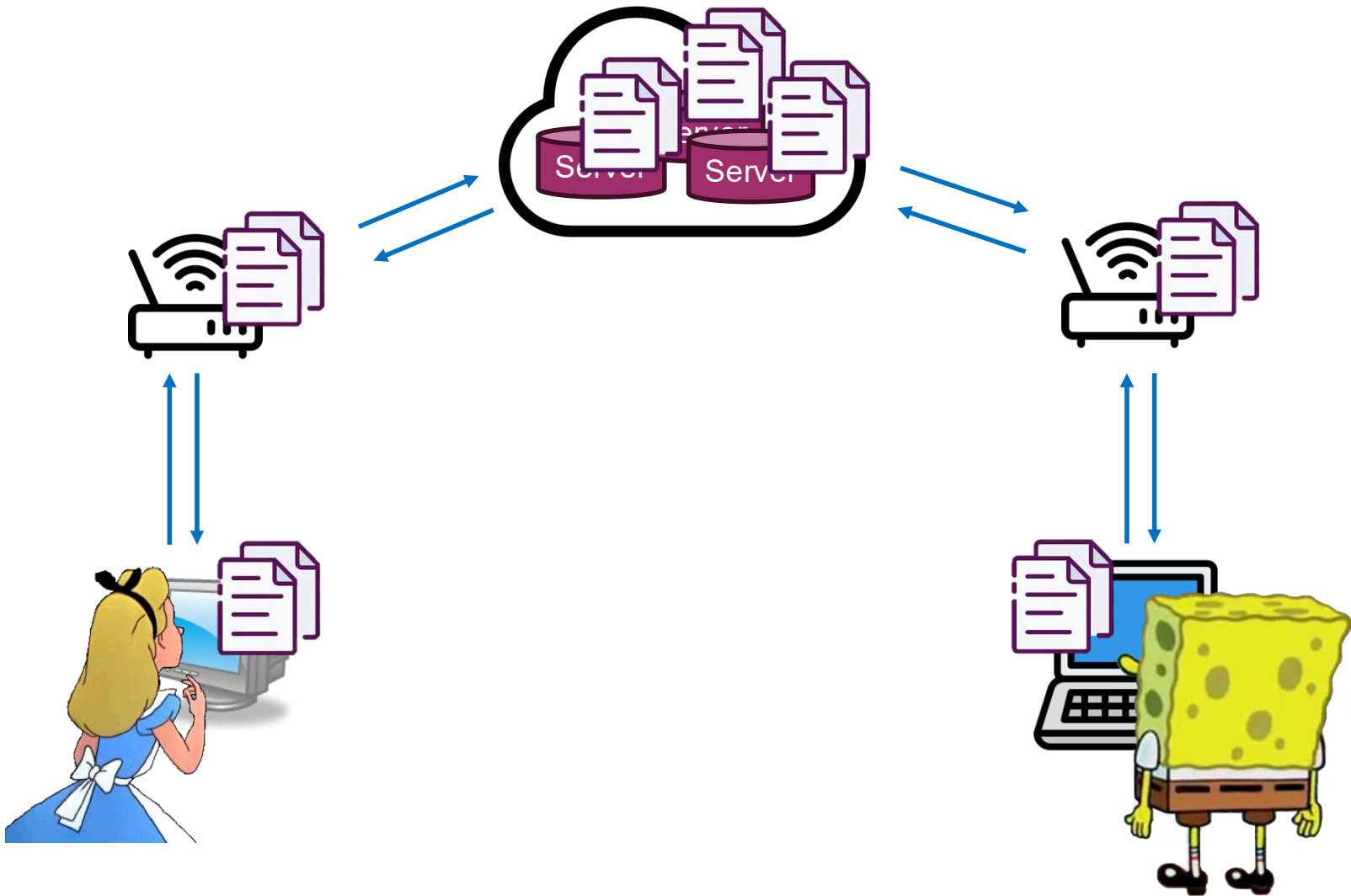


Lecture 7: Encrypted Systems

CS 181E

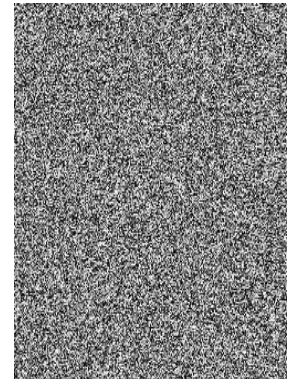
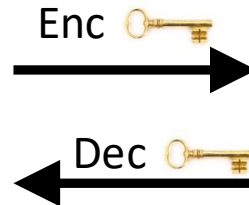
Fall 2026

Review: Computer Systems



Review: (Symmetric) Encryption

- $\text{Gen}(1^n)$: generate a **key** of length n
- $\text{Enc}(m; k)$: encrypt **message** under key k
- $\text{Dec}(m; k)$: decrypt **ciphertext** c with key k
- An encryption scheme is **correct** if $\text{Dec}(\text{Enc}(m; k); k) == m$



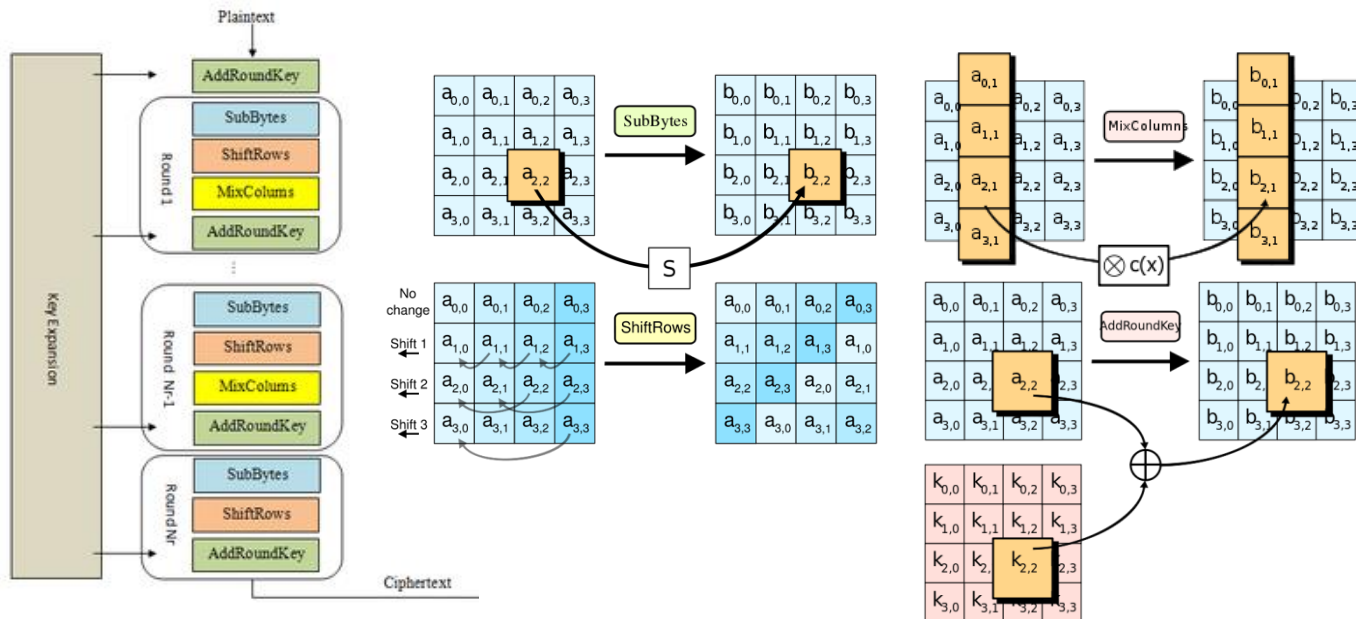
- A crypto scheme is **secure** if

$$\forall \text{PPT } A, \exists \delta \in O\left(\frac{1}{2^n}\right) \text{ s.t. } \forall n, \forall m, m' \text{ s.t. } |m| = |m'| = n,$$

$$\Pr[A(\text{Enc}(m; k)) = m] \leq \Pr[A(\text{Enc}(m'; k)) = m] + \delta(n)$$

Example: AES

- Advanced Encryption Standard (AES) result of 2001 NIST competition
- **Block cipher** that operates on fixed-size messages
- Currently no known practical attacks, approved by NSA for top-secret
- $\text{Gen}(1^n) :=$ generate a random bitstring of length n

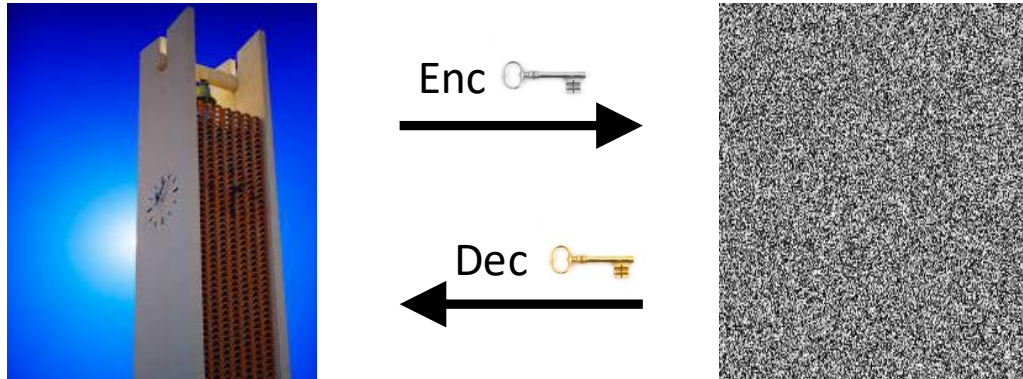


Review: Encryption for Privacy

- people with sk can read data (**correctness**)
- people without sk can't read data (**security**)
- so can control access by controlling who gets sk
- to distribute keys, we can use more crypto!

(Public-Key) Encryption algorithms

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



- A crypto scheme is **secure** if

$$\forall \text{PPT } A, \exists \delta \in O\left(\frac{1}{2^n}\right) \text{ s.t. } \forall n, \forall m, m' \text{ s.t. } |m| = |m'| = n,$$

$$\Pr[A(\text{Enc}(m; pk)) = m] \leq \Pr[A(\text{Enc}(m'; pk)) = m] + \delta(n)$$

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 $ed = 1 \bmod (p - 1)(q - 1)$
 - $pk = (n, e)$, $sk = (p, q, d)$

- Enc(m , pk)

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

- Dec(c , sk):

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



Exercise: RSA

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

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

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

RSA is correct

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

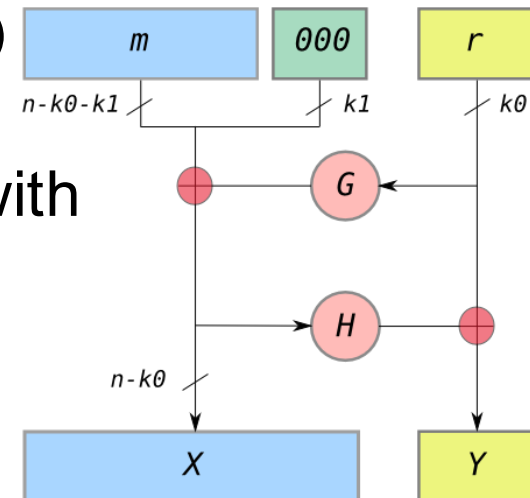
$$\begin{aligned}\text{Dec}(\text{Enc}(m; pk); sk) &= (m^e \bmod pq)^d \bmod pq \\ &= (m^e)^d \bmod pq \\ &= m^{ed} \bmod pq \\ &= m \bmod pq\end{aligned}$$

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

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

Problems with Textbook RSA

- *Insecure keys*: There are known efficient attacks for some choices of e , d , p , q
 - use good crypto library, never implement yourself
- *Deterministic*: given same plaintext and key, always produces the same ciphertext
 - OAEP (Optimal Asymmetric Encryption Padding)
- *Inefficient*: computing modular arithmetic with large numbers is very slow
 - only use pke to share/agree on key



Option 1: Hybrid Encryption



Symmetric Encryption

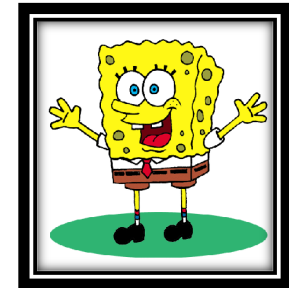
- shared secret keys
- e.g., AES
- Pros: fast, works on arbitrary-length messages
- Cons: need n^2 keys, key distribution

Public-key Encryption

- keypair (sk, pk) per person
- e.g., RSA
- Pros: don't need secret enc
- Cons: (very!) slow, key authentication

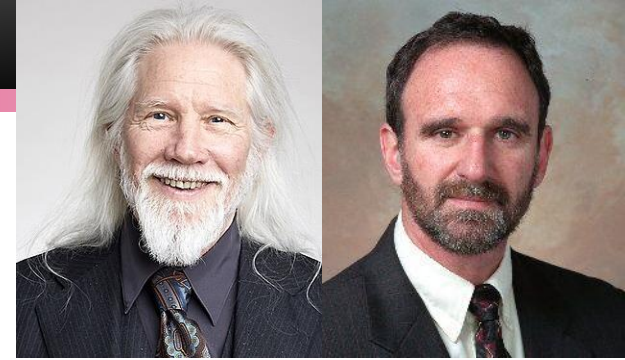
Hybrid Encryption: generate fresh AES key each time, use other person's public key to encrypt AES session key and send it to them

Option 1: Hybrid Encryption



0. B: $(pk_B, sk_B) = \text{Gen_PKE}(\text{len_PKE})$
publish (B, pk_B)
1. A: $k_s = \text{Gen_SKE}(\text{len_SKE})$
 $c1 = \text{Enc_PKE}(k_s; pk_B)$
 $c2 = \text{Enc_SKE}(m; k_s)$
2. A $\rightarrow$ B: $c1, c2$
3. B: $k_s = \text{Dec_PKE}(c1; sk_B)$
 $m = \text{Dec_SKE}(c2; k_s)$

Option 2: Key Agreement



[Diffie and Hellman, 1976]

Shared Turing Award in 2015: *fundamental contributions to modern cryptography*

- Gen(len):
 - Agree on integer n , finite cyclic group G of order n , and generator g
 - Agree a prime p and value $1 < g < p$
- Person 1:
 - Pick value $1 < a < p$
 - compute $x = g^a \bmod p$
 - send x to Person 2
- Person 2:
 - Pick value $1 < b < p$
 - compute $y = g^b \bmod p$
 - sent y to Person 1
- compute $k = y^a \bmod p$
- compute $k = x^b \bmod p$

Exercise: DH Key Agreement

- Assume $g=4$ and $p = 47$
- Assume Alice chooses $a=13$ and sends $x=43$ to Bob
- Assume that Bob then chooses $b=21$

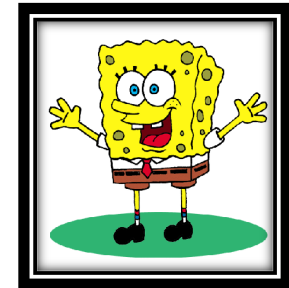
1. What value y will Bob send to Alice?
2. What secret key will be generated by Bob?
3. What secret key will be generated by Alice?

DH is Correct

- Theorem: Diffie-Hellman key-agreement always results in Alice and Bob ending up with the same key.

$$\begin{aligned}k_{Alice} &= y^a \bmod p \\&= (g^b \bmod p)^a \bmod p \\&= (g^b)^a \bmod p \\&= g^{b \cdot a} \bmod p \\&= (g^a)^b \bmod p \\&= (g^a \bmod p)^b \bmod p \\&= x^b \bmod p \\&= k_{Bob}\end{aligned}$$

Option 2: Key Agreement



0. B: pick $p, g, y = g^b \bmod p$
publish (g, p, y)
1. A: pick $a, x = g^a \bmod p$
 $k_s = y^a \bmod p$
 $c = \text{Enc_SKE}(m; k_s)$
2. A $\rightarrow$ B: x, c
3. B: $k_s = x^b \bmod p$
 $m = \text{Dec_SKE}(c; k_s)$

Problem: Key Authentication

Hybrid Encryption

0. B: $(pk_B, sk_B) = \text{Gen_PK}$
 $\text{publish}(B, pk_B)$
1. A: $k_s = \text{Gen_SK}$
 $c1 = \text{Enc_PKE}(k_s; pk_B)$
 $c2 = \text{Enc_SKE}(m; k_s)$
2. A $\rightarrow$ B: $c1, c2$
3. B: $k_s = \text{Dec_PKE}(c1; sk_B)$
 $m = \text{Dec_SKE}(c2; k_s)$

Encryption w/ KA

0. B: pick p, g ,
 $y = g^b \bmod p$
 $\text{publish}(g, p, y)$
1. A: pick a ,
 $x = g^a \bmod p$
 $k_s = y^a \bmod p$
 $c = \text{Enc_SKE}(m; k_s)$
2. A $\rightarrow$ B: x, c
3. B: $k_s = x^b \bmod p$
 $m = \text{Dec_SKE}(c; k_s)$

Public Key Infrastructure

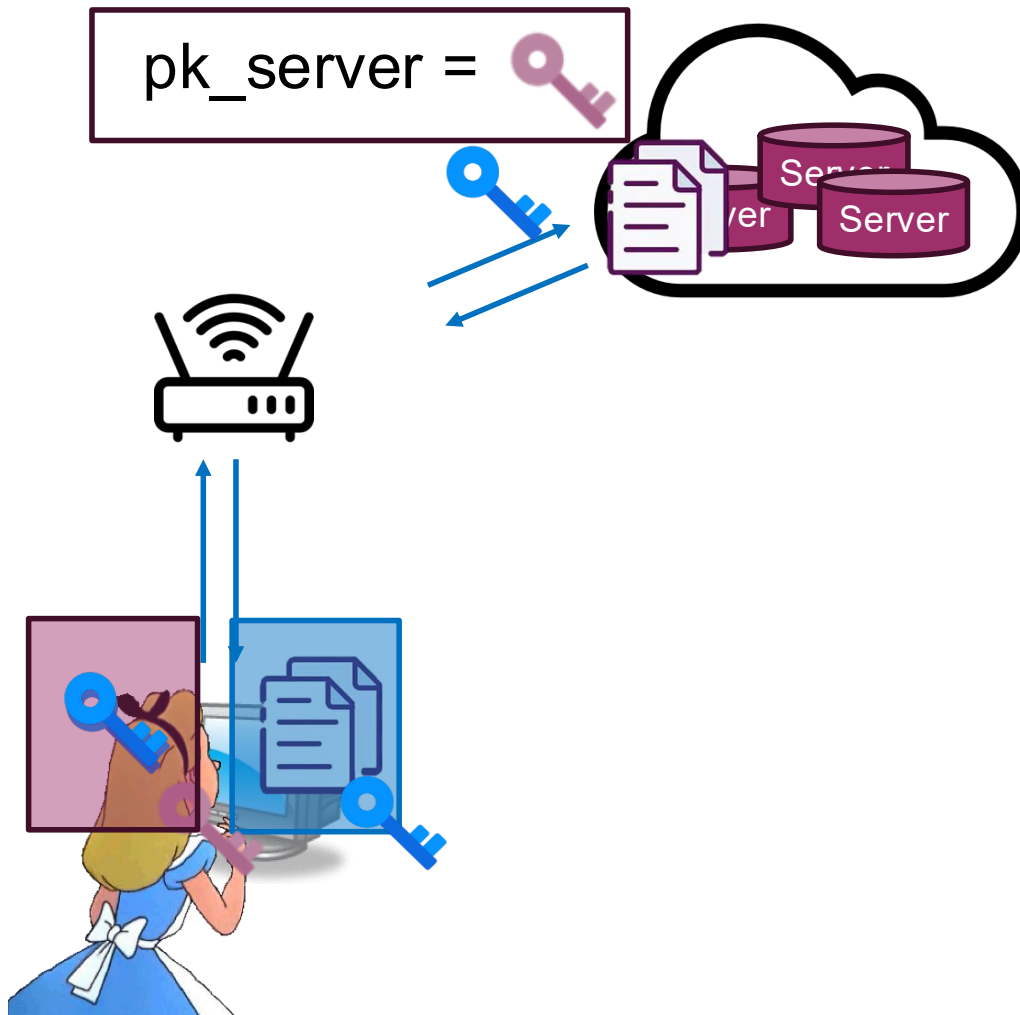
X.509 Certificates

- **Digital certificate** is a document binding together:
 - **identity** of principal
 - **public key** of that principal (might be RSA pk or DH info)
- binding together = signed
- standard: X.509 certs

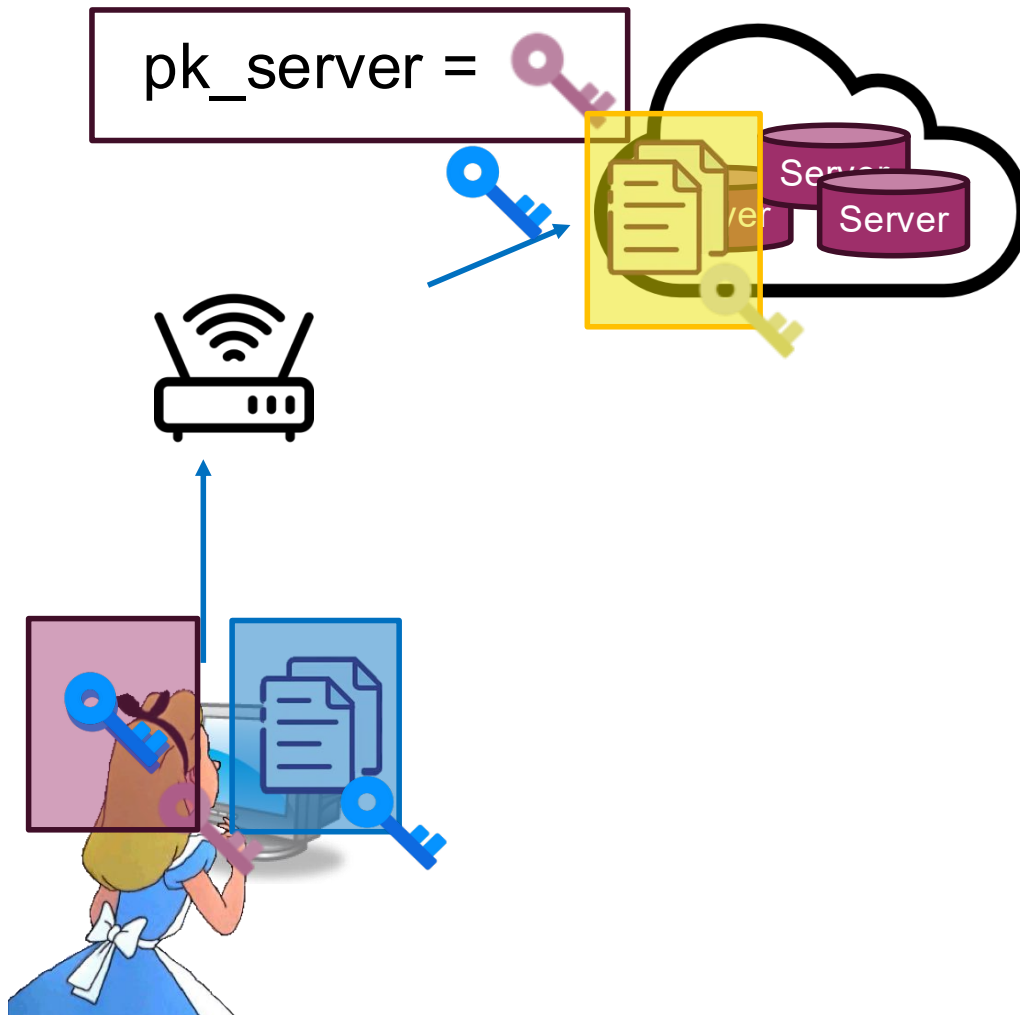
Certificate Authorities (CAs)

- responsible for verifying identity and signing certs
- Can also sign certs saying they trust other CAs
- You decide which root CAs you trust
- more precisely, this is defined in a file on your computer

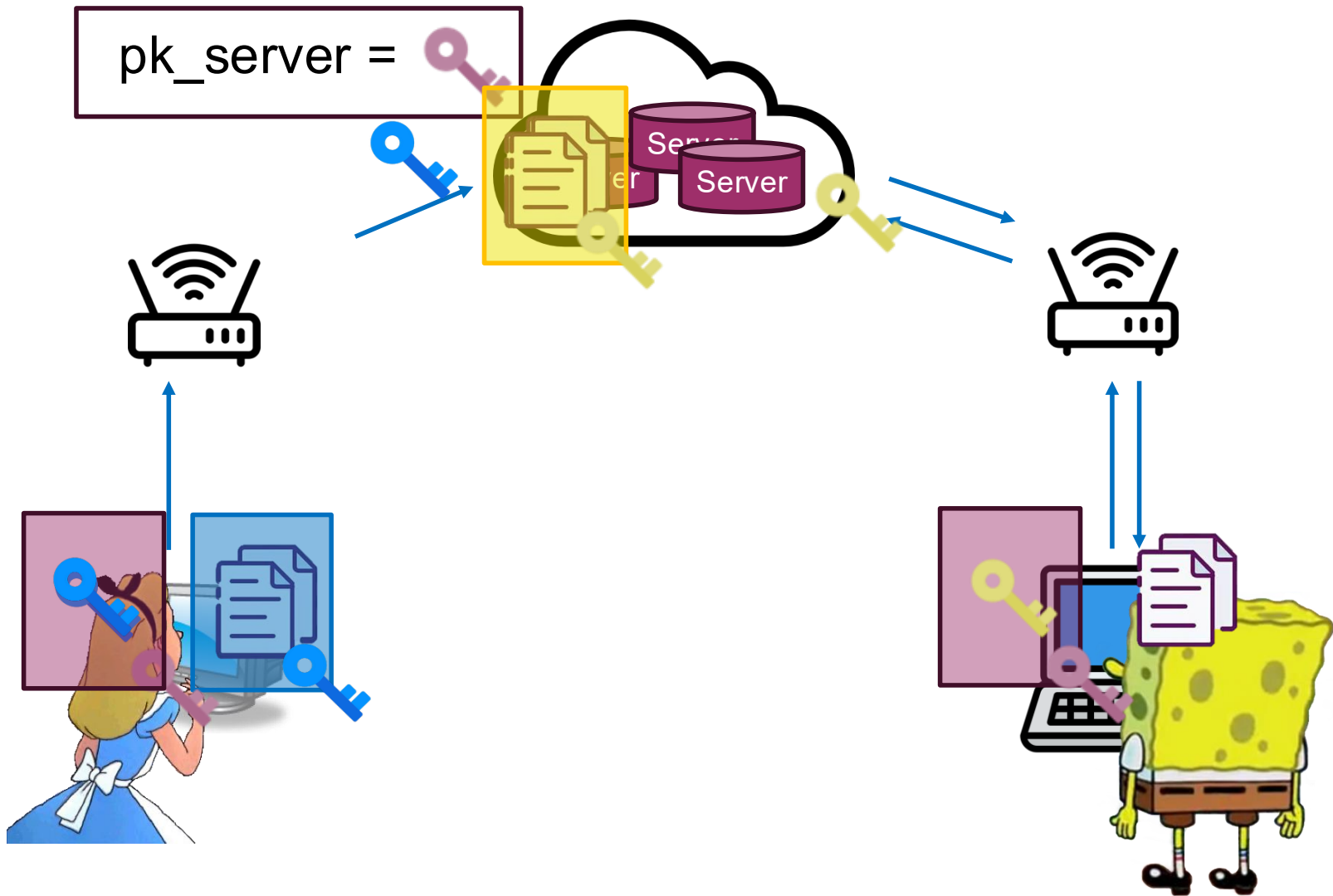
HTTPS



Cloud storage



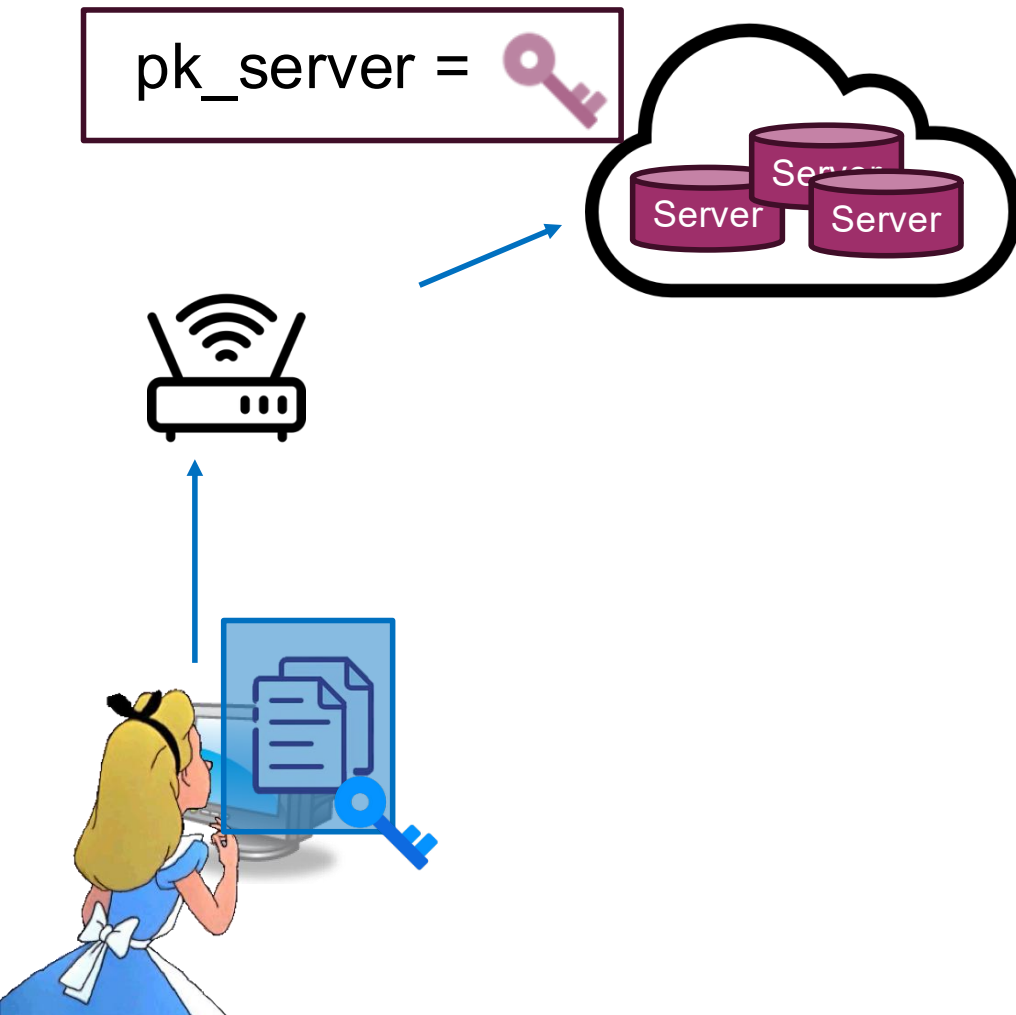
Point-to-Point Encryption



Implications

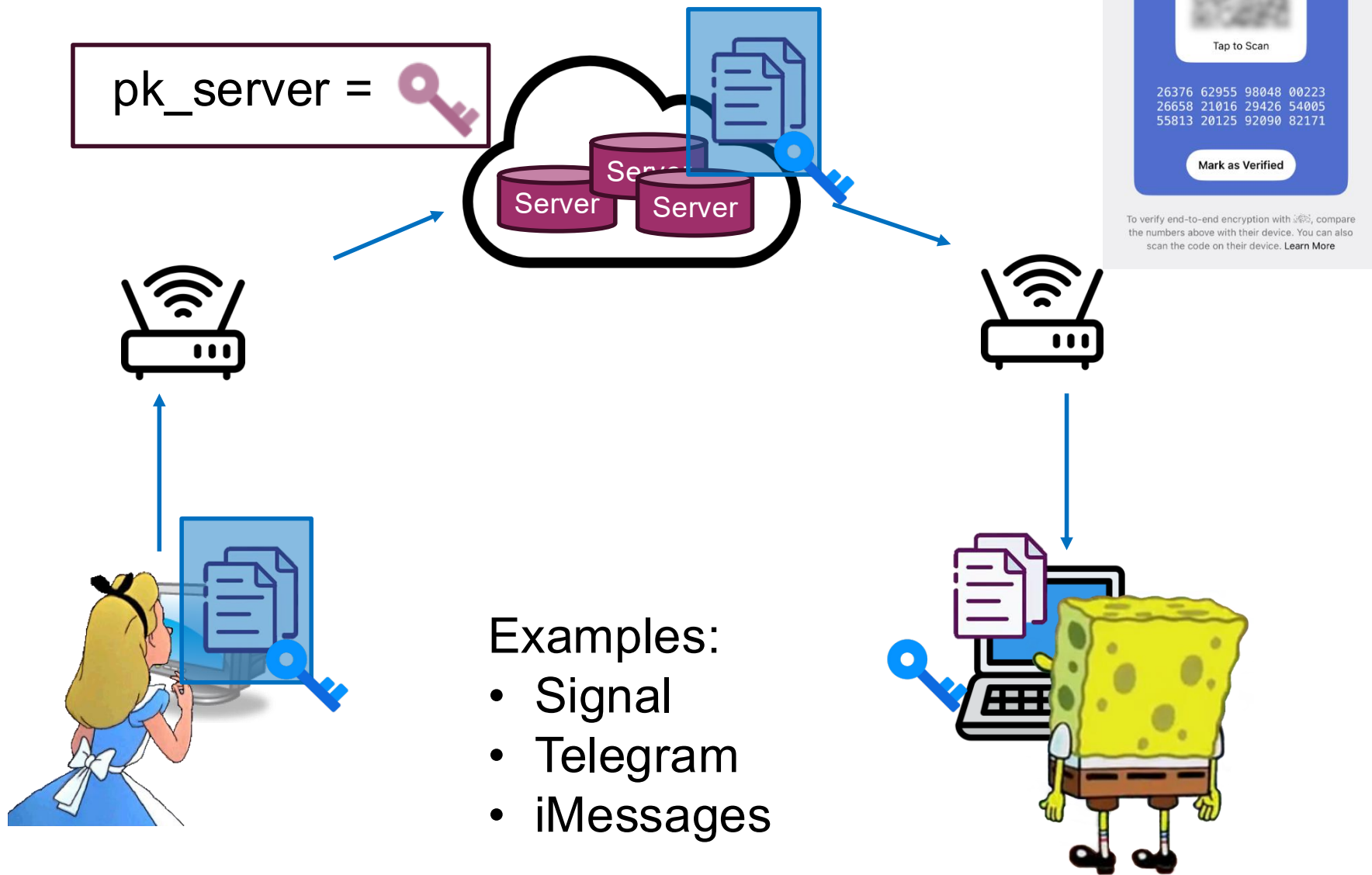
- All messages/content is encrypted in transit
- All messages/content is encrypted when stored
- Server can decrypt all messages/content

Client-Side encryption



- Can store encrypted files locally
 - iOS does by default when enable passcode
- Can backup encrypted files in cloud
 - Example: iOS Advanced Data Protection
 - Sometimes called E2E
- key often generated from password
- can also be chosen at random

End-to-End Encryption



The Debate over E2E encryption

Pros

- Good for privacy
- Don't have to trust server
- Important for journalists/ protesters, esp. in suppressive regimes

Cons

- Can't monitor for bad content
 - violations of T&C
 - illegal content
- Gov can't subpoena data