О подходах к синтезу режимов древовидного хэширования

Иван Лавриков Григорий Маршалко Василий Шишкин

18.03.2015

Древовидное хэширование

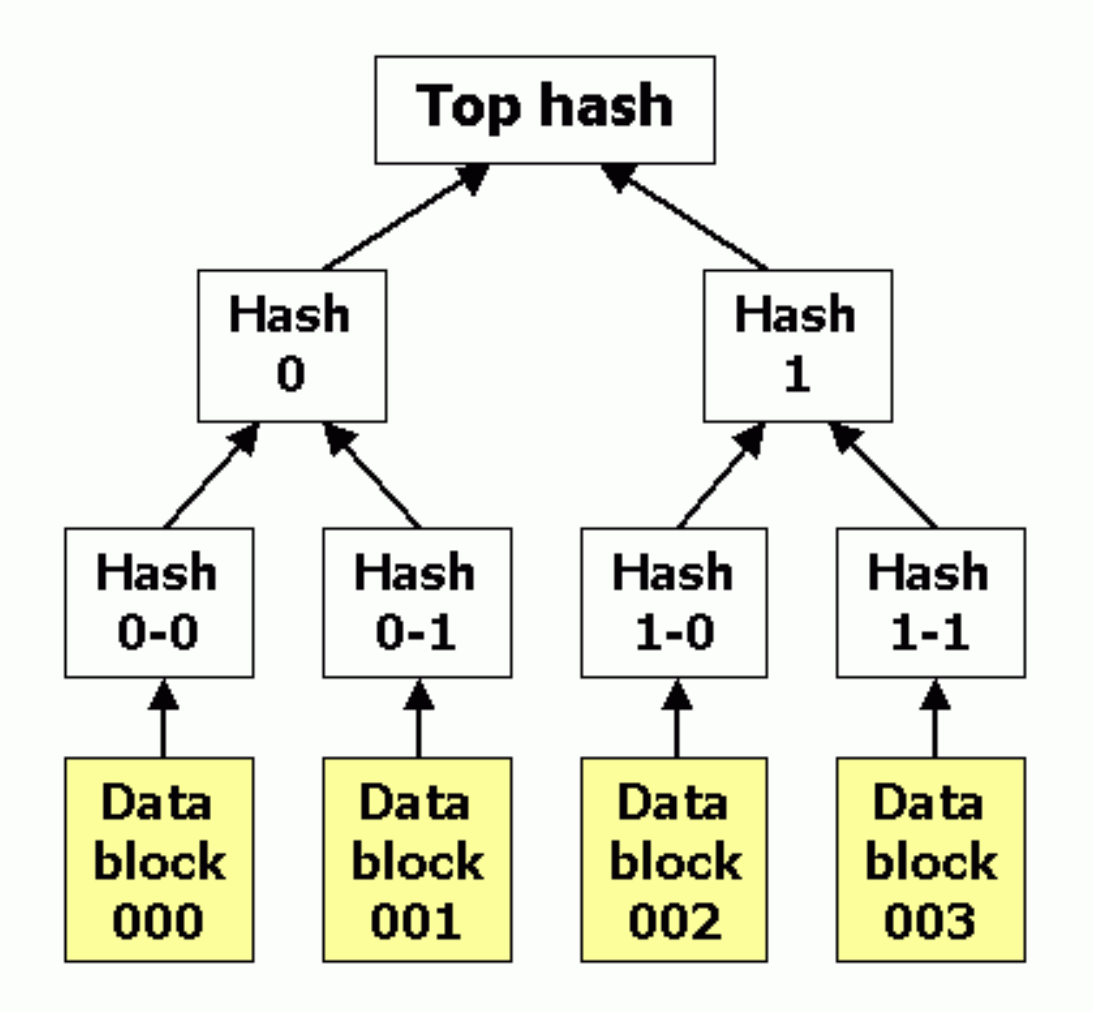
- Проверка целостности данных при хранении и передаче данных

Древовидное хэширование

- Проверка целостности данных при хранении и передаче данных
- Позволяет быстро вычислять хэш-код при изменении части данных

Древовидное хэширование

- Проверка целостности данных при хранении и передаче данных
- Позволяет быстро вычислять хэш-код при изменении части данных
- Предложен Ральфом Мерклем в 1982 г.


Top hash
Hash 0
Hash 1
Hash 0-0
Hash 0-1
Hash 1-0
Hash 1-1
Data block 000
Data block 001
Data block 002
Data block 003

Базовые требования к криптографическим функциям

- стойкость к коллизиям;
- стойкость к построению прообраза;
- стойкость к построению второго прообраза.

Первое обоснование стойкости – И. Дамгорд (1989 г.)

Пусть $\mathcal{F}$ – семейство устойчивых к коллизиям функций со входом и выходом фиксированной длины, отображающих строки длины m в строки длины $t(m)$. Тогда существует семейство устойчивых к коллизиям функций $\mathcal{H}$, отображающих строки произвольный длины в строки длины $t(m)$, со следующими свойствами:
Пусть h некоторый представитель семейства H с длиной входа равной m. Тогда значение хэш-кода от сообщения длины n может быть вычислено за $O(log_2(n/t(m))t(m)/(m-t(m))$ операций вычисления (функций из множества $\mathcal{F}$) с использованием $m/2t(m)$ процессоров.

"Классический"вариант

Пусть h некоторый представитель семейства H с длиной входа равной m. Тогда значение хэш-кода от сообщения длины n с помощью одного процессора может быть вычислено за $n/(m-t(m)+1)+1$ операций вычисления функций из множества $\mathcal{F}$.

- Недостаток – число процессоров пропорционально длине сообщения
- Альтернативный вариант (Саркар и Шелленберг, 2002) – способ позволяющий использовать произвольное число процессоров

Саркар и Шелленберг, 2002

На основе стойкой функции хэширования $f : V_m \to V_t$, $m \geq 2t$ и двоичного дерева, состоящего из k процессоров можно построить стойкие функции хэш-функции h^*, которые хэшируют сообщения длины меньшей 2^{m-t} и h^∞ хэширующие сообщения произвольной длины. Число операций, необходимых для вычисления длины n, при этом составляет $\lfloor \frac{n}{2^k(m-t)} \rfloor + t$.

Додис, Рейзин, Ривест, Шен, 2009

С позиции теории доказуемой стойкости (а именно доказательства неотличимости конкретной конструкции от случайного отображения) был сформулированы свойства, при которых конкретная конструкция древовидного хэширования будет неотличима от случайного отображения.

Додис, Рейзин, Ривест, Шен, 2009

- **Уникальный способ синтаксического анализа.**
  - *Предикат родителя.*
  - *Предикат листа.*
- **Предикат корня.**
- **Линейная структура алгоритма обработки сообщения.**
- **Обработка финального выходного значения.**
- **Восстановление исходного сообщения.**

Додис, Рейзин, Ривест, Шен, 2009

- **Уникальный способ синтаксического анализа.** Каждый аргумент $x \in V_m$ функции сжатия, который возникает в процессе хэширования режимом h некоторого сообщения $X \in V^*$ должен быть эффективно и единственным образом представлен в виде последовательности блоков (возможно различной длины), каждый из которых является выходом функции сжатия, частью исходного сообщения или вспомогательными данными.
  - *Предикат родителя.*
  - *Предикат листа.*
- **Предикат корня.**
- **Линейная структура алгоритма обработки сообщения.**
- **Обработка финального выходного значения.**
- **Восстановление исходного сообщения.**

Додис, Рейзин, Ривест, Шен, 2009

- **Уникальный способ синтаксического анализа.**
  - *Предикат родителя.* Для фиксированного способа синтаксического анализа определим предикат $parent(x, y, i)$, аргументами которого являются $x, y \in V_n$ и натуральное число i. Он принимает истинное значение, если $f(y)$ равно выходу i-функции сжатия при обработке входа x. В этом случае мы будем говорить, что x является родителем y, а y в свою очередь является потомком x.
  - *Предикат листа.*
- **Предикат корня.**
- **Линейная структура алгоритма обработки сообщения.**
- **Обработка финального выходного значения.**
- **Восстановление исходного сообщения.**

Додис, Рейзин, Ривест, Шен, 2009

- **Уникальный способ синтаксического анализа.**
  - *Предикат родителя.*
  - *Предикат листа.* Для фиксированного способа синтаксического анализа определим предикат $leaf(x)$, аргументом которого является входной вектор функции сжатия, и который принимает истинное значение только в том случае, если x содержит только биты хэшируемого сообщения и вспомогательные данные и не содержит выходных значений функции хэширования. В этом случае мы будем говорить, что x является листом.
- **Предикат корня.**
- **Линейная структура алгоритма обработки сообщения.**
- **Обработка финального выходного значения.**
- **Восстановление исходного сообщения.**

Додис, Рейзин, Ривест, Шен, 2009

- **Уникальный способ синтаксического анализа.**
  - *Предикат родителя.*
  - *Предикат листа.*
- **Предикат корня.** Должен существовать эффективно вычисляемый предикат $root(x)$, который принимает истинное значение при финальном вызове функции сжатия, и ложное – в остальных случаях.
- **Линейная структура алгоритма обработки сообщения.**
- **Обработка финального выходного значения.**
- **Восстановление исходного сообщения.**

Додис, Рейзин, Ривест, Шен, 2009

- **Уникальный способ синтаксического анализа.**
  - *Предикат родителя.*
  - *Предикат листа.*
- **Предикат корня.**
- **Линейная структура алгоритма обработки сообщения.** Алгоритм вычисления хэш-кода должен быть линеен. Он выполняет последовательность вызов функции сжатия так, что на i-том вызове аргументом функции является само сообщение, вспомогательные данные и выходы от j предыдущих вызовов $i_1, i_2, ..., i_j < i$. Такая конструкция должна эффективно вычисляться для любого входного сообщения вне зависимости от функции сжатия f. За исключением последнего, каждое значение хэш-кода должно использоваться для вычисления аргумента какой-либо другой функции сжатия. $\Sigma(M)$ – последовательность вызовов функции сжатия.
- **Обработка финального выходного значения.**
- **Восстановление исходного сообщения.**

Додис, Рейзин, Ривест, Шен, 2009

- **Уникальный способ синтаксического анализа.**
  - *Предикат родителя.*
  - *Предикат листа.*
- **Предикат корня.**
- **Линейная структура алгоритма обработки сообщения.**
- **Обработка финального выходного значения.** Должно выполняться равенство $h(M) = \zeta(f(x))$, где x – аргумент функции сжатия при финальном вызове, а ζ – эффективно вычислимая функция, такая что из множество ее прообразов для данного h должно эффективно осуществлять случайную выборку.
- **Восстановление исходного сообщения.**

Додис, Рейзин, Ривест, Шен, 2009

- **Уникальный способ синтаксического анализа.**
  - *Предикат родителя.*
  - *Предикат листа.*
- **Предикат корня.**
- **Линейная структура алгоритма обработки сообщения.**
- **Обработка финального выходного значения.**
- **Восстановление исходного сообщения.** Должна существовать эффективно вычислимая функция ρ, которая по последовательности Π вызовов функции сжатия восстанавливает исходное сообщение M, если $\Phi = \Sigma(M)$, и выдает пустое множество в противном случае.

Если $f : V_m \to V_t$ – случайное отображение, тогда конструкция древовидного хэширования $h : V^* \to V_d$, использующая отображение f (l, q_F, q_S, ϵ)-неотличима от случайного отображения $F : V^* \to V_d$, где $\epsilon = (2m/t + 1)q_l^2/2^c$, $l = O(q_l^3 + q_l\kappa(q_l))$, для любых значений q_F – числа обращений к функции сжатия и q_S – числа обращений различителя к функции сжатия, которые в сумме не превосходят q_l.

Бертони, Даемен, Питерс, Ван Аше, 2009

Для более широкого класса конструкций, в частности, допускается использование в качестве функций сжатия отображений с произвольной длиной входа и выхода, что позволяет использовать в качестве функций сжатия другие режимы хэширования.

- Режим должен быть декодируемым, т.е. не должно существовать дерева, которое одновременно соответствует дереву, описывающему режим в целом или его собственному поддереву, содержащему конечную вершину;
- Режим должен быть полным, т.е. по исходное сообщение должно однозначно восстанавливаться из данных, содержащихся в узлах соответствующего ему дерева.
- Режим должен обеспечивать отличие конечной вершины, от любой другой вершины дерева, с тем, чтобы обеспечить невозможность применения атак, основанных на расширении исходного сообщения.