ЛЕКЦИЯ 9

ПРОТОКОЛЫ АУТЕНТИФИКАЦИИ И РАСПРЕДЕЛЕНИЯ КЛЮЧЕЙ

На прошлой лекции рассматривались схемы распределения ключей. Были выделены две возможных структуры, которые используются в мире: инфраструктура, основанная на **корневых доверенных центрах**, и **сетевая инфраструктура**, которая используется в программах типа PGP. В данной лекции будет рассмотрено построение сетей доверия и распространение ключей в маленьких организациях (до 10 или 100 тысяч человек). Обычно такие задачи возникают у IT-директоров, начальников разработки компании или на посту произвольного ответственного лица, в задачу которого входит построение надёжной информационной системы.

Задачи будут рассматриваться в рамках классической криптографии. Самая актуальная проблема — это построение секретного (защищённого) канала между собеседниками (в **секретном канале** злоумышленники не могут подслушать или изменить данные).

Защищённым каналом можно назвать любой канал, при котором собеседники общаются с использованием шифрования (шифруя все данные одним и тем же ключом). Только они могут пересылать эти данные по каналу и считывать, потому что только им известен секретный ключ. Собеседникам необходимо выработать общий ключ. Предположим, что у собеседников есть ключи друг друга (рассматривается ситуация, когда собеседники используют не один ключ). Они могут использовать любой из ключей, либо сгенерировать третий ключ, основываясь на своих закрытых ключах. Им нужно понять, какой ключ использовать. Это нетрудно, пока получателей всего двое. Проблема возникает, если получателей становится больше.

Если количество абонентов порядка ста, то количество парных ключей будет равно:

$$\frac{100 \cdot (100 - 1)}{2} \approx 5000.$$

Для ста абонентов нужно пять тысяч ключей. Для десяти тысяч абонентов нужно $\approx 5 \cdot 10^6$ ключей. То есть количество ключей в системе намного превышает количе-

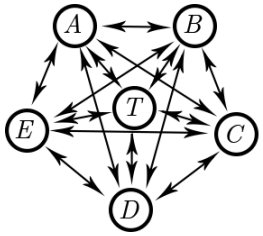

Рис. 9.1

ство абонентов. Причём в надёжной защищённой системе нужно регулярно обновлять ключи. Даже если заниматься обновлением ключей раз в 3 года (что достаточно редко), то будет затрачиваться слишком много усилий: каждый ключ необходимо надёжно сгенерировать, отправить получателям, убедиться, что они их получили, не будучи перехваченными.

Нужно организовать такую систему, чтобы в большом количестве ключей не было необходимости. Для этого можно использовать так называемый **доверенный центр** Трент. Именно в этом доверенном центре хранятся все ключи всех абонентов. Этот доверенный центр управляется самой организацией или главным системным администратором. У него содержится информация, каким ключом нужно шифровать данные, чтобы передать их собеседникам. У каждого из абонентов тоже есть соответствующий парный ключ. То есть, если в системе n абонентов, у Трента есть n ключей, отвечающих за связи с этими абонентами, причём эти ключи взаимные. Ключ конкретного собеседника одновременно известен и Тренту, и самому собеседнику. Когда они будут обмениваться данными, Трент будет шифровать данные этим ключом, и собеседник тоже будет шифровать данные этим же ключом, общим ключом его и Трента.

1. Протоколы выработки общих сеансовых ключей

В рассматриваемых протоколах будет использоваться промежуточный центр. В этих протоколах предполагается, что доверенный центр известен заранее, и все пары ключей между абонентами и доверенным центром тоже известны заранее, но нет никакой связи между абонентом и вторым абонентом. Максимум, что они знают — это имена друг друга.

1.1. «Лягушка с широко открытой глоткой» (Wide-Mouth Frog)

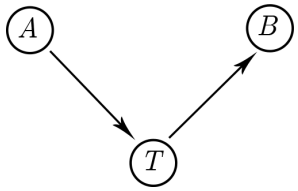

Рис. 9.2

Первый собеседник отправляет сообщение Тренту — промежуточному центру. В сообщение входит его идентификатор (обозначим его буквой A) и зашифрованная часть.

В зашифрованную часть входит метка времени, идентификатор второго собеседника и ключ — случайное число, которое первый собеседник предполагает использовать в качестве сеансового ключа связи.

После отправки сообщения Тренту тот передаёт второму собеседнику зашифрованное сообщение, в которое входит метка времени, идентификатор первого собеседника и ключ.

$$1 \to \{A, E_A(T_A, B, K)\} \to Trent,$$

$$Trent \to \{E_B(T_T, A, K)\} \to 2.$$

В первом случае сообщение зашифровано совместным ключом первого собеседника и Трента, а во втором — совместным ключом второго собеседника и Трента.

В классической криптографии нет необходимости отдельно подписывать данные: сам факт того, что кто-то смог зашифровать сообщение совместным ключом первого собеседника и Трента, означает, что либо это тот собеседник, либо Трент, потому что больше никто этот ключ не знает. Никакие дополнительные подтверждения по электронной подписи, удостоверению корректности сообщения в данном случае не нужны.

Трент, когда получает данное сообщение, знает, что оно пришло от конкретного абонента, и расшифровывает сообщение соответствующим ключом. Трент посылает сообщение второму абоненту, и тот знает, что ему такие сообщения могут прийти только от Трента. Он расшифровывает их своим ключом и находит идентификатор первого собеседника и ключ, после чего приходит к выводу, что с ним хочет связаться конкретный абонент, и надо использовать соответствующий ключ для сеанса связи. На этом сеанс связи между абонентами можно считать установившимся.

Примечательно, что заранее они друг с другом не связывались. У них не было никакой общей информации кроме того факта, что они оба соединены с доверенным центром.

У Трента содержится вся информация, которую пересылали друг другу абоненты, все их ключи. Он может себя выдать за любого абонента. С другой стороны, никто не может подменить Трента просто так. Если злоумышленник проникнет в систему и попытается выдать себя за Трента, ему начнут посылать сообщения, часть которых зашифрована ключом, который установил администратор, а у злоумышленника этого ключа нет, и он не сможет расшифровать сообщение. С одной стороны, это уязвимое место — недостаток, но с другой стороны, это единственное уязвимое место, которое достаточно просто защитить.

Обычно Трент-сервер достаточно мощный, чтобы обработать любой поток сообщений, и это предотвращает **DDOS-атаку** на него. Но возможна также следующая ситуация: если Трент является публичным сервисом, то абоненты могут регистрироваться на этом сервисе без контроля администраторов. В какой-то момент тысячи абонентов посылают Тренту запрос на общение с каким-то конкретным пользователем. И промежуточный центр шлёт пользователю большой объём сообщений от всех возможных получателей. Это есть **DOS-атака** на пользователя, но не DDOS (не distributed). Это атака с отражением: поток данных отражается от Трента и направляется к пользователю.

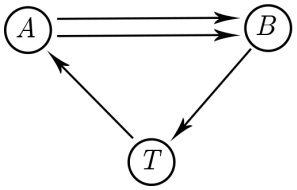

Рис. 9.3

1.2. Yahalom

В этом протоколе, как изображено на диаграмме, сначала собеседник 1 посылает сообщение собеседнику 2, затем 2 посылает сообщение Тренту, Трент — 1, 1 — 2.

$$1 \to \{A, R_{\text{A}}\} \to 2;$$

$$2 \to \{B, E_{\text{B}}(A, R_{\text{A}}, R_{\text{B}})\} \to Trent;$$

$$Trent \to \{E_{\text{A}}(B, K, R_{\text{A}}, R_{\text{B}})\} \to 1;$$

$$Trent \to \{E_{\text{B}}(A, K)\} \to 1;$$

$$1 \to \{E_{\text{B}}(A, K)\} \to 2;$$

$$1 \to \{E_{\text{K}}(R_{\text{B}})\} \to 2.$$

1 посылает сообщение 2, включая некое случайное число. Затем второй посылает Тренту сообщение, уведомляя того о том, что со вторым абонентом хочет связаться некто, представившийся первый абонентом. После этого Трент связывается с первым собеседником и отсылает ему ключ. При этом это шифруется ключом первого абонента. Причём Трент посылает два сообщения: одно для абонента 1, а также вторую часть, которая уйдёт второму (эта часть зашифрована ключом 2, и первый абонент с этой частью ничего сделать не может). После этого первый отсылает оставшуюся часть второму (внутри этой части оказывается идентификатор и ключ). После этого первый собеседник дополнительно шифрует ключом случайное число второго, тем самым подтверждая, что знает этот ключ.

С точки зрения абонента 2, происходит следующее: с ним сначала связывается первый, после чего он отправил запрос Тренту, вложив туда случайное число. После этого второму абоненту ещё раз пишет первый, пересылает от Трента новый сессионный ключ и присылает доказательство, что он этот ключ знает, зашифровав случайное число второго.

Такая сложная схема направлена сразу на устранение нескольких недостатков: например, если бы не было случайных чисел, то была бы возможной **атака повтором** (злоумышленник может запомнить все пересылаемые данные, потом выбрать данные, которые шифровались этим сеансовым ключом, и вскрыть ключ).

Сессионные ключи считаются ненадёжными: сессионные ключи максимум через год будут вскрыты. В связи с этим недавно возникла проблема: криптографы и системные администраторы обнаружили, что такие организации как АНБ, ФБР и ФСБ могут записывать весь трафик в надежде на то, что когда-то в будущем они смогут его расшифровать. То есть, шифруя трафик, надо всегда понимать, что даже если его сейчас нельзя вскрыть, когда-то в будущем это вполне будет возможно.

Секретный ключ K генерируется случайным образом. Отличие от предыдущего протокола состоит в том, что здесь сессионный ключ для связи двух абонентов генерируется на стороне доверенного центра. В сервере доверенного центра может быть специальная плата, которая генерирует истинно случайные числа. В предыдущем протоколе, где это число генерирует один из собеседников, такая степень надежности не достигается: у абонента может быть плохое программное и аппаратное обеспечение, и он может быть неспособн сгенерировать по-настоящему случайное число. Именно это и есть одно из преимуществ подобного протокола: случайное число K генерируется именно на доверенном сервере, где оно может быть сделано надёжно.

Даже с использованием случайных чисел (R_A, R_B), в этом протоколе есть серьёзный недостаток. Предположим, что злоумышленник взломал ключ K. Тогда он может выдать себя за одного из собеседников: он может отослать второму абоненту числа A и R_A, причём R_A может быть совсем другим. Собеседник злоумышленника посылает сообщение Тренту, но в этот момент, пока до Трента оно не дошло, злоумышленник можно послать повторно старый ключ, зашифрованный совместным ключом Трента и его собеседника. Дальше злоумышленник подтверждает, что он этот ключ знает. Тогда собеседник 2 общается со злоумышленником, будучи уверенным, что это собеседник 1.

Но недостатком этого протокола является очень сложная структура связи. Здесь нельзя обойтись без онлайн общения: этот протокол нельзя разорвать, разнести во времени. Также лучше заранее сгенерировать набор ключей для общения с абонентом.

1.3. Протокол Нидхема – Шрёдера

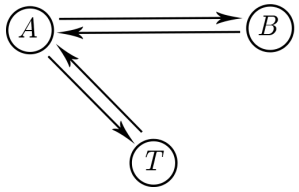

Рис. 9.4

Первый посылает сообщение Тренту, уведомляя доверенный центр о намерении начать общение со вторым абонентом. Трент в ответ посылает ему большой набор данных, зашифрованный ключом первого абонента и Трента. После чего первый начинает общаться со вторым и доказывает, что у него есть сессионный ключ, полученный от Трента.

Примечательно, что общение первого абонента и Трента можно отделить от общения первого и второго абонентов. То есть чисто теоретически, первый собеседник может заранее заготовить несколько ключей для общения на будущее и использовать их.

$$1 \to \{A, B, R_\text{A}\} \to Trent;$$

$$Trent \to \{E_\text{A}(R_\text{A}, B, K, E_\text{B}(K, A))\} \to 1;$$

$$1 \to \{E_\text{B}(K, A)\} \to 2;$$

$$2 \to \{E_\text{K}(R_\text{B})\} \to 1;$$

$$1 \to \{E_{\text{K}}(R_{\text{B}} - 1)\} \to 2.$$

Сначала вся информация шифруется ключом первого абонента. К Тренту попадает случайное число, полученное от первого абонента, идентификатор, сессионный ключ и набор данных, которые будут переданы второму собеседнику. Дальше первый абонент получает набор данных, с которым он ничего не может сделать. Второй абонент посылает первому запрос на ключ, который первый собеседник должен был получить от доверенного центра, после чего первый присылает второму абоненту доказательство того, что он действительно знает ключ.

В этом протоколе содержится недостаток: злоумыленник может через некоторое время взломать ключ K, после чего он может послать данные второму собеседнику вместо первого (ключ, внутри идентификатор первого абонента). Второй абонент пошлет злоумыленнику запрос на ключ, и злоумыленник подтвердит, что он знает ключ (который он взломал). Второй собеселник не может защититься от подобной схемы.

Также есть еще один мелкий недостаток: на четвертом шаге сообщение шифруется ключом второго собеседника, а потом совместным ключом второго абонента и Трента и совместным ключом первого абонента и Трента. Но шифровать это бессмысленно, потому что на следующем шаге эта зашифрованная часть передаётся в открытом виде. То есть эти два числа можно вынести из функции шифрования.

В качестве достоинств потокола Нидхема – Шрёдера можно отметить удобство схемы общения абонентов и доверенного центра.

Если исправить два недостатка предыдущего протокола, то получится **протокол Kerberos**.

1.4. Kerberos

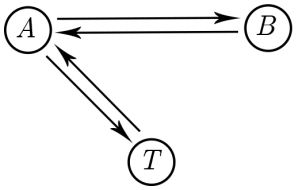

Рис. 9.5

В протоколе Kerberos вместо случайных чисел используются метки времени и время жизни (T и L — timestamp и lifetime).

$$1 \to \{A, B\} \to Trent;$$

$$Trent \to \{E_{\text{A}}(T, L, K, B), E_{\text{B}}(T, L, K, A)\} \to 1;$$
$$1 \to \{E_{\text{K}}(A, T), E_{\text{B}}(T, L, K, A)\} \to 2;$$
$$2 \to \{E_{\text{K}+1}(T + 1)\} \to 1.$$

Сначала первый абонент посылает Тренту сообщение, которым уведомляет его о намерении общаться со вторым абонентом, после чего Трент отсылает первому собеседнику сообщение с ключом и частью, которую первому абоненту будет необходимо отправить второму. В той части, которая была предназначена для первого абонента, содержится

время жизни ключа. Когда второй собеседник получает часть, которую ему переслал первый, он видит, что там указан определенный временной отрезок: дата, год, часы. И через год после того, как злоумышленник, возможно, взломает сессионный ключ, уже за одного из участников беседы он себя выдать не сможет.

Этот протокол короче предыдщего на один шаг. Раньше второй собеседник посылал первому случайное число, а первый отвечал второму, изменив это число на единицу. Сейчас же первый абонент сразу посылает второму доказательство, что он и есть именно тот, за кого себя выдает. То есть он посылает некую метку «1» и время T (то же самое T, что и на втором шаге) и тем самым доказывает, что он знает ключ K. Второй абонент прибавляет единицу к метке времени, зашифровывает ключом и отправляет первому. Тем самым достигается меньшее количество шагов, чем в предыдущем протоколе. Хотя количество данных чуть больше на последних шагах.

2. Разделение секрета

Если в прошлой части предпринимались попытки создать общий секрет для общения между двумя абонентами, то сейчас будет решаться противоположная задача — его разделение.

Задача состоит в следующем: имеется общий ключ, используемый, например, для запуска баллистических ракет. Необходимо разделить этот ключ между тремя генералами таким образом, чтобы никто из генералов не смог запустить ракеты самостоятельно.

Самый простой вариант — например, разделить ключ на три части. Однако эта система разделения секрета будет неидеальной (**идеальной** называется система, зная любой из следов в которой или любое количество следов менее требуемого, нельзя восстановить ни один бит информации о ключе).

Задача состоит в разделении ключа M между несколькими участниками таким образом, чтобы только собравшись вместе, они могли его восстановить. Каждый из них получает некое число или набор чисел, собрав вместе которое, они восстановят сообщение.

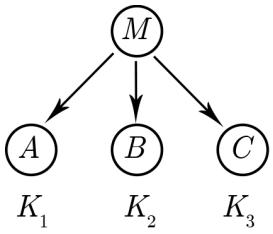

Рис. 9.6

Назовём эти числа «следами». След генерируется либо участниками совместно, либо существует какой-то доверенный центр, который сначала эти следы генерирует, а потом его ликвидирует (потому что он знал исходные сообщения). Поэтому центр генерирует новый ключ, вводит его в центр управления ракетой, разделяет на три части, отдаёт трём генералам, а потом ликвидируется. Идеальной системой будет такая система, в которой взаимная информация между сообщением и всеми тремя следами будет информацией о сообщении. Необходимо уметь восстанавливать сообщение, если есть в наличии

все три следа. Но во всех остальных случаях, то есть если какого-то следа не хватает, должно быть невозможным получить ни одного бита информации о ключе.

$$I(M, k_1, k_2, k_3) = H(M),$$

$$I(M, k_1, k_2) = 0.$$

Схема может быть вида **(n, k)**. (n, k) — пороговая схема, в которой сообщение разделяется на n частей, а для его восстановления надо собрать k частей. То есть, например, если всего 3 генерала, то для запуска ракеты достаточно любых двух генералов. Но при этом один генерал не то что ракету не может запустить, а никакой информации о ключе не должен получить, зная только свой след.

Именно поэтому система, в которой сообщение M делится на 3 части и раздётся каждому поровну, является плохой системой, неидеальной. Рассмотрим систему с точки зрения одного из участников

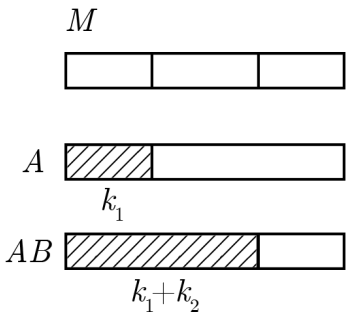

Рис. 9.7

Допустим, у одного из них есть след. Остальную часть он будет пытаться перебирать, причем время перебора будет меньше в соответствии с числом битов, которое ему уже известно. Если сговорились два участника, то им уже окажется известно две третьих части ключа, и им будет намного проще найти перебором остальную часть ключа. Именно поэтому идеальная система разделения секрета не должна давать такой возможности.

3. Схема Блекли

Здесь и далее будем рассматривать пороговые схемы, т. е. схемы, которые не требуют, чтобы все участники собрали свои следы.

В схеме Блекли в качестве секрета берётся координата какой-то точки. Через эту точку проводят n гиперплоскостей (в плоском случае это будут прямые, в случае пространства — плоскости и так далее). Каждому из участников системы раздаются параметры плоскости (гиперплоскости). Для того чтобы восстановить координаты исходной точки нужно, чтобы все участники использовали свои параметры вместе. То есть размерность пространства задаёт число k, то есть сколько необходимо собрать вместе участников для восстановления секрета.

Количество таких следов может быть, сколько участников системы — неограниченно. Однако криптографией везде используются только целые числа. Именно поэтому в схеме Блекли все параметры берутся по модулю числа p, где p — некое простое число. При

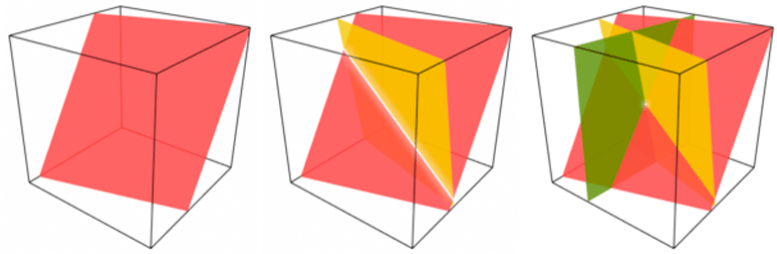

Рис. 9.8

этом вся геометрия остаётся прежней в предположении, что p — простое число. Если оно простое, то две прямые будут пересекаться только в одной точке.

Таким образом, в схеме Блекли в качестве сообщения берётся координата точки x. Дальше генерируется набор плоскостей, которые проходят через эту точку, и раздаются участникам четыре числа, которые описывают в трёхмерном случае уравнение плоскости. Все вычисления проходят по модулю числа p, которое будет являться секретом. Собравшись вместе, три человека могут построить систему уравнений и восстановить секрет.

Предположим, что злоумышленникам известно только две плоскости в трёхмерном случае. Две плоскости пересекаются по прямой, а это значит, что те, кто пытается восстановить секрет, не имея всех следов, знают уравнение прямой, на которой находится искомая точка. Казалось бы, они ограничили число вариантов, но это не так. Дело в том, что на этой прямой координата точки x может принимать любые значения.

4. Схема Шамира

Схему Шамира также часто называют схемой интерполяционных полиномов Лагранжа, или просто схемой Лагранжа. В её основе лежит факт, что для восстановления полинома второй степени нужно задать три точки на плоскости, для полинома третьей степени — четыре точки на плоскости. А через две точки может проходить бесконечное число полиномов второй степени.

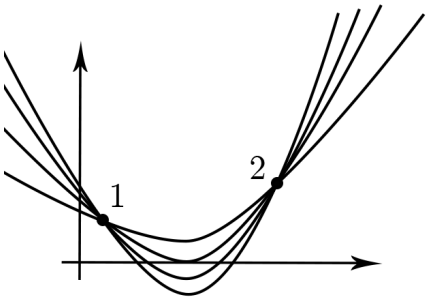

Рис. 9.9

Предположим, злоумышленники знают две точки из трёх и пытаются восстановить секрет. Секрет — это свободный коэффициент в многочлене, то есть в уравнении параболы. Злоумышленники пытаются восстановить его по двум точкам, но через две точки

проходит бесконечное множество парабол со всеми возможными свободными коэффициентами. Получается, что, даже зная (k–1) след, злоумышленники не могут восстановить секрет.

При составлении схемы выбирают некоторое простое число $p > M$. Оно задаёт конечное поле размера p. Над этим полем строится многочлен степени (k–1) (то есть случайно выбираются все коэффициенты многолчена, кроме M).

$$F(x) = (a_{k-1}x^{k-1} + a_{k-2}x^{k-2} + \dots + a_1 x + M) \mod p.$$

В этом многочлене M — это разделяемый секрет, а остальные коэффициенты $a_{k-1}, a_{k-2}, \dots, a_1$ — некоторые случайные числа, которые нужно будет «забыть» после завершения процедуры разделения секрета.

Затем вычисляются координаты различных n точек:

$$k_1 = F(1) = (a_{k-1}1^{k-1} + a_{k-2}1^{k-2} + \dots + a_1 \cdot 1 + M) \mod p,$$

$$k_2 = F(2) = (a_{k-1}2^{k-1} + a_{k-2}2^{k-2} + \dots + a_1 \cdot 2 + M) \mod p,$$

$$\dots$$

$$k_n = F(n) = (a_{k-1}n^{k-1} + a_{k-2}n^{k-2} + \dots + a_1 \cdot n + M) \mod p.$$

Аргументы (номера секретов) не обязательно должны идти по порядку, главное — чтобы все они были различны под модулю p.

Чтобы восстанавить секрет, можно записать набор уравнений по модулю числа p. Дальше нужно использовать следы. В качестве следа берётся координата точки и размерность. Для восстановления уравнения можно решить систему уравнений (в случае плоскости нужно собрать три нужных следа и составить три уравнения, из которых найти коэффициенты параболы).

Есть ещё один способ — схема полиномов Лагранжа.

$$F(x) = \sum_i l_i(x) y_i \mod p,$$

$$l_i(x) = \prod_{i \neq j} \frac{x - x_j}{x_i - x_j} \mod p.$$

5. Схемы Миньотта и Асмута – Блума

Эти схемы основываются на **китайской теореме об остатках**.

Теорема 1 (Китайская теорема об остатках) Если натуральные числа a_1, a_2, ..., a_n попарно взаимно просты, то для любых целых r_1, r_2, ..., r_n таких, что $0 \leqslant r_i < a_i$ при всех $i \in \{1, 2, \dots, n\}$, найдётся число N, которое при делении на a_i даёт остаток r_i при всех $i \in \{1, 2, \dots, n\}$.

Более того, если найдутся два таких числа N_1 и N_2, то $N_1 \equiv N_2 \mod a_1 \times a_2 \times \dots \times a_n$.*

Основная идея этой теоремы состоит в том, что можно ввести некий базис разложения чисел по взаимно простым числам. Координатами в этом базисе будут остатки

от деления на $a_1 \dots a_n$. Любые простые числа от нуля до произведения a_n они будут задаваться однозначно; если же какие-то два числа раскладываются по этому базису с одними и теми же координатами, то эти два числа равны с точностью до модуля произведений чисел a_i.

Обе эти схемы основываются на следующей идее: выбирается набор чисел D (достаточно большой). Эти числа перемножаются. Получается некое число, и дальше за пределами этого числа выбирается секрет. Чтобы этот секрет восстановить, потом потребуются остатки от деления этого числа на выбранные числа $d_1, d_2, \dots, d_n$.

В схеме Миньотта строится (k, n)-последовательность Миньотта:

$$d_1 < d_2 < \dots < d_n \quad (d_i, d_j) = 1 \text{ для } i \neq j;$$

$$M > \prod_{i=1}^{n} d_i.$$

В схеме Асмута – Блума выбирается простое число $p > M$. Затем выбираются простые числа $d_1, d_2, \dots, d_n$, такие, что:

$$d_i > p, \quad d_i < d_{i+1}.$$

Затем вычисляется их произведение. После этого со случайным числом r вычисляется $M' = M + rp$. Потом вычисляются доли: $k_i = M' \mod d_i$. Секретами будут являться наборы чисел: $\{p, d_i, k_i\}$.

6. Схема Блома

Схему Блома ошибочно относят к схемам разделения секрета, но она ей не является.

Схема Блома — это схема распределения ключей с доверенным центром. Можно рассматривать это как вариант протокола распределения ключей, но тут уже используется в некотором роде асимметричная криптография.

Эта схема используется в протоколе High-speed Digital Content Protection. Это схема, запатентованная компанией Intel, для защиты контента от копирования. Именно она защищает контент, передаваемый между видеокартой и цифровым монитором или телевизором. Для выработки сеансового ключа в кабеле передачи данных шифрованию подвергается большое количество данных. Эта схема работает над конечным полем $GF(p)$.

Имеется доверенный центр T (назовём его условно компанией Intel), который владеет большой симметричной матрицей $D_{k,k}$ над полем $GF(p)$. Каждый из участников системы выбирает себе вектор I, который называется идентификатором. Если этот вектор k достаточно длинный (128), то все идентификаторы будут линейно независимы.

Центр T умножает идентификатор на матрицу и получает секретный ключ g:

$$g = D_{k,k} \cdot I.$$

После чего между, например, видеокартой и монитором можно установить канал связи чтобы абсолютно секретным образом передать на монитор изображение или фильм (но это не защищает он видеокамеры, поставленной перед монитором (проблема «анало-

говой дырки»)). Видеокарта и монитор пересылают друг другу свои идентификаторы, и потом эти идентификаторы умножаются на секретный ключ.

$$A \to \{I_A\} \to B,$$

$$B \to \{I_B\} \to A.$$

$$A: \quad S_A = G_A\, {}^t I_B,$$

$$B: \quad S_B = G_B\, {}^t I_A.$$

$$S = S_A = S_B.$$

Если эти ключи сгенерированы на одной и той же матрице D (то есть одного и того же центра), то получится общий ключ: значения S_A и S_B, вычисленные на мониторе и видеокарте, совпадут. И это число можно использовать как пароль шифрования.

Линейная независимость применяется в этой схема по той причине, что если ключи будут линейно зависимыми, то тогда, зная ключи между двумя участниками, можно найти третий ключ с третьим участником, у которого идентификатор линейно связан с одним из этих двух.

Матрица D может быть восстановлена, если известно k линейно независимых идентификаторов и соответствующие им закрытые ключи. Зная только один I и g, ничего нельзя будет сделать. Если известно 56 штук, то можно попытаться взломать схему, которую использовала компания Intel.