УДК 004.052.42

МАРКОВСКИЙ А.П.,
МУХАММАД МЕФЛЕХ АЛИСА АБАБНЕ.,
ЗЮЗЯ А.А.,
ГАРАЗД В.М.

К ПРОБЛЕМЕ ЗАЩИТЫ ОПЕРАНДОВ МОДУЛЯРНОГО ЭКСПОНЕНЦИРОВАНИЯ ОТ ИХ РЕКОНСТРУКЦИИ АНАЛИЗОМ ДИНАМИКИ ПОТРЕБЛЕНИЯ МОЩНОСТИ

Целью представленных в статье исследований является анализ опасности реконструкции операндов модулярного экспоненцирования анализом динамики потребления мощности и разработка способов противодействия. Показано, что степень модулярного экспоненцирования, являющаяся ключом алгоритмов RSA, El-Gamal, DSA может быть реконструирована временным анализом потребляемой мощности. Для противодействия разработан специальный алгоритм модулярного экспоненцирования. Алгоритм не имеет условных операторов и включает ложные операции, затрудняющие временной анализ мощности. Применение предложенного подхода увеличивает на 25% время выполнения модулярного экспоненцирования.

The goal of presented by article research is to point out the potential vulnerabilities of modular exponentiation operands reconstruction by power dynamic analysis and to elaborate countermeasures. It has been shown that exponent of modular exponentiation which is secret key of RSA, El-Gamal and DSA can be reconstruction by timing power analysis. For countermeasure the special algorithm for modular exponenttiation has been worked out. Proposed algorithm does not conditional operators use and include the false operators which inhibit to timing power analysis. It has been shown that implementation of proposed approach demand about 25% more time for modular exponentiation.

Введение

Динамическое углубление информационной интеграции на основе компьютерных сетей играет доминирующую роль на современном этапе развития большинства сфер человеческой деятельности. Важным условием расширения информационной интеграции является решение проблемы эффективной защиты данных и размежевание прав доступа к информации в компьютерных системах и сетях. Расширение использования компьютерных и сетевых технологий в сфере финансовой деятельности, для управления сложными техническими системами, связанными с техногенным риском требует обеспечения высокой надежности защиты информации в компьютерных системах управления и сетях. Таким образом, существует объективная необходимость повышения эффективности средств защиты информации с точки зрения расширения сфер использования

интегрированных систем. С другой стороны, объективно существует ряд факторов, снижающих надежность средств защиты данных. В последние годы, одной из наиболее потенциально опасных угроз информационной безопасности в системах компьютерного управления является несанкционированный доступ к данным посредством измерения и анализа динамики изменения параметров работы вычислительного устройства во время реализации им программ [1]. Особенно актуальной является проблема защиты данных от их реконструкции посредством измерения и анализа динамики изменения потребляемой мощности. В современных компьютерных сетях значительную часть терминальных устройств составляют микроконтроллеры, для которых измерение потребляемой мощности и сопоставление ее с выполняемыми командами осуществляется относительно просто [1]. Вместе с тем, такие терминальные устройства поддерживают сетевые протоколы защиты информации, при реализации которых используются секретные данные – ключи и пароли. Современные технологии позволяют выполнять статистическую реконструкцию этих секретных данных на основе анализа динамики потребляемой мощности [2]. Анализ таких технологий свидетельствует о том, что они наиболее эффективны для доступа к постоянным компонентам систем защиты информации – ключам и паролям. Это особенно опасно для механизмов защиты данных, ключи которых редко меняются. К таким механизмам, в частности, относятся алгоритмы с открытым ключом, наиболее распространенными из которых является RSA, El-Gamal, DSA. Изменение ключей при каждом сеансе работы этого алгоритма принципиально нарушает концепцию открытости ключей. Поэтому ключи упомянутых алгоритмов наиболее уязвимы для несанкционированного доступа к ним путем анализа динамики потребления мощности. Вместе с тем, эти алгоритмы лежат в основе сетевых протоколов информационной безопасности, поэтому потеря контроля над их ключами грозит серьезными нарушениями в работе систем защиты информации в сетях.

Таким образом, проблема защиты от реконструкции анализом динамики потребления мощности ключей алгоритмов, в основе которых лежит модулярное экспоненцирование является важной и актуальной на современном этапе развития технологии защиты данных в сетях.

Анализ проблемы защиты данных от реконструкции, осуществляемой анализом динамики потребляемой мощности

Базовой вычислительной операцией при программной реализации большей части алгоритмов защиты информации с открытым ключом, в том числе RSA, является модулярное экспоненцирование $X^E \bmod M$, причем X, $E < M$. Количество n двоичных разрядов чисел X, E и M существенно превышает разрядность k процессора. Каждое из чисел X, E и M, можно представить в виде s слов, состоящих из k двоичных разрядов

($s=n/k$), каждое из которых лежит в интервале от 0 до 2^k-1: $X=\{x_{s-1}, \ldots, x_1, x_0\}$, $E=\{e_{s-1},\ldots,e_1,e_0\}$ и $M=\{m_{s-1},\ldots,m_1,m_0\}$, $\forall j\in\{0,...,s-1\}$, $0\le x_j,e_j,m_j\le 2^k-1$. Для указания двоичных разрядов слов X, E и M используются обозначения с двумя индексами, например $e_0=\{e_{0,k-1},e_{0,k-2},\ldots,e_{0,1},e_{0,0}\}$; $\forall l\in\{0,...,k-1\}$: $x_{j,l}$, $e_{j,l}$, $m_{j,l}\in\{0,1\}$:

$$X=\sum_{j=0}^{s-1}x_j\cdot 2^{j\cdot k}=\sum_{j=0}^{s-1}\sum_{l=0}^{k-1}x_{j,l}\cdot 2^{j\cdot(k-1)+l}, E=\sum_{j=0}^{s-1}e_j\cdot 2^{j\cdot k}=\sum_{j=0}^{s-1}\sum_{l=0}^{k-1}e_{j,l}\cdot 2^{j\cdot(k-1)+l},$$

$$M=\sum_{j=0}^{s-1}m_j\cdot 2^{j\cdot k}=\sum_{j=0}^{s-1}\sum_{l=0}^{k-1}m_{j,l}\cdot 2^{j\cdot(k-1)+l}$$

Процесс модулярного экспоненцирования сводится к последовательному выполнению $\log_2 E=n$ циклов, в каждом из которых осуществляется операция возведения в квадрат полученного на предшествующем цикле результата и, дополнительно, в зависимости от текущего бита степени E, выполняется операция умножения. Исходя из порядка, в котором анализируются разряды степени E, существует две разновидности модулярного экспоненцирования: справа – налево и слева – направо.

Алгоритм двоичного экспоненцирования справа – налево.

1. $A = 1$; $S = X$;
2. for (j=0; j< s ; j++)
   2.1. for (l=0; l< k; l++)
   {
      2.1.1. if ($e_{jl} = 1$) $A = A\cdot S$ mod M;
      2.1.2. $S = S\cdot S$ mod M;
   }

Результат $A=X^E$ mod M.

Для вычисления весов используется возведение в квадрат, а для формирования результата – умножение. Среднее число операций умножения – $1.5\cdot n$.

На практике, большее распространение получили алгоритмы, которые предполагают анализ разрядов степени E начиная со старших разрядов (слева-направо): в этом случае множитель при выполнении операции умножения представляет собой постоянной число, равное X, что создает потенциальные предпосылки для повышения скорости умножения.

1. $A = 1$;
2. for (j= s-1; j >=0; j --)
   2.1. for (l=k-1; l>=0; l--)
   {
      2.1.1. $A = A \cdot A$;
      2.1.2. if ($e_{j,l} = 1$) $A = A\cdot X$;
   }

Результат $A=X^E$ mod M.

При использовании алгоритмов, основанных на модулярном экспоненцировании, в качестве секретного ключа выступает код экспоненты *E,* который практически не меняется. Соответственно, основной задачей нарушения защиты является получение кода *E*. В качестве побочной задачи выступает задача доступа к значению кода *X* или результату - $A=X^E \bmod M$ при неизвестном, но постоянном *E* [2] .

Как уже отмечалось, значительную часть терминальных устройств сетей составляют встроенные микроконтроллеры и микропроцессоры, используемые для сбора данных или управления технологическим оборудованием. К 2005 году доля таких терминальных устройств составляла до 60% [2], причем имеет место тенденция ее роста с развитием компьютерных и сетевых технологий, а также с расширением сфер их применения.

Этот класс терминальных устройств поддерживает протоколы защиты информации в сетях, в процессе реализации которых использует постоянные секретные параметры – пароли, ключи криптографических алгоритмов. Для микроконтроллеров и микропроцессоров относительно просто измерить во время выполнения программ динамику потребляемой мощности и сопоставить ее с выполняемыми командами. Все процессорные устройства вычислительной техники построены на логических вентилях, основной частью которых являются транзисторы. Токи, протекающие через транзисторы, зависят от их рабочих точек и меняются скачкообразно при переключении вентилей. При этом, величина потребляемого тока зависит от значения битов данных, которые обрабатываются в текущий момент времени [3]. Для измерения тока, потребляемого вычислительным устройством, в его цепь питания добавляется резистор, напряжение на котором измеряется высокоточным цифровым микровольтметром [3]. Исследовательские цифровые микровольтметры позволяют производить измерения с частотой дискретизации 1-2 ГГц. и точностью до 1мкВ. Промышленные общедоступные образцы обеспечивают частоту дискретизации 50 МГц и точность 5мВ.; Большинство микроконтроллеров, встроенных микропроцессоров и смарт-карт работает с тактовой частотой 20-50 МГц.

С помощью описанного оборудования можно выполнить привязку команд выполняемой программы (для вычислительных устройств рассматриваемого класса они практически не меняются) к диаграмме изменения потребляемой мощности. К настоящему времени известно две технологии реконструкции данных по результатам измерения динамики потребляемой мощности. Первая из этих технологий получила название SPA (Simple Power Analysis) - простого анализа мощности [4]. Сущность этой технологии состоит в непосредственной интерпретации измерений потребляемой мощности во время выполнения известной программы.

На рис.1. показана диаграмма измерений мощности при выполнении модулярного экспоненцирования, предусмотренного алгоритмом RSA на смарт-карте [2]. На диаграмме отчетливо видны циклы умножения и возведения в квадрат. Динамика потребляемой мощности зависит от выполняемой команды и данных, над которыми эта команда выполняется.

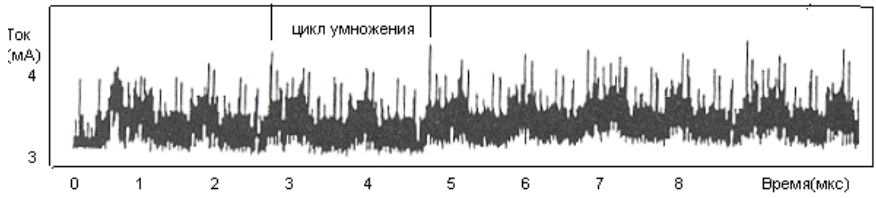

Рис.1. Фрагмент диаграммы потребляемой мощности при модулярном экспоненцировании на смарт-карте.

С использованием технологий SPA достаточно точно можно установить последовательность выполнения команд. Поскольку каждая из команд занимает определенное время, технология SPA в определенном смысле смыкается с временным анализом выполнения команд программы [1].

Другой технологией использования измерений динамики потребляемой мощности для реконструкции обрабатываемых данных является дифференциальный анализ мощности (DPA-Differential Power Analysis). Эта технология также описана впервые в 1999 году [5] и ориентирована на реконструкцию постоянно используемых в программах данных путем статистической обработки большого числа диаграмм потребления мощности при выполнении одной и той же программы. В частности, технология DPA может быть эффективной при анализе большого числа диаграмм потребления мощности при выполнении одной и той же программы, реализующей криптографический алгоритм, которая обрабатывает разные данные, но используемые ключи остаются неизменными. Эта ситуация является типичной при использовании микроконтроллеров и смарт-карт в качестве терминальных устройств компьютерных сетей.

Судя по результатам выполненного анализа публикаций [2], касающихся технологий реконструкции данных с использованием технологий SPA и DPA, в настоящее время не существует математической модели влияния битов обрабатываемой информации на потребляемую мощность. Но, это влияние имеет место и может быть выявлено статистическими методами, на которых построена идея DPA.

В первом приближении можно говорить о том, что технология SPA эффективна для реконструкции данных, от которых зависит порядок выполнения команд, в то время, как DPA позволяет реконструировать данные, непосредственно используемые в вычислениях.

Исходя из этого, для реконструкции экспоненты E наибольшую опасность представляет SPA, поскольку, как следует из приведенных выше алгоритмов, она прямо не участвует в вычислениях, но определяет порядок выполнения команд. Проанализируем потенциальную опасность применения SPA для получения значений разрядов экспоненты E.

Одним из узловых вопросов такого анализа является оценка возможности различения операций возведения в квадрат и умножения. Полученные в работе [2] результаты позволяют положительно оценивать такую возможность даже при условии, что для обеих операций используется один и тот же алгоритм умножения. Исходя из этого, значения всех разрядов E могут быть достаточно просто восстановлены путем отслеживания по диаграмме участков, соответствующих пп.2.1.1 (для экспоненцирования справа-налево) или пп.2.1.2. (для экспоненцирования слева-направо).

Вторая опасность применения SPA состоит в возможности отслеживания по диаграмме изменения последовательности команд в зависимости от условного перехода, выполняемого по результату анализа текущего бита экспоненты E. Упомянутая опасность не зависит от того, различимы ли по диаграмме операции возведения в квадрат и умножение. Пользуясь такой технологией, реконструкция E может быть полнена достаточно просто.

Технология DPA представляет опасность при решении задачи реконструкции кода X путем анализа диаграммы при неизвестном значении экспоненты E. Очевидно, что такая возможность открывается при использовании для модулярного экспоненцирования алгоритма слева-направо, поскольку при реализации пп. 2.1.2. участвует постоянный код X. Этот код потенциально может быть восстановлен путем статистического анализа диаграмм мощности при выполнении соответствующих операций. Следовательно, с точки зрения защиты операндов модулярного экспоненцирования от реконструкции с использованием DPA, предпочтительным является алгоритм справа-налево, в котором нет постоянных операндов.

Таким образом, проведенный анализ показал, что базовые алгоритмы модулярного экспоненцирования не обеспечивают защиты от реконструкции кода экспоненты E с использованием SPA.

Обзор способов противодействия реконструкции закрытых данных, осуществляемой анализом динамики потребляемой мощности

К настоящему времени для противодействия реконструкции данных посредством анализа динамики потребления мощности предложено ряд решений [1-6], которые укладываются в рамках следующих направлений:

- маскирование всех операндов, зависящих от постоянных ключей, случайными кодами, меняющимися при каждом выполнении алгоритма;

- случайные перестановки операций, не нарушающие результата работы алгоритма;
- вставка случайных команд, не влияющих на результат и затрудняющих "привязку" диаграммы потребления мощности к операциям алгоритма.

Маскирование операндов является наиболее эффективным способом противодействия реконструкции постоянных ключей путем анализа динамики потребляемой мощности [6].

Применительно к маскированию ключа RSA -экспоненты E наиболее известный подход [1] состоит в том, что в качестве аддитивной маски E используется произведение случайного числа r на $\phi(M)=(p-1)\cdot(q-1)$, причем $p\cdot q=M$. Маскированная экспонента E' может быть, таким образом, представлена в виде: $E' = E + r\cdot\phi(M)$. Поскольку для любого X выполняется $X^{r\cdot\phi(M)} \bmod M = 1$, то $X^{E'} \bmod M = X^E \bmod M$. Это означает, что для снятия маски нет необходимости в специальных операциях. Недостатком изложенного способа маскирования является то, что наложение аддитивной маски $r\cdot\phi(M)$ существенно увеличивает разрядность n' экспоненты E'. При известном значении модуля M, код $\phi(M)$ достаточно просто определить. Если не использовать в комбинировании с маскированием специальных средств защиты от SPA, то значения n' и E', как показано в предыдущем разделе, могут быть реконструированы простым анализом диаграммы потребления мощности. Учитывая, что $r\cdot\phi(M) >> E$, приближенное значение r' можно определить, как частное $r' \approx E'/\phi(M)$. В силу того, что величины E и $\phi(M)$ являются величинами одного порядка (n-разрядными двоичными числами), то истинное значение $r \le r'$ и отличается от r' на единицы (при близких значениях E и $\phi(M)$, наиболее вероятно, что $r = r'-1$). Это означает, что можно, в принципе, восстановить значение r путем небольшого подбора, а значит и реконструировать значение E.

Следовательно, без применения специальных мер противодействия SPA, маскирование экспоненты E не позволяет исключить возможность реконструкции ключа RES.

Для противодействия SPA наиболее широко используется случайная перестановка операций с использованием комбинирования двух базовых алгоритмов модулярного экспоненцирования [2]. Для этого выбирается случайная, лежащая в интервале от 0 до s-1 стартовая точка $d \in \{0,\ldots,s-1\}$ и осуществляется модулярное экспоненцирование справа-налево от 0 до точки d ,с фиксацией результата в переменной A и степени в переменной S. Затем, имея в качестве начальных полученные значения A и S выполняется модулярное экспоненцирование слева-направо от s-1 до d+1. При использовании такого способа, по мнению его авторов [2] при проведении SPA сложно установить порядок следования битов экспоненты. Однако, следует иметь ввиду, что число вариантов стартовой точки d очень неве-

лико (не более 10^3) и все эти варианты можно перебрать, если знать X и результат $X^E \bmod M$. На практике, при постоянной экспоненте E, имея в распоряжении диаграммы потребления мощности при реализации нескольких операций модулярного экспоненцирования достаточно просто реконструировать E.

Другим способом противодействия SPA является перестановка операций возведения в квадрат и умножения. Этот способ предполагает в качестве алгоритмической основы экспоненцирование справа-налево. Сущность способа состоит в том, что вначале выполняются все возведения в квадрат с сохранением в памяти тех результатов (значений S), которые участвуют в операциях умножения (пп. 2.1.1. соответствующего алгоритма). Затем из памяти извлекаются все сохраненные квадраты, участвующие в умножении и подряд выполняются все операции умножения [1]. Ниже представлена соответствующая модификация алгоритма справа – налево.

1. $A = 1$; $S = X$; $h = 0$;
2. for (j=0; j< s ; j++)
   2.1. for (l=0; l< k; l++)
   { 2.1.1. if ($e_{jl} = 1$) { 2.1.1.1. $Z[h] = S$; 2.1.1.2. h ++ ; }
   2.1.2. $S = S \cdot S \bmod M$; }
3. for (q =0; q<h; q++) $A = A \cdot Z[q] \bmod M$;

Результат $A = X^E \bmod M$.

Недостатком такого способа является то, что с использованием технологии SPA можно зафиксировать проверку условия, осуществляемого в пп.2.1.1. Фрагменты диаграммы потребления мощности при выполнении команд пп. 2.1.1.1. и 2.1.1.2. могут быть достаточно просто выделены.

Проведенный краткий обзор существующих способов противодействия SPA, показал, что они не обеспечивают надежной защиты от реконструкции экспоненты E анализом динамики потребления мощности.

Целью работы является создание способов защиты от восстановления кода экспоненты E по результатам анализа динамики мощности, потребляемой вычислительным устройством в процессе выполнения модулярного экспоненцирования.

Организация модулярного экспоненцирования, затрудняющая реконструкцию экспоненты временным анализом потребляемой мощности

Приведенный анализ известных способов противодействия реконструкции экспоненты, осуществляемой с использованием SPA, показал, что узловой проблемой является исключение возможности отслеживания средствами SPA условных операторов, выполняющих последовательное тестирование разрядов E. Для защиты от SPA необходимо модифицировать алгоритм модулярного экспоненцирования таким образом, чтобы

исключить из него условные операторы. В качестве основы модифицированного алгоритма выбрано модулярное экспоненцирование справаналево, поскольку оно не использует постоянных элементов и, соответственно, более устойчиво к DPA. Ниже представлена предлагаемая модификация алгоритма модулярного экспоненцирования:

1. $A[0] = 1$; $A[1]=1$; $S = X$; $y = 1$;
2. for (j=0; j< s ; j++)
   2.1. for (l=0; l< k; l++)
   {
   2.1.1. $q = e_j \& y$;
   2.1.2. $e_j = e_j >> 1$;
   2.1.3. $A[q] = S \cdot A[1] \bmod M$;
   2.1.4. $S = S \cdot S \bmod M$;
   }

Результат $A[1] = X^E \bmod M$.

В предлагаемой модификации отсутствуют условные операторы, на каждом цикле выполняется одна операция возведения в квадрат и умножения, что не позволяет средствами SPA определить значение обрабатываемого в этом цикле разряда экспоненты. Исключение условных операторов выполнено за счет введения паразитной операции умножения, результат которого фиксируется в $A[0]$, а накопление правильного результата осуществляется в $A[1]$. Вполне очевидно, что за счет введения паразитной операции умножения, вычислительная сложность модулярного экспоненцирования возрастает на 25%.

В качестве второго, эффективного способа противодействия SPA-реконструкции экспоненты E предлагается выполнение модулярного экспоненцирования с использованием аддитивных цепочек.

Аддитивной цепочкой V длиной L для положительного целого числа E называется последовательность $u_0,u_1,\ldots,u_s$ положительных чисел и связанная с ними последовательность $w_1,w_2,\ldots,w_s$ пар $w_i=<i_1,i_2>$, $0 \le i_1,i_2 < i$, которые обладают следующими свойствами:

1. $u_0 = 1$ и $u_L = E$;
2. $\forall\, u_i$, $1 \le i \le L$, $u_i = u_{i_1} + u_{i_2}$

Алгоритм модулярного экспоненцирования $X^E \bmod M$ на основе аддитивных цепочек имеет следующий вид:

1. $G[0] = X$;
2. for(j=1; j<=L; j++) $G[j] = G[j_1] \cdot G[j_2] \bmod M$;

Результат: $G[L] = X^E \bmod M$.

Например, для E=12, можно сформировать аддитивную цепочку в виде множества чисел {1,2,3,6,12}, которым соответствуют пары: w_1=<0,0>, w_2=<0,1>, w_3=<2,2>, w_4=<3,3>. Для той же экспоненты E=12, можно сформировать другую аддитивную цепочку, которая задается множеством

чисел {1,2,4,5,10,12}, которым соответствуют пары: w_1=<0,0>, w_2=<1,1>, w_3=<0,2>, w_4=<3,3>, w_5=<1,4>. В соответствии с приведенным выше алгоритмом, вычисление X^{12} с использованием последней цепочки выполняется в следующей последовательности: $G[0]=X$;

$G[1]=G[0]\cdot G[0] \bmod M = X^2 \bmod M$;

$G[2]=G[1]\cdot G[1] \bmod M = X^4 \bmod M$;

$G[3]=G[0]\cdot G[2] \bmod M = X^5 \bmod M$;

$G[4]=G[3]\cdot G[3] \bmod M = X^{10} \bmod M$;

$G[5]=G[1]\cdot G[4] \bmod M = X^{12} \bmod M$.

Поскольку приведенный алгоритм не содержит условных операторов, то временной анализ динамики потребления мощности при его реализации практически не позволит восстановить значение E, даже, если удастся отличить операции умножения и возведения в квадрат. Поскольку для используемой на практике разрядности n экспоненты (порядка 1024) вариантов ее разложения на аддитивные цепочки существует очень много, то восстановить конкретно используемую цепочку не представляется возможным по результатам анализа динамики потребляемой мощности.

Эффективность использования аддитивных цепочек при модулярном экспоненцировании, как средства противодействия SPA, определяется тем, что на практике экспонента E, являясь закрытым ключом RSA абонента не меняется. Это позволяет сгенерировать, с использованием случайных чисел, аддитивную цепочку, в соответствии с которой скомпилировать программу, которая будет непосредственно выполняться.

Недостатком использования аддитивных цепочек является необходимость в дополнительном объеме памяти для массива G и хранения пар w, которые задают связи компонент цепочек. При этом требуемый объем памяти определяется длиной L цепочки. Если считать, что среднее количество единиц в n-разрядной экспоненте E равно $n/2$, то нижняя граница значения длины L цепочки составляет $n + \log_2 n - 2$. Для реализации приведенного выше алгоритма требуется хранение относительно небольшого числа элементов массива G: в принципе можно выполнить алгоритм при хранении всего 2-х чисел указанного массива. Для хранения пар w, требуется объем V_w памяти, определяемый формулой :

$$V_w = 2 \cdot (n + \log_2 n - 2) \cdot \lfloor \log_2 (n + \log_2 n - 2) \rfloor \qquad (1)$$

Например, для наиболее часто встречающегося в практике применения RSA значения n=1024, V_w = 2838 байт, что не превышает объем встроенной памяти большинства современных встроенных микроконтроллеров.

Выводы

Проведенные исследования, направленные на разработку способов противодействия восстановлению экспоненты, осуществляемого путем измерения и анализа динамики потребляемой вычислительной платформой

мощности при модулярном экспоненцировании позволяют сделать выводы:

1. Операция модулярного экспоненцирования лежат в основе ряда криптографических алгоритмов, таких как RSA, El-Gamal, DSA, которые широко используются в протоколах защиты информации в сетях. Экспонента указанной операции является частью закрытого ключа этих алгоритмов, который меняется весьма редко.

2. Наибольшую опасность для реконструкции экспоненты представляет временной анализ потребления мощности вычислительной платформой при выполнении на ней операции модулярного экспоненцирования.

3. Эффективным способом противодействия реконструкции экспоненты временным анализом потребления мощности является исключение условных операторов. Предложен алгоритм, удовлетворяющий этому требованию, включающий паразитные операции, которые затрудняют эффективный анализ динамики потребления мощности. Реализация предложенного алгоритма требует на 25% больше времени по сравнению с обычным модулярным экспоненцирования.

4. Предложен подход к решению задачи противодействия временному анализу потребления мощности для реконструкции секретного кода экспоненты. Подход основан на использовании технологии модулярного экспоненцирования на основе аддитивных цепочек.

Предложенные способы противодействия доступа к закрытым ключам алгоритмов защиты информации, в основе которых лежит операция модулярного экспоненцирования, могут быть эффективно использованы для повышения информационной безопасности компьютерных сетей.

Список использованной литературы

1. Cohher P. Timing Attack on Implementations of Diffie-Hellman, RSA, DSS and other systems.// Proceeding of Advances in Cryptology-“CRYPTO-96”. LNCS-1882. Springer-Verlag.- 1996, PP.104-113.
2. Messerges T.S., Dabbish E.A., Sloan R.H. Power Analysis Attacks of Modular Exponentiation in Smartcards // Proceeding of 1-th International Workshop “Cryptographic Hardware and Embedded Systems”(CHES-1999), LNCS-1717. Springer-Verlag.- 1999.- P. 145-157.
3. Akkar M.L., Bevan C., Dischamp P., Moyart D. Power analysis, what is now possible // Proceeding of International Workshop “Asiacrypt-2000”. LNCS-1976. Springer-Verlag.- 2000.- P. 489-502.
4. Mayer-Sommer R., Smartly analyzing the simplicity and power of simple power analysis on smartcards // Proceeding of 2-th International Workshop “Cryptographic Hardware and Embedded Systems”(CHES-2000), LNCS-1965. Springer-Verlag.- 2000.- P. 78-92.
5. Kocher P., Jaffe J., Jun B. Differential Power Analysis // Proceeding of CRYPTO’99 .-Springer-Verlag.-1999.- PP.388-404.
6. Марковский А.П., Абабне О.А., Ияд Мохд Маджид Ахмад Шахрури. Способ защиты ключей алгоритма ГОСТ 28.147-89 от реконструкции анализом дина-

мики потребляемой мощности //Вісник національного технічного університету України ”КПІ”. Інформатика, управління та обчислювальна техніка. – 2007.- № 46.- С.128-138.