

Lecture 24: Digital Signatures using RSA Assumption

- Bob wants to receive encrypted messages. So, Bob fixes n , the number of bits in the primes he wants to choose. Bob picks two random n -bit primes p and q . Bob computes $N = p \cdot q$. Bob samples a random $e \in \mathbb{Z}_{\varphi(N)}^*$. Bob computes $d \in \mathbb{Z}_{\varphi(N)}^*$ such that $e \cdot d = 1 \pmod{\varphi(N)}$ using the extended GCD algorithm. Bob set $pk = (n, N, e)$ and $trap = d$.
- The public-key for Bob pk is broadcast to everyone
- To encrypt a message $m \in \{0, 1\}^{n/2}$, Alice runs the $Enc_{pk}(m)$ algorithm defined as follows. Alice samples $r \in \{0, 1\}^{n/2}$ and computes $c = (r||m)^e \pmod{N}$. The cipher-text is c .
- After receiving a cipher-text \tilde{c} , Bob runs the decryption algorithm $Dec_{pk, trap}(\tilde{c})$. Bob computes $(\tilde{r}, \tilde{m}) = \tilde{c}^d \pmod{N}$.

- **Correctness.** We have seen that this public-key encryption is always correct (relies on the fact that $\gcd(e, \varphi(N)) = 1$)
- **Security.** We have seen that this public-key encryption scheme is secure as long as the randomness r used in every encryption algorithm is distinct against computationally bounded eavesdroppers (relies on the birthday bound and the RSA assumption)

Abstraction

- Recall that we have seen that the function $f_e: \mathbb{Z}_N^* \rightarrow \mathbb{Z}_N^*$ defined by $f_e(x) = x^e \pmod N$ is a bijection that is efficient to evaluate. We shall abstract this concept as “Evaluation is efficient”
- Recall that the inverse function $f_e^{-1}: \mathbb{Z}_N^* \rightarrow \mathbb{Z}_N^*$ is efficient to evaluate given d , where $e \cdot d = 1 \pmod{\varphi(N)}$; otherwise, not. We shall abstract this concept as “Inversion is inefficient”
- In a public-key encryption we want that the “encryption algorithm is efficient” and “decryption algorithm is inefficient.” So, we used the evaluation of f_e for encryption and the inversion of f_e for decryption.

Digital Signature

- In a digital signature scheme, the signer publishes a public-key pk and keeps a trapdoor $trap$ with herself
- Later, if the signer wants to endorse a message m then she uses an algorithm $Sign_{pk, trap}(m)$ to generate a signature σ
- Everyone should be able to verify that “the publisher of the public-key pk endorses the message \tilde{m} using the signature $\tilde{\sigma}$ ” by running the verification algorithm $Ver_{pk}(\tilde{m}, \tilde{\sigma})$ ”
- An adversary who sees the public-key pk and a few message-signature pairs $(m_1, \sigma_1), (m_2, \sigma_2), \dots, (m_k, \sigma_k)$ cannot forge a valid signature σ' on a new message m'

- First observe that we want “verification to be efficient” and “signing to be inefficient”
- So, using the ideas in the “abstraction slide,” the idea is to use “evaluation of f_e ” for verification and “inversion of f_e ” for signing

- Alice decides to endorse messages using n -bit primes. Alice picks two random n -bit prime numbers p, q . Alice computes $N = p \cdot q$ and samples a random $e \in \mathbb{Z}_{\varphi(N)}^*$. Alice computes d such that $e \cdot d = 1 \pmod{\varphi(N)}$. Alice sets $\text{pk} = (n, N, e)$ and $\text{trap} = d$
- To sign a message $m \in \{0, 1\}^n$, Alice runs $\text{Sign}_{\text{pk}, \text{trap}}(m)$ defined as follows. Compute $\sigma = m^d \pmod{N}$.
- To verify a message-signature pair $(\tilde{m}, \tilde{\sigma})$, Bob runs the verification algorithm $\text{Ver}_{\text{pub}}(\tilde{m}, \tilde{\sigma})$ defined as follows. Output $\tilde{m} == \tilde{\sigma}^e \pmod{N}$.

THIS SCHEME IS INSECURE!

Attack on the Previous Scheme

- Pick any $\sigma' \in \mathbb{Z}_N^*$
- Compute $m' = (\sigma')^e \pmod N$
- Note that this is an efficient attack
- Note that we did not even need to see any other message-signature pairs
- Although, we do not have any “control” over the message. It is a valid forgery nonetheless

- We want to use the fact that in the previous forgery attack, the adversary did not have any control over the message that was being signed
- So, here is the idea underlying the fix. We shall pick a random $r \in \{0, 1\}^{n/2}$ and include r in the public-key pk . To sign a message $m \in \{0, 1\}^{n/2}$, we compute $(r||m)$ and compute the signature $\sigma = (r||m)^d \pmod N$. To verify a message-signature pair $(\tilde{m}, \tilde{\sigma})$, Bob (the verifier) checks $(r, \tilde{m}) == (\tilde{\sigma})^e \pmod N$
- The formal scheme is presented next

Gen(1^n):

- Pick random n -bit primes p and q .
- Compute N and $\varphi(N)$
- Sample $e \in \mathbb{Z}_{\varphi(N)}^*$
- Compute d such that $e \cdot d = 1 \pmod{\varphi(N)}$
- Sample random $r \in \{0, 1\}^{n/2}$
- Return $\text{pk} = (n, N, e, r)$ and $\text{trap} = d$

$\text{Sign}_{\text{pk,trap}}(m)$:

- Return $(r\|m)^d \pmod N$

$\text{Ver}_{\text{pk}}(\tilde{m}, \tilde{\sigma})$:

- Return $(r\|\tilde{m}) \stackrel{?}{=} \tilde{\sigma}^e \pmod N$

In the next lecture we shall learn how to sign arbitrary-length messages $m \in \{0, 1\}^*$