Assignment 7. Asymmetric encryption

Crypto basics 1

Symmetric encryption advantages:

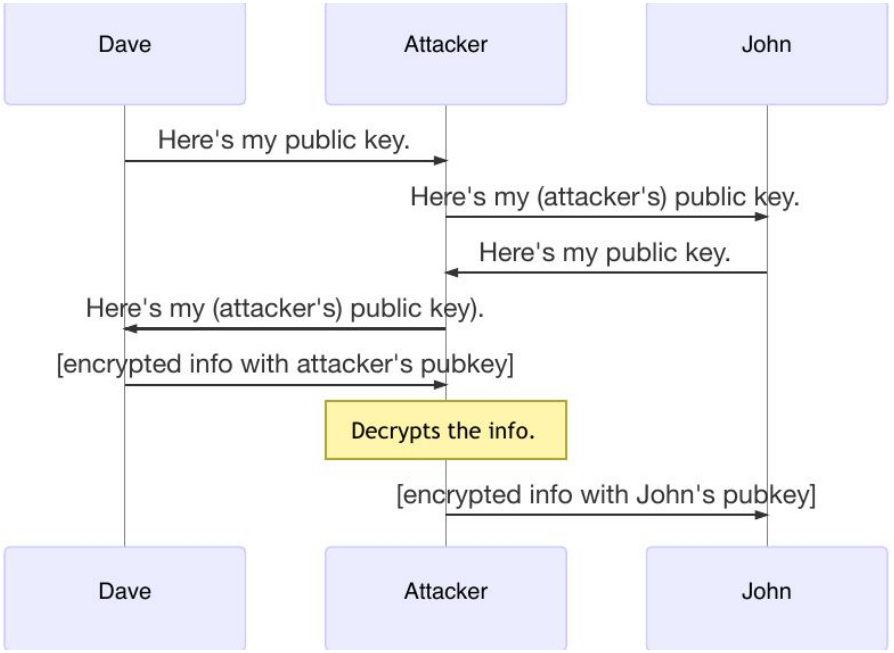
1. Performance on large inputs. Block ciphers (which symmetric ciphers usually are) require linear time with respect to input size, while asymmetic ciphers usually require a lot of computations for medium and large messages.
2. A different secret key is used for communication with every different party. If a key is compromised, only one channel starts to be insecure.

Asymmetric encryption advantages:

1. It allows message authentication.
2. It solves the problem of distributing the key for encryption.
3. A digitally signed message cannot be modified without invalidating the signature.

Crypto basics 2

We could execute a Man-in-the-Middle attack, assuming Dave and John exchange keys at the beginning of the session. It will go as follows:



Task 3
A good RNG is crucial for safe key generation in RSA. If we could compromise the RNG, we could derive the primes, used to generate the key, and this gives us the private key automatically (because RSA resides on hardness of factoring).

Examples:

Link to Wikipedia

Crypto basics 4

These two moduli have a common p, so calculating the gcd between them is enough.
p=10203801967042374235685280645002533035889748143620041545203837785813406324297888435282527410219630227951092551238279135374642983832916325943221010777029779
q1=11878237524775475196878100535651544663844036732241732441062854072852048704416336062649570806120219315147757407244428591007407698214764830194265368726450209
q2=10986465177048114626282811252238570277491143146008653030054666590003861689351980709991854057154755631106579071224698859499114676935534595284454540003310129

Code:

```
from math import gcd
n1=0xac995c273e44e927526687078ec617fad75a453183bbc32ddebfb3b0722d596df8ef2e1709669d5b9713bc4be23c9edf37cb744726b94c33c3db489cf14a9a42d05887fe4080f7afcc8afb245ba265deebcf6264af853cbfe13fd13930dd463dc3b6a252db4d41e426edb0c9ba9f19206865bec5773bba2dbc03ecb988405ef3

n2=0x9fa417de91002f7cb01eee83de402e2c84cbb0cfd45b388b762cfb3f0eba67f26273b4bf84a1e2bd08a831d49fb3c0c63d86512a0f2c931887ad4720f24980967c4fb82fa7e4aa96a90113c98019d383527f7f2290d085adbc8bb251319b7331b08af43ceee8acbddb277b174541495ba61865ab62cd4b023b7455d05d3ba623
p = gcd(n1, n2)
q1 = n1 // p
q2 = n2 // p
print(p)
print(q1)
print(q2)
```

Crypto basics 5

Outputs:
Private key:
(mpz(220369424400546429606663736519282905620048720683228220830629133441343630153120769780181821356779533676889260355386674751288925588513062010518661322131218434555049216449294525113539171192163999520717876698264636400576361082839786192440243924440461322091574099803985309060589214760893034705627707403323364170591212031834203005362836650550856055980910267963757755214568460233927389965842164233791000017462287435222890931954626711132089090736821841057852637271721701610873165608649614213758277248097678994701797080458195362870088572610169111200666320905404700669999088537421504026910504501691732167692513765582073537738 11)
Message:
CEk4f9n7ayDaLfkmWs+xyMK60xGA/SalOL1EKuPyFTQddcKupuY4/T1WC0D6l+0cUKD0AyE=
Code:

```
import base64
import binascii
import gmpy2
from math import gcd
n1=0xac995c273e44e927526687078ec617fad75a453183bbc32ddebfb3b0722d596df8ef2e1709669d5b9713bc4be23c9edf37cb744726b94c33c3db489cf14a9a42d05887fe4080f7afcc8afb245ba265deebcf6264af853cbfe13fd13930dd463dc3b6a252db4d41e426edb0c9ba9f19206865bec5773bba2dbc03ecb988405ef3
n2=0x9fa417de91002f7cb01eee83de402e2c84cbb0cfd45b388b762cfb3f0eba67f26273b4bf84a1e2bd08a831d49fb3c0c63d86512a0f2c931887ad4720f24980967c4fb82fa7e4aa96a90113c98019d383527f7f2290d085adbc8bb251319b7331b08af43ceee8acbddb277b174541495ba61865ab62cd4b023b7455d05d3ba623
p = gcd(n1, n2)
q1 = n1 // p
q2 = n2 // p
phi = (p - 1) * (q1 - 1)
e = 11
d = gmpy2.invert(e, phi)
m = int(binascii.hexlify(b"Daria"), 16)
c = pow(m, e, n1)
c_hex = hex(c)[2:]
c_hex = c_hex.zfill(len(c_hex) + len(c_hex) % 2)
c_hex = binascii.unhexlify(c_hex)
print("Private key: " + str((d, n1)))
print("Message: " + base64.b64encode(c_hex).decode("utf-8"))
```