3. *Lamberger M., Mendel F., Rechberger C., et al.* The Rebound Attack and Subspace Distinguishers: Application to Whirlpool. Cryptology ePrint archive, Report 2010/198, 2010. http://eprint.iacr.org/2010/198

УДК 519.7, 004.056.2, 004.056.53

ОЦЕНКИ СЛОЖНОСТИ ПОИСКА КОЛЛИЗИЙ ДЛЯ ХЭШ-ФУНКЦИИ RIPEMD

Г. А. Карпунин, Е. З. Ермолаева

Хэш-функция RIPEMD [1] была разработана в 1992 г. в рамках европейского проекта RIPE (RACE Integrity Primitives Evaluation) как альтернатива популярной на то время хэш-функции MD4 [2]. Фактически, функция сжатия RIPEMD представляет собой две работающие параллельно функции сжатия MD4 (левая и правая ветки RIPEMD), отличающиеся друг от друга аддитивными константами. Уже в 1997 г. Х. Доббертин [3] нашел коллизии для урезанной до двух раундов версии RIPEMD, а в 2001 г. К. Дебарт и Г. Гилберт [4] показали, что по отдельности и левая и правая ветки RIPEMD не устойчивы к коллизиям. Для полной версии RIPEMD коллизии были построены лишь в 2004 г. и предъявлены в знаменитой заметке К. Вонг и др. [5], чуть позднее те же авторы в [6] опубликовали детали своего алгоритма поиска коллизий и привели оценку средней трудоёмкости, которая является наилучшей на сегодняшний день. Однако корректность этой оценки вызывает сомнения в силу краткого и недетального изложения алгоритма.

К текущему моменту разработаны усиленные варианты хэш-функции RIPEMD: RIPEMD-128, RIPEMD-160, RIPEMD-256, RIPEMD-320 [7, 8], которые рекомендуются к использованию во многих международных и национальных стандартах, в частности ISO/IEC 10118-3:2004. Хэш-функции семейства RIPEMD-х получили широкое распространение и на практике, например RIPEMD-160 используется для генерации ключа шифрования на основе пароля в популярном программном комплексе создания шифрованных дисков TrueCrypt [9]. Поскольку все усиленные варианты наследуют идеологию первой конструкции RIPEMD, её подробный криптоанализ и получение точных оценок стойкости по-прежнему остается актуальным.

В настоящей работе восстанавливаются опущенные детали алгоритма [6] поиска коллизий для RIPEMD и проводится экспериментальная проверка заявленной в [6] трудоёмкости этого алгоритма. Такая необходимость возникает в силу того, что в [6] отсутствует полное обоснование оценок трудоёмкости, а приводятся лишь основные идеи алгоритма, которые состоят в следующем. Строится приводящая к коллизии дифференциальная характеристика $(\Delta M, \Delta Q)$, где ΔM — набор разностей между сообщениями, а ΔQ — набор разностей между промежуточными переменными сцепления. Выписывается некоторый набор достаточных условий на промежуточные переменные сцепления Q и неявно утверждается, что если для одного сообщения M все промежуточные значения переменных сцепления удовлетворяют этим условиям, то автоматически другое сообщение $M + \Delta M$ образует коллизию с M. Затем используются две техники для подбора такого сообщения M, что при вычислении его хэш-значения все переменные сцепления удовлетворяют набору достаточных условий. Первая техника называется однократной модификацией сообщения и позволяет добиться выполнения достаточных условий на первых 16 шагах функции сжатия. Сложность этого этапа, как неявно предполагают авторы [6], составляет от 1 до 4 условных операций, где за одну условную операцию принимается одно вычисление функции сжатия. Однако

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.