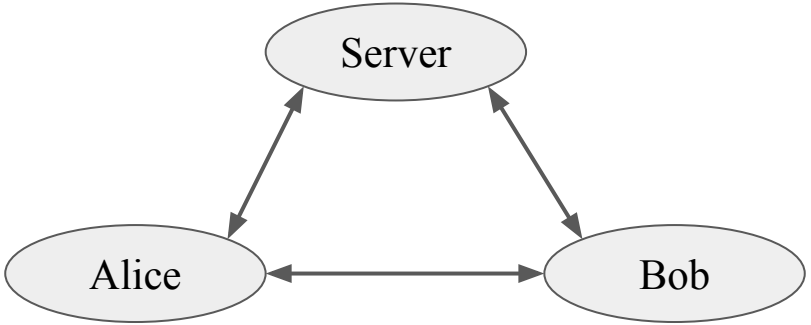
Cryptography Project 2
Chat Room

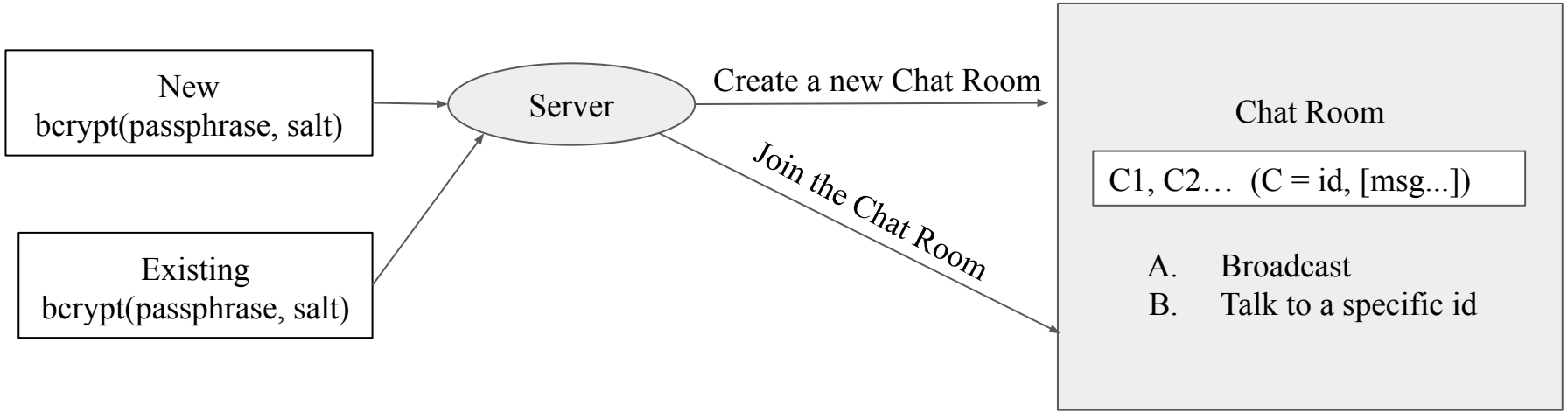
Marcus Barbu, Roy Xu, Mina Zhou

Application Scenario

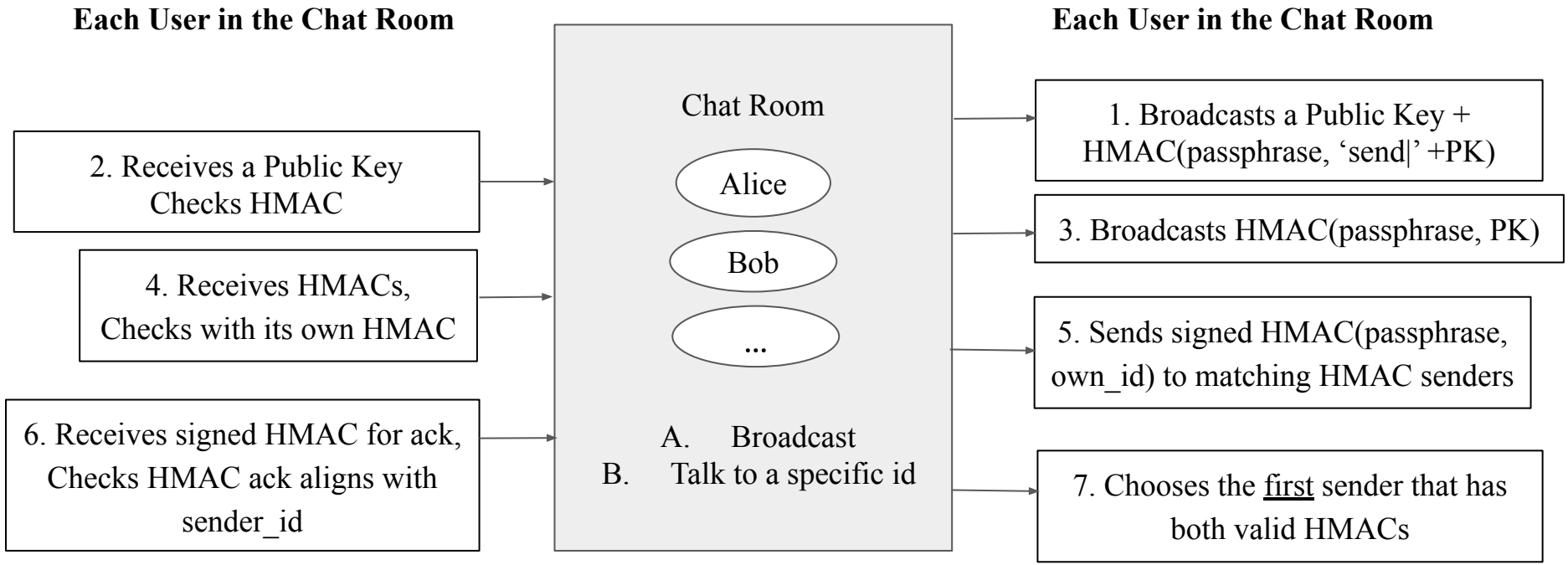
- Alice and Bob can and only can talk to each other
- Messages should be private from other users
- Messages should be private from the server



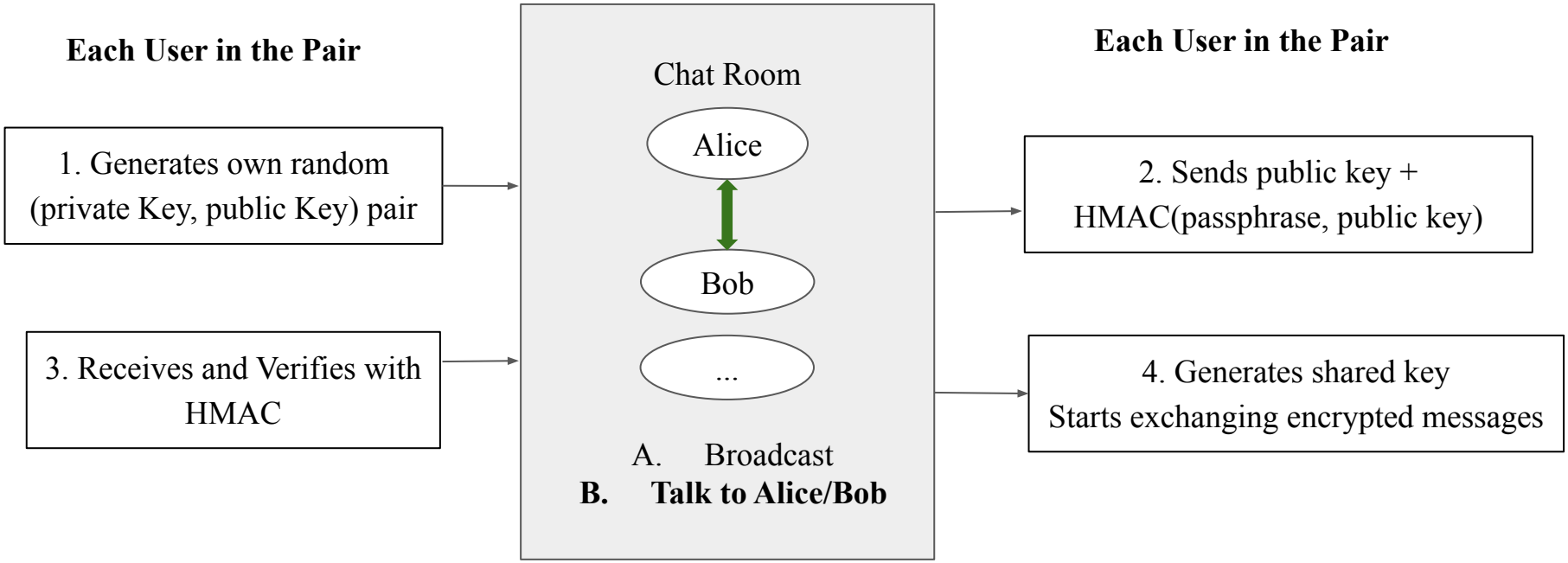
Application Detail - Server



Application Detail - Client, Pairing



Application Detail - Client, Encryption Key Exchange



Demo

Server Starts

```
~/NYU/CryptoProject2/server (roy) > python3 serve.py --debug --port 8000
INFO:root:ready to connect
```

First Client Connects

```
~/NYU/CryptoProject2/client (roy) > python client.py --port 8000
INFO:root:did not receive password
INFO:root:assigned password: a93b78a2-419f-4e45-aa37-2eca51bc1c22
INFO:root:attempting to connect
```

```
~/NYU/CryptoProject2/server (roy) > python3 serve.py --debug --port 8000
INFO:root:ready to connect
INFO:root:new connection
DEBUG:root:hash: b'JDJiJDEyJFNOdS5tMDlqWS5RcTV5YTJXWmlFYy56dnVRWDZmY0o2T21QVWVxcjQxe
VFBVFgxNi9DLzJx'
DEBUG:root:recv from b'0' type b'public': b'GQnSZlEW1272Ku1d6RX3VYRlmH+4kG3H9qFBMrex
6E18bc4dBy39sqNoOJ0axkVlLrPs/vDZ65Rqy6RIIcbAn2M='
```

Second Client Connects

```
~/NYU/CryptoProject2/client (roy) > python client.py --port 8000 --pass a93b78a2-419
f-4e45-aa37-2eca51bc1c22
INFO:root:attempting to connect
INFO:root:received valid hmac of public key
INFO:root:received public key
INFO:root:partner received valid hmac of public key
INFO:root:found valid partner
INFO:root:connected
```

```
~/NYU/CryptoProject2/client (roy) > python client.py --port 8000
INFO:root:did not receive password
INFO:root:assigned password: a93b78a2-419f-4e45-aa37-2eca51bc1c22
INFO:root:attempting to connect
INFO:root:received public key
INFO:root:partner received valid hmac of public key
INFO:root:received valid hmac of public key
INFO:root:found valid partner
INFO:root:connected
```

Second Client Connects (Server)

```
~/NYU/CryptoProject2/server (roy) > python3 serve.py --debug --port 8000
INFO:root:ready to connect
INFO:root:new connection
DEBUG:root:hash: b'JDJiJDEyJFNOdS5tMDlqWS5RcTV5YTJXWmlFYy56dnVRWDZmY0o2T21QVWVxcjQxe
VFBVFgxNi9DLzJx'
DEBUG:root:recv from b'0' type b'public': b'GQnSZlEW1272Ku1d6RX3VYRlmH+4kG3H9qFBMrex
6E18bc4dBy39sqNoOJ0axkVlLrPs/vDZ65Rqy6RIIcbAn2M='
INFO:root:new connection
DEBUG:root:hash: b'JDJiJDEyJFNOdS5tMDlqWS5RcTV5YTJXWmlFYy56dnVRWDZmY0o2T21QVWVxcjQxe
VFBVFgxNi9DLzJx'
DEBUG:root:recv from b'0' type b'public': b'0lM4adV3QaurUtM53cT90mN3RQ1SOMxCsApRplLP
UFl8/BSb8CzrKXbcIMWIg9KWdPZfE2KBxVQozN4DMo6ZCgs='
DEBUG:root:send to   b'0300abd2-db41-4397-a22f-84e45bbab4f6' type b'public': b'0lM4a
dV3QaurUtM53cT90mN3RQ1SOMxCsApRplLPUFl8/BSb8CzrKXbcIMWIg9KWdPZfE2KBxVQozN4DMo6ZCgs='
DEBUG:root:recv from b'0300abd2-db41-4397-a22f-84e45bbab4f6' type b'hmac': b'hYgYuoE
ZYquU5+e/MXp+/4lCINvzG1x7kvrCQek8pDg='
DEBUG:root:recv from b'0300abd2-db41-4397-a22f-84e45bbab4f6' type b'public': b'GQnSZ
lEW1272Ku1d6RX3VYRlmH+4kG3H9qFBMrex6E18bc4dBy39sqNoOJ0axkVlLrPs/vDZ65Rqy6RIIcbAn2M='
DEBUG:root:send to   b'360572c1-b14b-4305-8f8c-8c2d4f3adf2e' type b'hmac': b'hYgYuoE
ZYquU5+e/MXp+/4lCINvzG1x7kvrCQek8pDg='
DEBUG:root:send to   b'360572c1-b14b-4305-8f8c-8c2d4f3adf2e' type b'public': b'GQnSZ
lEW1272Ku1d6RX3VYRlmH+4kG3H9qFBMrex6E18bc4dBy39sqNoOJ0axkVlLrPs/vDZ65Rqy6RIIcbAn2M='
DEBUG:root:recv from b'360572c1-b14b-4305-8f8c-8c2d4f3adf2e' type b'ack': b'e10Kkrxf
7DuVE+zJZhSj+Nczdi4Rj/IfgRTn+h5lE9Y='
DEBUG:root:recv from b'360572c1-b14b-4305-8f8c-8c2d4f3adf2e' type b'hmac': b'FH3LLt5
aUynklXLsflHY4s3jdGhxR9ghpLUzShwWn6g='
DEBUG:root:send to   b'0300abd2-db41-4397-a22f-84e45bbab4f6' type b'ack': b'e10Kkrxf
7DuVE+zJZhSj+Nczdi4Rj/IfgRTn+h5lE9Y='
DEBUG:root:send to   b'0300abd2-db41-4397-a22f-84e45bbab4f6' type b'hmac': b'FH3LLt5
aUynklXLsflHY4s3jdGhxR9ghpLUzShwWn6g='
DEBUG:root:recv from b'0300abd2-db41-4397-a22f-84e45bbab4f6' type b'ack': b'y9CvC6oY
uM2yvwxUzw+Ok9VAOShAmafGkzgN4zamIQU='
DEBUG:root:send to   b'360572c1-b14b-4305-8f8c-8c2d4f3adf2e' type b'ack': b'y9CvC6oY
uM2yvwxUzw+Ok9VAOShAmafGkzgN4zamIQU='
```

Message Exchange

```
them: hello
you: hi
```

```
you: hello
them: hi
```

```
DEBUG:root:recv from b'0300abd2-db41-4397-a22f-84e45bbab4f6' type b'msg': b'Gn+77x1l
j0Nq6nth3HBiZeMXJ6MJXfPJtdOzpEMVq8Zlzlr/8qV7G9Af/H+RSQ=='
DEBUG:root:send to   b'360572c1-b14b-4305-8f8c-8c2d4f3adf2e' type b'msg': b'Gn+77x1l
j0Nq6nth3HBiZeMXJ6MJXfPJtdOzpEMVq8Zlzlr/8qV7G9Af/H+RSQ=='
DEBUG:root:recv from b'360572c1-b14b-4305-8f8c-8c2d4f3adf2e' type b'msg': b'JrLH8t1X
di9KgXsc4ksPh1GSCXYcD1pssBtow1Soyeum4pBsrZGR4/DKBg=='
DEBUG:root:send to   b'0300abd2-db41-4397-a22f-84e45bbab4f6' type b'msg': b'JrLH8t1X
di9KgXsc4ksPh1GSCXYcD1pssBtow1Soyeum4pBsrZGR4/DKBg=='
```

Attack Scenarios - Data Eavesdropping

Attack Method	Prevention/Mitigation
Intercept the bcrypt(keyphrase, salt) from server	It takes a long time to brute-force long keyphrase from bcrypt w/salt.
Monitor traffics on client pairing process	All broadcasted public keys are signed with hmac, cannot insert own public key without passphrase (breaks integrity)
Intercept conversation between Alice and Bob	Messages are encrypted Alice’s and Bob’s (private, public) key pairs are long and random
Intercept the public key exchange process	Private keys unknown by the attacker, cannot generate shared key

Attack Scenarios - Data Modification

Attack Method	Prevention/Mitigation
Modify HMACs during client pairing process	The attacker will fail HMAC checks and will not pair with the user
Modify public key during Alice and Bob's key exchange process	Because the passphrase is unknown, the attacker cannot modify HMAC(passphrase, public key), which will fail the HMAC check
Modify encrypted conversation	The attacker will not be able to obtain useful information from this. Invalid messages are discarded by clients.

Attack Scenarios - Data Replay

Attack Method	Prevention/Mitigation
Intercept the bcrypt(keyphrase, salt) from server, and join the Chat Room with legit users	Since the attacker doesn't know the plaintext passphrase, he/she will fail verification process
Sniff public key HMAC during Client pairing process, send the same hash as the legit sender	The attacker can not modify the public key because of the HMAC verification. Attacker does not have access to corresponding private key, so unable to generate shared secret
Sniff ack HMAC during Client pairing process, send the same hash as the legit sender	Because the receiver checks signed HMAC of id, attacker will not be able to insert their own id without knowing passphrase

Thank you :)