Task 1 descriptions and examples

We begin at a point when it is assumed that A and B have exchanged public keys these public keys are in files publicA.pem and publicB.pem. We assume that there is a public announcement for public key distribution.

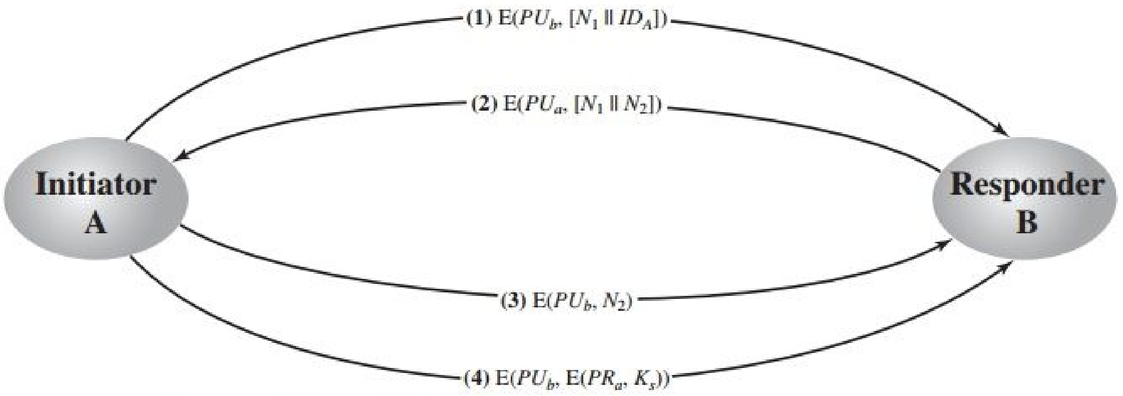
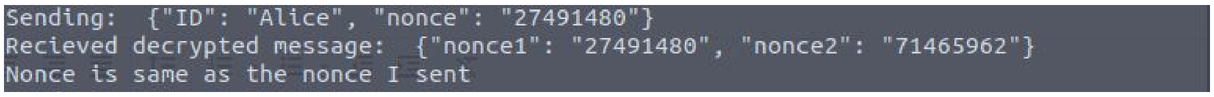

(1) E(PU_b, [N_1 ‖ ID_A])
(2) E(PU_a, [N_1 ‖ N_2])
Initiator A
Responder B
(3) E(PU_b, N_2)
(4) E(PU_b, E(PR_a, K_s))

Figure 14.8 Public-Key Distribution of Secret Keys

We followed the above diagrams' technique for distributing the symmetric key. We used RSA for the asymmetric key encryption.

MESSAGE 1 & 2:
A uses B's public key to encrypt a message to B containing an identifier of A(IDA) and a nonce (N1), which is used to identify this transaction uniquely. B sends a message to A encrypted with PUa and containing A's nonce (N1) as well as a new nonce generated by B (N2), because only B could have (N2). Because only B could have decrypted message (1), the presence of N1 in message (2) assures A that the correspondent is B.

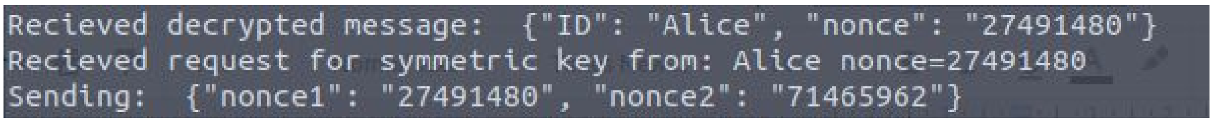
In the output of bob.py and alice.py each message that is sent and each message that is received is printed to the terminal. Here is an example of Alice's first three lines of output (alice.py).



```
Sending:  {"ID": "Alice", "nonce": "27491480"}
Recieved decrypted message:  {"nonce1": "27491480", "nonce2": "71465962"}
Nonce is same as the nonce I sent
```

The first line of the output is her sending message 1, the second line is alice receiving message 2.

Below is an example of output from Bob for the same first two messages. The first line is Bob receiving the ID and nonce (message 1). The third line is bob sending the two nonces (message 2).

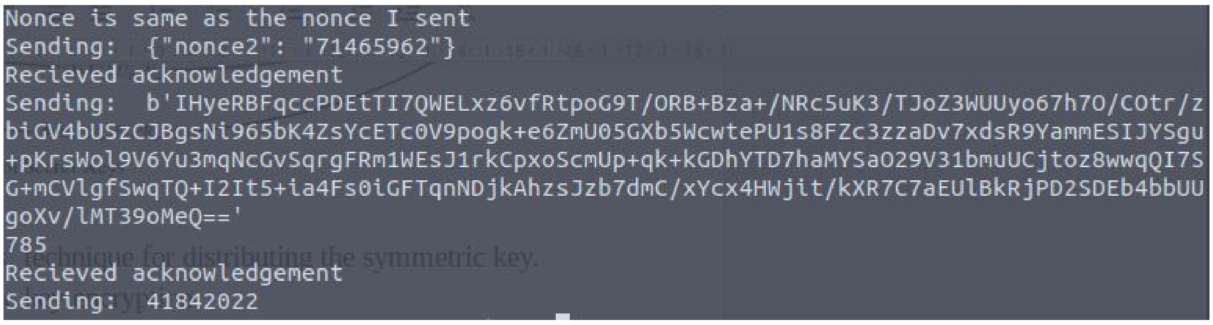


```
Recieved decrypted message:  {"ID": "Alice", "nonce": "27491480"}
Recieved request for symmetric key from: Alice nonce=27491480
Sending:  {"nonce1": "27491480", "nonce2": "71465962"}
```

MESSAGE 3 & 4

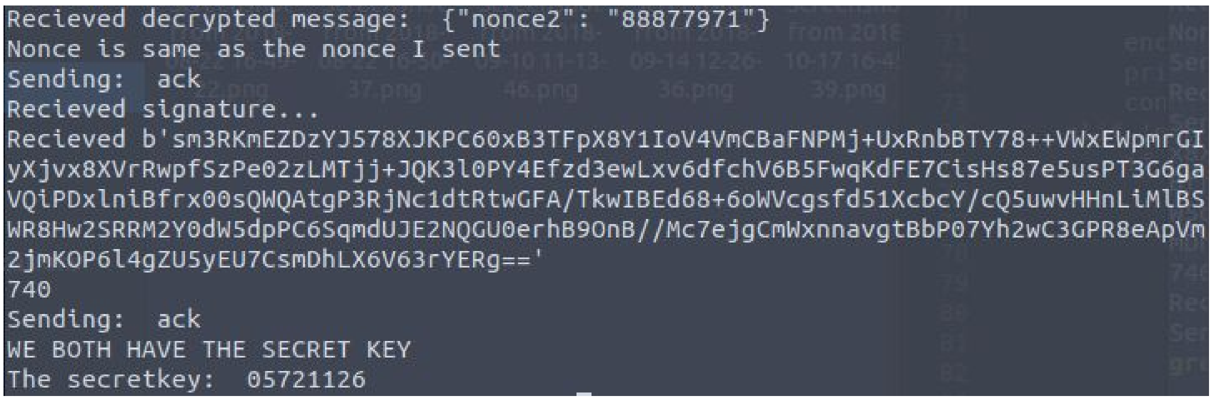
A returns *N*2, encrypted using B's public key, to assure B that its correspondent is A. A selects a secret key *Ks* and sends *M* = E(*PUb*, E(*PRa*, *Ks*)) to B. Encryption of this message with B's public key ensures that only B can read it; encryption
with A's private key ensures that only A could have sent it (signing it). B computes D(*PUa*, D(*PRb*, *M*)) to recover the secret key.

The signing step is a little different because of the way that the crypto RSA library works in python. Instead of sending a encrypted signed key. We first send the encrypted signature and then receive an acknowledgement and afterwards send the encrypted key. You however need both the key and the signature to verify the signature. Here is the output of alice.py sending message 3 (N2) and sending the signature followed by the sending the secret key (message 4).



```
Nonce is same as the nonce I sent
Sending:  {"nonce2": "71465962"}
Recieved acknowledgement
Sending:  b'IHyeRBFqccPDEtTI7QWELxz6vfRtpoG9T/ORB+Bza+/NRc5uK3/TJoZ3WUUyo67h7O/COtr/z
biGV4bUSzCJBgsNi965bK4ZsYcETc0V9pogk+e6ZmU05GXb5WcwtePU1s8FZc3zzaDv7xdsR9YammESIJYSgu
+pKrsWol9V6Yu3mqNcGvSqrgFRm1WEsJ1rkCpxoScmUp+qk+kGDhYTD7haMYSaO29V31bmuUCjtoz8wwqQI7S
G+mCVlgfSwqTQ+I2It5+ia4Fs0iGFTqnNDjkAhzsJzb7dmC/xYcx4HWjit/kXR7C7aEUlBkRjPD2SDEb4bbUU
goXv/lMT39oMeQ=='
785
Recieved acknowledgement
Sending:  41842022
```

Below is a screenshot of Bob receiving message 2, the signature, the secretkey and verifying the secretkey (message 3). When the verification returns true the output is "WE BOTH HAVE THE SECRET KEY" as seen in the screenshot.



```
Recieved decrypted message:  {"nonce2": "88877971"}
Nonce is same as the nonce I sent
Sending:  ack
Recieved signature...
Recieved b'sm3RKmEZDzYJ578XJKPC60xB3TFpX8Y1IoV4VmCBaFNPMj+UxRnbBTY78++VWxEWpmrGI
yXjvx8XVrRwpfSzPe02zLMTjj+JQK3l0PY4Efzd3ewLxv6dfchV6B5FwqKdFE7CisHs87e5usPT3G6ga
VQiPDxlniBfrx00sQWQAtgP3RjNc1dtRtwGFA/TkwIBEd68+6oWVcgsfd51XcbcY/cQ5uwvHHnLiMlBS
WR8Hw2SRRM2Y0dW5dpPC6SqmdUJE2NQGU0erhB9OnB//Mc7ejgCmWxnnavgtBbP07Yh2wC3GPR8eApVm
2jmKOP6l4gZU5yEU7CsmDhLX6V63rYERg=='
740
Sending:  ack
WE BOTH HAVE THE SECRET KEY
The secretkey:  05721126
```