**Kripptx. Общие положения.**

**Kripptx** – это алгоритм для шифрованного обмена данными между двумя пирами сети. Для работы Kripptx необходимо два подключенных друг к другу любым транспортом (с гарантированной доставкой) пира. После подключения оба пира становятся алгоритмически равнозначными. Способ установки шифрованного канала связи является вероятностным, а, следовательно, невозможно на 100% доказать:

* истинность факта установки соединения;
* истинность факта передачи данных;
* истинность факта расшифровывания сообщения.

**Kripptx. Шифрование.**

В основе алгоритма шифрования лежит собственное изобретение – **динамически обновляемый криптопул** (**ДОКП**). **ДОКП** – это массив из 256 байт (нумеруемых от 0 до 255 включительно), каждый из которых может принимать значение от 0 до 255 включительно, причем каждое такое значение встречается в массиве только один раз. ДОКП создается во время инициализации шифрованного соединения программ и совпадает на обеих сторонах на протяжении всего времени работы соединения, синхронно модифицируясь.

**ДОКП. Инициализация.**

При инициализации шифрованного соединения создание ДОКП происходит следующим образом. Вначале ДОКП очищается, то есть делаются свободными все 256 ячеек. Суть составления ДОКП заключается в том, чтобы все 256 ячеек, от 0 до 255 включительно, были заполнены различными значениями, от 0 до 255, чтобы каждое значение присутствовало в ДОКП один и только один раз. Значит необходимо произвести 256 записей в ДОКП различных значений, то есть 256 раз выбрать *куда* записать и *что* записать. Говоря то же самое другими словами, требуется 256 раундов решить, какое *значение* по какому *указателю* на ячейку записать в ДОКП. Суть разделения секрета заключается в том, что процесс составления ДОКП является для клиента и сервера коллективным, а по открытому каналу связи передаются недостаточные данные для восстановления ДОКП злоумышленником.

Итак, в начале каждого раунда Алиса и Боб (клиент и сервер, или два экземпляра приложения)случайным образом решают каждый сам для себя, вне зависимости друг от друга, что они будут генерировать в текущем раунде – указатель или значение. Среди Алисы и Боба есть 3 варианта: указатель и указатель, указатель и значение, значение и значение.

Если Алиса генерирует указатель, а Боб – значение (заранее это проверить невозможно), то в дальнейшем происходит следующее: Алиса и Боб обмениваются по открытому каналу связи сгенерированными величинами, и, так как Алиса знает для себя, что сгенерировала указатель, то она ждет от Боба значения, а Боб, зная для себя, что он сгенерировал значение, ждет от Алисы указатель. Например, Алиса сгенерировала указатель 11, а Боб – значение 73. Значит, Алиса ждет от Боба значение, которое нужно поместить по указателю 11 в ДОКП, а Боб ждет указатель на ДОКП, куда нужно записать значение 73. По открытому каналу связи транслируются две величины – 11 и 73.При этом злоумышленник не знает, какая из этих величин указатель, а какая – значение, однако Алиса и Боб знают, что у кого, и, таким образом, в конце текущего раунда и у Алисы и у Боба в ДОКП на месте 11 будет значение 73. Злоумышленнику же, если он хочет моделировать ДОКП, придется только в первом раунде смоделировать уже две ситуации: когда 73-значение а 11-указатель, и когда 73-указатель а 11-значение.Так, 255 раз, раунд за раундом, передаются и заполняются все значения всего содержимого ДОКП.

Но что если Боб и Алиса оба сгенерировали указатели? Или значения? В таком случае в этом и/или последующим раунде(ах) неизбежны коллизии (а именно – попытка записи в уже заполненный байт ДОКП), при обнаружении которых партнеру подается команда сброса, ДОКП вновь очищается и начинает создаваться заново с первого раунда. Так происходит до тех пор, пока ДОКП не будет полностью заполнен, а, значит, создан.

Отметим, что указатель и значение выбираются сторонами при генерации из числа еще не использованных. Например, если до сих пор при составлении ДОКП коллизий не обнаруживалось, то, если Алиса уже генерировала указатели 15, 77 и 210, то вновь она их сгенерировать уже не имеет права, равно как и Боб, имеющий точную копию ДОКП, так как значения по этим указателям уже заполнены. То же касается значений, используемых единожды.

Так как невозможно заранее договориться о том, кто в какой раунд придумывает значение, а кто — указатель, вероятность составления правильного полного ДОКП Алисой и Бобом равна 1/2256. То есть вероятность составления ДОКП стремится к единице при числе попыток, стремящемся к 2256 = 11579208923731619542357098500868790785326998466564056403945758400 7913129639936.

Несмотря на фундаментально беспрецедентную защищенность алгоритма, из-за постоянных коллизий и сбросов время составления чистого ДОКП, таким образом, может достигать астрономических величин, поэтому для ускорения строительства ДОКП вводится алгоритмически неопределенная согласованность в действиях сторон. Алгоритмически неопределенной она должна быть для того, чтобы злоумышленник, прослушивающий канал, не смог алгоритмически реконструировать создание ДОКП и, тем самым, получить доступ к обмену данными. Согласованность вводится вероятностная.

**ДОКП. Инициализация. Вероятностный алгоритм. Высокая защищенность.**

В качестве вероятностного алгоритма согласований действия Алисы и Боба при составлении ДОКП используется следующий.

Для каждого из персонажей пусть *p* – вероятность генерации им в текущем раунде указателя, а *q–* вероятность генерации значения. В момент перед генерацией ДКОП величины устанавливаются в *p*= 0.5, *q* = 0.5. В любой момент времени у каждого персонажа *p* + *q* = 1. В конце каждого раунда величины *p* и *q* будут меняться. Если персонаж фактически сгенерировал в текущем раунде указатель, и запись значений прошла без коллизий, то в конце раунда его *p* модифицируется следующим образом: новое , где *V* – величина значения, использованного в текущем раунде, а новое *q* = 1 – *p*. Таким образом, вероятность генерации указателя в следующем раунде упадет либо в 2, либо в 3 раза. Если же персонаж фактически сгенерировал в текущем раунде значение, то в конце раунда его , а *p* = 1 – *q*. Таким образом, вероятность генерации значения в следующем раунде упадет либо в 2, либо в 3 раза. Например, если в первом раунде, когда *p*= 0.5, *q* = 0.5, Алиса сгенерировала значение, и это значение *V* = 173, то хотелось бы (с точки зрения вероятностного распределения ролей), чтобы в следующем раунде она сгенерировала указатель, посему в конце раунда её , а *p* = 0.8333. Это вероятностно обуславливает равномерность генераций указателей и значений Алисой и Бобом в разных раундах, однако не гарантирует отсутствие раундов, в которых Алиса и Боб, несмотря на соответствующие*p* и *q*, смогут запросто сгенерировать одинаковые или даже сути.

**Kripptx. Обмен сообщениями.**

При отправке сообщения оно шифруется на отправляющей стороне с использованием существующего ДОКП по специальному алгоритму **sE\***, перемещается по (открытому) каналу связи в зашифрованном виде другой стороне, где дешифруется с использованием точно такого же ДОКП по алгоритму **sD\***, после чего ДОКП синхронно изменяется на обеих сторонах согласно алгоритму **sS\***.

**Обмен сообщениями. Алгоритм sE\*.**

Необходимо закодировать сообщение. Байты . Алгоритм кодирует сообщение с помощью существующего криптопула. Вначале заметим, что ДОКП являет собой биективную подстановку , где - использованные один и только один раз все значения от 0 до 255 включительно, а, значит, обратимую операцию. Кодирование состоит из *N* одинаковых раундов (*N* выбирается пользователями в настройках программы, чем оно больше, тем лучше). Каждый раунд состоит из 4 операций. На вход первому раунду подается исходное сообщение . Раунд модифицирует сообщение с помощью 4 операций, в результате получая . На вход второму раунду подается ’, и т.д. В результате получаем модифицированную строку.

**Обмен сообщениями. Алгоритм sE\*. Четыре операции раунда.**

Первая обратимая операция – битовое инвертирование:

Вторая обратимая операция – сумма над целым положительным циклическим полем :

К текущему элементу прибавляем значение ДОКП, расположенное по указателю . Очевидно, операция обратима. Так как длина сообщения может быть *L*> 255, необходимо обезопасить себя от обращения к элементу ДОКП с индексом, большим, чем 255. Посему в качестве выбора элемента подстановки используем не , а .

Третья обратимая операция – табличная замена согласно подстановке:

Четвертая обратимая операция – кольцевой сдвиг вправо всех байт сообщения, иначе говоря, сдвигается на *m* шагов вправо и превращается в

, где *m* определяется как . Очевидно, что операция сдвига обратима.

**Обмен сообщениями. Алгоритм sE\*. Пример над полем** .

Пусть, например, ДОКП = , *S* = 01248917653. Пусть также *N* = 1. После проведения первой операции (инвертирование, что в данном случае является операцией поиска дополнения до 10-ти)

*S*’ = 98751082346

Вторая операция подразумевает:

Третья операция - табличная замена:

Для проведения четвертой операции подсчитаем

, тогда необходимо циклически сдвинуть вправо на 4 элемента:

Таким образом, за первый раунд (и единственный, так как *N* = 1), с помощью сообщение *S* = 01248917653 преобразовалось в .

**Обмен сообщениями. Алгоритм sD\*. Пример над полем** .

Декодируем полученное в прошлый раз сообщение *S* = 77587575133. Должно получиться исходное 01248917653. По-прежнему, *N* = 1.

Применим обратное к четвёртой операции преобразование обратного сдвига, снова подсчитав. Значит, перед четвёртым преобразованием *S* было равно 75751337758.

Применим обратное к третьей операции преобразование нахождения прообраза по биективной подстановке:

* первому элементу «7» образа соответствует элемент «5» прообраза
* второму элементу «5» образа соответствует элемент «9» прообраза
* и так далее
* последнему элементу «8» образа соответствует элемент «2» прообраза

Таким образом, перед третьим преобразованием *S* было равно 59591775592.

Применим обратное ко второй операции преобразование поэлементного вычитания над полем :

* первый элемент исходной последовательности есть корень уравнения (x + 6) *mod* 10 = 5, то есть (5 – 6 + 10) *mod* 10 = 9
* второй элемент исходной последовательности есть корень уравнения (x + 1) *mod* 10 = 9, то есть (9 – 1 + 10) *mod* 10 = 8
* и так далее
* последний элемент исходной последовательности есть корень уравнения (x + 6) *mod* 10 = 2, то есть (2 – 6 + 10) *mod* 10 = 6

Таким образом, перед вторым преобразованием *S* было равно 98751082346.

Наконец, применим обратное к первой операции преобразование поиска дополнения до 10-ти: исходное сообщение *S* было равно 01248917653.

**Обмен сообщениями. Алгоритм sS\*.**

Алгоритм sS\* подразумевает изменение ДОКП согласно полученному/отправленному сообщению. При этом sS\* должен изменять ДОКП очень сильно, вне зависимости от того, было ли *S* длинным или коротким, порой, состоящим из одного байта. Для увеличения силы изменения необходимо производить замыкание ДОКП на себя при использовании параметров сообщения *S*. Также необходимо помнить о том, чтобы модифицированный ДОКП остался биективной подстановкой. Это можно достичь параметрическим сдвигом и подстановкой элементов .

|  |  |
| --- | --- |
| Цель алгоритмов sE\* и sS\* – сделать так, чтобы сообщение , являющееся продуктом применения закрытого от злоумышленника ключа ДОКП, само стало закрытым ключом для последующей модификации ДОКП. То есть сообщения – это закрытые ключи изменений ДОКП, а ДОКП – это закрытые ключи шифрации сообщений. Так как такая система криптографически замкнута, вторгнуться в неё, не зная ДОКП, невозможно (даже при сборе прослушивающей канал стороной большого объема статистической информации). | **\*** |

Итак, в качестве изменений элементов будем использовать подстановки и сдвиги.

Определим операцию как смену местами элементов и , . Ясно, что . Так же ясно, что , .

Определим далее как циклический сдвиг ДОКП вправо на байт.

Наконец, определим операцию как применение подстановки самой к себе.

Определим раунд как последовательное применение операций , и к , , .

Тогда сутью алгоритма sS\* будет являться следующие действия:

*for i from 0 to 64 (inc)*

*{*

*for j from i+1 to 128 (inc)*

*{*

*}*

*}*

**ДОКП. Взлом перебором.**

Общее число возможных криптопулов есть суть значения числа перестановок из 256 элементов, то есть 256*!* возможных вариантов (8,57 х 10506 или число с 507 знаками). При попытке взлома канала связи методом подбора ДОКП перебором с перебором одного варианта в секунду понадобится 2,72 х 10499 лет, что в 1,98 х 10489 раз больше установленного времени существования Вселенной. Алгоритмический взлом ДОКП невозможен ввиду использования вероятностных переменных. Однако следует заметить, что подобрав ДОКП и зная алгоритмы sE\* и sD\* можно прочитать текущее сообщение, а, зная алгоритм sS\*, можно рассчитать последующий ДОКП и сообщения.

**ДОКП. Взлом прослушиванием канала.**

При попытке взлома ДОКП подслушиванием канала обмена значениями и указателями шансы злоумышленника обречены – в каждом раунде вероятность правильно угадать, где значение, а где указатель, равна ½ или 50%. Учитывая, что раундов создания ДОКП255, вероятность правильного определения ДОКП путем прослушивания канала составляет 1/2255.