УДК 003.26

Дали Ф.А.

Технический комитет по стандартизации «ТК26»

Миронкин В.О.

Национальный исследовательский университет Высшая школа экономики, Москва

О ДРЕВОВИДНЫХ РЕЖИМАХ РАБОТЫ ХЭШ-ФУНКЦИЙ

В статье предложены две модели древовидных режимов работы хэш-функций. Для каждой модели построены алгоритмы вычисления хэш-кода и найдены их численные характеристики. В терминах соответствующих моделей классифицирован ряд действующих алгоритмов параллельного хэширования, а также выявлены некоторые присущие им слабости.

ХЭШ-ФУНКЦИЯ, РЕЖИМ, ДЕРЕВО ХЭШИРОВАНИЯ, АЛГОРИТМ, ТРУДОЕМКОСТЬ, ВПИТЫВАНИЕ, УСЕЧЕНИЕ, КОЛЛИЗИЯ, ВТОРОЙ ПРООБРАЗ

Dali F.A.

Technical Committee for standardization «TC26», Moscow

Mironkin V.O.

National Research University Higher School of Economics, Moscow

ON THE TREE MODES OF HASH FUNCTIONS

Two models of the tree modes of hash functions are introduced. For each model algorithms of computing of the hash code are formulated and their numerical characteristics are obtained. In terms of the constructed models we classify some existing algorithms for parallel hashing and identify some weaknesses of corresponding primitives.

HASH FUNCTION, MODE, TREE, HASHING, ALGORITHM, COMPLEXITY, ABSORBING PHASE, SQUEEZING PHASE, COLLISION, THE SECOND PREIMAGE

Введение

В настоящее время одним из существенных требований, предъявляемых к режимам работы хэш-функций, является возможность распараллеливания процесса вычислений. Практическая реализация широко используемой итерационной конструкции Меркла-Дамгарда (рис. 1) уже не обеспечивает эффективной обработки большого объема данных, что требует построения принципиально новых моделей хэширования.

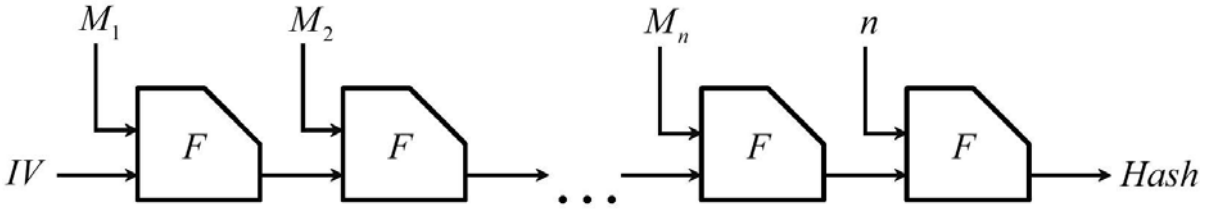

Рис. 1. Конструкция Меркла-Дамгарда

Одним из наиболее наглядных способов описания процесса функционирования режимов, обеспечивающих распараллеливание вычислений, является схематичное представление в виде ориентированных графов [2], например, деревьев Меркла [15]. Для подобных режимов введем следующее определение.

Определение 1. *Режимы работы хэш-функций, которые могут быть представлены с помощью ориентированных графов, будем называть древовидными режимами, а соответствующие графы - деревьями хэширования.*

Согласно [6-8, 15] вершины дерева хэширования будем называть *узлами*. Узлы, не имеющие предков, будем называть *листьями*, а единственный узел, не имеющий потомков, - *корнем*.

Из работы [17] следует существование стойких к коллизиям хэш-функций, не отличающихся от случайных равновероятных отображений [1, 4] и допускающих работу в древовидных режимах [10, 12].

Для дальнейшего изложения будем использовать понятия теории графов, в

частности, разделов, посвященных деревьям [2, 3, 5], а также терминологию и обозначения, введенные в работах [6-8]:

V^*	множество всех двоичных векторов-строк конечной длины, включая пустую строку;
V_n	множество всех двоичных векторов длины n, $n \in \mathbb{N} \cup \{0\}$;
$A \| B$	конкатенация векторов $A, B \in V^*$ - вектор из $V_{\|A\|+\|B\|}$, полученный приписыванием вектора B к вектору A справа;
M	хэшируемое сообщение, $M \in V^*$;
$H: V^* \to V_t$	хэш-функция, преобразующая сообщение $M \in V^*$ в хэш-код $H(M) \in V_t$;
$\mathcal{P}$	множество всех узлов дерева хэширования;
$\mathcal{L}$	множество листьев дерева хэширования;
$\mathcal{I}$	множество узлов дерева хэширования, не являющихся листьями;
l	арность дерева - число предков, с использованием которых формируется каждый узел из $\mathcal{I}$;
m	размер узлов из $\mathcal{P}$ в битах;
$h: V_m \to V_t$	функция сжатия, преобразующая произвольный вектор $x \in V_m$ в вектор $h(x) \in V_t$, $m > t$;
$LSB_n: V^* \to V_n$	отображение, ставящее в соответствие вектору $z_{k-1} \| \ldots \| z_1 \| z_0$, $k \geq n$, вектор $z_{n-1} \| \ldots \| z_1 \| z_0$;
$MSB_n: V^* \to V_n$	отображение, ставящее в соответствие вектору $z_{k-1} \| \ldots \| z_1 \| z_0$, $k \geq n$, вектор $z_{k-1} \| z_{k-2} \| \ldots \| z_{k-n}$;
$\lceil v \rceil$	целая часть сверху числа $v \in \mathbb{R}$: $\lceil v \rceil = \min\{n \in \mathbb{Z} : n \geq v\}$;
T	число параллельных процессов вычисления функции h, $T \in \mathbb{N}$;
M_i	i-й слой дерева хэширования, $i \in \overline{1, k+1}$;
n_i	длина i-го слоя дерева хэширования (суммарный размер узлов i-го слоя);

Определение 2. *Дерево хэширования называется l-арным, если каждый его узел из $\mathcal{I}$ имеет не более l предков.*

Определение 3. *l-арное дерево хэширования называется полным, если каждый его узел из $\mathcal{I}$ имеет в точности l предков.*

Определение 4. *Путем в дереве хэширования из узла u в узел v длины l называется дуга (u,v), а длины $k+1$ - произвольная последовательность дуг вида $(u,w_1),(w_1,w_2),...,(w_{k-1},w_k),(w_k,v)$, где $k \in \mathbb{N}$, $w_1,...,w_k$ - произвольные узлы из $\mathcal{I}$.*

Определение 5. *Расстоянием между двумя узлами дерева хэширования, связанных путем, называется длина этого пути.*

Определение 6. *Высотой k дерева хэширования называется максимальное расстояние от листьев до корня.*

Определение 7. *i-ым слоем дерева хэширования называется множество узлов, лежащих на расстоянии $k-i+1$ от корня.*

Заметим, что для формирования узлов дерева хэширования могут использоваться различные функции, в частности, функции сжатия [16], подстановочные преобразования [14] и хэш-функции [9].

Определение 8. *Функцию, используемую для формирования узлов из $\mathcal{I}$, будем называть внутренней функцией.*

В качестве внутренней функции будем рассматривать некоторую стойкую к коллизиям функцию сжатия $h: V_m \to V_t$, $m > t$.

Под вычислительной трудоемкостью в модели древовидного хэширования будем понимать общее число элементарных операций (вычислений значений внутренней функции $h(x)$, где $x \in V_m$), необходимых для выработки хэш-кода.

Будем также полагать, что в единицу времени может одновременно выполняться T элементарных операций. В этом случае под временной трудоемкостью будем понимать общее время, необходимое для выработки хэш-кода с использованием внутренней функции h.

1. Последовательное преобразование слоев

Определение 9. *Процедурой впитывания называется последовательное заполнение слоев дерева хэширования блоками исходного сообщения и, при необходимости, служебной информации.*

Определение 10. *Модель функционирования древовидных режимов работы хэш-функций, включающая в себя процедуры впитывания и построения полного дерева, называется моделью последовательного преобразования слоев.*

Данная модель используется, например, при построении деревьев хэширования для хэш-функций MD6 [16], Skein [13] и BLAKE2 [9].

При дальнейшем описании будем считать выполненным соотношение $\frac{m}{t} = l$, необходимое для построения полного l-арного дерева.

Для произвольного $s \in \mathbb{N}$ определим отображение $g_s : V_{sm} \to V_{st}$:

$$g_s(M) = h\left(M^{(1)}\right) \| h\left(M^{(2)}\right) \| \ldots \| h\left(M^{(s)}\right),$$

где $M = M^{(1)} \| M^{(2)} \| \ldots \| M^{(s)}$, $M^{(i)} \in V_m$, $i \in \overline{1,s}$. Тогда алгоритм вычисления хэш-кода $H(M)$ на основе последовательного преобразования слоев для произвольного сообщения $M \in V^*$, имеет вид:

Алгоритм 1

1. Принудительно дополняем сообщение $M \in V^*$ вектором $(1,0,...,0)$ до длины кратной m и делим его на блоки длины m:

$$M' = M^{(1)} \| M^{(2)} \| \ldots \| M^{(p)},$$

где $p = \lceil \frac{n}{m} \rceil$, где n - длина сообщения M'.

2. Полагаем $i = 1$, $n_1 = n$.

3. Если $\left\lceil \frac{n_1}{m} \right\rceil > T$, то переходим на шаг 5, в противном случае - на шаг 7.

4. Если $\left\lceil \frac{n_1}{m-t} \right\rceil > T$, то переходим на шаг 6, в противном случае - на шаг 7.

Процедура впитывания

5. Вычисляем $M_1 = LSB_{mT}(M)$. *Полагаем* $n_1 = n_1 - mT$, $M = MSB_{n_1}(M)$ *и переходим на шаг 4.*

6. Вычисляем $M_1 = LSB_{(m-t)T}(M) \| g_T(M_1)$. *Полагаем* $n_1 = n_1 - (m-t)T$, $M = MSB_{n_1}(M)$ *и переходим на шаг 4.*

Процедура построения дерева

7. Делим n_i *с остатком на* m: $n_i = jm + q_i$, $0 \le q_i < m$. *Дополняем при необходимости* M_i *до длины кратной* m *вектором специального вида* x *длины* $rm - q_i$ *и вычисляем*

$$M_{i+1} = g_{j+r}(x \| M_i), \quad n_{i+1} = j + r,$$

где r *- фиксировано и определяется конкретной реализацией.*

7. Определяем максимальное* $k \in \mathbb{N}$: $n_i = ml^k + p_i$, $0 \le p_i < m(l^{k+1} - l^k)$. *Вычисляем*

$$M_{i+1} = MSB_{p_i}(M_i) \| g_{l^k}\left(LSB_{n_i - p_i}(M_i)\right), \quad n_{i+1} = p_i + tl^k.$$

8. Если $n_i > m$, *то полагаем* $i = i + 1$, *и переходим на шаг 7 или 7*, в противном случае при необходимости дополняем* M_i *вектором специального вида* x *до длины кратной* m *и вычисляем* $H(M) = h(x \| M_i)$.

Схема одного из вариантов работы алгоритма 1 представлена на рис. 2.

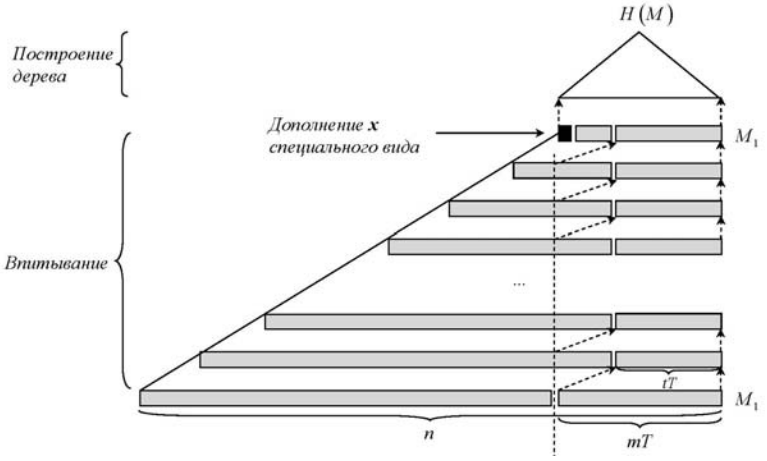

Рис. 2. Схема последовательного преобразования слоев

Замечание 1. *В общем случае в процессе выполнения алгоритма 1 может быть выбран как шаг 7, так и шаг 7*. При этом выбор соответствующего шага, а также величины* $r \in \mathbb{N} \cup \{0\}$ *определяется конкретной реализацией модели хэширования. Так, например, в MD6 используется только шаг 7, при этом* $r = 4m$, *а в Sakura используется только шаг 7*.*

При $n > mT$ алгоритм 1 (см. рис. 2) обладает структурным недостатком, присущим процедуре Меркла-Дамгарда и заключающимся в неравномерном использовании блоков исходного сообщения (первые блоки вносят значительно больший «вклад» в формирование $H(M)$, чем последние), что в свою очередь может быть использовано для построения коллизий.

В процессе выполнения алгоритма 1 длина $ml^k - p_i$ дополнения, позволяющего построить полное l-арное дерево, сокращается. Следовательно, добавление вектора специального вида x в верхних слоях дает возможность наиболее эффективно перейти к построению полного l-арного дерева (см., например, Sakura [11]). В свою очередь дополнение на последнем слое (в корне дерева) в некоторых случаях позволяет определить второй прообраз с трудоемкостью меньшей трудоемкости полного перебора.

Таким образом, определение способов добавления векторов специального вида в слоях дерева хэширования (шаги 7 и 7*) представляет собой одну из основных задач, возникающих при синтезе подобных режимов.

Положим

$$\alpha = n - mT - \left[\frac{n-mT}{(m-t)T}\right](m-t)T + tT,$$
$$\beta = \frac{n-mT}{(m-t)T}.$$

Пусть при этом $\alpha \geq ml^{k'}$, где k' - максимальное число с таким свойством, и пусть имеет место разложение:

$$\alpha = r_0 ml^{k'} + r_1 ml^{k'-1} + \ldots + r_{k'-1} ml + r_{k'} m + r_{k'+1}, \quad (1)$$

где $0 < r_0 < l$, $0 \le r_i < l$, $j = \overline{1,k'}$ и $0 \le r_{k'+1} < m$.

Сформулируем ряд результатов, позволяющих реализовать в процессе выработки хэш-кода с использованием алгоритма 1 однократное дополнение вектора специального вида x минимально возможной длины – эффективное однократное дополнение (далее - ЭОД).

Теорема 1. *Пусть сообщение M длины $n \ge mT$ хэшируется с помощью алгоритма 1, и пусть $T = l^k, k \in \mathbb{N}$. Тогда если $\beta \in \mathbb{N}$, то для реализации ЭОД дополнений не требуется, в противном случае требуется одно дополнение $\left(d + [\beta] + 2\right)$-го слоя дерева, где $d \in \overline{0,k'+1}$ - максимальное число:*

$$\alpha - ml^{k'} \le m \cdot \min\left(l^k - l^{k'} - 1, l^{k'+1-d} - l^{k'-d}\right).$$

Теорема 2. *Пусть сообщение M длины $n < mT$ хэшируется с помощью алгоритма 1, и пусть $T = l^k, k \in \mathbb{N}$. Тогда для реализации ЭОД вектор специального вида требуется добавлять при формировании $(d+1)$-го слоя, где $d \in \overline{0,k'+1}$ - максимальное число: $n \ge m\left(l^k - l^{k'} - \min\left(l^k - l^{k'} - 1, l^{k'+1-d} - l^{k'-d}\right)\right)$.*

Теорема 3. *Пусть сообщение M длины $n \ge mT$ хэшируется с помощью алгоритма 1. Пусть далее $ml^k < mT = ml^{k+1} - p$, $p > 0$, $\beta \notin \mathbb{N}$ и для α выполняется соотношение (1). Тогда для реализации ЭОД вектор специального вида требуется добавлять*

1. при $k' < k$ - в $\left([\beta] + d + 2\right)$-ом слое, где $d \in \overline{0,k+1}$ - максимальное число:
$\alpha \le m\left(l^{k'} + \min\left(T - l^{k'} - 1, l^{k'+1-d} - l^{k'-d}\right)\right)$;

2. при $k' = k$ и

- $p \in \overline{1,m-1}$, $r_i = l - 1$, $i = \overline{0,k}$, $0 \le r_{k+1} < \min\left(t, m - p\right)$, *либо* $p \in \overline{1,m-1}$, $\exists r_i < l - 1$, *- в $\left([\beta] + 3\right)$-ем слое;*
- $p \in \overline{1,m-1}$, $r_i = l - 1$, $i = \overline{0,k}$ *и* $t \le r_{k+1} < m - p$ *- в $\left([\beta] + k + 4\right)$-ом слое;*

- $p \geq m$ - *в* $\left(\left[\beta\right]+d+3\right)$-*ем слое,*

  где $d \in \overline{0,k+1}$ - *максимальное число:* $\frac{\alpha - r_{k+1}}{l} + r_{k+1} \leq m\left(l^{k-1} + l^{k-d-1}\left(l-1\right)\right)$.

Теорема 4. *Пусть сообщение* M *длины* $n < mT$ *хэшируется с помощью алгоритма 1. Пусть далее* $ml^k < mT = ml^{k+1} - p$, $p > 0$, $\beta \notin \mathbb{N}$ *и* $mT - n \geq ml^{k'}$, *где* k' - *максимальное число с таким свойством. Если при этом* $mT - n = r_0 ml^{k'} + r_1 ml^{k'-1} + \ldots + r_{k'-1} ml + r_{k'} m + r_{k'+1}$, *где* $0 < r_0 < l$, $0 \leq r_i < l$, $i = \overline{1,k'}$ *и* $0 \leq r_{k'+1} < m$, *то для реализации ЭОД вектор специального вида требуется добавлять*

1. при $k' < k$ - *в* $\left(d+1\right)$-*ом слое, где* $n \geq m\left(T - l^{k'} - \min\left(T - l^{k'} - 1, l^{k'+1-d} - l^{k'-d}\right)\right)$;

2. при $k' = k$ *и*

- $p \in \overline{1,m-1}$, $r_i = l-1$, $i = \overline{0,k}$, *и* $0 \leq r_{k+1} < \min\left(t, m-p\right)$, *либо* $p \in \overline{1,m-1}$, $\exists r_i < l-1$, - *во 2-ом слое;*
- $p \in \overline{1,m-1}$, $r_i = l-1$, $i = \overline{0,k}$ *и* $t \leq r_{k+1} < m - p$ - *в* $\left(k+3\right)$-*ом слое;*
- $p \geq m$ - *в* $\left(d+2\right)$-*ом слое,*

  где $d \in \overline{0,k}$ - *максимальное число:* $mT - n + \left(l-1\right)r_{k+1} \leq m\left(l^k + l^{k-d}\left(l-1\right)\right)$.

Процедура дополнения слоев дерева хэширования вектором специального вида x для ЭОД существенно зависит от длины исходного сообщения M, что в ряде случаев делает возможным построение атак на хэш-функцию H, основанных на структурных особенностях алгоритма 1. Так, например, при хэшировании сообщений длины $n = ml^k + u < mT$, где $u \in \overline{1,m-t}$, согласно теоремам 1-4 вектор x при реализации ЭОД будет добавлен в корень дерева хэширования, а при длине $n = ml^k + u < mT$, где $u \in \overline{m-t+1, m\left(l-1\right)}$, - в предпоследний слой, соответственно. Таким образом, при использовании специально подобранных сообщений можно добиться того, что на последних итерациях алгоритма 1 будет осуществляться

преобразование векторов, часть бит которых известна. Это, в свою очередь, позволяет применять атаки построения коллизий, использующие предвычисления.

Схема одной из реализаций алгоритма 1 с использованием ЭОД изображена на рис. 3.

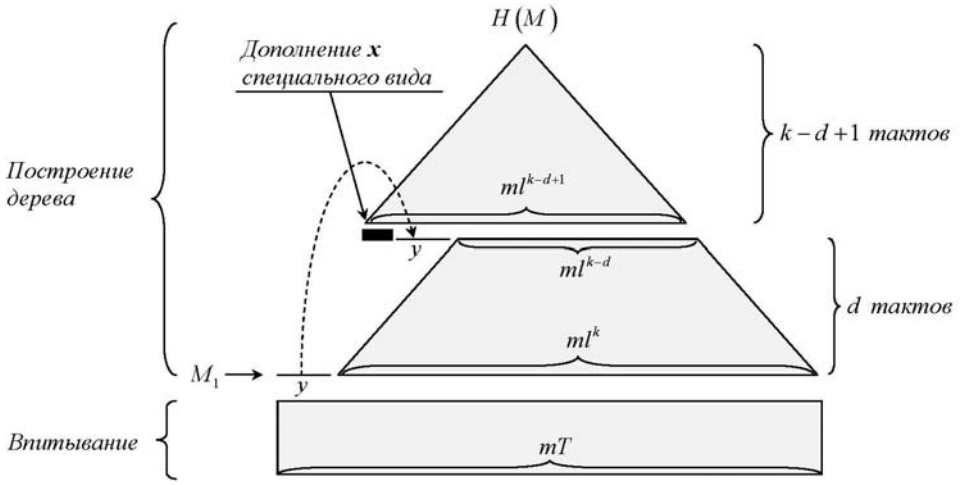

Рис. 3. Схема эффективного однократного дополнения

Через T_1 обозначим вычислительную трудоемкость алгоритма 1 в случае ЭОД. Тогда справедлив следующий результат.

Теорема 5. *Пусть на вход алгоритма 1 поступает сообщение* M *длины* $n \geq mT$, *и* $T = l^k$. *Тогда*

- *если* $\beta \in \mathbb{N}$, *то* $T_1 = (1+\beta)l^k + \frac{l^k - 1}{l-1}$;
- *если* $\beta \notin \mathbb{N}$, *то*
  - $T_1 = (1+[\beta])l^k + \frac{l^{k+1}-1}{l-1}$ *при* $k' = k-1$, $d = 0$;
  - $T_1 = (1+[\beta])l^k + l^{k'-d+1} + \frac{l^{k'+1}-1}{l-1}$ *при* $k' \neq k-1$, $d \in \overline{0, k'+1}$ - *максимальное число:* $\alpha - ml^{k'} \leq m \cdot \min\left(l^k - l^{k'} - 1, l^{k'+1-d} - l^{k'-d}\right)$.

Теорема 6. *Пусть на вход алгоритма 1 поступает сообщение* M *длины* $n < mT$, *и* $T = l^k$. *Пусть при этом* $mT - n \geq ml^{k'}$, *где* k' - *максимальное с таким свойством. Тогда* $T_1 = l^{k'-d+1} + \frac{l^{k'+1}-1}{l-1}$, *где* $d \in \overline{0, k'+1}$ - *максимальное число:* $n \geq m\left(l^k - l^{k'} - \min\left(l^k - l^{k'} - 1, l^{k'+1-d} - l^{k'-d}\right)\right)$.

Замечание 2. *Модель последовательного преобразования слоев позволяет наряду с операцией конкатинации использовать операцию XOR, что, например, реализовано в Phash [14]. Однако такая модификация в некоторых случаях увеличивает вероятность появления внутренних коллизий [6] по сравнению с режимами, использующими конкатенацию узлов.*

2. Параллельное преобразование слоев

Определение 11. *Процедурой усечения дерева хэширования называется последовательное, начиная с листьев, уменьшение числа слоев дерева, участвующих в формировании хэш-кода.*

Определение 12. *Модель функционирования древовидных режимов работы хэш-функций, включающая в себя процедуры впитывания и усечения, называется параллельным преобразованием слоев.*

Основное структурное отличие данной модели от модели последовательного преобразования слоев заключается в использовании уже сформированного дерева хэширования, имеющего дополнительный узел с обратной функциональной связью [17]. Далее при определении множества $\mathcal{P}$, $\mathcal{I}$ будем учитывать указанный узел.

Пронумеруем все узлы дерева хэширования последовательно от 0 до T, начиная с дополнительного узла и корня дерева и заканчивая листьями. Пусть при этом каждый узел дерева с номером $i \in \overline{0,T}$, содержит пару накопителей u_i, z_i с длинами m и t бит, соответственно.

В процессе хэширования блок сообщения размера m бит, поступающий на вход узла с номером i, попадает в накопитель u_i, при этом заполнение накопителя z_i формируется следующим образом:

- если накопитель u_i содержит вектор $x \in V_m$, то в накопитель z_i, записывается значение $y \in V_t$: $y = h(x)$;

- если накопитель u_i содержит пустую строку, то текущее заполнение накопителя z_i остается неизменным.

Алгоритм вычисления хэш-кода $H(M)$ произвольного сообщения $M \in V_n$, $n \in \mathbb{N}$, на основе параллельного преобразования слоев имеет вид:

Алгоритм 2

1. Накопители u_i, z_i, $i \in \overline{0,T}$, *содержат пустые строки. Полагаем* $j = 0$.

2. Если $n \le m$, *то дополняем* M *вектором* x *специального вида до длины* m *и вычисляем* $H(M) = h(x \| M)$, *и алгоритм завершает работу.*

3. Полагаем $M' = M$.

4. Если $m < n \le m\left(\frac{l^{k+1}-1}{l-1}+1\right)$, *то вычисляем* d *такое, что* $\frac{l^d-1}{l-1} < \left\lceil \frac{n-m}{m} \right\rceil \le \frac{l^{d+1}-1}{l-1}$ *и переходим на шаг 9.*

5. Если $n > m\left(\frac{l^{k+1}-1}{l-1}+1\right)$, *то последовательно для* $i \in \overline{0,T}$ *выполняем:*

5.1. $u_i = LSB_m(M')$, $z_i = h(u_i)$,

5.2. $n = n - m$.

5.3. $M' = MSB_n(M')$,

и переходим на шаг 6.

Процедура впитывания

6. Если $n > m - 2t + (m - lt)\frac{l^k-1}{l-1} + ml^k$, *то вычисляем* $u_0 = LSB_{m-2t}(M')$, $z_0 = h(u_0 \| z_0 \| z_1)$, $n = n - m + 2t$, $M' = MSB_n(M')$ *и последовательно для* $i \in \overline{1,T}$ *выполняем:*

6.1. $u_i = LSB_{m-lt}(M')$, $z_i = h\left(u_i \| z_{il+1} \| \ldots \| z_{(i+1)l}\right)$,

6.2. $n = n - m + lt$,

6.3. $M' = MSB_n(M')$,

и переходим на шаг 6.

7. *Если* $n \le m - 2t$, *то дополняем* M' *вектором* x *специального вида до длины* $m - 2t$ *и полагаем* $H(M) = h(x \| M' \| z_0 \| z_1)$, *и алгоритм завершает работу.*

8. *Если* $n \le m - 2t + (m - lt)\frac{l^k - 1}{l - 1}$, *то вычисляем* d *такое, что* $\frac{l^d - 1}{l - 1} < \left\lceil \frac{n - m + 2t}{m - lt} \right\rceil \le \frac{l^{d+1} - 1}{l - 1}$ *и переходим на шаг 9, в противном случае полагаем* $d = k$ *и переходим на шаг 9.*

Процедура усечения дерева

9. *Если* $j < d$, *то вычисляем* $z_0 = h(u_0 \| z_0 \| z_1)$ *и последовательно для* $i \ge 1$ *вычисляем* $z_i = h\left(u_i \| z_{il+1} \| \ldots \| z_{(i+1)l}\right)$, *полагаем* $j = j + 1$ *и переходим на шаг 9, в противном случае полагаем* $H(M) = z_0$, *и алгоритм завершает работу.*

На рис. 4 и 5 приведена схема одной из реализации алгоритма 2.

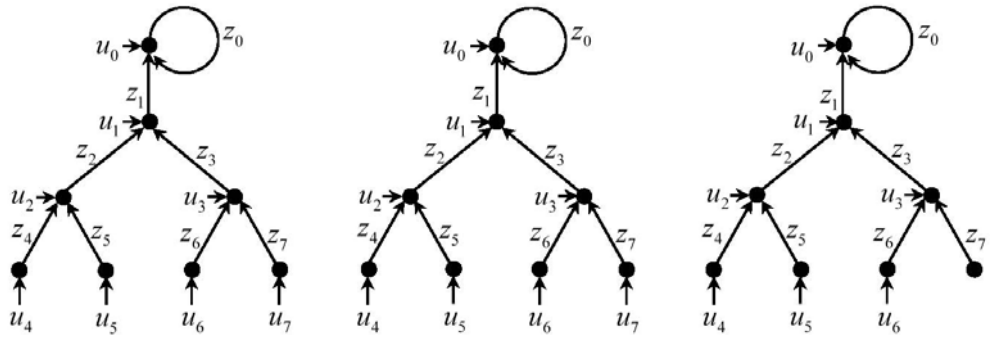

Рис. 4. Процедура впитывания

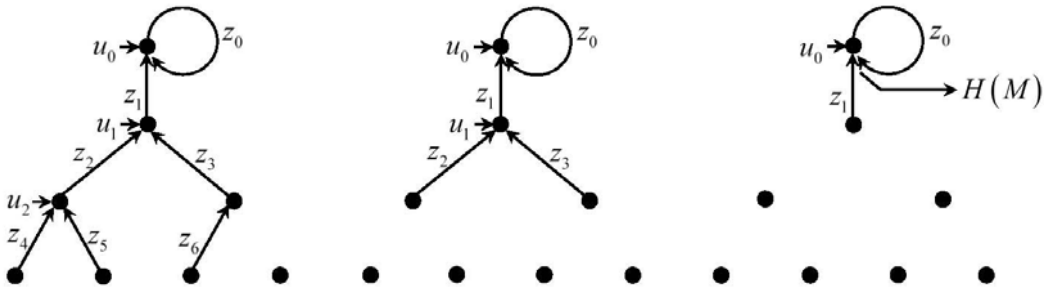

Рис. 5. Процедура усечения

Функциональной особенностью алгоритма 2 является необходимость принудительного завершения работы (параметр d определяет число шагов алгоритма после окончания процедуры впитывания). Действительно, при

отсутсвии ограничения на число щагов алгоритма с $(d+1)$-го шага процедуры усечения будет вырабатываться вырожденная последовательность вида $H(M),\gamma,\ldots,\gamma$, где $\gamma \in V_t$ - константа, сформированная в процессе выработки хэш-кода.

Замечание 3. *Для определения длины вектора специального вида x, используемого в алгоритме 2, требуются предвычисления.*

Через T_2 обозначим вычислительную трудоемкость алгоритма 2. Тогда справедлив следующий результат.

Теорема 7. *Пусть на вход алгоритма 2 поступает сообщение M длины $n > m\left(\frac{l^{k+1}-1}{l-1}+1\right)$ и пусть $\frac{n}{j} \le m-2t+\frac{l^k-1}{l-1}(m-tl)+ml^k$, где $j \in \mathbb{N}$ - минимальное с таким свойством. Тогда*

- *если* $\frac{n+ml^k}{j} > m\left(l^k+1\right)-2t+\frac{l^k-1}{l-1}(m-tl)$, *то*

$$T_2 = (j-1)\left(1+\frac{l^{k+1}-1}{l-1}\right)+l\left(l^{k+1}-1\right)+\frac{l(k+1)}{l-1},$$

- *если* $\frac{n+ml^k}{j} \le m\left(l^k+1\right)-2t+\frac{l^k-1}{l-1}(m-tl)$, *то*

$$T_2 = (j-1)\left(1+\frac{l^{k+1}-1}{l-1}\right)+l\left(l^{k+1}-1\right)+\frac{lk+1}{l-1}.$$

Теорема 8. *Пусть на вход алгоритма 2 поступает сообщение M длины $n \le m\left(\frac{l^{k+1}-1}{l-1}+1\right)$. Если при этом $n \le m$, то $T_2 = 1$. Если же $n > m$, то $T_2 = l^{d+1}-l+\frac{dl+l-1}{l-1}$, где d: $\frac{l^d-1}{l-1} < \left\lceil \frac{n-m}{m} \right\rceil \le \frac{l^{d+1}-1}{l-1}$.*

Теорема 9. *Значение трудоемкости алгоритма 1 в случае ЭОД не превосходит значения трудоемкости алгоритма 2.*

Доказательство. Если $n \le mT$, то трудоемкости процедур впитывания в алгоритмах 1 и 2 совпадают, в противном случае, начиная со второй итерации процедуры впитывания, обрабатываются различные объемы данных: $(m-t)T$ - за одну итерацию в алгоритме 1 и $(m-2t)+(m-tl)(T-1)$ - в алгоритме 2. В этом

случае при $l \geq 1$ трудоемкость процедуры впитывания в алгоритме 1 меньше, чем в алгоритме 2.

В свою очередь трудоемкость процедуры построения полного l-арного дерева в алгоритме 1 не превосходит трудоемкости процедуры усечения дерева в алгоритме 2. □

Говоря об эффективности предложенных моделей хэширования, следует также отметить, что в общем случае они позволяют уменьшить временную трудоемкость вычисления хэш-кода до порядка $\log_l \lceil \frac{n}{m} \rceil$ ед. времени по сравнению с упомянутой выше итерационной процедурой Меркла-Дамгарда, для которой временная трудоемкость имеет порядок $\lceil \frac{n}{m} \rceil$ ед. времени. При этом вычислительная трудоемкость моделей древовидного хэширования увеличивается не существенно – не более, чем в l раз.

Заключение

Предложенные модели функционирования древовидных режимов работы хэш-функций позволяют классифицировать ряд действующих алгоритмов параллельного хэширования, например, [9, 11, 13, 16, 17], а также описывать их криптографические свойства.

Список литературы

1. *Колчин В.Ф.* Случайные отображения. - М.: Наука, 1984.
2. *Колчин В.Ф.* Случайные графы (2-е изд.). - М.: ФИЗМАТЛИТ, 2004.
3. *Колчин В.Ф., Севастьянов Б.А., Чистяков В.П.* Случайные размещения. - М.: Наука, 1976.
4. *Сачков В.Н.* Вероятностные методы в комбинаторном анализе. - М.: Наука, 1978.
5. *Сачков В.Н.* Введение в комбинаторные методы дискретной математики. - М.: Наука. Гл. ред. физ.-мат. лит., 1982.

6. *Дали Ф.А., Миронкин В.О. О.* вероятностных характеристиках одного класса моделей древовидного хэширования. - Обозрение прикладной и промышленной математики. - М.: ОПиПМ, 23:4 (2016), с. 345-347.
7. *Дали Ф.А., Миронкин В.О.* О некоторых моделях древовидного хэширования. - Обозрение прикладной и промышленной математики. - М.: ОПиПМ, 24:4 (2017), с. 330-334.
8. *Дали Ф.А., Миронкин В.О.* Обзор подходов к построению древовидных режимов работы некоторых хэш-функций. – Проблемы информационной безопасности. Компьютерные системы. - СПб.: СПбПУ, 2017, № 2, с. 46-55.
9. *Aumasson J.-P., Neves S., Wilcox-O'Hearn Z., Winnerlein C.* BLAKE2: simpler, smaller, fast as MD5, 2013, https://blake2.net.
10. *Bertoni G., Daemen J., Peeters M., Van Assche G.* Sufficient conditions for sound tree hashing modes, Symmetric Cryptography (Dagstuhl, Germany) (Handschuh H., Lucks S., Preneel B., Rogaway P., eds.), Dagstuhl Seminar Proceedings, no. 09031, Schloss Dagstuhl Leibniz-Zentrum fuer Informatik, Germany, 2009.
11. *Bertoni G., Daemen J., Peeters M., Van Assche G.* Sakura: a flexible coding for tree hashing, Cryptology ePrint Archive, Report 2013/231, 2013, http://eprint.iacr.org.
12. *Dodis Y., Reyzin L., Rivest R., Shen E.* Indifferentiability of permutation-based compression functions and tree-based modes of operation with applications to MD6. Fast So ware Encryption (O. Dunkelman, ed.), Lecture Notes in Computer Science, vol. 5665, Springer, 2009, pp. 104-121.
13. *Ferguson N., Lucks S., Schneier B., Whiting D., Bellare M., Kohno T., Callas J.* The Skein Hash Function Family, Version 1.3., October, 2010.
14. *Kaminsky A., P. Radziszowski S.* A case for a parallelizable hash Department of Computer Science Rochester Institute of Technology, Rochester, NY, USA MILCOM, November, 2008.
15. *Merkle R.C.* Secrecy, authentication, and public key systems, PhD thesis, UMI Research Press, 1982.
16. *Rivest R.L., Agre B., Bailey D.V., Crutchfield C., Dodis Y., Fleming K.E., Khan A., Krishnamurthy J., Lin Y., Reyzin L., Shen E., Sukha J., Sutherland D., Tromer E., Yin Y.L.* The MD6 hash function. A proposal to NIST for SHA-3, http://csrc.nist.gov - Massachusetts Institute of Technology, Cambridge, MA, October, 2008.
17. *Sarkar P., Schellenberg P.J.* A parallelizable design principle for cryptographic hash functions, Cryptology ePrint Archive, Report 2002/031, 2002, http://eprint.iacr.org.