\leq l}$ . Обозначим  $x \sim y$  если  $x$  префикс  $y$  или  $y$  префикс  $x$  (т.е.  $\sim$  отношение «префиксности» на  $X^{\leq l}$ ).

Пусть  $\epsilon$  – действительное число,  $0 \leq \epsilon \leq 1$ . Вероятностное  $\epsilon$ -префиксное кодирование это функция  $prf: K \times M \rightarrow X_{\geq 0}^{\leq l}: m_0, m_1 \in M, m_0 \neq m_1$ :

$$\Pr[prf(k, m_0) \sim prf(k, m_1)] \leq \epsilon$$

Где вероятность рассматривается при случайном равновероятном выборе  $k \in K$

Пример -  $prf(k, (a_1, \dots, a_v)) = (a_1, \dots, a_v, (a_v \oplus k)) \in X_{\geq 0}^{\leq l}$

# Беспрификсное кодирование с рандомизацией

Пусть  $PF$  – непрификсная PRF на  $(K, X^{\leq l}, Y)$ ,  $prf: K_1 \times M \rightarrow X_{>0}^{\leq l}$  – вероятностное  $\epsilon$ -префиксное кодирование.

Определим PRF  $F$  на  $(K \times K_1, M, Y)$ :

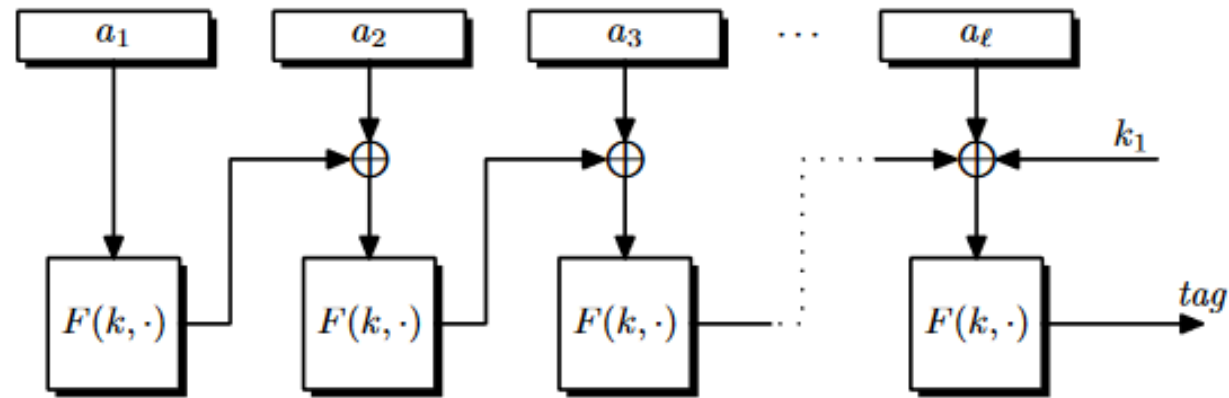
$$F((k_1, k_2), m) = PF(k, prf(k_1, m))$$

**Теорема 10.5.** Если  $PF$  - непрификсная PRF,  $prf$  - вероятностное  $\epsilon$ -префиксное кодирование, тогда  $F$ , введённая выше – стойкая PRF:

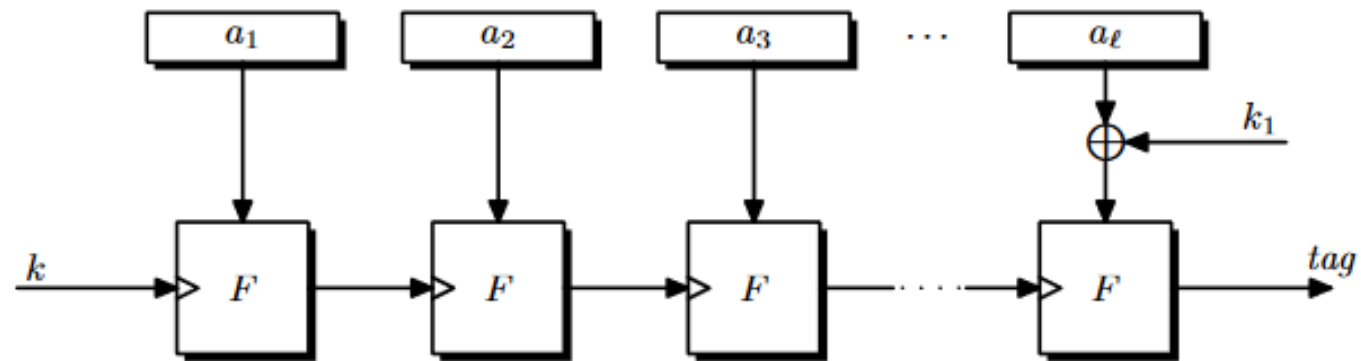
$$PRF_{adv}[A, F] \leq PRF_{adv}^{pf}[B_1, PF] + PRF_{adv}^{pf}[B_2, PF] + \frac{Q^2 \epsilon}{2}$$

▷ без доказательства ◁

# Беспрификсное кодирование с рандомизацией



(a) *rpf* applied to CBC



(b) *rpf* applied to cascade

# MAC для сообщений, некратных длине блока

Все рассмотренные до этого схемы были применимы только для сообщений длины кратных длине блока PRF (блочного шифра).

Пусть  $F$  – PRF на  $(K, X^{\leq l+1}, Y)$ ,  $inj: \{0,1\}^{\leq nl} \rightarrow X^{\leq nl}$  – инъекция.  
Определим PRF  $F_{bit}$ :

$$F_{bit} = F(k, inj(x))$$

**Теорема 10.6.** PRF введенная выше – стойкая PRF на  $(K, \{0,1\}^{\leq nl}, Y)$

▷очевидно◁

# Построение инъективных функций

Пусть  $X = \{0,1\}^n$ ,  $inj: \{0,1\}^{\leq nl} \rightarrow X^{\leq l+1}$

$inj$ :

- Если входное сообщение имеет длину не кратную  $n$  – добавить 10...00 до длины кратной  $n$
- Иначе – добавить  $n$ -блок  $(1||0^{n-1})$
- Инъективна и обратима

case 1: 

$a_1$	$a_2$
-------	-------

 $\longrightarrow$ 

$a_1$	$a_2$	1000
-------	-------	------

case 2: 

$a_1$	$a_2$
-------	-------

 $\longrightarrow$ 

$a_1$	$a_2$	1000000
-------	-------	---------

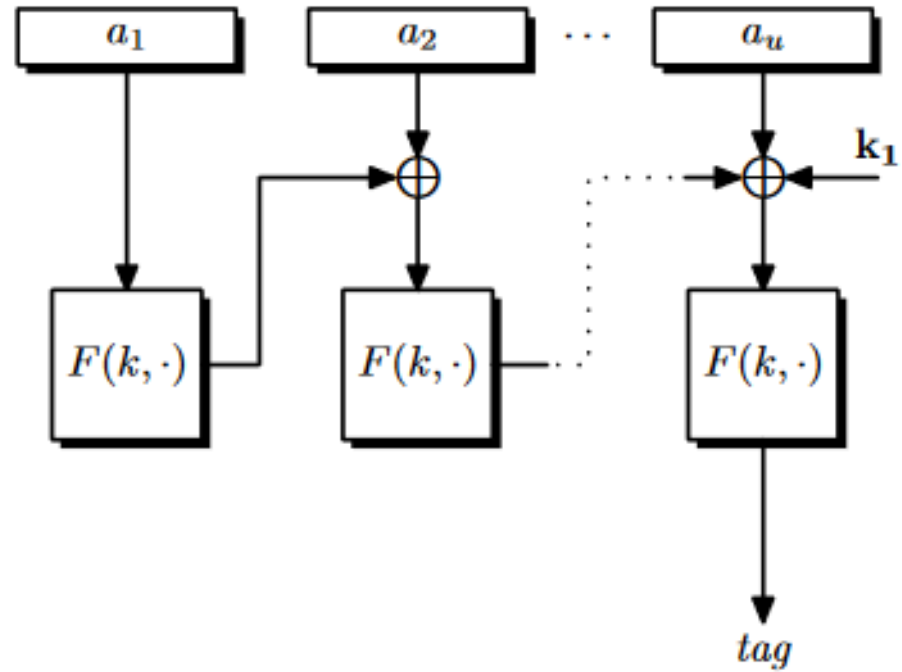


# CMAC

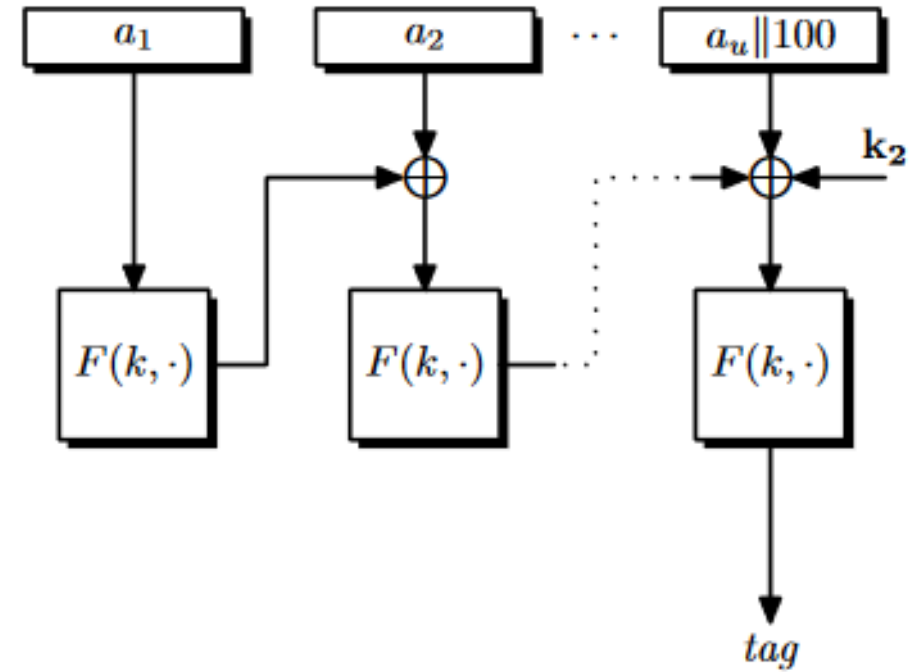
- Стандарт NIST
- Один из наиболее популярных алгоритмов вычисления MAC (самый популярных после HMAC)
- Использует три различных ключа (могут быть выработаны на основе одного ключа)

# CMAC

(a) when  $\text{length}(m)$  is a positive multiple of  $n$



(b) otherwise



# OMAC

- В текущей вариации (OMAC) использует единственный ключ для генерации этих трех ключей для некоторой константы  $R_n$ :

input: key  $k \in \mathcal{K}$

output: keys  $k_0, k_1, k_2 \in \mathcal{X}$

$k_0 \leftarrow k$

$L \leftarrow F(k, 0^n)$

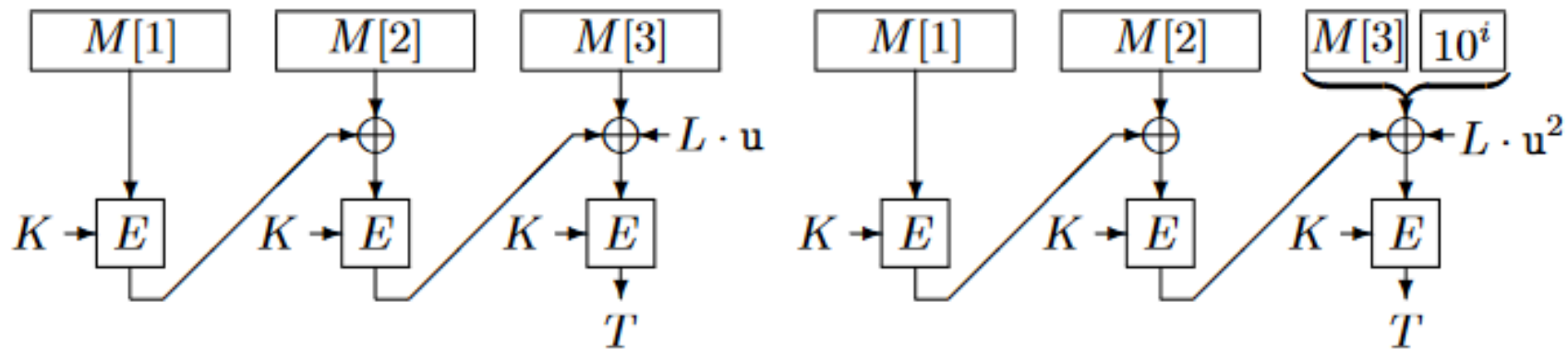
(1) if  $\text{msb}(L) = 0$  then  $k_1 \leftarrow (L \ll 1)$  else  $k_1 \leftarrow (L \ll 1) \oplus R_n$

(2) if  $\text{msb}(k_1) = 0$  then  $k_2 \leftarrow (k_1 \ll 1)$  else  $k_2 \leftarrow (k_1 \ll 1) \oplus R_n$

output  $k_0, k_1, k_2$ .

# OMAC

- Фактически для получения трех ключей реализуется умножение в кольце многочленов на некоторую константу  $u$



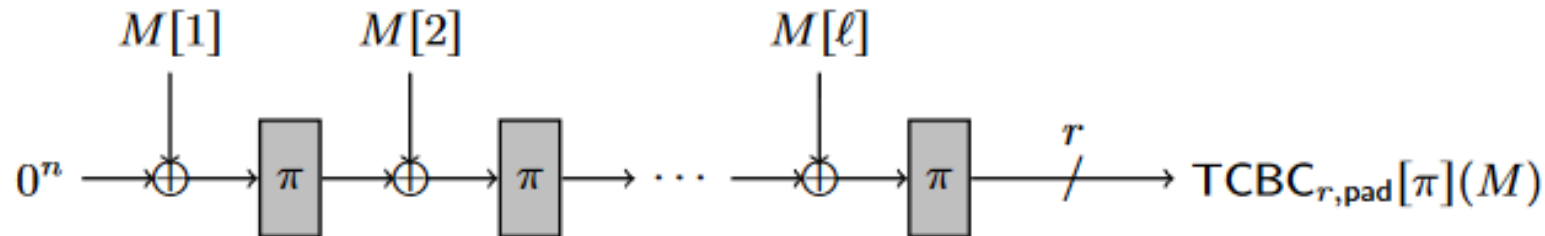
# Trunkated CBC MAC

Основная идея – не дать противнику возможность воспользоваться MAC для осуществления префиксной атаки.

Использование части кода аутентичности. Используется в ГОСТ 28147-98

Оптимально использовать половину исходного MAC

Основной недостаток – фактически понижаем достижимый параметр стойкости в 2 раза



**Fig. 1. Truncated CBC.** Representation of  $\text{TCBC}_{r,\text{pad}}[\pi]$ . Here,  $M[1], \dots, M[\ell]$  are  $n$ -bit blocks resulting from applying the padding scheme  $\text{pad}$  to the input message  $M \in \{0, 1\}^*$ .

# PMAC

PMAC – параллельный MAC

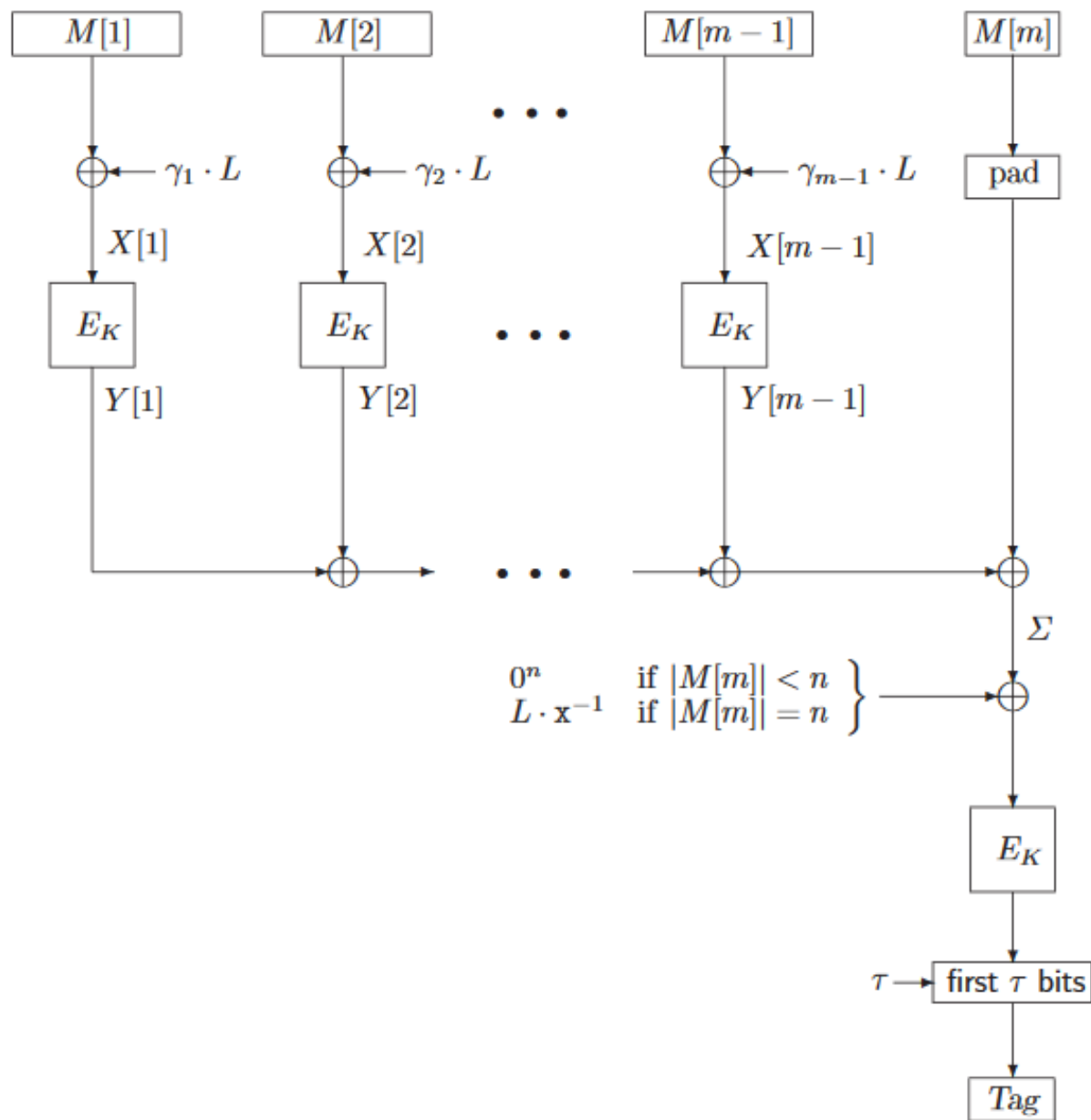
Возможность добавлять и удалять блоки из итогового значения MAC

Основная идея – использование «различных» ключей для каждого блока, полученных через умножение в кольце многочленов

Возможность вычислять MAC параллельно для всех блоков

Был патентован (США), разрешено бесплатное использование в образовательных и open-source проектах. В настоящий момент патент истёк

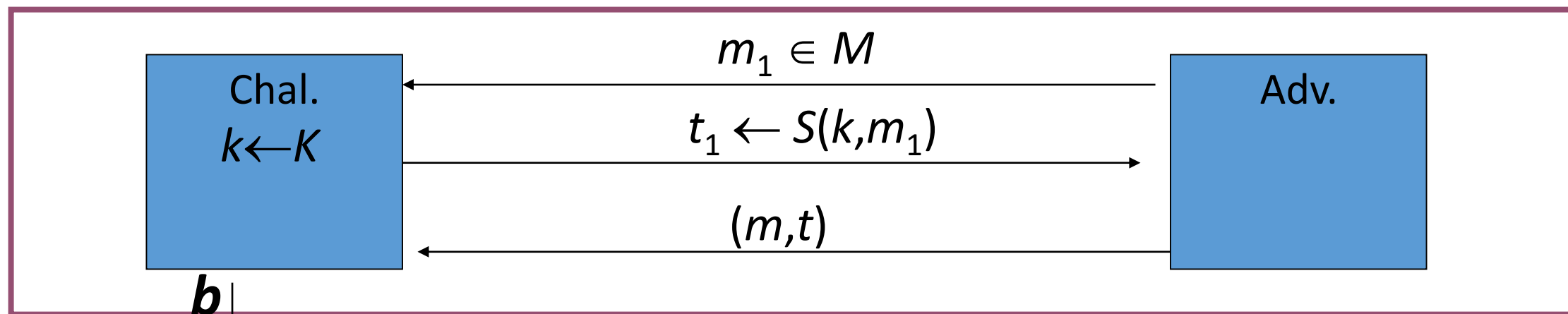
# PMAC



# Одноразовый MAC

(по аналогии с одноразовым блокнотом)

Введём игру



$$\begin{cases} \mathbf{b}=1 & \text{если } V(k, m, t) = 1 \text{ and } (m, t) \neq (m_1, t_1) \\ \mathbf{b}=0 & \text{иначе} \end{cases}$$

$I = (S, V)$  стойкий одноразовый MAC, если  $Adv_{mac1}[A, I] = \Pr[b = 1] \leq \epsilon$ ,  $\epsilon$  – пренебрежимо малая величина



# Одноразовый МАС : пример

Стойкий против любых (не только эффективных) противников

Пусть  $q$  большое простое число (пример (хоть и не больше) -  $q = 2^{128} + 51$  )

$$key = (a, b) \in \{1, \dots, q\}^2$$

$$msg = (m[1], \dots, m[L])$$

$P_{msg}(x) = x^{L+1} + m[L] * x^L + \dots + m[1] * x$  – полином степени  $L + 1$

$$S(key, msg) = P_{msg}(a) + b \pmod{q}$$

# Одноразовый MAC $\Rightarrow$ Многоразовый MAC

Пусть  $(H, V)$  стойкий одноразовый MAC на  $(K, M, \{0,1\}^n)$ .

Пусть  $F: K_F \times \{0,1\}^n \rightarrow \{0,1\}^n$  стойкая PRF.

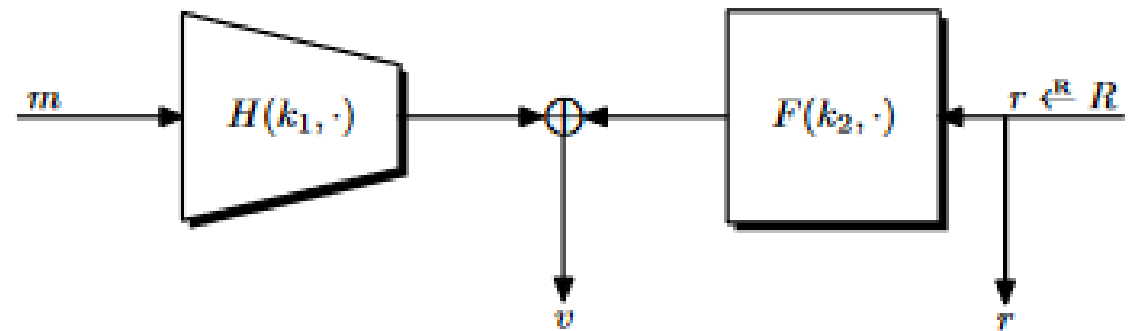
$r \leftarrow^R R$ .  
Медленная, Быстрая,  
Короткий вход Длинный вход

Carter-Wegman MAC:

$$S_{cw}((k_1, k_2), m) = (r, F(k_2, r) \oplus H(k_1, m)) = (r, v)$$

$$V_{cw}(k, (r, v), m) = \begin{cases} 1, & v = F(k_2, r) \oplus H(k_1, m) \\ 0, & \text{иначе} \end{cases}$$

Является недетерминированным



# Carter-Wegman MAC

Наиболее быстрые современные MAC

- VMAC
- UMAC
- Poly1305-AES

# Poly1305

Poly1305:

Пусть  $m[0], m[1], \dots, m[l - 1]$  – сообщение,  $q = \frac{l}{16}$  (округление сверху)

$$c_i = m[16i - 16] + 2^8 m[16i - 15] + \\ + 2^{16} m[16i - 14] + \dots + 2^{120} m[16i - 1] + 2^{128}$$

Если 16 не делит  $l$ :

$$c_q = m[16q - 16] + 2^8 m[16q - 15] + \dots + \\ + 2^{8(l \bmod 16) - 8} m[l - 1] + 2^{8(l \bmod 16)}$$

Простыми словами – дополнить каждые 16 байт до 17, добавляя 1. Если не хватает до 16 байт – добавить 100...000 чтоб хватало.

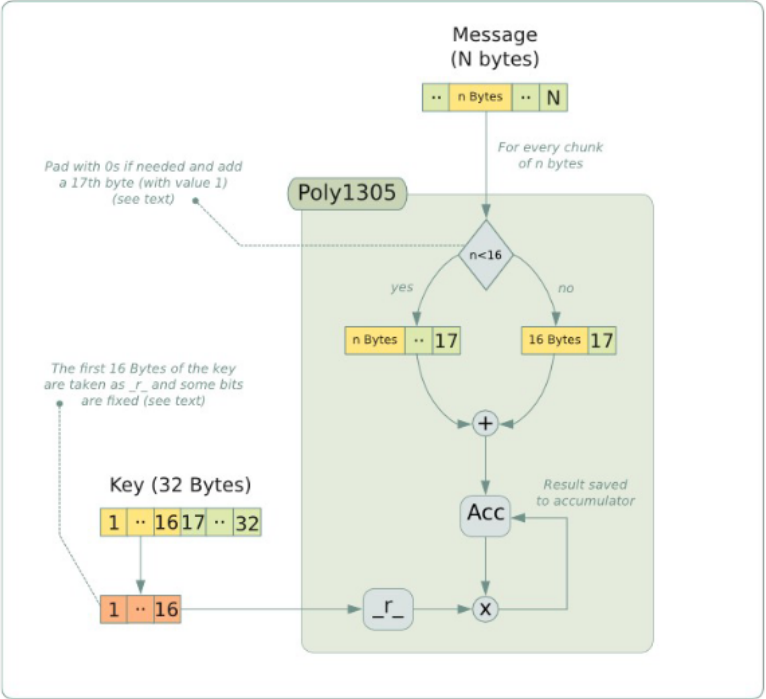
# Poly1305

$$\text{Poly1305}_r(m, \text{AES}_k(n)) \\ = \left[ \left( (c_1 r^q + c_2 r^{q-1} + \dots + c_q r^1) \bmod 2^{130} - 5 \right) \right]$$

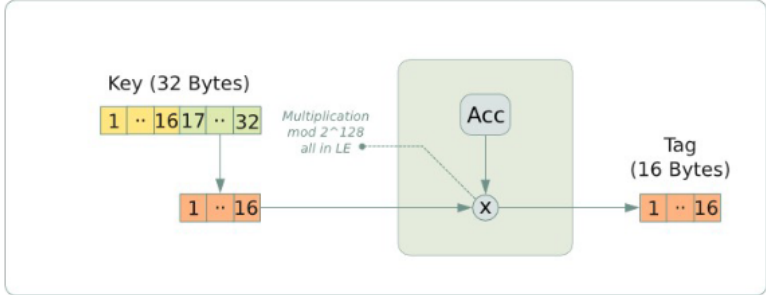
```
#include <gmpxx.h>

void poly1305_gmpxx(unsigned char *out,
    const unsigned char *r,
    const unsigned char *s,
    const unsigned char *m, unsigned int l)
{
    unsigned int j;
    mpz_class rbar = 0;
    for (j = 0; j < 16; ++j)
        rbar += ((mpz_class) r[j]) << (8 * j);
    mpz_class h = 0;
    mpz_class p = (((mpz_class) 1) << 130) - 5;
    while (l > 0) {
        mpz_class c = 0;
        for (j = 0; (j < 16) && (j < l); ++j)
            c += ((mpz_class) m[j]) << (8 * j);
        c += ((mpz_class) 1) << (8 * j);
        m += j; l -= j;
        h = ((h + c) * rbar) % p;
    }
    for (j = 0; j < 16; ++j)
        h += ((mpz_class) s[j]) << (8 * j);
    for (j = 0; j < 16; ++j) {
        mpz_class c = h % 256;
        h >>= 8;
        out[j] = c.get_ui();
    }
}
```

1. First, initialize `_r_` then process groups of 16 bytes



2. Finally, add the last 16 bytes of the key and generate the tag



## Poly 1305



### Introduction

- Poly1305 is a Wegman-Carter, one-time authenticator designed by D. J. Bernstein
- It is used to calculate a Message Authentication Code (MAC) for a message
- Poly 1305 uses a 32 Byte key and operates on an N byte message

### Operation

- The first 16 bytes of the one-time key are interpreted as a number `_r_` with the following modifications:
  - ▶ The top 4 bits of bytes 3, 7, 11, 15 are set to 0
  - ▶ The bottom 2 bits of bytes 4, 8, 12 are set to 0
  - ▶ The 16 bytes are interpreted as a little endian value
- The accumulator (Acc in the diagram) is set to 0
- For every n bytes read from the N byte message, if  $n = 16$  then just add a 17th byte having a value of 1 and the 17 bytes are treated as a little endian number
- If  $n < 16$  then pad with 0s until there are 16 bytes and add the 17th byte as in the case when  $n = 16$
- The number is then added to the accumulator which is multiplied by `_r_` and the result is saved back to the accumulator
- Note : These operations are all mod  $2^{130} - 5$
- Finally, the last 16 bytes of the key are interpreted as a little endian number and this number is added to the accumulator mod  $2^{128}$
- The result is then written out as a little endian number and this is taken as the 16 byte tag

NB: картинка слева предполагает, что правая часть ключа Poly1305 уже получена ранее как выход некоторой PRF, вычисленной от некоторого  $n$  поэтому явного вычисления PRF нет.

В качестве PRF используется не только AES, но и ChaCha20 (обычно в рамках построения аутентифицированного шифрования ChaCha 20 – Poly 1305)