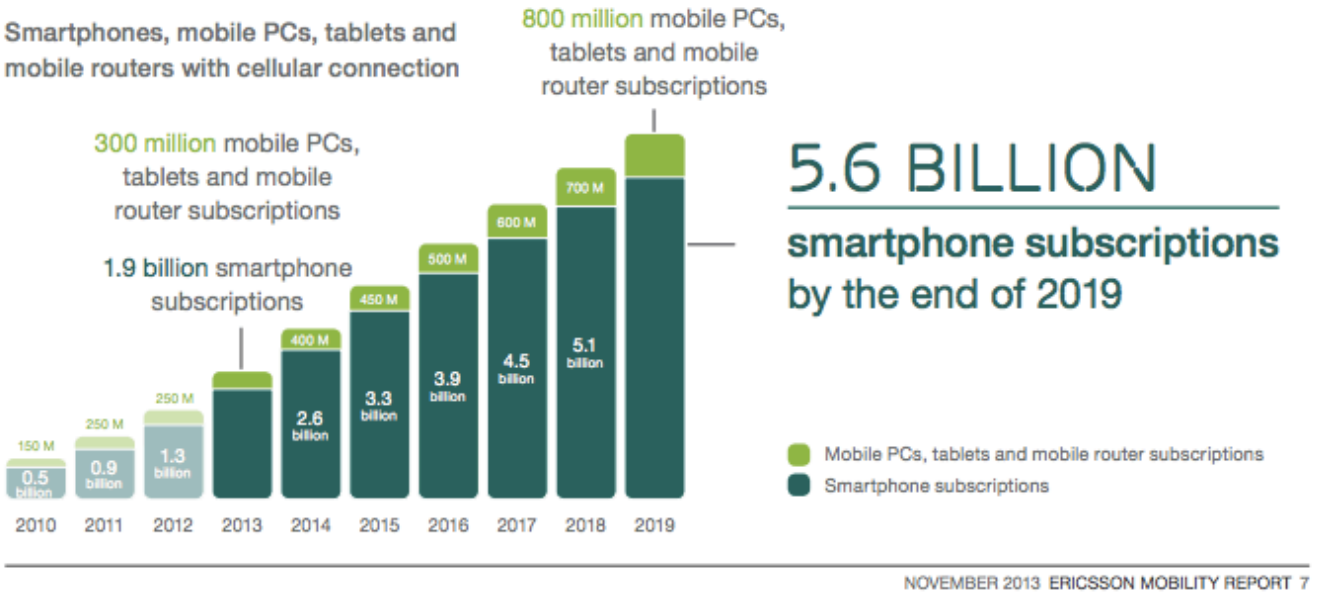
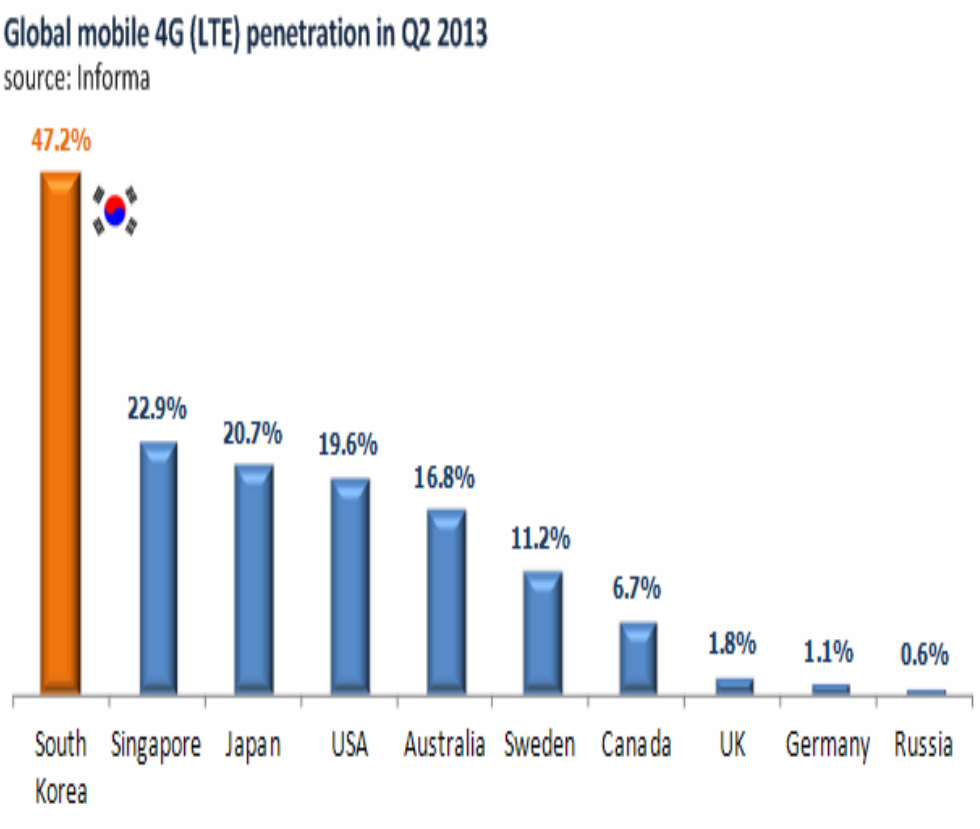
Introduction

Mobile Phone Users are rapidly switching over to Smartphones



Introduction



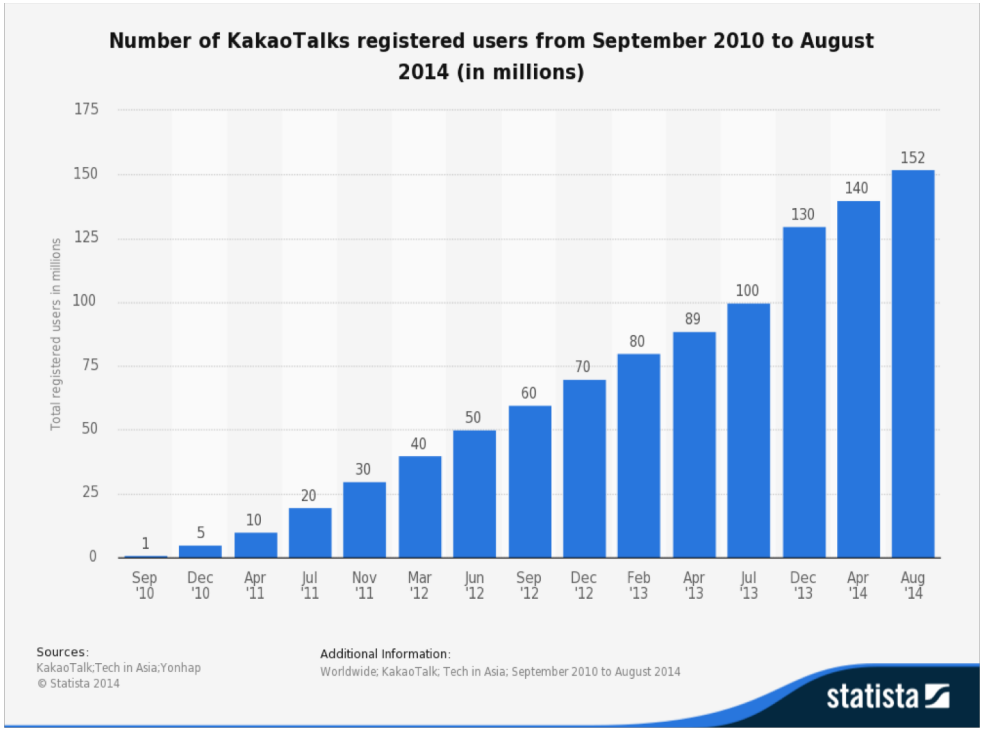
Motivation

- Instant Message Application User Growth by using smartphone
- Some people buy smartphone to use

Instant Message Application

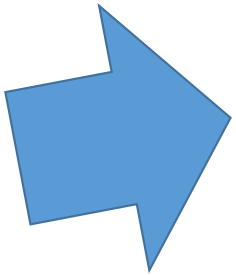
Motivation

- A couple of months ago, Instant Message Application Security Issue in Korea
- No secure algorithm
- Monitor
- Eavesdrop?



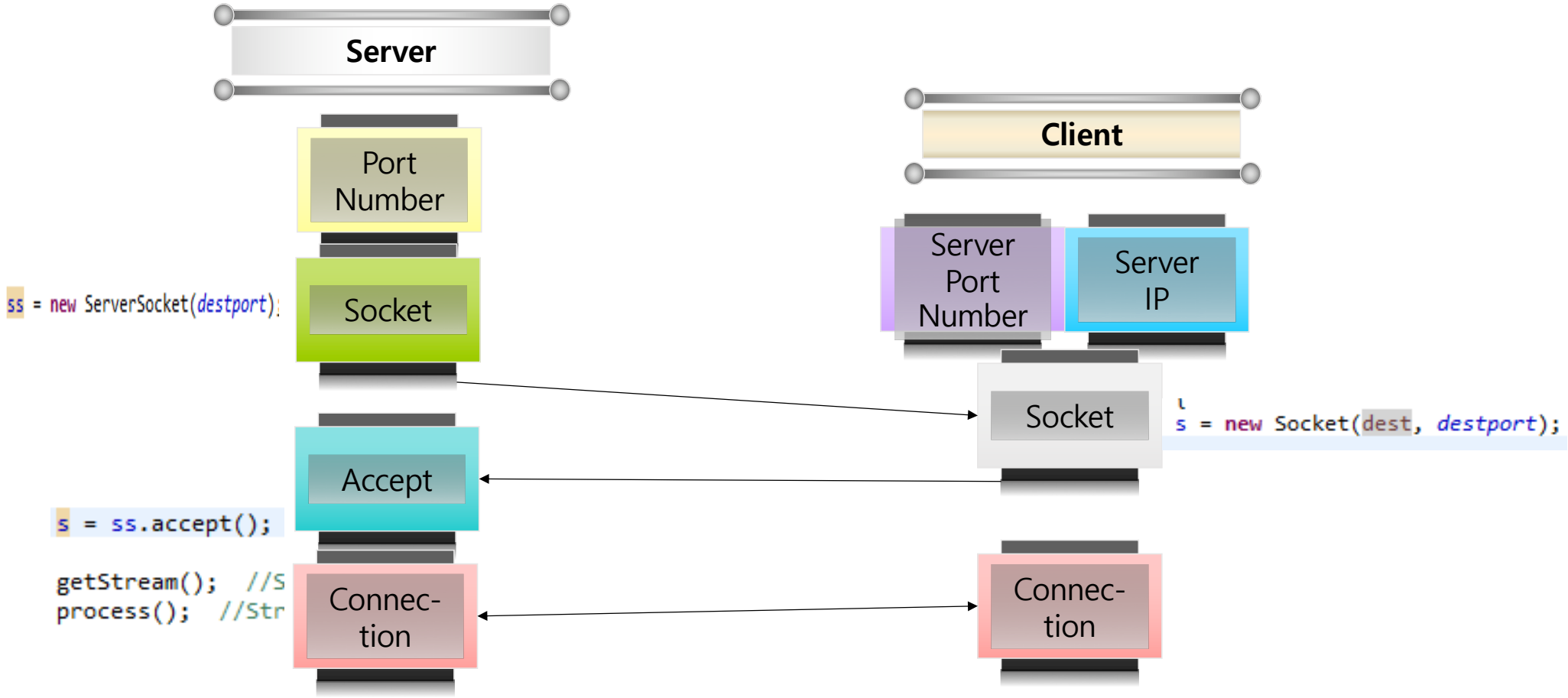
Purpose

- Messenger program using RSA algorithm



- Messenger program using AES
- Because Key Exchange issue

Socket Programming



Secure Algorithm

AES

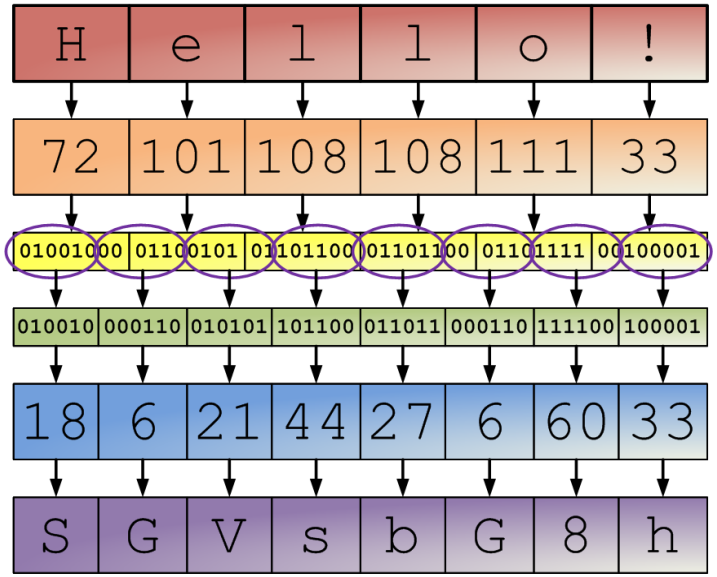
- Encrypt message
- Decrypt message

```
raw1 = skey.getEncoded(); // Make Key
deskey = new SecretKeySpec(raw1, "AES");

c = Cipher.getInstance("AES");
c.init(Cipher.ENCRYPT_MODE, deskey);

byte [] cipheroutput = message.getBytes();
byte [] ciphermessage = c.doFinal(cipheroutput);
```

```
String ciphertext = Base64.encode(ciphermessage);
String encryptedKey = Base64.encode(raw1);
```

Base64

- Encoding / decoding encrypted message & key



Demo

Demo

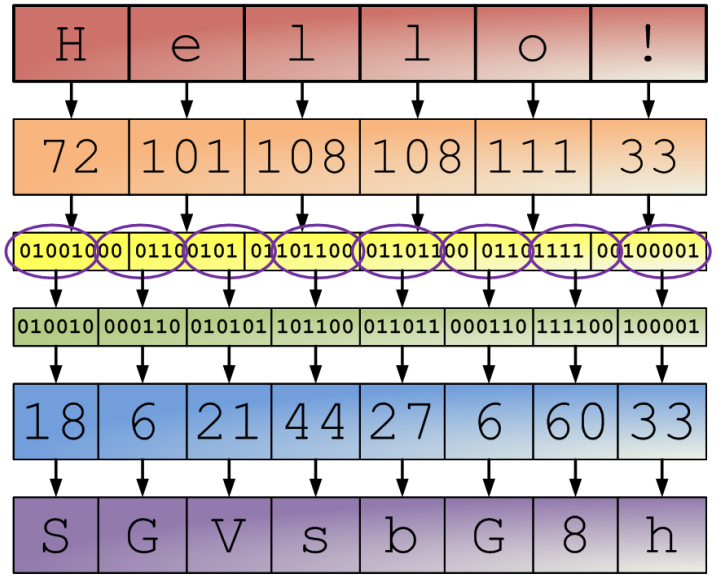
Conclusion / Future Work

- When using instant message app, the provider can give message security using secure algorithm
- Complete RSA
- Add signature


?
Q
&
&
A
!

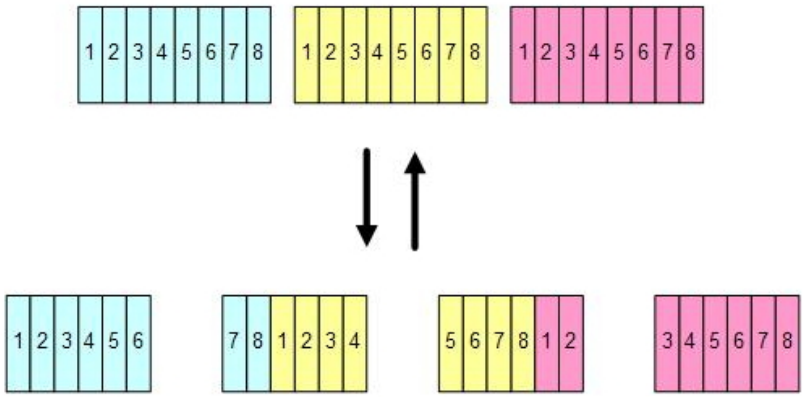
Base64

- Binary-to-Text ecoding schemes
- Use when there is a need to encode binary data
- Guarantee that data remains intact without modification during transport



Base64

- 1. 8bits -> 6 bits
- 2. 6 bits character
- 3. If not 24 bits, use padding.



Byte character	a (97)								b (98)								c (99)							
8 bit value	0	1	1	0	0	0	0	1	0	1	1	0	0	0	1	0	0	1	1	0	0	0	1	1
6 bit value	0	1	1	0	0	0	0	1	0	1	1	0	0	0	1	0	0	1	1	0	0	0	1	1
6 bit character	Y (24)								W (22)								J (9)							
																				j (35)				

Encode vs Encrypt

- Encoding transforms data into another format **using a scheme** that is publicly available so that it can easily be reversed.

  - For maintaining data usability and uses schemes that are publicly available

- Encryption transforms data into another format in such a way that **only specific individual can reverse** the transformation

  - For maintaining data confidentiality and thus the ability to reverse the transformation (keys) are limited to certain people

Key Exchange problem

- Whenever sending the message, the key creates.

```
textarea.append("\nSERVER>>> " + message);

KeyGenerator kgen = KeyGenerator.getInstance("AES");
SecretKey skey = kgen.generateKey();
kgen.init(128);
raw1 = skey.getEncoded(); // Make Key
deskey = new SecretKeySpec(raw1, "AES");

c = Cipher.getInstance("AES");
c.init(Cipher.ENCRYPT_MODE, deskey);

byte [] cipheroutput = message.getBytes();
byte [] ciphermessage = c.doFinal(cipheroutput);

//BASE64Encoder encoder = new BASE64Encoder();
String ciphertext = Base64.encode(ciphermessage);
String encryptedKey = Base64.encode(raw1);

textarea.append("\nSERVER(C)>>> " + ciphertext);

output.writeObject(ciphertext);
output.writeObject(encryptedKey);
```

```
byte[] raw1 = Base64.decode(decKey);

skeySpec = new SecretKeySpec(raw1, "AES");
c = Cipher.getInstance("AES");
c.init(Cipher.DECRYPT_MODE, skeySpec);

//BASE64Decoder decoder = new BASE64Decoder();

byte[] decodedMessage = Base64.decode(message);
byte[] clearmessage = c.doFinal(decodedMessage);

String cleartext = new String(clearmessage);

textarea.append("\nCLIENT>>> " + cleartext);
```