Список использованных источников

1. *Wang Y., Moulin P.,* «Perfectly Secure Steganography: Capacity, Error Exponents, and Code Constructions», IEEE Transactions on Information Forensics and Security, 54:6 (2008), 2706–2722.
2. *Harmsen J. J., Pearlman W. A.,* «Capacity of Steganographic Channels», IEEE Transactions on Information Theory, 55 (2009), 1775–1792.
3. *Filler T., Ker A. D., Fridrich J.,* «The Square Root Law of Steganographic Capacity for Markov Covers», Media Forensics and Security XI. Volume 7254 of Proc. SPIE, San Jose, 2009, 0801–0811.
4. *Харин Ю. С., Вечерко Е. В.* «Статистическое оценивание модели вкраплений в двоичную цепь Маркова», Дискретная математика, 25:2 (2013), 135–148.
5. *Пономарев К. И.,* «Параметрическая модель вкрапления и ее статистический анализ», Дискретная математика, 21:4 (2009), 148–157.
6. *Пономарев К. И.,* «Об одной статистической модели стеганографии», Дискретная математика, 21:2 (2009), 138–145.
7. *Харин, Ю. С., Вечерко, Е. В.* «Распознавание вкраплений в марковскую последовательность», Вопросы защиты информации, 107:4 (2014), 73-76.
8. *Ker A.,* «A capacity result for batch steganography», IEEE Signal Processing Letters, 14:8 (2007), 525–528

ПРИНЦИПЫ ПОСТРОЕНИЯ КРИПТОГРАФИЧЕСКИХ ПРОГРАММНЫХ БИБЛИОТЕК: БИБЛИОТЕКА BEE2

С. В. АГИЕВИЧ

Криптографические программные библиотеки должны не только точно и эффективно реализовывать криптографические алгоритмы и протоколы, но и соответствовать дополнительным принципам безопасности. В докладе рассматриваются принципы построения библиотеки Bee2, разработанной автором. На сегодняшний день в библиотеке реализованы алгоритмы и протоколы национальных криптографических стандартов (СТБ 34.101.31, 45, 47, 60, 66), а также алгоритмы электронной цифровой подписи, стандартизированные в Российской Федерации и Украине.

Переносимость. Библиотека написана на языке Си, без ассемблерных вставок и поэтому компилируется практически на любой аппаратно-программной платформе с помощью средств GCC. Расширения C99 не использованы, поэтому библиотека компилируется также в среде Visual Studio любой версии.

Контроль памяти. В низкоуровневые функции библиотеки могут передаваться указатели state и stack. Эти указатели ссылаются на память, в которой размещаются состояния алгоритмов/протоколов и могут размещаться локальные переменные. Память может содержать критические объекты (ключи), и поэтому взята под контроль. Функции объявляют о потребностях в памяти через дополнительные сервисные функции (они имеют суффиксы keep и deep). Высокоуровневые функции обрабатывают объявления и определяют общий объем памяти для состояния и стека. Интересно, что объем памяти оказывается довольно небольшим. Так, в текущей реализации достаточно сложного протокола BMQV, определенного в СТБ 34.101.66, на максимальном уровне стойкости $l = 256$ каждой стороне требуется не более 4096 байтов (одной страницы) памяти.

Защита памяти. Динамическая память выделяется только в высокоуровневых функциях и только одним блоком: в нем размещаются и состояние, и стек всех подчиненных функций. Максимальный отказ от динамической памяти снижает риск типичных для языка Си ошибок, повышает контроль над критическими данными. В определенных операционных системах блок памяти может защищаться от попадания в файл подкачки. При завершении работы с блоком его очистка выполняется так, что не может показаться бесполезной оптимизатору, и он ее не исключит из кода библиотеки.

Регуляризация. Если в криптографических программах имеются условные переходы, и условия переходов определяются обрабатываемыми данными (но не их раз-

мерностями), то эти программы подвержены атакам, основанным на замерах времени или питания. Даже если условных переходов нет, флуктуации времени выполнения могут быть индуцированы вариабельными задержками при загрузке данных из массивов в связи с логикой работы кэш-памяти современных процессоров.

Регулярные вычисления – это «равномерные» вычисления, схемно одинаковые при обработке любых данных определенной размерности. В Bee2 реализуется программа полной регуляризации. Критические нерегулярные функции, начиная с простейших функций сравнения буферов памяти, заменяются регулярными аналогами. Регулярные библиотеки, реализующие универсальные алгоритмы арифметики больших чисел, арифметики эллиптических кривых и пр., автору неизвестны. Библиотека Bee2 может быть одной из первых.

Предусловия и ожидания. Низкоуровневые (неэкспортируемые) функции Bee2 сделаны максимально простыми. Функции по возможности не проверяют входные данные и не возвращают коды ошибок, которые затем придется обрабатывать. Вместо этого функции объявляют о запрещенных ситуациях с помощью предусловий, обязанных соблюдаться при вызовах функций. Предусловия детально проверяются в отладочных версиях через директиву ASSERT. Предусловий довольно много, они создают предпосылки описания контрактов между функциями. Такие контракты могут служить основой для верификации программ.

Существуют предусловия (они названы ожиданиями), проверить которые вычислительно трудно: простота числа, неприводимость многочлена, корректность эллиптической кривой. Функции не могут полагаться на безусловное выполнение таких условий, поэтому корректно работают даже при их нарушении. Например, функция сложения точек эллиптической кривой над простым полем GF(p) корректно завершит сложение, даже если p – составное. Условия называются *ожиданиями*, для их описания используется директива EXPECT. Некоторые ожидания могут частично проверяться. Например, ожидание EXPECT (p – нечетное простое) может быть поддержано предусловием ASSERT(p – нечетное).

Интерфейсы. Высокоуровневые функции Bee2 организуют максимально полную проверку входных данных. Высокоуровневые функции также объявляют о своих ожиданиях и при их нарушении возвращают соответствующие коды ошибок.

Высокоуровневые функции объединяются в связки. Функции связки используют общее состояние и стек, они похожи на методы класса C++. В необходимых случаях объявляются ожидания относительно последовательности вызовов функций связки.

Высокоуровневые интерфейсы достаточно развитые. Например, интерфейс хэширования, как обычно, разрешает обработку данных фрагментами. Дополнительно, даже после обработки определенного числа фрагментов и формирования результирующего хэш-значения, обработку можно продолжить. Другими словами, поддержан полезный механизм hash-and-continue.

Случайные числа. Случайные числа формируются с помощью доступных источников случайности. В среде Linux обязательно используется системный файл dev/urandom, в среде Windows – сервис CryptGenRandom стандартного криптопровайдера. Дополнительно опрашивается генератор случайных чисел, иногда размещаемый на процессорах Intel, и аккумулируются разности между показаниями высокоточного таймера (регистра RDTSC) при приостановке текущего потока на 0 мс, т. е. при передаче управления ядру операционной системы с немедленным возвратом. Предусмотрена возможность использования любых дополнительных источников случайности.

Данные от источников объединяются, и по ним формируется ключ алгоритма генерации псевдослучайных чисел, определенного в СТБ 34.101.47. При каждом вызове алгоритма кроме ключа, состояния и монотонного счетчика обязательно обрабатываются данные от доступных источников случайности.

Алгоритмы. Большое внимание уделено выбору оптимальных арифметических алгоритмов. Были проведены достаточно подробные вычислительные эксперименты, а их результаты учтены в окончательных редакциях программ. Например, при создании кольца классов вычетов целых чисел по модулю m реализуется следующая основанная на экспериментах стратегия выбора функции редукции $x \to x \bmod m$: если m имеет вид $2^{nw} - c$, где w – длина машинного слова, $n \geq 2$ и $c < 2^w$, то используется редукция Крэндалла; иначе, если m – нечетное, – редукция Монтгомери; иначе, если $m > 2^{4w}$, – редукция Барретта, иначе – обычная редукция через деление уголком.

Разработано несколько новых алгоритмов арифметики больших чисел, например, алгоритм деления большого числа на машинное слово, в котором каждое деление машинных слов заменяется на два умножения. На многих процессорах этот алгоритм работает намного быстрее обычного.

В функциях работы с машинными словами активно применяются алгоритмические трюки из прекрасной книги [1].

Алгебраическая абстракция. Работа с алгебраическими структурами реализована через довольно общие, но достаточно насыщенные интерфейсы. Например, интерфейс qr описывает работу с абстрактным кольцом вычетом по модулю его идеала. Предусмотрены функции сложения, вычитания, аддитивного обращения, умножения, возведения в квадрат, мультипликативного обращения, деления, экспорта в строку октетов и импорта из такой строки. Реализация всех функций интерфейса не обязательна. В Bee2 интерфейс qr инстанциируется многими способами: zm – кольцо вычетов целых чисел, pp – кольцо многочленов, gfp – простое поле из $p > 2$ элементов, gf2 – поле характеристики 2. Арифметика эллиптических кривых описывается интерфейсов ec, который является надстройкой над qr.

Доверие. Главный фактор доверия к криптографической программе – открытые исходные тексты. Библиотека Bee2 является открытым программным обеспечением, распространяется на условиях GNU General Public License версии 3. Исходные тексты размещены по адресу https://github.com/agievich/bee2.

Список использованных источников

1. *Уоррен, Генри Мл.* Алгоритмические трюки для программистов / Генри Мл. Уоррен. – М.: Изд. дом «Вильямс», 2003.

МАЛОПАРАМЕТРИЧЕСКИЕ МАРКОВСКИЕ МОДЕЛИ ДЛЯ ОПИСАНИЯ СЛОЖНЫХ ЗАВИСИМОСТЕЙ В ПСЕВДОСЛУЧАЙНЫХ ПОСЛЕДОВАТЕЛЬНОСТЯХ

М. В. МАЛЬЦЕВ

Цепь Маркова условного порядка (ЦМУП) – математическая модель, разработанная в НИИ прикладных проблем математики и информатики БГУ для описания сложных зависимостей, возникающих в псевдослучайных последовательностях [1; 2]. Эта модель относится к классу так называемых малопараметрических марковских моделей, которые строятся на основе цепи Маркова высокого порядка. Число параметров, необходимое для задания матрицы вероятностей одношаговых переходов цепи Маркова порядка s с N состояниями, составляет $N^s(N–1)$. Данное обстоятельство ограничивает практическое использование этой универсальной модели сравнительно небольшими значениями s. В связи с этим для поиска зависимостей большой глубины в выходных последовательностях криптографических генераторов применяются малопараметрические модели [3], представляющие собой цепь Маркова порядка s, матрица вероятностей одношаговых переходов которой имеет специальный вид. Число параметров для таких моделей не возрастает экспоненциально