

GR 8

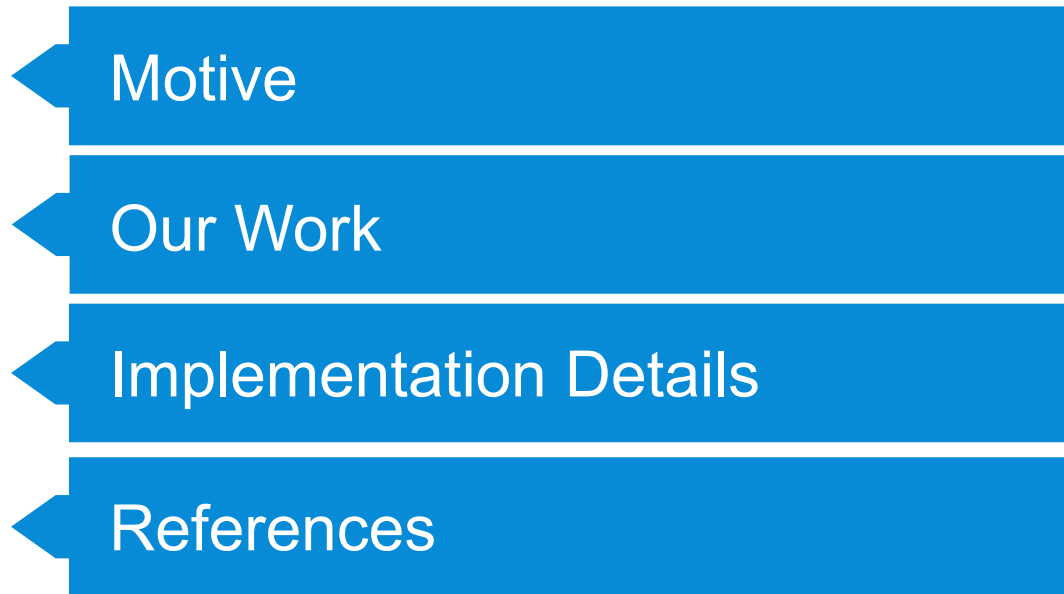
Secure Instant Messaging

Chenguang Zhang

Ziqi Chen

Lingqi Meng

Context



Motive



- Replay
- Eavesdropping
- Replay
- ...

Our work

1

A secure instant messaging system

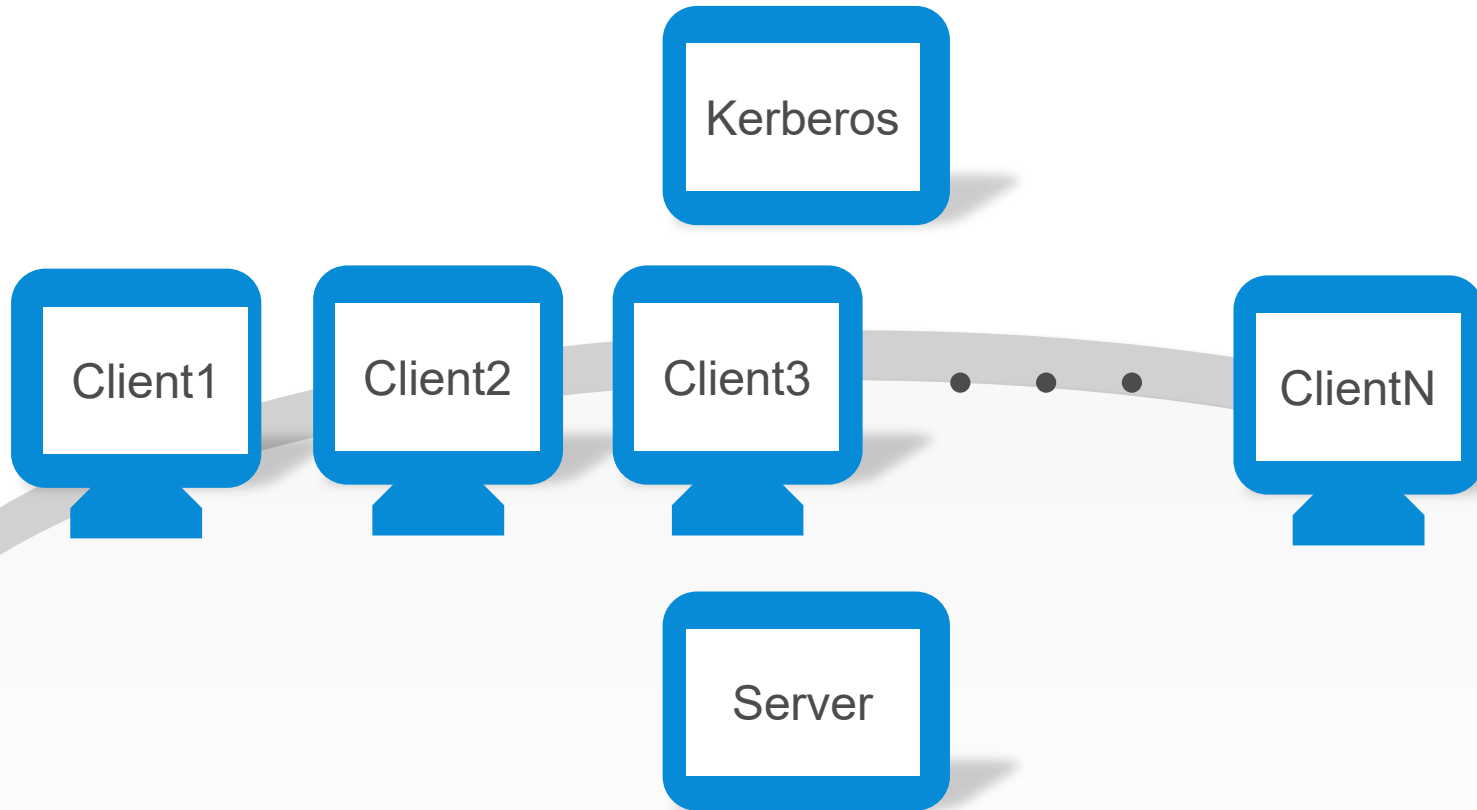
2

Functionality: logging in, online list, secure text messaging, etc.

3

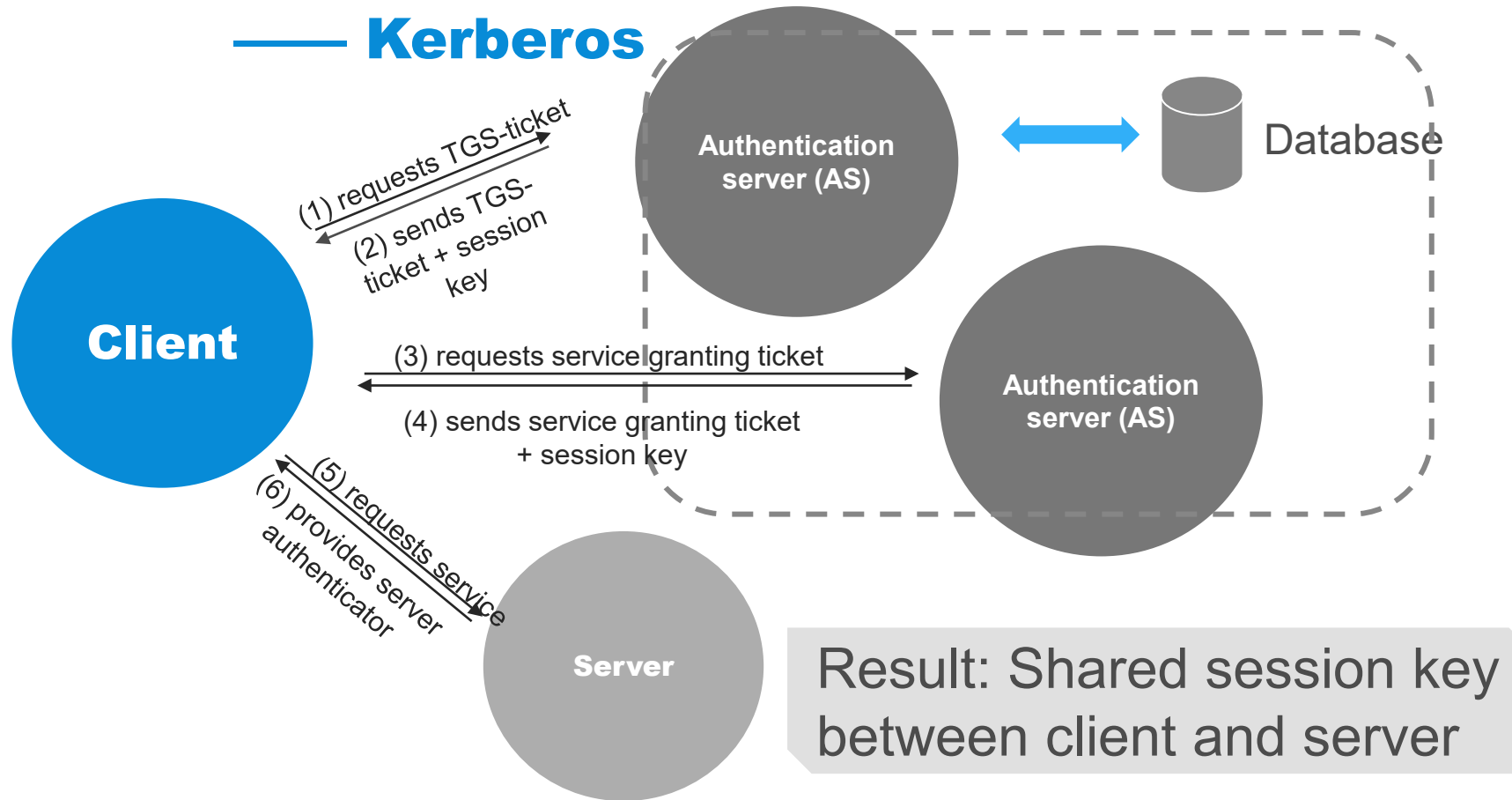
Security Services: confidentiality, peer-entity and message authentication, integrity, nonrepudiation

Client-Server Architecture with Kerberos Authentication

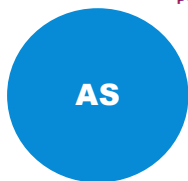
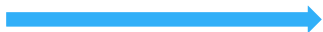


Phase I : Authentication

— Kerberos



Phase I

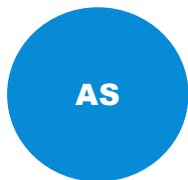
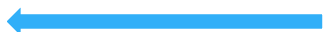


```
public AS() {  
    sessiongeneration aeskey = new sessiongeneration();  
    sessionkeytgscient = aeskey.aeskey();  
    begintime = a.begintime();  
    endtime = a.endtime();  
    try {  
        ASsocket = new ServerSocket(10800);  
    } catch (IOException e) {  
        e.printStackTrace();  
    }  
}
```

AS listening
at port 10800

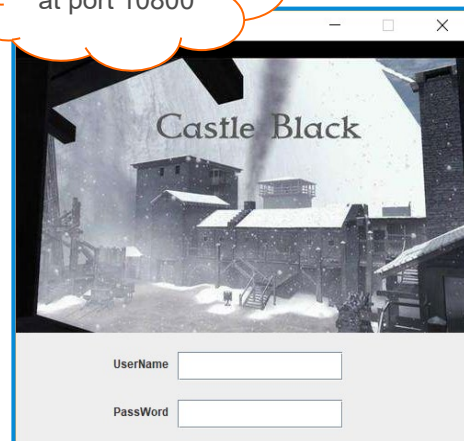
```
String transmit = clientid + "@" + tgsid + "@" + begintime;
```

04fc711301f3c784d66955d98d399afb@11111111@20171212200526

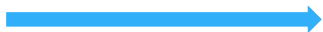


```
if (tgsidfromsocket.equals(tgsid) && clientIdflag) {  
    String tickettgsumprocess = sessionkeytgscient + "@" + clientIdfromsocket + "@" + tgsid;  
    String tickettgs = AES.Encrypt(tickettgsumprocess, tgskey);  
    String resultunprocess = sessionkeytgscient + "@" + tgsid + "@" + tickettgs;  
    result = AES.Encrypt(resultunprocess, clientkey);  
    System.out.println("the TGS ticket in AS is " + tickettgs);  
} else {  
    result = "unauthorized client";  
}
```

suTguRN2FjschXw=@11111111@
4f15e903add2d7b557613ea42588
53b0255e3fe4442166b59d692cd8c
969ca58ffaf45e56109b649a4cb2d
b78f13c9e35cd8b565ba938435203
67282e30cbf4

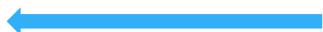


Request TGS



```
String auth = AES.Encrypt(clientid, sessionkeytgsclient);
String transmit = serverid + "@" + tickettgs + "@" + auth;
```

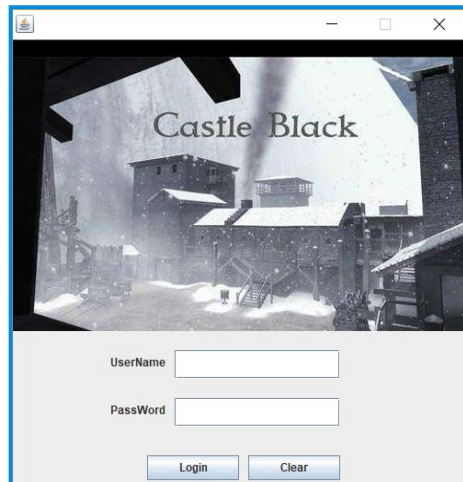
```
22222222@4f15e903add2d7b557613ea4258853b0255e3fe4442166b59d69
2cd8c969ca58ffaf45e56109b649a4cb2db78f13c9e35cd8b565ba9384352036
7282e30cbf4@d6a80c862c1039de8bc9d21a2251b25
```



```
ofBtvBWdGZ3LDio=@22222222
@442956401baf0a28359cd224f54
ccfbf395df880e32f65cc66610139d
98b3ba4ed76e6e2f9c1003d26117d
6d8d80ab04
```

```
public TGS() {
    sessiongeneration aeskey = new sessiongeneration();
    sessionkeyclientserver = aeskey.aeskey();
    try {
        TGSsocket = new ServerSocket(10801);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

TGS listening
at port 10801



```
if(clientidfromAS.equals(clientidfromclient)&&TGSidfromAS.equals(tgsid)&&serveridfromclient.equals(serverid)){
    String ticketserverunprocess = sessionkeyclientserver + "@" + clientidfromclient + "@" + serverid;
    String ticketserver = AES.Encrypt(ticketserverunprocess, serverkey);
    String transmitunprocess = sessionkeyclientserver + "@" + serverid + "@" + ticketserver;
    result = AES.Encrypt(transmitunprocess, sessionkeytgsclient);
}else{
    result = "unauthorized client";
}
```


Connect to server

Client

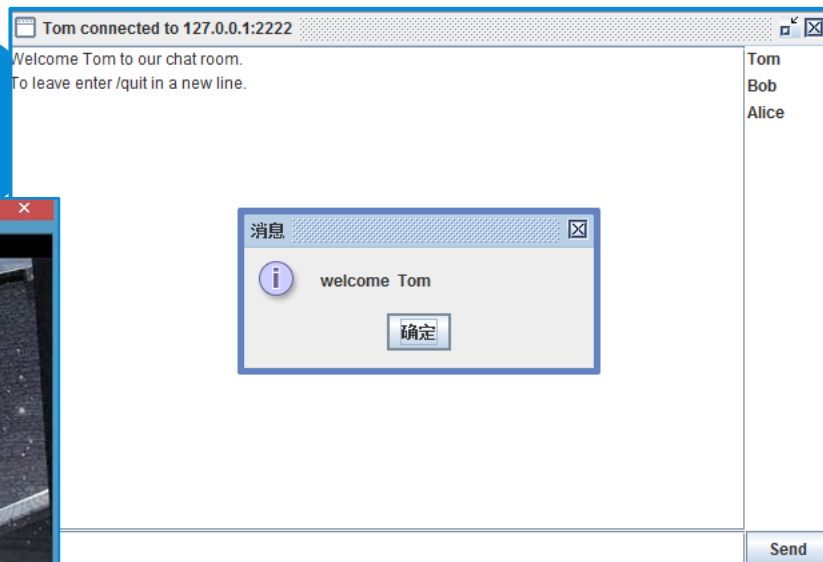
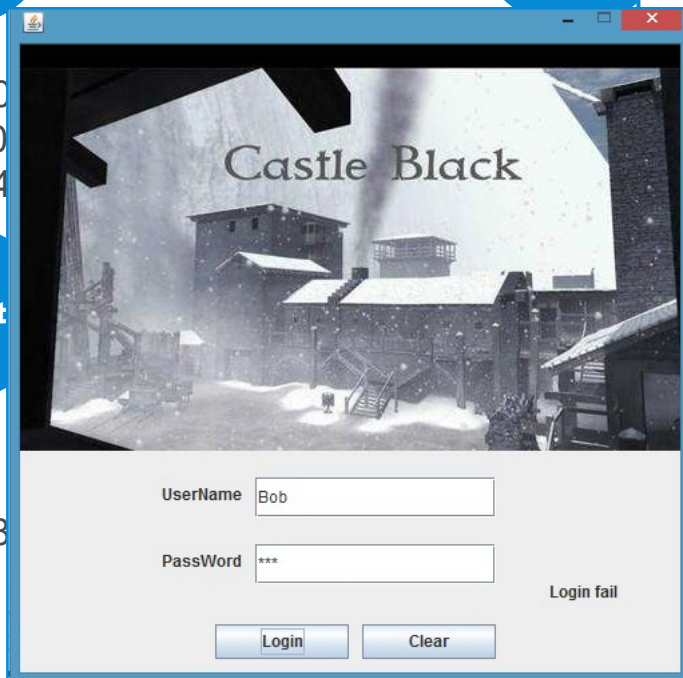
ticket@auth

Server

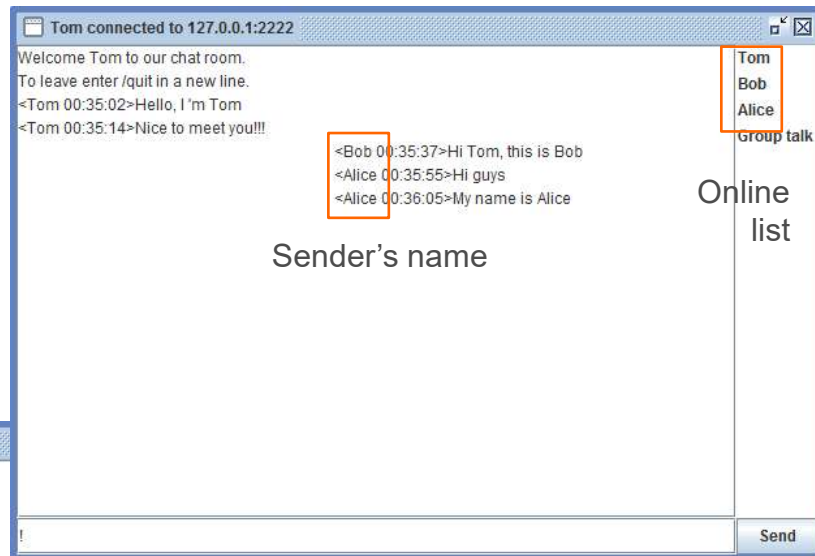
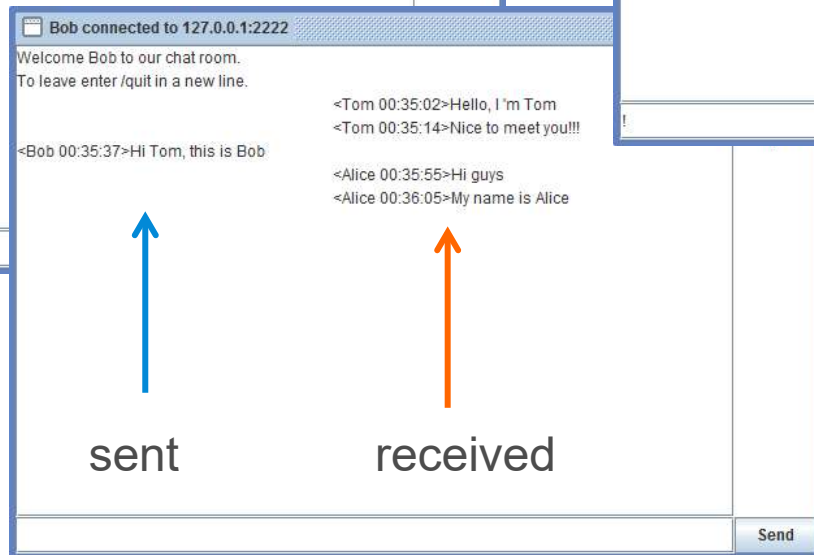
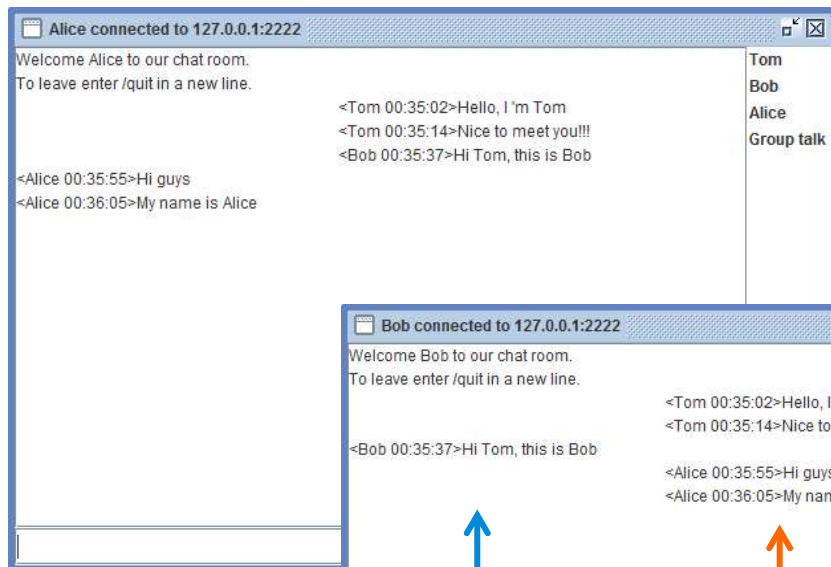
44295640
5cc66610
8d80ab04

Client

b268



Group Chat (default)



Private Chat

Alice connected to 127.0.0.1:2222

Welcome Alice to our chat room.
To leave enter /quit in a new line.

<Tom 00:35:02>Hello, I'm Tom
<Tom 00:35:14>Nice to meet you!!!
<Bob 00:35:37>Hi Tom, this is Bob

<Alice 00:35:55>Hi guys
<Alice 00:36:05>My name is Alice

@Bob <Alice 00:39:50>Nice to meet you too

@Alice <Bob 00:38:48>Hi Alice, Nice to meet you
@Alice <Bob 00:42:42>Do you have time tomorrow
@Alice <Bob 00:44:16>I know a new restaurant
@Alice <Bob 00:44:52>It is pretty good

<Alice 00:45:20>Great, we can have a try tomorrow!

Tom
Bob
Alice
Group talk

Private message to Alice

Tom connected to 127.0.0.1:2222

Welcome Tom to our chat room.
To leave enter /quit in a new line.

<Tom 00:35:02>Hello, I'm Tom
<Tom 00:35:14>Nice to meet you!!!

<Bob 00:35:37>Hi Tom, this is Bob
<Alice 00:35:55>Hi guys
<Alice 00:36:05>My name is Alice

Tom
Bob
Alice
Group talk

Tom cannot receive the private message between Alice and Bob

Bob connected to 127.0.0.1:2222

Welcome Bob to our chat room.
To leave enter /quit in a new line.

<Tom 00:35:02>Hello, I'm Tom
<Tom 00:35:14>Nice to meet you!!!

<Bob 00:35:37>Hi Tom, this is Bob

<Alice 00:35:55>Hi guys
<Alice 00:36:05>My name is Alice

@Bob <Alice 00:39:50>Nice to meet you too

@Alice <Bob 00:38:48>Hi Alice, Nice to meet you
@Alice <Bob 00:42:42>Do you have time tomorrow
@Alice <Bob 00:44:16>I know a new restaurant
@Alice <Bob 00:44:52>It is pretty good

<Alice 00:45:20>Great, we can have a try tomorrow!

Tom
Bob
Alice
Group talk

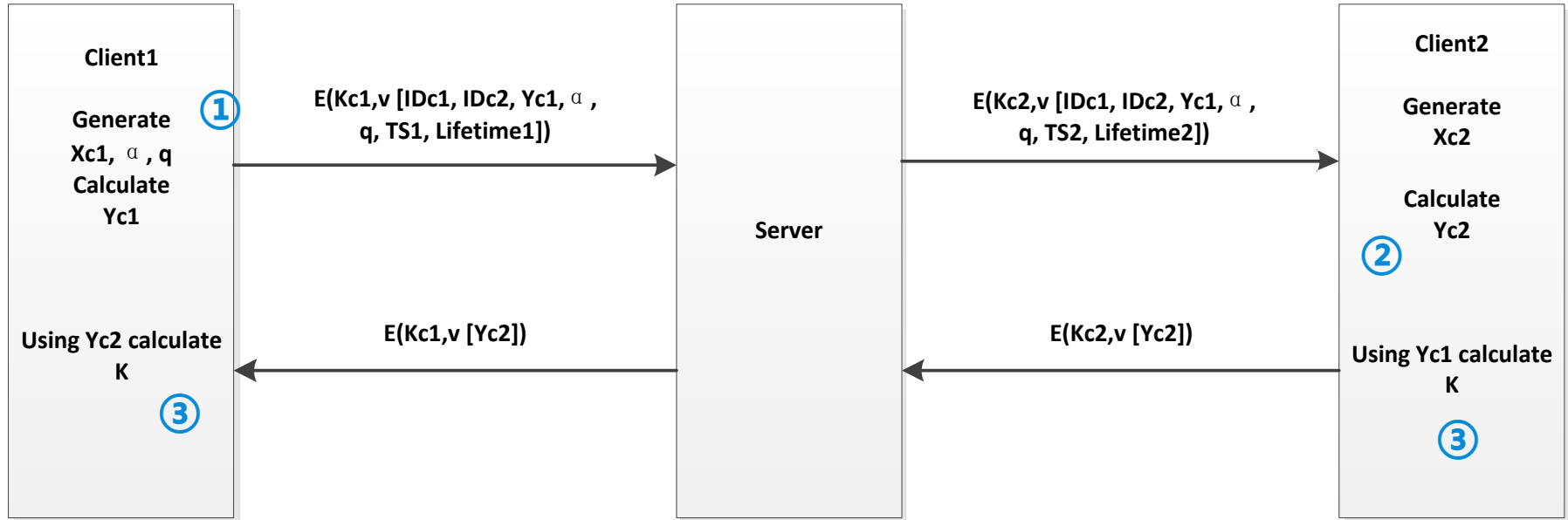
Message aim

It is pretty good

Send

Phase II : Client Key Exchange

— Diffie-Hellman Algorithm



Result: Exchange secret key K between Client1 and Client2

Phase II : Client Key Exchange

— Diffie-Hellman Algorithm

①

```
public HQKeyPair initPartyAKey(int keySize) throws NoSuchAlgorithmException
{
    KeyPairGenerator keyPairGen = KeyPairGenerator.getInstance(ALGORITHM);
    keyPairGen.initialize(keySize);
    KeyPair keyPair = keyPairGen.generateKeyPair();
    return new HQKeyPair(keyPair);
}
```

②

```
public HQKeyPair initPartyBKey(byte[] partyAPublicKey) throws Exception
{
    X509EncodedKeySpec x509KeySpec = new X509EncodedKeySpec(partyAPublicKey);
    KeyFactory keyFactory = KeyFactory.getInstance(ALGORITHM);
    PublicKey pubKey = keyFactory.generatePublic(x509KeySpec);

    DHParameterSpec dhParamSpec = ((DHPublicKey) pubKey).getParams();

    KeyPairGenerator keyPairGenerator = KeyPairGenerator.getInstance(keyFactory.getAlgorithm());
    KeyPair keyPair = keyPairGenerator.generateKeyPair();
    return new HQKeyPair(keyPair);
}
```

③

```
private SecretKey getSecretKey(byte[] publicKey, byte[] privateKey, String algorithm) throws Exception
{
    KeyFactory keyFactory = KeyFactory.getInstance(ALGORITHM);
    X509EncodedKeySpec x509KeySpec = new X509EncodedKeySpec(publicKey);
    PublicKey pubKey = keyFactory.generatePublic(x509KeySpec);

    PKCS8EncodedKeySpec pkcs8KeySpec = new PKCS8EncodedKeySpec(privateKey);
    Key priKey = keyFactory.generatePrivate(pkcs8KeySpec);

    KeyAgreement keyAgree = KeyAgreement.getInstance(ALGORITHM);
    keyAgree.init(priKey);
    keyAgree.doPhase(pubKey, KeyAgreement.DH_STEP1);

    SecretKey secretKey = keyAgree.generateSecret(ALGORITHM);

    return secretKey;
}
```

```
login (1) [Java Application] D:\software\JDK\bin\javaw.exe (2017-12-14 上午4:16:25)
B_PublicKey: MIIBpjCCARsGCSqGSIb3DQEDATCCAQwCgYEA/X9TgR11Ei1S30qcLuzk5/YRt1I870QAwx4/gLZRJm1FXUAIUftZPY1Y+r/F9bow9subVWzXgTuAHTv8mZgt2uZUKWk...
A_PrivateKey: MIIBZwIBADCCARsGCSqGSIb3DQEDATCCAQwCgYEA/X9TgR11Ei1S30qcLuzk5/YRt1I870QAwx4/gLZRJm1FXUAIUftZPY1Y+r/F9bow9subVWzXgTuAHTv8mZgt2u...
secretKey: EDxcpK7f6hy5JvQ=
=====
before encrypt: <Bob 04:16:55>Hello everyone
after encrypt: a5d047b7a246c2033084630ed8a581fa8d04ac0dedfe0cca00669eac7f220cdd
=====
before_decrypt: a5d047b7a246c2033084630ed8a581fa8d04ac0dedfe0cca00669eac7f220cdd
after_decrypt: <Bob 04:16:55>Hello everyone
=====
```

Console print from client

Phase III : Encrypted Communication

— AES

```
// encryption
public static String Encrypt(String plaintext, String sKey) throws Exception {
    if (sKey == null) {
        System.out.println("Key cannot be null");
        return null;
    }

    if (sKey.length() != 16) {
        System.out.println("Key's length must be 16");
        return null;
    }

    byte[] raw = sKey.getBytes();
    SecretKeySpec skeySpec = new SecretKeySpec(raw, "AES");
    Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
    IvParameterSpec iv = new IvParameterSpec("0102030405060708".getBytes());
    cipher.init(Cipher.ENCRYPT_MODE, skeySpec, iv);
    byte[] encrypted = cipher.doFinal(plaintext.getBytes());

    return byte2hex(encrypted).toLowerCase();
}
```

```
public static String Decrypt(String myCipher, String sKey) throws Exception {
    try {
        if (sKey == null) {
            System.out.println("Key cannot be null");
            return null;
        }

        if (sKey.length() != 16) {
            System.out.println("Key's length must be 16");
            return null;
        }

        byte[] raw = sKey.getBytes("ASCII");
        SecretKeySpec skeySpec = new SecretKeySpec(raw, "AES");
        Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
        IvParameterSpec iv = new IvParameterSpec("0102030405060708".getBytes());
        cipher.init(Cipher.DECRYPT_MODE, skeySpec, iv);
        byte[] encrypted1 = hex2byte(myCipher);

        try {
            byte[] original = cipher.doFinal(encrypted1);
            String originalString = new String(original);
            return originalString;
        } catch (Exception e) {
            System.out.println(e.toString());
            return null;
        }

        } catch (Exception ex) {
            System.out.println(ex.toString());
            return null;
        }
    }
}
```

Bad Test --Sniffing

The image shows a Windows environment with two windows. The top window is Npcap, capturing traffic on the Npcap Loopback Adapter. It displays a list of network packets. The bottom window is a Java application titled 'LogStat', which is a client-server application. The console output shows the server's response to the client's request, which is encrypted.

Npcap Packet List:

No.	Time	Source	Destination	Protocol	Length	Info
159	53.061072	127.0.0.1	127.0.0.1	TCP	212	2222 → 51444 [PSH, ACK] Seq=925 Ack=463 Win=254 Len=64
160	53.061210	127.0.0.1	127.0.0.1	TCP	84	51444 → 2222 [ACK] Seq=463 Ack=989 Win=252 Len=0
161	53.061649	127.0.0.1	127.0.0.1	TCP	88	2222 → 51444 [PSH, ACK] Seq=989 Ack=463 Win=254 Len=2
162	53.061802	127.0.0.1	127.0.0.1	TCP	84	51444 → 2222 [ACK] Seq=463 Ack=991 Win=252 Len=0
163	53.062262	127.0.0.1	127.0.0.1	TCP	212	2222 → 51444 [PSH, ACK] Seq=925 Ack=529 Win=254 Len=64
164	53.062392	127.0.0.1	127.0.0.1	TCP	84	51444 → 2222 [ACK] Seq=529 Ack=989 Win=252 Len=0
165	53.062677	127.0.0.1	127.0.0.1	TCP	88	2222 → 51444 [PSH, ACK] Seq=989 Ack=529 Win=254 Len=2
166	53.062808	127.0.0.1	127.0.0.1	TCP	84	51444 → 2222 [ACK] Seq=529 Ack=991 Win=252 Len=0
167	111.028715	192.168.0.11	224.0.0.252	IGMPv2	68	Membership Report group 224.0.0.252
168	111.029085	192.168.0.11	239.255.255.250	IGMPv2	68	Membership Report group 239.255.255.250
169	111.029326	192.168.0.11	224.0.0.251	IGMPv2	68	Membership Report group 224.0.0.251

Java Application Console Output:

```
ClientServer (1) [Java Application] D:\software\jdk\bin\javaw.exe (2017-12-14 上午4:16:08)
Server is ready
Now using port number=2222

=====
A client joined which is: Tom
=====

A client joined which is: Bob
=====

messages received by Server: a5d047b7a246c2033084630ed8a581fa8d04ac0dedfe0cca00669eac7f220cdd
messages received by Server: a5d047b7a246c2033084630ed8a581fa8d04ac0dedfe0cca00669eac7f220cdd
```

Network Data (Hex):

```
0000 02 00 00 00 45 00 00 68 70 3e 40 00 80 06 00 00 ....E..h p>@....
0010 7f 00 00 01 7f 00 00 01 08 ae c8 f4 1d cc 7d 15 .....}.
0020 71 bf c8 20 50 18 00 fe a7 ea 00 00 39 66 36 32 q..P...9f62
0030 33 35 35 36 65 63 66 63 36 31 31 66 66 36 31 66 3556ecfc 611ff61f
0040 34 63 66 31 63 66 39 65 65 39 38 36 32 38 30 65 4cf1cf9e e986280e
0050 61 32 62 37 38 32 34 65 61 36 61 39 63 36 66 61 a2b7824e a6a9c6fa
0060 66 65 35 66 38 31 33 64 61 34 63 33 fe5f813d a4c3
```

Console print from server

All messages has been encrypted, so even if attackers sniff the message, it is meaningless

Reference

1. Hiroaki Kikuchi, Minako Tada. (2004) Secure Instant Messaging Protocol Preserving Confidentiality against Administrator
2. William Stallings. (2011) Network Security Essentials: Applications and Standards, Fourth Edition.
3. Jimmy Florez Z, Camilo Logreira R. (2016) Architecture of Instant Messaging Systems for Secure Data Transmission
4. Changji Wang, Welong Lin (2013) Design of An Instant Messaging System Using Identity Based Cryptosystems.
5. B.Clifford Neuman, Theodore Ts'o. (1994) Kerberos: An Authentication Service for Computer Networks.
6. Anita Narwal, Sunita Tomar. (2015) Kerberos Protocol: A Review
7. Michael Steiner, Gene Tsudik. (1996) Diffie-Hellman Key Distribution Extended to Group Communication.

Thank you

Chenguang Zhang

Ziqi Chen

Lingqi Meng