Ежегодная международная научно-практическая конференция
«РусКрипто’2021»

# О контроле целостности хранимых данных с использованием хэширования

**Дмитрий Бобровский,**
**Дмитрий Задорожный,**
**Алиса Коренева,**
**Тимур Набиев,**
**Владимир Фомичёв**

КОД безопасности

# Актуальность

Проблема вычислений с высокой ресурсоемкостью:

- Динамический контроль больших объемов данных.
- Контроль целостности (КЦ) исполняющей среды функционирования.
- Оперативный аудит целевых систем.

Используемые подходы для КЦ:

Вычислительно сложные

- хэш-функции (SHA, MD, ГОСТ34.11-2018);
- хэш-функции с ключом (HMAC);
- блочные шифры в режиме выработки имитовставки (CMAC).

Высокопроизводительные

- некриптографические методы с использованием кодов, обнаруживающих и/или исправляющих ошибки (коды Хэмминга, циклические коды (CRC [1]) и др.).

**Актуальная задача** – построение альтернативных алгоритмов КЦ с компромиссом между криптографическими характеристиками и скоростью.

[1] Stigge, Martin, H. Plötz, W. Müller and Jens-Peter Redlich. “Reversing CRC – Theory and Practice.” (2006).

# Научный фундамент

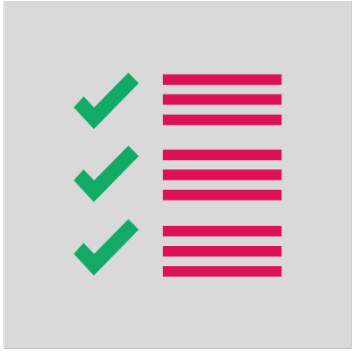
- Коренева А.М. Оценки экспонентов перемешивающих орграфов регистровых преобразований, используемых в системах защиты информации: дис. канд. физ.-мат. наук. МГУ имени М.В. Ломоносова, Москва, 2018. Научный руководитель – д.ф.-м.н., профессор Фомичёв В. М.
- В. М. Фомичев, А. М. Коренева, Т. Р. Набиев, «О новом алгоритме контроля целостности данных», РусКрипто’20.
- В. М. Фомичев, А. М. Коренева, Т. Р. Набиев, “Характеристики алгоритма контроля целостности данных на основе аддитивных генераторов и s-боксов”, ПДМ. Приложение, 2020, № 13, с.62–66.

# Цель

- Разработка способа встраивания высокопроизводительного алгоритма генерации кода контроля целостности[1], представленного на РусКрипто'2020, в функцию хэширования, определенную в ГОСТ 34.11–2018 (256 бит).

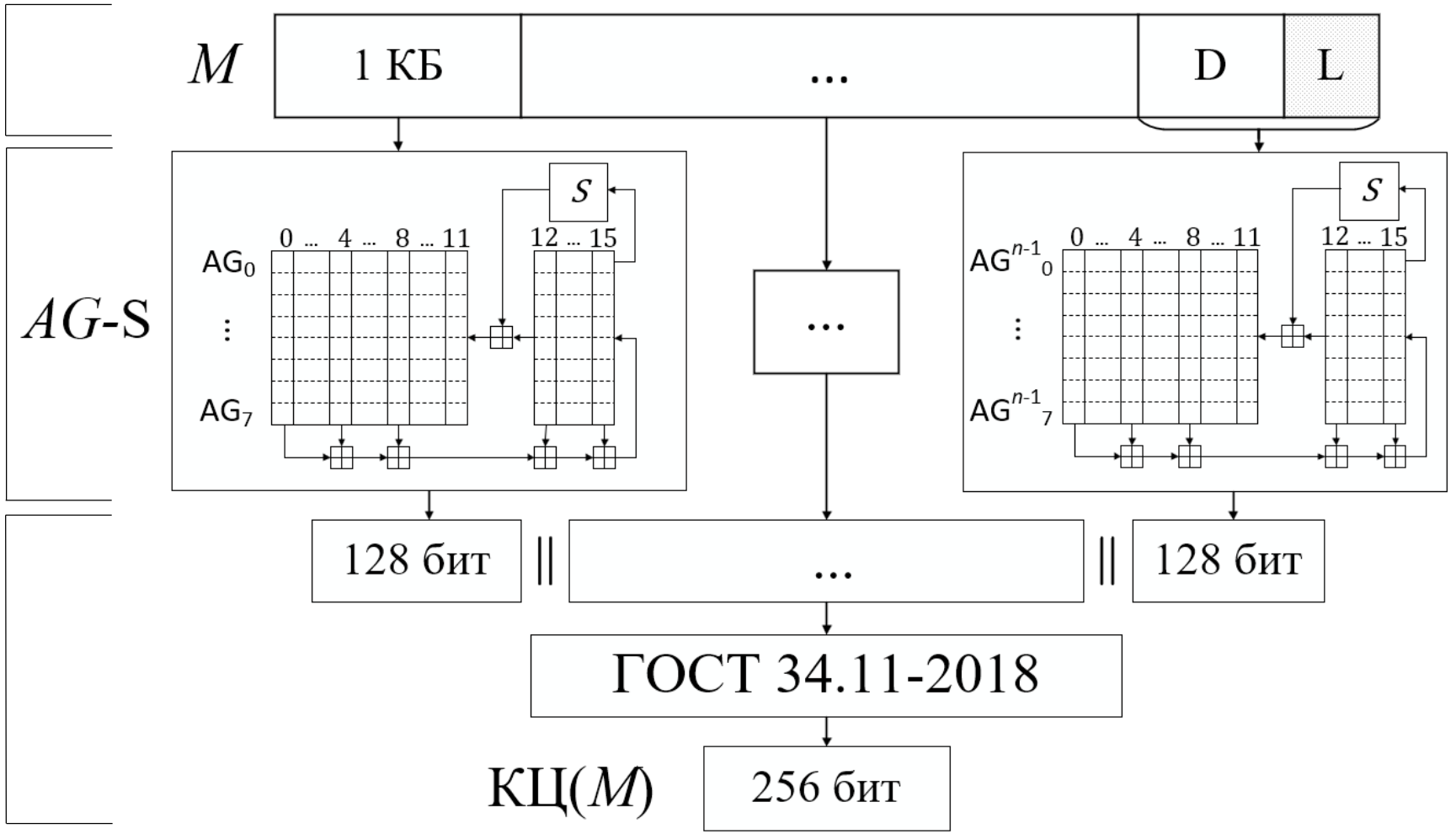
## Задачи

- Обоснование выбора параметров для реализации высокопроизводительного алгоритма генерации кода КЦ;
- Построение корректной и экономной процедуры дополнения блоков данных;
- Оценка производительности и криптографических свойств построенного алгоритма и сравнение его характеристик с другими алгоритмами.



[1] В. М. Фомичев, А. М. Коренева, Т. Р. Набиев, «О новом алгоритме контроля целостности данных», РусКрипто’20.

# Комбинированный алгоритм *AG-S*-Стрибог



# О процедуре дополнения

При расчете КС файла требуется, чтобы его длина была кратна величине $l = 2^{13}$ бит. Для этого при работе с файлами произвольной длины может потребоваться применение процедуры дополнения файла до требуемой длины.

Проанализированы схемы дополнения:

- Дополнение битами (00..0, 00…01).
- Двухступенчатое дополнение SHA, MD5.
- Дополнение случайными байтами (ISO 10126).

Разработана схема дополнения на основе выходов линейного конгруэнтного генератора (ЛКГ) MMIX Д. Кнута.

Начальное состояние ЛКГ формируется на основе:

- Данных дополняемого блока
- Исходной длины

По результатам анализа производительности и свойств дополнения установлены преимущества схемы на основе ЛКГ по сравнению с другими схемами.

# Процедура дополнения

Пусть $D$ – блок длины $p$ бит, требуется его дополнить до блока длины $l = 2^{13}$ бит, где $1 \leq p < l$.

С помощью ЛКГ генерируем блок длины $l$ - $p$.

Описание процедуры: Формирование начального значения ЛКГ $X_0$:

- Строке $P$ присваивается значение $D$, т.е. $P = D$.
- Строка $P$ дополняется нулями до длины кратной $r = 64$, т.е. $P^* = \left(P \,||\, 0^{r-(p \bmod r)}\right)$.
- Строка $P^*$ разбивается на $\left\lceil \frac{p}{r} \right\rceil$ блоков длины $r$: $P^* = (P_1 || \dots || P_{\left\lceil \frac{p}{r} \right\rceil})$, где $P_1, \dots, P_{\left\lceil \frac{p}{r} \right\rceil} \in V_r$.
- $X_0 = (P_1 \boxplus \cdots \boxplus P_{\left\lceil \frac{p}{r} \right\rceil}) \ll \left(\left(\frac{p}{8}\right) \bmod 64\right)$, где $a \ll b$ – операция циклического сдвига элементов строки a на b позиций.

На основе выходов ЛКГ порождается последовательность из $\left\lceil \frac{(l-p)}{r} \right\rceil$ элементов, из которых формируется строка $L^* = (X_1 || \dots || X_{\left\lceil \frac{(l-p)}{r} \right\rceil})$.

Дополнение $L$ есть старшие $l - p$ бит строки $L^*$, т.е. $L = MSB_{l-p}(L^*)$.

# Высокопроизводительный алгоритм *AG-S*

Пусть $D$ – начальное двоичное состояние ячеек АГ длины $l = 2^{13}$ бит, требуется сгенерировать 128-битовый ККЦ $Q$ данного блока.

Алгоритм, использующий для быстрого перемешивания данных S-боксы как функцию для аргументов из различных АГ, вычисляет $\psi(g^8)$: $V_{8192} \to V_{128}$, где $g$: $V_{8192} \to V_{8192}$ — преобразование множества состояний схемы из 8 идентичных аддитивных генераторов длины 16 над множеством двоичных 64-разрядных векторов $AG_0,\dots,AG_7$. Функция обратной связи АГ: $f(X_0,\dots,X_{15}) = X_0 \boxplus X_4 \boxplus X_8 \boxplus X_{12} \boxplus X_{15}$.

На такте $t$, $0 < t \leq 8$, преобразование $S^t$ задано формулой:

$$S^t = \left(s_0^t(\omega^t), \dots, s_7^t(\omega^t)\right),$$

где $s$ – преобразование, реализуемое s-боксом из ГОСТ 34.11-2018;

$s_0^t(\omega^t) = s(\omega^t)$, $s_j^t(\omega^t) = s\left(s_{j-1}^t(\omega^t) \oplus \omega^t\right)$, $j = 1, \dots, 7$;

$\omega^t = \left(\sigma\left(\bar{X}_{0,15}^{(t)}\right), \dots, \sigma\left(\bar{X}_{7,15}^{(t)}\right)\right)$;

$\bar{X}_{i,j}^{(t)} = (x_{i,j,0}^{(t)}, \dots, x_{i,j,63}^{(t)})$ — состояние $t$-м такте $j$-й ячейки $AG_i$;

$\sigma(x_0, \dots, x_{63}) = x_0 \oplus \dots \oplus x_{63}$.

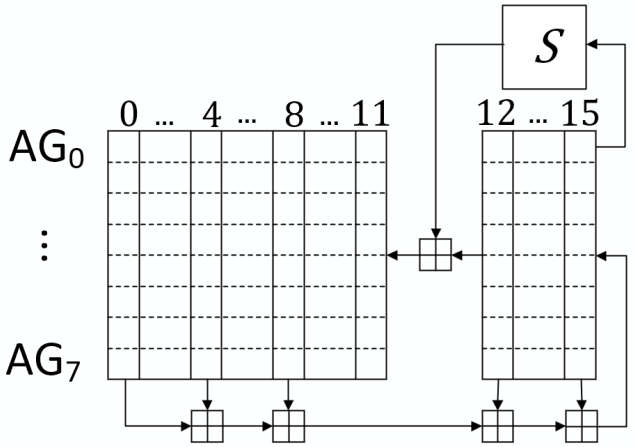



Рисунок – Регистр над $((Z_{2^{64}})^8, \boxplus)$

$$\left(X_{i,0}^{t+1}, \dots, X_{i,15}^{t+1}\right) = \left(Y_{i,1}^t, \dots, Y_{i,15}^t, f\left(Y_{i,0}^t, \dots Y_{i,15}^t\right)\right),$$

где $Y_{i,j}^t = X_{i,j}^t$ при $j \neq 12$ и $Y_{i,12}^t = X_{i,12}^t \boxplus S^t$.

$$Q = \left(X_{0,15}^{(8)} \boxplus X_{1,15}^{(8)} \boxplus X_{2,15}^{(8)} \boxplus X_{3,15}^{(8)}, X_{4,15}^{(8)} \boxplus X_{5,15}^{(8)} \boxplus X_{6,15}^{(8)} \boxplus X_{7,15}^{(8)}\right).$$

# Свойства алгоритма *AG*-S

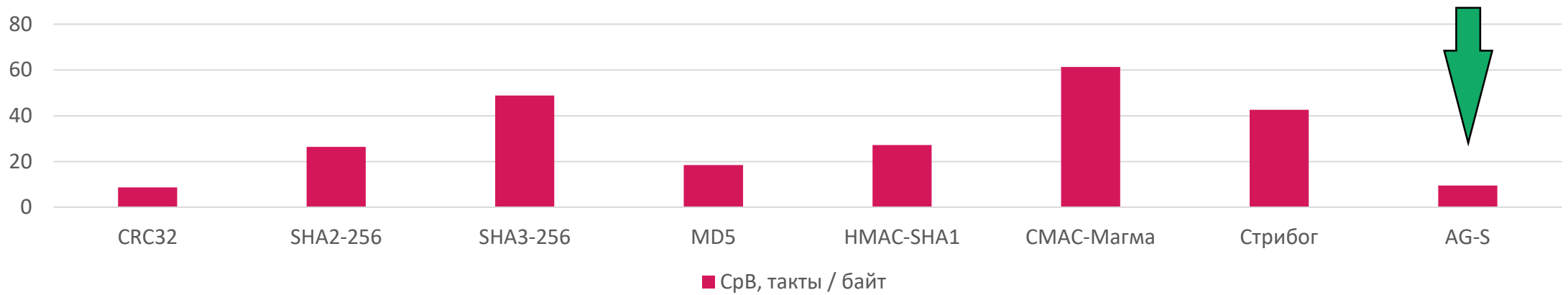
- Преобразование множества состояний $AG_0, \ldots, AG_7$ биективно.
- Оценка сложности поиска блоков с одинаковыми ККЦ (коллизий) - порядка $2^{64}$ опробований входов.
- Экспонент перемешивающего графа преобразования генератора равен 4, благодаря конструкции, составленной из АГ и S-боксов.
- С помощью матрично-графового подхода к оценке перемешивающих свойств выбраны параметры преобразований *AG*-S, а также оценена компромиссная глубина итерации преобразования *AG*-S, приемлемая для требуемых производительности и криптографических свойств.

# Результаты экспериментальных исследований-1

Измерена производительность генерации контрольных кодов разными алгоритмами при входных блоках 1 КБ: AG-S производительнее от 2 до 6 раз, чем криптографические функции хэширования.

Таблица. Характеристики производительности

| Алгоритм | CRC32 | SHA2-256 | SHA3-256 | MD5 | HMAC-SHA1 | CMAC-GOST | Стрибог | AG-S |
|---|---|---|---|---|---|---|---|---|
| CpB, такты / байт | 8.664 | 26.435 | 48.855 | 18.405 | 27.202 | 61.306 | 42.55 | 9.516 |




ПЭВМ Intel Core i5-8600 3.1 GHz без использования процессорных инструкций SHANI, SSE4.2.

# Результаты экспериментальных исследований-2

Проведено сравнение алгоритмов помощью набора тестов *SMHasher*[1,2].

Таблица. Результаты применения набора тестов *SMHasher*

| Тест \ Алгоритм | 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 |

- *Sanity* – на идентичность реализации хэш-функции и ее математической модели;
- *Avalanche* – на выполнение строгого лавинного критерия;
- *Chi2* – на равномерность распределения хэш-значений с помощью статистики $\chi^2$;
- *Differential* – на поиск коллизий во 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.

[1] Набор тестов SMHasher: https://github.com/rurban/smhasher
[2] Хэш-функция t1ha (Positive Technologies): https://github.com/PositiveTechnologies/t1ha

# Выводы

- Предложено альтернативное решение по контролю целостности на основе комбинации высокопроизводительного алгоритма *AG*-S и хэш-функции ГОСТ 34.11–2018, отвечающей современным криптографическим требованиям.
- Экспериментально исследованы производительность и криптографические свойства нового алгоритма.

Перспективные направления исследования:

- Корректировка алгоритма с целью более эффективного прохождения пакета тестов SMHasher.
- Доработка алгоритма с целью представления его на публичной платформе GitHub.

# Спасибо за внимание!

## Вопросы

# Контактная информация

Электронная почта:

d.bobrovskiy@securitycode.ru

Facebook:

facebook.com/sec.code.russia

Сайт:

www.securitycode.ru