|  |  |  |
| --- | --- | --- |
|  | gerb |  |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |

Институт Кибернетики

Кафедра информационной безопасности

|  |  |
| --- | --- |
| **КУРСОВАЯ РАБОТА** | |
| **по дисциплине** | |
| «Криптографические методы защиты информации» | |
| **Тема курсовой работы:**  **«Cовременные подходы к преодолению парольной защиты»** | |
| Студент группы ККСО-04-19 | *Савилов Д. А.* |
| Руководитель курсовой работы | *Хомутов Д.Г.* |
|  |  |

|  |  |  |
| --- | --- | --- |
| Работа представлена к защите | «\_\_»\_\_\_\_\_\_\_2023 г. | *(подпись студента)* |
|  |  |  |
| Допущен к защите | «\_\_»\_\_\_\_\_\_\_2023 г. | *(подпись руководителя)* |

Москва   
2023

Оглавление

[Введение 3](#_Toc103717980)

[Постановка проблемы 4](#_Toc103717981)

[Параметры количественной оценки стойкости пароля 5](#_Toc103717982)

[Cовременные подходы к преодолению парольной защиты 7](#_Toc103717983)

[Обход парольной системы в аспекте хранения паролей в базе данных 7](#_Toc103717984)

[Обход парольной системы в аспекте передачи по сети 15](#_Toc103717985)

[Основные атаки на криптографические протоколы 15](#_Toc103717986)

[Противодействие обходу парольной защиты. 18](#_Toc103717987)

[Общие рекомендации 19](#_Toc103717988)

[Методы противодействия подбору пароля при хранении пароля в хэшированном виде 20](#_Toc103717989)

[Методы противодействия перехвату пароля при передаче по сети 24](#_Toc103717990)

[Заключение 27](#_Toc103717991)

[Список литературы 28](#_Toc103717992)

## Введение

Начиная с восьмидесятых годов компьютеры начали активно внедряться в нашу жизнь. Люди стали доверять им важнейшую информацию, попадание которой в чужие руки грозило тяжелыми последствиями. Однако информация оказалась незащищенной и остро стал вопрос о ее защите. Этот вопрос беспокоит как специалистов в области компьютерной безопасности, так и многочисленных рядовых пользователей персональных компьютеров. Это связано с глубокими изменениями, вносимыми компьютерной технологией в нашу жизнь. Изменился сам подход к понятию «информация». Информация по большей части стала товаром, который можно купить, продать, обменять на что-то другое и т.д. При этом стоимость подобного товара зачастую превосходит в десятки, а то и в сотни раз стоимость самой вычислительной техники, в рамках которой он функционирует. Более того, информация, в данное время, является одним из основных товаров. Естественно, возникает потребность защитить информацию от несанкционированного доступа, кражи, уничтожения, изменения и других преступных действий. Однако большая часть пользователей не осознает, что постоянно рискует безопасностью своей информации и личными тайнами. И лишь немногие, хоть каким-либо образом защищают свои данные. Пользователи компьютеров регулярно оставляют полностью незащищенными даже такие данные как налоговая и банковская информация, деловая переписка и электронные таблицы. Нередки случаи усложнения проблемы также из-за того, что появился тип людей, который не просто охотится за информацией в преступных целях, а делают это для развлечения. Также, ситуация осложняется, когда люди начинают работать в сети, так как злоумышленнику намного легче в это время заполучить или уничтожить информацию, находящуюся на компьютере. Существуют в различных законодательствах законы, в котором информация, безусловно, признается объектом собственности человека, и человек, как владелец своей информации имеет право определять правила ее обработки и зашиты, (например, Закон Российской Федерации "Об информации, информатизации и защите информации", принятый 25 января 1995 г). В соответствии с ним любой российский гражданин может предпринимать необходимые меры для предотвращения утечки, хищения, утраты, искажения и подделки информации. Вопрос состоит в том, какие действия являются на самом деле необходимыми для адекватной защиты информации. Запирать входные двери не очень удобно, однако без этого люди не выходят из дома. Во-первых, необходимо обладать минимумом технических знаний, чтобы подобрать и установить надежный замок. Во-вторых, требуется постоянный контроль за состоянием замка, чтобы содержать его в исправности. В-третьих, чтобы замок предотвращал проникновение в дом посторонних людей, вы должны соблюдать определенные правила (хранить ключи в надежном месте, а также не оставлять дверь незапертой). Подобные же правила применимы и в случае защиты информации в компьютерных системах. Именно поэтому так важно отыскать разумный компромисс между ценностью данных и неудобствами, связанными с использованием необходимых мер безопасности. И одним из самых распространенных программных средств защиты является парольная защита.

## Постановка проблемы

Для организации парольных систем требуется создать и поддерживать базу данных, содержащую пароли или свертки паролей всех пользователей. Их слабой стороной является то, что получение злоумышленником базы данных позволяет ему проходить аутентификацию от имени любого пользователя.

Хэширование не обеспечивает защиту от подбора паролей по словарю в случае получения базы данных злоумышленником. При выборе алгоритма хэширования который будет использован для вычисления сверток паролей, необходимо гарантировать несовпадение значений сверток, полученных на основе различных паролей пользователей. Кроме того, следует предусмотреть механизм, обеспечивающий уникальность сверток в том случае, если два пользователя выбирают одинаковые пароли. Для этого при вычислении каждой свертки обычно используют некоторое количество «случайной» информации, например выдаваемой генератором псевдослучайных чисел.

Способы, составляющие вторую разновидность, обеспечивают более высокую степень безопасности парольной системы, т. к. проверочные значения, хотя они и зависят от паролей, не могут быть непосредственно использованы злоумышленником для аутентификации.

Аутентификация без предоставления проверяющей стороне какой бы то ни было информации о пароле обеспечивает наибольшую степень защиты. Этот способ гарантирует безопасность даже в том случае, если нарушена работа проверяющей стороны Процесс аутентификации строится по типу «вопрос – ответ».

Особым подходом в технологии проверки подлинности являются криптографические протоколы аутентификации. Такие протоколы описывают последовательность действий, которую должны совершить стороны для взаимной аутентификации. Кроме того, эти действия, как правило, сочетаются с генерацией и распределением криптографических ключей для шифрования последующего информационного обмена. Корректность протоколов аутентификации вытекает из свойств задействованных в них математических и криптографических преобразований и может быть строго доказана.

### Параметры количественной оценки стойкости пароля

Статья [3] приводит следующие параметры количественной оценки стойкости парольных систем:

|  |  |
| --- | --- |
| Параметр | Способ определения |
| Скорость подбора паролей *V*.  Для интерактивного режима определяется как скорость обработки одной попытки регистрации проверяющей стороной.  Для режима *off-line* (на основе свертки пароля) определяется как скорость вычисления значения свертки для одного пробного пароля | Может быть искусственно снижена для защиты от данной угрозы.  Задается используемым алгоритмом вычисления свертки. Алгоритм, имеющий медленные реализации, повышает стойкость по отношению к данной угрозе |
| Мощность алфавита паролей *А* Длина пароля *L* | Могут варьироваться для обеспечения заданного значения *S* (*S* = *AL*) (\*\*) |
| Мощность пространства паролей *S* | Вычисляется на основе заданных значений *Р*, *T* или *V* по формуле (\*) |
| Срок действия пароля (задает промежуток времени, по истечении которого пароль должен быть обязательно сменен) *Т* | Определяется исходя из заданной вероятности *Р* или полагается заданным для дальнейшего определения *S* |
| Вероятность подбора пароля в течение его срока действия (подбор продолжается непрерывно в течение всего срока действия пароля) *Р* | Выбирается заранее для дальнейшего определения *S* или *Т* |

Таблица 2. Параметры количественной оценки стойкости пароля

Рассмотрим следующий иллюстрирующий пример [3]:

Пусть нам дана вероятность подбора . Необходимо найти минимальную длину пароля, обеспечивающую его стойкость в двух недель непрерывных попыток подобрать пароль со скоростью интерактивного подбора паролей в минуту

**Решение:**За месяц мы переберем паролей. Далее, исходя из формулы (\*), получим

Далее, мы смотрим на мощность нашего алфавита. Пусть мы используем только латинские буквы без учета регистров + цифры. Тогда имеем, согласно (\*\*),

Если же мы хотим обойтись без цифр, то, в теории, минимальная длина должна увеличиться, но в нашем случае она будет равна

Так же найденной мощности будут соответствовать, например, пары и

## Cовременные подходы к преодолению парольной защиты

Введение перечисленных выше количественных характеристик парольной системы позволяет рассмотреть вопрос о связи стойкости парольной системы с криптографической стойкостью в двух аспектах: при хранении паролей в базе данных и при их передаче по сети.

### Обход парольной системы в аспекте хранения паролей в базе данных

Возможны следующие варианты хранения паролей:

1. в открытом виде;
2. в виде сверток (хэширование);
3. зашифрованными на некотором ключе.

Наиболее безопасное хранение паролей обеспечивается при их хэшировании и последующем шифровании полученных сверток, т. е. при комбинации второго и третьего способов. Стойкость парольной системы определяется ее способностью противостоять атаке злоумышленника, завладевшего базой данных учетных записей и пытающегося восстановить пароли, и зависит от скорости максимально быстрой реализации используемого алгоритма хэширования. При хранении паролей в виде сверток стойкость парольной системы зависит от криптографических свойств алгоритма шифрования или хэширования паролей.

Если потенциальный злоумышленник имеет возможность перехватывать передаваемые по сети преобразованные значения паролей, при выборе алгоритма необходимо обеспечить невозможность (с заданной вероятностью) восстановить пароль при наличии достаточного количества перехваченной информации.

Рассмотрим следующие механизмы подбора пароля:

#### Метод прямого перебора (англ. Brute-Force)

Требует очень много времени для подбора, тем не менее, приносит 100% результат. Устойчивость к атаке полным перебором экспоненциально зависит от длины пароля. В простейшем случае пароль длиной в *N* битов взламывается, в наихудшем случае, за время, пропорциональное . Среднее время взлома в этом случае в два раза меньше и составляет 2*N*-1. Любой пароль может быть подобран путём полного перебора. При этом, даже если вычисление целевой функции от каждого конкретного возможного решения задачи может быть осуществлено за полиномиальное время, в зависимости от количества всех возможных решений атака может потребовать экспоненциального времени работы.

Для демонстрации временной неэффективности рассмотрим таблицу, приведенную в [1]:

|  |  |  |  |
| --- | --- | --- | --- |
| Кол-во знаков x | Кол-во вариантов | Стойкость | Время перебора |
| 1 | 36 | 5 бит | менее секунды |
| 2 | 1296 | 10 бит | менее секунды |
| 3 | 46 656 | 15 бит | менее секунды |
| 4 | 1 679 616 | 21 бит | 17 секунд |
| 5 | 60 466 176 | 26 бит | 10 минут |
| 6 | 2 176 782 336 | 31 бит | 6 часов |
| 7 | 78 364 164 096 | 36 бит | 9 дней |
| 8 |  | 41 бит | 11 месяцев |
| 9 |  | 46 бит | 32 года |
| 10 |  | 52 бита | 1 162 года |
| 11 |  | 58 бит | 41 823 года |
| 12 |  | 62 бита | 1 505 615 лет |

Таблица 1. Скорость и время перебора пароля в зависимости от длины  
Скорость перебора – 100000 паролей в секунду  
Размер алфавита – 36 символов (26 латинских прописных букв + 10 цифр)

Как видно из данного примера, пароли с длиной ниже 8 – ненадёжные, и в дальнейшем эта длина будет расти за счет ,например, следующих вариантов распараллеливания вычислений:

* **Построение алгоритма.**

Пусть алгоритм можно представить в виде цепочки простейших действий (операций). Пусть NN — количество процессоров, в которых запрограммирован порядок. i-ый процессор выполняет три одинаковые по времени операции:

1. получение данных от (i−1)-го процессора
2. выполнение операции
3. передача данных (i+1) -ому процессору.

Эта операция может занять всего сотую долю секунды. Тогда NN соединённых параллельно и синхронно работающих процессоров работают со скоростью (v/3так как всего три операции), где vv  — скорость выполнения одной операции одним процессором.

* **Разбивание на множества.**

Множество всех возможных ключей — KK  разбивается на непересекающиеся подмножества. Система с QQ процессорами перебирает ключи так, что, например, ii-ая машина осуществляет перебор ключей из подмножества Ki,i=1...Q Система прекращает работу, если одна машина подобрала правильный ключ. Но если каждый процессор начнёт вычисление не с первого возможного ключа, время перебора может увеличиться, но алгоритм упростится. Среднее число подборов в этом случае составит где K — число элементов во множестве ключей, а  — число процессоров.

#### Подбор по словарю (англ. Dictionary attack).

Словарная атака (dictionary attack) – это метод вскрытия пароля путем перебора возможных значений данной информации.Название атаки произошло благодаря тому, что основу множества перебираемых паролей составляют слова какого-либо языка. Словарные атаки базируются на использовании легко запоминаемых паролей, к которым часто относятся различные слова и их варианты (например, замена части букв слова на похожие по написанию цифры или спецсимволы).

Словарные атаки часто классифицируют как один из вариантов атаки методом «грубой силы, поскольку здесь также выполняется перебор вариантов пароля. По сравнению с методом «грубой силы» словарные атаки осуществляют перебор по существенно меньшему множеству возможных значений.

Существуют следующие виды словарных атак [2]:

1. **Словарная атака, основанная на предварительных вычислениях**

Смысл данного варианта словарной атаки состоит в предварительном вычислении хэш-значений паролей, принадлежащих какому-либо множеству.  
Путем такого вычисления хакер получает таблицу соответствий входящих в множество паролей и их хэш-значений

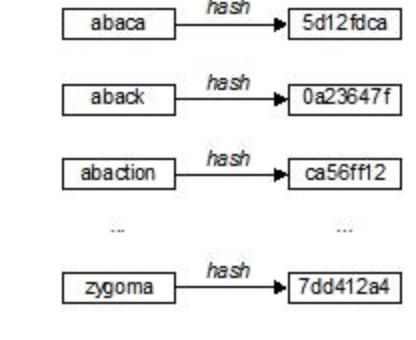


Рис. 1 Таблица хэш-значений

Для нахождения искомого пароля, соответствующего известному злоумышленнику хэш-значению h, выполняется поиск хэш-значений в таблице, в результате которого делается один из следующих выводов:

* если в таблице нашлась запись, соответствующая искомому значению h, то пароль p (для которого ) найден; при этом стоит учитывать, что, поскольку возможно возникновение коллизий найденный пароль может быть не искомым паролем, а паролем с эквивалентным искомому хэш-значением;
* если записи со значением h в таблице нет, то искомый пароль не принадлежит к множеству, покрываемому данной таблицей; можно продолжить поиск с помощью других таблиц или применением других методов.

Основным недостатком данной атаки является достаточно большой объем памяти, требуемой на хранение таблицы, размер которого в битах для n-битного алгоритма хэширования можно оценить следующим образом:

где P – множество паролей, покрываемое таблицей T, k – средний размер пароля в битах.

1. **Цепочки хэш-значений**

Использование цепочек хэш-значений (hash chains) – это улучшенный метод словарной атаки. Улучшение состоит в том, что при использовании цепочек на хранение таблицы паролей и соответствующих им хэш-значений требуется существенно меньше памяти, чем в случае классической словарной атаки.

Принцип действия цепочек хэш-значений основан на введении дополнительной функции , с помощью которой любому хэш-значению h можно псевдослучайным образом сопоставить некий пароль p из множества паролей P:

С помощью функции R() и функции хэширования  можно сформировать цепочки из паролей  и соответствующих им хэш-значений

(1)

Рассмотрим простой пример функции  если множеством паролей является множество M-значных десятичных чисел, то выходным значением функции  могут быть первые M цифр хэш-значения в десятичном представлении.

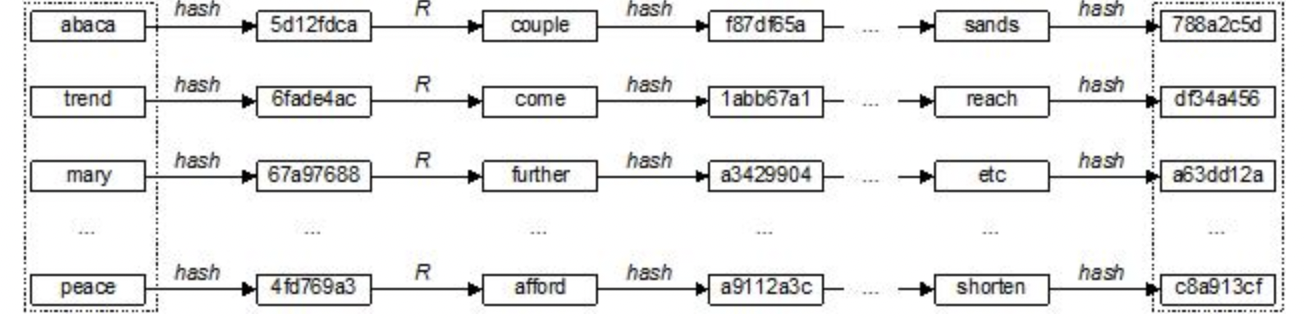
Исходные значения  для каждой цепочки могут выбираться по любому принципу, в том числе, генерироваться случайным образом в пределах покрываемого таблицей множества паролей.  
Атакующий объединяет вычисленные таким образом цепочки в таблицу 

Рис. 2 Цепочка хэш-значений

Принципиальным моментом является то, что для уменьшения требуемой для хранения памяти сохраняется не вся таблица, а лишь первое и последнее значения (т. е.  и ) для каждой строки.

При необходимости найти пароль, соответствующий заданному хэш-значению hx, атакующий вычисляет следующую цепочку:

    (2)

Каждое из получаемых в процессе данных вычислений значений сравнивается с хранящимися в таблице значениями каждой строки. Если на каком-либо этапе обнаружены эквивалентные значения  и , то дальнейшие вычисления цепочки (2) не выполняются, а поиск требуемого пароля выполняется восстановлением данной строки таблицы, т. е. вычислением цепочки (1) до нахождения некоторого значения  равного *x*. При нахождении такого значения искомый пароль определяется как значение  данной цепочки.

Поскольку возможно возникновение коллизий, восстановление строки таблицы может не приводить к нахождению требуемого значения

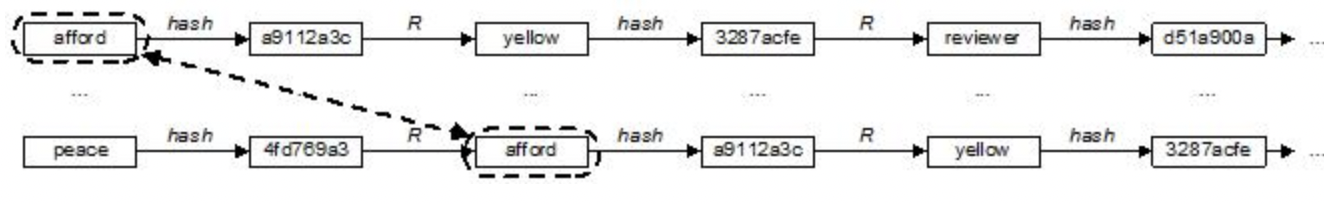


Рис. 3 Коллизия

Данный пример показывает также то, что коллизии уменьшают покрываемое таблицей множество паролей. В этом случае атакующий продолжает вычисления цепочки (2) до нахождения следующего совпадения  и . Если вычисления цепочки (2) завершены, а совпадения не найдены, то атакующим делается вывод о том, что данная таблица не содержит искомый пароль.

Значение N является параметром таблицы цепочек, от которого зависят две важные характеристики таблицы:

* размер памяти для хранения таблицы цепочек обратно пропорционален значению N при эквивалентном покрытии множества паролей (без учета коллизий, вероятность которых нелинейно возрастает с ростом N);
* сложность нахождения требуемого пароля в таблице, которая нелинейно возрастает с ростом N.

#### Радужная таблица (англ. rainbow table).

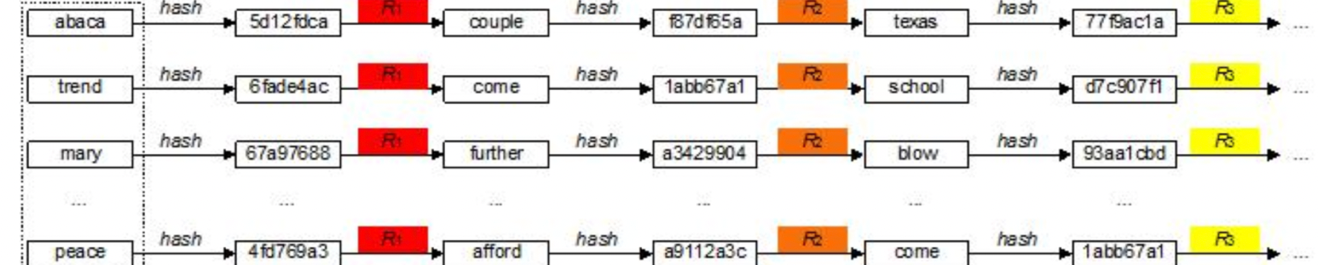
Радужная таблица (rainbow table) – это дальнейшее усиление эффективности цепочек хэш-значений. В радужных таблицах также используется последовательность функций преобразования из хэш-значения в пароль, но каждая функция  используется для преобразования i-го хэш-значения в (i + 1)-й пароль  Таким образом, в таблице последовательно используются функции

Рис. 4 Радужные таблицы

Радужные таблицы обеспечивают более высокую скорость поиска паролей по сравнению с описанными выше вариантами при эквивалентных размерах (а также несколько более высокий процент покрытия множества паролей).  
Две описанные выше проблемы таблиц цепочек хэш-значений решаются радужными таблицами достаточно элегантно:

* «зацикленные» строки невозможны в принципе, поскольку в любой строке используется вся последовательность функций преобразования;
* коллизия приводит к совмещению строк только в случае, если она возникает в одном столбце (вероятность чего в N раз меньше вероятности коллизии во всей таблице); следовательно, значения у таких строк становятся эквивалентными – лишнюю строку можно отбросить и пересчитать с другим начальным значением.

Поиск требуемого пароля в радужной таблице отличается от описанных выше вариантов; он выполняется следующим образом:

1. Сначала ищется совпадение заданного хэш-значения  с каким-либо из значений таблицы.
2. Если совпадение не обнаружено, то предполагается, что должно совпадать с каким-либо из  Следовательно, вычисляется:

и сравнивается со значениями

1. Если совпадение снова не найдено, предположение о местонахождении h*x* в таблице «смещается» еще на один столбец влево, т. е. с h*N* сравнивается уже следующее значение:
2. И так далее, до нахождения соответствия. Если соответствие найдено, то расчет искомого пароля выполняется с начала строки, как во всех описанных выше вариантах. Если же соответствие не найдено до завершения обработки предположения, что  то искомый пароль в данной таблице отсутствует.

Название радужных таблиц произошло от аналогии каждой функции R*i*() с каким-либо цветом, тогда таблица с длинными тонкими разноцветными полосками будет напоминать радугу. Радужные таблицы в настоящее время активно используются для атак на реальные криптосистемы (например, Free Rainbow Tables) [3]

Кроме того, специалистами рассматриваются варианты комбинирования радужных таблиц с таблицами, использующими специфические значения.

Стоит отметить, что все описанные словарные атаки изначально предлагались к использованию для нахождения ключа симметричного шифрования, т. е. в качестве паролей (например, в радужной таблице) фигурируют ключи, а в качестве хэш-значений – результаты зашифрования константного блока открытого текста на конкретном ключе. Следовательно, функция R() в данном случае преобразует блок шифртекста в некий возможный ключ.

### Обход парольной системы в аспекте передачи по сети

В большинстве случаев аутентификация происходит в распределён­ных системах и связана с передачей по сети информации о параметрах учетных записей пользователей. Если передаваемая по сети в процессе аутентификации информация не защищена надлежащим образом, возни­кает угроза ее перехвата злоумышленником и использования для нару­шения защиты парольной системы. Известно, что многие компьютерные системы позволяют переключать сетевой адаптер в режим прослушива­ния адресованного другим получателям сетевого трафика в сети, осно­ванной на широковещательной передаче пакетов данных. В связи с этим атакам на криптографические протоколы обмена информацией в парольных системах стоит уделять особое внимание.

Под атакой̆ на протокол будем понимать попытку проведения анализа сообщений протокола и/или выполнения не предусмотренных протоколом действий̆ с целью нарушения работы протокола и/или получения информации, составляющей секрет его участников (в нашем случае – пароль) [8]

### Основные атаки на криптографические протоколы

Рассмотрим протокол передачи секретного ключа k от A к B без использования какой-либо общей секретной информации, так называемый «бесключевой» протокол А. Шамира, либо — трехпроходный протокол Шамира – Ривеста – Адлемана:

Здесь E — коммутирующее шифрующее преобразование

при всех сообщениях и произвольных ключах k и . Заметим, что в этом протоколе можно использовать не каждое коммутирующее преобразование E. Например, для

протокол оказывается заведомо нестойким:

Поэтому в протоколе Шамира рекомендуется использовать преобразование вида

в котором константа a определяется ключом kA, а p — большое простое число. Укажем некоторые слабости и приведем примеры атак на этот протокол.

#### Атака по имитации: (impersonation attack)

Попытка подменить одного пользователя другим. Нарушитель, выступая от имени одной из сторон и полностью имитируя ее действия, получает в ответ сообщения определенного формата, необходимые для подделки отдельных шагов протокола.

**Слабость 1:** отсутствует аутентификация сторон.

Можно применить атаку путем подмены участника.

Противник C блокирует передачу к B, выступая от имени участника B. В результате C получил от A ключ k для связи с ним от имени B.

#### Атака повторного воспроизведения: (replay attack)

Повторное использование ранее переданного в текущем или предыдущем сеансе сообщения или какой-либо его части в текущем сеансе протокола.

**Слабость 2:** сообщения участников симметричны.

Поэтому протокол не является стойким и к атаке повторением.

Для ее проведения нарушитель C осуществляет доступ к сети от имени участника B, повторяя первое сообщение участника A. В результате ключ k появляется в канале связи в явном виде.

Можно защитить протокол от этой атаки, осуществляя проверку на втором шаге с целью отбраковки повторно переданных сообщений. Но все равно можно осуществить аналогичную атаку путем чередования сообщений двух различных сеансов.

#### Атака: (interleaving attack)

Подмена или другой метод обмана, использующий комбинацию данных из ранее выполненных протоколов, в том числе протоколов, ранее навязанных противником.

Здесь M — произвольное фиктивное сообщение.

#### Атака: («противник в середине»)

Атака, при которой в протоколе идентификации между A и B противник входит в телекоммуникационный канал и становится его частью при реализации протокола между A и B

**Протокол DH:**

Искомый общий ключ теперь вычисляется по формуле

**Слабость:** отсутствие аутентификации сторон.

Противник C:  
— выбирает числа   
— подменяет на соответственно;  
— вычисляет ключи В результате C полностью контролирует обмен сообщениями между A и B, A и B не замечают подмену — они уверены, что связываются непосредственно друг с другом.

## Противодействие обходу парольной защиты.

Зачастую главная проблема сложного пароля – человек. Поскольку требования к паролю все больше ужесточаются, а соответственно растет его длина и сложность, это приводит к трудности или нежеланию простых пользователей запоминать данный пароль. Это приводит к тому, что пользователь либо вовсе отказывается от использования пароля либо записывает его на листике и кладет на видном месте. Потому для создания грамотного и удобного пароля необходимо учитывать 3 момента: длина пароля, сложность пароля и возможность его запоминания.

### Общие рекомендации

Существуют различные рекомендации и требования к созданию стойких паролей. Например, Марк Барнетт в своей книге «Идеальные пароли» предлагает следующие методы повышения стойкости паролей [4].

* 1. Методы увеличения длины пароля: слияние слов, использование скобок, числовые шаблоны, смешные слова, повторение, метод префиксов и суффиксов, метод предложений.
  2. Методы увеличение сложности пароля: применение различных диалектов, разбиение на фрагменты, повторение, замена, бормотания и заикания, опечатки, неграмотность, применение сленга и иностранных языков.
  3. Методы запоминания пароля: рифма, повторение, визуализация, ассоциации, юмор и ирония, преувеличение, преуменьшение, грубость и оскорбления, метод разделения на фрагменты.

Так же следует придерживаться общих рекомендаций по использованию пароля [6]:

* - пароль должен состоять из букв нижнего и верхнего регистров, цифр и знаков;
* - пароль должен быть не менее 9 символов в длину; - пароль к важному объекту должен быть уникален; - нельзя вводить пароль в присутствии несанкционированных лиц, стоящих рядом с вами, которые имеют возможность подсмотреть его;
* - не разглашать пароль и не оставлять информацию, которая может помочь в подборе пароля несанкционированным лицом;
* - при нарушении безопасности пароля должна быть осуществлена его смена, необходимо менять пароль через определенный период времени (в зависимости от его сложности);
* - не рекомендуется при выборе пароля использовать даты рождения, имена, свою фамилию, фамилию матери, номера паспорта, автомобиля и т.д., при смене паро- ля не рекомендуется использовать старые пароли.
* - использовать антивирус и файервол для защиты компьютера от вторжений и вирусных атак, направленных на компрометацию данных;
* - своевременно устанавливать заплатки и патчи для защиты операционных систем и программного обеспечения.

### Методы противодействия подбору пароля при хранении пароля в хэшированном виде

Как один из вариантов защиты от подбора пароля представленным выше требованиям, проанализируем такой способ защиты, как генератор паролей.

#### Генераторы паролей

Рассмотрим самый простой метод генерации. Он заключается в использовании физических устройств, таких как кости, для генерации случайности. Например, возьмем таблицу символов 6 на 6[7]. Заполним ее построчно элементами латинского алфавита и цифрами от 0 до 9.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | a | b | c | d | e | f |
| 2 | g | h | i | j | k | l |
| 3 | m | n | o | p | q | r |
| 4 | s | t | u | v | w | x |
| 5 | y | z | 0 | 1 | 2 | 3 |
| 6 | 4 | 5 | 6 | 7 | 8 | 9 |

Таблица 3. Значения генератора пароля

Первый бросок кости выбирает строку в таблице, а второй - столбец. Так, например, при первом броске имеем значение кости 5, при втором – 4. В итоге получим, согласно таблице, цифру "1". Для генерации символов верхнего / нижнего регистра или некоторых символов можно использовать подбрасывание монеты, заглавный орел, нижний регистр решки. Если цифра была выбрана в бросках костей, бросок монеты heads может выбрать символ над ним на стандартной клавиатуре, например "\*” над "8" вместо "8".

Приведем пример работы данного генератора:

* 1. Поскольку нам нужен достаточно сильный случайный пароль против атаки перебором – подсчитаем информационную энтропию случайного процесса, который его произвел. Воспользуемся формулой , где *N* - количество возможных символов, а *L* - количество символов в пароле. Поскольку пока длину нашего пароля мы не знаем – для начала вычислим энтропию на символ. Наш набор состоит из 72 символов ( латинский алфавит с учетом регистра + 10 цифр с учетом верхнего регистра) Воспользуемся таблицей ниже [1]:

|  |  |  |
| --- | --- | --- |
| Набор символов | Количество символов N | Энтропия на символ H |
| Арабские цифры (0-9) | 10 | 3.32 бит |
| Шестнадцатеричные цифры (0-9, A–F) | 16 | 4.00 бит |
| Латинский алфавиит без регистра (a–z или A–Z) | 26 | 4.70 бит |
| Без учета регистра буквенно–цифровой (a–z или A-Z, 0-9) | 36 | 5.17 бит |
| Латинский алфавит с учетом регистра (a–z, A–Z) | 52 | 5.70 бит |
| Буквенно–цифровой (a–z, A-Z, 0-9) | 62 | 5.95 бит |
| Все Символы для печати ASCII | 94 | 6.55 бит |
| Список слов Diceware | 7776 | 12.9 бит |

Таблица 4. Энтропия на символ

Как мы видим, наше значение будет лежать между 5.95 и 6.55, а именно – 6.17

* 1. Далее, нам необходимо получить длину пароля. Для этого обратимся к следующей таблице

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| H | Арабские цифры | Шестнадцатеричный | Латинский алфавит без регистра | Буквенно-цифровой без учета регистра | Латинский алфавит с учетом регистра | Буквенно-цифровой с учетом регистра | Символы для печати ASCII | расширенные символы ASCII | Diceware |
| 32 бита | 10 | 8 | 7 | 7 | 6 | 6 | 5 | 5 | 3 |
| 40 бит | 13 | 10 | 9 | 8 | 8 | 7 | 7 | 6 | 4 |
| 64 бита | 20 | 16 | 14 | 13 | 12 | 11 | 10 | 9 | 5 |
| 80 бит | 25 | 20 | 18 | 16 | 15 | 14 | 13 | 11 | 7 |
| 96 бит | 29 | 24 | 21 | 19 | 17 | 17 | 15 | 13 | 8 |
| 128 бит | 39 | 32 | 28 | 25 | 23 | 22 | 20 | 17 | 10 |
| 160 бит | 49 | 40 | 35 | 31 | 29 | 27 | 25 | 21 | 13 |
| 192 бита | 58 | 48 | 41 | 38 | 34 | 33 | 30 | 25 | 15 |
| 224 бита | 68 | 56 | 48 | 44 | 40 | 38 | 35 | 29 | 18 |
| 256 бит | 78 | 64 | 55 | 50 | 45 | 43 | 40 | 33 | 20 |
| 384 бита | 116 | 96 | 82 | 75 | 68 | 65 | 59 | 50 | 30 |
| 512 бит | 155 | 128 | 109 | 100 | 90 | 86 | 79 | 66 | 40 |
| 1024 бит | 309 | 256 | 218 | 199 | 180 | 172 | 157 | 132 | 80 |

Таблица 4. Длина пароля в зависимости от энтропии и алфавита

Отсюда видно, что для более-менее стойкого пароля нам нужно сгенерировать пароль длиной 16 символов, тогда итоговая энтропия составит примерно 98 бит

* 1. Генерируем пароль 16-тью роллами двух костей и 16-тью бросками монеты:
     + 1, 3, Решка
     + 5, 6, Орел
     + 5, 1, Орел
     + 6, 5, Решка
     + 2, 4, Орел
     + 5, 2, Решка
     + 1, 6, Решка
     + 2, 2, Орел
     + 2, 6, Решка
     + 2, 1, Орел
     + 3, 2, Решка
     + 6, 4, Решка
     + 6, 2, Орел
     + 5, 5, Решка
     + 2, 5, Орел
     + 4, 3, Орел
  2. В результате получим комбинацию– c#Y8JzfHlGn7%qKU

Итоговый пароль достаточно стоек, так как тяжело поддается перебору (см. Таблицу 1) и не состоит из легко угадываемых по словарям частей.

#### Cоль

Рассмотрим еще один метод защиты от взлома касаемо хранения пароля в базе данных. Он применим к атаке с помощью радужных таблиц и заключается в использовании необратимых хеш-функций, которые включают *salt* («соль», «модификатор»). Существует множество возможных схем смешения затравки и пароля. Например, рассмотрим следующую функцию для создания хеша от пароля [9]:

Для восстановления такого пароля взломщику необходимы таблицы для всех возможных значений соли. При использовании такой схемы соль должна быть достаточно длинной (6‒8 символов), иначе злоумышленник может вычислить таблицы для каждого значения соли, случайной и различной для каждого пароля. Таким образом два одинаковых пароля будут иметь разные значения хешей, если только не будет использоваться одинаковая соль. По сути, соль увеличивает длину и, возможно, сложность пароля. Если таблица рассчитана на некоторую длину или на некоторый ограниченный набор символов, то соль может предотвратить восстановление пароля. Например, в старых Unix-паролях использовалась соль, размер которой составлял всего лишь 12 бит. Для взлома таких паролей злоумышленнику нужно было посчитать всего 4096 таблиц, которые можно свободно хранить на терабайтных жестких дисках. Поэтому в современных приложениях стараются использовать более длинную соль. Например, в алгоритме хеширования bcrypt используется соль размером 128 бит. Подобная длина соли делает предварительные вычисления просто бессмысленными. Другим возможным способом борьбы против атак, использующих предварительные вычисления, является растяжение ключа(англ. *keystretching*). Например:

ключ = хеш(пароль + соль)

for 1 to 65536 do

ключ = хеш(ключ + пароль + соль)

Этот способ снижает эффективность применения предварительных вычислений, так как использование промежуточных значений увеличивает время, которое необходимо для вычисления одного пароля, и тем самым уменьшает количество вычислений, которые злоумышленник может произвести в установленные временные рамки. Данный метод применяется в следующих алгоритмах хеширования: MD5, в котором используется 1000 повторений, и bcrypt. Альтернативным вариантом является использование усиления ключа (англ. *keystrengthening*), который часто принимают за растяжение ключа. Применяя данный метод, мы увеличиваем размер ключа за счёт добавки случайной соли, которая затем спокойно удаляется, в отличие от растяжения ключа, когда соль сохраняется и используется в следующих итерациях. Также рассмотрим LM-хеш — это старый алгоритм хеширования, который используется в Microsoft. Он чрезвычайно уязвим, так пароль, состоящий больше, чем из 7 символов, но меньше, чем из 15 символов, разбивается на две части, которые хешируются независимо друг от друга. Чтобы избежать создания LM-хеша, необходимо использовать пароли из 15 символов и более.

### Методы противодействия перехвату пароля при передаче по сети

#### Общие рекомендации

Для защиты от атак на криптографические протоколы следует придерживаться следующих общих правил:

1. Сохранение в тайне от противника информации, определяющей алгоритм идентификации;
2. Использование различных форматов сообщений, передаваемых на разных шагах протокола;
3. Вставка в сообщения специальных идентификационных меток, номеров сообщений, случайных чисел, возрастающей последовательности чисел и тд.
4. Привязка ключей не к одной, а к обеим взаимодействующим сторонам путем передачи обоих идентификаторов в зашифрованном виде (при наличии доверенной стороны)
5. Техника типа «запрос — ответ»
6. Обеспечении целостности сеансов протоколов и отдельных сообщений.
7. Использование протоколов с нулевым разглашением.

#### Метод защиты от «Противник посередине»

Отдельно разберём интересный подход к защите протокола DH от атаки «противник в середине [8].

Предположим, что участники A и B имеют секретные ключи a, соответственно и публикуют свои открытые ключи и Для выработки общего секретного ключа k они должны сгенерировать случайные числа , , соответственно, а затем обменяться следующими сообщениями:

Теперь участники A и B вычисляют общий ключ по формулам

,

Любая подмена сообщений приводит к тому, что все стороны получают различные значения ключа, в результате чего становится невозможным чтение всей передаваемой информации. Тем самым свойство аутентификации ключа протокола при атаке «противник в середине» не нарушено. Вместе с тем этот протокол не обеспечивает аутентификации сторон и подтверждения правильности получения ключа.

Покажем, что даже после усиления этого протокола путем добавления к пересылаемым сообщениям сертификатов он остается уязвимым к подмене содержания передаваемых сообщений, а именно, этот протокол не обеспечивает аутентификации сторон.

Атака на протокол MTI/A0, в которой нарушитель C подменяет сертификат участника A на свой собственный. Пусть участник A получил сертификат

Нарушитель C регистрирует для себя сертификат

где e — произвольное число, предназначенное для того, чтобы скрыть связь с ключом участника A. Далее он использует следующий протокол:

В результате участник B будет уверен, что полученный в результате выполнения этого протокола ключ

будет общим ключом с участником C, а не с участником A, который ничего об этом не подозревает. При этом участник A вычислит тот же ключ k. Поэтому вся информация, полученная от участника A и зашифрованная на этом ключе, будет восприниматься участником B как полученная от участника C, а не от A.

Таким образом, данный протокол хотя и обеспечивает аутентификацию ключа для участника A, в то же время не обеспечивает аутентификации сторон, а также аутентификации ключа для участника B.

## Заключение

Современные технологии и вычислительная мощность дают злоумышленнику большие возможности по преодолению парольной защиты операционных систем. В данной работе был проведен анализ существующих механизмов взлома паролей с методами защиты от них. К сожалению, разработанные специалистами по информационной безопасности требования к парольной защиты не всегда выполняются простыми пользователями ввиду сложности их использования. Впрочем, если пользователи не заботятся о защите своих аккаунтов и не соблюдают даже минимальные меры по обеспечению безопасности, все усилия изнутри могут оказаться напрасны. Сложность криптографических алгоритмов не играет абсолютно никакой роли, если злоумышленник может легко получить ключ.). Однако разобранный в работе простейший генератор криптографически стойких паролей показывает, что для надежной защиты своих данных порой даже не требуется сложных вычислений.

## Список литературы

1. Пароль по умолчанию | Справится даже начинающий хакер [Электронный ресурс]: <https://telegra.ph/Poiskovye-sistemy-dlya-interneta-veshchej-11-01>
2. Словарные атаки на хэш-функции, С. Панасенко, 2009 [Электронный ресурс]: <http://www.panasenko.ru/Articles/168/168.html>
3. Построение систем защиты от угрозы нарушения конфиденциальности информации. Системы разграничения доступа. Криптографические методы защиты [Электронный ресурс]:<https://learn.urfu.ru/resource/index/data/resource_id/40978/revision_id/0>
4. Free Rainbow Tables [Электронный ресурс]: http://www.freerainbowtables.com
5. Mark Burnett, Perfect passwords / M.Burnett. - Canada: Syngress Publishing, Inc., 2006. - 177 с.
6. Анализ устойчивости к взлому современных механизмов парольной защиты ОС, А.В. Снегуров В.Х. Чакрян – 28 с.
7. Random password generation [Электронный ресурс]: <https://wiki2.org/en/Random_password_generator>
8. Криптографические протоколы: основные свойства и уязвимости, А. В. Черемушкин, Ноябрь 2009 – ст. 120
9. Защита от радужных таблиц [Электронный ресурс]: <https://studopedia.ru/19_382934_zashchita-ot-raduzhnih-tablits.html>?