УДК 004.056

ЗЮЗЯ А.А.

СПОСОБ ПРОТИВОДЕЙСТВИЯ РЕКОНСТРУКЦИИ КЛЮЧЕЙ БЛОКОВЫХ АЛГОРИТМОВ ЗАЩИТЫ ИНФОРМАЦИИ АНАЛИЗОМ ДИНАМИКИ ПОТРЕБЛЯЕМОЙ МОЩНОСТИ

В статье рассмотрены методы реконструкции ключей криптоалгоритмов с помощью анализа динамики потребляемой мощности. Проведен анализ существующих программных контрмер для таких видов атак. Предложен новый способ противодействия анализу динамики потребляемой мощности, основанный на организации псевдопараллельной обработки блоков в режиме разделения времени. Способ отличается стохастическим переключением между обрабатываемыми блоками для обеспечения случайной временной локализации этапов обработки, что значительно снижает эффективность реконструкции ключа с помощью анализа динамики потребляемой мощности. Предложенный способ реализован для алгоритма ГОСТ 28.147-89. При этом падение производительности меньше, чем при использовании других способов противодействия анализу динамики потребляемой мощности.

The article studies power analysis key reconstruction methods for cryptographic algorithms. The analysis of existing software countermeasures for such types of attacks was done. A new power analysis obstruction method was proposed. It is based on pseudo parallel data processing in time sharing mode. The method is characterized by a stochastic switching between proceeding blocks of data for random time location of block processing stages. It significantly reduces effectiveness of the power analysis key reconstruction methods. Proposed method was implemented in software implementation of GOST 28.147-89 cryptographic algorithm. Decrease in performance of such modified implementation is less then via using other power analysis obstruction methods.

Введение

Рост интеграции информационных ресурсов на основе компьютерных технологий к началу 21-го века стал одним из решающих факторов поступательного развития всех сфер деятельности современного общества. Оборотной стороной расширения информационной интеграции является увеличение числа угроз нарушения конфиденциальности и безопасности данных. Для нейтрализации этих угроз широко используются системы защиты информации, в основе большинства из которых лежат криптографические алгоритмы. Современные алгоритмы криптографической защиты являются полностью открытыми (закрытым элементом является только ключ) и базируются на аналитически неразрешимых математических задачах. Соответственно, в большинстве случаев, для нарушения защиты данных необходимо узнать ключ.

Традиционно, для нарушения защиты – то есть реконструкции ключа используется перебор возможных его значений, при этом, для уменьшения объема перебора используются алгебраические свойства криптографических алгоритмов. Такие технологии нарушения защиты анализируют лишь математические аспекты алгоритма, абстрагируясь от вычислительной платформы, на которой реализуется система защиты данных.

В последнее десятилетие появились и активно развиваются новые технологии получения несанкционированного доступа к ключевой информации криптографических алгоритмов. Эти технологии базируются на получении информации о ключе криптографического алгоритма посредством анализа параметров технической реализации на конкретной вычислительной платформе. В литературе [1] такие технологии получения незаконного доступа к ключу алгоритмов защиты информации получили название атак по сторонним каналам. В отличии от традиционных методов, атаки по сторонним каналам учитывают особенности конкретной аппаратной реализации криптосистемы, а не только математического алгоритма, используемого в ней. Такое сужение специализации позволяет таким атакам достигать значительно более высокой эффективности при работе с конкретной криптографической системой [1].

Наиболее распространенной технологией реконструкции ключа алгоритма по параметрам его технической реализации является измерение и анализ динамики потребления мощности (АДПМ) вычислительным устройством, на котором выполняется алгоритм.

Для эффективного применения АДПМ важным является то, чтобы на устройстве реализовался только один вычислительный процесс. Это условие выполняется для микроконтроллеров, встроенных микропроцессоров и смарт-карт. Именно эти вычислительные устройства
являются основными объектами для применения методов реконструкции ключей криптографических алгоритмов с помощью АДПМ.

Учитывая, расширяющееся использование указанного класса вычислительных устройств в современных информационных технологиях, задача создания эффективных средств противодействия широкое несанкционированного доступа к ключам защиты посредством АДПМ является актуальной и важной.

Технологии реконструкции ключей на основе АДПМ и анализ методов противодействия

Впервые метод реконструкции ключей криптографических алгоритмов с использованием АДПМ был предложен Полом Кочером [2] и получил широкое распространение благодаря комбинации таких факторов, как высокая эффективность, низкая стоимость и невозможность обнаружения системой защиты информации того, что она стала объектом атаки (ввиду пассивности и неинвазивности). АДПМ базируется на различии в потребляемой мощности устройством при выполнении одних команд на различных данных. В основе большинства известных технологий реконструкции ключей криптографических алгоритмов с использованием АДПМ лежит модель утечки, называемая ”моделью Хеммингового веса” (Hamming weight model) [1]. Согласно этой модели, потребляемая мощность S (сигнал утечки) в момент обработки кода D определяется следующим образом:

$$S = \delta + \beta \cdot H(D) + \eta \qquad (1)$$

где δ – постоянное смещение, β – действительное число, $H(X)$ – Хеммингов вес кода X, η – шум с математическим ожиданием 0 и среднеквадратичным отклонением σ.

Согласно этой модели потребление мощности пропорционально числу единиц в обрабатываемых данных (Хэммингов вес).

Приведенная модель – наиболее простая. существуют и другие модели описания зависимости мгновенного значения потребляемой мощности от обрабатываемого кода. Так на практике часто используется “модель дифференциала Хеммингова веса”, согласно которой мгновенное значение потребляемой мощности пропорционально числу изменяемых при обработке разрядов кода. Ясно, что степень адекватности той или иной модели существенным образом зависит от используемой при изготовлении микроконтроллера интегральной технологии. Поэтому оправданным представляется создание модели утечки для каждого конкретного типа микроконтроллера.

На сегодняшний день существует ряд методов использования АДПМ для реконструкции ключей криптографических алгоритмов. Эти методы разделяются на две группы [3]:

- простой АДПМ (SPA – Simple Power Analysis) и его развитие – профильный АДПМ (Profiling Analysis) основаны не непосредственном сопоставлении динамики потребления мощности с командами обрабатываемого криптографического алгоритма; в большинстве случаев простой АПМД позволяет установить последовательность выполнения команд и если она зависит от битов ключевого кода – восстановить его.
- дифференциальный АДПМ или Д-АДПМ (DPA – Differential Power Analysis) и его развитие Д-АДПМ высоких порядков (HO-DPA – High Order Differential Power Analysis), основанные на статистическом анализе данных о динамике потребления мощности для ряда вычислений алгоритма при меняющихся данных и постоянном ключе.

Для противодействия методам реконструкции ключей криптографических алгоритмов с помощью АДПМ используют программные и аппаратные способы. Суть аппаратных способов заключается в том, чтобы сделать неотличимой разницу в потреб-

ляемой мощности для различных данных путем использования специальных схемных решений, таких как балансные схемы или шумящие схемные компоненты [2].

Существующие к настоящему времени программные средства противодействия реконструкции ключей АДПМ-технологиями основаны на использовании случайных и псевдослучайных элементов при реализации алгоритмов криптографической защиты. Базовыми критериями эффективности средств противодействия АДПМ-технологиями являются достигаемый ими уровень защищенности и объем дополнительно затрачиваемых вычислительных ресурсов. Последнее особенно важно с учетом того, что вычисления производятся на малоразрядных микроконтроллерах и смарт-картах, обладающих низкой производительностью и ограниченными ресурсами памяти. Для оценки достигаемого уровня защищенности от АДПМ-технологий в большинстве случаев используют упомянутые выше модели зависимости потребляемой мощности от обрабатываемого кода.

Основными направлениями при создании программных средств противодействия АДПМ-технологиям являются:

- маскирование – наложение на обрабатываемые данные и ключи случайных компонент перед обработкой и выполнение специальных вычислений для получения истинного результата – “снятие маски” после обработки. Маскирование применяют, преимущественно, для защиты ключей симметричных блоковых алгоритмов, таких как DES, Rijndael, ГОСТ 28.147-89 [5], в которых используются относительно простые операции. Существенным недостатком маскирования является возможность исключения влияния случайных масок путем специальной статистической обработки;
- случайное изменение порядка выполнения независимых по данным команд алгоритма (полиморфная реализация), что затрудняет сопоставление участков программы диаграмме потребляемой мощности. Полиморфная реализация алгоритма является эффективным средством противодействия простому АДПМ и наиболее часто используется для защиты ключей алгоритмов на основе мультипликативных операций модулярной арифметики – RSA, DSA, El-Gamal. Недостатком полиморфной реализации алгоритмов является необходимость значительных вычислительных ресурсах, требующихся для анализа зависимости по данным команд и реализации случайного их выполнения. Это существенно замедляет реализацию алгоритмов криптографической защиты данных на микроконтроллерах;
- разделение ключей, используемых в алгоритмах на несколько частей случайным образом с последующей раздельной их обработкой и компоновкой результата. Подход эффективен только для алгоритмов типа DES, IDEA, в которых используются только логические операции. Недостатком является также необходимость не менее, чем двукратном увеличении времени реализации криптографического алгоритма.
- случайное перемешивание выполнения операций над реальными и фиктивными данными; действенность подхода зависит от эффективности средств генерации фиктивных команд и данных ими обрабатываемых;
- использование сбалансированных кодов с постоянным числом единиц.

Указанные подходы могут использоваться как отдельно, так и комбинированно.

Эффективность этих способов в значительной степени зависит от вида АДПМ в используемом методе реконструкции ключей. Так, например, маскирование неэффективно против Д-АДПМ высоких порядков, если порядок анализа выше числа масок. Для обхода перемешивания используются интегральные модификации различных видов АДПМ. Общим недостатком рассматриваемых программных способов противодействия АДПМ-методам является снижение производительности криптографического алгоритма пропорционально числу масок при маскировании, либо фиктивных переменных при перемешивании.

Проведенный анализ показал, что наиболее действенным, с точки зрения достигаемого уровня защищенности от АДПМ, является динамичная модификация программной реализации криптографического алгоритма во время выполнения (полиморфизм кода). При этом последовательность команд для

реализации криптографического алгоритма каждый раз будет различной, как и динамика потребления мощности. Это позволяет противодействовать как профильному АДПМ, так и различным Д-АДПМ (в том числе высоких порядков). При этом, эффективность применения такого механизма определяется общим числом возможных модификаций кода и величиной разницы временной локализации этапов обработки данных для различных модификаций.

Эффективность полиморфной реализации существенно ограничивается зависимостью команд по данных в большинстве криптографических алгоритмов, а также от степени ветвления алгоритма. Очевидно, что наиболее эффективно использование полиморфизма для реализации маловетвящихся алгоритмов, к числу которых относятся блоковые алгоритмы шифрования, такие как ГОСТ 28.147-89, DES, IDEA, Rijndael.

Целью исследований является повышение эффективности защиты от АДПМ-технологий в рамках полиморфной реализации криптографических алгоритмов блочного типа за счет увеличения временного диапазона выполнения его операций.

Способ организации псевдопараллельной обработки блоков в криптографических алгоритмах

В качестве способа организации полиморфного исполнения для блочных криптографических алгоритмов предлагается использовать одновременную псевдопараллельную, в режиме разделения времени, обработку нескольких блоков данных.

Блочные алгоритмы состоят из k циклов обработки данных $R_i(D,K_i)$, $i=1..k$, в каждом из которых производится некоторое преобразование блока D данных с участием ключа K_i. В зависимости от типа блочного шифра цикл обработки может рассматриваться как некоторая макрооперация для сети Фейстеля либо последовательность макроопераций для SP-сети. В любом случае для преобразования блока данных выполняется конечная последовательность P таких макроопераций, длина K которой равна (сеть Фейстеля) либо кратна (SP-сеть) числу k циклов обработки данных.

Для противодействия АДПМ необходимо реализовать полиморфное исполнение алгоритма, при котором указанная последовательность макроопераций будет меняться при каждом выполнении алгоритма. Предлагается использовать помирфизм на двух уровнях: на уровне макроопераций и на уровне цикла обработки данных (псевдопараллельная обработка алгоритмом нескольких блоков данных).

Полиморфизм макроопераций блоковых алгоритмов шифрования основан на том, что длина блока существенно превышает разрядность обрабатываемого фрагмента. При этом, каждый из фрагментов может обрабатываться независимо. Так, в большинстве блочных алгоритмов нелинейное преобразование выполняется с использованием ряда табличных преобразователей (S-блоков), каждый из который преобразует относительно небольшое число разрядов кода. Так в DES используется 16 S-блоков, каждый из которых независимо обрабатывает 4 разряда. В силу наличия независимо обрабатываемых фрагментов существует возможность менять очередность их обработки, то есть организовать полиморфную обработку фрагментов. Выбор очередности обработки фрагментов может выполняться статически или динамически. В первом случае предварительно формируются и сохраняются в памяти ряд вариантов программной реализации цикла преобразования, отличающихся последовательностью обработки фрагментов. Выбор варианта выполняется случайно в начале каждого выполнения цикла криптографического преобразования. При динамическом полиморфизме программные фрагменты-заготовки имеют меньшую длину (вплоть до команды) и их выбор осуществляется непосредственно во время выполнения цикла преобразования

Полиморфная, на уровне макроопераций, реализация криптографического алгоритма потенциально уязвима для интегральной разновидности АДПМ в которой интервал интегрирования равен длине макрооперации. Следовательно для нейтрализации такого вида анализа необходимо обеспечить различную временную локализацию макроопераций из P на разных циклах обработки блоков данных – полиморфизм уровня цикла обработки данных. Для этого предлагается использовать псевдопараллельную организа-

цию обработки нескольких блоков данных в криптографическом алгоритме.

Возможность такой организации выполнения блоковых алгоритмов основана на том, что обработка блоков шифруемого (дешифруемого) сообщения выполняется независимо. Соответственно, сущность полиморфизма на уровне блоков сообщения состоит в том, что организуется одновременно-поочередная обработка нескольких блоков сообщения, причем выбор длина программного фрагмента обработки блока выбирается случайным образом. Псевдопараллельная организация подразумевает одновременную обработку m блоков данных на меньшем числе вычислительных устройств (ВУ) – чаще всего на одном в режиме разделения времени. При условии, что ВУ только одно, такая организация вычислений не приводит к повышению производительности (традиционная цель распараллеливания), однако позволит смешивать выполнение m последовательностей P_j, $j=1..m$ обработки различных блоков данных D_j, $j=1..m$.

При псевдопараллельном, в режиме разделения времени, исполнении последовательностей P_j, $j=1..m$ происходит их разделение на q подпоследовательностей $P_{j,l}$, $l=1..q$ различной длины, которые могут чередоваться различными способами с сохранением порядка расположения в P_j. Пример такого псевдопараллельного выполнения блоков P_1, P_2 и P_3 графически показан на рисунке 1. При полиморфном выполнении криптографического алгоритма разбиение на подпоследовательности и их чередование осуществляется стохастически.

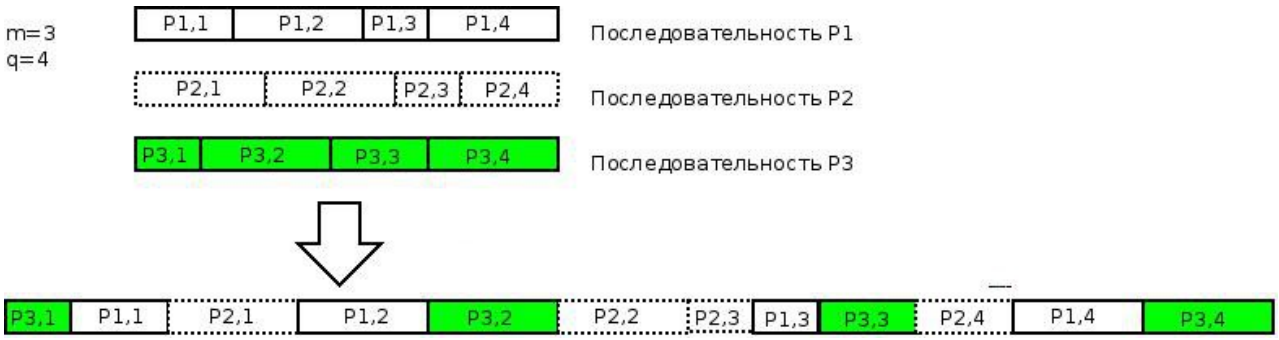

Рис.1. Пример чередования при выполнении трех последовательностей

Полиморфизм псевдопараллельной реализации обработки m блоков данных обеспечивается на этапах:

- случайного выбора длин q подпоследовательностей $P_{j,l}$ $l=1..q$ – при этом по случайному закону выбираются (q-1) длин подпоследовательностей, а последняя длина является дополнением суммы остальных длин до K;
- стохастического планирования выполнения подпоследовательностей $P_{j,l}j=1..m,l=1..q$ – во время работы алгоритма. Из m готовых к выполнению подпоследовательностей случайно выбирается одна, по окончании её исполнения ситуация повторяется; в конце обработки число готовых $P_{j,l}$ сокращается за счет завершения обработки все большего числа блоков данных; признаком окончания обработки всех m блоков данных является отсутствие готовых к выполнению подпоследовательностей.

Следует отметить, что оба этапа могут быть как статическими (выполняться перед обработкой данных) так и динамическими (во время обработки данных).

Для реализации псевдопараллельной обработки с разделением по времени необходимо обеспечить механизм сохранения/восстановления состояния ВУ при переключении между P_j. Реализация такого механизма требует дополнительных ресурсов, что снижает производительность алгоритма на величину пропорциональную числу переключений. Значительных временных затрат требует также программная генерация псевдослучайного кода. Исходя из этого, общее время t^m_{par} псевдопараллельной обработки m блоков данных определяется формулой :

$$t^m_{par} = m \cdot t^l + (m \cdot q - 1) \cdot (t_{state} + t_g) \qquad (2)$$

где t^1 – время последовательной обработки одного блока данных, t_{state} – время, затрачиваемое на сохранение/восстановление со-

стояния при переключении, t_g – время программной генерации псевдослучайного числа.

Таким образом, для уменьшения негативного влияния предложенной псевдопараллельной реализации на производительность необходимо использовать малые значения q и минимизировать объем данных при сохранении состояния. Это достигается за счет рационального выбора размеров макроопераций и их атомарного выполнения – переключение возможно только между макроопрерациями.

Вполне очевидно, что максимальное значение вариации Δt_m времени выполнения команды определяется формулой:

$$\Delta t_m = (m-1)\cdot t_{par}^{m-1} + \frac{(q-k)\cdot t^1}{q\cdot k} \qquad (3)$$

Среднее значение времени Δt_a вариации момента выполнения команды вычисляется следующим образом:

$$\Delta t_a = \frac{m\cdot t^1}{k}\cdot(\frac{q}{k}+1) \qquad (4)$$

Анализ формулы (3) показывает, что среднее время вариации момента выполнения команды при использовании предлагаемого способа выполнения криптографических алгоритмов увеличивается в m раз.

Важным фактором производительности полиморфной реализации алгоритмов криптографической защиты данных является способ получения случайных чисел. Учитывая технологические сложности измерения физически случайных величин, для большинства применений терминальных микроконтроллеров в качестве источника случайных чисел можно использовать встроенный таймер вместо случайных чисел применять превдослучайные числа.

Полиморфная реализация алгоритма ГОСТ 28.147-89

Предложенный способ организации псевдопараллельной обработки конкретизирован для модификации реализации сертифицированного на Украине алгоритма ГОСТ 28.147-89. В качестве базовой использована реализация указанного алгоритма блочного шифрования для 32-х битных процессоров архитектуры Intel x86, разработанная Винокуровым А.Ю.[4].

Объектом модификации является макрооперация – цикл обработки данных алгоритма ГОСТ 28.147-89, реализованный в процедуре _gost32. В этой процедуре элементами наиболее уязвимыми для анализа динамики потребления мощности являются 4 операции табличного преобразования (xlat) после наложения ключа на данные. В этих операциях данные, содержащие часть ключевой информации выставляются на шину, для обращения к памяти. Остальные операции (кроме наложения ключа) регистровые и потребление мощности при их выполнении существенно меньше, чем при работе с памятью. Следовательно, основным элементом при получении множества эквивалентных реализаций процедуры _gost32 должны быть именно эти операции. Поскольку операции табличного преобразования независимы по данным, порядок их выполнения может быть произвольным, что позволило получить 24 варианта реализации процедуры _gost32 на каждом из 32-х циклов.

Для псевдопараллельной обработки в режиме разделения времени использовались следующие параметры m=2, q=4·32=128. При этом вместо случайного планирования выполнения подпоследовательностей реализовано циклическое переключение со случайным выбором стартовой подпоследовательности. В процедуре _gost32 единственный свободный регистр EDI использован для хранения длин подпоследовательностей $P_{j,l}$ l=1..q=4,j=1..2. Поскольку общее число подпоследовательностей равно 8, то на хранение длины отводится 4 бита, что позволяет хранить значение до 15 (из 32 возможных). Объем данных которые необходимо сохранить при переключении последовательностей достаточно мал – текущий блок данных (EDX:EAX) и указатель на текущее смещение в развернутом ключе (ESI). Ввиду отсутствия свободных регистров сохранение ведется в стек. Таким образом, для сохранения и восстановления состояния при переключении достаточно 4 и 6 команд соответственно. Значение t_{state} для произведенной реализации равно:

$$t_{state} = 3\cdot t_{PUSH} + 3\cdot t_{POP} + t_{SUB} + t_{MOV} + \\ + t_{AND} + t_{JMP} = 10\cdot t_C \qquad (5)$$

где t_{PUSH} , t_{POP} , t_{SUB}, t_{MOV} , t_{AND} , t_{JMP} – время выполнения команд PUSH, POP, SUB, MOV, AND и JMP соответственно, а t_C – время выполнения цикла процессора Intel Pentium.

Для реализации псевдослучайного генератора случайных чисел, используемого для выбора обрабатываемого цикла и подпоследовательности в нем, наиболее целесообразным с точки зрения времени реализации представляется использование сдвигового регистра с линейной функцией обратной связи (LFSR – Linear Feedback Shift Register), выбираемой случайно из некоторого наперед заданного набора. Для генерации псевдослучайного числа требуется: вычислить значение линейной функции обратной связи (команда AND – для наложения маски обратной связи, 3 команды MOV и команда сдвига для копирования флага четности в флаг переноса) и сдвинуть регистр (одна команда ROL) – всего 6 команд.

При этом полный цикл обработки блока данных в реализации процедуры _gost32 равен: $t^1 = 1408t_C$. Численное значение времени t_{par}^2 выполнения 2-х блоков, вычисленное по формуле (2) составляет: $t_{par}^2 = 2 \cdot 1408t_C + 127 \cdot (10+6) \cdot t_C = = 4848 \cdot t_C$.

Таким образом, при псевдопараллельной обработке блоков в алгоритме ГОСТ 28.147-89 соотношение времени выполнения пары блоков – t_{par}^2 к времени $2 \cdot t^1$ их последовательного выполнения равно: $t_{par}^2/(2 \cdot t^1) = 4848/2816 = 1.721$.

Полученное значение подтверждается результатами экспериментальных исследований для m=2, q=128. При этом, определяющим фактором уменьшения производительности реализации криптографического алгоритма является величина q.

Выводы

Методы реконструкции секретных ключей криптографических алгоритмов посредством анализа динамики потребляемой мощности (АДПМ) представляют на сегодняшний день одну из наиболее серьезных угроз для средств защиты информации с одним исполняющим блоком (смарт-карты, встраиваемые системы и др.). Аппаратные способы противодействия АДПМ-технологиям требуют дорогостоящей специализированных средств. Использование традиционных программных средств противодействия имеет следствием деградацию производительности алгоритмов и малоэффективно против некоторых видов атак (например Д-АДПМ высоких порядков).

В качестве эффективной контрмеры АДПМ-технологиям несанкционированного доступа к ключам криптографических алгоритмов предложен способ их полиморфной реализации. Основной эффект противодействия достигается за счет того, что каждый раз криптографическое преобразование производится с помощью различной последовательности команд. Это препятствует сбору корректных образцов-профилей для профильного АДПМ и использованию статистических методов усреднения в различных Д-АДПМ. Отличительной особенностью предложенного способа является организация пседопараллельной криптографической обработки m блоков данных, что обеспечивает в m раз увеличение среднего значения вариации момента времени выполнения каждой команды алгоритма. Это значительно снижает эффективность существующих технологий реконструкции ключа с помощью анализа динамики потребляемой мощности.

Способ может быть эффективно использован для защиты ключей алгоритмов как симметричного шифрования: DES, Rijndael, ГОСТ 28.147-89, так и несимметричного типа RSA.

Предложенный способ конкретизирован в виде программной реализации стандартизированного в Украине алгоритма шифрования ГОСТ 28.147-89. Экспериментально доказано увеличение роста вариации момента времени выполнения команд алгоритма при снижении производительности в 1.7 раз, что меньше деградации производительности при использовании таких альтернативных способов противодействия АДПМ-технологиям, как маскирование и чередование [6].

Предложенный способ и его реализация для алгоритма ГОСТ 21147-89 могут быть использованы для повышения эффективности защиты информации на смарт-картах и встроенных терминальных устройствах компьютерных сетей.

Список литературы

1. Rivain M., Prouff E., Doget J. Higher-order masking and shuffling for software implementations of block ciphers // Cryptographic Hardware and Embedded Systems - CHES 2009. of Lecture Notes in Computer Science. - Springer, Heidelberg.- 2009.- Vol. 5747. - P. 171-188.
2. Kocher P., Jafft J., Jub B. Differential power analysis // Advances in Cryptology - CRYPTO '99 Lecture Notes in Computer Science. - Springer, 1999. - Vol. 1666 . - P. 388—397.
3. Chari S., Jutla C., Rao J., Rohatgi P. Towards Sound Approaches to Counteract Power-Analysis Attacks // Advances in Cryptology – CRYPTO ’99 Lecture Notes in Computer Science. - Springer, 1999. - Vol. 1666 . - P. 398–412.
4. Винокуров А. Алгоритм шифрования ГОСТ 28.147-89, его использование и реализация для компьютеров платформы Intel x86 [Электронный ресурс]. - Работа на правах рукописи, доступна на веб-сайте http://www.enlight.ru/crypto/articles/vinokurov/gost_i.htm
5. Марковский А.П., Абабне О.А., Ияд Мохд Маджид Ахмад Шахрури. Способ защиты ключей алгоритма ГОСТ 28.147-89 от реконструкции анализом динамики потребляемой мощности // Вісник національного технічного університету України ”КПІ”. Інформатика, управління та обчислювальна техніка. – 2007.- № 46.- С.128-138.

Поступила в редакцию 18.12.2009