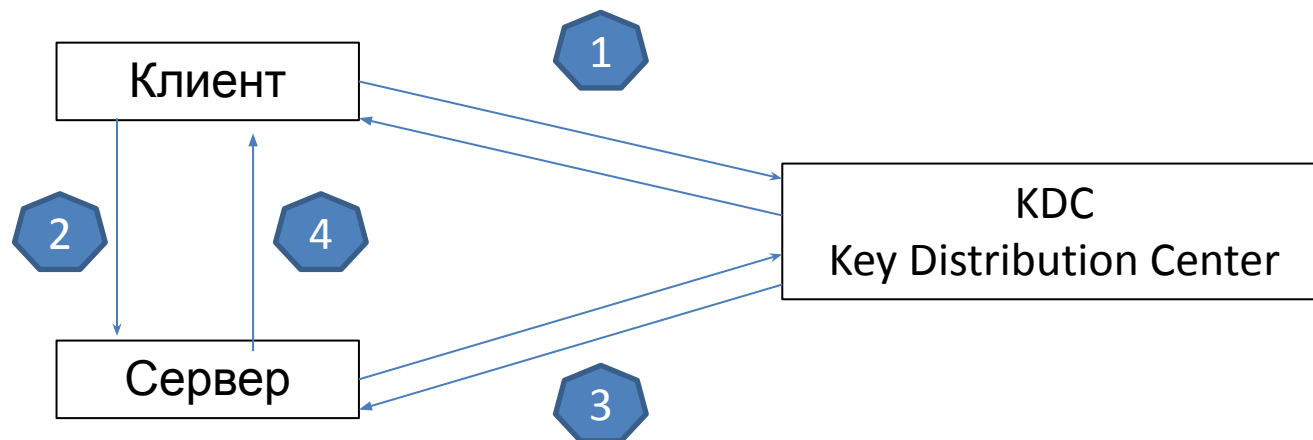


# Kerberos

сетевой протокол аутентификации,  
позволяющий передавать данные  
через незащищённые сети для  
безопасной идентификации

# Протокол аутентификации Нидхема – Шрёдера



MIT – Massachusetts Institute of Technology

`security/krb5`

UNIX(R) Heimdal Kerberos (`security/heimdal`) FreeBSD

# Ticket:

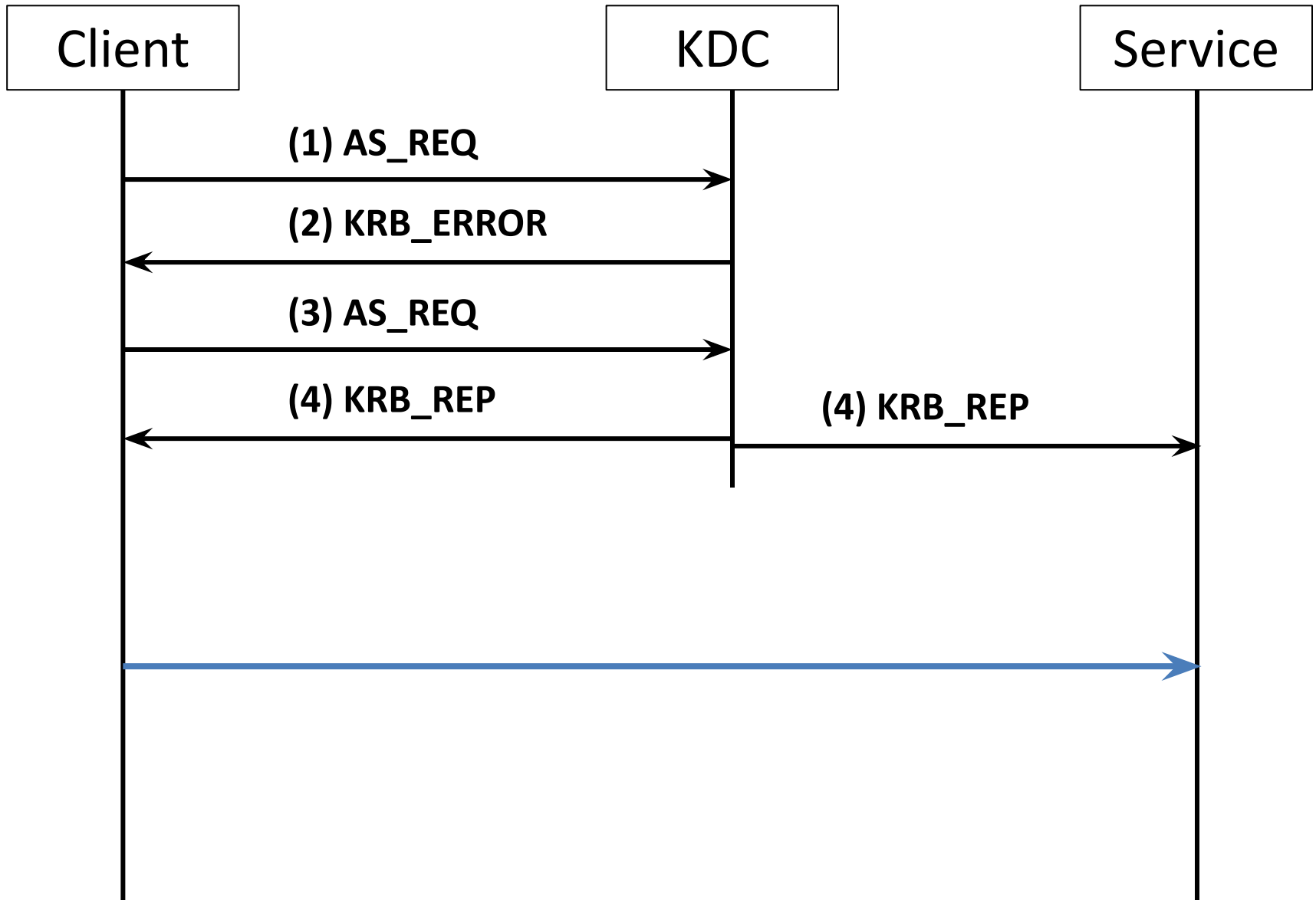
- TGT – Ticket Granting Ticket
- TGS – Service Ticket

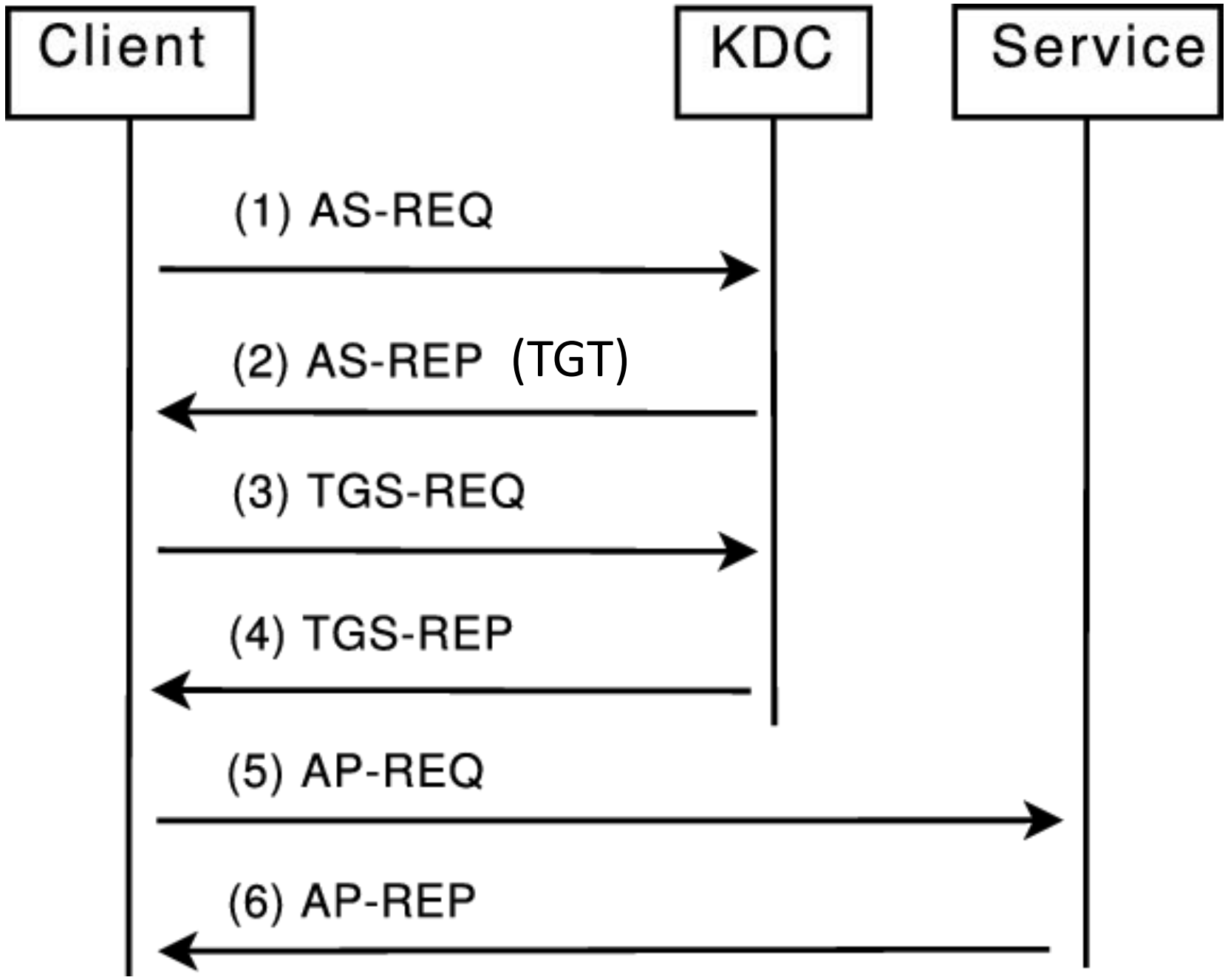
## Kerberos Ticket

<b>Kerberos Version (5)</b>		
<b>Server Realm</b>		<b>Server Name</b>
<b>flags</b>	<b>Session Key</b>	
<b>Client Realm</b>		<b>Client Name</b>
<b>Validity Start Time</b>		<b>Validity End Time</b>

↑ Encrypted with Service Key ↓

- **Realm = administrative domain**
- **Kerberos Tickets and all messages are ASN.1 encoded**
- **Tickets contain some additional, optional fields**





## Шифрование открытым ключом

- У. Диффи, М. Хеллман «Новые направления в современной криптографии» 1976 г.
- RSA (Rivest, Shamir и Adleman).

$x \rightarrow f(x)$

$y = f(x) \rightarrow ? x.$

**SIN**

1. Преобразование исходного текста должно быть условно необратимым и исключать его восстановление на основе открытого ключа.
2. Определение закрытого ключа на основе открытого также должно быть невозможным на современном технологическом уровне.

# Типы односторонних преобразований.

1. Разложение больших чисел на простые множители (алгоритм RSA).
2. Вычисление дискретного логарифма или дискретное возведение в степень (алгоритм Диффи-Хелмана-Меркла, схема Эль-Гамала).
3. Задача об укладке рюкзака (ранца) (авторы Хелман и Меркл).
4. Вычисление корней алгебраических уравнений.
5. Использование конечных автоматов (автор Тао Ренжи).
6. Использование кодовых конструкций.
7. Использование свойств эллиптических кривых.

# RSA

№ п/п	Описание операции	Пример
1	Выбираются два простых числа <b>p</b> и <b>q</b> .	$p=7, q=13$
2	Вычисляется произведение $n = p * q$ .	$n=91$
3	Вычисляется <b>функция Эйлера</b> $\varphi(n)$ .	$\varphi(n) = (7-1)(13-1) = 91-7-13+1 = 72$
4	Выбирается открытый ключ <b>e</b> - произвольное число ( $0 < e < n$ ), взаимно простое с результатом функции Эйлера ( $e \perp \varphi(n)$ ).	<b><math>e=5</math></b>
5	Вычисляется закрытый ключ <b>d</b> - обратное число к <b>e</b> по модулю $\varphi(n)$ из соотношения $(d * e) \bmod \varphi(n) = 1$ .	$(d*5) \bmod 72 = 1, \mathbf{d = 29}$



Процедуры шифрования и дешифрования выполняются по следующим формулам

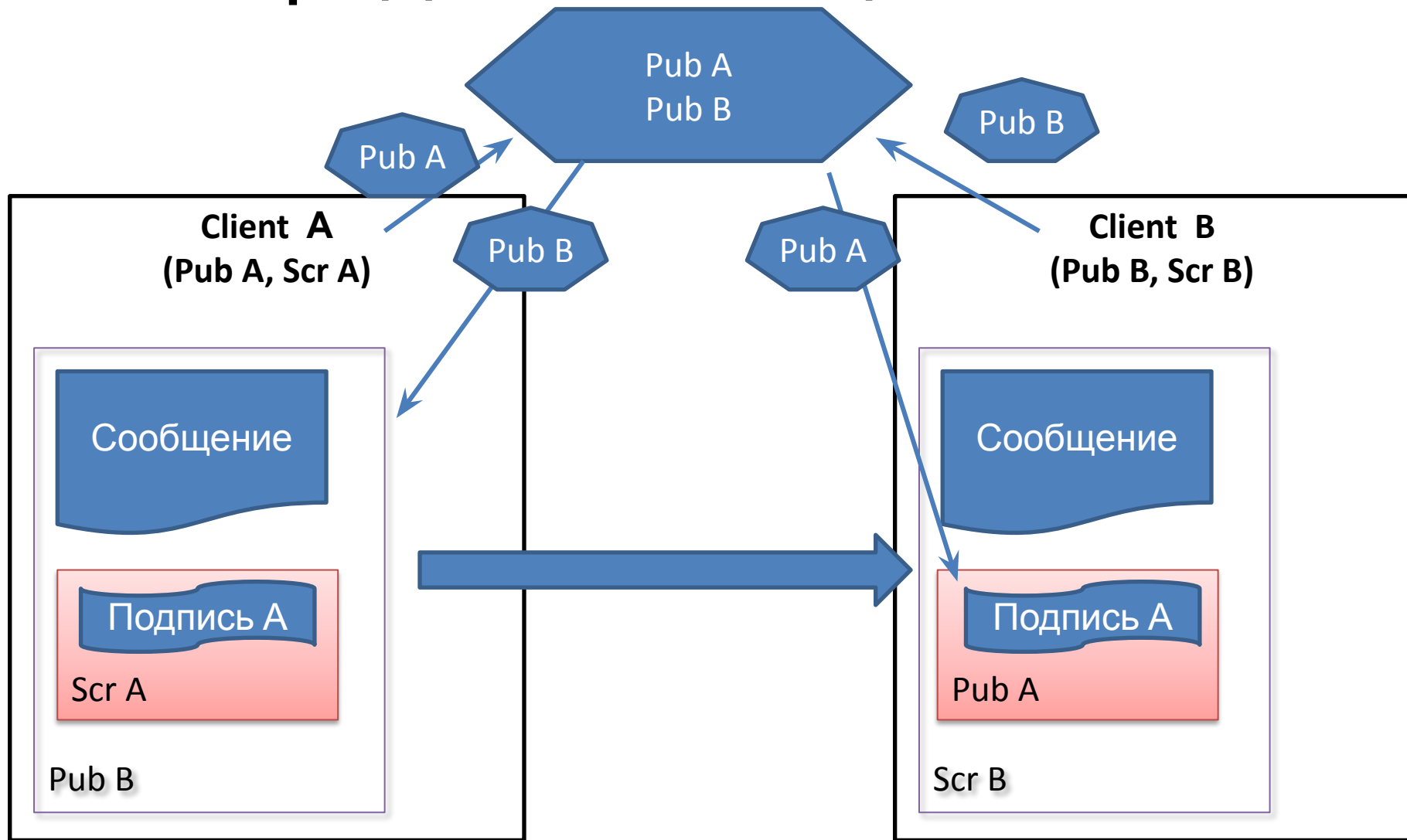
$$C = T^e \pmod n$$

$$T = C^d \pmod n$$

где  $T$ ,  $C$  - числовые эквиваленты символов открытого и шифрованного сообщения.

Открытое сообщение, $T$	Символ	А	Б	Р	А	М	О	В
	Код	1	2	18	1	14	16	3
Шифрограмма, $C = T^5 \pmod{91}$		1	32	44	1	14	74	61
Открытое сообщение, $T = C^{29} \pmod{91}$		1	2	18	1	14	16	3

# Передача сообщения RSA



# Основные алгоритмы

## шифрования

- **RSA (Ron Rivest, Adi Shamir, and Leonard Adleman Algorithm)** — криптографический алгоритм с открытым ключом. RSA
- **DES (Data Encryption Standard)** — симметричный алгоритм шифрования, разработанный фирмой IBM и утвержденный правительством США в 1977 году как официальный стандарт (FIPS 46-3). DES имеет блоки по 64
- **3DES (Triple Data Encryption Standard)** — симметричный блочный шифр, созданный Уитфилдом Диффи, Мартином Хеллманом и Уолтом Тачманном в 1978 году на основе алгоритма DES
- **AES (Advanced Encryption Standard)**, также известный как Rijndael — симметричный алгоритм блочного шифрования (размер блока 128 бит, ключ 128/192/256 бит)

**PKI (Public Key Infrastructure, Инфраструктура открытых ключей)** — технология аутентификации с помощью открытых ключей. Это комплексная система, которая связывает открытые ключи с личностью пользователя посредством удостоверяющего центра (УЦ).

**Certificate** — цифровой или бумажный документ, подтверждающий соответствие между открытым ключом и информацией, идентифицирующей владельца ключа. Содержит информацию о владельце ключа, сведения об открытом ключе, его назначении и области применения, название центра сертификации и т. д.

# Kerberos Operation



Authentication Server

Step 1

Ticket Granting Ticket : [client, address, validity,  $Key_{(client, TGS)}$ ]  $Key_{(TGS)}$   
[ $Key_{(client, TGS)}$ ]  $Key_{(client)}$

Step 2

TGT : service, [client, client address, validity,  $Key_{(client, TGS)}$ ]  $Key_{(TGS)}$   
Authenticator : [client, timestamp]  $Key_{(client, TGS)}$

Step 3

Ticket  $_{(client, service)}$  : service, [client, client address, validity,  $Key_{(client, service)}$ ]  $Key_{(service)}$   
[ $Key_{(client, service)}$ ]  $Key_{(client, TGS)}$

Step 4

Ticket  $_{(client, service)}$  : service, [client, client address, validity,  $Key_{(client, service)}$ ]  $Key_{(service)}$   
Authenticator : [client, timestamp]  $Key_{(client, service)}$



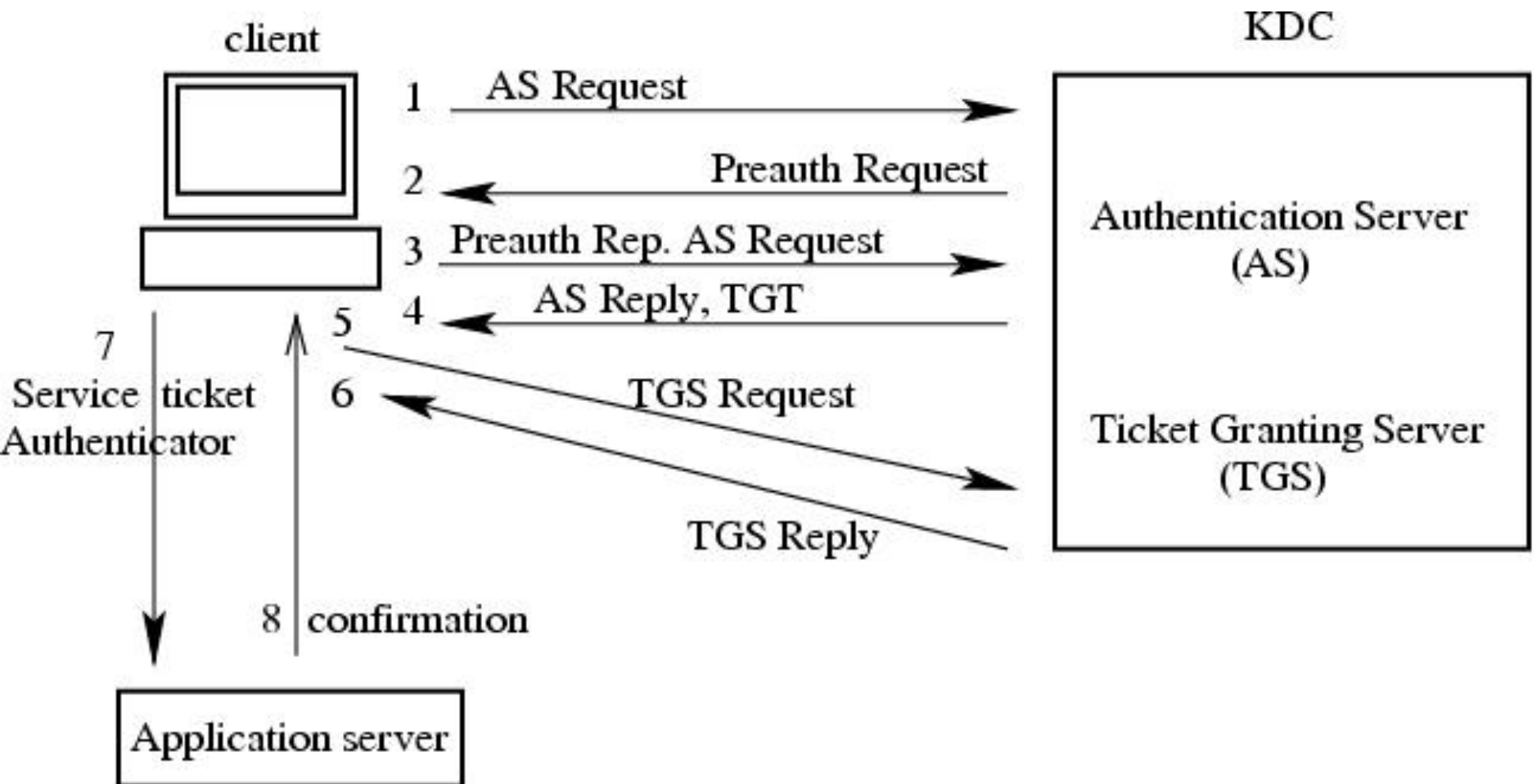
Ticket Granting S



Print Server

1) service is the networked resource that the client is trying to access (e.g. Print Server);

2) TGS is the Ticket Granting Server



# Authentication procedure: (процедура аутентификации)

1. Клиент посылает **user principal name** (kerberos account) -> KDC.
2. KDC (AS) отвечает **pre-authentication request**.
3. Клиент посылает **Authenticator**: the client's principal and the time stamp, зашифрованные при помощи user's key (password hash).
4. KDC (AS) посылает:  
**the Ticket Granting Ticket (TGT)**, зашифрованный при помощи the TGS key  
and a **client/TGS session key**, зашифрованный при помощи the user's key.

Клиент расшифровывает **the session key** и кэширует **the TGT**.

The TGT включает: the TGS copy of the *client/TGS* session key, client principal, ticket lifetime, KDC timestamp, client IP address.

# Kerberos services request procedure:

5. Клиент посылает KDC (TGS):

**TGT**

**Authenticator:** client principal and the time stamp, зашифрованные с помощью the client/TGS session key

запрашиваемое **service principal name**.

6. KDC (TGS) сравнивает TGT and the Authenticator, и посылает клиенту:

**Service ticket**, зашифрованный с помощью the Server key.

Service ticket: client/Server session key, client principal, ticket lifetime, KDC timestamp, client IP address.

client/Server session key шифруется с помощью the client/TGS session key.

7. Клиент посылает серверу приложений:

**the Service ticket и Authenticator** (the client principal and time stamp , зашифрованные с помощью the client/Server session key).

8. Сервер приложений расшифровывает the Service ticket своим ключом, хранящимся в keytab, /etc/krb5.keytab, сверяет Authenticator клиента и посылает a confirmation (client time stamp + 1), зашифрованные с помощью the client/Server session key.

9. Клиент расшифровывает the confirmation с помощью the client/Server session key и проверяет правильность изменения client time stamp . В случае успеха, клиент посылает запрос серверу.