Rublev Dmitry Pavlovich
Taganrog Institute of Technology – Federal State-Owned Autonomy Educational Establishment of Higher Vocational Education "Southern Federal University".
E-mail: rublev-d@yandex.ru.
2, Chekhova Street, Taganrog, 347928, Russia.
Phone: +78634371905.
The Department of Security in Data Processing Technologies; Associate Professor.

Makarevich Oleg Borisovich
E-mail: mak@tsure.ru.
Phone: +78634312018.
The Department of Security in Data Processing Technologies; Head of the Department.

Fedorov Vladimir Mikhailovich
E-mail: vladmih@rambler.ru.
Phone: +78634371905.
The Department of Security in Data Processing Technologies; Associate Professor.

Panchenko Eugene Mikhailovich
Institute of Physics, Southern Federal University.
E-mail: kordon@kordon-rnd.ru.
194, Pr. Strikes, Rostov-on-Don, 344090, Russia.
Phone: +78632905121.
The Deputy Director on Scientific Work

УДК 681.3

И.А. Калмыков, О.И. Дагаева

РАЗРАБОТКА ПСЕВДОСЛУЧАЙНОЙ ФУНКЦИИ ПОВЫШЕННОЙ ЭФФЕКТИВНОСТИ

Рассмотрены вопросы синтеза псевдослучайной функции повышенной эффективности. Доказано, что псевдослучайная функция может быть построена на основе каскадной схемной реализации. Полученные результаты свидетельствуют о том, что разработанная псевдослучайная функция обладает стойкостью к атакующим алгоритмам не хуже, чем у ранее известных реализаций таких функций, при меньшей размерности секретного ключа.

Псевдослучайные функции; параллельная стойкость; каскадная конструкция; l-DDH предположение.

I.A. Kalmikov, O.I. Dagaeva

DESIGN OF ALGEBRAIC PSEUDORANDOM FUNCTIONS WITH IMPROVED EFFICIENCY

There is considered issues related with constructing an algebraic pseudorandom function with improved efficiency. It is proved that the pseudorandom function can be based on a cascade construction. Our pseudorandom function is secure to attacking algorithms not less then counterparts and it uses shorter private keys.

Pseudorandom functions; parallel security; cascade construction; l-DDH assumption.

Введение. В настоящее время псевдослучайные функции (ПСФ) нашли широкое применение в различных сферах. Современная информатика широко использует псевдослучайные числа в самых разных приложениях – от метода Монте-Карло и имитационного моделирования до криптографии. При этом от качества используемых ПСФ напрямую зависит качество получаемых результатов. Поэтому вопросам разработки алгоритмов вычисления псевдослучайных функций повышенной эффективности уделяется значительное внимание.

1. Постановка задачи исследования. Псевдослучайные функции стали неотъемлемым элементом любой компьютерной системы (КС), независимо от ее сложности и назначения. Как правило, для их реализации используются программные и программно-аппаратные средства генерации. Можно выделить следующие задачи, для решения которых используются генераторы ПСП:

- техническое диагностирование компонентов КС (в том числе встроенное диагностирование);
- контроль хода выполнения программ с использованием сторожевых процессоров;
- помехоустойчивое кодирование;
- защита информации.

Во всех вышеперечисленных случаях ПСФ используются либо непосредственно, либо на их основе строятся алгоритмы хеширования информации. В последних двух случаях качество операций генерации псевдослучайных функций и хеширования определяется в первую очередь эффективностью ПСФ. Таким образом, именно от свойств псевдослучайных функций ПСП, особенно в тех случаях, когда необходимо обеспечить устойчивую работу КС при наличии случайных и умышленных деструктивных воздействий, в значительной степени зависит надежность процессов сбора, обработки, хранения и передачи информации. Поэтому задача разработки высокоэффективных ПСФ является актуальной.

Псевдослучайные функции впервые были определены Голдрайхом (Oded Goldreich), Голдвассером (Shafi Goldwasser) и Микали (Silvio Micali) в 1986 г. и с тех пор являются фундаментальным строительным блоком в современной криптографии, имея множество различных применений [1]. На сегодняшний день ПСФ широко используются в следующих областях криптографии:

- в шифровании;
- в системах электронных платежей;
- для обеспечения целостности передаваемого сообщения;
- в механизмах электронно-цифровых подписей;
- для генерации ключей из некоторого секретного значения (например, мастер-ключа);
- для аутентификации пользователей.

В настоящее время известно несколько алгоритмов, позволяющих получать довольно хорошие псевдослучайные функции. Так в работе [2] была представлена ПСФ Наора-Рейнголда, стойкость которой была выведена из сложности решения проблемы принятия решения Диффи-Хеллмана (DDH). Алгоритм вычисления такой ПСФ определяется следующим образом: на ее вход поступают m-битная строка $b = b_1,\ldots,b_m \in \{0,1\}^m$ и секретный ключ $\left(h, x_1,...,x_m\right)$, результатом работы является

$$F_{NR}\left(\left(h,x_1,\ldots,x_m\right),\left(b_1,\ldots,b_m\right)\right) := h^w, \quad \text{где } w := \prod_{i=1}^{m} x_i^{b_i}. \qquad (1)$$

Вычислительная сложность этой функции составляет m–1 модулярных умножений для вычисления w и одно заключительное возведение в степень.

Однако алгебраическая конструкция ПСФ Наора-Рейнголда, несмотря на то, что она лежит в основе многих криптографических схем и даже таких алгебраических конструкций, как верифицируемые случайные функции, рассеянные и распределенные ПСФ, обладает недостатками, среди которых можно выделить большой размер секретного ключа и довольно низкое быстродействие ее технической реализации [1–3].

С целью устранения данных недостатков была разработана алгебраическая ПСФ, имеющая точно такой же размер области определения, как ПСФ Наора-Рейнголда и Бонеха-Монтгомери-Рагунатана (БМР ПСФ) [3], но использующая более короткий секретный ключ по сравнению с ПСФ Наора-Рейнголда. Рассмотрим особенности построения такой псевдослучайной функции.

Для некоторых параметров ℓ и n разработанная ПСФ на вход принимает входную последовательность $(x_1,\ldots,x_n)\in\mathbb{Z}_\ell^n$ вместе с ключом $(h,s_1,\ldots,s_n)$ и на выход выдаёт

$$F\left((h,s_1,\ldots,s_n),(x_1,\ldots,x_n)\right):=h^w, \text{ где } w:=\prod_{i=1}^{n}s_i^{x_i}. \quad (2)$$

Для области определения размером 2^m значение $n=m/\log_2\ell$, вследствие чего при вычислении данной функции требуется в $\log_2\ell$ раз меньше умножений, но на $m/\log_2\ell$ больше возведений в степень по сравнению с (1) для вычисления значения *w*. В общей сумме нам необходимо $2m-1-m/\log_2\ell$ умножений для вычисления значения *w*. Основным преимуществом данной ПСФ является использование меньшего объема памяти для вычисления значения функции, так как она использует ключ в $\log_2\ell$ раз меньший размером по сравнения с ПСФ Наора-Рейнголда. Стойкость ПСФ основывается на предположении о сложности решения ℓ-DDH проблемы. Взаимосвязь между стойкостью функции и значением ℓ следующая: чем больше ℓ, тем более строгим становится это предположение, и тем менее стойкой становится функция. Таким образом, необходимо выдерживать небольшое значение ℓ. В качестве примера можно использовать значения ℓ в пределах от 16 или 256.

Для доказательства стойкости получившейся ПСФ предлагается использование определенной в [5] конструкции расширенного каскада, позволяющего конструировать ПСФ с большой областью определения из ПСФ с малой областью определения. Интересным свойством рассматриваемого расширенного каскада является отсутствие взаимосвязи между его стойкостью и стойкостью, лежащей в его основе ПСФ. В работе [4] сформулировано достаточное условие параллельной стойкости, лежащей в основе ПСФ, существование которой обеспечивает стойкость расширенного каскада.

С использованием теоремы о стойкости расширенного каскада была построена новая ПСФ с большой областью определения введением в расширенней каскад ПСФ в малой областью определения. Исходная функция получает в качестве входных данных *m*-битную строку $x\in[\ell]$ и секретный ключ (h,s) и выдает значение

$$F\left((h,s),x\right):=h^{s^x}. \quad (3)$$

Для определения стойкости разработанной функции необходимо доказать, что алгоритм вычисления ПСФ (3) позволяет получить псевдослучайную функцию, которая обладает свойством параллельной стойкости.

С этой целью необходимо первоначально выбрать ПСФ с малой областью определения, которая ляжет в основу, а затем расширить ее область определения на основе конструкции расширенного каскада. Стойкость новой функции определяется утверждением о том, что достаточным условием стойкости расширенного каскада является удовлетворение лежащей в его основе функции свойству парал-

лельной стойкости. Для проверки соответствия этому свойству строится функция $F^{(q)}$ (3) и доказывается ее стойкость.

Если $\mathbb{G}$ – группа простого порядка p, тогда исходная функция представляет отображение $f:(\mathbb{Z}_p\times\mathbb{G})\times[\ell]\to\mathbb{G}$, где $(\mathbb{Z}_p\times\mathbb{G})$ – ключевое пространство, $[\ell]=\{0,1,\ldots,\ell-1\}$ – область определения и $\mathbb{G}$ – область значения, и определяется как

$$f\left((s,h),x\right):=h^{s^x}. \tag{4}$$

Введение функции (4) в конструкцию расширенного каскада позволяет получить ПСФ с экспоненциальной областью определения $[\ell]^n$ следующего вида:

$$F_{new}:=f^{*n}\left((s_1,\ldots s_n,h),(x_1,\ldots x_n)\right):=h^{\prod_{i=1}^{n}s_i^{x_i}}. \tag{5}$$

В результате сравнения алгоритма (1) получения ПСФ Наора-Рейнголда с алгоритмом (5) можно сделать вывод о том, что отличиетльной особенностью последнего алгоритма является повышенный размер области определения.

Приведенная ниже теорема определяет стойкость новой ПСФ путем сведения задачи взлома ПСФ к решению одной из существующих тяжелоразрешимых теоретико-сложностных проблем.

Теорема 1. Псевдослучайная функция, определенная выражением (5), является стойкой при условии неразрешимости ℓ-DDH проблемы в группе $\mathbb{G}$.

Для доказательства теоремы воспользуемся леммами.

При оценке стойкости новой ПСФ достаточно показать то, что лежащая в основе расширенного каскада функция, определенная в (4), удовлетворяет свойству q-параллельной стойкости. Для этого необходимо определить функцию $f^{(q)}$, эмулирующую q копий функции (4) и доказать ее стойкость.

Пусть функция $f^{(q)}:(\mathbb{Z}_p\times\mathbb{G}^q)\times([\ell]\times[q])\to\mathbb{G}$ определяется как

$$f^{(q)}\left((s,h_1,\ldots,h_q),(x,i)\right):=h_i^{s^x}, \tag{6}$$

где $s\in\mathbb{Z}_p$, $h_1,\ldots,h_q\in\mathbb{G}^q$, $x\in[\ell]$, $i\in[q]$. Докажем, что она является стойкой ПСФ для всех полиномиальных значений q.

Лемма 1. Если функция f, определяемая как (4), является стойкой ПСФ и проблема DDH является тяжелоразрешимой в $\mathbb{G}$, то функция f является q-параллельно стойкой для любого полинома q в параметре безопасности.

В частности, для каждого атакующего ПСФ алгоритма $\mathcal{A}$ существуют такие атакующие алгоритмы $\mathcal{B}_1$ и $\mathcal{B}_2$, имеющие время выполнения приблизительно равное времени выполнения $\mathcal{A}$ вплоть до полиномиального множителя, что

$$PRF_{adv}\left[\mathcal{A},f^{(q)}\right]\le PRF_{adv}\left[\mathcal{B}_1,f\right]+q\cdot DDH_{adv}\left[\mathcal{B}_2,\mathbb{G}\right]. \tag{7}$$

Так как предположение о сложности решения k-DDH-проблемы подразумевает сложность решения DDH проблемы (при k = 1 k-DDH-предположение является просто DDH-предположением), необходимость в дополнительном доказательстве сложности решения DDH-проблемы в $\mathbb{G}$ отпадает.

Таким образом, стойкость новой ПСФ зависит от выполнения двух условий: стойкости, лежащей в основе каскада функции и трудноразрешимости проблемы DDH в $\mathbb{G}$. Следующая лемма ограничивает условия стойкости ПСФ *f*, определенной в (4).

Лемма 2. Если ℓ-DDH-проблема является трудноразрешимой для группы $\mathbb{G}$, то функция *f*, определенная в (4), является стойкой ПСФ с полиномиальным размером области определения ℓ в качестве параметра безопасности.

В частности, для каждого алгоритма $\mathcal{A}$, атакующего ПСФ, существует такой атакующий алгоритм $\mathcal{B}$ с приблизительно таким же временем выполнения, что и $\mathcal{A}$, вплоть до полиномиального множителя:

$$PRF_{adv}\left[\mathcal{A}, f\right] \le \ell \cdot \mathsf{DDH}^{(\ell)}_{adv}\left[\mathcal{B}, \mathbb{G}\right]. \tag{8}$$

Для того чтобы доказать стойкость функции *f*, определенной в (4), достаточно продемонстрировать, что ℓ выходных значений этой функции *f* представляют собой выходную последовательность криптографически сильного (стойкого) псевдослучайного генератора (ПСГ). Таким образом, доказательство стойкости ПСФ сводится к доказательству того, что выходная последовательность

$$G(s,h) := h, h^s, h^{s^2}, \ldots, h^{s^{\ell-1}}. \tag{9}$$

является выходной последовательностью криптографически стойкого ПСГ при условии существования ℓ-DDH-предположения в $\mathbb{G}$.

Для определения псевдослучайного генератора необходимо ввести понятие вычислительной неразличимости двух распределений. Формально, два распределения $\mathcal{D}_1$, $\mathcal{D}_2$ называются вычислительно неразличимыми, если для любого полиномиального вероятностного алгоритма $\mathcal{A}$

$$\left| Pr_{x \leftarrow \mathcal{D}_1}\left[\mathcal{A}(x) = 1\right] - Pr_{x \leftarrow \mathcal{D}_2}\left[\mathcal{A}(x) = 1\right] \right| = \nu(|x|). \tag{10}$$

Неформально можно представить задачу следующим образом. Пусть *X* и *Y* – такие множества, что по выбранному случайным образом элементу $x \in X$ возможно с помощью некой специальной функции *G*(*x*) породить якобы случайный элемент из *Y*. Будем рассматривать только случаи с мощностью множества *Y* намного большей мощности множества *X*, иначе задача существенно упрощается. Например, рассмотрим случай, когда *X* = {0...1000}, *Y* = {0...100}. Тогда, выбрав случайный $x \in X$, можно легко вычислить *y* = *x* div 10, который, очевидно, будет принадлежать *Y* и будет случайным в силу случайности выбора переменной *x*.

В случае, когда мощность множества *Y* намного большей мощности множества *X*, функция *G*: *X* → *Y* называется псевдослучайным генератором, если $G(\mathcal{U}_X)$ и $\mathcal{U}_Y$ вычислительно неразличимы ($\mathcal{U}_X, \mathcal{U}_Y$ – равномерные распределения).

Известно, что ℓ-DDH-строка имеет вид $\left(g^{1/x}, g, g^{x^2}, \ldots, g^{x^{\ell-1}}\right)$. Если рассматривать $x = s, g = h^s$, то отсюда обратный элемент определяется как $g^{1/x} = h$, $g^2 = h^{s^2}$. Тогда имеем $\left(h, h^s, h^{s^2}, \ldots, h^{\ell-1}\right) = \left(g^{1/x}, g, g^{x^2}, \ldots, g^{x^{\ell-1}}\right)$. Следовательно, выходная последовательность ПСФ является представленной в ином виде ℓ-DDH строкой.

Лемма 3. Если ℓ-DDH-предположение существует для группы $\mathbb{G}$, то элемент h является неотличимым от случайно выбранного элемента $h^{\mathrm{pow}(s,\ell)}$. Более точно, для алгоритма $\mathcal{A}$ преимущество

$$\mathsf{DDH}_{adv}^{(\ell)}[\mathcal{A},\mathbb{G}] := \left|\Pr\left[\mathcal{A}\left(h^{\mathrm{pow}(s,\ell)},h\right)=1\right]-\Pr\left[\mathcal{A}\left(h^{\mathrm{pow}(s,\ell)},u\right)=1\right]\right| = \mathrm{negl}, \quad (11)$$

где $u,h \leftarrow^R \mathbb{G}$, $s \leftarrow^R \mathbb{Z}_p$, $\mathrm{pow}(s,\ell)=\left(1,s,s^2,\ldots,s^{\ell-1}\right)\in\mathbb{Z}_p^{\ell}$.

Для доказательства леммы 3 воспользуемся гибридным аргументом.

Пусть имеется следующая последовательность ℓ+1 гибридных экспериментов между запросчиком и атакующим алгоритмом $\mathcal{A}$. Пусть $\mathcal{A}$ – алгоритм, умеющий различать псевдослучайную последовательность, выработанную ПСГ G от случайной входной последовательности, от настоящей случайной последовательности. В гибридном эксперименте i запросчик заменяет первые i выходных элемента ПСГ настоящими случайными числами, в то время как последние ℓ-i элементов являются выработанными ПСГ.

Более точно, при $i=0,\ldots,\ell$ поведение запросчика в гибридном эксперименте Exp_i определяется следующим алгоритмом:

ВВОД: ключи $s\in_R \mathbb{Z}_p$ и $h\in_R \mathbb{G}$

ВЫВОД:

ЕСЛИ $i\neq 0$ ТО

ЦИКЛ ДЛЯ j ОТ 1 ДО n [ШАГ 1]:

$u_j \leftarrow_R \mathbb{G}$

КЦ

ИНАЧЕ

ЦИКЛ ДЛЯ j ОТ i+1 ДО ℓ [ШАГ 1]:

$a_{j+1} := f\left((s,h),j\right) := h^{s^j}$

КЦ.

Для $i=1,\ldots,\ell$ определим W_i как вероятность совпадения битов b и b', т.е. правильного распознавания алгоритмом $\mathcal{A}$ последовательности в гибридном эксперименте i. Следует отметить, что в гибридном эксперименте Exp_0 противник $\mathcal{A}$ получает для анализа псевдослучайную последовательность, выданную ПСГ, в то время как в гибридном эксперименте Exp_ℓ противник получает не что иное, как совершенно случайную последовательность. Отсюда

$$\mathit{PRG}_{adv}[\mathcal{A},f] = \left|W_\ell - W_0\right|. \quad (12)$$

Из техники гибридного аргумента следует, что существует такое значение $t\in[1,\ell]$, что

$$\mathit{PRG}_{adv}[\mathcal{A},f] = \ell\cdot\left|W_{t-1} - W_t\right|. \quad (13)$$

Другими словами, в гибридных экспериментах Exp_t и Exp_{t+1} получаемые противником последовательности вычислительно неразличимы:

$$H_t =_c H_{t+1} : \left(u_1,\ldots,u_t,h,h^s,\ldots,h^{s^{\ell-1-i}}\right) =_c \left(\underbrace{u_1,\ldots,u_{t+1}}_{t\ times},h^s,\ldots,h^{s^{\ell-1-i}}\right). \quad (14)$$

Для сведения решения проблемы различения абсолютно случайной и псевдослучайной последовательностей к теоретико-сложностному предположению конструируем такой атакующий алгоритм $\mathcal{B}$, умеющий решать ℓ-DDH-проблему, что

$$DDH_{adv}^{(\ell)}\left[\mathcal{B},\mathbb{G}\right]=\left|W_{t-1}-W_t\right|. \quad (15)$$

Комбинирование (14) и (15) доказывает Лемму 2.

В гибридных экспериментах Exp_t и Exp_{t+1} атакующий алгоритм $\mathcal{B}$, с одной стороны, взаимодействует с ℓ-DDH запросчиком, а с другой стороны, параллельно эмулирует запросчика ПСГ для атакующего алгоритма $\mathcal{A}$.

На этапе инициализации ℓ-DDH-запросчик выбирает случайным образом $u_t,h\in\mathbb{G},s\in\mathbb{Z}_p$ и бит $b\in\{0,1\}$. В случае, если $b=0$, запросчик выдает алгоритму $\mathcal{B}$ ℓ-DDH-строку $\left(u,h^s,h^{s^2},\ldots,h^{\ell-1}\right)$, а если $b=1$, то запросчик выдают последовательность случайных элементов $\left(h,h^s,h^{s^2},\ldots,h^{\ell-1}\right)$.

Атакующий алгоритм $\mathcal{B}$ осуществляет свои действия в следующем порядке. На этапе инициализации случайным образом выбираются элементы $u_1,\ldots,u_t,h\in\mathbb{G},s\in\mathbb{Z}_p$. Затем алгоритм $\mathcal{B}$ получает от своего ℓ-DDH-запросчика некоторую последовательность $\left(y,h^s,h^{s^2},\ldots,h^{\ell-1}\right)$, где может быть $y=u_t$ (т.е. y – случайно выбранный элемент из множества $\mathbb{G}$) или $y=h$. Затем алгоритм $\mathcal{B}$ к t случайным элементам присоединяет полученную от своего запросчика строку и отправляет результирующую последовательность $\left(u_1,\ldots,u_{t-1},y,h^s,h^{s^2},\ldots,h^{\ell-t-1}\right)$ атакующему алгоритму $\mathcal{A}$. Наконец, атакующий алгоритм $\mathcal{A}$ выдает бит $b'\in\{0,1\}$ своему запросчику (т.е. алгоритму $\mathcal{B}$), который свидетельствует о его решении по поводу принятой им последовательности. Атакующий алгоритм $\mathcal{B}$, в завершение всего гибридного эксперимента, просто пересылает тот же самый бит своему запросчику в качестве ответа на то, случайной или ℓ-DDH-стройкой была принятая им последовательность.

В случае если ℓ-DDH-запросчик алгоритма $\mathcal{B}$ выбирает элемент $y=u_t$, выдаваемая им строка является случайной, и таким образом алгоритм $\mathcal{B}$ эмулирует эксперимент Exp_{t+1} между ПСГ-запросчиком и атакующим его алгоритмом $\mathcal{A}$.

В случае если ℓ-DDH-запросчик алгоритма $\mathcal{B}$'s выбирает элемент $y=h$, выдаваемая им строка представляет собой ℓ-DDH-строку, и таким образом алгоритм $\mathcal{B}$ эмулирует эксперимент Exp_t между запросчиком и атакующим его $\mathcal{A}$.

Таким образом, мы можем говорить о том, что различение выходной последовательности ПСГ от последовательности случайных чисел сводится к решению ℓ-DDH-проблемы и поэтому может считаться вычислимо неразрешимой задачей. Как и требуется, равенство (8) выполняется, что завершает доказательство леммы 2.

Доказательство леммы 1. Для доказательства леммы необходимо продемонстрировать явным образом, что функция $f^{(q)}$, определенная в (11), является стойкой ПСФ. Для этого предлагается использовать описанную Бонехом, Монтгомери и Рагунатаном в [5] схему доказательства.

Доказательство стойкости представляется в виде последовательности трех игр между запросчиком и атакующим функцию $f^{(q)}$ алгоритмом $\mathcal{A}$. Для $i = 0, 1, 2$ обозначение W_i определяет вероятность выигрыша атакующего алгоритма $\mathcal{A}$, т.е. вероятность того, что бит b в i-й игре угадан алгоритмом правильно $(b' = b)$.

Игра № 0. В этой игре запросчик играет роль обычного запросчика для функции $f^{(q)}$, предоставляя атакующему алгоритму $\mathcal{A}$ доступ к оракулу функции $f^{(q)}$, используемому случайный ключ.

Игра № 1. В данном случае запросчик выбирает случайную функцию вида $u : [\ell] \to \mathbb{G}$ и случайным образом генерирует значения $r_1, \ldots, r_q \in \mathbb{Z}_p$. Затем на запрос $(x, i) \in [\ell] \times [q]$ от атакующего алгоритма возвращает вычисленное значение $u(x)^{r_i}$.

Очевидно, что игры 0 и 1 неотличимы друг от друга в том случае, если функция f – стойкая ПСФ. В частности, существует такой атакующий алгоритм $\mathcal{B}_1$, время выполнения которого приблизительно равно времени выполнения атакующего алгоритма $\mathcal{A}$, что

$$|W_0 - W_1| = PRF_{adv}[\mathcal{B}_1, f]. \tag{16}$$

Атакующий алгоритм $\mathcal{B}_1$ взаимодействует с запросчиком, предоставляющим доступ к оракулу функции f либо оракулу случайной функции $u : [\ell] \to \mathbb{G}$ с одной стороны, а с другой – сам играет роль запросчика для атакующего алгоритма $\mathcal{A}$, предоставляя доступ к оракулу функции $f^{(q)}$ либо случайному оракулу.

Атакующий алгоритм $\mathcal{B}_1$ работает описанным ниже образом. На этапе инициализации случайным образом выбираются значения $r_1, \ldots, r_q \in \mathbb{Z}_p$. После этого атакующий алгоритм $\mathcal{B}_1$ готов обрабатывать запросы алгоритма $\mathcal{A}$ в качестве запросчика и переходит в режим ожидания. Каждый раз, получая запрос от атакующего алгоритма $\mathcal{A}$ вида $(x, i) \in [\ell] \times [q]$, атакующий алгоритм $\mathcal{B}_1$ обращается к своему запросчику с вопросом x, на что последний возвращает некоторое значение y. Затем атакующий алгоритм $\mathcal{B}_1$ вычисляет значение y^{r_i} и отправляет результат в качестве ответа на запрос атакующему алгоритму $\mathcal{A}$. После выполнения алгоритмом $\mathcal{A}$ q-запросов он принимает решение о том, с оракулом какой функции он имел дело, и в соответствии с этим выдает атакующему алгоритму $\mathcal{B}_1$ выходной бит b. Игра завершается пересылкой принятого алгоритмом $\mathcal{B}_1$ бита своему запросчику в качестве решения о том, ответы какого оракула атакующий алгоритм $\mathcal{B}_1$ получал.

Когда запросчик атакующего алгоритма $\mathcal{B}_1$ эмулирует доступ к оракулу функции f со случайным образом выбранным ключом (s, h), то он на запрос x атакующего алгоритма $\mathcal{B}_1$ отвечает результатом вычисления значения ПСФ $y = h^{s^x}$. Если для $i = 1,\ldots,q$ определить $h_i = h^{r_i}$, то можно считать, что атакующий алгоритм $\mathcal{A}$ на запрос (x,i) алгоритма $\mathcal{A}$ выдает значение $h_i^{s^x}$, которое как раз является результатом вычисления псевдослучайной функции which is precisely $f^{(q)}$. Следовательно, в этом случае атакующий алгоритм $\mathcal{B}_1$ эмулирует игру Game 0 запросчика с атакующим алгоритмом $\mathcal{A}$.

Когда запросчик атакующего алгоритма $\mathcal{B}_1$ эмулирует доступ к случайному оракулу, то есть к случайной функции вида $u:[\ell]\to\mathbb{G}$, то ответом запросчика атакующего алгоритма $\mathcal{A}$ (то есть ответом атакующего алгоритма $\mathcal{B}_1$) на запрос (x,i) атакующего алгоритма $\mathcal{A}$ является просто некоторое значение $u(x)^{r_i}$, которое полностью совпадает с выданным бы запросчиком в игре № 1.

Таким образом, два вышеприведенных аргумента полностью доказывают утверждение (14).

Игра № 2. Запросчик предоставляет атакующему алгоритму $\mathcal{A}$ доступ к случайному оракулу, т.е. оракулу случайной функции $w:[\ell]\times[q]\to\mathbb{G}$.

Свидетельством неразличимости игр № 1 и № 2 при условии тяжелоразрешимости проблемы DDH для группы $\mathbb{G}$ может служить лемма 1. Применительно к нашему случаю, в частности, существует такой атакующий DDH проблему алгоритм $\mathcal{B}_2$, что

$$W_1 - W_2 \le q \cdot \mathsf{DDH}_{adv}[\mathcal{B}_2,\mathbb{G}]. \tag{17}$$

Пусть $(x_1,i_1),\ldots,(x_q,i_q)$ – запросы атакующего алгоритма $\mathcal{A}$ к своему запросчику. Как уже было сказано, в игре Game 1 запросчик отвечает на запросы атакующего алгоритма $\mathcal{A}$ с использованием случайной функции $u:[\ell]\to\mathbb{G}$ с выбранными случайным образом значениями $r_1,\ldots,r_q\in\mathbb{Z}_p$. Пусть $A\in\mathbb{Z}_p^{q\times q}$ – матрица вида $A\in(r_i s_j)_{1\le i,j\le q}$. Ясно, что A имеет ранг 1.

По окончании процедуры выдачи запросов и получения ответа в игре 1 атакующий алгоритм $\mathcal{A}$ имеет q записей в матрице $g^A\in\mathbb{Z}_p^{q\times q}$. В игре № 2 атакующий алгоритм $\mathcal{A}$ получает q случайных элементов, принадлежащих множеству $\mathbb{G}$, которые мы можем рассматривать в виде q записей в случайной матрице над $\mathbb{G}$ размера $q\times q$. Согласно лемме 1, существует атакующий алгоритм $\mathcal{B}_2$, который, как и требуется, удовлетворяет условию (17).

Комбинируя (16) и (17), получаем

$$\mathsf{PRF}_{adv}\left[\mathcal{A},f^{(q)}\right] = |W_0 - W_2| \le |W_0 - W|_1 - |W_1 - W_2| \le \mathsf{PRF}_{adv}[\mathcal{B}_1,f] + q\cdot\mathsf{DDH}_{adv}[\mathcal{B}_2,\mathbb{G}]$$

что и завершает доказательство теоремы 1.

В результате доказательства теоремы 1 было установлено, что для вычисления разработанной ПСФ требуется $2m-1-m/\log_2 \ell$ умножений при возведении в степень методом «возведения в квадрат и умножения».

Заключение. Разработанная псевдослучайная функция позволяет снизить сложность вычисления ПСФ по сравнению с ранее известными алгоритмами. Кроме того, предложенная псевдослучайная функция сокращает размер секретного ключа в $\log_2 \ell$ раз, что приводит к уменьшению затрат на память. При этом для сокращения временных затрат, необходимых на вычисление синтезированной ПСФ, можно использовать конвейерную организацию.

Помимо этого, амортизирующая сложность вычисления ПСФ будет гораздо ниже, чем сложность вычисления функции для каждого входного значения. Если рассматривать ПСФ с точки зрения аппаратной реализации, то каскадная структура позволяет осуществлять конвейерные вычисления, повышая быстродействие.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Goldreich Oded, Goldwasser Shafi, Micali Silvio*. How to construct random functions // JACM. – 1986. – Vol. 33, № 4. http://www.wisdom.weizmann.ac.il/~oded/ggm.html.
2. *Naor Moni, Reingold Omer.* Number-theoretic constructions of efficient pseudo-random functions // In FOCS'97. – 1997. – P. 458-467. http://www.wisdom.weizmann.ac.il/~naor/PAPERS/gdh_abs.html.
3. *Bellare Mihir, Canetti Ran, Krawczyk Hugo*. Pseudorandom functions revisited: The cascade construction and its concrete security. In FOCS'96, 1996. http://cseweb.ucsd.edu/~mihir/ papers/ cascade.html.
4. *Dan Boneh, Shai Halevi, Michael Hamburg, and Rafail Ostrovsky*. Circular-secure encryption from decision Diffie-Hellman. In CRYPTO'08, 2008. – P. 108-125. http://crypto.stanford.edu/~dabo/abstracts/circular.html.
5. *Boneh Dan, Montgomery Hart, Raghunathan Ananth.* Algebraic pseudorandom functions with improved efficiency from the augmented cascade. In ACM Conference on Computer and Communications Security – CCS 2010 (to appear), 2010. http://crypto.stanford.edu/~dabo/pubs/ abstracts/algebprf.html.

Статью рекомендовал к опубликованию д.т.н., профессор В.В. Копытов.

Калмыков Игорь Анатольевич
Северо-Кавказский государственный технический университет.
E-mail: zik@ncstu.ru.
355000, г. Ставрополь, просп. Кулакова, 2.
Тел.: +79034163533.
Д.т.н., профессор; профессор кафедры защиты информации.

Дагаева Ольга Игоревна
Аспирант кафедры защиты информации.

Kalmykov Igor Anatol'evich
North-Caucasus State Technical University, Stavropol.
E-mail: zik@ncstu.ru.
2, Kulakova Pr., Stavropol, 355000, Russia.
Phone: +79034163533.
Dr. of Eng. Sc., Professor; professor of Chair for Information Security.

Dagaeva Olga Igorevna
Postgraduate Student of Chair for Information Security.