1. Атака дней рождений

Следует отметить, что не доказано существование необратимых хэш-функций, для которых вычисление какого-либо прообраза заданного значения хэш-функции теоретически невозможно. Обычно нахождение обратного значения является лишь вычислительно сложной задачей.

**Атака «дней рождения»** позволяет находить коллизии для хэш функции с длиной значений n бит в среднем за вычислений хэшфункции. Поэтому n-битная хэш-функция считается криптостойкой, если вычислительная сложность нахождения коллизий для нее близка к за .

Для криптографических хэш-функций важно также, чтобы при малейшем изменении аргумента значение функции сильно изменялось (это свойство называется лавинным эффектом). В частности, значение хэша не должно давать утечки информации даже об отдельных битах аргумента. Это требование служит залогом криптостойкости алгоритмов, хэширующих пользовательский пароль для получения ключа

1. Сравнение семейства SHA c RIPEMD
2. RIPEMD – 160 является улучшенной версией RIPEMD, которая в свою очередь использовала принципы MD4 и по производительности сравнима с более популярной SHA–1.
3. **Насколько лучше RIPEMD, чем SHA1? (если он лучше), в 5 раз медленнее вычислять, но размер хэша совпадает с SHA1.**
4. **Теоретическая атака 2 ^ 60 бит возможна на SHA-1 , что означает, что алгоритм слабее, чем задумано.**
5. **Оба SHA1 и RIPEMD уязвимы для атак на день рождения**
6. RIPEMD–160 разработана в открытом академическом сообществе, в отличие от SHA–1 и SHA–2, которые были созданы NSA. Использование RIPEMD–160 не ограничено какими-либо патентами.
7. Алгоритмы семейства SHA используются в SSL, SSH, S/MIME, DNSSEC, X.509, PGP, IPSec, при передаче файлов по сети (BitTorrent). RIPEMD-128 нашла своё применение в некоторых банковских приложениях.
8. У семейства SHA c RIPEMD выполняют шаги 1 и 2 алгоритма (добавление дополнительных бит и добавление исходной длины сообщения в основе лежит алгоритм MD5) аналогичным образом. Шаг 3 (определение констант и используемых функций) также уникальны для каждой версии алгоритмов (единственное, что генерируются 16-ти ричные сообщения). Результатом будет закодированная последовательность 16-ти ричных блоков.
9. Использование H(x) в H-MAC

**H-MAC** – расшифровывается как код аутентификации сообщений, использующий хеш-функции с ключом.

В криптографии один из механизмов проверки целостности информации, позволяющий гарантировать то, что данные, передаваемые или хранящиеся в ненадёжной среде, не были изменены посторонними лицами. **MAC** — стандарт, описывающий способ обмена данными и способ проверки целостности передаваемых данных с использованием секретного ключа. Два клиента, использующие MAC, как правило, используют общий секретный ключ. **HMAC** — надстройка над MAC; механизм обмена данными с использованием секретного ключа (как в MAC) и хеш-функций. В названии может уточняться используемая хеш-функция: HMAC-MD5, HMAC-SHA1, HMAC-RIPEMD128, HMAC-RIPEMD160 и т. п.

Преимущества HMAC:

* возможность использования хеш-функций, уже имеющихся в программном продукте;
* отсутствие необходимости внесения изменений в реализации существующих хеш-функций (внесение изменений может привести к ухудшению производительности и ухудшению криптостойкости);
* возможность замены хеш-функции в случае появления более безопасной или более быстрой хеш-функции.

скорость работы хеш-функций (например, MD5, SHA-1, RIPEMD128, RIPEMD-160) обычно выше скорости работы симметричных блочных шифров (например, DES). Возникло желание использовать хеш-функции в MAC, а наличие готовых библиотек с реализациями различных хеш-функций только подтолкнуло эту идею.