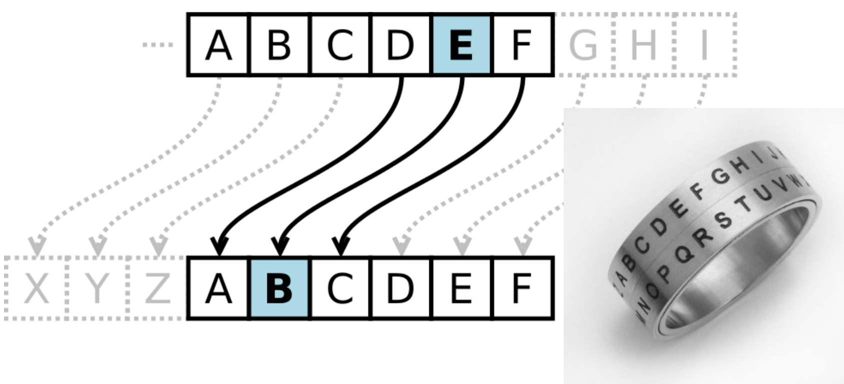
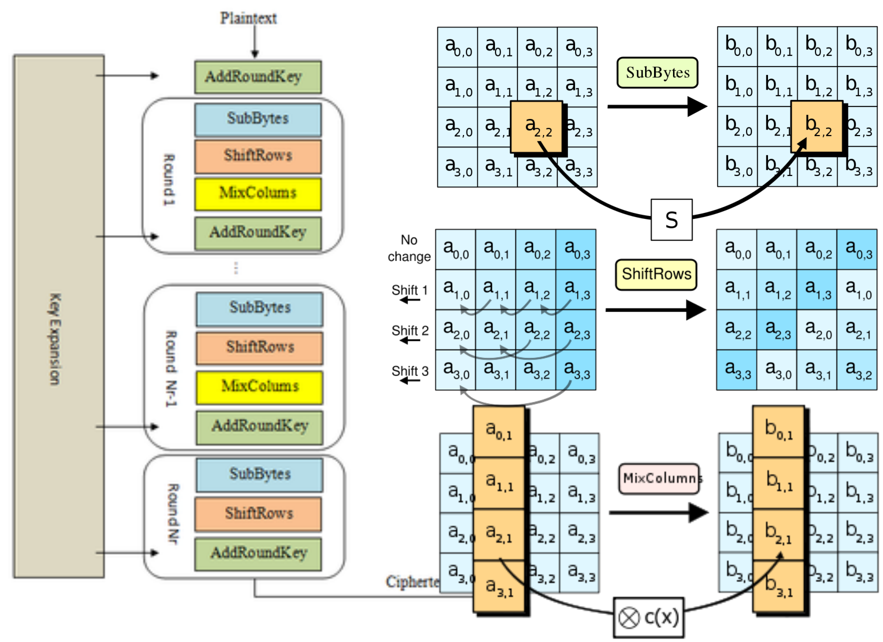
Lecture 9: Public-Key Cryptography

CS 181S

September 26, 2018

Crypto Thus Far…





Key pairs

- Instead of sharing a key between pairs of principals...
- ...every principal has a pair of keys
  - **public key:** published for the world to see
  - **private key:** kept secret and never shared

Protocol to exchange encrypted message

```
1. A:  c = Enc(m; pk_B)
2. A -> B:  c
3. B:  m = Dec(c; sk_B)
```

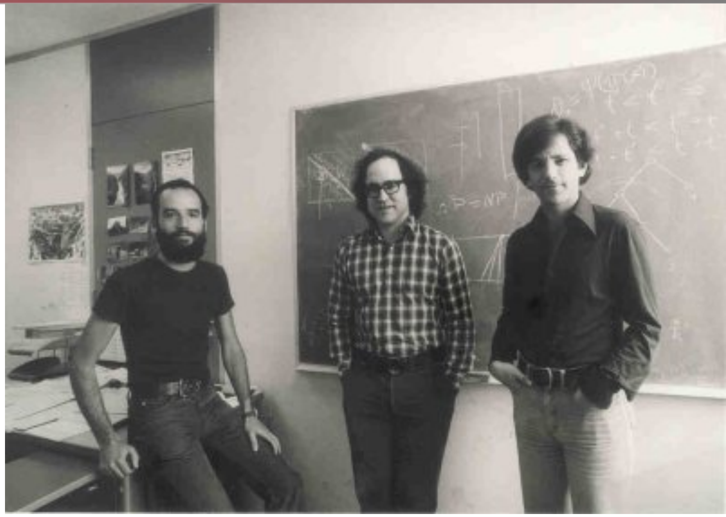
key pair: `(pk_B, sk_B)`

Public keys

```
0. B:   (K_B, k_B) = Gen(len)
1.   ...
```

- All public keys published in "phonebook"
- So A can lookup B's key to send message
- Length of phonebook is O(n)
- So quadratic problem reduced to linear!
- Eliminates key distribution problem!

RSA



[Rivest, Shamir, Adleman 1977]

Shared Turing Award in 2002: *ingenious contribution to making public-key crypto*

- Gen(len):
  - Pick primes p, q
  - Choose e, d such that $ed = 1 \bmod \mathrm{lcm}(p-1, q-1)$
  - $pk = (n, e), \;\; sk = (p, q, d)$
- Enc(m, pk)

$$c = m^e \bmod n$$

- Dec(c, sk):

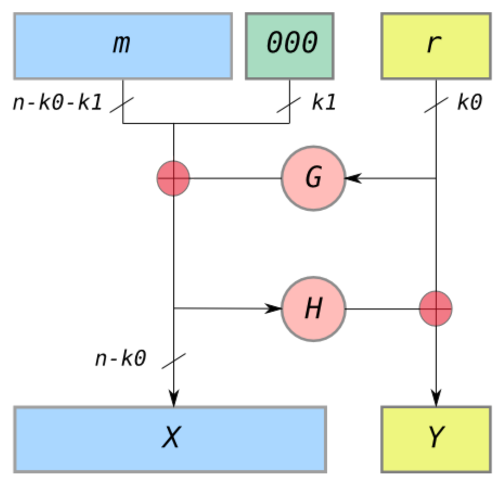
$$m = c^d \bmod n$$

Problems with Textbook RSA

- *Deterministic*: given same plaintext and key, always produces the same ciphertext
- *Small numbers*: if m^e < n, then log is easy to compute
- *Big numbers*: if m > n, can't compute do math mod n

Solution 1: Padding

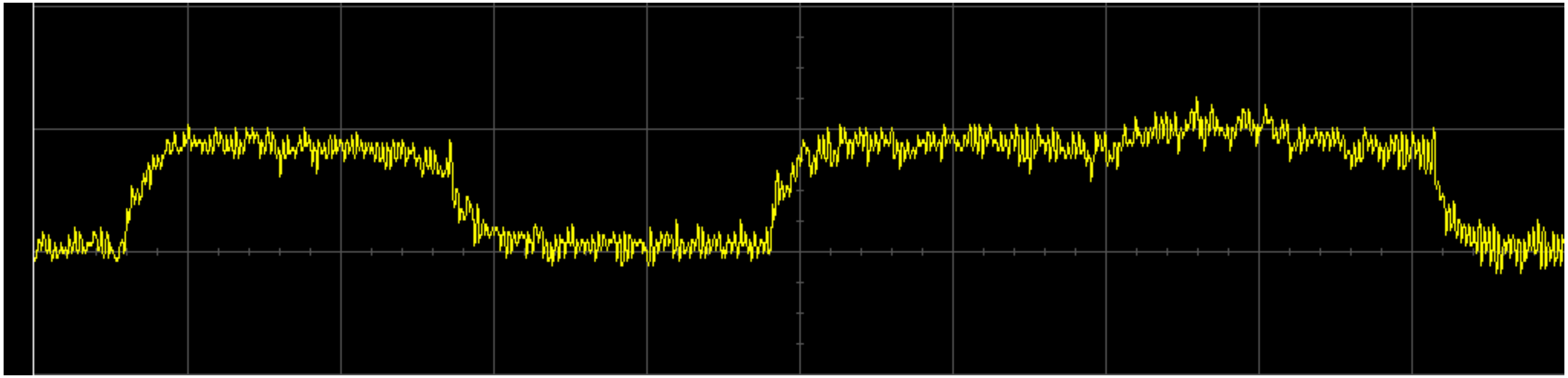
- PKCS#1 v1.5: 0x00 0x02 [non-zero bytes] 0x00 [message]
  - Vulnerable to a padding oracle attack!
- OAEP (Optimal Asymmetric Encryption Padding)
  - Security proof (with assumptions)



Square-and-Multiply

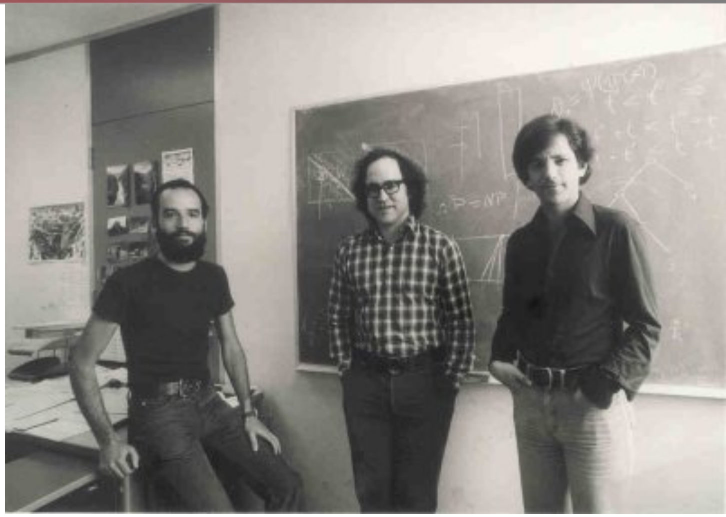
```
res = 1;
while (exp > 0) {
        if (exp % 2 == 1){
                res = res * base % p;
        }
        base = base^2 % p;
        exp >> 1;
}
return res;
```

Side Channels



- Power
- Timing
- EM Radiation
- Acoustics

Blinded RSA



[Rivest, Shamir, Adleman 1977]

Shared Turing Award in 2002: *ingenious contribution to making public-key crypto*

- Gen(len):
  - Pick primes p, q
  - Choose e, d such that $ed = 1 \bmod \mathrm{lcm}(p-1, q-1)$
  - $pk = (n, e), \;\; sk = (p, q, d)$
- Enc(m, pk)

$$c = (mr)^e \cdot r^{-e} \bmod n$$

- Dec(c, sk):

$$m = c^d \bmod n$$

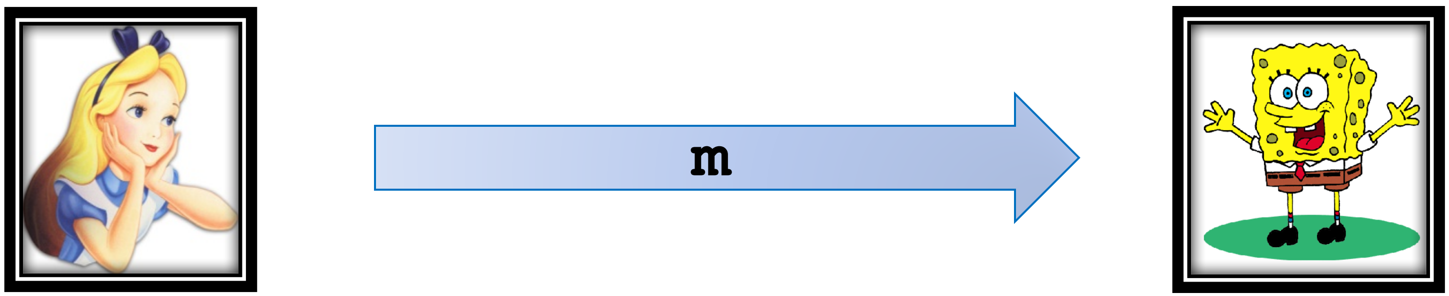
Solution 2: Hybrid encryption

- Assume:
  - Symmetric encryption scheme (Gen_SE, Enc_SE, Dec_SE)
  - Public-key encryption scheme (Gen_PKE, Enc_PKE, Dec_PKE)

- Use public-key encryption to establish a shared session key
  - Avoids quadratic problem, assuming existence of phonebook
  - Avoids problem of key distribution

- Use symmetric encryption to exchange long plaintext encrypted under session key
  - Gain efficiency of block cipher and mode

Protocol to exchange encrypted message




```
0.   B: (pk_B, sk_B) = Gen_PKE(len_PKE)
        publish (B, pk_B)
1.   A: k_s = Gen_SE(len_SE)
        c1 = Enc_PKE(k_s; pk_B)
        c2 = Enc_SE(m; k_s)
2.   A -> B: c1, c2
3.   B: k_s = Dec_PKE(c1; sk_B)
        m = Dec_SE(c2; k_s)
```

Session keys

- If key compromised, only those messages encrypted under it are disclosed
- Used for a brief period then discarded
  - cryptoperiod*:* length of time for which key is valid
  - in this case, for a single (long) message
  - not intended for reuse in future messages
- only intended for unidirectional usage:
  - A->B, not B->A

DIGITAL SIGNATURES

Recall: Key pairs

- Instead of sharing a key between pairs of principals...
- ...every principal has a pair of keys
  - **public key:** published for the world to see
  - **private key:** kept secret and never shared

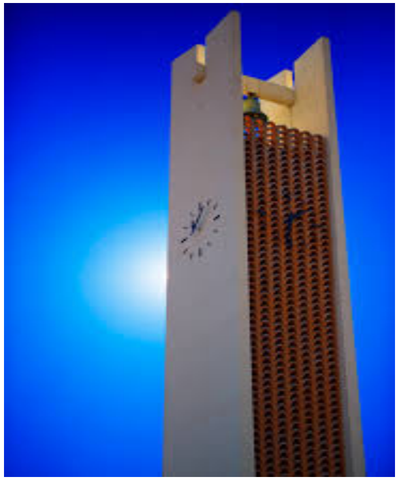
Key pair terminology

	Encryption	Digital Signatures
Public key	Encryption key	Verification key
Private key	Decryption key	Signing key

Digital signature scheme

A digital signature scheme is a triple (Gen, Sign, Ver):

- Gen(len): generate a key pair (pk,sk) of length len
- Sign(m; sk): sign message m with key sk, producing signature s as output
- Ver(m, s; sk): verify signature s on message m with key pk



Protocol to exchange signed message

```
0. A: (K_A,k_A) = Gen(len)
1. A: s = Sign(m; k_A)
2. A -> B: m, s
3. B: accept if Ver(m; s; K_A)
```

- Message is sent in plaintext: no protection of confidentiality
- Goal is to detect modification **not** prevent

Security of digital signatures

- Must be hard to forge signature for a message without knowledge of key

  ...like handwritten signatures

- Even if in possession of multiple (message, signature) pairs for that key

  ...unlike handwritten signatures

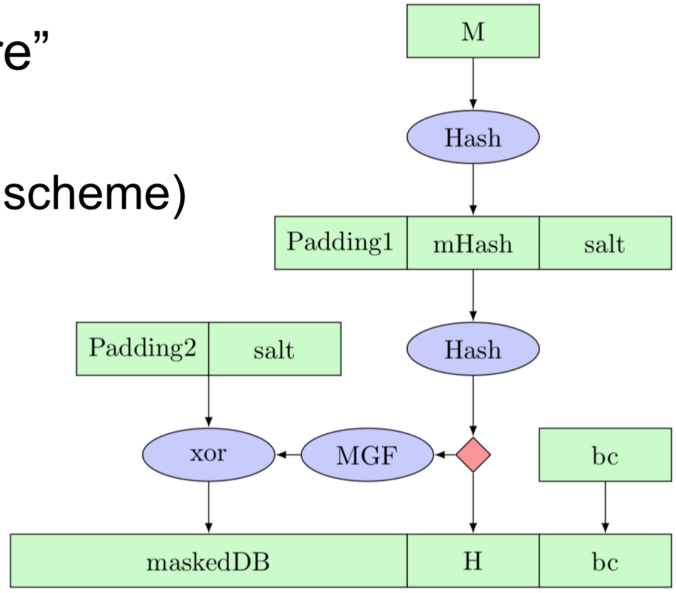
DSA

DSA: Digital Signature Algorithm [Kravitz 1991]

- Standardized by NIST and made available royalty-free in 1991/1993
- Used for decades without any serious attacks
- Closely related to Elgamal encryption

RSA

- Core ideas are the same as RSA encryption
- Common mistake: “RSA sign = encrypt with private key”
- Truth (in real world, outside of textbooks):
  - there's a core RSA function R that works with either pk or sk
  - RSA encrypt = do some prep work on m then call R with pk
  - RSA sign = do **different** prep work on m then call R with sk
  - Prep work: recall “textbook RSA is insecure”
    - (For encryption: OAEP)
    - For signatures: PSS (probabilistic signature scheme)
  - Also need to handle long messages…



Signatures with hashing

```
1. A: s = Sign(H(m); k_A)
2. A -> B: m, s
3. B: accept if Ver(H(m); s; K_A)
```

Blind signatures

[Chaum 1983]

- Purpose: signer doesn't know what they are signing
- Two additional algorithms: Blind and Unblind
- Unblind(Sign(Blind(m); k)) = Sign(m; k)
- Uses: e-cash, e-voting

Group signatures

[Chaum and van Heyst 1991]

- Purpose: one member of group signs anonymously on behalf of group
- Introduces a *group manager* who controls membership
- Two new protocols: Join and Revoke, to manage membership
- One new algorithm: Open, which manager can run to reveal who signed a message