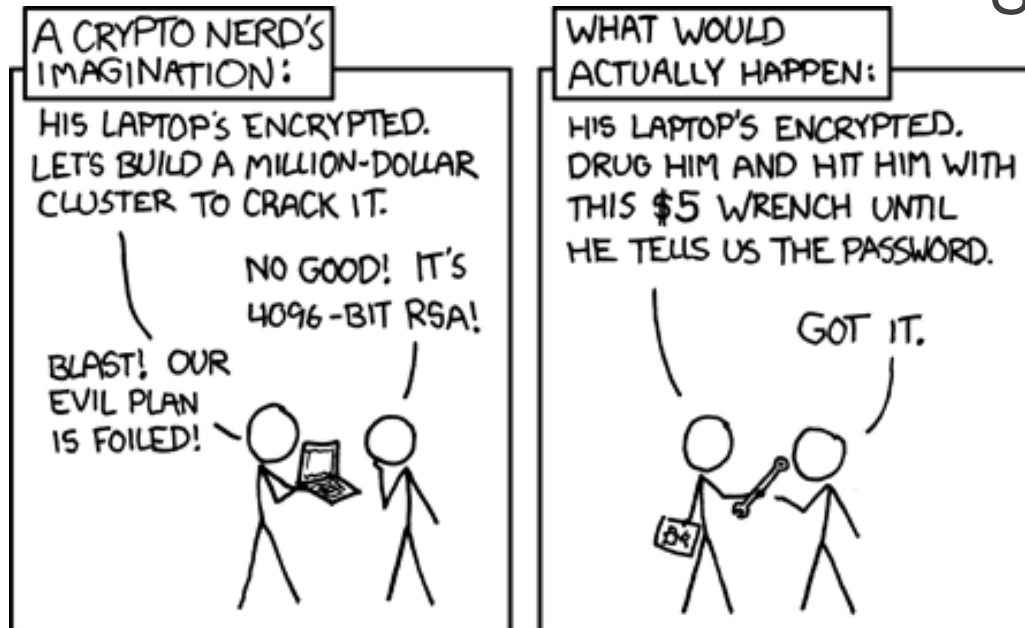


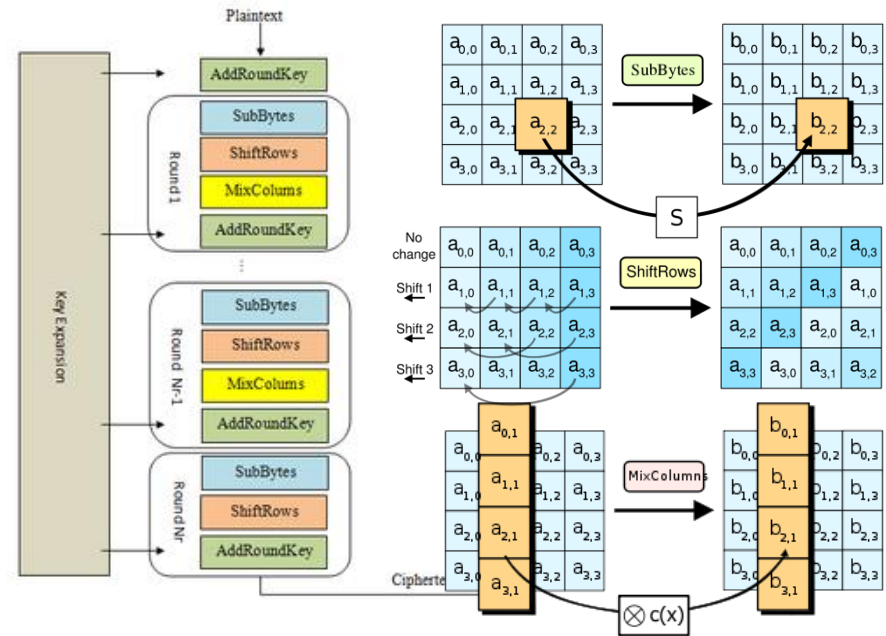
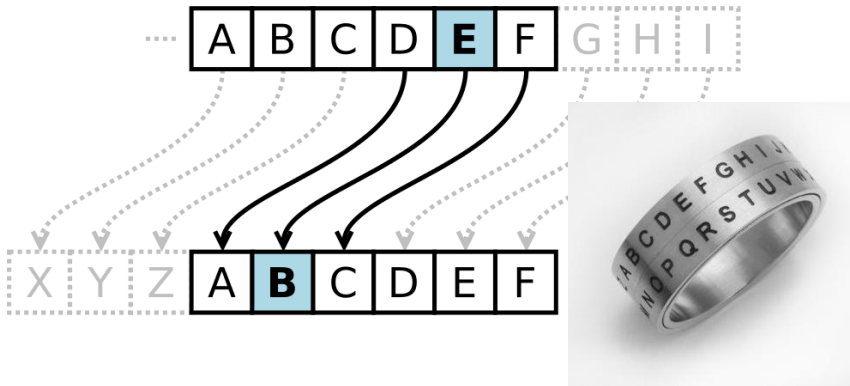
Lecture 7: Public-Key Encryption

CS 138

Spring 2026



Crypto Thus Far...



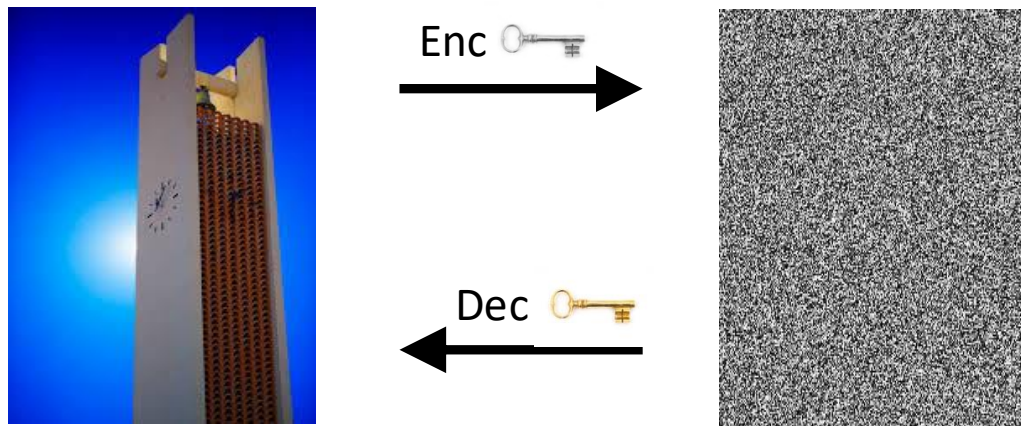
Key pairs

- 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



(Public-Key) Encryption algorithms

- $\text{Gen}(1^n)$: generate a **keypair** (pk, sk) of length n
- $\text{Enc}(m; pk)$: encrypt **message** under public key pk
- $\text{Dec}(c; sk)$: decrypt **ciphertext** c with secret key sk



(Gen, Enc, Dec) is a public-key **encryption scheme** aka **cryptosystem**

RSA

[Rivest, Shamir, Adleman 1977]

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

- Gen(len):
 - Pick primes p, q , define $n = p \cdot q$
 - Choose e, d such that $ed = 1 \pmod{(p-1)(q-1)}$
 - $pk = (n, e)$, $sk = (p, q, d)$

- Enc(m , pk)

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

- Dec(c , sk):

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



Exercise 1: RSA

- Let $pk = (n, e) = (21, 5)$ and $sk = (p, q, d) = (3, 7, 5)$
- Observe that $ed = 5 \cdot 5 = 25 = 1 \pmod{12}$

1. Compute $c = \text{Enc}(17; pk)$

2. Compute $\text{Dec}(c; sk)$

RSA

- Theorem: RSA is a correct public-key encryption scheme.
- Theorem: $(m^e \bmod pq)^d \bmod pq == m$

$$\begin{aligned}\text{Dec}(\text{Enc}(m; pk); sk) &= (m^e \bmod pq)^d \bmod pq \\ &= (m^e)^d \bmod pq \\ &= m^{ed} \bmod pq \\ &= m \bmod pq\end{aligned}$$

$$\begin{aligned}m^{ed} \bmod p &= m^{1+k(p-1)(q-1)} \bmod p \\ &= m^1 \cdot (m^{p-1})^{k(q-1)} \bmod p \\ &= m \cdot (m^{p-1} \bmod p)^{k(q-1)} \bmod p \\ &= m \cdot (1)^{k(q-1)} \bmod p \\ &= m \bmod p\end{aligned}$$

$$\begin{aligned}m^{ed} \bmod q &= m^{1+k(p-1)(q-1)} \bmod q \\ &= m^1 \cdot (m^{q-1})^{k(p-1)} \bmod q \\ &= m \cdot (m^{q-1} \bmod q)^{k(p-1)} \bmod q \\ &= m \cdot (1)^{k(p-1)} \bmod q \\ &= m \bmod q\end{aligned}$$

RSA

- ~~• **Theorem:** RSA is a secure public-key encryption scheme.~~
- ~~• **Theorem:** If factoring is hard, then RSA is a secure public-key encryption scheme.~~



Problems with Textbook RSA

- *Insecure keys*: There are known efficient attacks for some choices of e , d , p , q

- *pq is too small*: $n = pq$ can be factored

make sure n is 2048 bits

- *small values of e* : if you send the same message with multiple keys, plaintext message can be recovered
- *small values of e* : if $m^e < n$ can compute log efficiently
- *small values of d* : Boneh-Durfee attack for $d < n^{0.292}$

$e = 65537$

- *p or q is shared with other key*: can compute gcd to factor

use high-quality randomness

- *p similar to q* : $n = pq$ can be efficiently factored
- *p or q are bad primes*: $n = pq$ can be efficiently factored
- *$\gcd(e, \text{lcm}(p-1, q-1)) > 1$* :

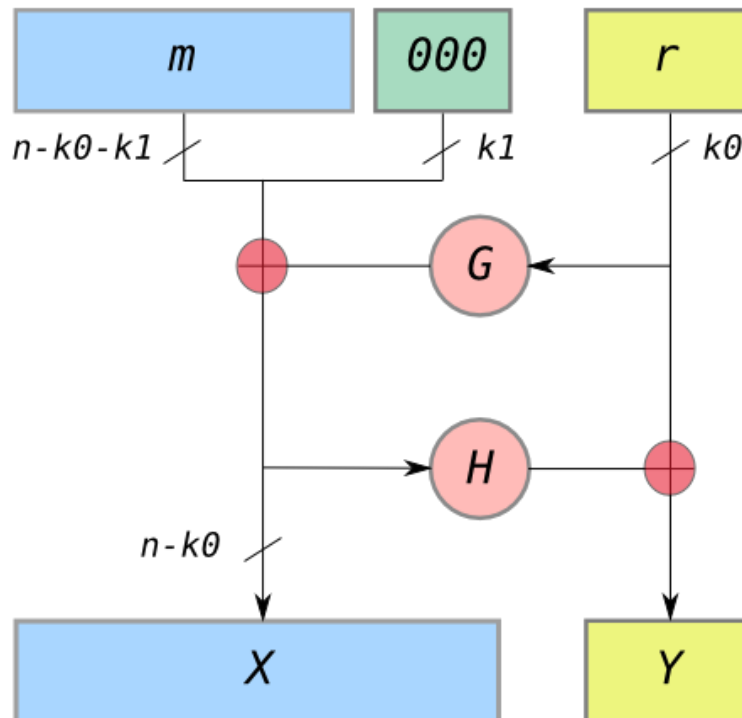
use a good library, don't implement RSA yourself!

Problems with Textbook RSA

- *Insecure keys*: There are known efficient attacks for some choices of e , d , p , q
- *Deterministic*: given same plaintext and key, always produces the same ciphertext

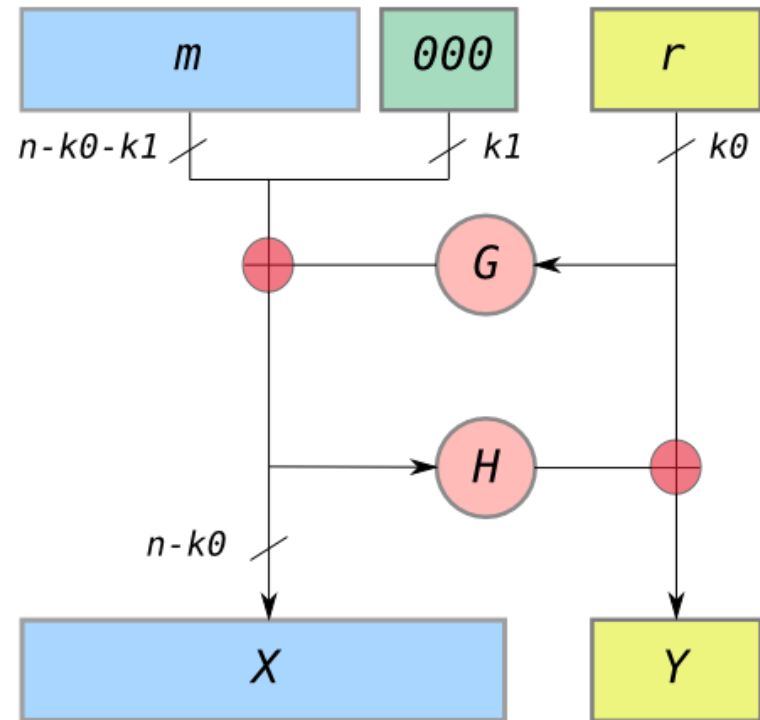
Solution 1: Padding

- 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)



Exercise 2: OAEP

- Define a function in C to compute m given values X and Y , bitlengths k_0 and k_1 , and hash functions G and H



Problems with Textbook RSA

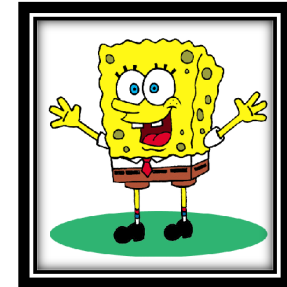
- *Insecure keys*: There are known efficient attacks for some choices of e , d , p , q
- *Deterministic*: given same plaintext and key, always produces the same ciphertext
- *Big numbers*: if $m > n$, can't compute $m \bmod n$

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: $(pk_B, sk_B) = \text{Gen_PKE}(\text{len_PKE})$
publish (B, pk_B)
1. A: $k_s = \text{Gen_SE}(\text{len_SE})$
 $c1 = \text{Enc_PKE}(k_s; pk_B)$
 $c2 = \text{Enc_SE}(m; k_s)$
2. A \rightarrow B: $c1, c2$
3. B: $k_s = \text{Dec_PKE}(c1; sk_B)$
 $m = \text{Dec_SE}(c2; k_s)$

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

Problems with Textbook RSA

- *Insecure keys*: There are known efficient attacks for some choices of e , d , p , q
- *Deterministic*: given same plaintext and key, always produces the same ciphertext
- *Big numbers*: if $m > n$, can't compute $m \bmod n$
- *Side channel attacks*: interfaces can leak information about secret key

Square-and-Multiply

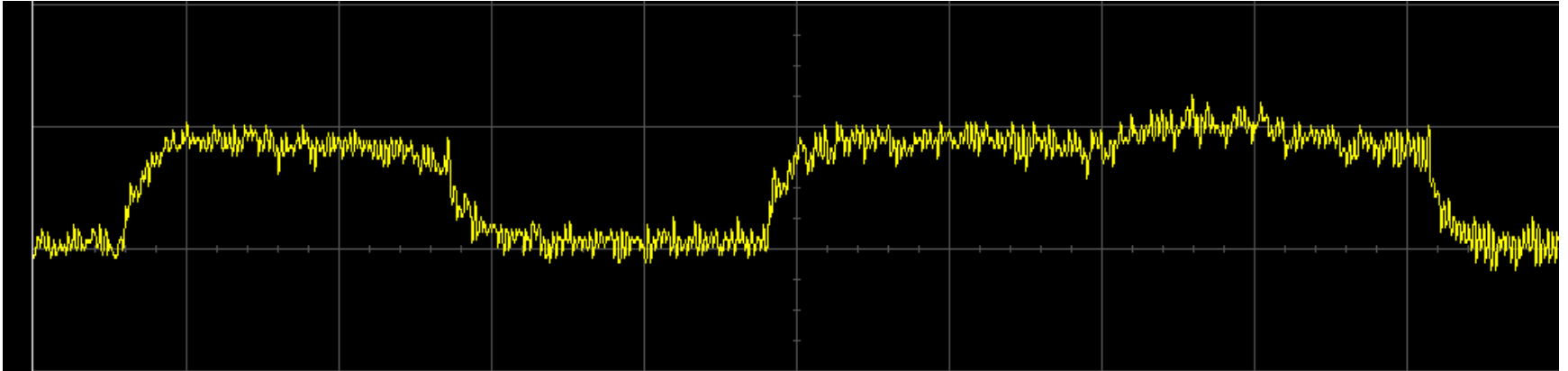
```
int modular_exp(x, n, p) {
    res = 1;
    while (n > 0) {
        if (n % 2 == 1) {
            res = res * x % p;
        }
        x = x^2 % p;
        n >> 1;
    }
    return res;
}
```

Exercise 3: Square-and Multiply

- Compute $3^5 \bmod 21$ using square and multiply

```
int modular_exp(x, n, p) {
    res = 1;
    while (n > 0) {
        if (n % 2 == 1) {
            res = res * x % p;
        }
        x = x^2 % p;
        n >> 1;
    }
    return res;
}
```

Side Channels



- Power
- Timing
- EM Radiation
- Acoustics

Solution 3: 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 $ed = 1 \pmod{\text{lcm}(p-1, q-1)}$
 - $pk = (n, e)$, $sk = (p, q, d)$

- Enc(m , pk)

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

- Dec(c , sk):

$$m = ((cr)^d \pmod n) \cdot r^{-d}$$

Problems with Textbook RSA

- *Insecure keys*: There are known efficient attacks for some choices of e , d , p , q
- *Deterministic*: given same plaintext and key, always produces the same ciphertext
- *Big numbers*: if $m > n$, can't compute $m \bmod n$
- *Side channel attacks*: interfaces can leak information about secret key
- *Key Management*: secure storage and authentication

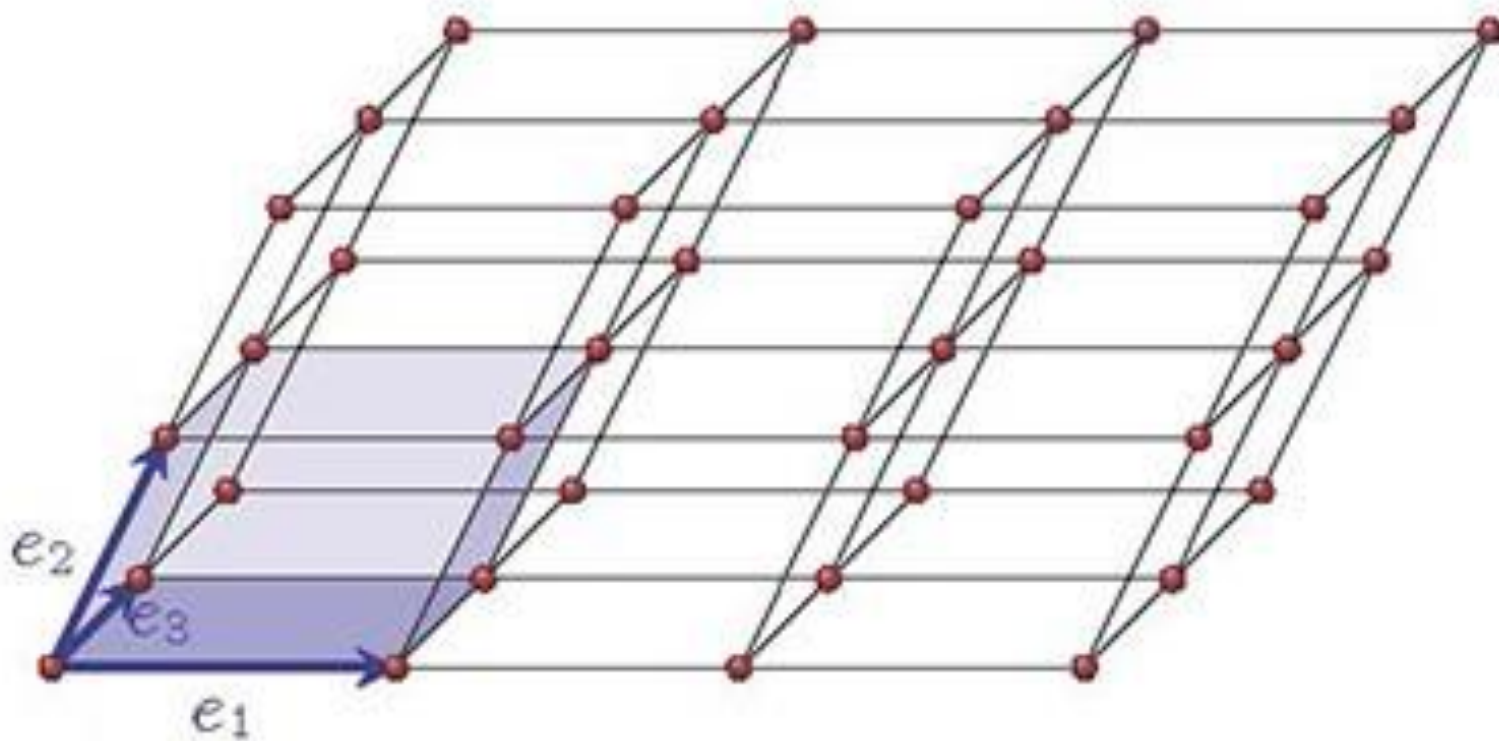
Solution 4: Key Management

- Store private keys offline
- Store private keys in protected files
- PKI infrastructure to authenticate public keys
- Use another solution
 - PBE: Memorize the keys (sort of)
 - DH: Generate shared keys on the fly

Problems with Textbook RSA

- *Insecure keys*: There are known efficient attacks for some choices of e , d , p , q
- *Deterministic*: given same plaintext and key, always produces the same ciphertext
- *Big numbers*: if $m > n$, can't compute $m \bmod n$
- *Side channel attacks*: interfaces can leak information about secret key
- *Key Management*: key storage and authentication
- *Quantum Computers*: provably breakable with different hardware

Solution 5: Post-Quantum Cryptography



Public Key Encryption

