в отличие от случая хэш-функции MD4, для которой это неявное предположение справедливо, сложность этого этапа для RIPEMD при нашей экспериментальной проверке оказалась равной в среднем $2^{16{,}49}$. Вторая техника называется многократной модификацией сообщения. Она, по заверению авторов [6], позволяет добиться выполнения на остальных шагах всех оставшихся условий, за исключением «примерно 16», что даёт вероятность успеха второго этапа около 2^{-16}. При этом сложность второго этапа полагается равной от 1 до 4 условных операций.

Таким образом, по полученным экспериментальным данным можно вывести нижнюю оценку средней трудоёмкости алгоритма [6]. Она составляет не менее $2^{32{,}49}$ условных операций, что примерно в квадрат раз больше заявленной в [6].

Косвенным доказательством как наличия проблем при восстановлении деталей алгоритма [6], так и его высокой средней трудоёмкости (гораздо выше заявленной в [6]) служит тот факт, что, насколько известно авторам, в Интернете отсутствуют программные реализации данного алгоритма поиска коллизий для хэш-функции RIPEMD, в то время как для хэш-функций MD4 и MD5 они есть [10].

ЛИТЕРАТУРА

1. RIPE. Integrity Primitives for Secure Information Systems. Final Report of RACE Integrity Primitives Evaluation (RIPE-RACE 1040). LNCS. 1995. V. 1007. P. 69–111.
2. *Rivest R. L.* The MD4 Message Digest Algorithm // LNCS. 1991. V. 537. P. 303–311.
3. *Dobbertin H.* RIPEMD With Two-Round Compress Function Is Not Collision-Free // J. Cryptology. 1997. No. 10. P. 51–69.
4. *Debaert C. and Gilbert H.* The RIPEMD^L and RIPEMD^R Improved Variants of MD4 Are Not Collision Free // LNCS. 2002. V. 2355. P. 52–65.
5. *Wang X., Feng D., Lai X., and Yu X.* Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD // IACR Cryptology ePrint Archive. 2004. Report No. 199.
6. *Wang X., Lai X., Feng D., et al.* Cryptanalysis of the Hash Functions MD4 and RIPEMD // LNCS. 2005. V. 3494. P. 1–18.
7. *Dobbertin H., Bosselaers A., and Preneel B.* The hash function RIPEMD-160. Dedicated web-page. `http://homes.esat.kuleuven.be/~bosselae/ripemd160.html`
8. *Dobbertin H., Bosselaers A., and Preneel B.* RIPEMD-160, a strengthened version of RIPEMD // LNCS. 1996. V. 1039. P. 71–82.
9. TrueCrypt. Free open-source disk encryption software for Windows 7/Vista/XP, Mac OS X, and Linux. Dedicated web-page. `http://www.truecrypt.org/`
10. *Stach P.* MD4 Collision Generator. `http://packetstormsecurity.org/files/41550/md4coll.c.html`

УДК 004.056.55

РЕАЛИЗАЦИЯ НА ПЛИС ШИФРА FAPKC-4

Д. С. Ковалев

Данная работа является продолжением исследований автоматного шифра FAPKC (Finite Automata Public Key Cryptosystem) [1], которые были начаты в [2], в плане оценки эффективности его реализации на базе ПЛИС (Программируемая логическая интегральная схема). В работе сравниваются характеристики базового варианта FAPKC с последней модификацией FAPKC-4, стойкой к атаке, предложенной в [3]. Проведено также сравнение по быстродействию ПЛИС-реализации шифра FAPKC-4 с его программной реализацией на языках Perl и PHP.

В шифре FAPKC-4 используются обратимые с задержкой автоматы, т. е. автоматы, у которых входное слово восстанавливается по выходному с задержкой на несколько тактов работы, а также автоматы с конечной памятью, значение выходного символа которых зависит от конечного количества входных и выходных символов в предыдущие такты работы. Закрытый ключ состоит из двух обратимых нелинейных автоматов A и B (с небольшой задержкой τ_1 и τ_2 соответственно), обратные к которым могут быть построены с полиномиальной сложностью. Открытый ключ есть последовательная композиция автоматов закрытого ключа при известном начальном состоянии. При этом по выбранному состоянию композиции вычисляются начальные состояния автоматов закрытого ключа.

При шифровании к открытому тексту добавляются произвольные $\tau_1 + \tau_2$ символов. Шифртекст есть реакция автомата открытого ключа в выбранном начальном состоянии на «расширенное» входное слово. Таким образом, длина шифртекста увеличивается на $\tau_1 + \tau_2$ символов по сравнению с открытым текстом. При расшифровании сначала находится реакция β на зашифрованное слово автомата, обратного к B, в его начальном состоянии. Исходный открытый текст получается как реакция на входное слово β автомата, обратного к A, в его начальном состоянии, при этом начальные $\tau_1 + \tau_2$ символов отбрасываются.

При подписании сообщения δ сначала находится реакция γ автомата, обратного к B, в его начальном состоянии на «расширенное» на $\tau_1 + \tau_2$ символов входное слово δ. Подпись s под сообщением δ получается как реакция на входное слово γ автомата, обратного к A, в его начальном состоянии. При проверке подписи s подписанное сообщение δ сравнивается с реакцией x автомата открытого ключа в состоянии, равном первым $\tau_1 + \tau_2$ символам подписи s, на входное слово, образованное остальными символами подписи s.

Для изучения эффективности аппаратной реализации шифра FAPKC-4 на базе ПЛИС исследовано изменение количества используемых ресурсов и производительности ПЛИС по сравнению с такими же характеристиками базовой шифрсистемы FAPKC, а также проведено сравнение ПЛИС-реализации шифра FAPKC-4 с его программной реализацией.

На языке VHDL реализованы и в САПР Xilinx WebPack ISE на ПЛИС Spartan-3 XC3S1500 промоделированы следующие процедуры шифрсистемы FAPKC-4: шифрование, расшифрование, подписание и верификация цифровой подписи. Усложнение алгоритма существенно повлияло (в сторону уменьшения в 5–10 раз) только на максимальную рабочую частоту ПЛИС, а число используемых ресурсов ПЛИС практически не изменилось. При этом коэффициент эффективности ПЛИС-реализации FAPKC-4, как и у базового FAPKC, остался на порядок лучше, чем у RSA [4], т. е. использование шифра FAPKC-4 остаётся, как и ранее, предпочтительней (при этом для FAPKC-4 бралась задержка, обеспечивающая его стойкость, сравнимую со стойкостью RSA-1024). Шифртекст, получаемый в результате шифрования при величине задержки, обеспечивающей стойкость к атаке декомпозиции, длиннее открытого текста на 16 байт. Процедура получения цифровой подписи имеет характеристики, незначительно отличающиеся от аналогичных характеристик процедуры расшифрования, а верификация подписи требует значительно меньшее число ресурсов, чем шифрование, и работает приблизительно в пять раз быстрее.

Программно шифрсистема FAPKC-4 реализована на языках Perl и PHP и протестирована на компьютере с процессором Intel Core i5-2300, при этом PHP-реализация

имеет быстродействие в среднем в 1,34 раза меньше, чем ПЛИС-реализация на Virtex-5 XCVLX330T, а Perl-реализация в 2,6 раза медленнее PHP-реализации.

В целом, проведённые исследования показывают, что шифр FAPKC-4, как и базовый FAPKC, имеет коэффициент эффективности ПЛИС-реализации намного больше, чем RSA, но FAPKC-4 существенно медленнее FAPKC. Кроме того, реализация FAPKC-4 на ПЛИС имеет более высокое быстродействие по сравнению с программными реализациями этой шифрсистемы.

ЛИТЕРАТУРА

1. *Tao R. J.* Finite Automata and Application to Cryptography. Tsinghua University Press and Springer, 2008.
2. *Ковалев Д. С., Тренькаев В. Н.* Реализация на ПЛИС шифра FAPKC // Прикладная дискретная математика. Приложение. 2011. № 4. С. 33–34.
3. *Bao F. and Igarashi Y.* Break Finite Automata Public Key Cryptosystem // LNCS. 1995. V. 944. P. 147–158.
4. *Wollinger T., Guajardo J., and Paar C.* Cryptography on FPGAs: State of the art implementations and attacks // ACM Transactions on Embedded Computing Systems. 2004. V. 3. Iss. 3. P. 534–574.

УДК 519.6

КРИПТОГРАФИЧЕСКИЕ СВОЙСТВА РАЗВЕТВЛЕНИЙ ФУНКЦИЙ ВЕКТОРНЫХ ПРОСТРАНСТВ

К. Г. Когос, В. М. Фомичев

К основным задачам алгебраического анализа криптографических систем относятся разработка методов решения систем алгебраических уравнений (САУ) и оценка вычислительной сложности их решения. Один из приёмов решения состоит в том, что от нелинейной САУ можно перейти к решению нескольких линейных систем (СЛАУ) путем фиксации константами нескольких переменных. Этот подход достаточно детально рассмотрен в [1] на примере для разветвлений функций, реализуемых генератором «δ–τ шагов» и генератором с перемежающимся шагом. Вместе с тем не только линеаризация систем уравнений представляет интерес для анализа криптографических свойств. Более общий подход заключается в сведении САУ к системам уравнений определённого вида, левая часть которых задает преобразования, имеющие заданный признак [2, гл. 9].

Представим некоторые результаты исследования разветвлений криптографических функций векторных пространств над полем $\mathrm{GF}(2)$ на преобразования, имеющие заданный признак.

При натуральных n, m рассмотрим множество $\Phi_{n,m}$ всех отображений $V^n \to V^m$, $n > m$, $V = \{0, 1\}$. Требуется решить нелинейную систему булевых уравнений

$$g(x_1, \ldots, x_n) = a, \tag{1}$$

где $g \in \Phi_{n,m}$ и $a \in V^m$. Пусть функция $g(x_1, \ldots, x_n)$ задается системой координатных функций $\{g_1(x_1, \ldots, x_n), \ldots, g_m(x_1, \ldots, x_n)\}$. Определим умножение булева монома $\mu = x_{i_1} \cdot \ldots \cdot x_{i_p}$ степени p на отображение $g \in \Phi_{n,m}$:

$$\mu \cdot g(x_1, \ldots, x_n) = \{\mu \cdot g_1(x_1, \ldots, x_n), \ldots, \mu \cdot g_m(x_1, \ldots, x_n)\}.$$