Ключи и протоколы для них

Сергей Николенко

Computer Science Club, 2015

Outline

1. Введение и Диффи-Хеллман
   - Введение в протоколы согласования ключа
   - Протокол Диффи-Хеллмана
   - Цели протоколов согласования ключа
2. Протоколы согласования ключа
   - Протоколы на криптографии с закрытым ключом
   - Протоколы согласования ключа с Центром
   - Протоколы на криптографии с открытым ключом
3. Атаки, распределение ключей и разделение секрета
   - Классические атаки
   - Распределение ключей
   - Разделение секрета

Идея

- Мы уже научились передавать сообщения криптосистемами с открытым ключом.
- Но это довольно медленно и большой overhead получается.
- С секретным ключом всё было бы гораздо быстрее.
- Что делать?

Идея

- Мы уже научились передавать сообщения криптосистемами с открытым ключом.
- Но это довольно медленно и большой overhead получается.
- С секретным ключом всё было бы гораздо быстрее.
- Что делать?
- Надо при помощи протоколов с открытым ключом установить секретный ключ, а потом шифровать им.

Протоколы согласования ключа

- И вот мы уже знаем простейший протокол согласования ключа:
  1. Алиса публикует публичный ключ для какой-нибудь криптосистемы с открытым ключом;
  2. Боб выбирает секретный ключ K, шифрует его, передаёт Алисе.
  3. Алиса декодирует, и теперь у Алисы и Боба есть общий ключ K, которым можно шифровать.
- Если Алиса и Боб абсолютно доверяют друг другу и верят, что враг может проводить только пассивные атаки (только следить за сообщениями, не изменяя их и не прикидываясь участниками протокола), то так можно сделать.
- Но нужно готовиться к худшему.

Что будет сегодня

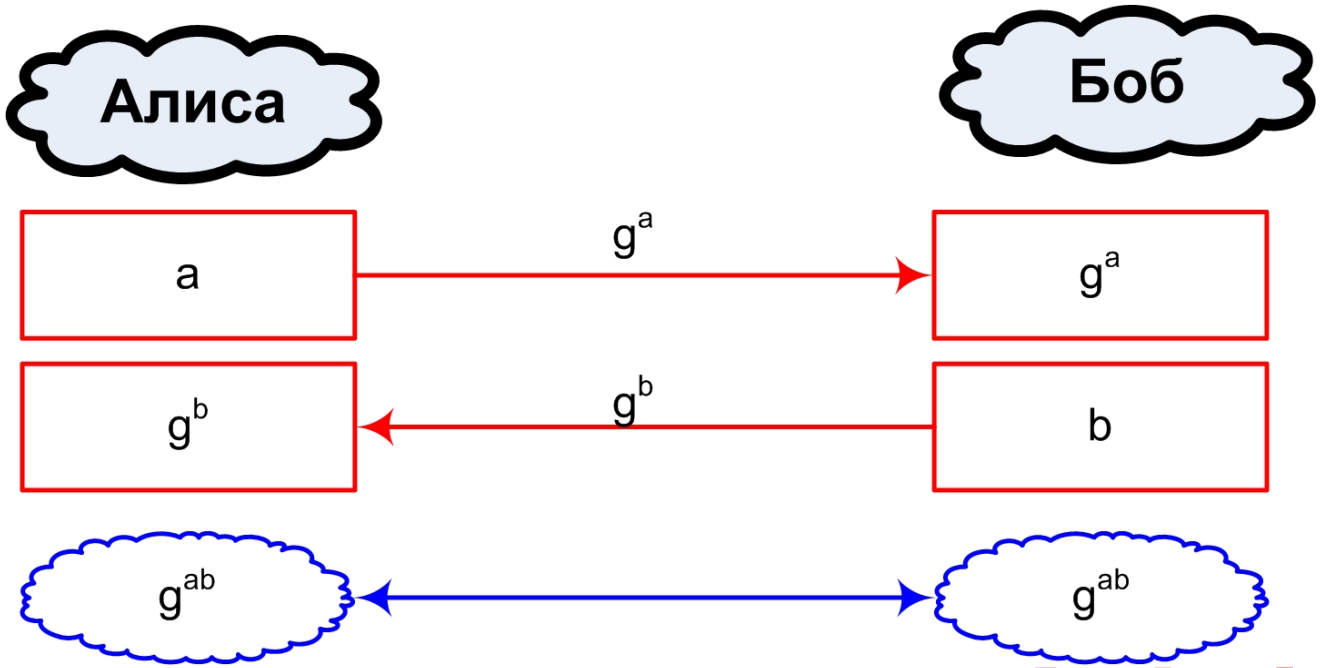
- Мы сегодня начнём с исторически первого примитива криптографии с открытым ключом вообще.
- Затем поймём, что он никакой критики не выдерживает.
- Затем попробуем придумать что-нибудь получше.
- А затем обобщим задачу согласования ключа с двух участников на большее их количество.

История

- Whitfield Diffie, Martin Hellman, 1976.
  - Как водится, потом выяснилось, что британец Malcolm J. Williamson, работавший на разведку, придумал то же самое раньше, но его засекретили.
- Это будет для нас первый протокол, основанный на задаче дискретного логарифма.

Идея и структура

- Цель: установить общий ключ. Публично выбирается модуль n и число g, взаимно простое с n.
- Алиса выбирает a, Боб выбирает b. Алиса посылает g^a, Боб посылает g^b. В результате каждый может вычислить g^{ab}.



Замечания

- Чем мы пользовались для группы $\mathbb{Z}_n$? На какие группы можно обобщить?

Замечания

- Чем мы пользовались для группы $\mathbb{Z}_n$? На какие группы можно обобщить?
- Протокол Диффи-Хеллмана работает в любой коммутативной группе.
- А какая задача стоит перед пассивным взломщиком?

Замечания

- Чем мы пользовались для группы $\mathbb{Z}_n$? На какие группы можно обобщить?
- Протокол Диффи-Хеллмана работает в любой коммутативной группе.
- А какая задача стоит перед пассивным взломщиком?
- По g^a и g^b найти g^{ab}. Это не дискретный логарифм, а Diffie-Hellman problem, как в RSA.

Криптосистема Эль-Гамаля

- Из протокола Диффи-Хеллмана можно сделать настоящую криптосистему.
- Генерация ключей.
  1. Алиса выбирает случайную степень x, подсчитывает $h = g^x$.
  2. Выдаёт h, g и группу в виде публичного ключа. Секретный ключ — x.
- Кодирование.
  1. Боб выбирает случайный y, вычисляет $c_1 = g^y$. У него получается «общий секрет» $s = h^y$.
  2. Сообщение m Боб кодирует как $c_2 = ms$.
  3. Посылает Алисе (c_1, c_2).
- Декодирование.
  1. Алиса вычисляет $s = c_1^x$ и декодирует $m = c_2 s^{-1}$.

Анализ надёжности схемы Diffie-Hellman

- Мы знаем, что DH надёжен против пассивных атак, если DHP не решить.
- Т.е. против Чарли, который подглядывает за Алисой и Бобом, DH надёжен. А ещё?
- Может ли Алиса быть уверенной, что разговаривает именно с Бобом?

Анализ надёжности схемы Diffie-Hellman

- Мы знаем, что DH надёжен против пассивных атак, если DHP не решить.
- Т.е. против Чарли, который подглядывает за Алисой и Бобом, DH надёжен. А ещё?
- Может ли Алиса быть уверенной, что разговаривает именно с Бобом?
  - Конечно, нет. Чарли может легко имитировать Боба, нет никакого подтверждения личности.
- Может ли Алиса быть уверенной, что только Боб узнает секретный ключ, а Чарли, даже если будет имитировать Боба, не узнает?

Анализ надёжности схемы Diffie-Hellman

- Мы знаем, что DH надёжен против пассивных атак, если DHP не решить.
- Т.е. против Чарли, который подглядывает за Алисой и Бобом, DH надёжен. А ещё?
- Может ли Алиса быть уверенной, что разговаривает именно с Бобом?
  - Конечно, нет. Чарли может легко имитировать Боба, нет никакого подтверждения личности.
- Может ли Алиса быть уверенной, что только Боб узнает секретный ключ, а Чарли, даже если будет имитировать Боба, не узнает?
  - Конечно, нет, по той же причине.
- Но прежде чем формулировать, что мы от протоколов хотим, давайте сформулируем, какие нас могут интересовать протоколы.

Протоколы

- *Протокол передачи ключа* (key transport): Алиса секретно передаёт Бобу конкретный выбранный Алисой ключ K.
- *Протокол согласования ключа* (key agreement): Алиса и Боб договариваются о каком-нибудь общем ключе.
- *Протокол обновления ключа* (key update): у Алисы и Боба уже есть секретный ключ, но они для новых сессий используют свежие разные ключи.
- *Протокол раздачи ключей* (key distribution): Центр раздаёт резидентам ключи, при помощи которых можно общаться друг с другом и с Центром.

Атаки

- *Пассивный противник* может только подсматривать (и называется eavesdropper).
- *Активный противник* вставляет свои сообщения (intruder), прикидывается честным участником (impersonator), может модифицировать любые передаваемые честными участниками сообщения, может начинать несколько разговоров с честным участником и бросать один из них на полпути.
- Ещё сильнее — *known-key attack*: протокол устойчив против неё, если Чарли, узнав секретные ключи от нескольких разговоров Алисы и Боба, не сможет разгадать новый свежий секретный ключ (сгенерированный с прежними перманентными ключами).

Желаемые свойства протоколов

- *Аутентификация личности* (entity authentication): Алиса уверена, что с ней разговаривает Боб.
- *Аутентификация источника данных* (data origin authentication): Алиса уверена, что именно Боб написал это сообщение.
- *Аутентификация ключа* (key authentication): Алиса уверена, что только Боб (и ещё, может быть, Центр) сможет узнать секретный ключ.
- *Подтверждение ключа* (key confirmation): Алиса уверена, что Боб ключ получил и готов его использовать.

Желаемые свойства протоколов

- *Идеальная прямая безопасность* (perfect forward secrecy): если Чарли узнает настоящий перманентный секретный ключ, это не поможет ему взломать частные ключи от предыдущих, уже состоявшихся обменов между Алисой и Бобом.
- *Свежесть ключа* (key freshness): участники уверены, что ключ свежий и раньше не использовался.
- Как нетрудно видеть, DH мало чему удовлетворяет. Но зато даёт perfect forward secrecy: нету никаких перманентных ключей.

Outline

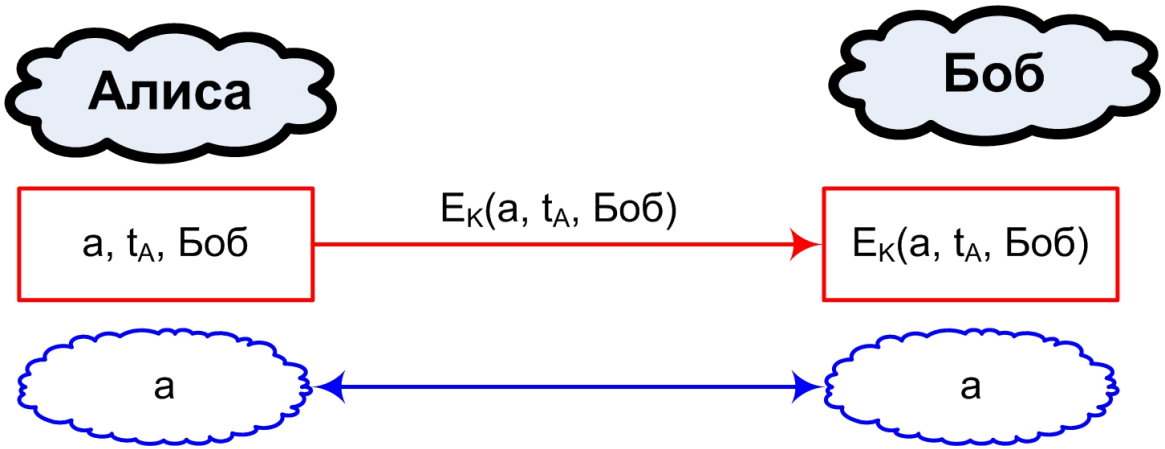
1. Введение и Диффи-Хеллман
   - Введение в протоколы согласования ключа
   - Протокол Диффи-Хеллмана
   - Цели протоколов согласования ключа
2. Протоколы согласования ключа
   - Протоколы на криптографии с закрытым ключом
   - Протоколы согласования ключа с Центром
   - Протоколы на криптографии с открытым ключом
3. Атаки, распределение ключей и разделение секрета
   - Классические атаки
   - Распределение ключей
   - Разделение секрета

Идея

- Протоколы будут основаны на алгоритмах симметрической криптографии (т.е. криптографии с закрытым ключом).
- Часто мы будем предполагать, что у Алисы и Боба уже есть секретный ключ, но они хотят для каждой сессии устанавливать новый секретный ключ, чтобы меньше давать Чарли информации. Т.е. фактически мы занимаемся key update.
- Начнём с передачи ключа, когда Алиса хочет передать Бобу выбранный ею ключ.

Однопроходовый key transport

- Простейший протокол:

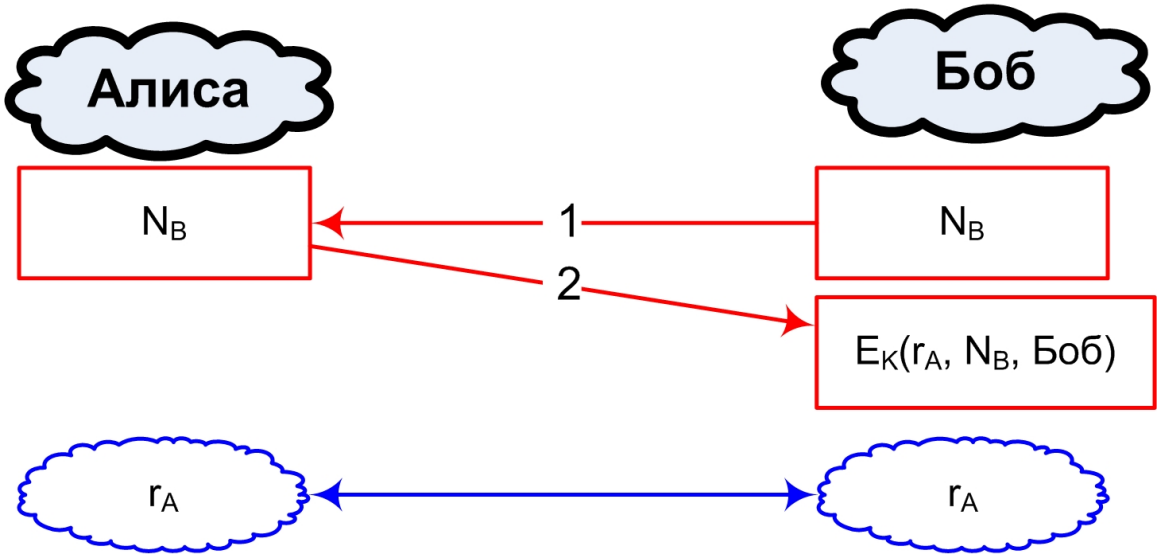


Однопроходовый key transport

- Хорошо:
  - Key authentication: Алиса не уверена, что Боб получил ключ, но уверена, что получить его может только Боб.
  - Key freshness: Алиса может добавить в *a* timestamp, и Боб будет уверен, что ключ свежий.
  - Односторонний key confirmation: Алиса добавляет в *a* что-то повторяющееся, и Боб уверен, что *a* придумала именно Алиса, а не Чарли послал случайное число. Но Алиса ни в чём не уверена, конечно.
- Плохо:
  - Против known-key attack устойчивости нет.
  - Алиса ни в чём не уверена, потому что ничего не получает.
  - Perfect forward secrecy нет.

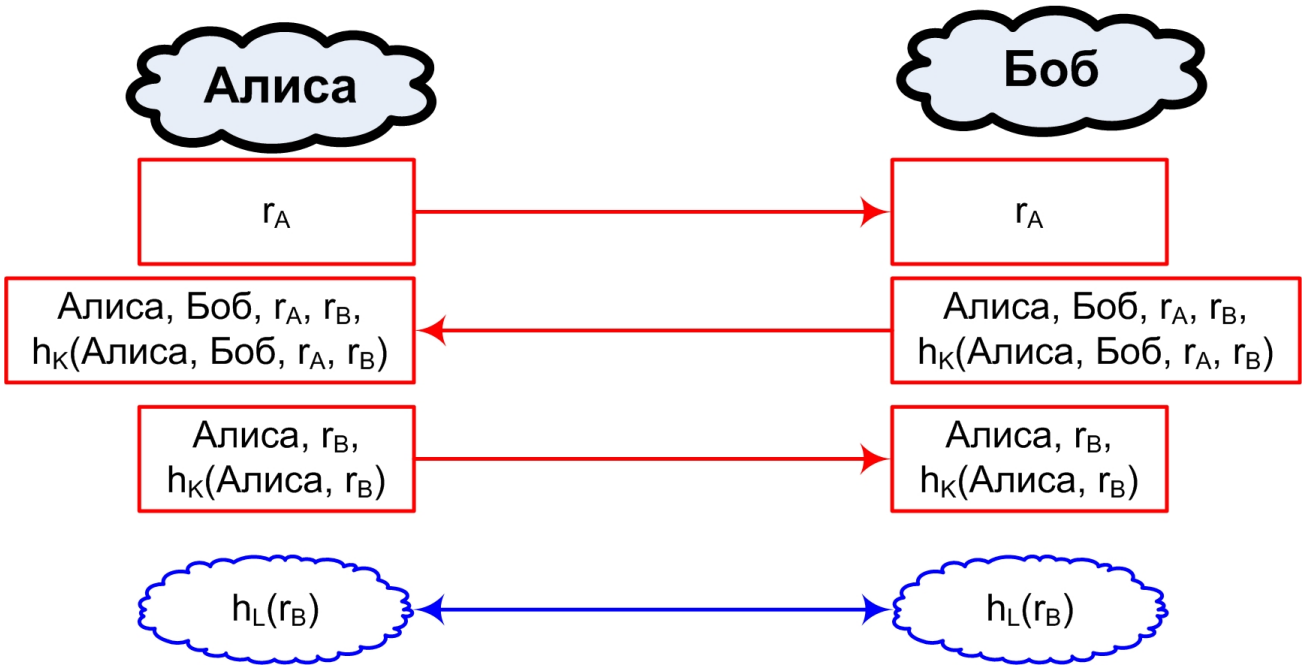
Challenge-response

- Протокол challenge-response. Теперь Боб точно уверен, что Алиса — это Алиса; но проблемы те же.



Authenticated Key Exchange Protocol

- За три прохода можно сделать аутентификацию, причём без декодирования, через хеш-функции. Предположим, что мы умеем вычислять h_K (семейство хеш-функций с ключом), и у Алисы и Боба есть два ключа K и L.

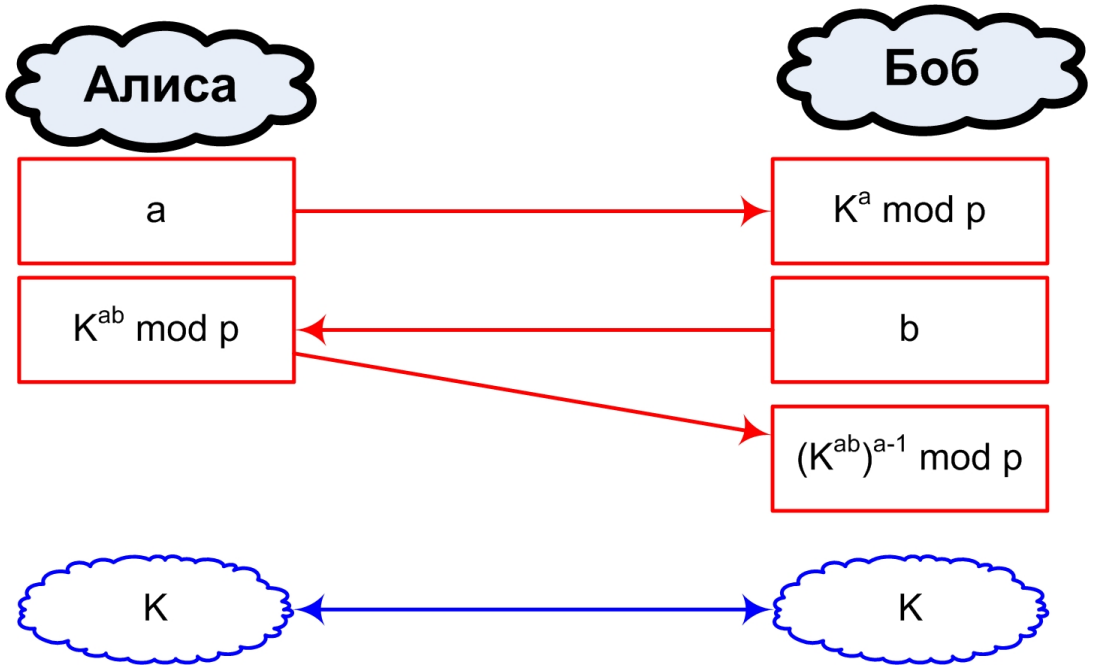


Authenticated Key Exchange Protocol

- Теперь Алиса уверена, что Боб — это Боб и что он получил ключ.
- А Боб уверен, что Алиса — это Алиса и что она получила ключ.
- Т.е. взаимное key authentication и key confirmation.

Протокол Шамира

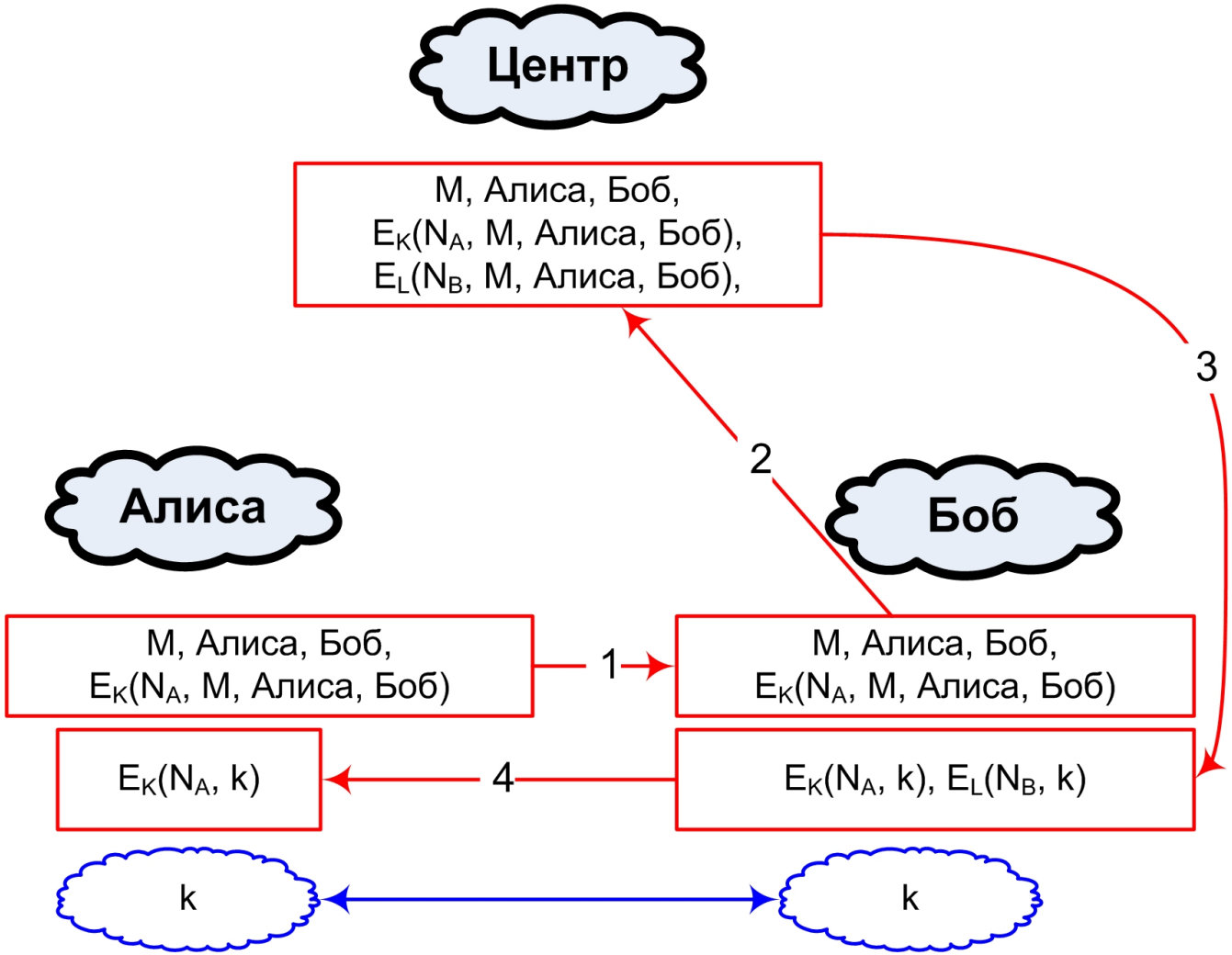
- А можно и вообще без ключей, организовать как в DH, но key transport.



Протоколы с сервером

- Теперь давайте предположим, что у нас есть сервер, которому верят и Алиса, и Боб.
- У сервера и Алисы есть заранее секретный ключ K, а у сервера и Боба — секретный ключ L.
- Алиса и Боб хотят сделать временный секретный ключ для общения друг с другом.
- Протокол Отвэя-Рииса (Otway-Rees).
- На следующем слайде Алиса выбирает числа M и N_A, Боб выбирает N_B, а центр — k.

Протокол Отвэя-Рииса



Протокол Отвэя-Рииса

- Здесь, если все проверки успешно пройдены, то:
  - Алиса уверена, что получила ключ k и что Боб его знает;
  - Боб уверен, что настоящий Центр выдал ему новый временный ключ k для общения с Алисой.
- Т.е. достигается key authentication и key freshness.

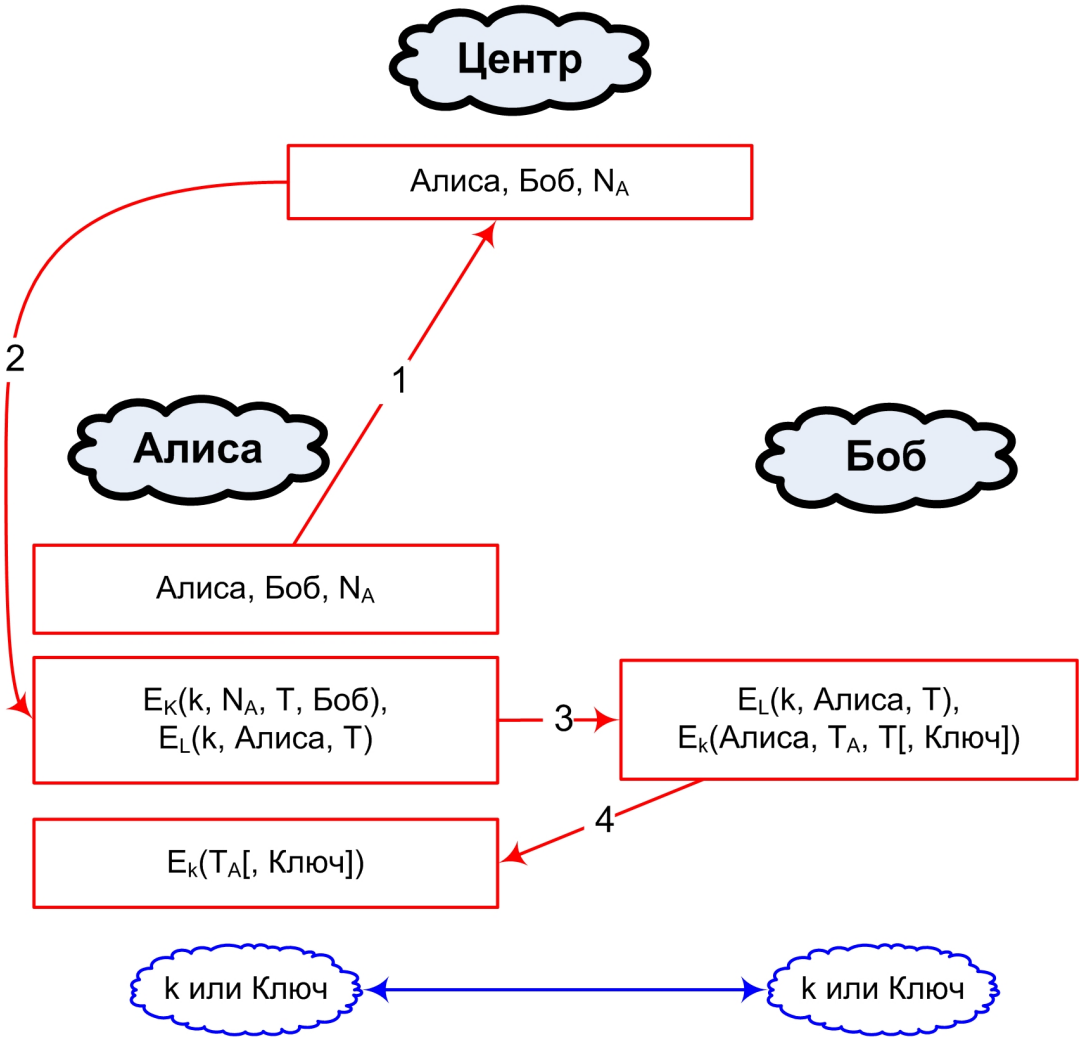
Kerberos

- Kerberos — это:
  - распределённая система аутентификации, созданная в рамках проекта Athena в MIT;
  - софт, созданный для этой системы;
  - протокол аутентификации, разработанный для этой системы.
- Главная цель — аутентификация (как раз entity), но как побочный эффект и ключ согласовывается.
- Сейчас используется в Windows (начиная с 2000), Mac OS X, Red Hat, Solaris, FreeBSD как система аутентификации по умолчанию.

Kerberos

- Обозначения:
  - Алиса и Центр знают секретный ключ K;
  - Боб и Центр знают секретный ключ L;
  - Алиса выбирает N_A и вводит timestamp T_A по своим часам;
  - Центр выбирает временный ключ k для Алисы и Боба;
  - T — период валидности (lifetime), выбираемый Центром.
- Идея: Центр посылает Алисе ticket, зашифрованный ключом Боба, и Алиса пересылает его как подтверждение своих намерений.

Kerberos


Центр
Алиса, Боб, N_A
2
1
Алиса
Боб
Алиса, Боб, N_A
$E_K(k, N_A, T, Боб)$,
$E_L(k, Алиса, T)$
3
$E_L(k, Алиса, T)$,
$E_k(Алиса, T_A, T[, Ключ])$
4
$E_k(T_A[, Ключ])$
k или Ключ
k или Ключ

Kerberos

- Алгоритм Kerberos.
  1. Сначала Алиса запрашивает аутентификацию у центра прямым текстом.
  2. Центр посылает Алисе зашифрованные её ключом подтверждение, ключ k и время жизни ticket'а, а также сам ticket — ключ k, время его жизни и личность Алисы, зашифрованные ключом Боба.
  3. Алиса пересылает их Бобу вместе со своей личностью и lifetime'ом, зашифрованными новым ключом, и тот может проверить, что это Алиса, и что сервер ей действительно выдал новый ключ, и что ticket ещё жив.
  4. Чтобы подтвердить, что он всё понял, Боб высылает Алисе обратно зашифрованный новым ключом T_A.
- Если нужно сделать key transport, то Алиса может просто добавить в своё письмо Бобу желаемый ею ключ, а Боб вернёт его же, подтвердив, что понял

Kerberos

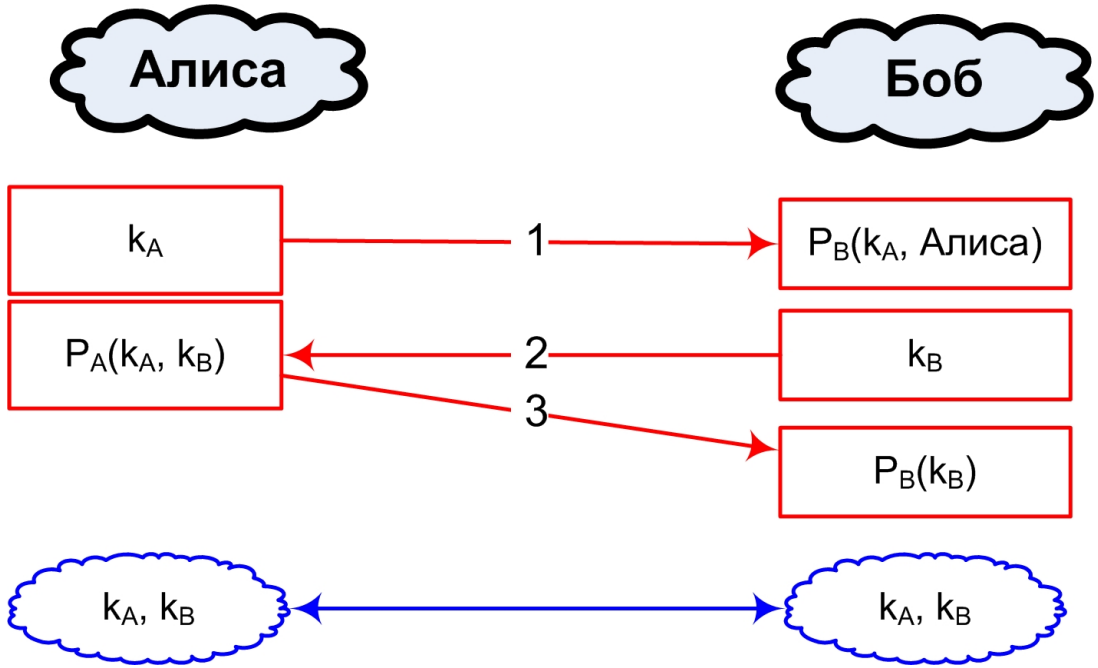
- Получаются взаимные entity authentication и key confirmation.
- Если Алисе нужно ещё раз поговорить с Бобом с другим ключом, она может просто переслать тот же ticket и новые authenticator, не обращаясь к серверу вовсе.
- Если время T ещё не истекло, Боб примет тот же ticket. Это упрощает key update.
- Timestamp T_A нужен, чтобы защититься от атакующего, повторяющего те же сообщения (общий приём). Сообщение будет валидно только в течение короткого времени и, более того, два сообщения с одинаковым timestamp Боб не примет.
- Отсюда проблема: надо, чтобы часы были у всех синхронизированы.

Простейший протокол

- Теперь рассмотрим протоколы, основанные на криптографии с открытым ключом.
- У Алисы и Боба есть публичные процедуры кодирования P_A и P_B.
- Простейший протокол: Боб, желая передать ключ k Алисе, кодирует его и передаёт $P_A(k)$.
- Боб уверен, что только Алиса сможет узнать ключ (key authentication), но не уверен, что Алиса его получила; Алиса ничего о Бобе не знает.

Needham-Schroeder

- Простой протокол, предоставляет взаимную передачу ключей k_A и k_B, а также взаимную аутентификацию.



Цифровая подпись

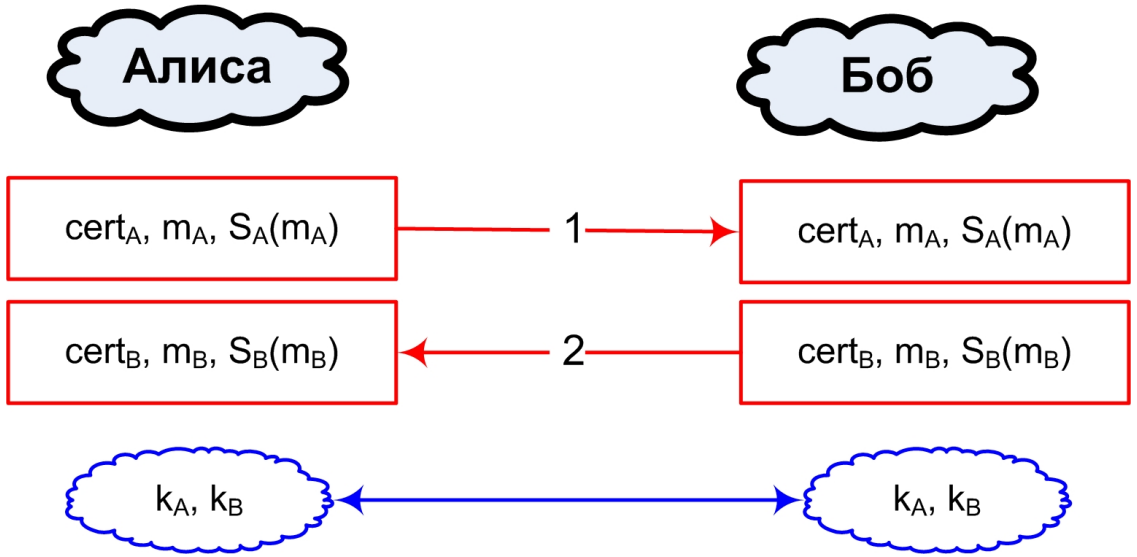
- Теперь давайте предположим, что Алиса и Боб умеют ещё и *подписывать* свои сообщения.
- Напоминание: цифровая подпись — это когда каждый может проверить, что подписала Алиса, но никто не может подписать сам за Алису.
- Например, в RSA-схеме Алиса выбирает модуль n, экспоненту e и обратную к ней d и подписывает сообщение m так:
$$S_A(m) = m^d \pmod{n}.$$
- Каждый может проверить публичным ключом, но никто, не зная секретного, не может подделать подпись.
- Будем предполагать, что у Алисы есть подпись S_A, у Боба — S_B.

X.509

- Алгоритм X.509 — стандарт аутентификации с открытым ключом.
- Он предполагает систему *сертификатов*, которые выпускают специальные доверенные стороны.
- Сертификат Алисы, выпущенный Центром, содержит публичные ключи Алисы для подписи и кодирования, а также подписан Центром, т.е. его никто не может подделать.

X.509 за два прохода

- Здесь сообщение Алисы $m_A = (t_A, r_A, \text{Боб}, P_B(k_A))$, а сообщение Боба $m_A = (t_B, r_B, \text{Алиса}, r_A, P_A(k_B))$ (t — timestamp, который не должен оказаться слишком давним, r — случайное число, которое не должно повториться).
- В результате происходит аутентификация и обмен секретными ключами k_A и k_B.



Outline

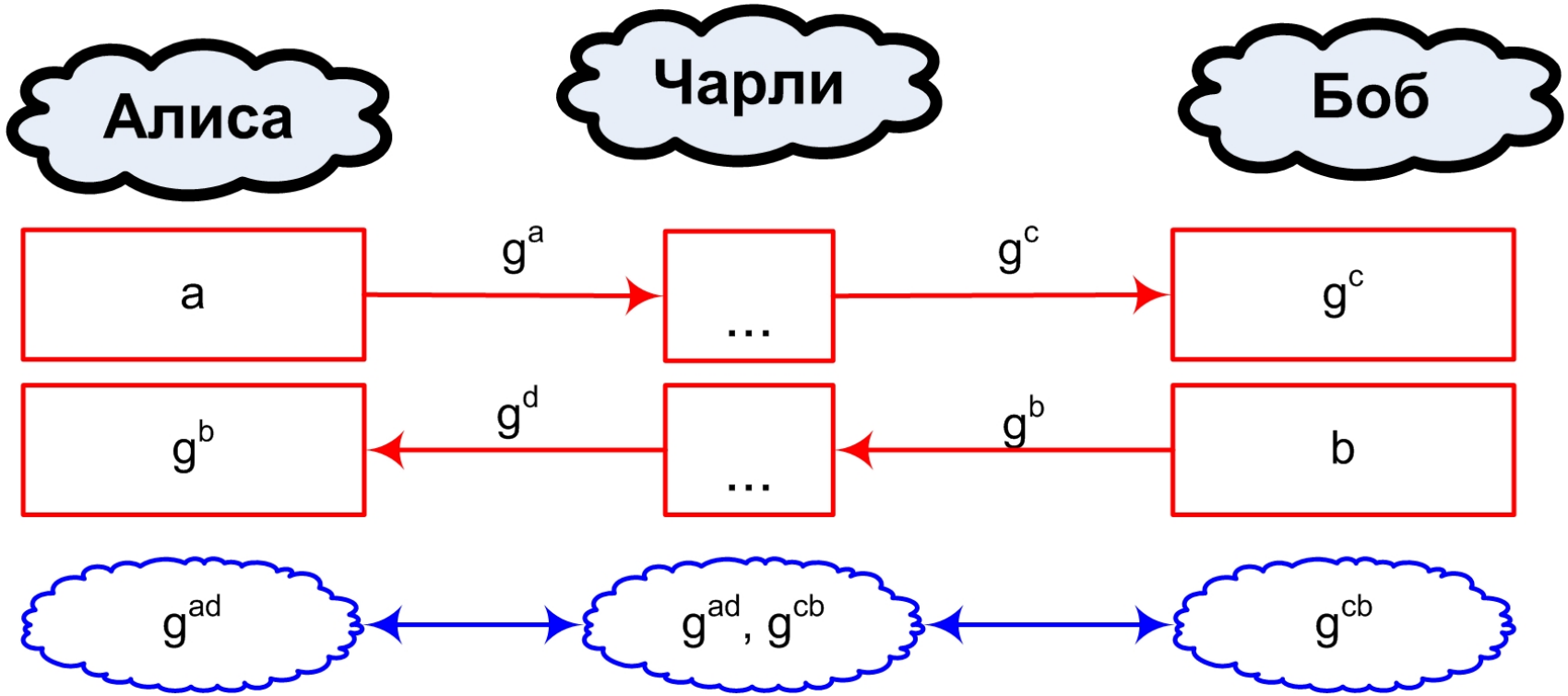
1. Введение и Диффи-Хеллман
   - Введение в протоколы согласования ключа
   - Протокол Диффи-Хеллмана
   - Цели протоколов согласования ключа
2. Протоколы согласования ключа
   - Протоколы на криптографии с закрытым ключом
   - Протоколы согласования ключа с Центром
   - Протоколы на криптографии с открытым ключом
3. Атаки, распределение ключей и разделение секрета
   - Классические атаки
   - Распределение ключей
   - Разделение секрета

Атаки

- Сейчас мы поговорим о том, чего стоит опасаться при разработке протоколов согласования ключа.
- Всякий реальный протокол должен быть устойчив против известных атак.
- Мы сейчас приведём несколько примеров классических уязвимостей в протоколах.

Man-in-the-middle

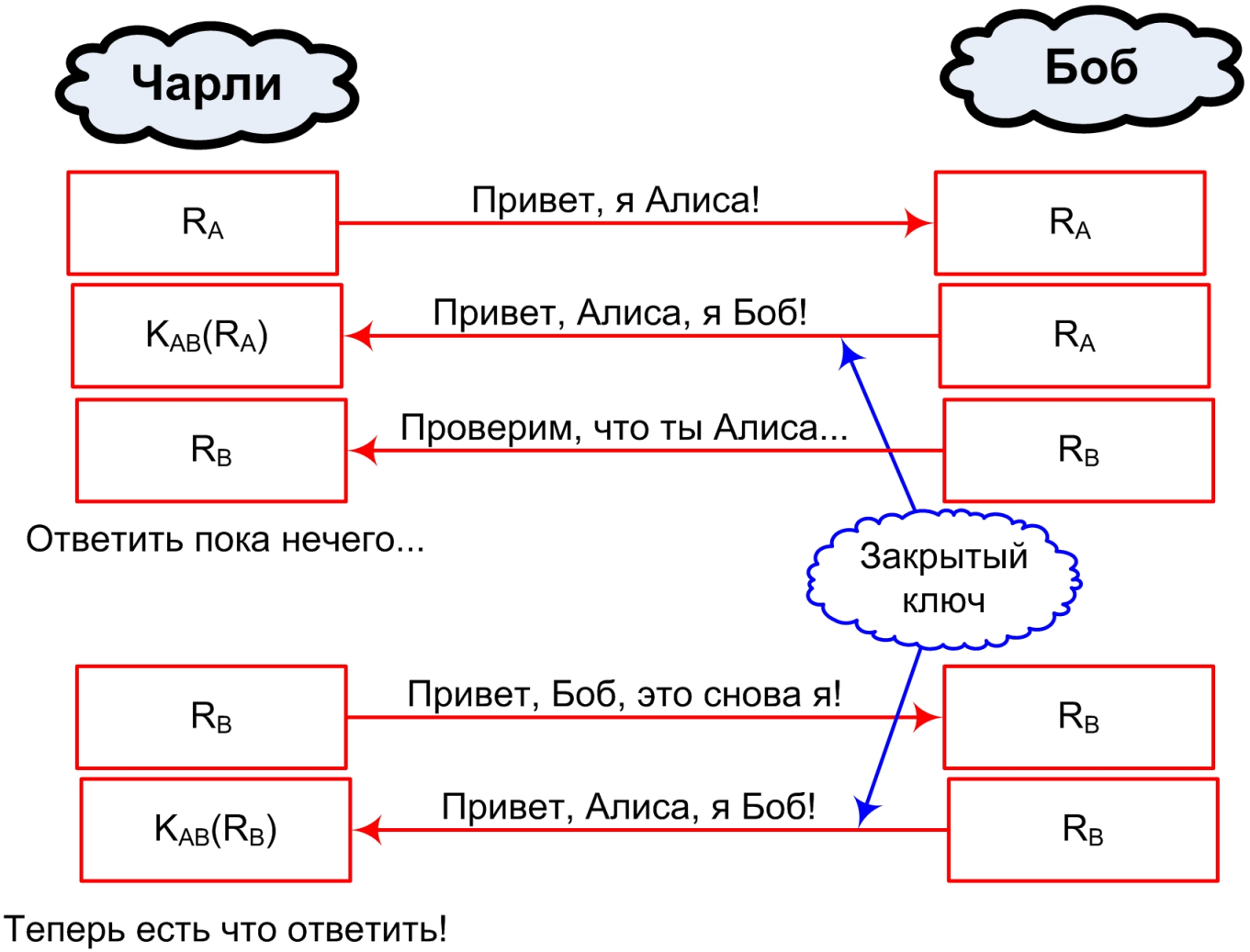
- Обычный Diffie-Hellman не справляется с атакой man-in-the-middle.



Reflection attack

- Предположим, что Алиса и Боб имеют общий секрет K и аутентифицируют друг друга так:
  1. Алиса посылает Бобу r_A;
  2. Боб отвечает $E_K(r_A, r_B)$;
  3. Алиса отвечает r_B.
- Как взломать такую систему?

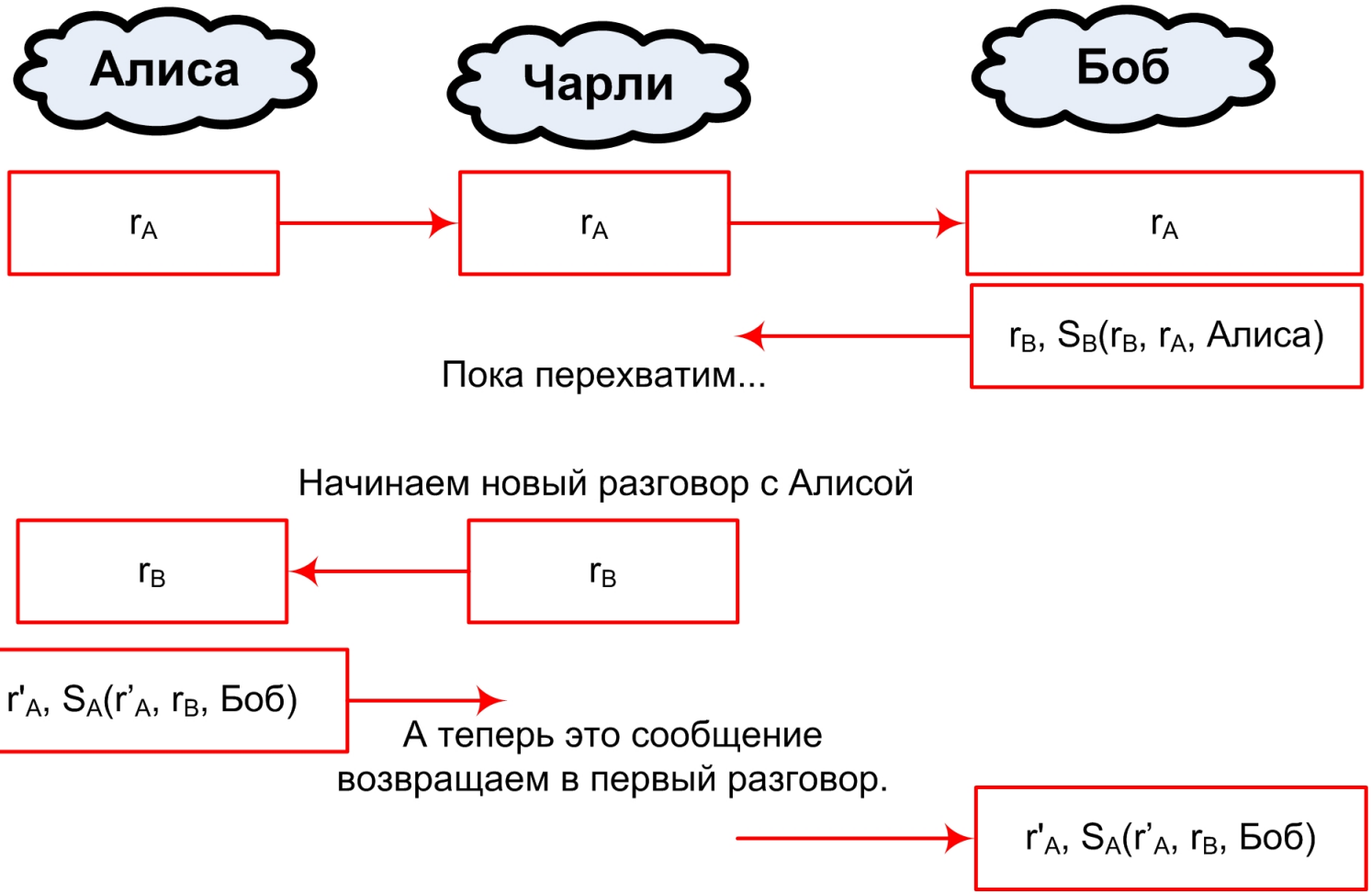
Reflection attack



Interleaving attack

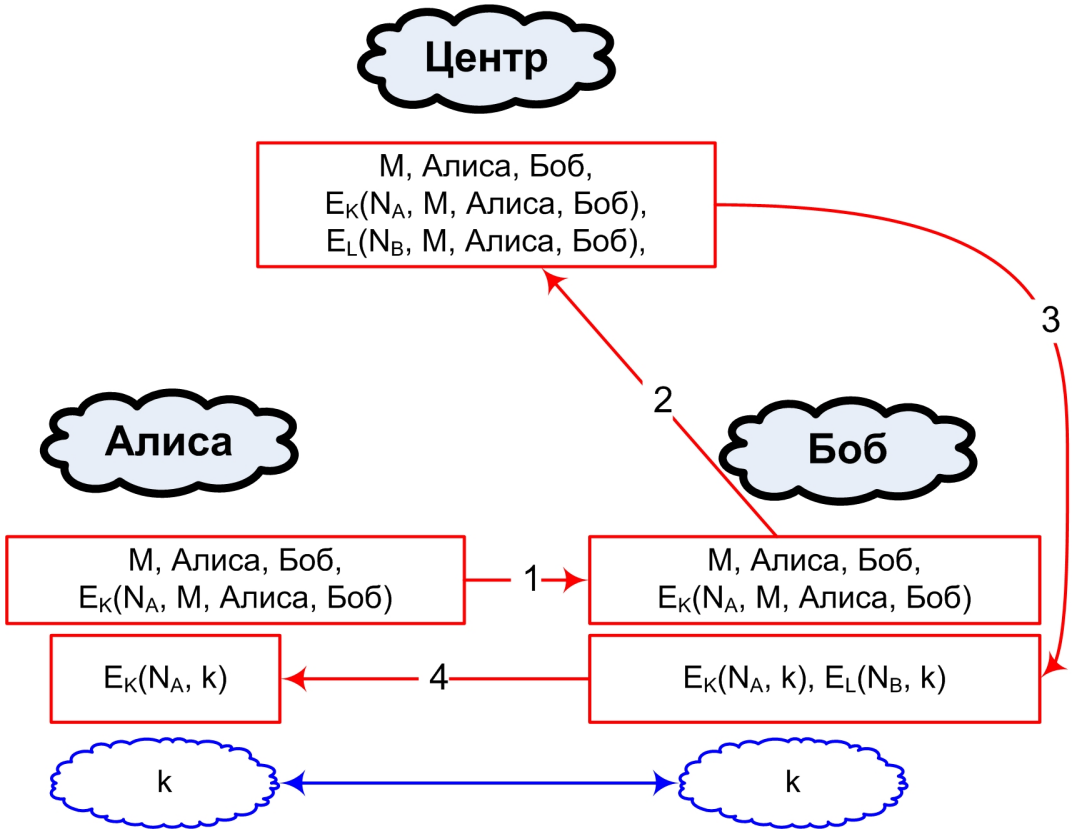
- Предположим, что Алиса и Боб аутентифицируют друг друга при помощи цифровых подписей:
  1. Алиса посылает Бобу r_A;
  2. Боб отвечает $r_B, S_B(r_A, r_B, \text{Алиса})$;
  3. Алиса отвечает $r'_A, S_A(r'_A, r_B, \text{Боб})$.
- Казалось бы, здесь должна быть и свежесть (за счёт новых r_A, r_B), и entity authentication...

Interleaving attack



Misplaced trust

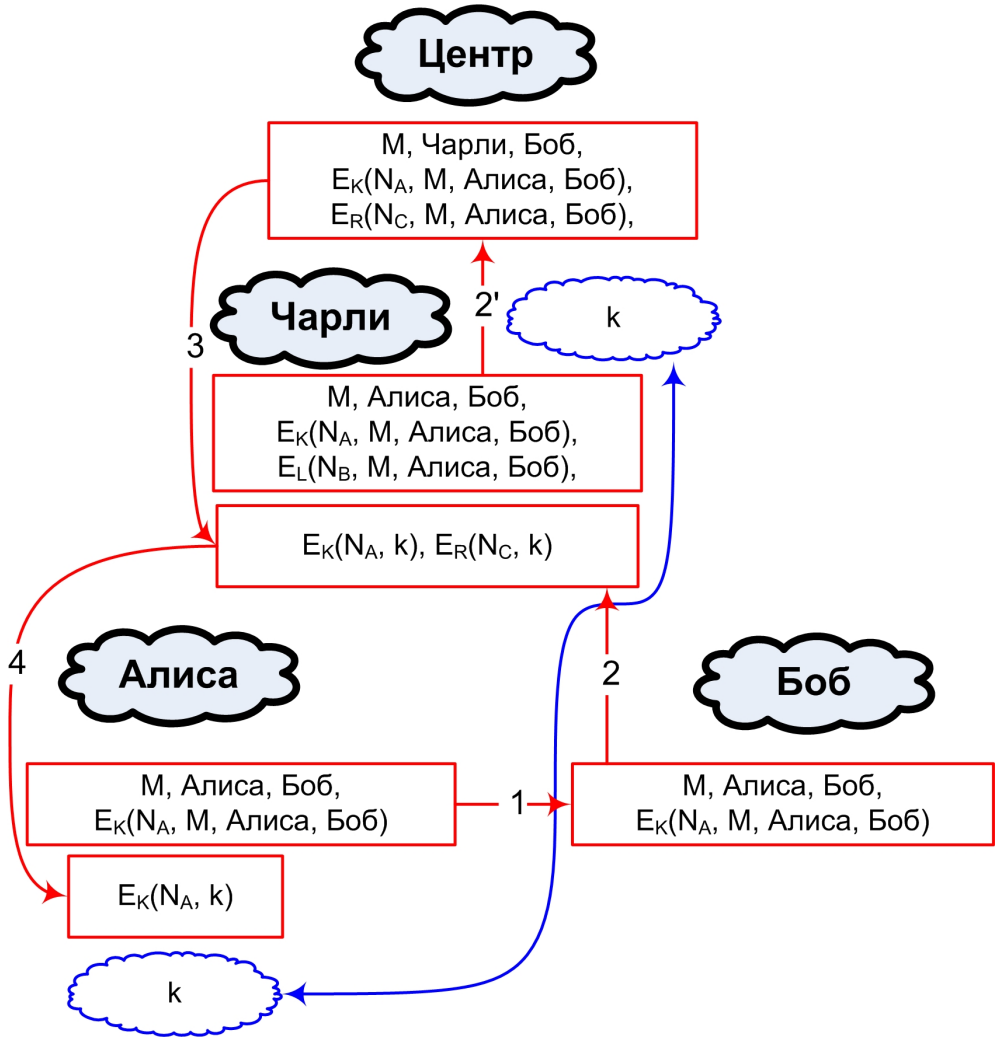
- Вспомним протокол Отвея-Рииса.



Misplaced trust

- Сервер здесь должен проверить, совпадают ли зашифрованные части сообщения 2 и соответствуют ли они тому, что пришло в открытом виде.
- Если эту проверку убрать, то Чарли сможет украсть identity Боба.
- Пусть у Чарли с Центром секретный ключ R.

Misplaced trust



Суть

- Есть Центр, у которого много клиентов. Между Центром и каждым клиентом есть надёжная связь.
- Центр хочет раздать клиентам ключи так, чтобы они могли общаться друг с другом.
- Можно просто раздать каждой паре клиентов по ключу. Но это очень уж много ключей: каждому из n нужно раздать $n-1$ ключ. Нельзя ли меньше?
- Хочется выдать клиенту секретную информацию, по которой он и другой клиент смогут составить себе ключ.
- Но тогда информация будет зависимой, и другие клиенты, может быть, смогут вычислить ключи этой пары...

Надёжность и нижняя оценка

- Система распределения ключей *k-надёжна*, если никакая коалиция из k и менее пользователей не может угадать ключ какой-либо пары пользователей, не входящих в коалицию, лучше, чем алгоритм без знания ключей этой коалиции.
- Нижняя оценка Блома (теоретико-информационная): для каждой k-надёжной системы, раздающей ключи по m бит, у каждого клиента должно быть не меньше $(k+1)m$ бит секретной информации.
- То есть против коалиций размером $n-2$ придётся действительно раздавать $n-1$ ключ каждому. А против меньших можно лучше.

Упражнение. Доказать эту оценку.

Конструкция Блома

- Рассмотрим код, исправляющий ошибки, переводящий сообщение длины k в код длины n, у которого расстояние между кодовыми словами равно в точности $n - k + 1$ (maximum distance separable code).
- В MDS-коде:
  - любые k компонент кодового слова определяют его;
  - любая $k - 1$ компонента кодового слова не даёт о нём никакой информации.

Упражнение. Докажите эти свойства

Конструкция Блома

- Центр выбирает поле $\mathbb{F}_q$, выбирает матрицу генератора G (n, k)-MDS-кода, создаёт случайную симметрическую $k \times k$ матрицу D над $\mathbb{F}_q$ и раздаёт пользователям n строк матрицы $S = (DG)^\top$.
- Теперь два пользователя i и j могут вычислить себе секрет K размером $\log q$:
  - пользователь i вычисляет (i, j)-й элемент матрицы $K = (DG)^T G$;
  - пользователь j вычисляет (j, i)-й элемент матрицы $K = (DG)^T G$;
  - осталось заметить, что K симметрическая.
- Надёжность происходит из того, что матрица K состоит из кодовых слов MDS-кода, а любые k пользователей могут узнать только k её строк (или, что то же самое, столбцов).
- Значит, конструкция j-надёжна для $j \leq k - 1$.

Постановка задачи

- Мотивация: мы тут всё рассуждаем о криптографических секретах (ключах).
- Их надо хранить, и делать бэкап.
- Но чем больше сделать копий, тем хуже будет с надёжностью.
- Поэтому неплохо бы разделить копии так, чтобы по отдельности они ничего не значили, а только вместе.
- Другое применение: распределённый контроль — по отдельности ни один из генералов не может запустить ядерную боеголовку, только все вместе.

Простейшая схема

- Пусть у нас есть n пользователей, один на всех секрет m, и мы хотим раздать его так, чтобы только все пользователи вместе смогли его восстановить.
- Как это сделать?

Простейшая схема

- Пусть у нас есть n пользователей, один на всех секрет m, и мы хотим раздать его так, чтобы только все пользователи вместе смогли его восстановить.
- Как это сделать?
- Очень просто: раздать им случайные числа $r_1, \ldots, r_{n-1}$ и $m \oplus r_1 \oplus \ldots \oplus r_{n-1}$.
- Тогда все вместе смогут восстановить m, а у любой коалиции из $n-1$ участника нет никакой об m информации.
- Но вдруг одного генерала убьют по дороге? Что делать, если нужно, чтобы любые k из n пользователей смогли запустить ракету?

Вопрос

- Чтобы организовать разделение секрета, нужно придумать объект, который k элементами задаётся однозначно, а менее чем k — совсем не задаётся.
- Какие вы знаете такие объекты?

Вопрос

- Чтобы организовать разделение секрета, нужно придумать объект, который k элементами задаётся однозначно, а менее чем k — совсем не задаётся.
- Какие вы знаете такие объекты?
- Многочлен степени $k-1$ задаётся значениями в k точках.
- $(k-1)$-мерная гиперплоскость задаётся k точками.

Схема Шамира

- Секрет — это число a_0.
- Центр выбирает случайные числа $a_1, \ldots, a_{k-1}$ и определяет многочлен

$$f = a_0 + a_1 x + \ldots + a_{k-1} x^{k-1}.$$

- Центр раздаёт участникам числа $f(1), f(2), \ldots, f(n)$ (или значения в любых других точках).
- Любые k участников теперь могут воспользоваться интерполяцией по Лагранжу, а любые $k-1$ не могут.

Схема Блэкли

- Исторически первая и менее удобная.
- Участникам раздаются точки в k-мерном пространстве, лежащие на $(k-1)$-мерной гиперплоскости.
- Каждые k участников могут её восстановить, а $k-1$ никак не может.

Thank you!

Спасибо за внимание!