Lecture 34: RSA Encryption

Recall: RSA Assumption

- We pick two primes uniformly and independently at random $p, q \xleftarrow{\$} P_n$
- We define $N = p \cdot q$
- We shall work over the group $(\mathbb{Z}_N^*, \times)$, where $\mathbb{Z}_N^*$ is the set of all natural numbers $< N$ that are relatively prime to N, and $\times$ is integer multiplication $\mod N$
- We pick $y \xleftarrow{\$} \mathbb{Z}_N^*$
- Let $\varphi(N)$ represent the size of the set $\mathbb{Z}_N^*$, which is $(p-1)(q-1)$
- We pick <u>any</u> $e \in \mathbb{Z}_{\varphi(N)}^*$, that is, e is a natural number $< \varphi(N)$ and is relatively prime to $\varphi(N)$
- We give (n, N, e, y) to the adversary $\mathcal{A}$ 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 $y \xleftarrow{\$} \mathbb{Z}_N^*$, 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 $e \cdot d = 1 \mod \varphi(N)$, 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.

Key-Agreement

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

$$
\begin{array}{ccc}
\underline{\text{Alice}} & & \underline{\text{Bob}} \\
 & & p, q \xleftarrow{\$} P_n \\
 & & N = p \cdot q \\
r \xleftarrow{\$} \mathbb{Z}_N^* & \xleftarrow{\quad \mathrm{pk} = (n, N, e) \quad} & \text{Pick any } e \in \mathbb{Z}_{\varphi(N)}^* \\
y = r^e & \xrightarrow{\quad y \quad} & \widetilde{r} = y^d
\end{array}
$$

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

Public-key Encryption after the Key-Agreement Protocol

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

$$\begin{array}{ccc} \underline{\text{Alice}} & & \underline{\text{Bob}} \\ c = m \cdot r & \xrightarrow{\quad\quad c \quad\quad} & \widetilde{m} = c \cdot \mathrm{inv}(\widetilde{r}) \end{array}$$

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

Putting the two together: RSA Encryption

$$
\begin{array}{ccc}
\underline{\text{Alice}} & & \underline{\text{Bob}} \\
 & & p, q \xleftarrow{\$} P_n \\
 & & N = p \cdot q \\
r \xleftarrow{\$} \mathbb{Z}_N^* & \xleftarrow{\quad \mathsf{pk} = (n, N, e) \quad} & \text{Pick any } e \in \mathbb{Z}_{\varphi(N)}^* \\
y = r^e & & \\
c = m \cdot r & \xrightarrow{\quad (y, c) \quad} & \widetilde{r} = y^d \\
 & & \widetilde{m} = c \cdot \mathrm{inv}(\widetilde{r})
\end{array}
$$

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.