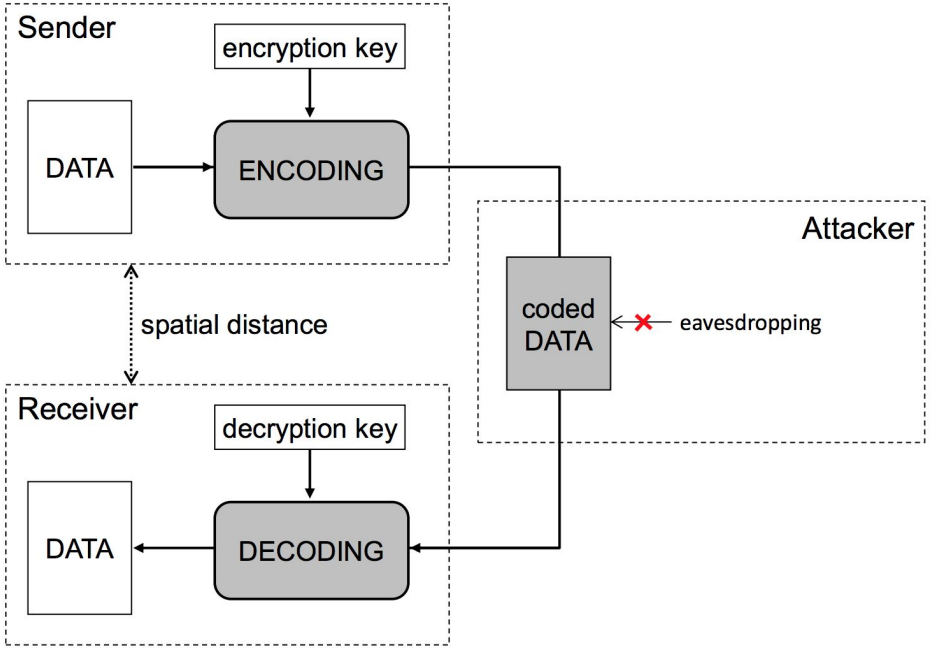
Blockchain

Nepal

Public Key Cryptography(1)

October 9, 2017

Public Key Cryptography



- Encryption & decryption keys are different
- Encryption key is (usually) public
- Decryption key is secret
- Security is based on some hard problems of mathematics

Terminology

Key-pair Generation: $G( ) = (K^+, K^-)$

K^+ - public key

K^- - private key

Encryption: $E(K^+, X) = Y$

X - plain text message

Y - cipher text message

Decryption: $D(K^-, Y) = X$

Examples: RSA, ElGamal

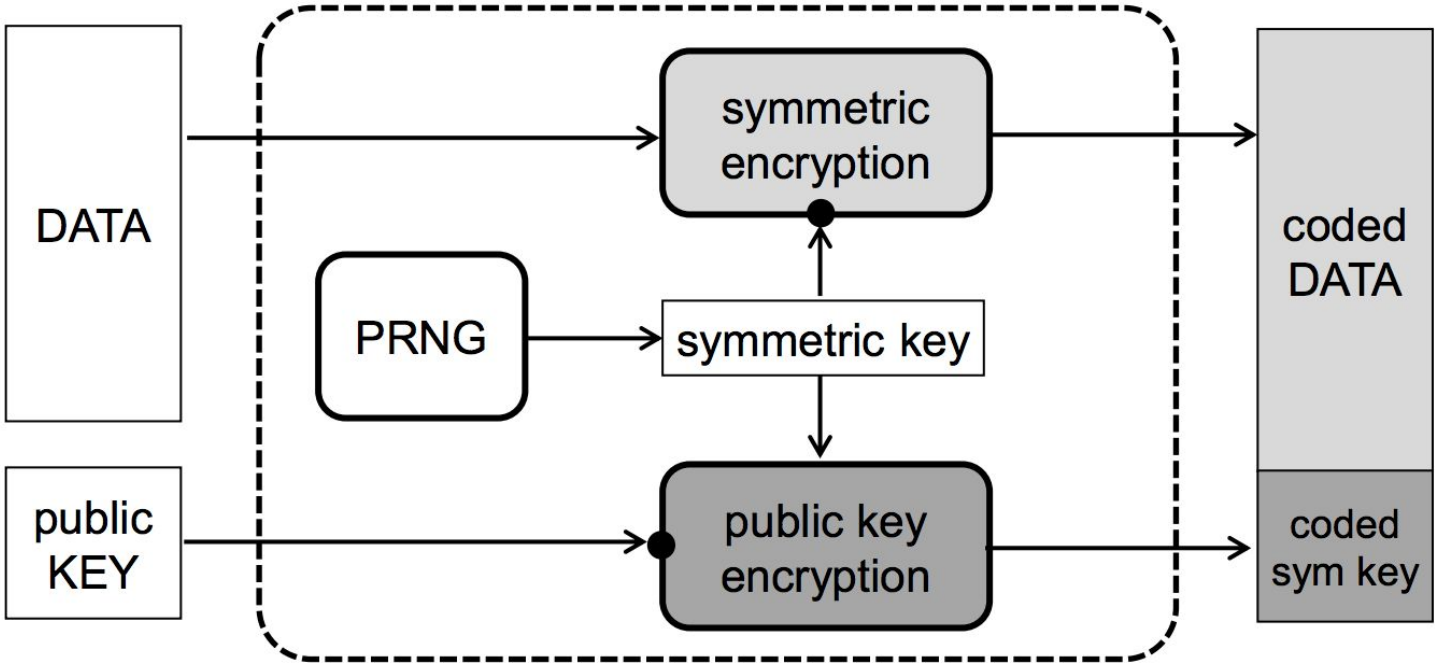
Security of Public-key Crypto Schemes

Security is based on problems which are believed to be hard to solve
➜ no polynomial time solutions exists

- Factoring
  given a positive integer N, find its prime factors

- Discrete logarithm
  given a prime p, a generator g of Z_p^* , and an element y in Z_p^* , find the integer x, $0<=x<=p-2$, such that $g^x \bmod p = y$

Digital Envelop



RSA (1/2)

Key-pair Generation Algorithm

- Choose two large primes **p** and **q**
- **n** = pq, $\phi(n) = (p-1)(q-1)$

- Choose **e**, such that $1 < e < \phi(n)$ and $\gcd(e, \phi(n)) = 1$
- Compute the inverse **d** of e mod $\phi(n)$,
  i.e., $ed \bmod \phi(n) = 1$ (easy if p and q are known)

- Output public key: **(e, n)**
- Output private key: **d**

RSA (2/2)

Encryption Algorithm

- Represent the plaintext message as an integer $m \in [0, n-1]$
- Compute the ciphertext $c = m^e \bmod n$

Decryption Algorithm

- Compute the plaintext from the ciphertext c as $m = c^d \bmod n$
- This works, as $c^d \bmod n = m^{ed} \bmod n$

$$= m^{k\Phi(n)+1} \bmod n$$
$$= m \bmod n$$
$$= m$$

Java - RSA Key Generation

```
int keySize = 2048;  // 1024, 2048, 3072, 4096

KeyPairGenerator keyPairGenerator = KeyPairGenerator.getInstance("RSA");

keyPairGenerator.initialize(keySize);

KeyPair keypair = keyPairGenerator.genKeyPair();


// Public key
PublicKey pubKey = keyPair.getPublic();

// Private key
PrivateKey privateKey = keyPair.getPrivate();
```

Java - RSA Encryption/Decryption

```
// Encryption

Cipher cipher = Cipher.getInstance("RSA");
cipher.init(Cipher.ENCRYPT_MODE, publicKey);
byte[] cipherText = cipher.doFinal(message.getBytes());
```

```
// Decryption

Cipher cipher = Cipher.getInstance("RSA");
cipher.init(Cipher.DECRYPT_MODE, privateKey);
byte[] plainText = cipher.doFinal(cipherText);
```

Exercise

Encrypt the content of the following file using RSA-1024 bit encryption
https://www.gutenberg.org/files/55693/55693-0.txt

Use hybrid encryption with AES 256 bit symmetric key to create a digital envelop of the file.