In today’s lecture we are going to address the following problem. Suppose Bob receives a message which was supposedly sent by Alice. How does Bob ensure that the message received actually came from Alice? More specifically, if the message was actually sent by Alice, how does Bob ensure that the message was not tampered with by any malicious intermediary?

In day-to-day life, we use signatures to solve the forementioned problems. In today’s class, we describe two cryptographic notions of signatures – message authentication codes and digital signatures. Message Authentication Codes (MACs) are used in the private key setting – only people who know the secret key can check if a message is valid. However, a very natural requirement of any signature scheme is that only Alice should be able to sign a message sent by Alice; anyone else who “knows” Alice should be able to verify her signature. This is precisely the notion of Digital Signatures. Only people who know Alice’s secret key should be able to generate a digital signature; anyone who knows Alice’s public key should be able to verify a digital signature generated by Alice.

1 Message Authentication Codes

Definition 1 (MAC) \((Gen, Tag, Ver)\) is a MAC over the message space \(\{M\}_n\) if the following hold:

- \(Gen\) is a PPT that returns a key \(k \leftarrow Gen(1^n)\).
- \(Tag\) is a PPT that on input key \(k\) and message \(m\) outputs a tag \(\sigma \leftarrow Tag_k(m)\).
- \(Ver\) is a deterministic PTime algorithm that on input \(k, m\) and \(\sigma\) outputs “accept” or “reject”.

\[\forall n \in N, \forall m \in M_n \Pr[k \leftarrow Gen(1^n) : Ver_k(m, Tag_k(m)) = \text{“accept”}] = 1\]

The above definition requires that verification algorithms always correctly “accepts” a valid signature.

The goal of an adversary is to forge a MAC. More specifically, the adversary’s goal is to construct a tag \(\sigma’\) such that it is a valid signature for some message. We could consider many different adversaries with varying powers depending on whether the adversary has
access to signed messages; whether the adversary has access to a signing oracle; and whether the adversary can pick the message to be forged.

The strongest adversary is the one who has oracle access to Tag and is allowed to forge any chosen message.

**Definition 2 (Security of a MAC)** A MAC \((\text{Gen}, \text{Tag}, \text{Ver})\) is secure is for all \(PPT\) \(A\), \(\exists\) a negligible function \(\epsilon(n)\) such that for all \(n\),

\[
\Pr[k \leftarrow \text{Gen}(1^n); m, \sigma \leftarrow A^{\text{Tag}(\cdot)}(1^n) : \\
A \text{ did not query } m \land \text{Ver}_k(m, \sigma) = \text{“accept”}] \leq \epsilon(n)
\]

We now show a construction of a MAC using pseudorandom functions.

**Theorem 1** If there exists a pseudorandom function, then there exists a MAC over the message space \(\{0, 1\}^n\).

**Proof.** Let \(F = \{f_s\}\) be a family of pseudorandom functions such that \(f_s : \{0, 1\}^{|s|} \rightarrow \{0, 1\}^{|s|}\). Construct the MAC as follows:

- \(\text{Gen}(1^n)\): \(k \leftarrow \{0, 1\}^n\).
- \(\text{Tag}_k(m)\): \(F_k(m)\).
- \(\text{Ver}_k(m, \sigma)\): “accept” if \(F_k(m) = \sigma\).

Consider the above scheme when a random function \(RF\) is used instead of the pseudorandom function \(F\). In this case, \(A\) succeeds with a probability at most \(2^{-n}\), since \(A\) only wins if \(A\) is able to guess the \(n\) bit random string which is the output of \(RF_k(m)\). From the definition of a pseudorandom function, there is no non uniform ppt distinguisher which can distinguish the output of \(F\) and \(RF\) with a non negligible probability. Hence, we conclude that \((\text{Gen}, \text{Tag}, \text{Ver})\) is secure.

**2 Digital Signatures**

As we briefly mention before, digital signatures mirror real-life signatures in that anyone who knows Alice can verify a signature generated by Alice. Moreover, digital signatures possess the property of non-repudiability, i.e., if Alice signs a message and sends it to Bob, then Bob can prove to a third party the validity of the signature. Hence, digital signatures can be used as certificates in a public key infrastructure.
Definition 3 (Digital Signatures) \((Gen, Sign, Ver)\) is a digital signature over \(\{M\}_{n \in \mathbb{N}}\) if

- \(Gen(1^n)\) is a PPT which on input \(n\) outputs a public key \(pk\) and a secret key \(sk\): \(pk, sk \leftarrow Gen(1^n)\).
- \(Sign\) is a PPT which on input a secret key \(sk\) and message \(m\) outputs a signature \(\sigma\): \(\sigma \leftarrow Sign_{sk}(m)\).
- \(V er\) is a deterministic PTime algorithm which on input a public key \(pk\), a message \(m\) and a signature \(\sigma\) returns either “accept” or “reject”.

\[
\forall n \in \mathbb{N}, \forall m \in \mathcal{M}_n, \Pr[pk, sk \leftarrow Gen(1^n) : V er_{pk}(m, Sign_{sk}(m)) = "accept"] = 1
\]

The security of a digital signature can be defined in terms very similar to the security of a MAC. Note that by definition of a public key infrastructure, the adversary has free oracle access to the verification algorithm \(V er\).

Definition 4 (Security of Digital Signatures) \((Gen, Sign, Ver)\) is secure if \(\forall PPT\), \(\exists\) a negligible function \(\epsilon(n)\) such that \(\forall n \in \mathbb{N}\),

\[
\Pr[pk, sk \leftarrow Gen(1^n); m, \sigma \leftarrow A^{Sign_{sk}(\cdot)}(1^n) : \neg A \text{ did not query } m \land V er_{pk}(m, \sigma) = "accept"] \leq \epsilon(n)
\]

2.1 Constructing a Digital Signature

As a first attempt, we construct a signature scheme using trapdoor permutations. However, we will show that such signature schemes are not secure. Consider the following scheme:

- \(Gen(1^n): pk = i\) and \(sk = t\), the trapdoor.
- \(Sign_{sk}(m) = f^{-1}(m)\) using \(t\).
- \(V er_{pk}(m, \sigma) = "accept"\) if \(f_i(\sigma) = m\).

The above scheme is not secure if the adversary is allowed to choose the message to be forged. Picking \(m = f_i(0)\) guarantees that \(0\) is the signature of \(m\).

If a specific trapdoor function like RSA is used, adversaries can forge larger class of messages. In the RSA scheme,
• Gen(1^n): pk = e, N and sk = d, N, such that ed = 1 mod φ(N), and N = pq, p, q primes.
• Sign_{sk}(m) = m^d mod N.
• Ver_{pk}(m, σ) = “accept” if σ^e = m mod N.

Given signatures on σ_1 = m_1^d mod N and σ_2 = m_2^d mod N an adversary can easily forge a signature on m_1m_2 by multiplying the two signatures modulo N.

To avoid such attacks, in practice, the message is first hashed using some “random looking” function h to which the trapdoor signature scheme can applied. It is secure if h is a random function RF. We cannot, however, use a pseudorandom function. In order to use a PRF for signatures, the adversary would have to know the hashing function and hence the seed of the PRF, in which case the PRF ceases to be computationally indistinguishable from a random function RF.

2.2 Construction of a Provably Secure Digital Signature

In today’s lecture we construct a one-time signature.

Definition 5 (One Time Secure Signature) A digital signature scheme (Gen, Sign, Ver) is one-time secure if ∀PPT, ∃ a negligible function ε(n) such that ∀n ∈ N,

\[
\Pr[pk, sk \leftarrow \text{Gen}(1^n); m, σ \leftarrow \text{Sign}_{sk}(1^n) : \text{A makes at most one query to the Sign}_{sk} \text{ oracle } \land \text{A did not query } m \land \text{Ver}_{pk}(m, σ) = “accept”] \leq \epsilon(n)
\]

Theorem 2 Existence of a OWF implies the existence of a one-time secure signature scheme on \{\{0, 1\}^n\}_{n \in \mathbb{N}}.

Proof. Let f be a one way function. Construct the signature scheme as follows:

• Gen(1^n): for i = 1 to n, let x_0^i and x_1^i be random n-bit strings. Let y_0^i = f(x_0^i) and y_1^i = f(x_1^i). The public key pk is the set of all y’s. The secret key sk is the set of all x’s.
• Sign_{sk}(m): for i = 1 to n, σ_i = x_{m[i]}^i. Output σ = σ_1σ_2…σ_n.
• Ver_{pk}(m, σ) = “accept” if ∀i ∈ [n], f(σ_i) = y_{m_i}^i.
Suppose the above scheme is not one-time secure. That is, assume that $A$ succeeds with a non-negligible probability $\epsilon(n)$ to forge a signature. In such a case, we construct a ppt algorithm $B$ that inverts the one way function $f$ with probability at least $\frac{\epsilon(n)}{\text{poly}(n)}$ which leads to the required contradiction.

The new machine $B$ works as follows. $B$ takes as input $y$ and attempts to outputs $f^{-1}(y)$.

- Pick a random $j \leftarrow [n]$ and $c \leftarrow \{0, 1\}$.
- Generate $pk$ and $sk$ honestly but let $y^j_c = y$. We do not know the inverse of $y^j_c$, i.e., we can only sign messages where the $j^{th}$ bit is not $c$.
- Internally run $m', \sigma' \leftarrow A(pk, 1^n)$. When simulating $A$, $A$ makes oracle queries to $\text{Sign}_{sk}(\cdot)$. Answer these queries honestly if the $j^{th}$ bit is not $c$. Abort otherwise.
- If $m'_j = c$, output $\sigma'_j$ otherwise $\perp$.

$B$ succeeds with non negligible probability at least $\frac{\epsilon(n)}{2^n}$. ■