УДК 519.688

КОДОВЫЕ ШАРАДЫ

А. Л. Чмора,
ведущий специалист
ОАО «Инфотекс»

Рассматривается метод противодействия DoS-атаке с использованием шарад, построенных на основе кодов, исправляющих ошибки. Обосновывается практическая состоятельность итеративного метода конструирования шарад. Итеративные кодовые шарады позволяют исключить применение квантового компьютера и массированного распараллеливания в целях снижения трудоемкости отыскания решения.

Ключевые слова *— DoS-атака, вычислительные шарады, коды, исправляющие ошибки, линейные коды.*

Введение

DoS-атаки (*Denial of Service*) широко распространены в сети Интернет [1, 2]. Задача атакующего – создать искусственную ситуацию, в которой добросовестному потребителю будет отказано в предоставлении соответствующих услуг. Для объяснения воспользуемся следующей бытовой аналогией. Предположим, что в ресторане имеется некоторое количество столиков и каждый можно зарезервировать, позвонив по телефону. Звонки принимаются до часа дня включительно. Злоумышленник, воспользовавшись простейшей стратегией, способен причинить ресторану ощутимый финансовый и репутационный ущерб. Для этого достаточно зарезервировать все доступные столики до установленного часа. Очевидно, что если все столики уже зарезервированы, то большинство потенциальных посетителей откажется от запланированного посещения и предпочтет другой ресторан. Конечно же, в какой-то момент руководство ресторана обнаружит факт злоупотребления и аннулирует резервирование, но с непренебрежимой вероятностью некоторое количество столиков в течение вечера не будет востребовано.

Сетевая DoS-атака

Поглощающая ресурсы стратегия относится к наиболее распространенному типу DoS-атаки. Так, злоумышленник способен инициировать аномальное количество сетевых соединений, но предусмотренные протоколом правила взаимодействия при этом умышленно игнорируются. Рассмотрим подробнее, как это происходит на примере TCP-соединения [3].

Обычно TCP-соединение устанавливается при помощи полуторараундного протокола [1]. Клиент инициирует соединение, передав серверу специальный SYN-пакет. В ответ сервер передает пакет SYN ACK и тем самым подтверждает соединение. По факту подтверждения соединения сервер выделяет некоторую область памяти. Подчеркнем, что объем выделенной под соединения памяти конечен. В завершение клиент выполняет квитирование и передает серверу ACK-пакет. Атакующий инициирует множество соединений, в которых пакеты SYN и SYN ACK передаются в соответствии с протоколом, но умышленно опускает квитирование, т. е. ACK-пакет никогда не передается. В итоге соединение не устанавливается, но при этом сервер не освобождает выделенную область памяти. Если количество таких незавершенных соединений достаточно велико, то происходит переполнение памяти и сервер перестает реагировать на какие-либо запросы. Понятно, что на прикладном уровне описанная сетевая атака приводит к отказу в обслуживании.

Вычислительные шарады

Воспользуемся вычислительной задачей, для которой трудоемкость отыскания решения варьируется в широком диапазоне значений и задается параметрически. Для этой цели подходят такие задачи, про которые известно, что для них не существует иных, более эффективных, методов

[1] Раунд состоит из запроса и ответного подтверждения. Полуторараундный протокол включает дополнительное сообщение-квитанцию в ответ на подтверждение.

решения. Например, никакие выполненные заранее предвычисления не способствуют снижению трудоемкости. Решение может быть найдено исключительно при помощи *силовой атаки*, т. е. методом проб и ошибок с исчерпывающим перебором вариантов. Для обозначения таких задач воспользуемся термином *шарада* (puzzle). Идеологическая подоплека метода шарад, а также исследование их свойств восходят к пионерской работе Р. Меркля [4].

Назовем *экзаменатором* того, кто создает шараду и по построению располагает ее решением, а *экзаменуемым* того, кто выполняет отыскание решения по заданию экзаменатора.

В работе [5] предложен практический метод противодействия сетевой атаке. Сервер предлагает решить шараду каждый раз, когда устанавливается соединение. Память под соединение выделяется только при условии предоставления правильного решения. Атакующий продуцирует запросы на установление соединения. Поскольку число таких запросов аномально велико, и в этом суть атаки, то и число шарад также велико. Искусственно созданная сетевая нагрузка возвращается к атакующему в виде вычислительной нагрузки, и для достижения поставленной цели он вынужден инвестировать. Тогда DoS-атака перестает быть беззатратной и атакующий вынужден платить за ее осуществление. Добросовестный пользователь, в отличие от атакующего, продуцирует умеренное число запросов, и для него вычислительная нагрузка не обременительна. Следует подчеркнуть, что эффективность механизма противодействия обусловлена исключительно разницей в количестве запросов на установление соединения. В работах [6—8] метод шарад применяется для противодействия DoS-атаке в ряде других приложений.

Сформулируем набор требований к шарадам в контексте DoS-атаки.

1. Собственно шарады не должны быть инструментом атаки. Вычислительная трудоемкость построения шарады и проверки ее решения не должна быть чрезмерной.

2. Трудоемкость решения шарады должна быть регулируемой. Необходимо, чтобы сервер имел возможность гибко настраивать трудоемкость, оперативно реагируя на увеличение или снижение сетевой нагрузки.

3. Решение шарады возможно при наличии определенного вычислительного потенциала. Алгоритм решения должен быть задан строго. Трудоемкость решения должна быть ограничена сверху. Решение может быть получено за конечное время. Не должно существовать известных методов повышения эффективности решения.

Недостатки метода шарад

Известные шарады [5, 9] допускают возможность отыскания решения независимыми вычислителями, причем каждый из таких вычислителей выполняет поиск в пределах некоторого подмножества претендентов, мощность которого меньше мощности исходного множества. Назовем такой подход к поиску решения *распараллеливанием*. На практике это означает, что атакующий может воспользоваться методом распределенных вычислений. Тогда применение N независимых вычислителей или вычислителя с N-ядерной архитектурой позволяет получить результат в N раз быстрее. Очевидно, что для организации таких вычислений атакующему необходимо выполнить предварительную подготовку заданий с последующим их распределением при помощи специализированного протокола. Как только решение найдено одним из вычислителей, все остальные должны по команде прекратить обработку заданий.

Напротив, шарады с последовательным алгоритмом решения [10] не допускают распараллеливания, и в этом их бесспорное преимущество. Однако эти шарады уязвимы с точки зрения атаки при помощи квантового компьютера. Известно, что трудоемкость отыскания решения шарады [10] может быть эффективно снижена в результате факторизации. Метод с полиномиальной трудоемкостью, известный как алгоритм факторизации Шора [11, 12], в существенной мере использует принцип квантовых вычислений. Объявлено о создании прототипа программируемого квантового компьютера с ограниченными вычислительными возможностями [13]. Известен также пример факторизации при помощи алгоритма Шора на реальном квантовом вычислителе [14]. Для эффективной факторизации числа из k двоичных разрядов понадобится квантовый вычислитель с регистром на $K \approx 2k$ квантовых состояний (кубит) [15]. Логично ожидать появления полноценного квантового компьютера в ближайшие десятилетия. Шарады [9] не только эффективно решаются при помощи квантового вычислителя [11], но также допускают распараллеливание.

Выделим следующие типы шарад с регулируемой трудоемкостью решения.

1. Шарады, допускающие распараллеливание с применением квантовых/неквантовых вычислителей, для которых неизвестен эффективный квантовый алгоритм решения [5].

2. Шарады, допускающие распараллеливание с применением квантовых/неквантовых вычислителей, для которых известен эффективный квантовый алгоритм решения [9].

3. Шарады, не допускающие распараллеливания с применением квантовых/неквантовых вычислителей, для которых известен эффективный квантовый алгоритм решения [10].

Таким образом, к недостаткам шарад перечисленных типов следует отнести **возможность распараллеливания и существование эффективного квантового алгоритма решения**.

Следует также упомянуть отсутствие строгого доказательства того факта, что фундаментальные задачи, лежащие в основе известных шарад, относятся к классу *NP*-трудных. По мнению ряда специалистов, наличие подобного доказательства предоставляет достаточные гарантии относительной неуязвимости в долгосрочной перспективе.

Постановка задачи

Задача заключается в конструировании шарады с регулируемой трудоемкостью отыскания решения, максимально широким диапазоном трудоемкости с возможностью плавной регулировки, минимальными объемом памяти и накладными расходами при передаче, которая не поддается эффективному распараллеливанию и для которой не известен эффективный квантовый алгоритм. Для решения задачи воспользуемся методами теории помехоустойчивого кодирования и линейной алгебры.

Кодовые шарады

Мотивация подхода в том, что вопрос об эффективном решении на квантовом компьютере фундаментальных задач теории помехоустойчивого кодирования в настоящее время остается открытым. Более того, в работе [16] показано, что эти задачи относятся к классу *NP*-трудных. Известно также [17], что трудоемкость зашифрования для кодовой асимметричной криптосистемы ниже, чем трудоемкость зашифрования для криптосистемы RSA. Логично предположить, что шарады на основе кодов не превосходят по трудоемкости построения шарады из работы [10] и сравнимы с шарадами из работы [9].

Пусть имеется k-мерный линейный код C с минимальным расстоянием d [18]. Код задан $k\times n$ порождающей матрицей $\mathbf{G}$. Тогда существует кодовое слово $\mathbf{c}=\mathbf{pG}$, $\mathbf{c}\in\ker(\mathbf{H})$, где $\mathbf{H}$ — $(n-k)\times n$ проверочная матрица кода C и $\mathbf{p}$ — информационная последовательность. Множество $\mathbb{F}_q^n=\{\mathbf{x}=(x_1, \ldots, x_n)\,|\,x_j\in\mathbb{F}_q\}$ рассматривается как линейное пространство размерности n над $\mathbb{F}_q$.

Через $h(\cdot)$ обозначим криптографическую хэш-функцию $f_\lambda:\{0,1\}^*\to\{0,1\}^\lambda$ для некоторого λ. Свойства $h(\cdot)$ подробно описаны [19]. Отметим, что $h(\cdot)$ доступна как экзаменатору, так и экзаменуемому.

Назовем *кодовыми* шарады, построенные на основе кода C. Отметим, что код C известен как экзаменатору, так и экзаменуемому.

Для заданного $[n, k, d]_q$-кода экзаменатор выполняет следующие действия.

1. Выбирает информационную последовательность $\mathbf{p}\in_R\mathbb{F}_q^k$.
2. Сохраняет $\varphi=h(\mathbf{p})$ для проверки решения.
3. Выбирает ошибку $\mathbf{e}\in_R\mathbb{F}_q^n$ такую, что вес Хэмминга $wt(\mathbf{e})=t$ и $\lceil(d-1)/2\rceil<t<\lceil n/2\rceil$.
4. Вычисляет значение хэш-функции $\psi=h(\mathbf{p}\,\|\,\mathbf{e})$.
5. Вычисляет сумму $\check{\mathbf{c}}=\mathbf{pG}+\mathbf{e}$.
6. Передает $\{\check{\mathbf{c}}, \psi, t\}$ экзаменуемому.

Для поиска решения экзаменуемый выполняет следующие действия.

1. Выбирает ошибку $\mathbf{e}\in\mathbb{F}_q^n$ такую, что $wt(\mathbf{e})\le t$.
2. Вычисляет сумму $\hat{\mathbf{c}}=\check{\mathbf{c}}+\mathbf{e}$.
3. Выполняет декодирование $\hat{\mathbf{c}}$ и получает $\hat{\mathbf{p}}$.
4. Проверяет $h(\hat{\mathbf{p}}\,\|\,\mathbf{e})\stackrel{?}{=}\psi$.
5. Если $h(\hat{\mathbf{p}}\,\|\,\mathbf{e})=\psi$, то переходит к 6, иначе к 1.
6. Предъявляет $\varphi'=h(\hat{\mathbf{p}})$ в качестве решения.

Заметим, что в кодовой шараде секретный ключ состоит из двух компонент – информационной последовательности $\mathbf{p}$ и ошибки $\mathbf{e}$. Если известно $\mathbf{p}$, то легко вычислить $\mathbf{e}$, так как $\mathbf{e}=\check{\mathbf{c}}+\mathbf{pG}$, и наоборот.

Вес t всегда должен превышать половину кодового расстояния на величину ε, которая обеспечивает маскировку кода с алгоритмом декодирования полиномиальной трудоемкости под код, для которого возможно только корреляционное декодирование. Размерность кода следует выбирать так, чтобы трудоемкость поиска решения перебором по $\mathbf{p}$ превышала трудоемкость поиска решения перебором по $\mathbf{e}$. Действительно, если

$$q^k<\sum_{i=\lceil(d-1)/2\rceil+\varepsilon}^{t}(q-1)^i\binom{n}{i}, \qquad (1)$$

то экзаменуемый найдет решение шарады $\{\check{\mathbf{c}}, \psi, t\}$ в среднем за q^{k-1} испытаний, воспользовавшись следующим методом.

1. Выбирает информационную последовательность $\hat{\mathbf{p}}\in\mathbb{F}_q^k$.
2. Вычисляет кодовое слово $\hat{\mathbf{c}}=\hat{\mathbf{p}}\mathbf{G}$.
3. Вычисляет сумму $\mathbf{e}=\hat{\mathbf{c}}+\check{\mathbf{c}}$.
4. Проверяет $h(\hat{\mathbf{p}}\,\|\,\mathbf{e})\stackrel{?}{=}\psi$.
5. Если $h(\hat{\mathbf{p}}\,\|\,\mathbf{e})=\psi$, то переходит к 6, иначе к 1.
6. Предъявляет $\varphi'=h(\hat{\mathbf{p}})$ в качестве решения.

Использование критерия $\psi=h(\mathbf{p}\,\|\,\mathbf{e})$ не позволяет экзаменуемому получить искомое решение за меньшее число испытаний. Предположим, $\psi=h(\mathbf{p})$ и необходимо найти $\mathbf{p}$. Пусть выбрано $\check{\mathbf{e}}$

такое, что $wt(\check{\mathbf{e}}+\mathbf{e}) \le \lfloor (d-1)/2 \rfloor$. Если вес ошибки не превышает половины кодового расстояния, то в результате декодирования последовательность $\mathbf{p}$ будет восстановлена правильно и решение будет получено при $\check{\mathbf{e}} \ne \mathbf{e}$. Очевидно, что $h(\mathbf{p} \| \check{\mathbf{e}}) = \psi$ только при $\check{\mathbf{e}} = \mathbf{e}$.

Кодовая шарада должна допускать регулировку трудоемкости отыскания решения. Вес Хэмминга t ошибки $\mathbf{e}$ — один из параметров, определяющий объем перебора при поиске решения. Обозначим через $\mathcal{R}$ множество всевозможных претендентов. Известно, что $|\mathcal{R}|$ достигает своего максимума при $t = \lfloor n/2 \rfloor$. Например, для $q = 2$ существует $\sum_{i=\lfloor (d-1)/2 \rfloor + \varepsilon}^{t} \binom{n}{i} \le \frac{n}{(n/2-t)^2} 2^{n-3}$ претендентов на $\mathbf{e}$. Для поиска решения при $t = n/2 - 1$ экзаменуемому потребуется выполнить в среднем не более $n2^{n-4}$ испытаний. С другой стороны, следует выбирать параметры кода так, чтобы трудоемкость отыскания решения варьировалась в широком диапазоне. Из указанных ограничений следует, что $\lfloor (d-1)/2 \rfloor + \varepsilon \le t \le \lfloor n/2 \rfloor$ и

$$\lfloor (d-1)/2 \rfloor + \varepsilon \le \lfloor n/2 \rfloor - 1. \qquad (2)$$

Например, если предположить равенство в (2), то при фиксированном n существует единственная шарада с максимальной трудоемкостью.

Назовем шараду *устойчивой*, если не существует иного, менее трудоемкого способа ее решения, кроме заданного по построению. Необходимо определить допустимый диапазон скоростей кода такой, чтобы гарантировать устойчивость кодовой шарады и обеспечить максимально широкий диапазон трудоемкости.

Чем меньше минимальное кодовое расстояние d, тем шире диапазон трудоемкости. Очевидно, что d обратно пропорционально размерности кода. Это означает, что для конструирования шарад предпочтительнее высокоскоростные коды, для которых отношение $R = k/n$ стремится к 1.

Пусть задан $[n, n-d+1, d]_q$-код Рида—Соломона $RS_q(n, d)$ над $\mathbb{F}_q$, $q = p^m$, где p — простое число; m — положительное целое, который имеет максимально возможную размерность при заданных n и d [18]. Тогда $d = n-k+1$ и код исправляет $t \le \lfloor (n-k)/2 \rfloor$ ошибок. Существуют $RS_q(n, d)$-коды с блоковой длиной $n = q-1$, расширенные с $n = q$ и дважды расширенные с $n = q+1$. Это означает, что всегда можно выбрать код с четным n. Шараду на основе $RS_q(n, d)$-кода будем называть $RS_q(n, d)$-шарадой.

Не существует обоснованных возражений против использования безызбыточных кодов. Тогда $RS_q(n, 1)$-шарада обладает максимально широким диапазоном трудоемкости, поскольку $\varepsilon = 0$ и вес ошибки варьируется в интервале $n/2 \ge t \ge 1$. Шарада безусловно устойчива, так как $k > n/2$. Следовательно, верхняя граница скорости кода совпадает с конструктивным ограничением.

Нижняя граница может быть получена при помощи следующих простых рассуждений. Очевидно, что при $k \le n/2$ наблюдается объективное сужение диапазона трудоемкости, поскольку вес ошибки $\lfloor (d-1)/2 \rfloor + \varepsilon \le t < k$. Если $k < t \le n/2$, то шарада неустойчива. Это объясняется тем, что справедливо неравенство (1) и трудоемкость отыскания решения будет ниже запланированной. Таким образом, для построения устойчивых $RS_q(n, d)$-шарад с широким диапазоном трудоемкости следует использовать коды со скоростями в интервале $0{,}5 < R \le 1$. $RS_q(n, 1)$-шарады представляются наиболее перспективными с практической точки зрения.

Отметим, что для $RS_q(n, 1)$-шарад ограничение на блоковую длину не является критичным, поскольку трудоемкость поиска решения в существенной степени определяется весом t.

$RS_q(n, d)$-шарады допускают распараллеливание. Попытаемся устранить этот недостаток. Воспользуемся следующим наблюдением. Как было отмечено, для организации параллельных вычислений необходимо выполнить распределение заданий. Если такое распределение выполняется однократно, то возникающими накладными расходами можно пренебречь. Предположим, что шарада состоит из нескольких подшарад. Процесс распределения заданий усложнится, если скомбинировать подшарады таким образом, что решение каждой последующей будет зависеть от решения предыдущей. Иначе говоря, атакующий будет вынужден распределять задания для каждой подшарады. При этом важно, чтобы подшарады не были известны заранее и раскрывались последовательно по мере получения промежуточных решений.

Сконструируем $RS_q^{\ell}(n, 1)$-шараду, состоящую из ℓ вложенных подшарад. Для этого зададим вес ошибки t_j для каждой из ℓ подшарад. В результате получим набор $\{t_1, \ldots, t_\ell\}$. Напомним, что для $RS_q(n, 1)$-шарады $t \in [1, n/2]$.

Для построения $RS_q^{\ell}(n, 1)$-шарады экзаменатор выполняет следующие действия.

1. Выбирает информационную последовательность $\mathbf{p} \in {}_R\mathbb{F}_q^n$.

2. Сохраняет $\varphi = h(\mathbf{p})$ для проверки решения.

3. Устанавливает $j := 1$ и $\check{\mathbf{p}} := \mathbf{p}$.

4. Выбирает ошибку $\mathbf{e} \in {}_R\mathbb{F}_q^n$ такую, что $1 \le wt(\mathbf{e}) \le t_j$.

5. Вычисляет значение хэш-функции $\psi_j = h(\check{\mathbf{p}} \| \mathbf{e})$.

6. Вычисляет $\check{\mathbf{c}} = \check{\mathbf{p}}\mathbf{G} + \mathbf{e}$.

7. Устанавливает $j := j+1$ и $\check{\mathbf{p}} := \check{\mathbf{c}}$.

8. Проверяет $j \stackrel{?}{=} \ell + 1$.

9. Если $j = \ell + 1$, то переходит к 10, иначе к 4.

10. Передает $\{\check{\mathbf{c}}, \ell, \{\psi_1, ..., \psi_\ell\}, \{t_1, ..., t_\ell\}\}$ экзаменуемому.

Для поиска решения экзаменуемый выполняет следующие действия.

1. Устанавливает $j := \ell$ и $\mathbf{p} := \check{\mathbf{c}}$.

2. Выбирает ошибку $\mathbf{e} \in \mathbb{F}_q^n$ такую, что $1 \le wt(\mathbf{e}) \le t_j$.

3. Вычисляет сумму $\tilde{\mathbf{c}}$.

4. В результате декодирования $\tilde{\mathbf{c}}$ получает $\check{\mathbf{p}}$.

5. Проверяет $h(\check{\mathbf{p}} \| \mathbf{e}) \stackrel{?}{=} \psi_j$.

6. Если $h(\check{\mathbf{p}} \| \mathbf{e}) = \psi_j$, то переходит к 7, иначе к 2.

7. Устанавливает $j := j - 1$ и $\mathbf{p} := \check{\mathbf{p}}$.

8. Проверяет $j \stackrel{?}{=} 0$.

9. Если $j = 0$, то переходит к 10, иначе к 2.

10. Предъявляет $\varphi' = h(\check{\mathbf{p}})$ в качестве решения.

Конструкция $RS_q^\ell(n, 1)$-шарады такова, что объем памяти для хранения результирующей шарады равен объему памяти для хранения отдельной подшарады. Однако потребуется дополнительная память для хранения ℓ пар $\{\psi_j, t_j\}$. Заметим, что все ψ_j имеют одинаковую разрядность λ и $t_j \le n/2$. Тогда для хранения $RS_q^\ell(n, 1)$-шарады при $n = p^m$ необходимо зарезервировать $O\left(nm\log_2 p + \ell\left(\frac{nm\log_2 p}{2} + \lambda\right)\right)$ двоичных разрядов.

Из представленной конструкции следует, что применение безызбыточного кода вполне оправдано — объем памяти для хранения подшарад не зависит от ℓ. Однако для ψ_j и t_j объем памяти растет линейно по ℓ. С точки зрения резервирования памяти вклад ψ_j и t_j неравнозначен. Как правило, веса ошибок подчиняются монотонной зависимости. Следовательно, можно хранить не веса ошибок, а описание функции, при помощи которой несложно вычислить произвольное t_j. Объем памяти для хранения такого описания не зависит от ℓ. Будем считать, что накладными расходами, связанными с хранением t_j, можно пренебречь. Для хранения ψ_j, в противоположность t_j, может потребоваться относительно большой объем памяти. Например, если воспользоваться хэш-функцией SHA-256, то при $\ell = 100$ и $n = 2^8$ объем памяти для хранения всех ψ_j на порядок превысит объем памяти для хранения $\check{\mathbf{c}}$ и t_j, $1 \le j \le \ell$.

Рассмотрим такую конструкцию шарады, у которой объем памяти для хранения ψ_j не зависит от ℓ.

Назовем *итеративным хэшированием* преобразование вида $\psi_{\ell-1} = h(\underbrace{h(...h(h(s))...)}_{\ell \text{ раз}})$, где ℓ — число итераций. Финальное значение $\psi_{\ell-1}$ получается из стартового s.

Понадобится также следующее свойство кода C. Пусть $\mathbf{c}_1, \mathbf{c}_2 \in C$. Тогда $\mathbf{c}_3 = \mathbf{c}_1 + \mathbf{c}_2 = (\mathbf{p}_1 + \mathbf{p}_2)\mathbf{G}$ и $\mathbf{c}_3 \in C$.

Подойдем к конструированию шарады следующим образом. Вначале зададим вес ошибки t_j для каждой из ℓ подшарад. Затем для $s \in {}_R\mathbb{Z}$ методом итеративного хэширования вычислим $\psi_{\ell-1}$, $\psi_{\ell-2}$, ..., ψ_1, ψ_0. Отобразим каждый ψ_j на линейное пространство размерности n над $\mathbb{F}_q$. Предположим, $n = p^m$, $b = \left\lceil \frac{\lambda}{m\log_2 p} \right\rceil$, $1 \le b \le n$ и каждый ψ_j состоит из b q-ичных символов. В результате отображения получим $\Psi_j = \mathbf{0}_q^{n-b} \| \psi_j$, где $\mathbf{0}_q^{n-b}$ — последовательность из $n-b$ нулевых символов поля $\mathbb{F}_q$ и $\Psi_j \in \mathbb{F}_q^n$, $0 \le j \le \ell - 1$.

Заданы наборы $\{\Psi_{\ell-1}, ..., \Psi_1, \Psi_0\}$ и $\{t_1, ..., t_\ell\}$. Для построения $RS_q^\ell(n, 1)^*$-шарады экзаменатор выполняет следующие действия.

1. Выбирает информационную последовательность $\mathbf{p} \in {}_R\mathbb{F}_q^n$.

2. Сохраняет $\varphi = h(\mathbf{p})$ для проверки решения.

3. Устанавливает $j := 1$ и $\check{\mathbf{p}} := \mathbf{p}$.

4. Выбирает ошибку $\mathbf{e} \in {}_R\mathbb{F}_q^n$ такую, что $1 \le wt(\mathbf{e}) \le t_j$.

5. Вычисляет $\check{\mathbf{c}} = (\check{\mathbf{p}} + \Psi_{\ell-j})\mathbf{G} + \mathbf{e}$.

6. Устанавливает $j := j + 1$ и $\check{\mathbf{p}} := \check{\mathbf{c}}$.

7. Проверяет $j \stackrel{?}{=} \ell + 1$.

8. Если $j = \ell + 1$, то переходит к 9, иначе к 4.

9. Передает $\{\check{\mathbf{c}}, \ell, s, \{t_1, ..., t_\ell\}\}$ экзаменуемому.

Экзаменуемый выполняет следующие действия.

1. Устанавливает $j := \ell$, $\mathbf{p} := \check{\mathbf{c}}$ и $\hbar := h(s)$.

2. Выбирает ошибку $\mathbf{e} \in \mathbb{F}_q^n$ такую, что $1 \le wt(\mathbf{e}) \le t_j$.

3. Вычисляет сумму $\tilde{\mathbf{c}} = \mathbf{p} + \mathbf{e}$.

4. В результате декодирования $\tilde{\mathbf{c}}$ получает $\check{\mathbf{p}}$.

5. Отображает $\Psi_{\ell-j} = \mathbf{0}_q^{n-b} \| \hbar$.

6. Проверяет $(\check{\mathbf{p}} + \Psi_{\ell-j})\mathbf{G} \stackrel{?}{=} \tilde{\mathbf{c}} + \Psi_{\ell-j}\mathbf{G}$.

7. Если $(\check{\mathbf{p}} + \Psi_{\ell-j})\mathbf{G} = \tilde{\mathbf{c}} + \Psi_{\ell-j}\mathbf{G}$, то переходит к 8, иначе к 2.

8. Устанавливает $j := j - 1$, $\mathbf{p} := \check{\mathbf{p}} + \Psi_{\ell-j}$ и $\hbar := h(\hbar)$.

9. Проверяет $j \stackrel{?}{=} 0$.

10. Если $j = 0$, то переходит к 11, иначе к 2.

11. Предъявляет $\varphi' = h(\mathbf{p})$ в качестве решения.

Предположим, что для представления числа s в памяти достаточно λ двоичных разрядов. Тогда для хранения $RS_q^\ell(n, 1)^*$-шарады потребуется зарезервировать не более $O\left(nm\log_2 p + \frac{\ell nm\log_2 p}{2} + \lambda\right)$ двоичных разрядов.

Трудоемкость отыскания решения не превышает

$$\sum_{j=1}^{\ell}\sum_{i=1}^{t_j}(q-1)^i\binom{n}{i} \qquad (3)$$

испытаний.

Проанализируем $RS_q^{\ell}(n, 1)^*$-шараду на устойчивость. Соответствующее итеративное преобразование можно представить в виде

$$\check{\mathbf{c}} = ((((...((((\mathbf{p}+\Psi_{\ell-1})\mathbf{G}+\mathbf{e}_1)+\Psi_{\ell-2})\mathbf{G}+\mathbf{e}_2)+ \\ +...+\Psi_1)\mathbf{G}+\mathbf{e}_{\ell-1})+\Psi_0)\mathbf{G}+\mathbf{e}_{\ell}),$$

где $\mathbf{p}, \mathbf{e}_j \in {}_R\mathbb{F}_q^n$ и $\Psi_j \neq \Psi_i$ для $i \neq j$, $0 \le i, j \le \ell$.

Необходимо ответить на следующий вопрос: способен ли экзаменуемый, располагая набором $\{\check{\mathbf{c}}, \ell, s, \{t_1, ..., t_\ell\}\}$, а также последовательностью $\Psi_{\ell-1}$, ..., Ψ_1, Ψ_0, уменьшить запланированную трудоемкость поиска решения?

Конструкция $RS_q^{\ell}(n, 1)^*$-шарады напоминает луковицу. Доступ к некоторому внутреннему слою возможен только после удаления всех объемлющих слоев. Каждый слой ассоциирован с отдельной подшарадой. Слой с номером j считается удаленным, если найдено решение для j-й подшарады. Поиск решения начинается с первого внешнего слоя, который всегда доступен. Собственно решение шарады располагается в сердцевине луковицы — в последнем внутреннем слое.

Если воспользоваться геометрической интерпретацией, то каждое кодовое слово $RS_q(n, 1)$-кода располагается в центре сферы нулевого радиуса и все такие сферы не пересекаются. Число сфер равно числу кодовых слов, которое для $RS_q(n, 1)$-кода совпадает с мощностью $\mathbb{F}_q^n$. Тогда произвольная ошибка веса $0 < t \le n$ переводит кодовое слово $RS_q(n, 1)$-кода в другое кодовое слово того же кода с единичной вероятностью. Это означает, что каждая подшарада $RS_q^{\ell}(n, 1)^*$-шарады имеет единственное решение.

При заданном s несложно вычислить $\Psi_{\ell-j}$, которое используется в качестве критерия. Если $\mathbf{c} = (\mathbf{p}+\Psi_{\ell-j})\mathbf{G}$, то в результате декодирования будет получена информационная последовательность $\mathbf{I} = \mathbf{p}+\Psi_{\ell-j}$. Из линейности кода следует, что $(\mathbf{I}+\Psi_{\ell-j})\mathbf{G} = \mathbf{c}+\Psi_{\ell-j}\mathbf{G}$. По построению кодовое слово $\mathbf{c}$ маскируется ошибкой $\mathbf{e}$, $1 \le wt(\mathbf{e}) \le t_j$, и $\hat{\mathbf{c}} = \mathbf{c}+\mathbf{e}$. Решение j-й подшарады заключается в нахождении ошибки $\mathbf{e}$. Пусть заданы $\hat{\mathbf{c}}$, $\Psi_{\ell-j}$ и некоторая ошибка $\check{\mathbf{e}} \neq \mathbf{e}$. Для $\check{\mathbf{c}} = \hat{\mathbf{c}}+\check{\mathbf{e}}$ в результате декодирования будет получена информационная последовательность $\check{\mathbf{I}} \neq \mathbf{I}$. Решение $\check{\mathbf{e}}$ будет отвергнуто, так как $(\check{\mathbf{I}}+\Psi_{\ell-j})\mathbf{G} \neq \check{\mathbf{c}}+\Psi_{\ell-j}\mathbf{G}$.

Поскольку применяется безызбыточный код, то для j-й подшарады существует q^n кодовых слов. При $1 \le t_j \le n/2$ справедливо неравенство $q^n > \sum_{i=1}^{t_j}(q-1)^i\binom{n}{i}$, и для отыскания решения исчерпывающий перебор ошибок ограниченного веса выгоднее, чем исчерпывающий перебор информационных последовательностей.

Помимо ширины диапазона значение также имеет функция, при помощи которой задается трудоемкость. Для шарад [5] трудоемкость отыскания решения определяется мощностью множества возможных претендентов и равна 2^r для некоторого параметра r. В ряде случаев экспоненциальное изменение трудоемкости не адекватно воздействию и поэтому не оправдано. Необходима плавная регулировка. $RS_q^{\ell}(n, 1)^*$-шарады допускают такую регулировку за счет полиномиальной по n функции изменения трудоемкости для каждой подшарады и общей линейной зависимости. Как было показано (3), трудоемкость отыскания решения для $RS_q^{\ell}(n, 1)^*$-шарады задается аддитивной функцией и допускает гибкую настройку шага изменения трудоемкости. Очевидно, что $\binom{n}{i+1} = \binom{n}{i}\frac{n-i}{i+1}$, $1 \le i \le n$. Тогда при увеличении/уменьшении на единицу веса t ошибки $\mathbf{e}$ трудоемкость отыскания решения возрастает/убывает в $(n-t)/(t+1)$раз. Поскольку $\binom{n}{1} = n$, то для $RS_q^{\ell}(n, 1)^*$-шарады минимальный шаг изменения трудоемкости равен n. Например, можно сконструировать шараду со средней трудоемкостью отыскания решения $\frac{\ell n}{2}$.

Заключение

В статье рассматривается метод шарад как способ противодействия DoS-атаке. Предложены конкретные конструкции шарад на основе кодов, исправляющих ошибки, в том числе и итеративная конструкция, которая гарантирует устойчивость, обладает широким диапазоном трудоемкости и допускает гибкую настройку.

Литература

1. **Naraine R.** Massive DDOS Attack Hit DNS Root Servers. Oct. 23, 2002. http://siliconvalley.internet.com/news/article.php/1486981 (дата обращения: 10.06.2010).
2. **Wagner J.** SCO Hit By Another DDoS Attack. Dec. 10, 2003. http://www.internetnews.com/dev-news/article.php/3287781 (дата обращения: 10.06.2010).

3. **Schuba C. L.** et al. **Analysis of a denial of service attack on TCP** // Proc. IEEE Symp. on Security and Privacy. IEEE Computer Society Press, May 1997. P. 208–223.
4. **Merkle R. C.** Secure Communications Over Insecure Channels // Communications of the ACM. Apr. 1978. Vol. 21. N 4. P. 294–299.
5. **Brainard J., Juels A.** Client Puzzles: A Cryptographic Countermeasure Against Connection Depletion Attacks // Proc. of the 1999 ISOC Network and Distributed System Security Symp. 1999. P. 151–165.
6. **Aura T., Nikander P., Leiwo J.** DoS-resistant authentication with client puzzles // 8th Intern. Workshop on Security Protocols. Springer-Verlag, 2000. P. 170–181.
7. **Dean D., Stubblefield A.** Using client puzzles to protect TLS//SSYM'01 Proc. of the 10th Conf. on USENIX Security Symp./ USENIX Association Berkeley. CA, USA, 2001. P. 1–8.
8. **Adkins D., Lakshminarayanan K., Perrig A., Stoica I.** Taming IP packet flooding attacks // Computer Communication Review. 2004. Vol. 34. N 1. P. 45–50.
9. **Waters B., Juels A., Halderman A., Felten E.** New Client Puzzle Outsourcing Techniques for DoS Resistance // ACM CCS. 2004. P. 246–256.
10. **Rivest R. L., Shamir A., Wagner D.** Time-lock puzzles and timed-release crypto // Technical Report MIT/LCS/TR-684. MIT, 1996. http://people.csail.mit.edu/rivest/RivestShamirWagner-timelock.pdf (дата обращения: 22.10.2010).
11. **Shor P. W.** Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer // SIAM J. of Computing. 1997. N 26. P. 1484–1509.
12. **Ekert A., Jozsa R.** Quantum computation and Shor's factoring algorithm // Rev. Mod. Phys. 1996. Vol. 68. N 3. P. 733–753.
13. **Hanneke D.** et al. Realization of a programmable two-qubit quantum processor // Nature Physics. 2009. N 6. P. 13–16.
14. **Vandersypen L. M. K.** et al. Experimental realization of Shor's quantum factoring algorithm using nuclear magnetic resonance // Nature. 2001. N 414. P. 883–887. http://www.nature.com/nature/journal/v414/n6866/abs/414883a.html (дата обращения: 22.10.2010).
15. **Beauregard S.** Circuit for Shor's algorithm using 2n + 3 qubits // Quantum Information and Computation. 2003. N 3. P. 175–185.
16. **Barg A.** Complexity Issues in Coding Theory // Electronic Colloquium on Computational Complexity (ECCC). 1997. Vol. 4. N 46. http://www.eccc.uni-trier.de/report/1997/046/ (дата обращения: 22.10.2010).
17. **Riek J. R.** Observations on the Application of Error-Correcting Codes to Public Key Encryption // Proc. of the IEEE 1990 Intern. Carnahan Conf. on Security Technology, Crime Countermeasures. 1990. P. 15–18.
18. **МакВильямс Ф. Д., Слоэн Н. Дж. Теория кодов, исправляющих ошибки.** — М.: Связь, 1979. — 744 с.
19. **Menezes A., van Oorschot P., Vanstone S.** Handbook of Applied Cryptography. 5th print. — CRC-Press, 2001. — 780 p. http://www.cacr.math.uwaterloo.ca/hac/about/chap9.pdf (дата обращения: 22.10.2010).