Attack approaches

$n = p \cdot q$

- **Mathematical attacks**: several approaches, all equivalent in effort to factoring the product of two primes. The defense against mathematical attacks is to use a large key size.
- **Timing attacks**: These depend on the running time of the decryption algorithm

  attacker c

- **Chosen ciphertext attacks**: this type of attacks exploits properties of the RSA algorithm by selecting blocks of data. These attacks can be thwarted by suitable padding of the plaintext, such as PKCS1 V1.5 in SSL

Receiver

RSA Decryption With Message Blinding

$c \rightarrow \boxed{c \cdot r^e} \rightarrow (c \cdot r^e)^d$

$m' \rightarrow m' \cdot r^{-1} \rightarrow m$

1. Choose random blinding factor

Pick random r with gcd(r,n)=1

2. Blind the ciphertext

Compute $c' = c \cdot r^e \mod n$ where e is the *public* exponent.

3. Decrypt the blinded ciphertext

Compute $m' = (c')^d \mod n$ → random input

4. Unblind the result

Compute $m = m' \cdot r^{-1} \mod n$ where r^{-1} is the modular inverse of r.

- The output m is the correct plaintext.

$m' = (c')^d \mod n$ $\quad m' \overset{?}{\Rightarrow} m$

Attacker doesn't know r

$= (c \cdot r^e)^d \mod n$

$= c^d \cdot r^{ed} \mod n$ $\quad ed = 1 \mod \phi(n)$

decryption time $\rightleftarrows$ SK

$= c^d \cdot r^{1 + k \cdot \phi(n)} \mod n$

$= c^d \cdot r \cdot (r^{\phi(n)})^k \mod n$

$\ll d$

$\cdot r$

$\underbrace{r^{\phi(n)}}_{1} \quad \gcd(r, n) = 1$

$= \underline{c^d} \cdot r = \underline{m} \cdot r'$

$\cdot \frac{1}{r} \longrightarrow m$

Attacker Aim: K. C.

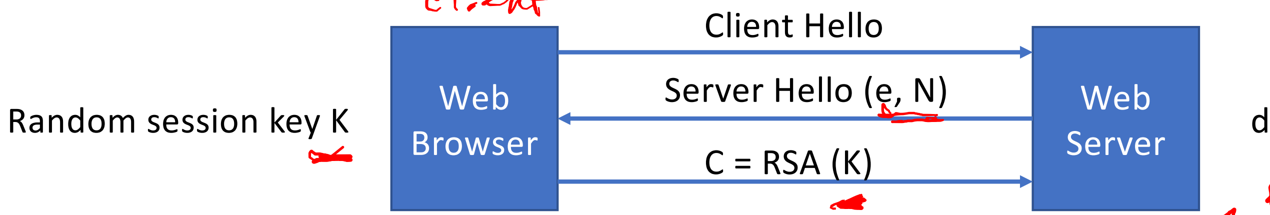
A simple attack on textbook RSA

TLS

Sum.

$C = (K_1 \cdot K_2)^e$

$K_1 = C / K_2^e$



Hash map

given.

- Session-key K is 64 bits. View $K \in \{0,\ldots,2^{64}\}$
  - Eavesdropper sees: $C = K^e \pmod N$. encryption

32

- Suppose $\mathbf{K = K_1 \cdot K_2}$ where $K_1, K_2 \le 2^{34}$.
  - Then: $\mathbf{C/K_1^e = K_2^e} \pmod N$

$O(2^{34}) + O(2^{34} \cdot 34) < 2^{6}$ $< 2^{40}$

- Build table: $C/1^e, C/2^e, C/3^e, \ldots, C/2^{34e}$. time: 2^{34}

  For $K_2 = 0,\ldots, 2^{34}$ test if K_2^e is in table. time: $2^{34} \cdot 34$

- Attack time: $\approx 2^{40} << 2^{64}$ → brute

K_2^e | C/K_1^e → K_1 | $C/1^e$ → 1 | $C/2^e$ → 2

$K = K_2 \cdot K_1$

for { Pick K_1, Calculate C/K_1^e, Store K_1 in table } → Index

One K_2 34 / # K_2 2^{36}

for Pick K_2, find if K_2^e is in table, Yes, K_1

Take-home exercise – no need to submit

- SW textbook (6th edition) problems: 3.14 & 3.15

n

RSA reading materials

- A Method for Obtaining Digital Signatures and Public-Key Cryptosystems

Homomorphic encryption

- Encryption scheme that allows computation on ciphertexts
  - an extension of public-key encryption scheme that allows anyone in possession of the public key to perform operations on encrypted data without access to the decryption key
- Partially Homomorphic Encryption: Initial public-key systems that allow this for either addition or multiplication, but not both.
  - i.e. RSA
- Fully homomorphic encryption (FHE)

$$E(m_1) \cdot E(m_2) = m_1^e \cdot m_2^e \mod n$$

$$= (m_1 \cdot m_2)^e \mod n$$

$$= E(m_1 \cdot m_2) \xleftarrow{sk}$$

$$m_1^e + m_2^e \neq (m_1 + m_2)^e$$

$$\begin{tikzcd} M \arrow[r, "\text{Mathematic}"] \arrow[d] & f(M) \\ C \arrow[r, "f(C)"] & C' \arrow[u, "sk"'] \end{tikzcd}$$

$m_1 \cdot m_2$ ← multiplicative

Application of homomorphic encryption

ccm

- One Use case: cloud computing
  - A weak computational device Alice (e.g., a mobile phone or a laptop) wishes to perform a computationally heavy task, beyond her computational means. She can delegate it to a much stronger (but still feasible) machine Bob (the cloud, or a supercomputer) who offers the service of doing so. The problem is that Alice does not trust Bob.