Р. Г. Коркикян, Е. Ю. Родионов, А. В. Мамаев

ТЕХНОЛОГИИ АНАЛИЗА УЯЗВИМОСТЕЙ АППАРАТНЫХ И ПРОГРАММНО-АППАРАТНЫХ РЕАЛИЗАЦИЙ КРИПТОГРАФИЧЕСКИХ АЛГОРИТМОВ

Введение

Криптография и криптоанализ всегда развивались параллельно. Во времена, когда использовались моноалфавитные и полиалфавитные шифры, для их взлома активно применялись методы частотного анализа. С появлением электромеханических машин, упрощающих процесс шифрования, были разработаны устройства, которые подбирали исходное положение частей машины для дешифрования сообщения [1]. Начиная с середины XX в., после публикации работ Шеннона [2], классический криптоанализ столкнулся с серьезными трудностями, так как наконец удалось формализовать понятие надежного шифра и разработать криптостойкие алгоритмы.

На данный момент все стандартизированные криптографические алгоритмы, такие как AES, RSA и другие, считаются устойчивыми к математическому способу анализа. Они используют такие преобразования, которые не позволяют злоумышленнику, опираясь лишь на открытые тексты и шифротексты, математическим способом найти ключ, а прямой перебор всех его возможных значений займет столько времени, что засекреченная информация потеряет актуальность к моменту дешифровки.

К сожалению, рассмотрение лишь математической стойкости алгоритма без исследования его конечного исполнения не может гарантировать того, что злоумышленник не получит доступа к секретному ключу или другой защищенной информации. С точки зрения системного подхода математическая модель шифра — это всего лишь подсистема его аппаратной или программно-аппаратной реализации, включающей также другие подсистемы, например программный код, который может содержать ошибки, вычислительные средства, которые, собственно, и выполняют алгоритм и т. д. (рис. 1).

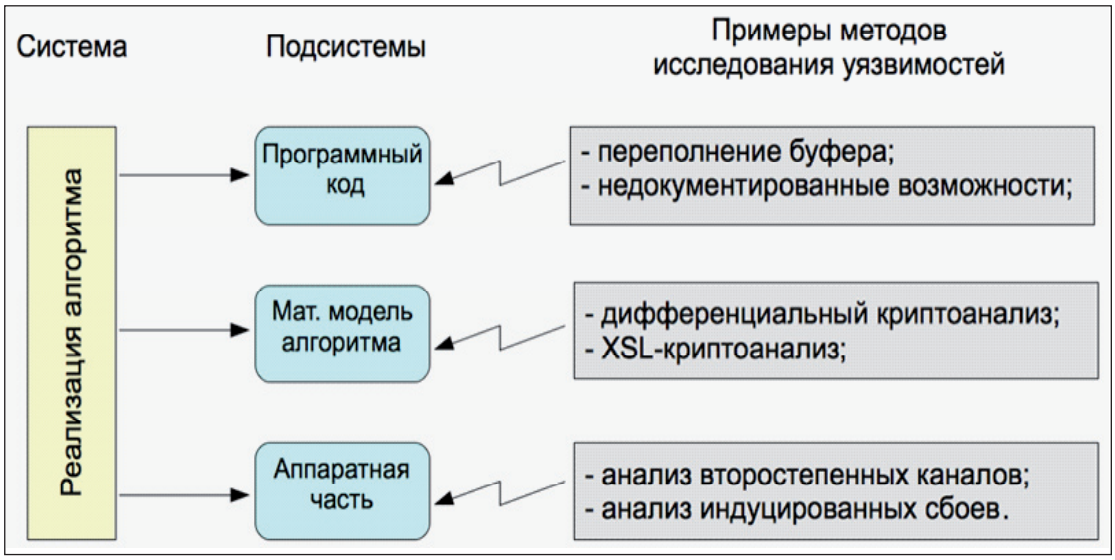


Рис. 1. Возможные методы тестирования уязвимостей подсистем алгоритма

Каждую из подсистем следует подвергать анализу на наличие уязвимостей. Программный код должен проверяться на отсутствие операций, позволяющих переполнить буфер памяти или

воспользоваться наличием каких-либо недокументированных возможностей, описанных, например, в [3]. Математическая модель шифра должна тщательно изучаться методами криптоанализа, такими как дифференциальный криптоанализ [4], XSL-анализ [5] и др. Существуют также методы, которые позволяют использовать физические особенности вычислительных средств для получения доступа к защищенной информации, например, для вычисления секретного ключа. Именно об этих методах и пойдет речь в данной статье.

1. Особенности систем на одном кристалле

Последней тенденцией в микроэлектронике является развитие систем на одном кристалле, то есть электронных схем, выполняющих функции целого устройства. Частными случаями таких систем являются микроконтроллеры и микропроцессоры, банковские карты, sim-карты, флеш-карты — в общем, практически любые вычислительные средства. Эти миниатюрные устройства получили крайне широкое распространение, так как обладают низкой стоимостью и позволяют решать широкий спектр задач, в том числе и задачи по защите информации, такие как обеспечение конфиденциальности личной и коммерческой информации, определение отправителя или получателя сообщений, создание цифровой подписи и т. д. Зачастую содержимое или функционал таких систем представляет интерес для злоумышленников, которые легко могут получить к ним полный физический доступ. В целях защиты системы на одном кристалле часто содержат криптографический модуль, призванный не дать злоумышленнику прочесть информацию, но нередко наличие уязвимостей нивелирует свойства шифра.

Система на одном кристалле — это реальный объект, во время работы которого в нем протекают физические процессы, используемые для вычислений. Параметры системы, такие как время работы, потребляемая мощность, электромагнитное излучение и другие, зависят от шага алгоритма, и, зная эту зависимость, можно попытаться вычислить обрабатываемые в текущий момент данные. Классическим примером является перезапись значения в регистре памяти, при которой количество измененных бит (с нуля на единицу и наоборот) прямо пропорционально изменению напряжения [6]. Другим классическим примером является время выполнения алгоритма, так как в целях оптимизации ресурсов часть кода будет выполняться только при определенных условиях, например, когда бит данных равен 1. Этот пример будет рассмотрен более подробно ниже.

Зависимость между физическими параметрами и данными представляет собой уязвимость, которой может воспользоваться злоумышленник. По той причине, что присутствие физических каналов утечки информации неочевидно, технология тестирования таких уязвимостей носит название «анализ по побочным каналам» [7], она широко используется против аппаратных и программно-аппаратных реализаций криптографических алгоритмов.

Близким к этому методу является анализ устройства микрозондированием [8]. Этот метод подразумевает, что злоумышленник может считать битовые данные с регистра памяти или с шины передачи информации. Для этого существуют несколько технологий. Во-первых, можно использовать специальный щуп, который регистрирует напряжение на устройстве. Фактически это своего рода вольтметр, с помощью которого можно считать передаваемый сигнал. Во-вторых, можно воспользоваться электронным или рентгеновским микроскопом и последовательно восстановить слои микроконтроллера. В некоторых случаях этот подход позволяет определить данные, которые зашиты на аппаратном уровне и не могут быть изменены [9]. Еще один способ — это регистрация отраженного и поглощенного излучения. Если облучать регистр памяти, то те ячейки, которые содержат в себе определенные значения, будут по-другому видны под микроскопом. Вероятно, существуют и другие способы считывания информации из регистров памяти и из шин передачи информации.

Анализ микрозондированием отличается от анализа устройства по побочным каналам тем, что здесь злоумышленнику не надо искать связь между физическими характеристиками устройства и данными, так как он уже считывает битовую информацию. Очевидно, что вследствие размеров систем на одном кристалле этот метод не применим в обычных условиях, так как требует дорогого оборудования.

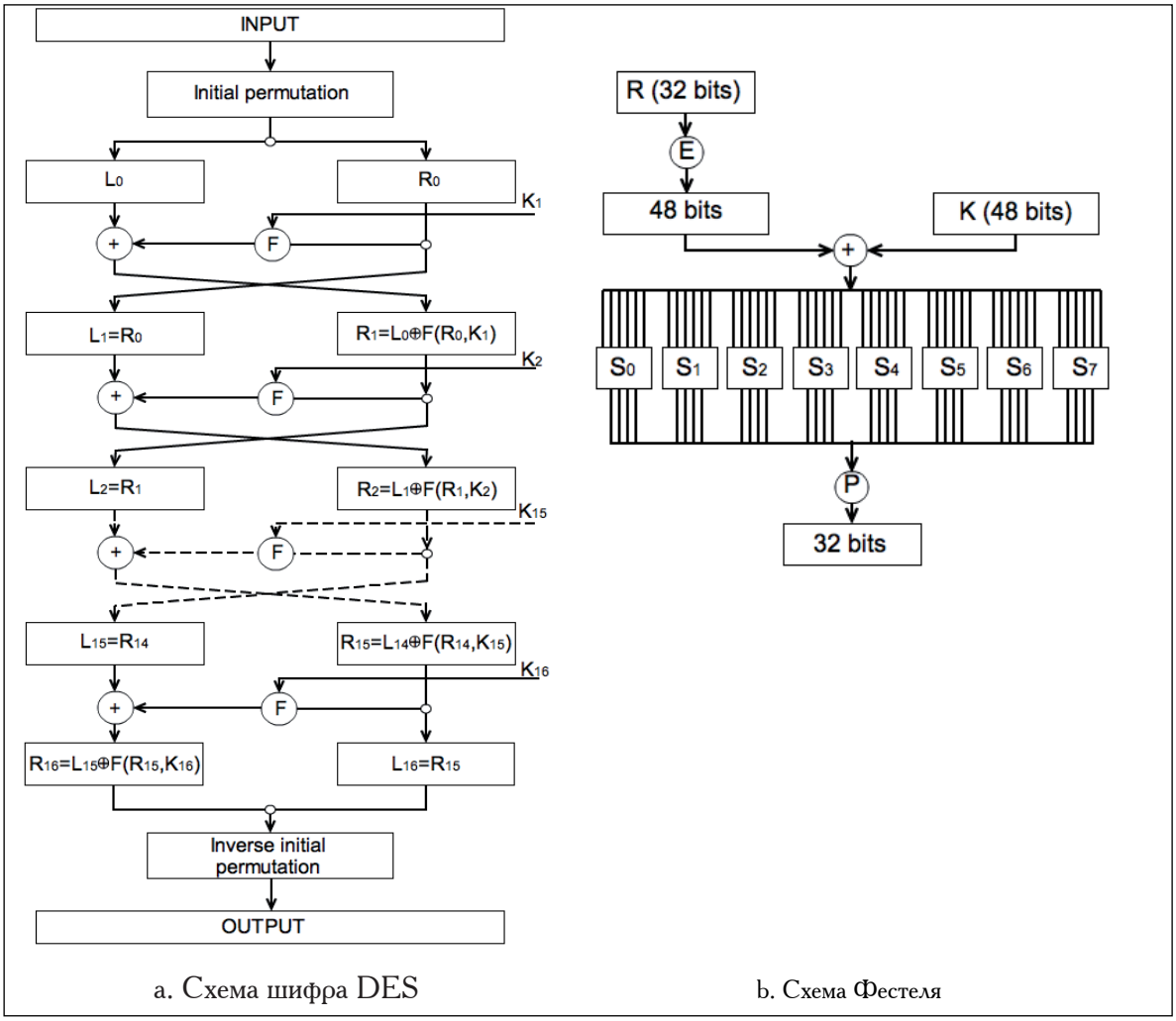
Еще один метод, ошибочно включаемый многими исследователями в анализ побочных каналов, носит название технологии индуцированных сбоев [10]. Система на одном кристалле может работать только при определенных внешних условиях, называемых нормальными: напряжение, температура и другие параметры должны находиться в заданных пределах. Что случается, если эти внешние условия нарушаются? Зачастую это приводит к некорректной работе устройства, то есть создает в нем нештатную ситуацию, ведущую к ошибке, которая может носить произвольный характер, например, может быть прочитана другая область памяти или стерты все значения из регистров. Эти ошибки открывают новые возможности для злоумышленника. Воздействуя на устройство, выполняющее алгоритм шифрования, он может попытаться изменить значения бита, одного или нескольких байт памяти, пропустить выполнение инструкции, стереть значения из ОЗУ или создать другие типы ошибок, которые могут быть использованы для раскрытия секретного ключа алгоритма [11]. Отличительная особенность этого метода от анализа побочных каналов в том, что злоумышленник влияет на работу устройства, создавая в нем ошибку. При этом он не использует физические данные с побочных каналов, а сравнивает результат правильного и неправильного шифрования и на основе сравнения получает секретный ключ.

Эти три ставших уже классическими метода анализа уязвимостей аппаратных и программно-аппаратных реализаций шифров будут более подробно рассмотрены на примере алгоритма DES.

2. Алгоритм DES

Симметричный алгоритм шифрования DES был стандартизирован правительством США в 1977 г. [12]. Он использовался для шифрования информации, которая по грифу доступности не относилась к секретной. В настоящий момент данный алгоритм используется лишь в устаревших системах, так как он был заменен более современным шифром AES, хотя во многих приложениях можно встретить алгоритм Triple-DES, в котором для шифрования блока данных последовательно применяется DES с тремя разными ключами.

Алгоритм DES является блочным — в нем происходит шифрование блоками по 64 бита. Длина ключа тоже 64 бита, но значащими являются лишь 56 бит, так как остальные 8 используются для контроля четности. Операция шифрования происходит за 16 раундов (рис. 2), во время которых к данным применяются два вида преобразований: линейные и нелинейные. К первым относятся начальная перестановка, конечная перестановка, функция расширения E, перестановка P и операция исключающего «или». К нелинейным функциям относится табличная функция преобразования S. Подробное описание этих функций приведено в стандарте и во многих пособиях по криптографии, например в [13, 14].



a. Схема шифра DES b. Схема Фестеля

Рис. 2. Операция шифрования алгоритма DES

С точки зрения рассматриваемых в данной статье методов анализа криптографических алгоритмов начальная и конечная перестановки не выполняют никаких функций защиты, скорее даже наоборот, так как по этим двум операциям можно определить, когда началось и закончилось выполнение шифра.

Выше уже говорилось о том, что часть преобразований алгоритма DES — линейные и обратимые, то есть, зная результат работы таких преобразований, можно определить, что подавалось на вход. К ним относятся все функции данного шифра за исключением операций $S_0 \ldots S_7$, приведенных на рис. 2b, на вход которых подается 6-битное значение, а на выходе получается 4-битное.

Данный алгоритм отчасти был стандартизирован по той причине, что его просто реализовать аппаратно, при этом количество логических элементов и потребляемая электроэнергия крайне невелики. В отличие от программных реализаций, аппаратные решения алгоритма DES работают в десятки раз быстрее [15], даже на более низких частотах.

3. Анализ побочных каналов утечки

Как уже говорилось выше, побочным каналом утечки информации может служить время шифрования. Если рассмотреть алгоритм DES, то любую перестановку в нем можно реализовать следующим образом. Вначале обнулить переменную, которая будет сохранять результат перестановки. Затем во входной переменной искать лишь единичные биты данных, то есть когда бит равен 1, то осуществлять его перестановку, а когда бит равен нулю, то переходить к

следующему биту. Проблема здесь заключается в том, что время выполнения программы прямо пропорционально зависит от количества единиц во входной переменной, которое носит название веса Хемминга. Ниже на языке ANSI C приведена программная реализация перестановки P, выполняемой в схеме Фестеля, по которой будет проверена связь между весом Хемминга и временем выполнения шифра.

```
uint_fast32_t Pperm(uint_fast32_t input) {
	int i = 0, j = 0, k = 0;
	uint_fast32_t res = 0x0ULL, one = 0x1ULL;
	int p_table[32] = { 16,  7, 20, 21, 29, 12, 28, 17,
	                     1, 15, 23, 26,  5, 18, 31, 10,
	                     2,  8, 24, 14, 32, 27,  3,  9,
	                    19, 13, 30,  6, 22, 11,  4, 25};
	/*For all bits from input*/
	for (i=0;i<32;i++) {
		/*if bit is 1 then permute, else go to next bit*/
		if ( GET_BIT(input,i) == 1){
			/*Find a position for this bit*/
			for (j=0;j<32;j++)
				if (p_table[j] == (32-i))
					k = j;
			/*Put the bit in a correct place*/
			res |= (one << (31 - k));
		}
	}
	return res;
}
```

Принцип работы этой программы подразумевает, что следует переставлять только единичные биты, поэтому время работы зависит от входных данных. Этот код был реализован и протестирован на компьютере под управлением Mac OS 10.5.7 с процессором Intel Core 2 Duo 2,1 ГГц и 1 Гб оперативной памяти. Время работы для некоторых исходных текстов и соответствующих им шифротекстов, измеренное в тактах процессора, приведено в таблице 1.

Таблица 1. Время работы шифра DES для различных исходных текстов

Исходный текст	Шифротекст	Кол-во тактов процессора
84BEEE5BBC5EDCF0	2ED19609CD8B3332	28711
1D83BDA069F99B9B	4D05EB9771BF6008	28628
06488608AD926A95	549C3A9847842EEC	29628

Как видно из таблицы 1, время работы шифра хоть и зависит от входных данных, но все же не настолько сильно. Более того, если перевести процессорные такты в секунды, то среднее время операции шифрования составит примерно 13,7 мкс, таким образом, за 1 с можно зашифровать более 72000 исходных текстов.

Теперь рассмотрим, как злоумышленник может использовать эти данные. Предположим вначале, что он имеет возможность записывать получившиеся шифротексты и измерять время работы (необязательно в процессорных тактах). Также он знает, что перестановка P осуществляется по алгоритму, приведенному выше. Что он может из этого извлечь?

Как известно, часть операций алгоритма DES обратимы, поэтому следующие выражения верны:

$$(R_{16}L_{16}) = FP^{-1}(\text{output}) \tag{1}$$

$$\begin{cases} L_{16} = R_{15} \\ R_{16} = L_{15} \oplus f(R_{15}, K_{16}) = L_{15} \oplus P\left(\text{Sbox}\left[E(R_{15}) \oplus K_{16}\right]\right) \end{cases} \Rightarrow \tag{2}$$
$$\Rightarrow R_{16} = L_{15} \oplus P\left(\text{Sbox}\left[E(L_{16}) \oplus K_{16}\right]\right)$$

Как уже было сказано выше, время выполнения операции $P(Sbox[E(L_{16})\oplus K_{16}]\}$ прямо пропорционально весу Хемминга величины $Sbox[E(L_{16})\oplus K_{16}]$, поэтому существует ненулевая корреляция между измеренным временем выполнения шифра T и весом Хемминга данной величины:

$corr\ (T, HW(Sbox[E(L_{16})\oplus K_{16}]))$

Параметр T — это время работы всего алгоритма шифрования с момента подачи исходного текста до момента получения шифротекста. Время может быть измерено как в секундах, так и в процессорных тактах, на результат атаки это не влияет.

Значение ключа, которому соответствует максимальный коэффициент корреляции, должно совпадать с действительным. Если перебирать все возможные значения ключа K_{16} и для каждого из них вычислять коэффициент корреляции по указанной выше формуле, то это будет похоже на атаку методом перебора, так как существует всего 2^{48} различных значений ключа K_{16}, следовательно, необходимо найти другой способ.

Операции $S_0 \ldots S_7$ вычисляются независимо друг от друга (3), поэтому можно утверждать, что вес Хемминга от результата каждой отдельной операции S_i коррелирует с общим временем выполнения шифра (4)[1].

$$R_{16}=L_{15}\oplus P(S_0[E(L_{16})^{\overline{1....6}} \oplus K_{16}{}^{\overline{1....6}}]....S_7[E(L_{16})^{\overline{42....48}} \oplus K_{16}{}^{\overline{42....48}}]) \tag{3}$$

$$corr\ (T, HW(S_i[E(L_{16})^{\overline{6\cdot i+1....6\cdot(i+1)}} \oplus K_{16}{}^{\overline{6\cdot i+1....6\cdot(i+1)}}])) \tag{4}$$

В формуле (4) верхний индекс обозначает, что из величин L_{16} и K_{16} берутся только биты, подающиеся на вход i-й таблицы S_i.

Это замечание позволяет искать ключ K_{16} частями по 6 бит, которые служат входом для операции S_i. Такой поиск осуществить гораздо проще, так как для нахождения всех бит ключа потребуется перебрать лишь 2^6 различных комбинаций и среди них выбрать ту, для которой найденный коэффициент корреляции будет максимальным. При наличии небольшого числа данных такой метод будет давать ошибочный результат, но с их ростом корреляция неправильного ключа

[1] Дополнительную информацию о связи одной операции S_i и времени выполнения всего шифра можно найти в открытом доступе, например в [16].

будет разве что уменьшаться, а корреляция правильного — увеличиваться. Данный метод носит название метода максимального правдоподобия.

Предположим, что у злоумышленника имеется M исходных текстов, M соответствующих им шифротекстов и время работы каждой отдельной операции шифрования. Для нахождения 6 бит ключа K_{16} может быть использован следующий алгоритм, приведенный в виде псевдокода:

```
/*HW[64][M] – hamming weight for different key candidates*/
For iEnc=0 to M
   (R16,L16) = InverseFinalPermutation(ciphertext_iEnc);
   For key=0 to 63
       HW[key][iEnc] = computeHammingWeight(Sbox[E(L16) ⊕ key]);
   EndFor
EndFor

For key =0 to 63
   corr[key] = computeCorrelation(time,HW[key]);
EndFor

best_guess = findCandidateWithMaximumCorrelation(corr);
```

Данный способ тестирования уязвимости был проверен на программной реализации алгоритма DES, указанной выше. Для этого на компьютере под управлением Mac OS 10.5.7 с процессором 2,1 ГГц и 1 Гб оперативной памяти было выполнено 50000 операций шифрования. Все результаты, равно как и время работы каждой отдельной полной операции шифрования (время измерялось для всех 16 раундов шифрования), были записаны.

Нахождение всего ключа K_{16} велось частями по 6 бит. В качестве коэффициента корреляции был выбран коэффициент Пирсона [17]. Тестировался только последний 16-й раунд (сама технология применима также к первому раунду) без использования каких-либо дополнительных решений, улучшающих результат. Минимальное количество операций шифрования, необходимых для полного восстановления ключа, составило 9980. При этом стоит отметить, что наличие операционной системы, в которой реализованы прерывания выполняемого кода, частота процессора и другие функции алгоритма DES, влияющие на время шифрования, — все это должно было усложнить нахождение ключа. На более медленных процессорах, а также при отсутствии многозадачной операционной системы результат должен быть получен быстрее. Также возможно использование техник, которые уменьшают количество операций шифрования, необходимых для полного восстановления ключа. К ним относятся одновременный анализ первого и последнего раундов с последующим слиянием результатов, использование различных коэффициентов корреляции и др.

В целом было показано, что каналом утечки информации могут служить неочевидные физические параметры, например время. Так как для измерения времени выполнения шифра злоумышленнику необязательно нужен полный физический доступ к устройству, такой метод может быть использован и для удаленных сервисов.

4. Анализ устройства микрозондированием

Этот метод подразумевает, что с помощью микрозонда или любым другим способом злоумышленник может считывать данные с регистров памяти или шин передачи информации. Если рассмотреть реализацию алгоритма DES, например, от компании Ocean Logic [18], в которой присутствуют всего два регистра памяти (L и R), то анализ микрозондированием может

быть выполнен следующим образом. Предположим, что злоумышленник может считать b-й бит из регистра памяти, в котором сохраняются промежуточные результаты L_i алгоритма DES. Зная значения этого бита для каждого раунда и результат шифрования, он может восстановить несколько битов из первого K_1 и последнего K_{16} раундовых ключей. Рассмотрим последний 16-й раунд шифрования (рис. 3):

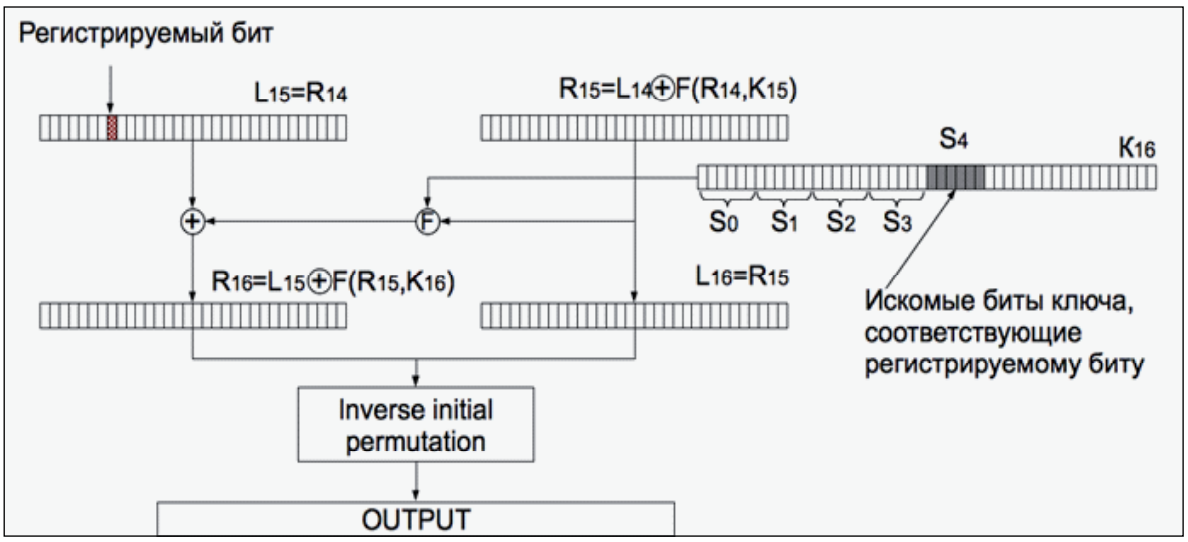


Рис. 3. Обобщенная схема соответствия данных алгоритма DES

Известно, что, зная шифротекст, можно определить значения R_{16} и L_{16} по формуле (1). Из формулы (2) следует, что:

$$R_{16} = L_{15} \oplus F(L_{16}, K_{16}).$$

Так как в схеме Фестеля присутствует операция перестановки, то необходимо найти соответствие между битом b и битами ключа K_{16}, которые будут складываться с ним. В случае, изображенном на рис. 3, злоумышленнику известен 8-й бит из значения L_{15}. Он складывается с выходом лишь одной операции S_i, номер которой легко определяется по таблице P. Согласно этой таблице, на 8-м месте стоит бит, который до перестановки был на 17-м месте, то есть он был частью выхода операции S_4. Следовательно, измеряя значения 8-го бита, злоумышленник может попытаться найти биты ключа K_{16} с 25-го по 30-й.

Проведя лишь одно шифрование, злоумышленник может воспользоваться выражением (5) и отбросить, как минимум, половину возможных значений ключа K_{16}:

$$L_{15}^{\,8} = R_{16}^{\,8} \oplus P^8(S_4[E(L_{16})^{\overline{25\ldots30}} \oplus K_{16}^{\overline{25\ldots30}}]) \qquad (5)$$

Здесь P^8 означает, что из всех битов операции S_4 берется только тот, который в сумме с битом $R_{16}^{\,8}$ будет давать бит $L_{15}^{\,8}$. За исключением 6 бит ключа K_{16} все переменные в формуле (5) известны, поэтому простой перебор позволяет из 64 возможных значений ключа оставить только 32. Обычно, используя данные с 8 операций шифрования, можно точно определить этот набор из 6 бит. Такой подход применим также для нахождения 6 бит ключа K_1.

Предположим, что имеется M операций шифрования, во время которых записывается значение b-го бита из регистра памяти L. Ниже в виде псевдокода приведен алгоритм, позволяющий определить 6 бит ключа K_{16}:

```
/* indBit - bit index (position of a recorded bit)*/
/* bits[M][16] - 16 bits recorded from register L in position b*/
/* keys[64] - 64 different key candidates (input for one Sbox)*/

For iEnc=0 to M-1
    (R16,L16)=InverseFinalPermutation(ciphertext_iEnc);
    For iKey=0 to 63
        If (keys[iKey] != -1)
            Res = R16 ⊕ P(Sbox[E(L16) ⊕ keys[iKey]]);
            bit = GetBit(Res, indBit);

            If (bit != bits[i][14])
                keys [j] = -1;
            EndIf
        EndIf
    EndFor
EndFor

FindKeyCandidateNotEqualToNULL(keys);
```

В таблице 2 приведены смоделированные данные, на основании которых был проверен приведенный выше алгоритм.

Таблица 2. Данные, смоделированные для проверки метода микрозондирования

Исходный текст	8-й бит регистра L, измеренный во всех 16 операциях шифрования	Шифротекст
AEBCF7869FE45F1F	1 1 0 0 1 1 0 1 0 0 0 1 1 1 1 1	BB73C4282F2C21FB
FC67EB4801E849AE	1 1 0 0 0 0 0 0 1 1 0 1 0 0 1 1	99B042D33D938996
4A3D1B87674E3B43	0 0 1 0 1 0 1 1 0 1 1 0 1 0 1 1	A654C1CCC4E4F39B
D62805AB308ED932	1 0 1 1 0 0 1 0 1 1 0 0 1 1 0 1	DED0503D46DEFC09
6249B5F9F55618D3	0 1 0 0 0 1 0 1 0 0 0 0 0 0 1 0	6A7FEA1AD3D0EA74
E6725C73B2DDE4EA	1 1 0 1 1 0 1 1 1 1 1 0 0 0 0 1 1	BD3E0F19D087226B
CE4927BFADC43D9F	1 1 0 0 1 1 0 1 0 0 0 0 1 1 1 1	FC1E4BEB0E113B37
B5F50EDA1F11D1CC	1 0 0 1 0 1 0 0 1 0 0 0 1 0 0 0	18D5E72277A5222C

По данным из таблицы 2 были вычислены биты ключа K_{16} с 25-го по 30-й, они получились равными 101111. Затем анализ первого раунда позволил вычислить 6 бит ключа K_1, они равны 100111. Эти значения совпали с действительными. На основе полученной информации можно проводить и дальнейший анализ секретного ключа K, который позволит существенно ускорить прямой перебор.

Таким образом, было показано, что для нахождения 12 бит секретного ключа K с помощью технологии микрозондирования требуется всего лишь 8 операций шифрования. Это говорит об

эффективности данного метода, однако высокая стоимость оборудования и переход на 90- и 65-нанометровые технологии не позволяет злоумышленнику активно применять данный подход.

5. Метод индуцированных сбоев

Метод индуцированных сбоев, вероятно, самый применяемый среди всех методов тестирования аппаратных и программно-аппаратных реализаций криптографических алгоритмов. Подача неправильного напряжения, изменение внутренней структуры устройства, облучение с помощью лазера, нагревание и множество других методов позволяют получить ошибку в процессе выполнения алгоритма шифрования, которая может быть использована для определения секретного ключа.

Важным понятием данной технологии является модель ошибки, то есть тип ошибки и время создания сбоя. Пожалуй, самые используемые типы ошибки — это изменение бита, одного или нескольких байт и пропуск инструкции. Время создания сбоя подразумевает, в какой именно момент выполнения алгоритма должна произойти ошибка, чтобы результат работы можно было использовать для анализа.

Если рассматривать метод индуцированных сбоев на примере алгоритма DES, то одной из самых удобных моделей ошибки является сбой в регистре R_{15}. В этом случае будет легко определить тип ошибки и, соответственно, использовать ее для вычисления раундового ключа K_{16}, что и будет показано далее. Для наглядности предположим, что в регистре R_{15} был изменен лишь один бит (рис. 4).

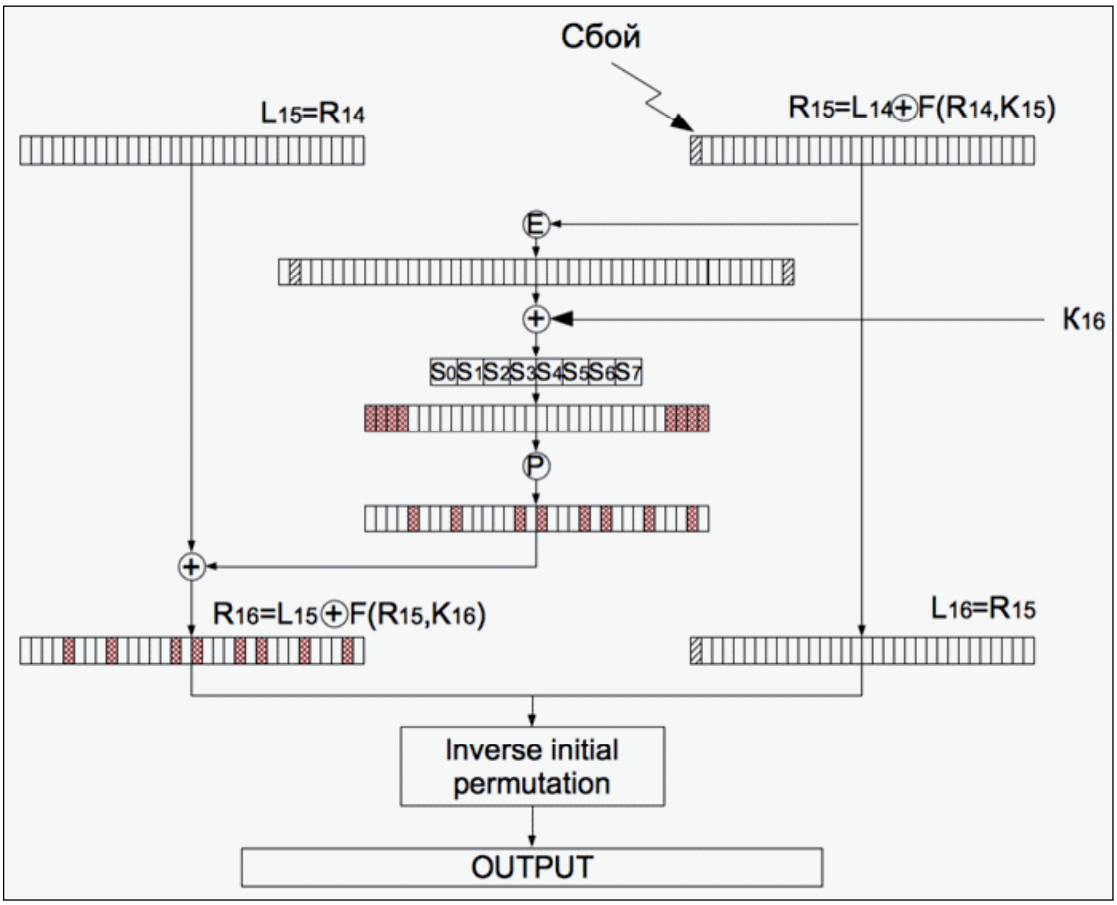


Рис. 4. Распространение ошибки, созданной в 16-м раунде алгоритма DES

Как можно понять из рис. 4, изменение одного бита в регистре R_{15} приведет к тому, что в 16-м раунде 8 бит R_{16} и 1 бит L_{16} будут изменены, поэтому данную модель ошибки довольно просто распознать. Предположим, что злоумышленник владеет правильным C и ошибочным $\hat{C}$

шифротекстами. Это дает ему возможность вычислить измененный бит и найти части раундового ключа K_{16} следующим образом:

$$R_{16} = L_{15} \oplus P(S[E(L_{16}) \oplus K_{16}])$$
$$\hat{R}_{16} = L_{15} \oplus P(S[E(\hat{L}_{16}) \oplus K_{16}]). \quad (6)$$

Сложив эти два выражения побитово по модулю 2, получим выражение, в котором присутствует лишь один неизвестный элемент, а именно K_{16}:

$$R_{16} \oplus \hat{R}16 = P(S[E(L_{16}) \oplus K_{16}]) \oplus P(S[E(\hat{L}_{16}) \oplus K_{16}]). \quad (7)$$

Можно воспользоваться тем свойством, что перестановка P линейна и обратима, это немного упростит выражение (7):

$$P^{-1}(R_{16} \oplus \hat{R}16) = S[E(L_{16}) \oplus K_{16}]) \oplus S[E(\hat{L}_{16}) \oplus K_{16}]. \quad (8)$$

Использовав независимость операций S_i, окончательно приходим к выражению (9):

$$P^{-1}(R_{16} \oplus \hat{R}16)|^{\overline{1...6}} = S_0[E(L_{16})^{\overline{1...6}} \oplus K_{16}^{\overline{1...6}}] \oplus S_0[E(\hat{L}_{16})^{\overline{1...6}} \oplus K_{16}^{\overline{1...6}}]$$
$$\ldots \quad (9)$$
$$P^{-1}(R_{16} \oplus \hat{R}16)|^{\overline{43...48}} = S_7[E(L_{16})^{\overline{43...48}} \oplus K_{16}^{\overline{43...48}}] \oplus S_7[E(\hat{L}_{16})^{\overline{43...48}} \oplus K_{16}^{\overline{43...48}}].$$

Для полного восстановления 6 бит ключа K_{16}, служащих входом для одной операции S_i, в среднем необходимо 3 пары правильного и ошибочного шифротекстов, желательно с разными ошибками. Алгоритм их поиска довольно прост и приведен ниже в виде псевдокода:

```
/* keys[64] - 64 different key candidates (input for one Sbox)*/
/* numPairs - a number of pairs (correctCipher,wrongCipher)*/

For iPair=0 to numPairs
    (R16c,L16c) = ReverseFinalPermutation(correctCipher[iPair]);
    (R16w,L16w) = ReverseFinalPermutation(wrongCipher[iPair]);
    res = ReversePPermutation(R16c ⊕ R16w);

    For iKey=0 to 63
        If (keys[iKey] != -1)
            ans = Sbox[E(L16c) ⊕ keys[iKey]] ⊕ …
                    Sbox[E(L16c) ⊕ keys[iKey]];

            If (get_four_bits(res) != ans)
                keys[iKey] = -1;
            EndIf
        EndIf
    EndFor
EndFor
PrintOutRemainingKeys(keys);
```

Для проверки данного алгоритма, программным способом была смоделирована однобитовая ошибка в регистре R_{15}. Результаты выполнения шифра DES с введенным сбоем и без него приведены в таблице 3.

Таблица 3. Правильный и ошибочный шифротексты

Исходный текст	Правильный шифротекст	Ошибочный шифротекст
E51783E95677DEED	DEB93BB4F31D80FC	DEBD3BB0F31C8068
D89C2474609A40D3	77AEAC6B8DB55455	77BFAC2B8DB554C1
0489C80201F05DB9	61A3AD91B86F75CA	61A6ADD5B86E755A

Было получено, что при использовании данных из таблицы 3 остаются два набора из 6 первых бит (100110 и 110110) и два набора из 6 последних бит (110010 и 110011) ключа K_{16}, которые удовлетворяют уравнению (9). Таким образом, прямой перебор ключа K_{16} был уменьшен с 2^{48} до 2^{38}.

Если ввести ошибку в другую часть регистра R_{15}, то можно получить недостающие части ключа. Также существуют другие модели ошибки, которые могут быть использованы против алгоритма DES. Если алгоритм реализован программным способом, то количество моделей ошибок увеличивается, так как в этом случае целью могут служить не алгоритмические составляющие шифра, а особенности кода, его реализующего. Например, можно изменить начальный адрес таблицы E так, что она будет возвращать все нули.

Метод индуцированных сбоев — это довольно эффективный способ определения ключа практически любого криптографического алгоритма. На конкретном примере была показана математическая база, необходимая для успешного использования данной методики, которая сводится к сравнению правильного и неправильного результатов шифрования. Из-за низкой стоимости оборудования, которое требуется для успешного введения ошибки, данная технология может быть легко использована злоумышленниками, поэтому следует применять дополнительные методы защиты криптографических алгоритмов.

Выводы

В данной статье на примере шифра DES были рассмотрены методы анализа аппаратных и программно-аппаратных реализаций криптографических алгоритмов. По типу получаемой информации среди них можно выделить: анализ побочных каналов, анализ микрозондированием и анализ индуцированных сбоев. Математическая логика этих трех подходов также различается. В случае использования побочных каналов в основном используются корреляционные методы вычисления секретного ключа. При микрозондировании нарушитель выбирает только те ключи, которые могут обеспечить полученные промежуточные данные. Анализ индуцированных сбоев подразумевает сравнение правильного и ошибочного результатов шифрования.

Повсеместное использование портативных устройств позволяет злоумышленнику легко получить полный физический доступ к ним, поэтому он может использовать любой из предложенных выше методов или даже объединять их. Следовательно, криптографический модуль, представленный на таком устройстве, должен быть стойким ко всем трем методам анализа. Этого можно добиться как при использовании специальных аппаратных решений, так и правильной программной реализацией, которая может включать маскировку данных, введение дополнительных проверок и другие методы.

СПИСОК ЛИТЕРАТУРЫ:

1. *Кан Д.* Война кодов и шифров: история четырех тысячелетий криптографии / Пер. с англ. Е. С. Алексеева. М.: Рипол Классик, 2004. — 528 с.
2. *Шеннон К.* Работы по теории информации и кибернетике. М.: Изд-во иностр. лит-ры, 1963. — 830 с.
3. *Скляров И.* Программирование боевого софта под Linux. СПб.: БХВ-Петербург, 2007. — 416 с.

4. *Панасенко С.* Современные методы вскрытия алгоритмов шифрования, часть 3 [Электронный ресурс] // Chief Information Officier. 2003. URL: http://computerra.ru/cio/old/weekly/295841/page2.html (дата обращения: 08.02.2014).
5. *Courtois N., Pieprzyk J.* Cryptoanalysis of block ciphers with overdefined system of equations [Электронный ресурс] // LNCS 2501. 2002. URL: http://eprint.iacr.org/2002/044.pdf (дата обращения: 08.02.2014).
6. *Kocher P., Jaffe J., Jun B.* Differential power analysis [Электронный ресурс] // Advances in Cryptography, Crypto 99. 1999. Vol. 1666. URL: http://www.cryptography.com/resources/whitepapers/DPA.pdf (дата обращения: 08.02.2014).
7. *Жуков А. Е.* Криптоанализ по побочным каналам (Side Channel Attacks) [Текст] / Жуков, А. Е. // Защита информации. Инсайд. — 2010. № 5. С. 28—33.
8. *Handschuh H., Paillier P., Stern J.* Probing attacks on tamper-resistant devices // First International Workshop, CHES'99. Lecture Notes in Computer Science. 1999. vol. 1717. Berlin: Springer Berlin Heilderberg, 1999. — 356 p.
9. *Skorobogatov S.* Semi-invasive attacks — A new approach to hardware security analysis [Электронный ресурс] // Technical Report UCAM-CL-TR-630. University of Cambridge, Computer Laboratory. 2005. URL: http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-630 (дата обращения: 08.02.2014).
10. *Boneh D., DeMillo R. A., Lipton R. J.* On the importance of checking cryptographic protocols for faults // Advances in cryptology — EUROCRYPT'97. Springer, 1997. P. 37—51.
11. *Bar-El H., Choukri H., Naccache D., Tunstall M., Whelan C.* The sorcerer's apprentice guide to fault attacks // Aims & Scope. Proceedings of the IEEE. Vol. 94. Issue 2. New York: Institute of Electrical and Electronics Engineering, Inc. 2006. — 484 p.
12. National Bureau of Standards. Data Encryption Standard. Federal Information Processing Standards Publication 46. Washington: National Bureau of Standards, U.S. Departmen of Commerce. 1977. —22 p.
13. *Нестеров С. А.* Информационная безопасность и защита информации: учеб. пособие. СПб.: Изд-во Политехн. ун-та, 2009. — 126 с.
14. *Фомичев В. М.* Методы дискретной математики в криптологии. М.: ДИАЛОГ-МИФИ, 2010. — 424 с.
15. *Vishwanath P., Joshi C., Saxena K.* FPGA implementation of DES using piplining concept with skew core key-scheduling // Journal of theoretical and applied information technology. Pakistan: Little Lion Scientific Islambad. 2009. P. 372.
16. *Korkikian R.* Timing attacks — part 1 [Электронный ресурс] // Блог компании Kudelski Security. URL: http://cybermashup.com/2013/12/13/timing-attacks-part-1/ (дата обращения 08.02.2014).
17. Общая теория статистики: Учебник / Под ред. Р. А. Шмойловой. 3-е изд., переработ. М.: Финансы и Статистика, 2002. — 560 с.
18. DES cryptoprocessor [Электронный ресурс] // Описание устройства компании Ocean logic Pty Ltd. URL: http://www.ocean-logic.com/pub/OL_DES.pdf (дата обращения: 08.02.2014).

REFERENCES:

1. *Kan D.* Voina kodov i shifrov: istoriia chetyrekh tysiacheletii kriptografii / Per. s angl. E. S. Alekseeva. M.: Ripol Klassik, 2004. — 528 p.
2. *Shennon K.* Raboty po teorii informatsii i kibernetike. M.: Izd. inostr. lit., 1963. — 830 p.
3. *Skliarov I.* Programmirovanie boevogo softa pod Linux. BKhV-Peterburg, 2007. — 416 p.
4. *Panasenko S.* Sovremennye metody vskrytiia algoritmov shifrovaniia, chast' 3 [Elektronnyi resurs] // Chief Information Officier. 2003. URL: http://computerra.ru/cio/old/weekly/295841/page2.html (data obrashcheniia: 08.02.2014).
5. *Courtois N., Pieprzyk J.* Cryptoanalysis of block ciphers with overdefined system of equations [Elektronnyi resurs] // LNCS 2501. 2002. URL: http://eprint.iacr.org/2002/044.pdf (data obrashcheniia: 08.02.2014).
6. *Kocher P., Jaffe J., Jun B.* Differential power analysis [Elektronnyi resurs] // Advances in Cryptography, Crypto 99. 1999. Vol. 1666. URL: http://www.cryptography.com/resources/whitepapers/DPA.pdf (data obrashcheniia: 08.02.2014).
7. *Zhukov A. E.* Kriptoanaliz po pobochnym kanalam (Side Channel Attacks) // Materialy konferentsii RusKripto — 2006.
8. *Handschuh H., Paillier P., Stern J.* Probing attacks on tamper-resistant devices // First International Workshop, CHES'99. Lecture Notes in Computer Science. 1999. vol. 1717. Berlin: Springer Berlin Heilderberg, 1999. —356 p.
9. *Skorobogatov S.* Semi-invasive attacks — A new approach to hardware security analysis [Elektronnyi resurs] // Technical Report UCAM-CL-TR-630, University of Cambridge, Computer Laboratory. 2005. URL: http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-630 (data obrashcheniia: 08.02.2014).
10. *Boneh D., DeMillo R. A., Lipton R. J.* On the importance of checking cryptographic protocols for faults // Advances in cryptology — EUROCRYPT'97. Springer, 1997. P. 37—51.
11. *Bar-El H., Choukri H., Naccache D., Tunstall M., Whelan C.* The sorcerer's apprentice guide to fault attacks // Aims & Scope. Proceedings of the IEEE. Vol. 94. Issue 2. New York: Institute of Electrical and Electronics Engineering, Inc. 2006. P. 484.
12. National Bureau of Standards. Data Encryption Standard. Federal Information Processing Standards Publication 46. Washington: National Bureau of Standards, U.S. Departmen of Commerce. 1977. — 22 p.
13. *Nesterov S. A.* Informatsionnaia bezopasnost' i zashchita informatsii: ucheb. posobie. SPb.: Izd-vo Politekhn. un-ta, 2009. — 126 p.
14. *Fomichev V. M.* Metody diskretnoi matematiki v kriptologii. M.: DIALOG-MIFI, 2010. — 424 p.
15. *Vishwanath P., Joshi C., Saxena K.* FPGA implementation of DES using piplining concept with skew core key-scheduling // Journal of theoretical and applied information technology. Pakistan: Little Lion Scientific Islambad. 2009. P. 372.
16. *Korkikian R.* Timing attacks — part 1 [Elektronnyi resurs] // Blog kompanii Kudelski Security. URL: http://cybermashup.com/2013/12/13/timing-attacks-part-1/ (data obrashcheniia 08.02.2014).
17. Obshchaia teoriia statistiki: Uchebnik / Pod red. R. A. Shmoilovoi. 3-e izdanie, pererabotannoe. M.: Finansy i Statistika, 2002. — 560 s.
18. DES cryptoprocessor [Elektronnyi resurs] // Opisanie ustroistva kompanii Ocean logic Pty Ltd. URL: http://www.ocean-logic.com/pub/OL_DES.pdf (data obrashcheniia: 08.02.2014).