*opad* be a similar sequence with the bits 01011100. The HMAC-$h$ function with key $k$ for message $m$ is

$$\text{HMAC-}h(k, m) = h(k' \oplus opad \;||\; h(k' \oplus ipad||m))$$

where $\oplus$ is exclusive or and $||$ is concatenation.

Bellare, Canetti, and Krawczyk [155] analyze the security of HMAC and conclude that the strength of HMAC depends on the strength of the hash function $h$. Emphasizing this, attacks on HMAC-MD4, HMAC-MD5, HMAC-SHA-0, and HMAC-SHA-1 have been developed, some of which recover partial [451, 1054] or full keys [710,1965]. Bellare [154] extends the analysis by explaining under what conditions HMAC is secure.

Various HMAC functions are used in Internet security protocols (see Chapter 12).

## 10.5 Digital Signatures

As electronic commerce grows, so does the need for a provably high degree of authentication and integrity. Think of Alice's signature on a contract with Bob. Bob not only has to know that Alice is the other signer and is signing it; he also must be able to prove to a disinterested third party (called a *judge*) that Alice signed it and that the contract he presents has not been altered since Alice signed it. Such a construct plays a large role in managing cryptographic keys as well. This construct is called a *digital signature*.

> **Definition 10–7.** A *digital signature* is a construct that authenticates both the origin and contents of a message in a manner that is provable to a disinterested third party.

The "proof" requirement introduces a subtlety. Let $m$ be a message. Suppose Alice and Bob share a secret key $k$. Alice sends Bob the message and its encipherment using $k$. Is this a digital signature?

First, Alice has authenticated the contents of the message, because Bob deciphers the enciphered message and can check that the message matches the deciphered one. Because only Bob and Alice know $k$, and Bob knows that he did not send the message, he concludes that it has come from Alice. He has authenticated the message origin and integrity. However, based on the mathematics alone, Bob cannot prove that he did not create the message, because he knows the key used to create it. Hence, this is not a digital signature.

Public key cryptography solves this problem. Let $d_{Alice}$ and $e_{Alice}$ be Alice's private and public keys, respectively. Alice sends Bob the message and its encipherment using $d_{Alice}$. As before, Bob can authenticate the origin and contents of the message, but in this situation a judge can determine that Alice signed the

message, because only Alice knows the private key with which the message was signed. The judge merely obtains Alice's public key $e_{Alice}$ and uses that to decipher the enciphered message. If the result is the original message, Alice signed it. This is in fact a digital signature.

A digital signature provides the service of nonrepudiation. If Alice claims she never sent the message, the judge points out that the originator signed the message with her private key, which only she knew. Alice at that point may claim that her private key was stolen, or that her identity was incorrectly bound in the certificate (see Chapter 15, "Representing Identity"). The notion of "nonrepudiation" provided here is strictly technical. In fact, Alice's key might have been stolen, and she might not have realized this before seeing the digital signature. Such a claim would require ancillary evidence, and a court or other legal agency would need to handle it. For the purposes of this section, we consider the service of nonrepudiation to be the inability to deny that one's cryptographic key was used to produce the digital signature.

### 10.5.1 Symmetric Key Signatures

All secret key digital signature schemes rely on a trusted third party. The judge must trust the third party. Merkle's scheme is typical [1322].

Let Cathy be the trusted third party. Alice shares a cryptographic key $k_{Alice}$ with Cathy. Likewise, Bob shares $k_{Bob}$ with Cathy. When Alice wants to send Bob a contract $m$, she enciphers the message using $k_{Alice}$ and sends it to Bob. Bob sends it to Cathy, who deciphers the message using $k_{Alice}$, enciphers it with $k_{Bob}$, and returns this to Bob. He can now decipher it. To verify that Alice sent the message, the judge has Cathy decipher the enciphered message Alice sent and the enciphered message Bob received from Cathy using Alice's and Bob's keys. If they match, the sending is verified; if not, one of them is a forgery.

### 10.5.2 Public Key Signatures

In our earlier example, we had Alice encipher the message with her private key to produce a digital signature. We now examine two specific systems.

#### 10.5.2.1 RSA Digital Signatures

Section 10.3.2 discussed the RSA system. We observe that using it to authenticate a message produces a digital signature. However, we also observe that the strength of the system relies on the protocol describing how RSA is used as well as on the RSA cryptosystem itself.

First, suppose that Alice wants to trick Bob into signing a message $m$. She computes two other messages $m_1$ and $m_2$ such that $m_1 m_2 \bmod n_{Bob} = m$. She has Bob sign $m_1$ and $m_2$. Alice then multiplies the two signatures together $\bmod n_{Bob}$, giving Bob's signature on $m$ (see Exercise 13). The defense is to not sign random

documents and, when signing, never sign the document itself; sign a cryptographic hash of the document [1684].

EXAMPLE: Let $n_{Alice} = 262631$, $e_{Alice} = 154993$, $d_{Alice} = 95857$, $n_{Bob} = 288329$, $e_{Bob} = 22579$, and $d_{Bob} = 138091$. Alice and Bob have many possible contracts, each represented by three letters, from which they are to select and sign one.

Alice first asks Bob to sign the sequence 225536, so she can validate his signature. Bob computes

$$225536^{138091} \bmod 288329 = 271316$$

Alice then asks Bob to sign contract "AYE" (002404):

$$002404^{138091} \bmod 288329 = 182665$$

Alice now computes

$$(002404)(225536) \bmod 288329 = 130024$$

She then claims that Bob agreed to contract "NAY" (130024). She presents the signature

$$(271316)(182665) \bmod 288329 = 218646$$

Judge Janice is called, and she computes

$$218646^{22579} \bmod 288329 = 130024$$

Naturally, Janice concludes that Bob is lying, because his public key deciphers the signature. So Alice has successfully tricked Bob.

Enciphering a message and then signing it creates a second problem [60]. Suppose Alice is sending Bob her signature on a confidential contract $m$. She enciphers it first, then signs it:

$$c = \left(m^{e_{Bob}} \bmod n_{Bob}\right)^{d_{Alice}} \bmod n_{Alice}$$

She then sends the result to Bob. However, Bob wants to claim that Alice sent him the contract $M$. Bob computes a number $r$ such that $M^r \bmod n_{Bob} = m$. He then republishes his public key as $(re_{Bob}, n_{Bob})$. Note that the modulus does not change. Now, he claims that Alice sent him $M$. The judge verifies this using his current public key. The simplest way to fix this is to require all users to use the same exponent but vary the moduli.

EXAMPLE: Smarting from Alice's trick, Bob seeks revenge. He and Alice agree to sign the contract "LUR" (112017). Alice first enciphers it, then signs it:

$$(112017^{22579} \bmod 288329)^{95857} \bmod 262631 = 42390$$

She sends it to Bob. Bob, however, wants the contract to be "EWM" (042212). He computes an $r$ such that $042212^r \bmod 288329 = 112017$; one such $r$ is $r = 9175$. He then computes a new public key $re_{Bob} \bmod \phi(n_{Bob}) = (9175)(22579) \bmod 287184 = 102661$. He replaces his current public key with (102661, 288329), and resets his private key to 161245. He now claims that Alice sent him contract "EWM," signed by her.

Judge Janice is called. She takes the message 42390 and deciphers it:

$$(42390^{154993} \bmod 262631)^{161245} \bmod 288329 = 042212$$

She concludes that Bob is correct.

This attack will not work if one signs first and then enciphers. The reason is that Bob cannot access the information needed to construct a new public key, because he would need to alter Alice's public key (see Exercise 27).

However, signing first and then enciphering enables the recipient to decipher the signed message, re-encrypt it using a third party's public key, and then forward it to the third party. The third party then cannot tell if the original sender sent it directly to him. This is the *surreptitious forwarding attack*. Several solutions to providing enciphered, authenticated messages have been proposed [510, 1101]. One simple solution is to embed the signer's and recipient's names in the signed message. Another is to sign the message, encrypt it, and sign the result; a variant is to encrypt the message, then sign that, and then encrypt the result.

#### 10.5.2.2 El Gamal Digital Signature

This scheme is based on the El Gamal cryptosystem presented in Section 10.3.1. Recall that the generator $g$ is chosen so that $1 < g < p$, where $p$ is a prime number with $p - 1$ having a large factor; the private key $d$ is chosen so that $1 < d < p - 1$; and the public key is $(p, g, y)$, where $y = g^d \bmod p$.

Suppose Alice wants to send Bob a signed contract $m$. She chooses a number $k$ that is less than, and relatively prime to, $p - 1$ and has not been used before. She computes $a = g^k \bmod p$ and then uses the Extended Euclidean Algorithm (see Appendix B) to find $b$ such that

$$m = (da + kb) \bmod p - 1$$

The pair $(a, b)$ is the signature.