Количество обратимых типа $\exists\forall$ функций равно

$$|D|^{|D_1|\cdot|D_2|} - C_{\text{необр}};$$

в частности, для $|D| = |D_1| = |D_2| = m$ получаем $m^{m^2} - (m^m - m!)^m$.

ЛИТЕРАТУРА

1. *Agibalov G. P.* Cryptanalytical finite automaton invertibility with finite delay // Прикладная дискретная математика. 2019. № 46. С. 27–37.
2. *Agibalov G. P.* Problems in theory of cryptanalytical invertibility of finite automata // Прикладная дискретная математика. 2020. № 50. С. 62–71.
3. *Agibalov G. P.* Cryptanalytic concept of finite automaton invertibility with finite delay // Прикладная дискретная математика. 2019. № 44. С. 34–42.

УДК 004.056 DOI 10.17223/2226308X/14/15

ЭКСПЕРИМЕНТАЛЬНОЕ ИССЛЕДОВАНИЕ ХАРАКТЕРИСТИК ОДНОГО СПОСОБА КОНТРОЛЯ ЦЕЛОСТНОСТИ ПРИ ХРАНЕНИИ ДАННЫХ БОЛЬШОГО ОБЪЁМА

Д. А. Бобровский, Д. И. Задорожный, А. М. Коренева, Т. Р. Набиев, В. М. Фомичёв

Описан способ встраивания высокопроизводительного алгоритма генерации кода контроля целостности, представленного авторами на РусКрипто'2020, в функцию хэширования, определенную в ГОСТ 34.11-2018 (256 бит). Полученные ранее результаты существенно улучшены. Проведены экспериментальные исследования производительности и криптографических свойств нового алгоритма. Установлено, что предложенный алгоритм производительнее известных криптографических функций хэширования, близок по производительности к CRC32, а по криптографическим свойствам значительно его превосходит.

Ключевые слова: *аддитивные генераторы, контроль целостности, матрично-графовый подход, перемешивающие свойства, регистры сдвига, AG-S, AG-S-Стрибог, SMHasher.*

Введение

Обеспечение целостности хранимых данных относится к основным задачам защиты информации. В настоящее время применяются различные алгоритмы и подходы: хэш-функции (MD, SHA, ГОСТ 34.11-2018 и др.), хэш-функции с ключом (HMAC), блочные шифры в режиме выработки имитовставки (CMAC). При контроле целостности (КЦ) применяются также некриптографические методы с использованием кодов, обнаруживающих и/или исправляющих ошибки (коды Хэмминга, циклические коды (CRC) и др.).

При динамическом контроле больших объёмов данных, КЦ исполняющей среды функционирования, а также при проведении оперативного аудита целевых систем возникает проблема вычислений с высокой ресурсоёмкостью. Непосредственное использование для решения данной проблемы известных подходов и алгоритмов затруднительно в силу имеющихся у них недостатков: высокой ресурсоёмкости, слабых криптографических характеристик и пр. В работе предложено альтернативное решение на основе комбинации высокопроизводительного алгоритма (например, CRC32) и хэш-функции, соответствующей современным требованиям к криптографической стойкости (например, ГОСТ 34.11-2018). Удачное решение подразумевает компромисс между скоростью

и криптографическими свойствами алгоритмов, исключающими применение вычислительно простых методов построения коллизий (примеры имеющих подобную слабость высокопроизводительных алгоритмов известны [1]).

Исследования при построении нового алгоритма были направлены на решение следующих задач:

— построение высокопроизводительных алгоритмов генерации кодов контроля целостности (ККЦ) и обоснование их свойств;
— обоснование выбора параметров для реализации конкретного алгоритма;
— определение корректной и экономной процедуры дополнения блоков данных до подходящих размеров;
— сложность поиска коллизий, а также оценка вероятности коллизий для случайных входов;
— оценка вычислительной сложности построенных алгоритмов генерации ККЦ;
— разработка способа встраивания высокопроизводительного алгоритма генерации ККЦ в функцию хэширования;
— оценка производительности и криптографических свойств построенного алгоритма.

В работе описан новый алгоритм контроля целостности хранимых данных с использованием хэширования. При реализации алгоритма массив входных данных сначала разбивается на фрагменты, для которых с помощью высокопроизводительного алгоритма генерируются уникальные ККЦ. Конкатенация полученных ККЦ образует вход криптографической хэш-функции, выход хэш-функции есть ККЦ исходных данных. Проведены экспериментальные исследования производительности и криптографических свойств этого алгоритма.

1. Описание комбинированного алгоритма

Схема комбинированного алгоритма КЦ дана на рис. 1. Массив входных данных M произвольного размера разбивается на блоки размера 1 кбайт (8192 бита). Если длина массива M не кратна размеру блока, то последний блок дополняется до 1 кбайт. Проанализированы различные алгоритмы дополнения и выбрана схема с учётом характеристик дополняемого блока.

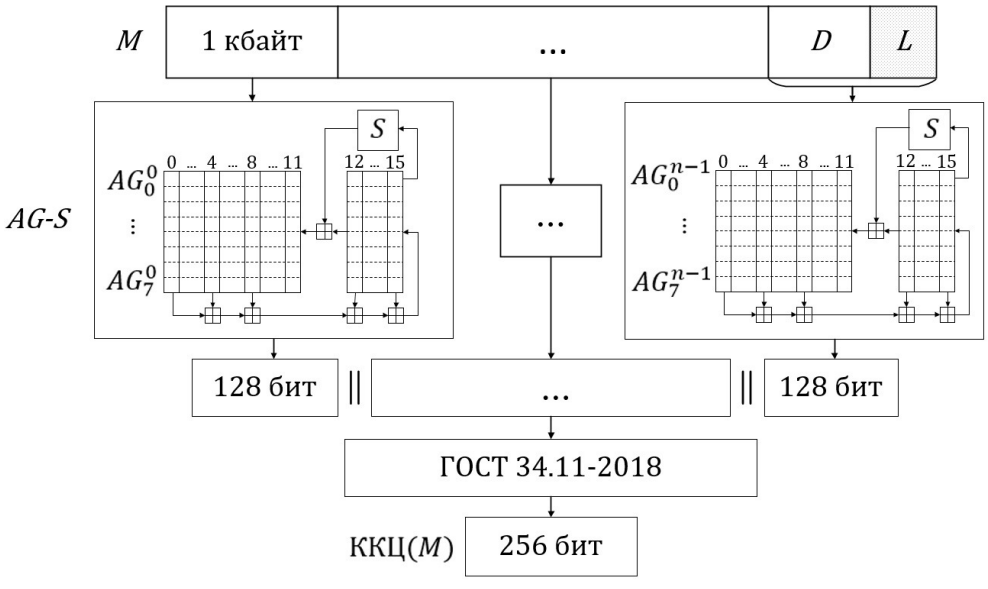

Рис. 1. Схема комбинированного алгоритма *AG-S*-Стрибог

Для генерации уникального ККЦ каждого блока данных размера 1 кбайт использована схема *AG-S* на основе восьми аддитивных генераторов $AG_0, \ldots, AG_7$ и *S*-бокса из ГОСТ 34.11-2018. Аддитивные генераторы (*AG*) реализуют преобразования регистра сдвига длины 16 над множеством V_{64} с функцией обратной связи, определённой в [2, 3]. Восемь тактов работы схемы обеспечивают перемешивание данных и формируют 128-битовый код.

Вход хэш-функции по ГОСТ 34.11-2018 есть конкатенация полученных ККЦ для всех блоков данных, а 256-битовое хэш-значение есть ККЦ массива данных M в целом. Алгоритм обозначен *AG-S*-Стрибог.

2. Теоретические исследования класса алгоритмов генерации ККЦ

Для класса преобразований на основе АГ и *S*-бокса (для предложенной схемы это преобразование $V_{8192} \to V_{8192}$) доказана биективность преобразования множества состояний $AG_0, \ldots, AG_7$. Это позволило более точно оценить вероятность совпадения ККЦ для двух различных случайных блоков. Оценена сложность поиска блоков с одинаковыми ККЦ (2^{64} опробований входов для схемы из п. 1).

Получены характеристики перемешивающего орграфа, определяющие обоснованный выбор параметров для реализации конкретных преобразований. В частности, с использованием оценки локального экспонента перемешивающего графа преобразования оценена приемлемая для производительности и криптографических свойств глубина итерации преобразования. Оценена теоретическая вычислительная сложность предложенных алгоритмов.

3. Экспериментальные исследования

Измерена производительность генерации контрольных кодов разными алгоритмами при входных блоках 1 кбайт (табл. 1). Эксперименты проведены на ПЭВМ с процессором Intel Core i5-8600 3.1 GHz без использования процессорных инструкций SHANI, SSE4.2.

Таблица 1

Характеристики производительности

Алгоритм	CRC32	SHA2-256	SHA3-256	MD5	CMAC-Magma	Стрибог	*AG-S*
CpB, такты/байт	8,664	26,435	48,855	18,405	61,306	42,55	9,516

Проведено сравнение алгоритмов CRC32, MD5, SHA2-256, SHA3-256, Стрибог и *AG-S* с помощью набора тестов SMHasher [4] для проверки по известной методике [5] реализаций хэш-функций. В табл. 2 приведены результаты следующих тестов: Sanity — на идентичность реализации хэш-функции и её математической модели; Avalanche — на выполнение строгого лавинного критерия; Chi2 — на равномерность распределения хэш-значений с помощью статистики χ^2; Differential — на поиск коллизий в множестве входов длины 64 и 128 бит, расстояние Хэмминга между которыми не более 5 и 4 соответственно; Collisions — на поиск коллизий и сравнение их количества с ожидаемым. Символами «+» и «−» в табл. 2 обозначено соответственно пройден тест или нет. Для тестов Differential и Collisions указана доля пройденных тестов из их общего числа.

Таблица 2

Результаты применения набора тестов SMHasher

Тест	CRC32	MD5	SHA2-256	SHA3-256	Стрибог	*AG-S*-Стрибог
Sanity	+	+	+	+	+	+
Avalanche	−	+	+	+	+	+
Chi2	−	+	+	+	+	+
Differential	1/2	2/2	2/2	2/2	2/2	2/2
Collisions	6/41	36/41	41/41	41/41	41/41	25/41

Выводы

1. Алгоритм *AG-S* от 1,93 до 6,44 раз превышает по производительности известные функции хэширования (бесключевые и ключевые) и близок по производительности к CRC32.

2. По результатам тестов Avalanche, Chi2 и Differential алгоритм *AG-S*-Стрибог не уступает известным функциям хэширования и значительно превосходит алгоритм CRC32. Для усиления свойств алгоритма *AG-S* следует продолжить исследование коллизий.

ЛИТЕРАТУРА

1. *Stigge M., Plotz H., Muller W., and Redlich J.-P.* Reversing CRC — Theory and Practice. HU Berlin Public Report, 2006.
2. *Фомичев В. М., Коренева А. М., Набиев Т. Р.* О новом алгоритме контроля целостности данных // Конференция Рускрипто'20, Московская область, 2020. `https://www.ruscrypto.ru/resource/archive/rc2020/files/02_koreneva_fomichev.pdf`
3. *Фомичев В. М., Коренева А. М., Набиев Т. Р.* Характеристики алгоритма контроля целостности данных на основе аддитивных генераторов и s-боксов // Прикладная дискретная математика. Приложение. 2020. № 13. С. 62–66.
4. `https://github.com/rurban/smhasher`.
5. Хэш-функция t1ha. `ttps://github.com/PositiveTechnologies/t1ha`.

УДК 004.056 DOI 10.17223/2226308X/14/16

ОБ АЛГОРИТМЕ ДОПОЛНЕНИЯ БЛОКОВ БОЛЬШОГО РАЗМЕРА В СИСТЕМАХ КОНТРОЛЯ ЦЕЛОСТНОСТИ

Д. А. Бобровский, Т. Р. Набиев, В. М. Фомичёв

В алгоритмах контроля целостности при расчёте контрольной суммы файла требуется, чтобы его длина была кратна заданной величине (l бит). При защите файла произвольной длины, как правило, выполняется его дополнение до требуемой длины. Представлена вычислительно простая и эффективная схема дополнения, предназначенная для систем контроля целостности, обрабатывающих большие блоки (порядка 1 кбайт). Схема построена на основе выходов линейного конгруэнтного генератора. Начальное состояние генератора формируется с помощью данных дополняемого блока и исходной длины файла. Результаты анализа криптографических свойств алгоритма контроля целостности и экспериментов по оценке производительности показали преимущества предложенной схемы по сравнению с известными стандартными схемами дополнения.

Ключевые слова: *алгоритм дополнения, широкий блок, линейный конгруэнт-*