Информационные системы электронной коммерции

Доцент Филиппов Ф.В.
E-mail: 9000096@mail.ru

Блокчейн

http://jrxv.net/x/16/chain.pdf

Блокчейн

Блокчейн (block chain) — *выстроенная по определённым правилам непрерывная последовательная цепочка блоков* (связный список), содержащих информацию.

Чаще всего копии цепочек блоков хранятся на множестве разных компьютеров независимо друг от друга.

https://marmelab.com/blog/2016/04/28/blockchain-for-web-developers-the-theory.html
https://habr.com/post/323128/

Блокчейн – это книга с фактами, реплицируемая на несколько компьютеров, объединенных в сеть равноправных узлов (P2P). Фактами может быть что угодно, от денежных операций и до подписания контента. Члены сети — анонимные лица, называемые узлами. Все коммуникации внутри сети используют криптографию, чтобы надежно идентифицировать отправителя и получателя. Когда узел хочет добавить факт в журнал, в сети формируется консенсус, чтобы определить, где этот факт должен появиться в журнале; этот консенсус называется **блоком**.

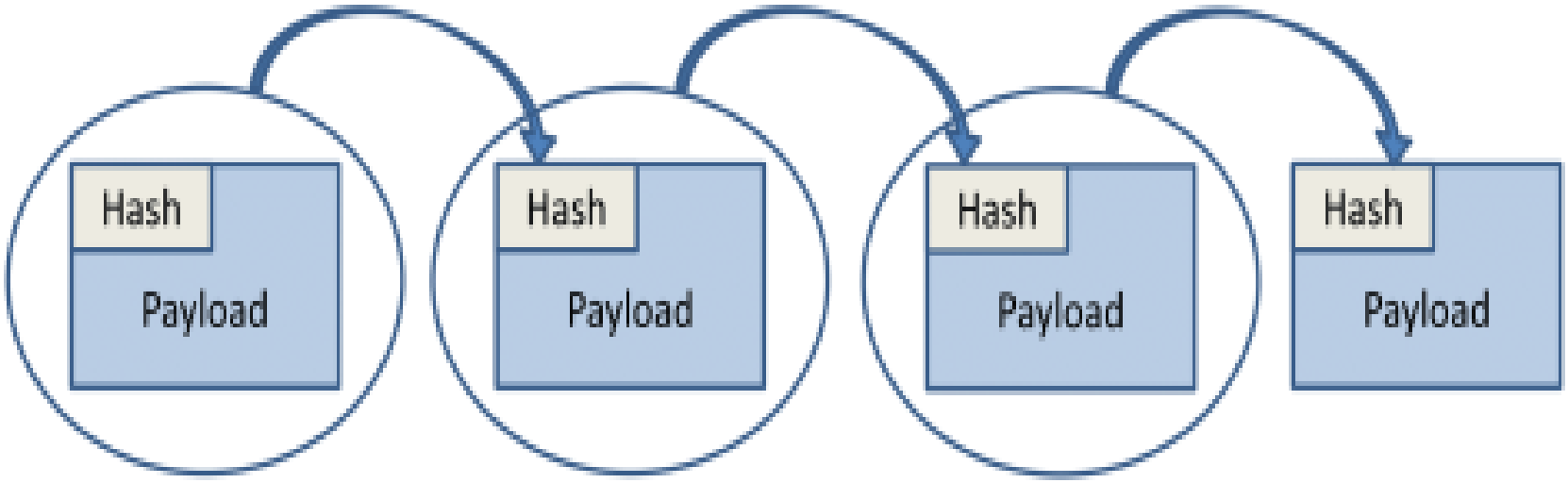
Хэш-функции

Хэш-функцией называется алгоритм, конвертирующий строку произвольной длины (сообщение) в битовую строку фиксированной длины, называемой хэш-кодом (хэшем, проверочной суммой, дайджестом или цифровым отпечатком).

При изменении одного символа в исходной строке существенно изменяется хэш.

Хэшчейн

Хэшчейн обладает важным свойством: данные в блоке не могут быть изменены без нарушения целостности цепочки. Например если пэйлоуд первого блока изменить, тогда нужно изменить хеш второго блока, а значит и третьего, и так далее.



Хэш-функции

Для очень большого количества технологий безопасности (например, аутентификации) применяются односторонние функции шифрования, называемые также **хэш-функциями**.

Основное назначение подобных функций – получение из сообщения произвольного размера его дайджеста – значения **фиксированного размера**. Дайджест может быть использован в качестве контрольной суммы исходного сообщения, обеспечивая таким образом (при использовании соответствующего протокола) контроль целостности информации.

Основные свойства хэш-функции:

- на вход хэш-функции подается сообщение произвольной длины;
- на выходе хэш-функции формируется блок данных фиксированной длины;
- значения на выходе хэш-функции распределены по равномерному закону;
- при изменении одного бита на входе хэш-функции существенно изменяется выход.

http://www.volpi.ru/umkd/zki/index.php?man=1&page=20

Требования к хэш-функциям

Для обеспечения устойчивости хэш-функции к атакам она должна удовлетворять следующим требованиям:

- если мы знаем значение хэш-функции h, то задача нахождения сообщения M такого, что H(M) = h, должна быть вычислительно трудной;
- при заданном сообщении M задача нахождения другого сообщения M’, такого, что H(M) = H(M’), должна быть вычислительно трудной.

Если хэш-функция будет удовлетворять перечисленным свойствам, то формируемое ею значение будет уникально идентифицировать сообщения, и всякая попытка изменения сообщения при передаче будет обнаружена путем выполнения хэширования на принимающей стороне и сравнением с дайджестом, полученным на передающей стороне.

Еще одной особенностью хэш-функций является то, что они не допускают обратного преобразования – получить исходное сообщения по его дайджесту невозможно. Поэтому их называют еще односторонними функциями шифрования.

Популярные хэш-функции (алгоритмы сжатия)

- **CRC32** — используется для создания контрольных сумм (так называемое избыточное кодирование). Данная функция не является криптографической. Есть много вариаций этого алгоритма (число после CRC означает длину получаемого хеша в битах), в зависимости от нужной длины получаемого хеша. Функция очень простая и нересурсоемкая. В связи с этим используется для проверки целостности пакетов в различных протоколах передачи данных.
- **MD5** — старая, но до сих пор очень популярная версия уже криптографического алгоритма, которая создает хеш длиной в 128 бит. Хотя стойкость этой версии на сегодняшний день и не очень высока, она все равно часто используется как еще один вариант контрольной суммы, например, при скачивании файлов из сети.
- **SHA-1** — криптографическая функция формирующая хеш-суммы длиной в 160 байт. Сейчас идет активная миграция в сторону SHA-2, которая обладает более высокой устойчивостью, но SHA-1 по-прежнему активно используется в качестве контрольных сумм и для хранения хешей паролей в базе данных сайта.
- **ГОСТ Р 34.11-2012** — текущий российский криптографический (стойкий к взлому) алгоритм введенный в работу в 2013 году (ранее использовался ГОСТ Р 34.11-94). Длина выходного хеша может быть 256 или 512 бит. Обладает высокой криптостойкостью и довольно хорошей скоростью работы. Используется для электронных цифровых подписей в системе государственного и другого документооборота.

http://implbits.com/products/hashtab/

Реализация хэш-функций

Хэш-функции строятся по итеративной схеме, когда исходное сообщение разбивается на блоки определенного размера, и над ними выполняются ряд преобразований с использованием как обратимых, так и необратимых операций.

Как правило, в состав хэширующего преобразования включается сжимающая функция, поскольку его выход зачастую по размеру меньше блока, подаваемого на вход.

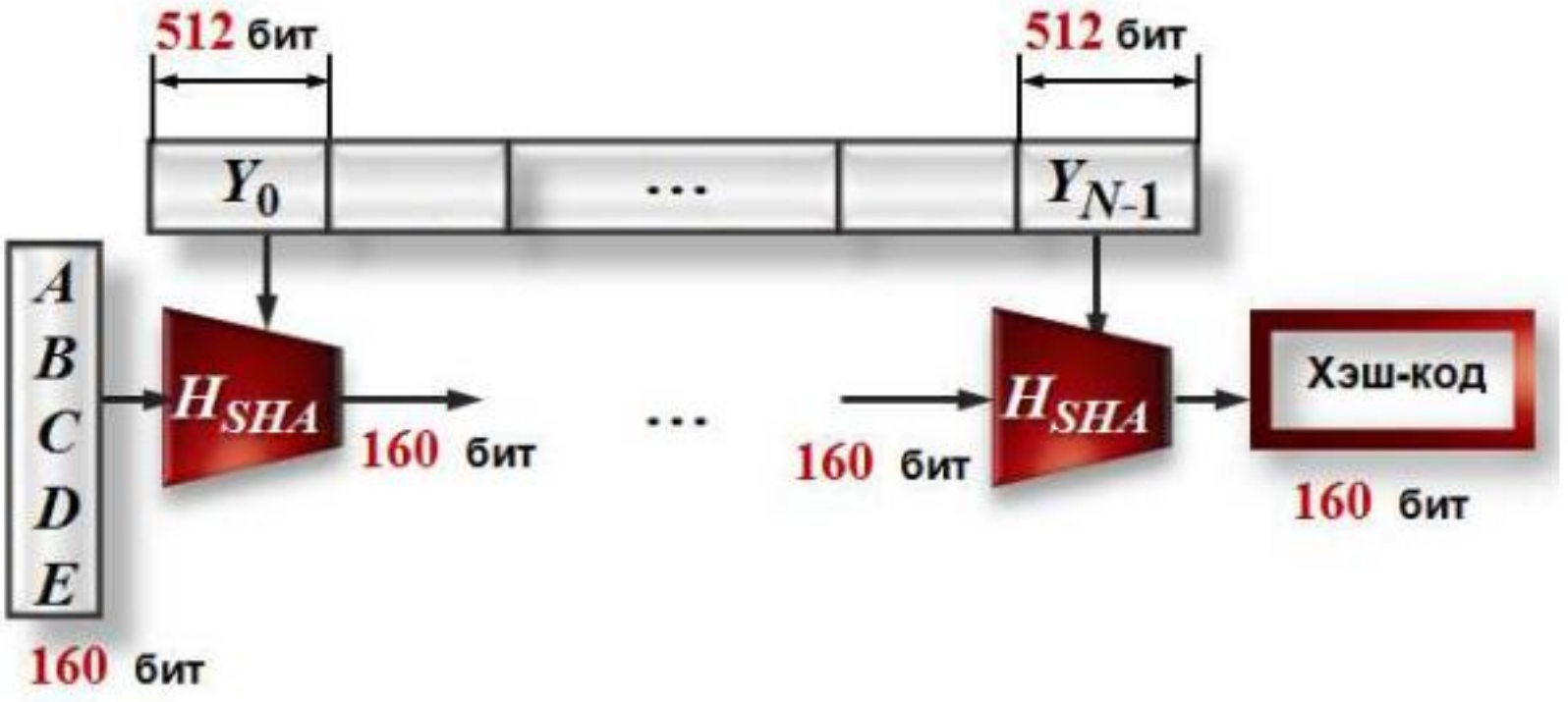
На вход каждого цикла хэширования подается выход предыдущего цикла, а также очередной блок сообщения.

Таким образом, на каждом цикле выход хэш-функции h_i представляет собой хэш первых i блоков.

SHA-1

Основные характеристики алгоритма:

1. Длина хэш-кода – 160 бит
2. Длина обрабатываемых блоков – 512 бит
3. Число шагов алгоритма - 80 (4 раунда по 20 шагов)
4. Максимальная длина хэшируемых данных – $2^{64}–1$.



http://bit.nmu.org.ua/ua/student/metod/cryptology/%D0%BB%D0%B5%D0%BA%D1%86%D0%B8%D1%8F17.pdf

Добавление недостающих битов и указание длины (шаг 1)

Исходное m-битовое сообщение (a_1, a_2 ,..., a_m) расширяется до длины, кратной 512, по принципу:

a_1, a_2 ,..., a_m, 1, 0, 0, ..., t_1,t_2, ..., t_{64},

где t_1,t_2, ..., t_{64} – двоичная запись числа m (кода длины сообщения)

Инициализация буфера (шаг 2)

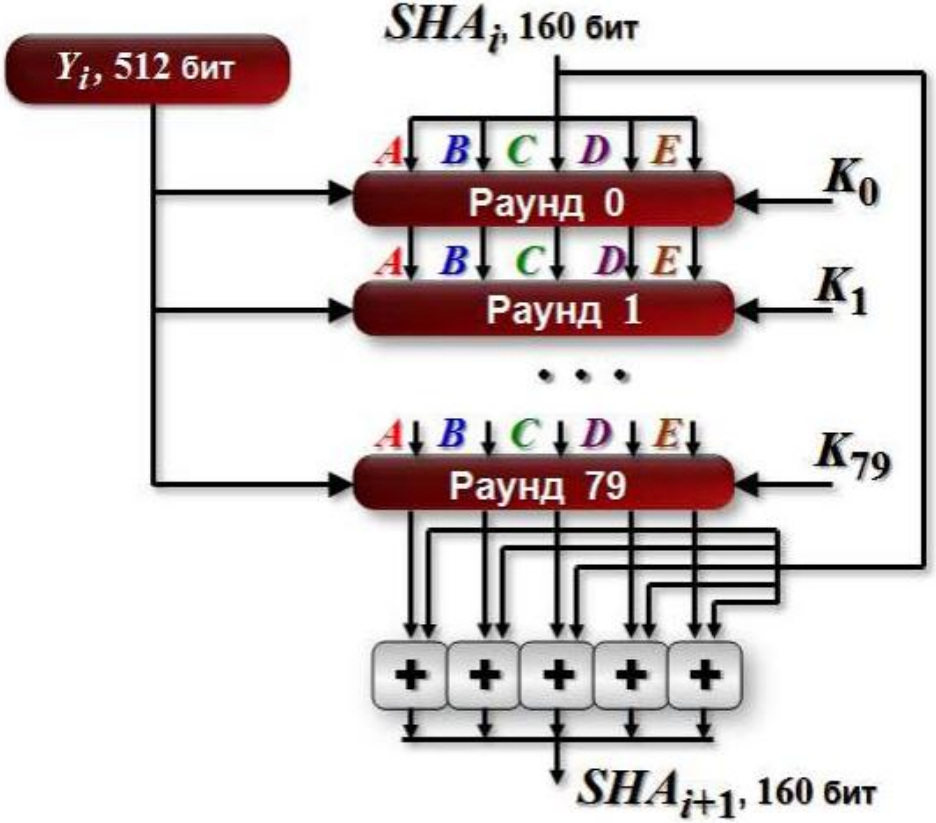
В алгоритме используется 160-битный буфер для хранения промежуточных и окончательных результатов хэш-функции. Буфер может быть представлен как пять 32-битных регистров A, B, C, D, E. Эти регистры инициализируются следующими шестнадцатеричными числами:

A = 67 45 23 01;
B = EF CD AB 89;
C = 98 BA DC FE ;
D =10 32 54 76 ;
E = C3 D2 E1 F0.

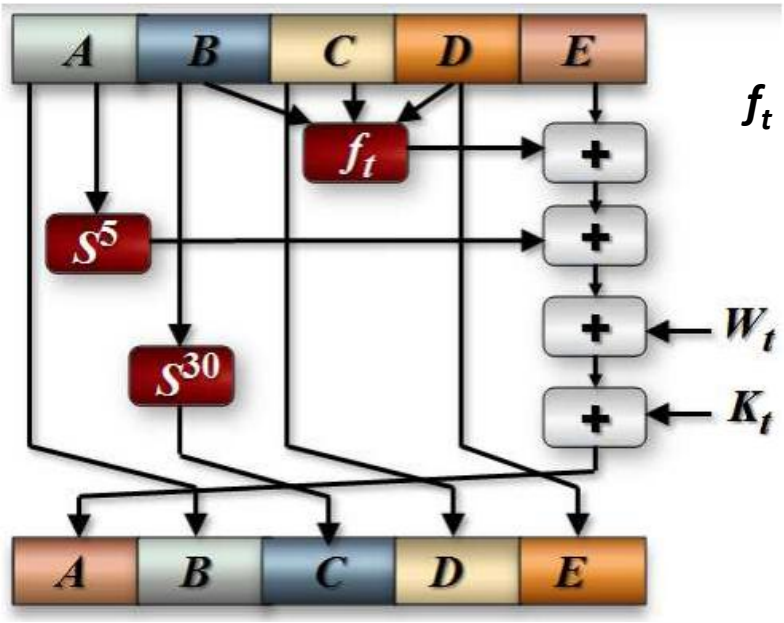
Обработка сообщения в 512-битных блоках (шаг 3)

Основой алгоритма является модуль, состоящий из 80 циклических обработок, имеющих одинаковую структуру.

Каждый цикл получает на входе текущий 512-битный обрабатываемый блок Y_i и 160-битное значение буфера ABCDE , и изменяет содержимое этого буфера.



Что делается на каждом раунде



$f_t(B, C, D) = (B \wedge C) \vee ((\neg B) ? D)$	для $0 \le t \le 19$
$f_t(B, C, D) = B \oplus C \oplus D$	для $20 \le t \le 39$
$f_t(B, C, D) = (B \wedge C) \vee (B \wedge D) \vee (C \wedge D)$	для $40 \le t \le 59$
$f_t(B, C, D) = B \oplus C \oplus D$	для $60 \le t \le 79$

$$5A827999, \quad 0 \le t \le 19;$$
$$6ED9EBA1, \quad 20 \le t \le 39,$$
$$8F1BBCDC, \quad 40 \le t \le 59;$$
$$CA62C1D6, \quad 60 \le t \le 79.$$

t – номер раунда;

$S^5(A)$ – циклический левый сдвиг 32-битового аргумента A на 5 бит;

$S^{30}(B)$ – циклический левый сдвиг 32-битового аргумента B на 30 бит;

+ означает сложение по модулю 2^{32} ;

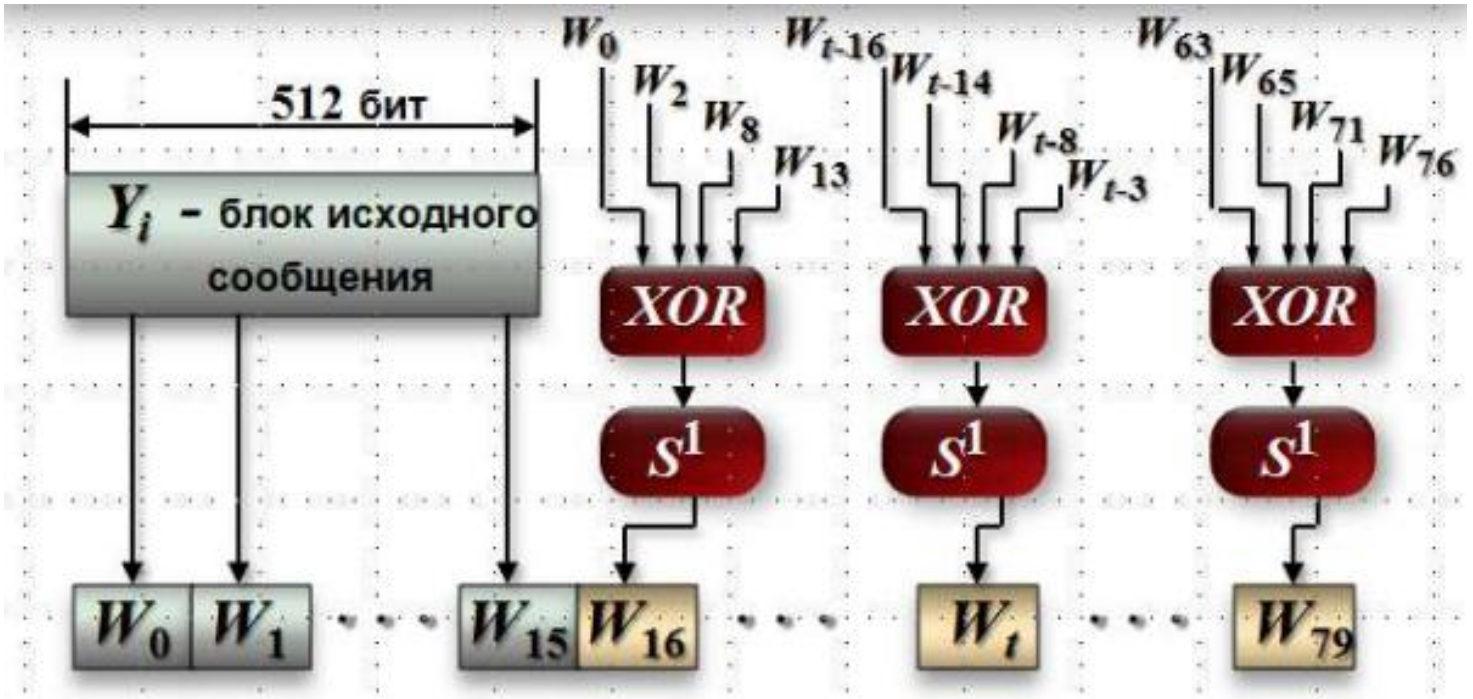
f_t – элементарная логическая функция (описана ниже);

K_t – пошаговые константы, равные

W_t – 32-битные слова, получаемые из очередного 512-битного блока сообщения:

- делим блок на группы из 16 слов $W_0, W_1, \ldots, W_{15}$ (W_0 самое левое слово).
- для t = 16 - 79 $W_t = S^1(W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16})$

Как формируются слова $W_0 – W_{79}$



Выход (шаг 4)

После обработки всех 512-битных блоков выходом является 160-битный дайджест сообщения.

Алгоритм SHA-1

$$SHA_0 = ABCDE_0$$
$$SHA_{i+1} = \sum 32(SHA_i, ABCDE_i)$$
$$SHA = SHA_{N-1}$$

где

- $ABCDE_0$ - начальное значение буфера ABCDE ;
- $ABCDE_i$ - результат обработки i-того блока сообщения;
- N - число блоков в сообщении, включая поля добавления и длины;
- $\sum 32$ - сумма по модулю 2^{32}, выполняемая отдельно для каждого слова буфера;
- SHA - значение дайджеста сообщения.

Упражнения

1. Каково дополнение для SHA-512, если длина сообщения:

    a) 5120 битов
    b) 5121 бит
    c) 6143 бита

2. Найдите результат функции f_{47} (B, C, D), если

    B = 1234 5678 ABCD 2345 34564 5678 ABCD 2468
    C = 2234 5678 ABCD 2345 34564 5678 ABCD 2468
    D = 3234 5678 ABCD 2345 34564 5678 ABCD 2468

3. Найдите результат функции f_{13} (B, C, D), если

    B = 1234 5678 ABCD 2345 34564 5678 ABCD 2468
    C = 2234 5678 ABCD 2345 34564 5678 ABCD 2468
    D = 3234 5678 ABCD 2345 34564 5678 ABCD 2468

Технология *Hasq*

Понятие права собственности предполагает наличие контроля над объектом собственности. Это утверждение верно как для реальных объектов, так и для абстрактных понятий, таких как банковский счет, копирайт или цифровые данные.

В действительности, возможность контроля ограничена фундаментальным свойством большинства вещей, которыми можно владеть - собственность может быть утрачена против воли ее владельца.

Однако, существует понятие, принципиально отличающееся от вышеописанных - авторское право. Авторское право не может быть отнято или передано другому лицу.

Свойства права собственности	Реальные объекты, деньги (наличные)	Деньги (банковский счет)	Копирайт	Авторское право	Токен Hasq
Может быть утеряно против желания	Да	Да	Да	Нет	Нет
Может быть передано по желанию	Да	Да	Да	Нет	Да

Объект **Токен Hasq** воплощает идею максимальной защиты от незаконного отъема.

Технология *Hasq*

Объект владения (**токен**) — это уникальный цифровой объект, хранящийся в реестре. Токен может быть модифицирован, либо передан новому владельцу только своим текущим владельцем.

Токен однозначно связан с реальным или виртуальным имуществом. Владение токеном определяет право владельца на распоряжение этим имуществом.

Распределенная сетевая база данных (**реестр**) — это база данных, части или копии которой расположены на различных серверах, связанных между собой. Каждый сервер в сети поддерживает свою копию базы данных, которая используется при выполнении команд пользователя.

Совокупность копий баз данных всех серверов образует распределенную сетевую **базу данных Hasq**.

Хэш-цепочки технологии *Hasq*

N S K {G1 G2 … } O [D]

N порядковый номер записи

S хэш от цифровых данных произвольного характера

K key
G generator
O owner
} хэши

D текстовое поле данных

Поля *K* (Key, Ключ), *G* (Generator, Генератор) и *O* (Owner, Владелец) используются для связывания записей. Количество полей *G* может быть произвольным, но фиксированным в пределах одной базы данных.

База данных состоит из записей, включающих различные токены *S*. Если отобрать записи, содержащие одно значение *S*, то список этих записей будет выглядеть следующим образом (поле *D* опущено для простоты, так как оно не участвует в связывании записей):

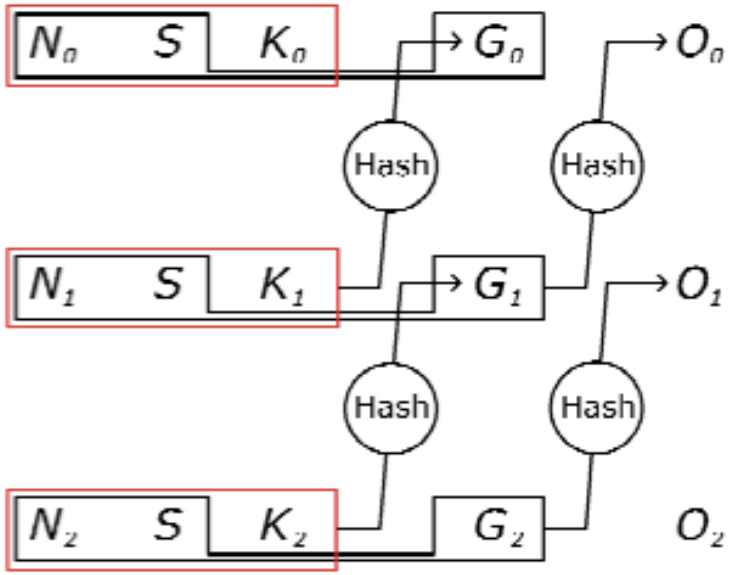
…
$N_0\ S\ K_0\ G_0\ O_0$
$N_1\ S\ K_1\ G_1\ O_1$
$N_2\ S\ K_2\ G_2\ O_2$
$N_3\ S\ K_3\ G_3\ O_3$
$N_4\ S\ K_4\ G_4\ O_4$
…

Правила связывания записей *Hasq*

G_0 = Hash(N_1, S, K_1) O_0 = Hash(N_1, S, G_1)

G_1 = Hash(N_2, S, K_2) O_1 = Hash(N_2, S, G_2)

...
N_0 S K_0 G_0 O_0
N_1 S K_1 G_1 O_1
N_2 S K_2 G_2 O_2
N_3 S K_3 G_3 O_3
N_4 S K_4 G_4 O_4
...



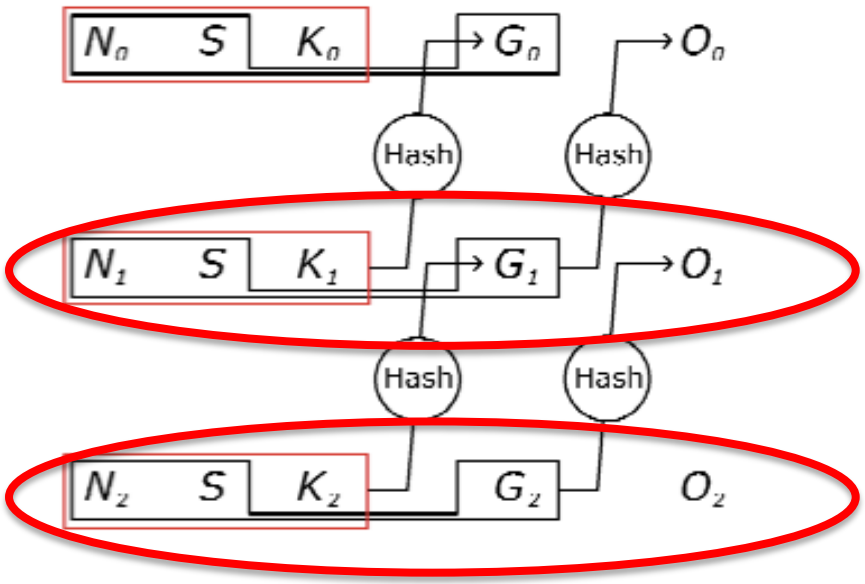
Право собственности на токен *S* с точки зрения пользователя реализуется следующим образом: пусть последняя запись для *S* имеет вид: N_0 S K_0 G_0 O_0. Пользователь владеет токеном *S* когда он **знает (секретные) ключи *K*1 и *K*2**.

Ключи, в свою очередь, могут быть сгенерированы из личного *пароля* пользователя по некоторому алгоритму. Например, простой метод генерации ключа может быть таким:

K_i = Hash(*i*, *S* , *пароль*)

Передача права владения токеном *Hasq*

- **Получатель генерирует** K_3, G_2 и O_1, затем отправляет O_1 текущему владельцу токена S (назовем его отправитель).
- **Отправитель публикует** запись N_1 S K_1 G_1 O_1, открывая свой первый секретный ключ K_1. После публикации получатель может убедиться, что O_1 появилось в базе данных.
- **В этот момент времени ни отправитель, ни получатель не владеют токеном *S***, поскольку K_2 известен только отправителю, а K_3 известен только получателю.
- **Получатель генерирует** O_2 (предварительно сгенерировав K_4 и G_3) и посылает G_2 и O_2 отправителю.
- **Отправитель публикует** запись N_2 S K_2 G_2 O_2.
- **Получатель становится новым владельцем токена *S***, поскольку только он знает новые секретные ключи K_3 и K_4.



Применение технологии *Hasq*

Hasq Technology Pty Ltd разработала высокопроизводительную распределенную систему, которая реализует описанную выше схему и решила многие технические вопросы, сопровождавшие разработку соответствующего программного продукта.

Копии базы данных Hasq расположены на серверах сети Hasq, общающихся между собой по простому сетевому протоколу. Ядро протокола составляет набор команд для получения имеющихся и добавления новых записей. Для работы системы не требуется доверие серверов друг другу, каждый из серверов работает независимо. При публикации новых записей сервера отправляют уведомления другим серверам. В силу независимости работы отдельных серверов полная синхронизация копий базы данных не требуется.

Схема связи записей Hasq обеспечивает следующие **свойства токенов**. Токен может:

- ❑ быть передан через Интернет или по телефону
- ❑ быть передан совершенно анонимно для отправителя или для получателя
- ❑ служить банкнотой, облигацией, проверяемым правом собственности
- ❑ выступать в роли посредника, дающего доступ к другим электронным данным

Если организация, выпускающая токены, определяет группу неанонимных пользователей, тогда часть токенов может использоваться для учета долга одних пользователей перед другими.