Lecture 24: RSA Encryption

RSA Encryption

・ロト ・ 御 ト ・ ヨ ト ・ ヨ ト ・

э

- We pick two primes uniformly and independently at random $p, q \stackrel{s}{\leftarrow} P_n$
- We define $N = p \cdot q$
- We shall work over the group (ℤ^{*}_N, ×), where ℤ^{*}_N is the set of all natural numbers < N that are relatively prime to N, and × is integer multiplication mod N

• We pick
$$y \stackrel{\$}{\leftarrow} \mathbb{Z}_N^*$$

- Let $\varphi(N)$ represent the size of the set \mathbb{Z}_N^* , which is (p-1)(q-1)
- We pick any e ∈ Z^{*}_{φ(N)}, that is, e is a natural number < φ(N) and is relatively prime to φ(N)
- We give (n, N, e, y) to the adversary A as ask her to find the e-th root of y, i.e., find x such that x^e = y

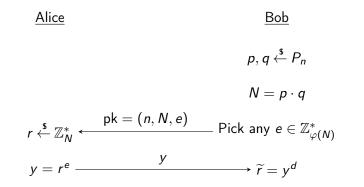
RSA Assumption. For any computationally bounded adversary, the above-mentioned problem is hard to solve

Recall: Properties

- The function $x^e \colon \mathbb{Z}_N^* \to \mathbb{Z}_N^*$ is a bijection for all e such that $gcd(e, \varphi(N)) = 1$
- Given (n, N, e, y), where y ← Z_N^{*}, it is difficult for any computationally bounded adversary to compute the *e*-th root of y, i.e., the element y^{1/e}
- But given d such that e · d = 1 mod φ(N), it is easy to compute y^{1/e}, because y^d = y^{1/e}

Now, think how we can design a key-agreement scheme using these properties. Once the key-agreement protocol is ready, we can use a one-time pad to create an public-key encryption scheme.

First, Alice and Bob establish a key that is hidden from the adversary



Note that $r = \tilde{r}$ and is hidden from an adversary based on the RSA assumption

(日)

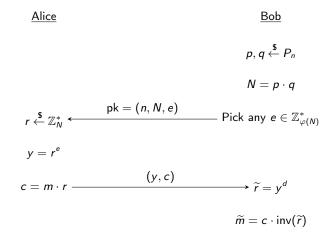
Using this key, Alice sends the encryption of $m \in \mathbb{Z}_N^*$ using the one-time pad encryption scheme.



Since, we always have $r = \tilde{r}$, this encryption scheme always decrypts correctly. Note that $inv(\tilde{r})$ can be computed only by knowing $\varphi(N)$.

(日本) (日本) (日本)

Putting the two together: RSA Encryption (First Attempt) I



RSA Encryption

◆□▶ ◆舂▶ ◆注▶ ◆注▶

We emphasize that this encryption scheme work only for $m \in \mathbb{Z}_N^*$. In particular, this works for all messages m that have a binary representation of length less than *n*-bits, becuase p and q are *n*-bit primes.

HOWEVER, THIS SCHEME IS INSECURE

伺 ト イヨ ト イヨト

• Let us start with a simpler problem.

Suppose I pick an integer x and give $y = x^3$ to you. Can you efficiently find the x?

- Running for for loop with $i \in \{0, ..., y\}$ and testing whether $i^3 = y$ or not is an inefficient solution
- However, binary search on the domain $\{0,\ldots,y\}$ is an efficient algorithm
- Then why does the RSA assumption that says "computing the e-th root is difficult if φ(N) is unknown" hold? Answer: Because we are working over Z^{*}_N and not Z! "Wrapping around" due to the modulus operation while cubing kills the binary search approach.

・ロト ・ 一下・ ・ 日 ・ ・ 日 ・

Insecurity of the First Attempt

- However, if x is such that x^e < N then the modulus operation does not take effect. So, if x < N^{1/e} then we can find the e-th root of y!
- Now, let us try to attack the "first attempt" algorithm
- Recall that we have $c = m \cdot r$ and $y = r^e$. So, we have $c^e = m^e \cdot r^e$. Now, note that $c^e \cdot inv(y) = m^e \cdot r^e \cdot y^{-1} = m^e$.
- So, the adversary can compute c^e · inv(y) to obtain m^e. If m < N^{1/e}, then the adversary can use binary search to recover m.
- There is another problem! If Alice is encrypting and sending multiple messages {m₁, m₂,...}, then the eavesdropper can recover {m₁^e, m₂^e,...}. So, she can find which of these {m₁^e, m₂^e,...} are identical. In turn, she can find out the messages in {m₁, m₂,...} that are identical (because x^e: Z_N^{*} → Z_N^{*} is a bijection).
- How do we fix these attacks?

▲御▶ ▲注▶ ▲注▶

RSA Encryption

- Our idea is to pad the message m with some randomness s. The new message s || m, with high probability, satisfies $(s || m)^e > N$ (that is, it wraps around)
- How does it satisfy the second attack mentioned above (Think: Birthday bound)
- Let us write down the new encryption scheme for $m \in \{0,1\}^{n/2}$

Enc_{n,N,e}(m):
Pick
$$r \leftarrow \mathbb{Z}_N^*$$

Pick $s \leftarrow \{0,1\}^{n/2}$
Compute $y = r^e$, and $c = (s||m) \cdot r$
Return (y, c)

- Note that masking with r is not helping at all! Let us call s ||m as the payload. An adversary can obtain the "e-th power of the payload" by computing c^e ⋅ y⁻¹
- So, we can use the following optimized encryption algorithm instead

Enc_{n,N,e}(m):
Pick
$$s \stackrel{\$}{\leftarrow} \{0,1\}^{n/2}$$

Return $c = (s || m)^e$

・ 同 ト ・ ヨ ト ・ ヨ ト

Let us summarize all the algorithms that we need to implement RSA algorithm

- Generating *n*-bit primes to sample p and q
- **②** Generating *e* such that *e* is relatively prime to $\varphi(N)$, where N = pq
- So Finding the trapdoor d such that $e \cdot d = 1 \mod \varphi(N)$