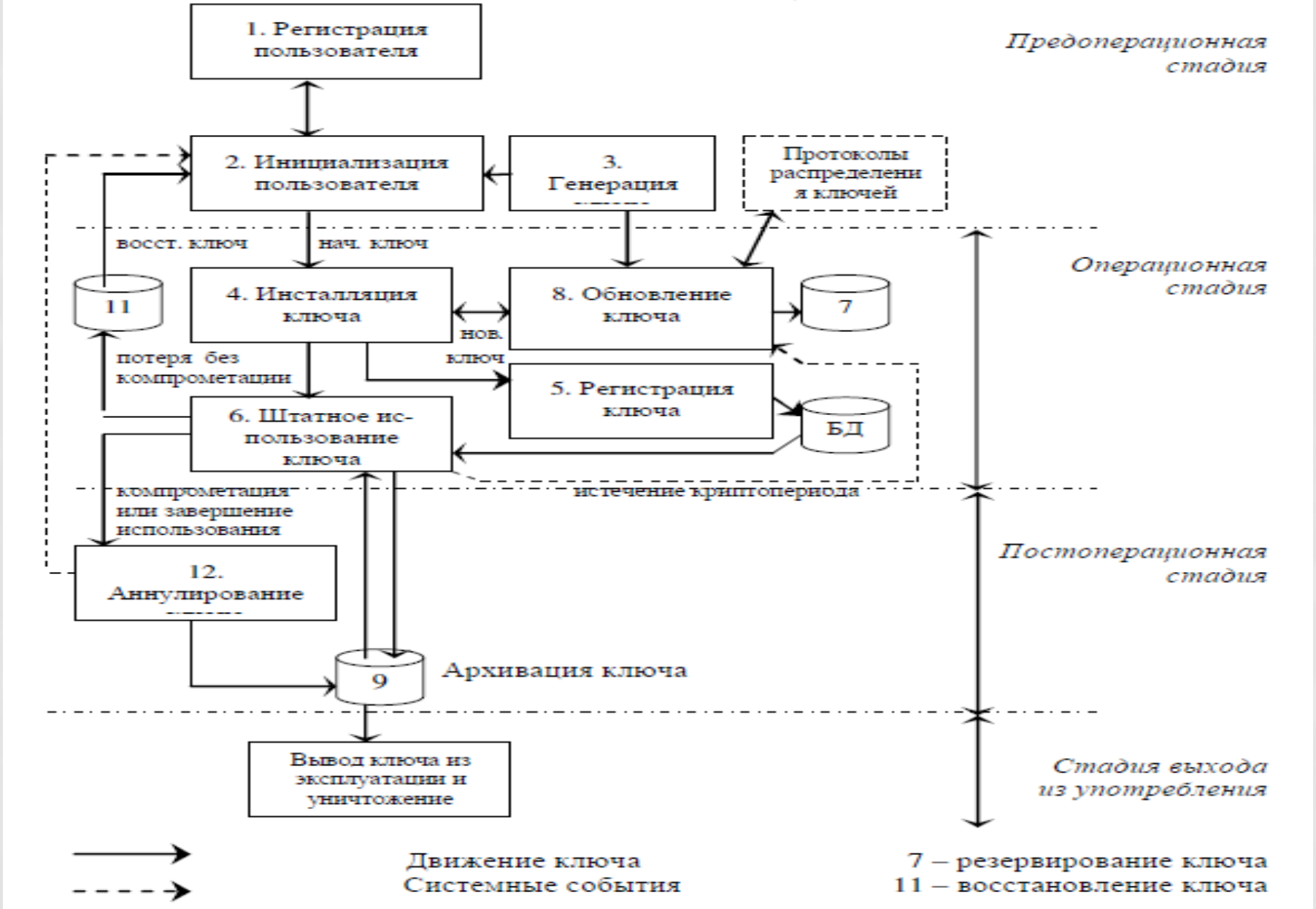
# Управление криптографическими секретными ключами

# Цель управления  ключами

- Определена в международном стандарте ISO/IEC 11770 – Key management
- Управление ключами - совокупность процедур и процессов, сопровождающих жизненный цикл ключей в криптосистеме
- Жизненный цикл – последовательность состояний, в которых пребывает ключевой материал за время своего существования в криптосистеме:  генерация, хранение, распространение, уничтожение и др.
- Цель управления  – обеспечение секретности, подлинности, целостности криптографических ключей на всех этапах жизненного цикла

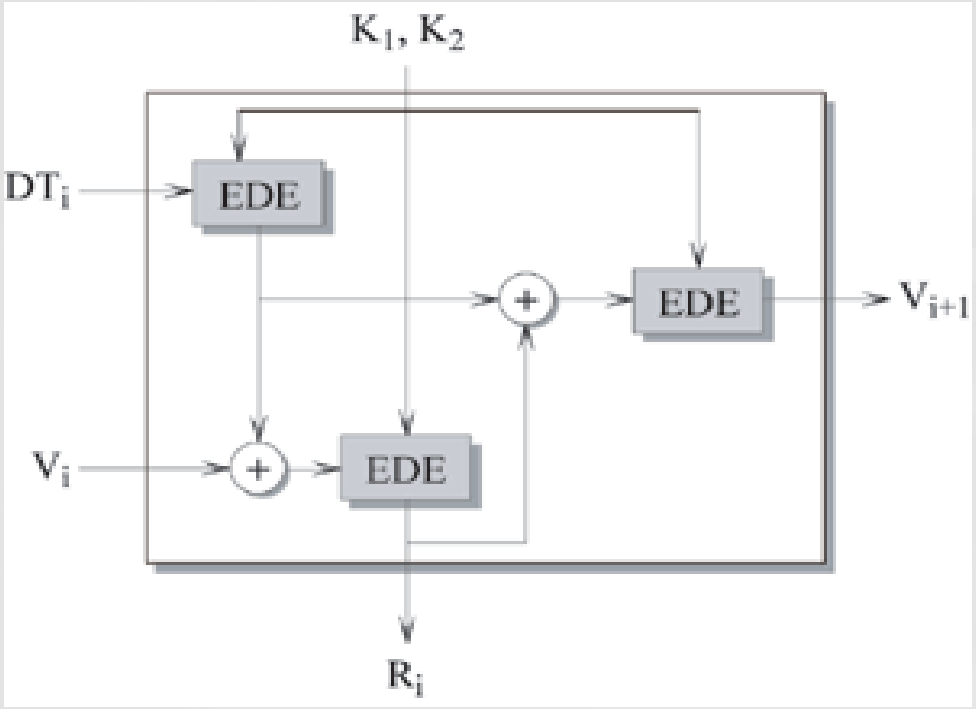
# Схема жизненного цикла ключей



# Задачи предоперационной стадии

- **Регистрация пользователя:** обмен первоначальной ключевой информацией с пользователем, такой, как общие пароли или PIN-коды, путём личного общения или пересылки через доверенного курьера
- **Инициализация:** пользователь устанавливает аппаратное оборудование и/или программные средства в соответствии с установленными рекомендациями и правилами
- **Генерация ключей:** *создание  и обеспечение необходимых криптографических качеств ключей. Ключи могут генерироваться как самостоятельно пользователем, так и специальным защищенным элементом системы, а затем передаваться пользователю по защищенному каналу*

# Генератор ключей стандарта ANSI X9.17

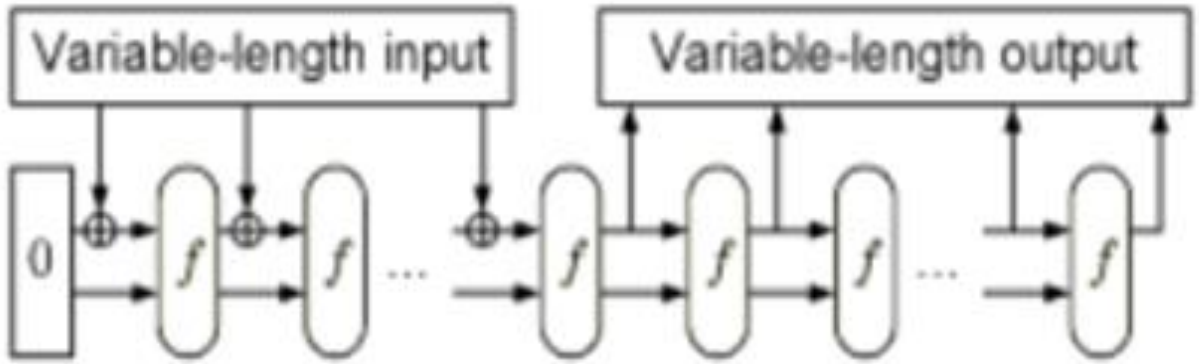



**EDE – Encrypt-Decrypt-Encrypt**

- Один из лучших генераторов. Применяется в приложениях финансовой безопасности и PGP
- DTi - значение даты и времени на начало i-ой стадии генерации
- Vi - начальное значение для i-ой стадии генерации.
- Ri - псевдослучайное число, созданное на i-ой стадии генерации.
- K1, K2 - ключи, используемые на каждой стадии.
- Тогда:
  - $R_i = EDE_{K_1K_2}[EDE_{K_1K_2}[\,DT_i] \oplus V_i\,]$
  - $V_{i+1} = EDE_{K_1K_2}[EDE_{K_1K_2}[\,DT_i] \oplus R_i\,]$

# Генератор на основе хэш-функции Keccak

- Mask generating functions, key derivation




- Использование на стадии «впитывания» входа, на стадии «отжатия» и выхода переменной длины. Может применяться для генерации симметричных ключей из паролей.

# Задачи операционной стадии

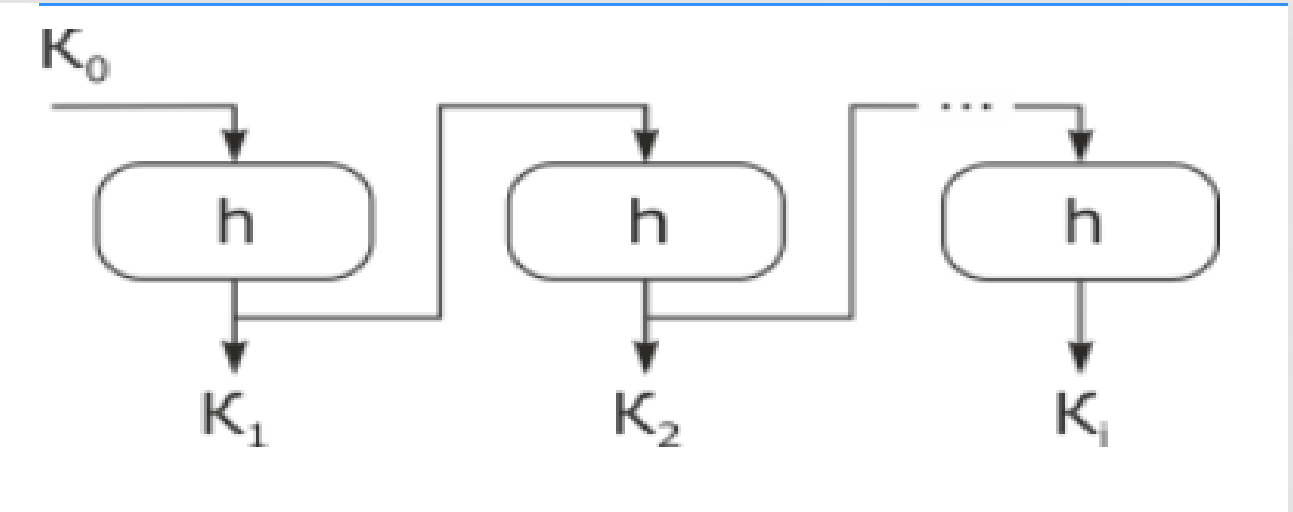
- **Инсталляция:** ключи устанавливаются в оборудование тем или иным способом. При этом первоначальная ключевая информация, полученная на стадии регистрации пользователей, может либо непосредственно вводиться в оборудование, либо использоваться для установления защищенного канала, по которому передается ключевая информация. Эта же стадия используется в последующем для смены ключевой информации
- **Регистрация ключа:** ключевая информация связывается регистрационным центром с именем пользователя и сообщается другим пользователям ключевой сети
- **Штатное использование**: шифрование и расшифрование данных и других ключей
- **Обновление:** *замена ключа осуществляется до истечения его срока действия и включает процедуры, связанные с генерацией ключей, протоколами обмена ключевой информацией между корреспондентами, а также с доверенной третьей стороной*

# Ограничение срока жизни ключа

- Срок жизни ключа (***key lifetime***) — объем данных, который можно "безопасно" обработать на одном ключе, т.е. без возможности скомпрометировать любую конфиденциальную информацию
- Нагрузка на ключ — это объем данных (количество блоков размера n), обработанных на одном ключе
- Практика показывает, что обработка большого количества сообщений на одном ключе и накопление результатов обработки может привести к потере стойкости (к компрометации ключа, дешифрованию сообщений):
  - Методы криптоанализа, основанные на свойствах используемого шифра (например, дифференциальный метод, срабатывают при нагрузке на ключ $2^n$
  - Методы криптоанализа, основанные на комбинаторных свойствах используемого режима работы шифра (например, атаки «парадокса дней рождения», срабатывают при нагрузке $2^{n/2}$
  - Методы, криптоанализа основанные на информации, полученной по побочным каналам ( измерение энергопотребления, электромагнитного излучения, акустического шума, времени работы алгоритма в случае обработке большого количества сообщений позволяет накапливать "опасную" информацию, полученная по побочным каналам)

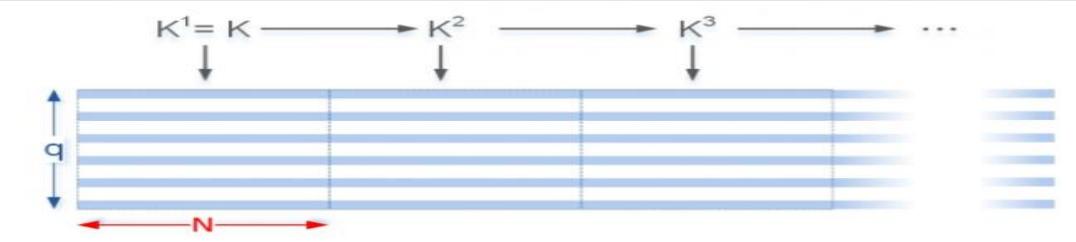
# Преобразования ключа (re-keying)

- Это подход, основная идея которого заключается в зашифровании (расшифровании) данных с помощью последовательности ключей, получаемых из первоначально согласованного ключа (начального) путем применения специально подобранных детерминированных преобразований
- Ресурсоемким примером способа преобразования  ключей является способ получении нового ключа путем хеширования старого.



# Внутреннее преобразование ключа (Internal re-keying)

- Подход заключается в модификации какого-то конкретного режима работы блочного шифра так, чтобы ключ, периодически изменялся по ходу обработки одного файла данных.
- Секция – это последовательность блоков файла, обрабатываемая на одном ключе до его преобразования, при этом такой ключ называется секционным. Размер секции является параметром расширенного режима работы шифра.



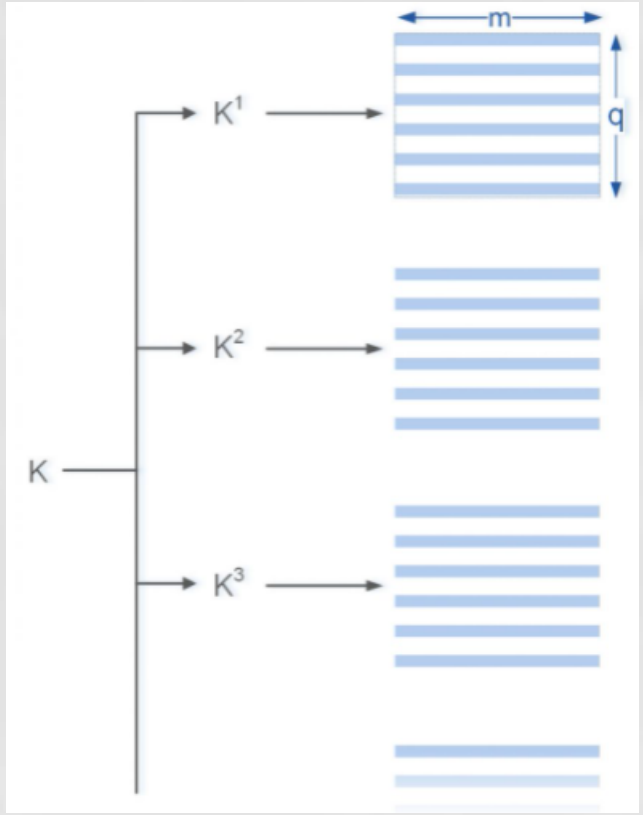

- Пример  способа преобразования ключа ACPKM (Advanced Cryptographic Prolongation of Key Material), который применяется к режиму шифрования CTR шифра «Кузнечек»:

$$K^{i+1} = E_{K^i}(W_1) \mid E_{K^i}(W_2),$$

где $W_1$ и $W_2$ – некоторые константные строки, а операция "|" – конкатенация.

# Внешнее преобразование ключа (external re-keying )

- Отличием этого подхода является то, что ключ меняется не в процессе обработки одного сообщения, а после обработки некоторого количества целых сообщений
- Применение данного подхода не влияет на внутреннее строение режима и не меняет порядка обработки отдельных сообщений
- Пример способа получения последовательности ключей для любого режима шифра «Кузнечек»:




$$K^1 \mid K^2 \ldots \mid K^t = E_K([0]) \mid E_K([1]) \mid \ldots \mid E_K([2t\text{-}1]),$$

где $[i]$ — строка длины 128 бит, которая является двоичным представлением числа $i$.

# Задачи операционной стадии (продолжение)

- **Резервирование:** создание копии ключевой информации для восстановления ключа на случай обстоятельств, не связанных с компрометацией
- **Восстановление**: восстановление ключа из хранимой копии, в случае, если ключевая информация была уничтожена, но не скомпрометирована (например, из-за неисправности оборудования или из-за того, что оператор забыл пароль)
- **Хранение ключа***: включает процедуры, необходимые для хранения ключа в надлежащих условиях, обеспечивающих его безопасность до момента его замены*

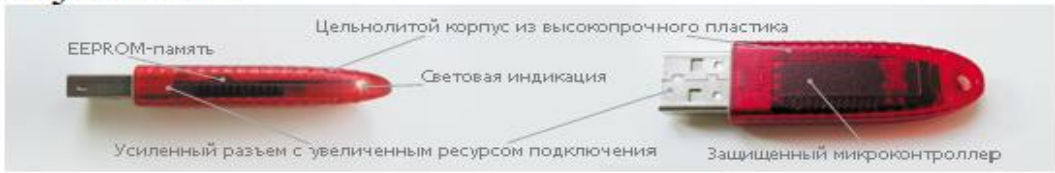
# Аппаратные средства хранения ключей

**Смарт-карта — устройство для одно- и двухфакторной аутентификации пользователей, хранения ключевой информации и проведения криптографических операций в доверенной среде.**


Разница между использованием криптографических токенов или смарт-карт и стандартных флэш-накопителей в том, что при использовании криптографического оборудованием ключ генерируется на самом оборудовании и никогда не экспортируется !

**Электронный идентификатор (токен) - компактное устройство в виде USB-брелока, которое служит для авторизации пользователя в сети или на локальном компьютере, защиты электронной переписки, безопасного удаленного доступа к информационным ресурсам, а также надежного хранения персональных данных.**

***«Рутокен»:***




***«eToken»:***

# Задачи постоперационной стадии

- **Архивирование: в** отдельных случаях ключевая информация после её использования для защиты информации может быть подвергнута архивированию для её извлечения со специальными целями (например, расшифровки материалов с грифом ДСП)
- **Аннулирование:** в случае компрометации ключевой информации возникает необходимость прекращения использования ключей до окончания срока их действия. При этом должны быть предусмотрены необходимые меры оповещения абонентов сети.
- **Вывод из эксплуатации:** после окончания сроков действия ключей они выводятся из обращения, и все имеющиеся их копии уничтожаются. При этом необходимо следить, чтобы тщательно уничтожалась и вся информация, по которой возможно их частичное восстановление.

# Классификация ключей по функциональному назначению(ANSI X9.17)

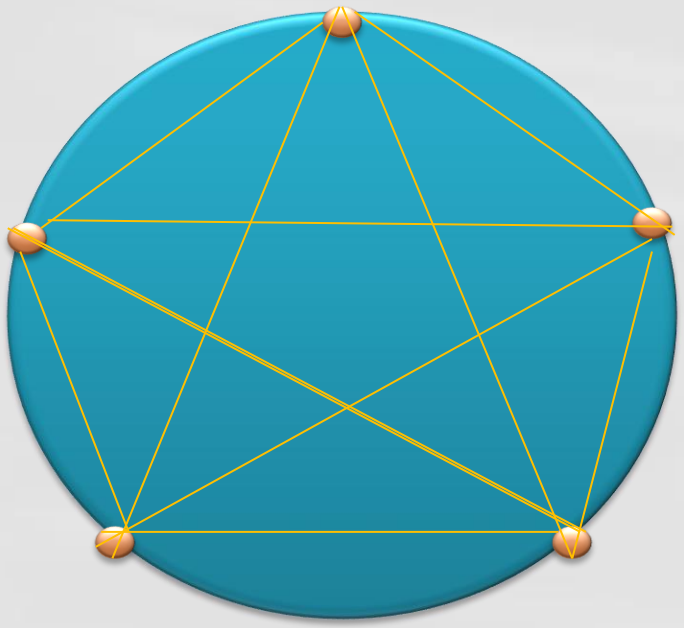
- **Мастер-ключи** – ключи высшего уровня иерархии, которые сами защищаются не криптографическими средствами: физическими, аппаратными, организационными, юридическими и пр. Распределяются вручную или инсталлируются в систему на предоперационной стадии, остаются неизменными в течение всего жизненного цикла. МК защищаются мерами процедурного контроля, физической или логической изоляцией их от посторонних субъектов
- **Ключи шифрования ключей (КШК)** – симметричные ключи, используемые для транспортировки или хранения других ключей, например, в протоколах транспортировки ключей. Безопасность КШК обеспечивается уровнем мастер-ключа
- **Ключи шифрования данных** – ключи, которые используются для выполнения криптографических операций над данными пользователя (шифрование, аутентификация). Обычно это кратковременные (сеансовые) ключи

# Классификация ключей по длительности использования

- **Криптопериод ключа** – это период времени, в течение которого ключ остается действительным для санкционированного использования в системе. Введение криптопериода ключей может служить следующим целям:
  - ограничению информации, связанной с определенным ключом, доступной для криптоанализа противником;
  - ограничению ущерба – количества раскрытой информации в случае компрометации ключа;
  - ограничению времени, доступного для криптоаналитика противника с мощными вычислительными ресурсами
- **Долговременные ключи (long-term keys)** – как правило, мастер-ключи, а часто также КШК и другие ключи, способствующие обмену ключами, защищают кратковременные ключи
- **Кратковременные ключи (short-term keys)** – ключи, выработанные посредством протоколов обмена ключами или транспортировки, которые используются как КШД или сеансовые ключи для одного сеанса связи между абонентами криптосистемы

# Распределение симметричных ключей

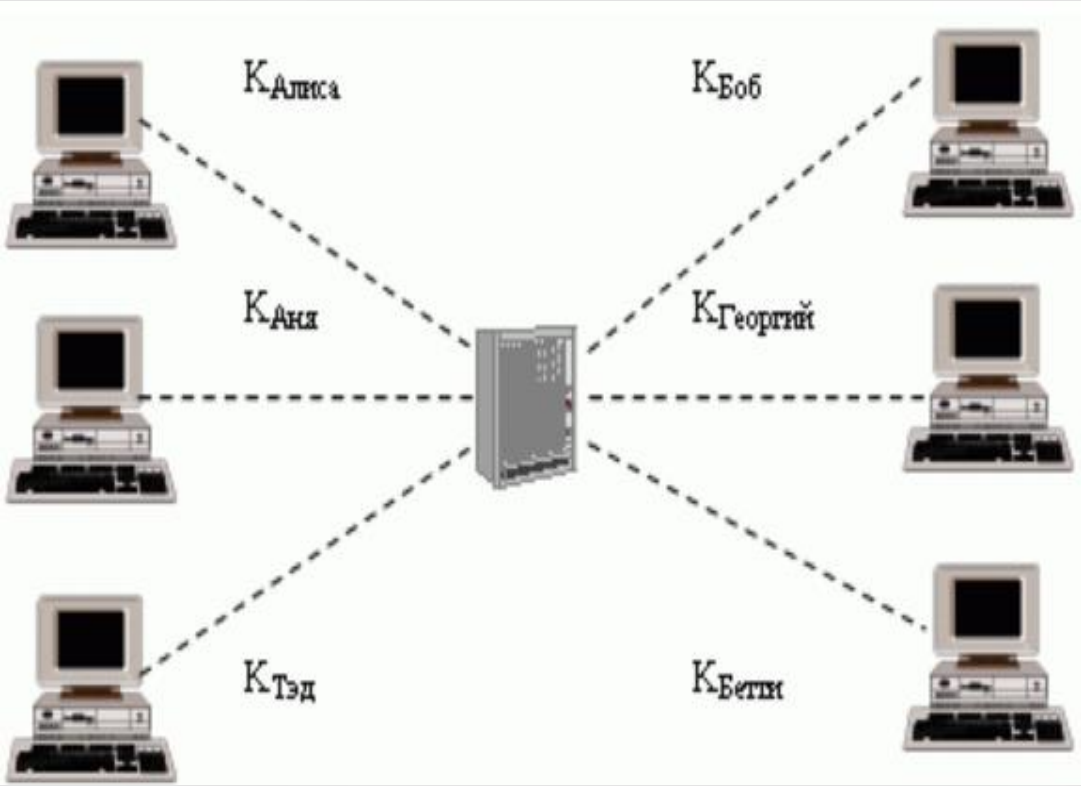
# Проблемы распределения симметричных ключей



$$R \sim \frac{N*(N-1)}{2}$$

- Необходим надежный способ первоначального распределения ключей (обмен ключами при личной встрече, доставка спецкурьером, передача частями по разным каналам, по протоколу с центром распределения ключей)
- Ключи должны время от времени меняться для снижения вероятности их компрометации Оптимальным считается использование для каждого сеанса обмена зашифрованными сообщениями своего уникального ключа (*session key)*
- При большом числе взаимодействующих сторон *N* требуется значительные ресурсы *R* для рассылки, хранения и смене ключей

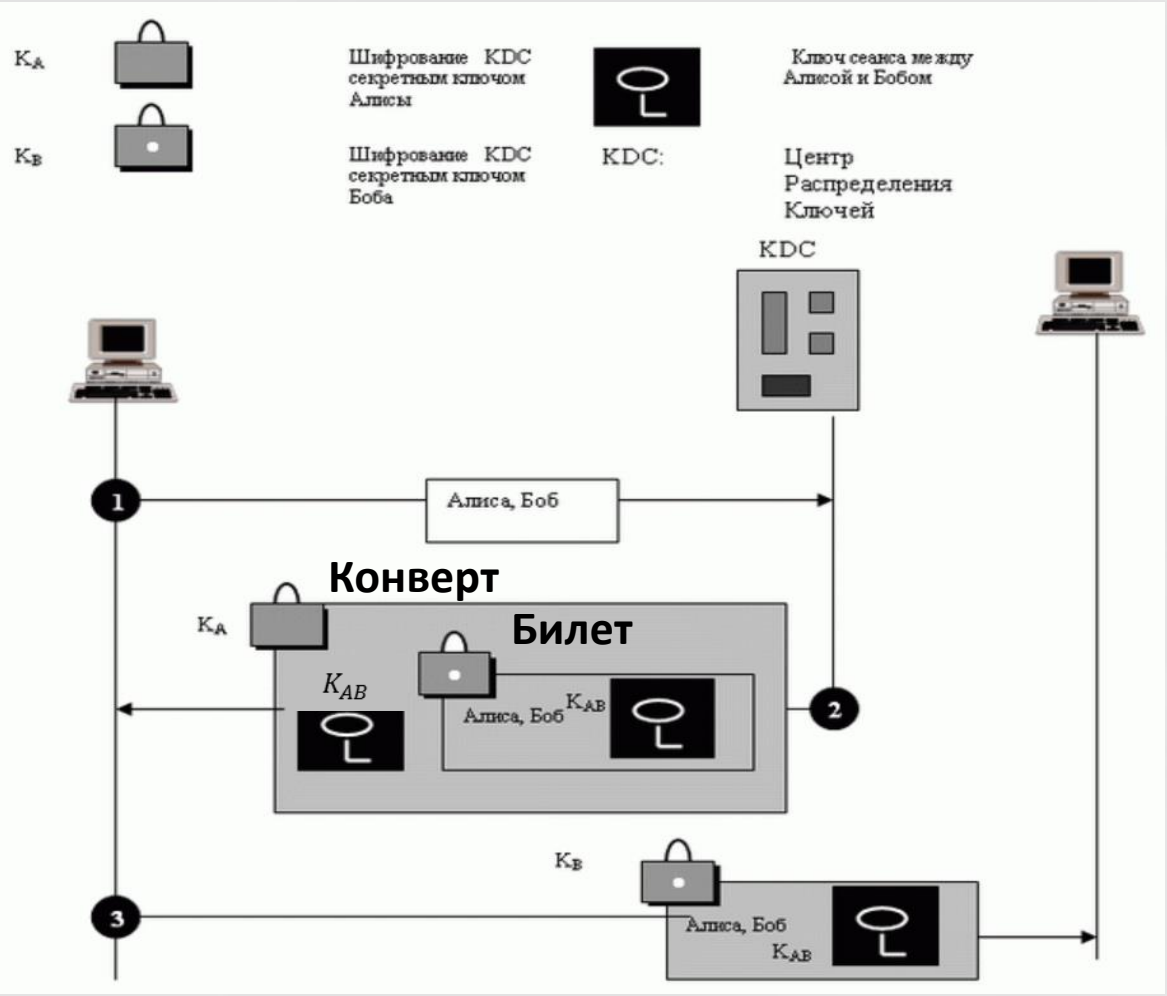
# Центр Распределения Ключей: KDC




- KDC-  это фактически привлечение третьего лица, которому доверяют
- Каждый абонент устанавливает ключ засекречивания с KDC
- Ключи засекречивания  абонентов используется, чтобы подтвердить их подлинность

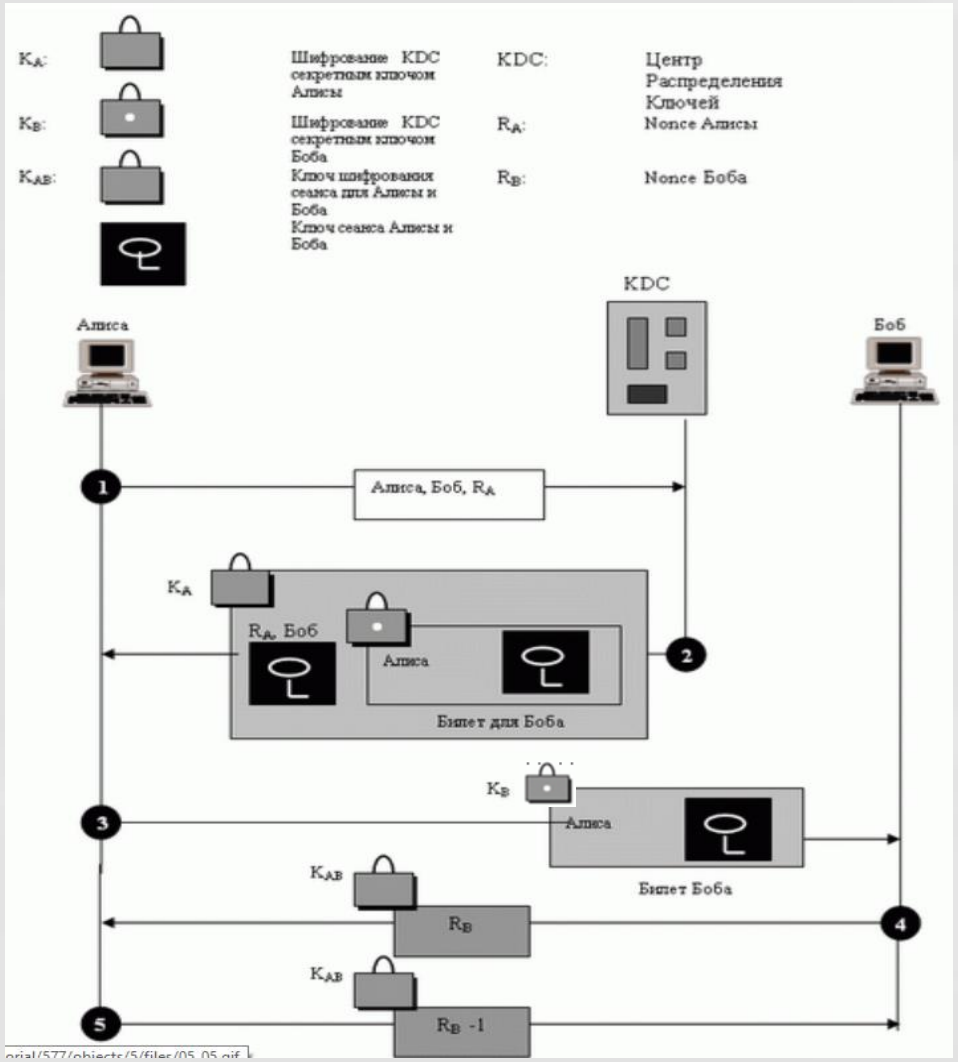
***KDC - Key-Distribution Center***

# Простой протокол получения сеансового ключа



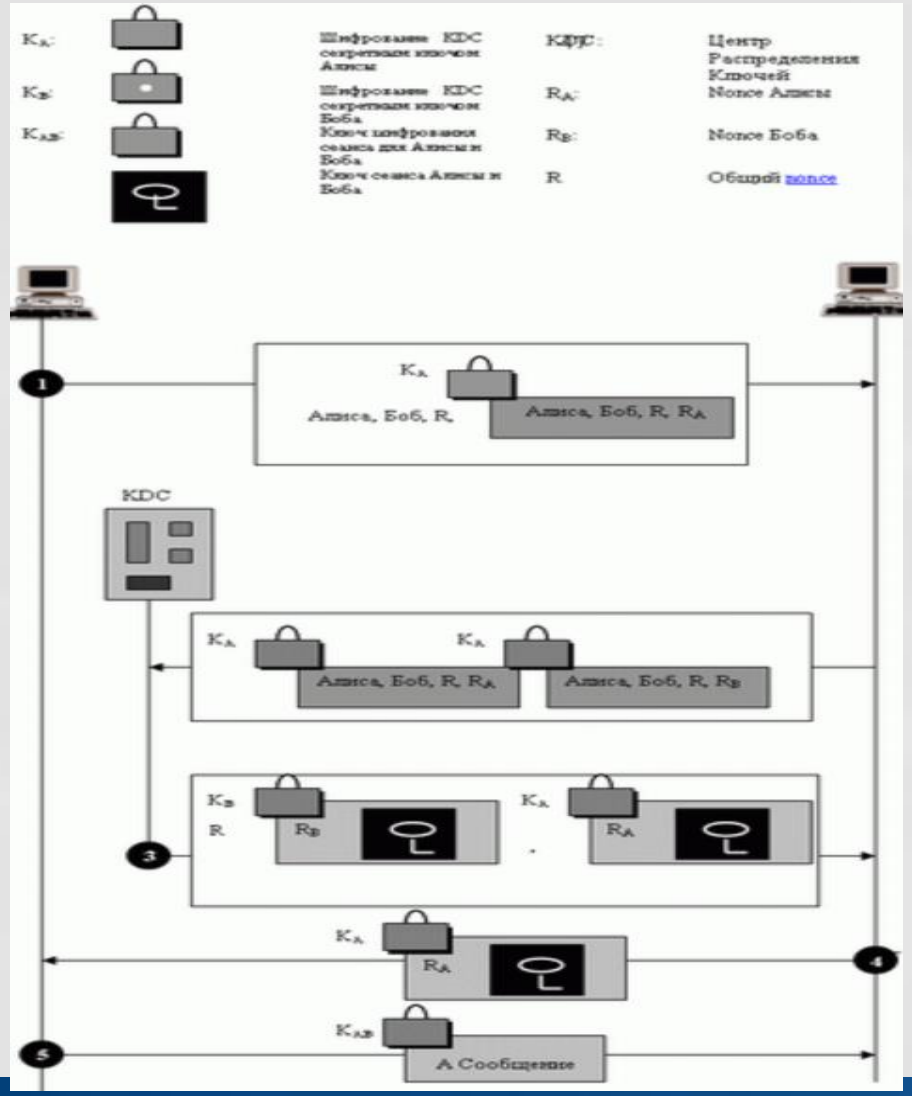

- (1) Открытый запрос  в KDC на установление канала связи
- (2) KDC создает билет (на ключе получателя) и конверт (на ключе отправителя)
- (3) Билет с сеансовым ключом отправитель пересылает получателю
- (!) Возможна атака ответа: можно сохранить сообщение шага 3 и использовать его позже

# Протокол Ниидома-Шрёдера (Needham-Schreder)




- (1) Отправитель передает сообщение KDC, в которое включает свой *nonce* $R_A$, свой опознавательный код и опознавательный код получателя
- (2) KDC передает зашифрованное сообщение отправителю, которое включает *nonce* $R_A$ , опознавательный код получателя, ключ сеанса и зашифрованный билет для получателя.
- (3) Отправитель передает билет получателя по адресу
- (4) Получатель передает свой *nonce* $R_B$ отправителю, зашифрованный ключом сеанса $K_{AB}$
- (5) Отправитель отвечает на запрос получателя, передавая *nonce* $R_B$-1, зашифрованное сеансовым ключом $K_{AB}$

# Протокол Отвея-Рисса (Otway-Rees)

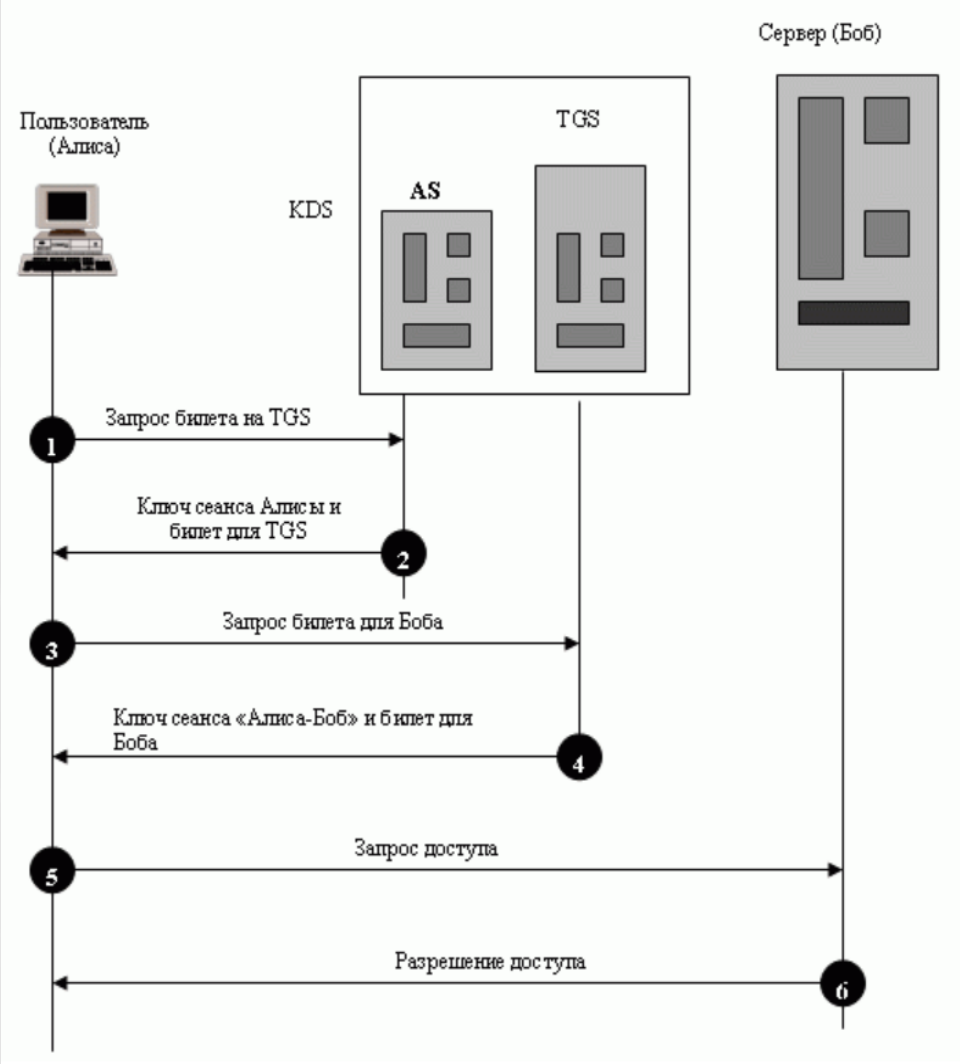



- (1) Отправитель передает сообщение получателю, включающее *nonce R*, идентификаторы отправителя и получателя и билет для KDC - в билет входят nonce $R_A$ , копия общего *nonce R* и идентификаторы отправителя и получателя
- (2) Получатель создает подобный билет, но с собственным *nonce* $R_B$. Оба билета передают KDC
- (3) KDC создает и передает получателю сообщение, которое содержит общий *nonce R*, билет для отправителя и билет для получателя. Билеты содержат ключ сеанса $K_{AB}$ и соответствующие *nonce* $R_A$ и $R_B$
- (4) Получатель пересылает билет отправителю
- (5) Отправитель подтверждает получение сообщением зашифрованным на ключе сеанса

# Протокол «Цербер» (Kerberos )

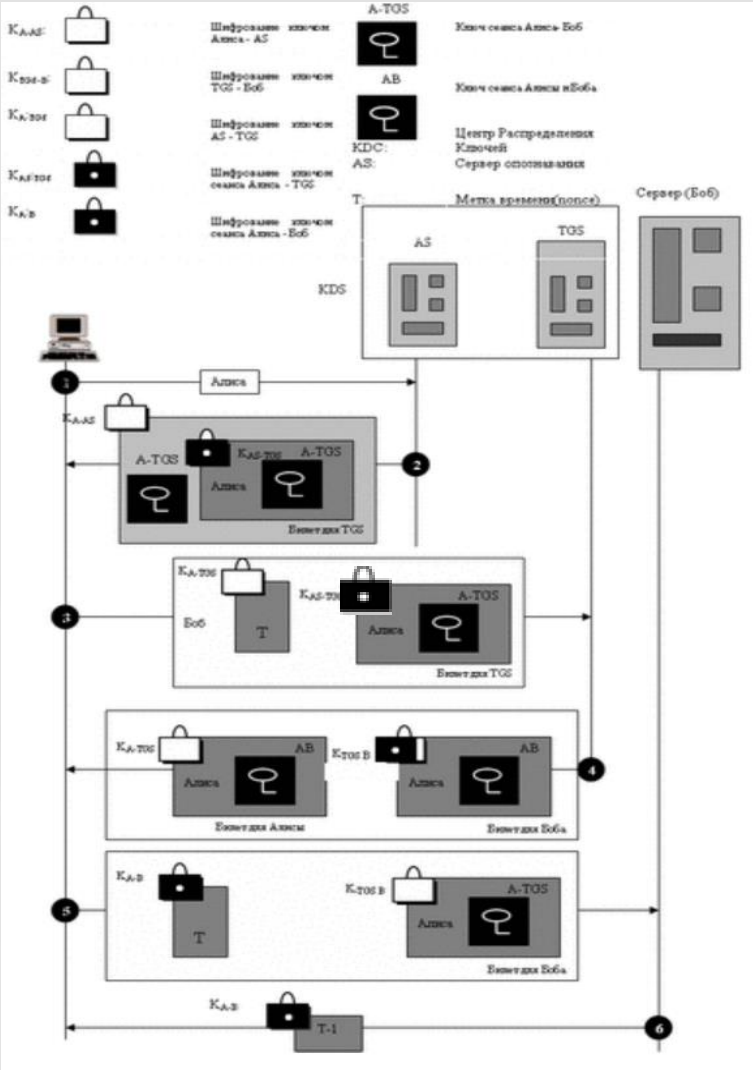
- Протокол проверки подлинности сторон ( симметричной аутентификации), обеспечивающий безопасную передачу данных в незащищенных сетях
- Протокол разработан для использования в системах с «клиент-серверной» архитектурой ( например, протокол передачи файлов FTP
- Протокол поддерживают, например, FreeBSD, Mac OS X, Red Hat Linux и прочие UNIX-подобные операционные системы

# Протокол «Цербер»: взаимодействие серверов



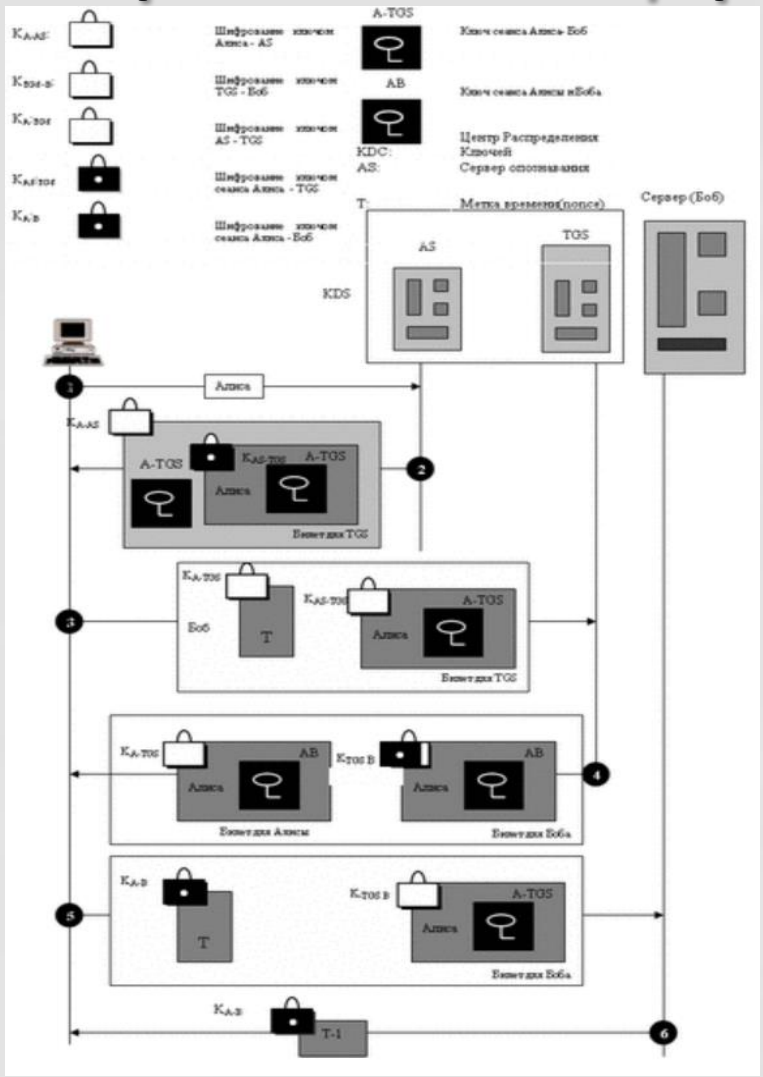

- Опознавательный сервер (Authentication Server). Каждому пользователю, зарегистрированному в AS, предоставляют пользовательский идентификационный код и пароль.. AS верифицирует пользователя, выдает ключ сеанса, который используется между клиентом и TGS, и передает билет для TGS
- Предоставляющий билет сервер (Ticket-Granting Server) вырабатывает билет для сервера услуг и обеспечивает ключ сеанса ( $K_{AB}$ ) между клиентом и сервером услуг.
- Сервер услуг предоставляет сервисы для клиента.

# Протокол «Цербер»: взаимодействие абонентов




- (1) Алиса передает свой запрос AS в открытом тексте, используя свой зарегистрированный код идентификации
- (2) AS передает сообщение, зашифрованное постоянным симметричным ключом клиента $K_{A-AS}$. Это сообщение содержит два объекта: ключ сеанса, $K_{A-TGS}$, который используется клиентом, чтобы войти в контакт с TGS, и билет для TGS, который зашифрован TGS-симметричным ключом $K_{AS-TSG}$ . Клиент не знает $K_{A-AS}$, но когда сообщение прибывает, он сообщает свой пароль, который служит для создания $K_{A-AS}$ и после уничтожается. Процесс использует $K_{A-AS}$ для того, чтобы расшифровывать переданное сообщение с $K_{AS-TSG}$ и билетом для TGS .
- (3) Клиент передает три объекта в TGS: билет, полученный от AS, имя сервера услуг, метку времени, которая зашифрована ключом $K_{A-TGS}$. Метка времени предотвращает ложный ответ нарушителя.

# Протокол «Цербер»: взаимодействие абонентов




- (4) Теперь TGS передает клиенту два билета: каждый содержит ключ сеанса $K_{A-B}$ между клиентом и сервером услуг , Билет для клиента - зашифрованный $K_{A-TGS}$ , билет для сервера - зашифрованный с ключом $K_{B-TGS}$.
- (5) Клиент передает билет серверу услуг и метку времени, зашифрованной ключом $K_{A-B}$.
- (6) Сервер услуг подтверждает, что получил эту информацию, прибавляя 1 к метке времени. Сообщение шифруется ключом $K_{A-B}$ и передается клиенту.

# Асимметричная криптография

Введение

# Концептуальные отличия асимметричной криптографии

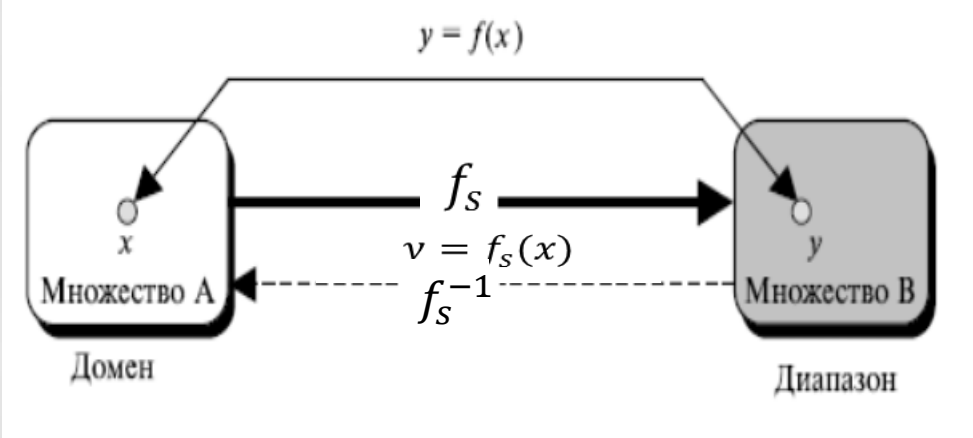
- Криптография с симметричными ключами базируется на совместном использовании секретного ключа, в то время как асимметричная криптография базируется на персональном ключе (закрытом)
- В криптографии с симметричными ключами, биты переставляются или заменяются другими; в асимметричной криптографии числа, представляющие открытые тексты, преобразуются с помощью математических функций

# Историческая справка

- Первой открытой публикацией в области асимметричной криптографии принято считать статью  Уитфилда Диффи (Whitfield Diffie) и Мартина Хеллмана (Martin Heilman) «Новые направления в криптографии», опубликованную в 1976 г.
- В «новой криптографии» введено понятие  односторонней функции с секретом
- Предложен  алгоритм, позволяющий паре пользователей выработать общий секретный ключ, не обмениваясь секретными данными по небезопасному каналу связи

# Односторонняя функция с секретом (люком)
## (TOWF — Trapdoor One Way Function)

- Зная x, при любом s легко вычислить y=$f_s$(x)
- По известному значению y и s легко вычислить x=$f_s^{-1}(y)$
- Сложно вычислить x=$f_s^{-1}(y)$ по известному y, если секрет s не известен

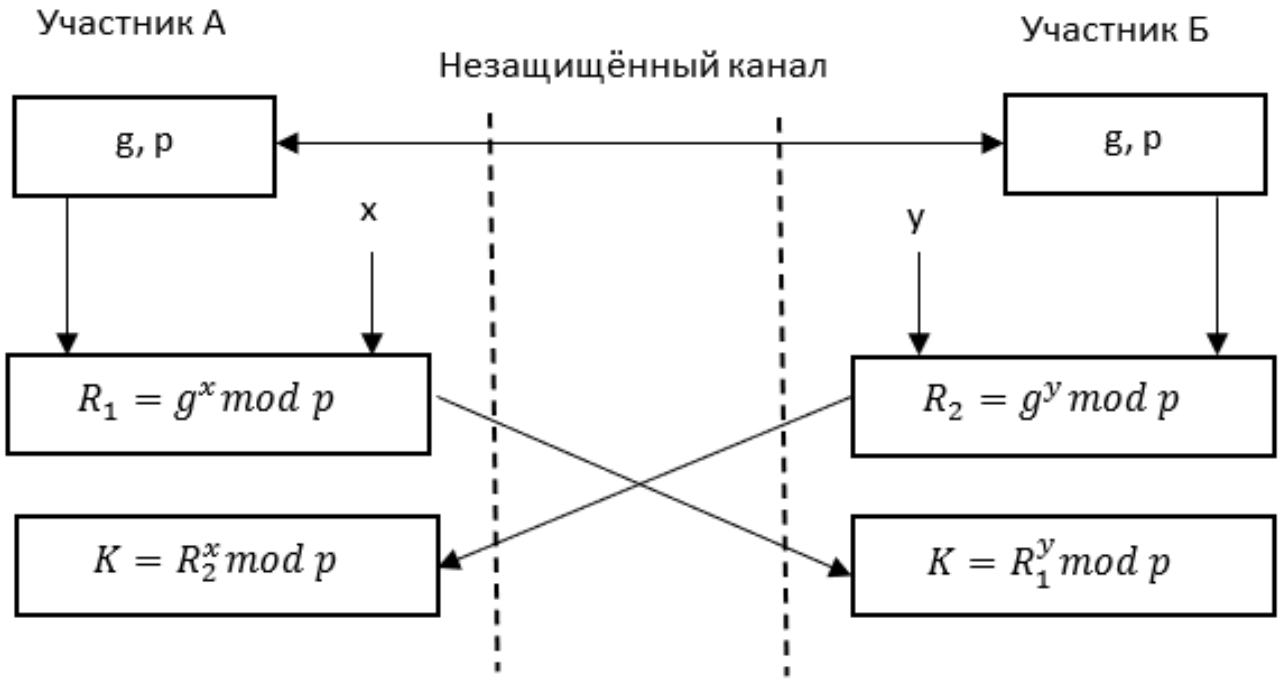



- Предположение о высокой сложности операции  обратимости односторонних функций положено в основу криптографии в условиях зарождающейся  тогда компьютерной эпохи

# Значимость TOWF

- Отказ от секретных каналов связи для предварительного обмена ключами;
- Включение в задачу вскрытия шифра трудную математическую задачу для повышения обоснованности стойкости шифра
- Решение новых криптографических задач, отличных от шифрования (электронная цифровая подпись и др.).

# Протокол Диффи-Хеллмана (Diffie-Hellman, DH)




$$K = {R_2}^x mod\ p = g^{yx} mod\ p = {R_1}^y mod\ p$$

- $(p,g,R_1)$ и $(p,g,R_2)$ - открытые ключи сторон
- $x$, $y$ - закрытые ключи сторон
- $R_2^x mod\ p$ и $R_1^y mod\ p$ - односторонние функции с секретом (TOWF)

# Математическая модель протокола

- *p* -большое простое число порядка 300 десятичных цифр (1024 бита)
- *g* – порождающий элемент циклической группы (генератор) порядка *p,* для которого справедливо:
  $g \bmod p, g^2 \bmod p, g^3 \bmod p \ldots g^{p-1} \bmod p$ являются различными целыми из *[1,p-1]*
- *x, y* - большие случайные числа такие, что *0 < x < p – 1, 0 < y < p – 1*
- Поскольку:

$$R_2{}^x \bmod p = (g^y \bmod p)^x \bmod p = g^{xy} \bmod p$$
$$R_1{}^y \bmod p = (g^x \bmod p)^y \bmod p = g^{xy} \bmod p$$

- Стороны фактически создают симметричный ключ сеанса без Центра распределения ключей (KDC)

# Атака дискретного логарифма

- Так как *x* и *y* являются закрытыми данными, противник может получить только следующие значения *g, p,* $R_1$ , $R_2$
- Для вычисления ключа атакующий должен решить две задачи дискретного логарифмирования: найти целые *x* и *y* из уравнений $R_1 = g^x mod\ p; R_2 = g^y mod\ p$
- Задача вычисления дискретные логарифмов становится трудноразрешимой, если:
  - Простое число *p* должно быть очень большим (более чем 300 десятичных цифр).
  - Простое число *p* должно быть выбрано так, чтобы *p - 1* имел по крайней мере один простой делитель (больше чем 60 десятичных цифр).
  - Генератор *g* должен быть первообразным  корнем по модулю *p*
  - Значения *x* и *y* должны использоваться только единожды

# Пример

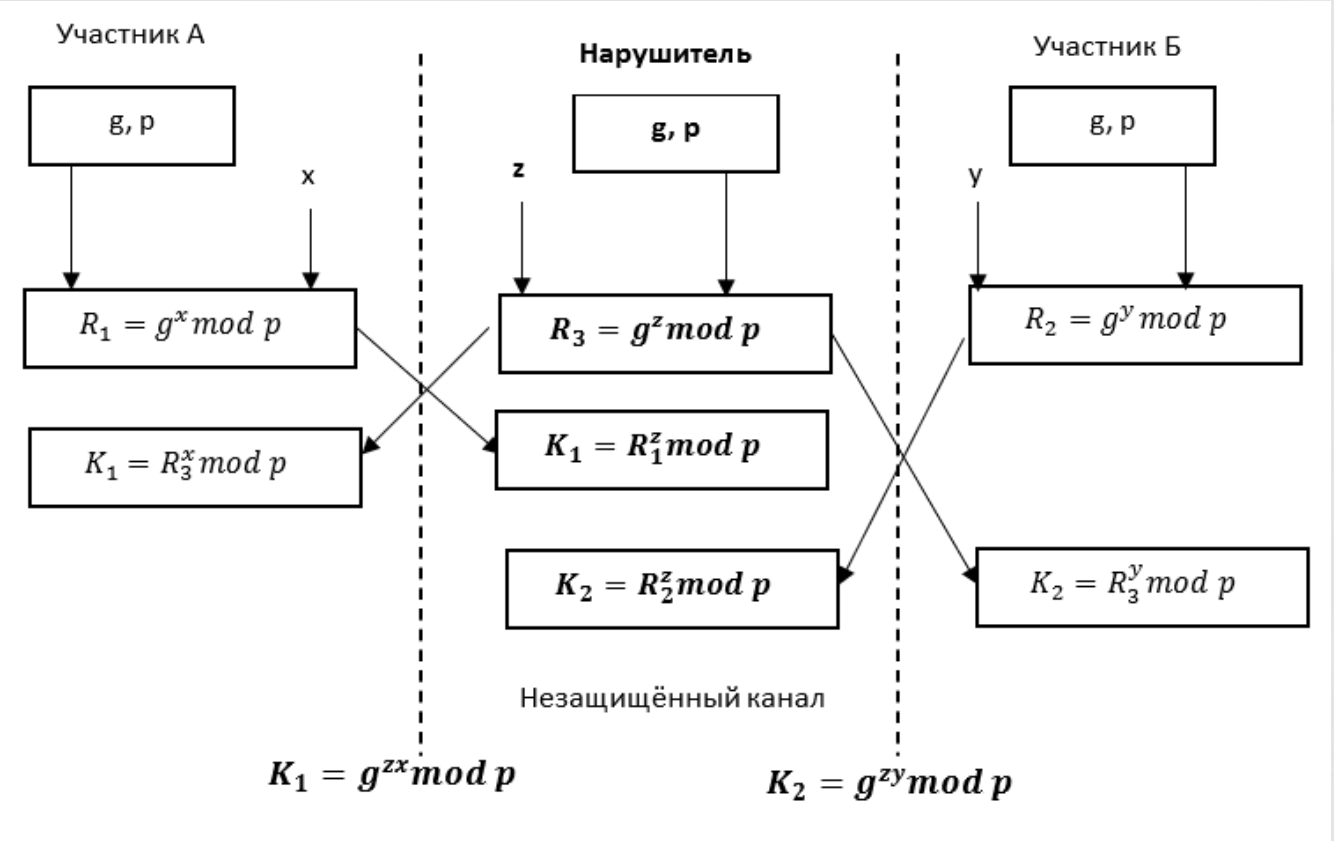
- Выбираем:

| | |
|---|---|
| p | 764624298563493572182493765955030507476338096726949748923573772860925<br>2356666607554236374233096611800333381061947301309504147387009991780436548785807987581 |
| g | 2 |
| x | 557 |
| y | 273 |

- Вычисляем:

| | |
|---|---|
| $R_1$ | 84492028420 665505216172947491035094143433698520012660862863631067673<br>619959280828586700802131859290945140217500319973312945836083821943065 966020157955354 |
| $R_2$ | 435262838709200379470747114895581627636389116262115557975123379218566<br>3100114357182083900401818764868417538311653426916302634211067215085896255201288594143 |
| K | 155638000664522290596225827523270765273218046944423678520320400146406<br>5008879366512042574267766083279110171530386745612522131516109765842001204086433617740 |

# Атака посредника на DH (Man in the Middle)




- Предполагается, что нарушитель может осуществить активную атаку, т.е. имеет возможность не только перехватывать сообщения, но и заменять их другими
- Нарушитель может перехватить открытые ключи участников $R_1$ и $R_2$ и создать свою пару открытого и закрытого числа $(R_3, \mathrm{z})$, чтобы послать их каждой стороне
- После этого каждый участник вычислит ключ, который будет общим с нарушителем, а не с другой стороной
- Если нет контроля подлинности сторон, то законные участники протокола не смогут обнаружить подобную подмену

# Обучающий ролик по DH-протоколу

https://www.youtube.com/watch?v=vFjq9pID4-E