**CS 276 Cryptography** **Spring 2004**

# Lecture 14b: Signature Schemes (March 15)

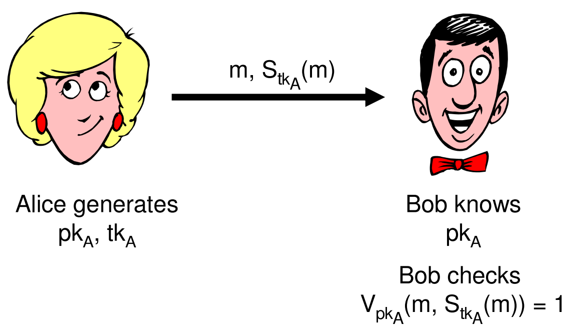
*Lecturer: Prof. David Wagner* *Scribe: Evan Chang*



## 14.1 Introduction

In the last few lectures, our focus has been on public-key encryption where the goal has been on message *secrecy*. We now again consider the problem of message *authenticity* (or message *integrity*), but in the public-key setting. Recall that message authenticity is not concerned with keeping messages secret, but rather with preventing a malicious adversary from tampering with the message.

In the public-key setting, Alice generates a public-private key pair, sends the message with a signature computed based the private key to Bob, and Bob is able to verify that the message has not been tampered with and that it came from Alice using only Alice's public key.




### 14.1.1 Basic Definitions

Before discussing notions of security, we describe the basic requirements of signature scheme. First, we assume we have a key generation algorithm

$$(pk, tk) \leftarrow G\,()$$

where $G$ is some randomized polynomial-time algorithm. Then, we have two polynomial-time algorithms $S$ and $V$—the signing and verification algorithms, respectively—with the following types:

$$\begin{array}{rcl} S & : & \mathit{Tk} \times \{0,1\}^* \rightarrow \mathit{Sigs} \\ V & : & \mathit{Pk} \times \{0,1\}^* \times \mathit{Sigs} \rightarrow \{0,1\} \end{array}$$

such that for all messages $m$ and public-private key pairs $(pk, tk)$, it is the case that $V_{pk}(m, S_{tk}(m)) = 1$. The sets $\mathit{Tk}$ and $\mathit{Pk}$ are the spaces of private keys and public keys, respectively, while $\mathit{Sigs}$ is the space of message signatures. The side-condition that $V_{pk}(m, S_{tk}(m)) = 1$ simply ensures that $V$ can always recognize the good messages.

### 14.1.2 Why MACs aren't enough?

The necessity of public-key signature schemes is illustrated by the following situation. Consider the case where Bob has received a contract signed using the scheme described above from Alice to buy his house. Later, Alice decides that she is not interested in the house anymore and wants to get out of it by denying having signed the contract in the first place. However, Bob can hold Alice to the contract because he can prove that the contract must have been signed by her, for it was signed with her private key. This property of preventing an entity from being able to deny having performed some action is called *non-repudiation.* With message authentication codes (MACs), there is no identity associated with signer since both parties have the same key, so non-repudiation cannot be achieved. Of course, actual non-repudiation is much more difficult to get in practice than using public-key signature schemes.

## 14.2 Security

A public-key signature scheme is useful only if it satisfies some notion of security where we can prevent an attacker from forging a signature. We assume that the adversary can see the signatures for a few chosen messages, but he/she should not be able to generate a signature for a new message that would be accepted by the verification algorithm. In our example, an adversary should not be able to fool Bob into accepting a new message not sent by Alice. This leads us to the following definition of security.

**Definition 14.1 (Security Against Existential Forgery)** *A public-key signature scheme is $(t, q, \varepsilon)$-secure against existential forgery if and only if for all algorithms $A$ running in time less than $t$, making at most $q$ queries to a signing oracle $S_{tk}$, and outputting a message-signature pair,*

$$\text{Adv } A \leq \varepsilon$$

*where*

$$\text{Adv } A = \Pr_{(pk,tk)\leftarrow G()}\left[(m', s') \leftarrow A^{S_{tk}}(pk); V_{pk}(m', s') = 1 \textit{ and where } m' \textit{ was not a query to } S_{tk}\right].$$

## 14.3 Example Signature Schemes

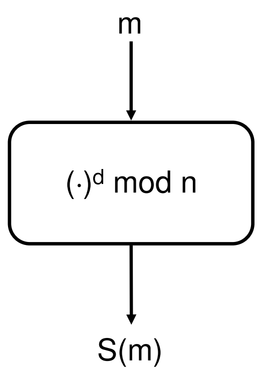
We now describe some example signature schemes built on RSA. Let $n$ denote the modulus, $m$ denote the message, the pair $(n, e)$ be the public key, and the pair $(n, d)$ be the private key. These example signature schemes are shown diagrammatically in Figure 14.1.

**Example 1 (Plain RSA)** *In this scheme, we directly use RSA as the signature algorithm. In other words, we define $S(m)$ and $V(m, s)$ (for messages $m$ and signatures $s$) as follows:*

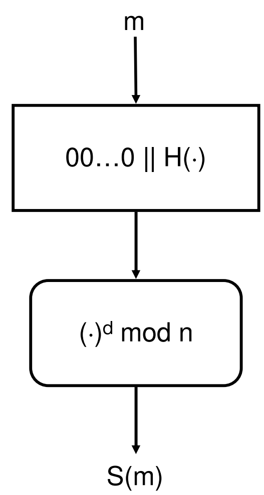
$$\begin{aligned} S(m) &\stackrel{def}{=} m^d \mod n \\ V(m, s) &\stackrel{def}{=} \begin{cases} 1 & \textit{if } s^e = m^d \pmod n \\ 0 & \textit{otherwise} \end{cases} \end{aligned}$$

Plain RSA is **insecure**. One possible attack uses the *homomorphic* property of RSA:

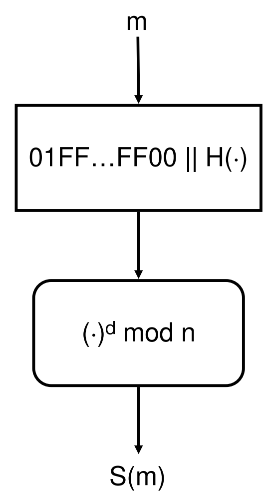
$$S(m_1) \cdot S(m_2) = S(m_1 \cdot m_2) \pmod n$$



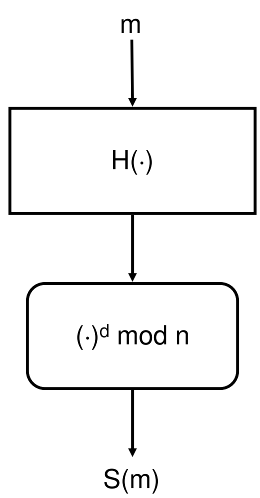

(a) Plain RSA



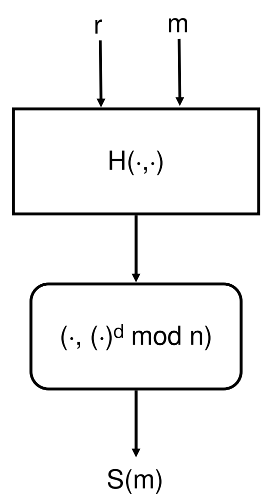

(b) Hash and Sign




(c) PKCS #1




(d) Full Domain Hash (FDH)




(e) Probabilistic Full Domain Hash (PFDH)

Figure 14.1: Diagrammatic Descriptions of Example Signature Schemes

In this attack, the adversary uses two chosen message queries to forge a valid signature for an arbitrary message $m$. The adversary can compute the signature for $m$ as follows:

$$S(m) = \frac{S(m' \cdot m)}{S(m')} \pmod{n}$$

for some $m' \neq m$. This suggests that we need to find a scheme that breaks the homomorphic property; a reasonable approach is to first hash the message.

**Example 2 (Hash and Sign)** *Let $H : \{0,1\}^* \to \{0,1\}^l$ be a hash function where $l \leq \lceil \log n \rceil$; in other words, the co-domain of the hash function may be smaller than $|(\mathbb{Z}/n\mathbb{Z})^*|$. Then, the signing algorithm is as follows:*

$$S(m) \stackrel{def}{=} (00\ldots0 \mid\mid H(m))^d \mod n$$

*where $00\ldots0 \mid\mid H(m)$ is the padding of the high-order bits of the output of the has function with enough zeros to be $\lceil \log n \rceil$-bits long. The hash function $H$ should be one-way and destroy the algebraic structure in order to avoid the attack on Plain RSA.*

Hash and Sign is also **insecure**. We should be suspicious, for the number of possible inputs to $(\cdot)^d \mod n$ is very small. Remember that the RSA assumption is only on random inputs. In general, this scheme is not provably secure for general trapdoor permuations. For RSA, there is a chosen message attack using products of small primes and some linear algebra.

**Example 3 (PKCS #1)** *This is a small variation on Hash and Sign. Instead of padding with 0s, we pad with the bytes* 00 FF ... FF 00*.*

PKCS #1 is **unjustifiable** (i.e., not shown to be insecure but not provably secure). The hope is that we avoid the problems with small primes by padding with mostly ones. Of course, the number of possible inputs to $(\cdot)^d \mod n$ is still just as small as in Hash and Sign.

**Example 4 (Full Domain Hash (FDH))** *Since padding appears the root cause for insecurity, we use a hash function $H$ that maps to all of $(\mathbb{Z}/n\mathbb{Z})^*$. Then, the signing algorithm is simply*

$$S(m) \stackrel{def}{=} \big(H(m)\big)^d \mod n$$

Full Domain Hash is provably **secure**. The proof will be given in the next lecture. However, we will see that the security result is not as strong as one like from a practical point of view.

**Example 5 (Probabilistic Full Domain Hash (PFDH))** *This a slight modification of FDH by considering a family of hash functions chosen based on a random index $r$. In this case, the signing function must also include the choice of $r$.*

$$S(m) \stackrel{def}{=} \langle r,\ \big(H(r,m)\big)^d \mod n \rangle$$

Probabilistic Full Domain Hash is provably **secure**. For the case of RSA-PFDH, we can get a stronger security result than for FDH. This will be shown in the next lecture.