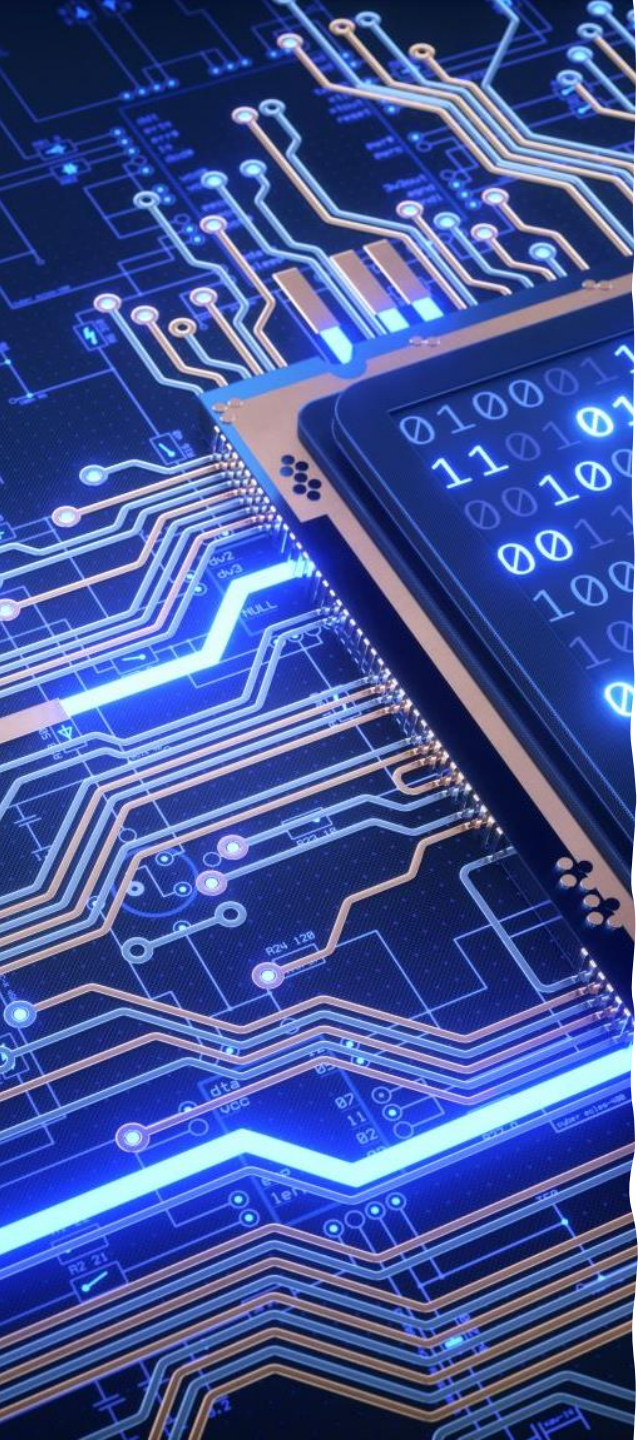


IoT-Cloud and Blockchain

RSA example

Dr. Phillip G. Bradford

University of Connecticut, Stamford CT. **USA**

phillip.bradford@uconn.edu

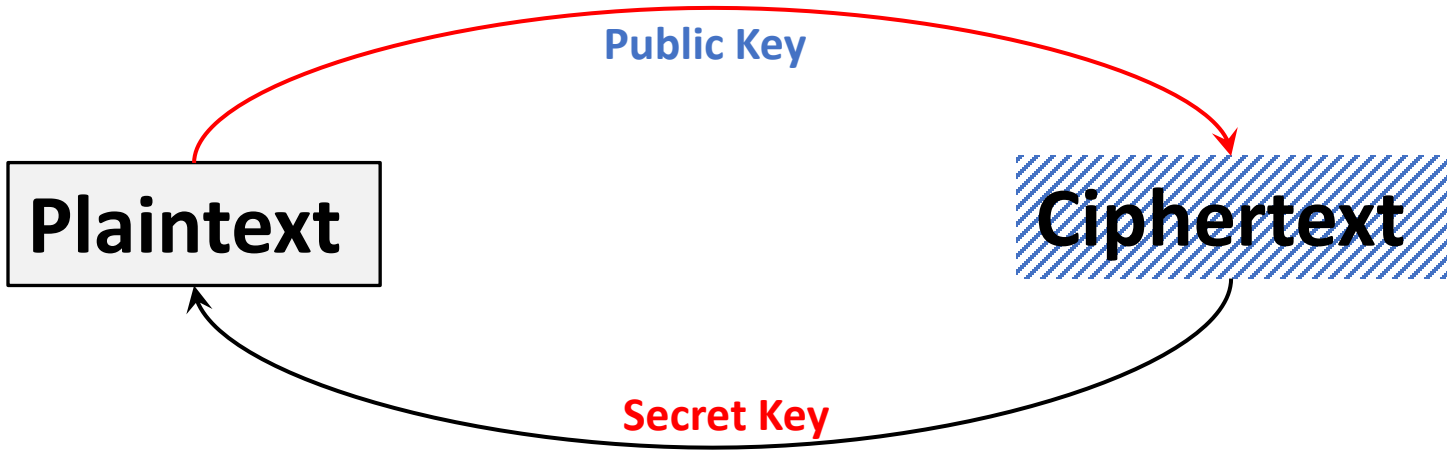
Outline

Example RSA

Code

Running

Public Key Systems: secret message



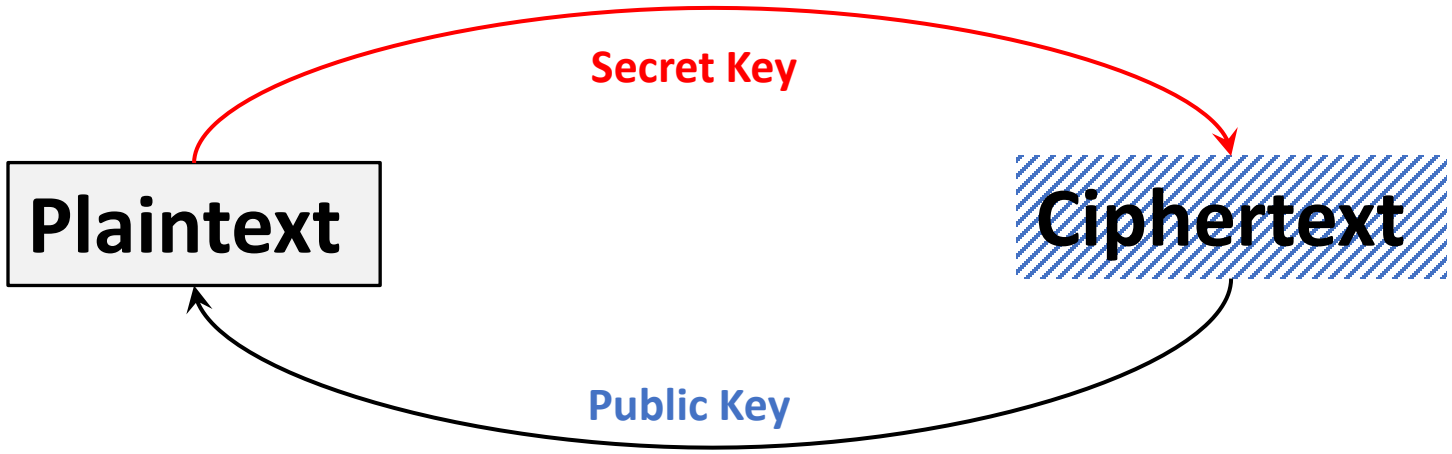
Generate two keys

- The public Key is open to the world
- The secret key is secret, **only** you know it

Knowing the public key, intractable to find secret key

Each key is the other's cryptographic inverse

Public Key Systems: proving secret key



Take today's news as plaintext

Encrypt it with secret key and post cipher text

Anyone with public key can get the plaintext

Public key systems

Proving you have a secret key

$$E_S[T] = C$$

$$D_P[C] = T$$

Letting anyone send you a secret message

$$E_P[T] = C$$

$$D_S[C] = T$$

Preparation

Python3

pip3 install Crypto

pip3 install pycryptodome

https://github.com/wonder-phil/BlockchainRPiExtras.git

Generating keys key_gen.py : PEM files

```
# From
# https://pycryptodome.readthedocs.io/en/latest/src/examples.html

key = RSA.generate(2048)
private_key = key.exportKey()
file_out = open("private.pem", "wb")
file_out.write(private_key)
file_out.close()

public_key = key.publickey().exportKey()
file_out = open("public.pem", "wb")
file_out.write(public_key)
file_out.close()
```

RSA Encrypt

```
# From https://pycryptodome.readthedocs.io/en/latest/src/examples.html

from Crypto.PublicKey import RSA
from Crypto.Cipher import PKCS1_OAEP

data = " The class is excellent and the students are brilliant! ".encode("utf-8")
file_out = open("encrypted_data.bin", "wb")

public_key = RSA.importKey(open(“public.pem").read())

cipher_rsa = PKCS1_OAEP.new(public_key)
enc_session_key = cipher_rsa.encrypt(data)

file_out.write(enc_session_key)
file_out.close()
```

RSA Decrypt

```
# From https://pycryptodome.readthedocs.io/en/latest/src/examples.html

from Crypto.PublicKey import RSA
from Crypto.Cipher import AES, PKCS1_OAEP

file_in = open("encrypted_data.bin", "rb")

private_key = RSA.importKey(open("private.pem").read())
cipher_text = file_in.read()

# Decrypt the session key with the private RSA key
cipher_rsa = PKCS1_OAEP.new(private_key)
plain_text = cipher_rsa.decrypt(cipher_text)

print(plain_text)
```