6

Signature Schemes

6.1 Introduction

In this chapter, we study *signature schemes*, which are also called digital signatures. A "conventional" handwritten signature attached to a document is used to specify the person responsible for it. A signature is used in everyday situations such a writing a letter, withdrawing money from a bank, signing a contract, etc.

A signature scheme is a method of signing a message stored in electronic form. As such, a signed message can be transmitted over a computer network. In this chapter, we will study several signature schemes, but first we discuss some fundamental differences between conventional and digital signatures.

First is the question of signing a document. With a conventional signature, a signature is physically part of the document being signed. However, a digital signature is not attached physically to the message that is signed, so the algorithm that is used must somehow "bind" the signature to the message.

Second is the question of verification. A conventional signature is verified by comparing it to other, authentic signatures. For example, when someone signs a credit card purchase, the salesperson is supposed to compare the signature on the sales slip to the signature on the back of the credit card in order to verify the signature. Of course, this is not a very secure method as it is relatively easy to forge someone else's signature. Digital signatures, on the other hand, can be verified using a publicly known verification algorithm. Thus, "anyone" can verify a digital signature. The use of a secure signature scheme will prevent the possibility of forgeries.

Another fundamental difference between conventional and digital signatures is that a "copy" of a signed digital message is identical to the original. On the other hand, a copy of a signed paper document can usually be distinguished from an original. This feature means that care must be taken to prevent a signed digital message from being reused. For example, if Bob signs a digital message authorizing Alice to withdraw $100 from his bank account (i.e., a check), he only wants Alice to be able to do so once. So the message itself should contain

information, such as a date, that prevents it from being reused.

A signature scheme consists of two components: a *signing algorithm* and a *verification algorithm*. Bob can sign a message x using a (secret) signing algorithm sig. The resulting signature $sig(x)$ can subsequently be verified using a public verification algorithm ver. Given a pair (x, y), the verification algorithm returns an answer "true" or "false" depending on whether the signature is authentic.

Here is a formal defintion of a signature scheme.

DEFINITION 6.1 *A **signature scheme** is a five-tuple $(\mathcal{P}, \mathcal{A}, \mathcal{K}, \mathcal{S}, \mathcal{V})$, where the following conditions are satisfied:*

1. *$\mathcal{P}$ is a finite set of possible **messages***
2. *$\mathcal{A}$ is a finite set of possible **signatures***
3. *$\mathcal{K}$, the **keyspace**, is a finite set of possible **keys***
4. *For each $K \in \mathcal{K}$, there is a **signing algorithm** $sig_K \in \mathcal{S}$ and a corresponding **verification algorithm** $ver_K \in \mathcal{V}$. Each $sig_K : \mathcal{P} \rightarrow \mathcal{A}$ and $ver_K : \mathcal{P} \times \mathcal{A} \rightarrow \{\text{true}, \text{false}\}$ are functions such that the following equation is satisfied for every message $x \in \mathcal{P}$ and for every signature $y \in \mathcal{A}$:*

$$ver(x, y) = \begin{cases} \text{true} & \text{if } y = sig(x) \\ \text{false} & \text{if } y \neq sig(x). \end{cases}$$

For every $K \in \mathcal{K}$, the functions sig_K and ver_K should be polynomial-time functions. ver_K will be a public function and sig_K will be secret. It should be computationally infeasible for Oscar to "forge" Bob's signature on a message x. That is, given x, only Bob should be able to compute the signature y such that $ver(x, y) = \text{true}$. A signature scheme cannot be unconditionally secure, since Oscar can test all possible signatures y for a message x using the public algorithm ver, until he finds the right signature. So, given sufficient time, Oscar can always forge Bob's signature. Thus, as was the case with public-key cryptosystems, our goal is to find signature schemes that are computationally secure.

As our first example of a signature scheme, we observe that the **RSA** public-key cryptosystem can be used to provide digital signatures. See Figure 6.1.

Thus, Bob signs a message x using the **RSA** decryption rule d_K. Bob is the only person that can create the signature since $d_K = sig_K$ is secret. The verification algorithm uses the **RSA** encryption rule e_K. Anyone can verify a signature since e_K is public.

Note that anyone can forge Bob's signature on a "random" message x by computing $x = e_K(y)$ for some y; then $y = sig_K(x)$. One way around this difficulty is to require that messages contain sufficient redundancy that a forged signature of this type does not correspond to a "meaningful" message x except with a very small probability. Alternatively, the use of hash functions in conjunction with signature schemes will eliminate this method of forging (cryptographic hash functions will be discussed in Chapter 7).

FIGURE 6.1
RSA Signature Scheme

Let $n = pq$, where p and q are primes. Let $\mathcal{P} = \mathcal{A} = \mathbb{Z}_n$, and define

$$\mathcal{K} = \{(n, p, q, a, b) : n = pq, p, q \text{ prime}, ab \equiv 1 \pmod{\phi(n)}\}.$$

The values n and b are public, and the values p, q, a are secret.

For $K = (n, p, q, a, b)$, define

$$sig_K(x) = x^a \bmod n$$

and

$$ver_K(x, y) = \text{true} \Leftrightarrow x \equiv y^b \pmod{n}$$

$(x, y \in \mathbb{Z}_n)$.

Finally, let's look briefly at how we would combine signing and public-key encryption. Suppose Alice wishes to send a signed, encrypted message to Bob. Given a plaintext x, Alice would compute her signature $y = sig_{\text{Alice}}(x)$, and then encrypt both x and y using Bob's public encryption function e_{Bob}, obtaining $z = e_{\text{Bob}}(x, y)$. The ciphertext z would be transmitted to Bob. When Bob receives z, he first decrypts it with his decryption function d_{Bob} to get (x, y). Then he uses Alice's public verification function to check that $ver_{\text{Alice}}(x, y) = \text{true}$.

What if Alice first encrypted x, and then signed the result? Then she would compute

$$y = sig_{\text{Alice}}(e_{\text{Bob}}(x)).$$

Alice would transmit the pair (z, y) to Bob. Bob would decrypt z, obtaining x, and then verify the signature y on x using ver_{Alice}. One potential problem with this approach is that if Oscar obtains a pair (z, y) of this type, he could replace Alice's signature y by his own signature

$$y' = sig_{\text{Oscar}}(e_{\text{Bob}}(x)).$$

(Note that Oscar can sign the ciphertext $e_{\text{Bob}}(x)$ even though he doesn't know the plaintext x.) Then, if Oscar transmits (z, y') to Bob, Oscar's signature will be verified by Bob using ver_{Oscar}, and Bob may infer that the plaintext x originated with Oscar. Because of this potential difficulty, most people recommend signing before encrypting.

FIGURE 6.2
ElGamal Signature Scheme

Let p be a prime such that the discrete log problem in $\mathbb{Z}_p$ is intractible, and let $\alpha \in \mathbb{Z}_p^*$ be a primitive element. Let $\mathcal{P} = \mathbb{Z}_p^*$, $\mathcal{A} = \mathbb{Z}_p^* \times \mathbb{Z}_{p-1}$, and define

$$\mathcal{K} = \{(p, \alpha, a, \beta) : \beta \equiv \alpha^a \pmod{p}\}.$$

The values p, α and β are public, and a is secret.

For $K = (p, \alpha, a, \beta)$, and for a (secret) random number $k \in \mathbb{Z}_{p-1}^*$, define

$$sig_K(x, k) = (\gamma, \delta),$$

where

$$\gamma = \alpha^k \bmod p$$

and

$$\delta = (x - a\gamma)k^{-1} \bmod (p-1).$$

For $x, \gamma \in \mathbb{Z}_p^*$ and $\delta \in \mathbb{Z}_{p-1}$, define

$$ver(x, \gamma, \delta) = \text{true} \Leftrightarrow \beta^\gamma \gamma^\delta \equiv \alpha^x \pmod{p}.$$

6.2 The ElGamal Signature Scheme

We now describe the **ElGamal Signature Scheme**, which was described in a 1985 paper. A modification of this scheme has been adopted as a digital signature standard by the National Institute of Standards and Technology (NIST). The **ElGamal Scheme** is designed specifically for the purpose of signatures, as opposed to **RSA**, which can be used both as a public-key cryptosystem and a signature scheme.

The **ElGamal Signature Scheme** is non-deterministic, as was the **ElGamal Public-key Cryptosystem**. This means that there are many valid signatures for any given message. The verification algorithm must be able to accept any of the valid signatures as authentic. The description of the **ElGamal Signature Scheme** is given in Figure 6.2.

If the signature was constructed correctly, then the verification will succeed, since

$$\beta^\gamma \gamma^\delta \equiv \alpha^{a\gamma} \alpha^{k\delta} \pmod{p}$$

$$\equiv \alpha^x \pmod{p},$$

where we use the fact that

$$a\gamma + k\delta \equiv x \pmod{p-1}.$$

Bob computes a signature using both the secret value a (which is part of the key) and the secret random number k (which is used to sign one message, x). The verification can be accomplished using only public information.

Let's do a small example to illustrate the arithmetic.

Example 6.1
Suppose we take $p = 467$, $\alpha = 2$, $a = 127$; then

$$\begin{aligned}\beta &= \alpha^a \bmod p \\ &= 2^{127} \bmod 467 \\ &= 132.\end{aligned}$$

Suppose Bob wants to sign the message $x = 100$ and he chooses the random value $k = 213$ (note that $\gcd(213, 466) = 1$ and $213^{-1} \bmod 466 = 431$). Then

$$\gamma = 2^{213} \bmod 467 = 29$$

and

$$\delta = (100 - 127 \times 29)431 \bmod 466 = 51.$$

Anyone can verify this signature by checking that

$$132^{29}29^{51} \equiv 189 \pmod{467}$$

and

$$2^{100} \equiv 189 \pmod{467}.$$

Hence, the signature is valid. □

Let's look at the security of the **ElGamal Signature Scheme**. Suppose Oscar tries to forge a signature for a given message x, without knowing a. If Oscar chooses a value γ and then tries to find the corresponding δ, he must compute the discrete logarithm $\log_\gamma \alpha^x \beta^{-\gamma}$. On the other hand, if he first chooses δ and then tries to find γ, he is trying to "solve" the equation

$$\beta^\gamma \gamma^\delta \equiv \alpha^x \pmod{p}$$

for the "unknown" γ. This is a problem for which no feasible solution is known; however, it does not seem to be related to any well-studied problem such as the

Discrete Logarithm problem. There also remains the possibility that there might be some way to compute γ and δ simultaneously in such a way that (γ, δ) will be a signature. No one has discovered a way to do this, but conversely, no one has proved that it cannot be done.

If Oscar chooses γ and δ and then tries to solve for x, he is again faced with an instance of the **Discrete Logarithm** problem, namely the computation of $\log_\alpha \beta^\gamma \gamma^\delta$. Hence, Oscar cannot sign a "random" message using this approach. However, there is a method by which Oscar can sign a random message by choosing γ, δ and x simultaneously: Suppose i and j are integers, $0 \leq i \leq p-2$, $0 \leq j \leq p-2$, and $\gcd(j, p-1) = 1$. Then perform the following computations:

$$\gamma = \alpha^i \beta^j \bmod p$$
$$\delta = -\gamma j^{-1} \bmod (p-1)$$
$$x = -\gamma i j^{-1} \bmod (p-1),$$

where j^{-1} is computed modulo $(p-1)$ (this is where we require that j be relatively prime to $p-1$).

We claim that (γ, δ) is a valid signature for the message x. This is proved by checking the verification condition:

$$\begin{aligned}
\beta^\gamma \gamma^\delta &\equiv \beta^{\alpha^i \beta^j} (\alpha^i \beta^j)^{-\alpha^i \beta^j j^{-1}} \pmod p \\
&\equiv \beta^{\alpha^i \beta^j} \alpha^{-ij^{-1}\alpha^i \beta^j} \beta^{-\alpha^i \beta^j} \pmod p \\
&\equiv \alpha^{-ij^{-1}\alpha^i \beta^j} \pmod p \\
&\equiv \alpha^{-\gamma i j^{-1}} \pmod p \\
&\equiv \alpha^x \pmod p.
\end{aligned}$$

We illustrate with an example.

Example 6.2
As in the previous example, suppose $p = 467$, $\alpha = 2$ and $\beta = 132$. Suppose Oscar chooses $i = 99$ and $j = 179$; then $j^{-1} \bmod (p-1) = 151$. He would compute the following:

$$\begin{array}{rcll}
\gamma & = & 2^{99} 132^{179} \bmod 467 & = 117 \\
\delta & = & -117 \times 151 \bmod 466 & = 41 \\
x & = & 99 \times 41 \bmod 466 & = 331.
\end{array}$$

Then $(117, 41)$ is a valid signature for the message 331, as may be verified by checking that

$$132^{117} 117^{41} \equiv 303 \pmod{467}$$

and

$$2^{331} \equiv 303 \pmod{467}.$$

Hence, the signature is valid. □

Here is a second type of forgery, in which Oscar begins with a message previously signed by Bob. Suppose (γ, δ) is a valid signature for a message x. Then it is possible for Oscar to sign various other messages. Suppose h, i and j are integers, $0 \leq h, i, j \leq p-2$, and $\gcd(h\gamma - j\delta, p-1) = 1$. Compute the following:

$$\lambda = \gamma^h \alpha^i \beta^j \bmod p$$
$$\mu = \delta\lambda(h\gamma - j\delta)^{-1} \bmod (p-1)$$
$$x' = \lambda(hx + i\delta)(h\gamma - j\delta)^{-1} \bmod (p-1),$$

where $(h\gamma - j\delta)^{-1}$ is computed modulo $(p-1)$. Then, it is tedious but straightforward to check the verification condition:

$$\beta^\lambda \lambda^\mu \equiv \alpha^{x'} \pmod{p}.$$

Hence (λ, μ) is a valid signature for x'.

Both of these methods produce valid forged signatures, but they do not appear to enable an opponent to forge a signature on a message of his own choosing without first solving a discrete logarithm problem. Hence, they do not seem to represent a threat to the security of the **ElGamal Signature Scheme**.

Finally, we mention a couple of ways in which the **ElGamal Scheme** can be broken if it is used carelessly (these are further examples of protocol failures, some of which were discussed in the exercises of Chapter 4). First, the random value k used in computing a signature should not be revealed. For, if k is known, it is a simple matter to compute

$$a = (x - k\gamma)\delta^{-1} \bmod (p-1).$$

Of course, once a is known, then the system is broken and Oscar can forge signatures at will.

Another misuse of the system is to use the same value k in signing two different messages. This also makes it easy for Oscar to compute a and hence break the system. This can be done as follows. Suppose (γ, δ_1) is a signature on x_1 and (γ, δ_2) is a signature on x_2. Then we have

$$\beta^\gamma \gamma^{\delta_1} \equiv \alpha^{x_1} \pmod{p}$$

and

$$\beta^\gamma \gamma^{\delta_2} \equiv \alpha^{x_2} \pmod{p}.$$

Thus

$$\alpha^{x_1-x_2} \equiv \gamma^{\delta_2-\delta_1} \pmod{p}.$$

Writing $\gamma = \alpha^k$, we obtain the following equation in the unknown k:

$$\alpha^{x_1-x_2} \equiv \alpha^{k(\delta_2-\delta_1)} \pmod{p},$$

which is equivalent to

$$x_1 - x_2 \equiv k(\delta_2 - \delta_1) \pmod{p-1}.$$

Now let $d = \gcd(\delta_2 - \delta_1, p-1)$. Since $d \mid (p-1)$ and $d \mid (\delta_2 - \delta_1)$, it follows that $d \mid (x_1 - x_2)$. Define

$$x' = \frac{x_1 - x_2}{d}$$
$$\delta' = \frac{\delta_2 - \delta_1}{d}$$
$$p' = \frac{p-1}{d}.$$

Then the congruence becomes:

$$x' \equiv k\delta' \pmod{p'}.$$

Since $\gcd(\delta', p') = 1$, we can compute

$$\epsilon = (\delta')^{-1} \bmod p'.$$

Then value of k is determined modulo p' to be

$$k = x'\epsilon \bmod p'.$$

This yields d candidate values for k:

$$k = x'\epsilon + ip' \bmod p$$

for some i, $0 \leq i \leq d-1$. Of these d candidate values, the (unique) correct one can be determined by testing the condition

$$\gamma \equiv \alpha^k \pmod{p}.$$

6.3 The Digital Signature Standard

The **Digital Signature Standard** (or **DSS**) is a modification of the **ElGamal Signature Scheme**. It was published in the Federal Register on May 19, 1994

and adopted as a standard on December 1, 1994 (however, it was first proposed in August, 1991). First, we want to motivate the changes that are made to **ElGamal**, and then we will describe how they are accomplished.

In many situations, a message might be encrypted and decrypted only once, so it suffices to use any cryptosystem which is known to be secure at the time the message is encypted. On the other hand, a signed message could function as a legal document such as a contract or will, so it is very likely that it would be necessary to verify a signature many years after the message is signed. So it is important to take even more precautions regarding the security of a signature scheme as opposed to a cryptosystem. Since the **ElGamal Scheme** is no more secure than the **Discrete Logarithm** problem, this necessitates the use of a large modulus p. Certainly p should have at least 512 bits, and many people would argue that the length of p should be 1024 bits in order to provide security into the forseeable future.

However, even a 512 bit modulus leads to a signature having 1024 bits. For potential applications, many of which involve the use of smart cards, a shorter signature is desirable. **DSS** modifies the **ElGamal Scheme** in an ingenious way so that a 160-bit message is signed using a 320-bit signature, but the computations are done using a 512-bit modulus p. The way that this done is to work in a subgroup of $\mathbb{Z}_p^*$ of size 2^{160}. The assumed security of the scheme is based on the belief that finding discrete logarithms in this specified subgroup of $\mathbb{Z}_p^*$ is secure.

The first change we make is to change the "$-$" to a "$+$" in the definition of δ, so

$$\delta = (x + \alpha\gamma)k^{-1} \bmod (p-1).$$

This changes the verification condition to the following:

$$\alpha^x \beta^\gamma \equiv \gamma^\delta \pmod{p}. \tag{6.1}$$

If $\gcd(x + \alpha\gamma, p-1) = 1$, then $\delta^{-1} \bmod (p-1)$ exists, and we can modify condition (6.1), producing the following:

$$\alpha^{x\delta^{-1}} \beta^{\gamma\delta^{-1}} \equiv \gamma \pmod{p}. \tag{6.2}$$

Now here is the major innovation in the **DSS**. We suppose that q is a 160-bit prime such that $q \mid (p-1)$, and α is a qth root of 1 modulo p. (It is easy to construct such an α: Let α_0 be a primitive element of $\mathbb{Z}_p$, and define $\alpha = \alpha_0^{(p-1)/q} \bmod p$.) Then β and γ will also be qth roots of 1. Hence, any exponents of α, β and γ can be reduced modulo q without affecting verification condition (6.2). The tricky point is that γ appears as an exponent on the left side of (6.2), and again — but not as an exponent — on the right side of (6.2). So if γ is reduced modulo q, then we must also reduce the entire left side of (6.2) modulo q in order to perform the verification. Observe that (6.1) will not work if the extra reductions modulo q are done. The complete description of the **DSS** is given in Figure 6.3.

FIGURE 6.3
Digital Signature Standard

Let p be a 512-bit prime such that the discrete log problem in $\mathbb{Z}_p$ is intractible, and let q be a 160-bit prime that divides $p-1$. Let $\alpha \in \mathbb{Z}_p^*$ be a qth root of 1 modulo p. Let $\mathcal{P} = \mathbb{Z}_p^*$, $\mathcal{A} = \mathbb{Z}_q \times \mathbb{Z}_q$, and define

$$\mathcal{K} = \{(p, q, \alpha, a, \beta) : \beta \equiv \alpha^a \pmod{p}\}.$$

The values p, q, α and β are public, and a is secret.

For $K = (p, q, \alpha, a, \beta)$, and for a (secret) random number k, $1 \leq k \leq q-1$, define

$$sig_K(x, k) = (\gamma, \delta),$$

where

$$\gamma = (\alpha^k \bmod p) \bmod q$$

and

$$\delta = (x + a\gamma)k^{-1} \bmod q.$$

For $x \in \mathbb{Z}_p^*$ and $\gamma, \delta \in \mathbb{Z}_q$, verification is done by performing the following computations:

$$e_1 = x\delta^{-1} \bmod q$$

$$e_2 = \gamma\delta^{-1} \bmod q$$

$$ver_K(x, \gamma, \delta) = \text{true} \Leftrightarrow (\alpha^{e_1}\beta^{e_2} \bmod p) \bmod q = \gamma.$$

Notice that is necessary that $\delta \not\equiv 0 \pmod{q}$ since the value $\delta^{-1} \bmod q$ is needed to verify the signature (this is analogous to the requirement that $\gcd(\delta, p-1) = 1$ when we modified (6.1) to obtain (6.2)). If Bob computes a value $\delta \equiv 0 \pmod{q}$ in the signing algorithm, he should reject it and construct a new signature with a new random k. We should point out that this is not likely to cause a problem in practice: the probability that $\delta \equiv 0 \pmod{q}$ is likely to be on the order of 2^{-160}, so for all intents and purposes it will almost never happen.

Here is a small example to illustrate.

Example 6.3
Suppose we take $q = 101$ and $p = 78q + 1 = 7879$. 3 is a primitive element in $\mathbb{Z}_{7879}$, so we can take

$$\alpha = 3^{78} \bmod 7879 = 170.$$

Suppose $a = 75$; then

$$\beta = \alpha^a \bmod 7879 = 4567.$$

Now, suppose Bob wants to sign the message $x = 1234$ and he chooses the random value $k = 50$, so

$$k^{-1} \bmod 101 = 99.$$

Then

$$\begin{aligned}\gamma &= (170^{50} \bmod 7879) \bmod 101 \\ &= 2518 \bmod 101 \\ &= 94\end{aligned}$$

and

$$\begin{aligned}\delta &= (1234 + 75 \times 94)99 \bmod 101 \\ &= 97.\end{aligned}$$

The signature $(94, 97)$ on the message 1234 is verified by the following computations:

$$\delta^{-1} = 97^{-1} \bmod 101 = 25$$

$$e_1 = 1234 \times 25 \bmod 101 = 45$$

$$e_2 = 94 \times 25 \bmod 101 = 27$$

$$(170^{45}4567^{27} \bmod 7879) \bmod 101 = 2518 \bmod 101 = 94.$$

Hence, the signature is valid. □

When the **DSS** was proposed in 1991, there were several criticisms put forward. One complaint was that the selection process by NIST was not public. The standard was developed by the National Security Agency (NSA) without the input of U. S. industry. Regardless of the merits of the resulting scheme, many people resented the "closed-door" approach.

Of the technical criticisms put forward, the most serious was that the size of the modulus p was fixed at 512 bits. Many people would prefer that the modulus size not be fixed, so that larger modulus sizes could be used if desired. In reponse to these comments, NIST altered the description of the standard so that a variety of

modulus sizes are allowed, namely, any modulus size divisible by 64, in the range from 512 to 1024 bits.

Another complaint about the **DSS** was that signatures can be generated considerably faster than they can be verified. In contrast, if **RSA** is used as a signature scheme and the public verification exponent is very small (say 3, for example), then verification can be performed much more quickly than signing. This leads to a couple of considerations concerning the potential applications of the signature scheme:

1. A message will only be signed once. On the other hand, it might be necessary to verify the signature many times over a period of years. This suggests that a faster verification algorithm would be desirable.
2. What types of computers are likely to be doing the signing and verifying? Many potential applications involve smart cards, with limited processing power, communicating with a more powerful computer. So one might try to design a scheme so that fewer computations are likely to be done by a card. But one can imagine situations where a smart card would generate a signature, and other situations where a smart card would verify a signature, so it is difficult to give a definitive answer here.

The response of NIST to the question of signature generation/verification times is that it does not really matter which is faster, provided that both can be done sufficiently quickly.

6.4 One-time Signatures

In this section, we describe a conceptually simple way to construct a one-time signature scheme from any one-way function. The term "one-time" means that only one message can be signed. (The signature can be verified an arbitrary number of times, of course.) The description of the scheme, known as the **Lamport Signature Scheme**, is given in Figure 6.4.

Informally, this is how the system works. A message to be signed is a binary k-tuple. Each bit is signed individually: the value $z_{i,j}$ corresponds to the ith bit of the message having the value j ($j = 0, 1$). Each $z_{i,j}$ is the image of $y_{i,j}$ under the one-way function f. The ith bit of the message is signed using the preimage $y_{i,j}$ of the $z_{i,j}$ corresponding to the ith bit of the message. The verification consists simply of checking that the each element in the signature is the preimage of the appropriate public key element.

We illustrate the scheme by considering one possible implementation using the exponentiation function $f(x) = \alpha^x \bmod p$, where α is a primitive element modulo p.

FIGURE 6.4
Lamport Signature Scheme

Let k be a positive integer and let $\mathcal{P} = \{0,1\}^k$. Suppose $f : Y \to Z$ is a one-way function, and let $\mathcal{A} = Y^k$. Let $y_{i,j} \in Y$ be chosen at random, $1 \leq i \leq k$, $j = 0, 1$, and let $z_{i,j} = f(y_{i,j})$, $1 \leq i \leq k$, $j = 0, 1$. The key K consists of the $2k$ y's and the $2k$ z's. The y's are secret while the z's are public.

For $K = (y_{i,j}, z_{i,j} : 1 \leq i \leq k, j = 0, 1)$, define

$$sig_K(x_1, \ldots, x_k) = (y_{1,x_1}, \ldots, y_{k,x_k})$$

and

$$ver_K(x_1, \ldots, x_k, a_1, \ldots a_k) = \text{true} \Leftrightarrow f(a_i) = z_{i,x_i}, 1 \leq i \leq k.$$

Example 6.4
7879 is prime and 3 is a primitive element in $\mathbb{Z}_{7879}$. Define

$$f(x) = 3^x \bmod 7879.$$

Suppose Bob wishes to sign a message of three bits, and he chooses the six (secret) random numbers

$$y_{1,0} = 5831$$
$$y_{1,1} = 735$$
$$y_{2,0} = 803$$
$$y_{2,1} = 2467$$
$$y_{3,0} = 4285$$
$$y_{3,1} = 6449.$$

Then he computes the images of the y's under the function f:

$$z_{1,0} = 2009$$
$$z_{1,1} = 3810$$
$$z_{2,0} = 4672$$
$$z_{2,1} = 4721$$
$$z_{3,0} = 268$$

$$z_{3,1} = 5731.$$

These z's are published. Now, suppose Bob wants to sign the message

$$x = (1, 1, 0).$$

The signature for x is

$$(y_{1,1}, y_{2,1}, y_{3,0}) = (735, 2467, 4285).$$

To verify this signature, it suffices to compute the following:

$$3^{735} \bmod 7879 = 3810$$
$$3^{2467} \bmod 7879 = 4721$$
$$3^{4285} \bmod 7879 = 268.$$

Hence, the signature is valid. □

Oscar cannot forge a signature because he is unable to invert the one-way function f to obtain the secret y's. However, the signature scheme can be used to sign only one message. For, given signatures for two different messages, it is (usually) an easy matter for Oscar to construct signatures for further messages (different from the first two).

For example, suppose the messages $(0, 1, 1)$ and $(1, 0, 1)$ are both signed using the same scheme. The message $(0, 1, 1)$ would have as its signature the triple $(y_{1,0}, y_{2,1}, y_{3,1})$, and the message $(1, 0, 1)$ would be signed with $(y_{1,1}, y_{2,0}, y_{3,1})$. Given these two signatures, Oscar can construct signatures for the messages $(1, 1, 1)$ (namely, $(y_{1,1}, y_{2,1}, y_{3,1})$) and $(0, 0, 1)$ (namely, $(y_{1,0}, y_{2,0}, y_{3,1})$).

Even though this scheme is quite elegant, it is not of great practical use due to the size of the signatures it produces. For example, if we use the modular exponentiation function, as in the example above, then a secure implementation would require that p be at least 512 bits in length. This means that each bit of the message is signed using 512 bits. Consequently, the signature is 512 times as long as the message!

We now look at a modification due to Bos and Chaum that allows the signatures to be made somewhat shorter, with no loss of security. In the **Lamport Scheme**, the reason that Oscar cannot forge a signature on a (second) message, given a signature on one message, is that the y's corresponding to one message are never a subset of the y's corresponding to another (distinct) message.

Suppose we have a set $\mathcal{B}$ of subsets of a set B such that $B_1 \subseteq B_2$ only if $B_1 = B_2$, for all $B_1, B_2 \in \mathcal{B}$. Then $\mathcal{B}$ is said to satisfy the *Sperner property*. Given a set B of even cardinality n, it is known that the maximum size of a set $\mathcal{B}$ of subsets of B having the Sperner property is $\binom{2n}{n}$. This can easily be obtained by taking all the n-subsets of B: clearly no n-subset is contained in another n-subset.

FIGURE 6.5
Bos-Chaum Signature Scheme

Let k be a positive integer and let $\mathcal{P} = \{0,1\}^k$. Let n be an integer such that $2^k \leq \binom{2n}{n}$, let B be a set of cardinality n, and let

$$\phi : \{0,1\}^k \to \mathcal{B}$$

be an injection, where $\mathcal{B}$ is the set of all n-subsets of B. Suppose $f : Y \to Z$ is a one-way function, and let $\mathcal{A} = Y^n$. Let $y_i \in Y$ be chosen at random, $1 \leq i \leq 2n$, and let $z_i = f(y_i)$, $1 \leq i \leq 2n$. The key K consists of the $2n$ y's and the $2n$ z's. The y's are secret while the z's are public.

For $K = (y_i, z_i, 1 \leq i \leq 2n)$, define

$$sig_K(x_1, \ldots, x_k) = \{y_j : j \in \phi(x_1, \ldots, x_k)\}$$

and

$$ver_K(x_1, \ldots, x_k, a_1, \ldots a_n) = \text{true}$$
$$\Leftrightarrow \{f(a_i) : 1 \leq i \leq n\} = \{z_j : j \in \phi(x_1, \ldots, x_k)\}.$$

Now suppose we want to sign a k-bit message, as before, and we choose n large enough so that

$$2^k \leq \binom{2n}{n}.$$

Let $|B| = n$ and let $\mathcal{B}$ denote the set of n-subsets of B. Let $\phi : \{0,1\}^k \to \mathcal{B}$ be a publicly known injection. Then we can associate each possible message with an n-subset in $\mathcal{B}$. We will have $2n$ y's and $2n$ z's, and each message will be signed with n y's. The complete description of the **Bos-Chaum Scheme** is given in Figure 6.5.

The advantage of the **Bos-Chaum Scheme** is that signatures are shorter than with the **Lamport Scheme**. For example, suppose we wish to sign a message of six bits (i.e., $k = 6$). Since $2^6 = 64$ and $\binom{8}{4} = 70$, we can take $n = 4$. This allows a six-bit message to be signed with four y's, as opposed to six with **Lamport**. As well, the key is shorter, consisting of eight z's as opposed to twelve with **Lamport**.

The **Bos-Chaum Scheme** requires an injective function ϕ that associates an n-subset of a $2n$-set with each possible binary k-tuple $x = (x_1, \ldots, x_k)$. We present one simple algorithm to do this in Figure 6.6. Applying this algorithm with $x = (0,1,0,0,1,1)$, for example, yields

$$\phi(x) = \{2,4,6,8\}.$$

FIGURE 6.6
Computation of ϕ in the Bos-Chaum Scheme

```
1.  X = Σ_{i=1}^{k} x_i 2^{i-1}
2.  φ(x) = ∅
3.  t = 2n
4.  e = n
5.  while t > 0 do
6.      t = t − 1
7.      if x > (t choose e) then
8.          x = x − (t choose e)
9.          e = e − 1
10.         φ(x) = φ(x) ∪ {t + 1}.
```

In general, how big is n in the **Bos-Chaum Scheme** as compared to k? We need to satisfy the inequality $2^k \leq \binom{2n}{n}$. If we estimate the binomial coefficient

$$\binom{2n}{n} = \frac{(2n)!}{(n!)^2}$$

using Stirling's formula, we obtain the quantity $2^{2n}/\sqrt{\pi n}$. After some simplification, the inequality becomes

$$k \leq 2n - \frac{\log_2(n\pi)}{2}.$$

Asymptotically, n is about $k/2$, so we obtain an almost 50% reduction in signature size by using the **Bos-Chaum Scheme**.

6.5 Undeniable Signatures

Undeniable signatures were introduced by Chaum and van Antwerpen in 1989. They have several novel features. Primary among these is that a signature cannot be verified without the cooperation of the signer, Bob. This protects Bob against the possibility that documents signed by him are duplicated and distributed electronically without his approval. The verification will be accomplished by means of a *challenge-and-response protocol.*

FIGURE 6.7
Chaum-van Antwerpen Undeniable Signature Scheme

Let $p = 2q + 1$ be a prime such that q is prime and the discrete log problem in $\mathbb{Z}_p$ is intractible. Let $\alpha \in \mathbb{Z}_p{}^*$ be an element of order q. Let $1 \leq a \leq q-1$ and define $\beta = \alpha^a \bmod p$. Let G denote the multiplicative subgroup of $\mathbb{Z}_p{}^*$ of order q (G consists of the quadratic residues modulo p). Let $\mathcal{P} = \mathcal{A} = G$, and define

$$\mathcal{K} = \{(p, \alpha, a, \beta) : \beta \equiv \alpha^a \pmod{p}\}.$$

The values p, α and β are public, and a is secret.

For $K = (p, \alpha, a, \beta)$ and $x \in G$, define

$$y = sig_K(x) = x^a \bmod p.$$

For $x, y \in G$, verification is done by executing the following protocol:

1. Alice chooses e_1, e_2 at random, $e_1, e_2 \in \mathbb{Z}_q{}^*$.
2. Alice computes $c = y^{e_1}\beta^{e_2} \bmod p$ and sends it to Bob.
3. Bob computes $d = c^{a^{-1} \bmod q} \bmod p$ and sends it to Alice.
4. Alice accepts y as a valid signature if and only if

$$d \equiv x^{e_1}\alpha^{e_2} \pmod{p}.$$

But if Bob's cooperation is required to verify a signature, what is to prevent Bob from disavowing a signature he made at an earlier time? Bob might claim that a valid signature is a forgery, and either refuse to verify it, or carry out the protocol in such a way that the signature will not be verified. To prevent this from happening, an undeniable signature scheme incorporates a *disavowal protocol* by which Bob can prove that a signature is a forgery. Thus, Bob will be able to prove in court that a given forged signature is in fact a forgery. (If he refuses to take part in the disavowal protocol, this would be regarded as evidence that the signature is, in fact, genuine.)

Thus, an undeniable signature scheme consists of three components: a signing algorithm, a verification protocol, and a disavowal protocol. First, we present the signing algorithm and verification protocol of the **Chaum-van Antwerpen Undeniable Signature Scheme** in Figure 6.7.

We should explain the roles of p and q in this scheme. The scheme lives in $\mathbb{Z}_p$; however, we need to be able to do computations in a multiplicative subgroup G of $\mathbb{Z}_p{}^*$ of prime order. In particular, we need to be able to compute inverses modulo

$|G|$, which is why $|G|$ should be prime. It is convenient to take $p = 2q + 1$ where q is prime. In this way, the subgroup G is as large as possible, which is desirable since messages and signatures are both elements of G.

We first prove that Alice will accept a valid signature. In the following computations, all exponents are to be reduced modulo q. First, observe that

$$\begin{aligned} d &\equiv c^{a^{-1}} \pmod{p} \\ &\equiv y^{e_1 a^{-1}} \beta^{e_2 a^{-1}} \pmod{p}. \end{aligned}$$

Since

$$\beta \equiv \alpha^a \pmod{p},$$

we have that

$$\beta^{a^{-1}} \equiv \alpha \pmod{p}.$$

Similarly,

$$y = x^a \pmod{p}$$

implies that

$$y^{a^{-1}} \equiv x \pmod{p}.$$

Hence,

$$d \equiv x^{e_1} \alpha^{e_2} \pmod{p},$$

as desired.

Here is a small example.

Example 6.5

Suppose we take $p = 467$. Since 2 is a primitive element, $2^2 = 4$ is a generator of G, the quadratic residues modulo 467. So we can take $\alpha = 4$. Suppose $a = 101$; then

$$\beta = \alpha^a \bmod 467 = 449.$$

Bob will sign the message $x = 119$ with the signature

$$y = 119^{101} \bmod 467 = 129.$$

Now, suppose Alice wants to verify the signature y. Suppose she chooses the random values $e_1 = 38$, $e_2 = 397$. She will compute $c = 13$, whereupon Bob will respond with $d = 9$. Alice checks the resonse by verifying that

$$119^{38} 4^{397} \equiv 9 \pmod{467}.$$

Hence, Alice accepts the signature as valid. □

We next prove that Bob cannot fool Alice into accepting a fradulent signature as valid, except with a very small probability. This result does not depend on any computational assumptions, i.e., the security is unconditional.

THEOREM 6.1
If $y \not\equiv x^a \pmod{p}$, *then Alice will accept* y *as a valid signature for* x *with probability* $1/q$.

PROOF First, we observe that each possible challenge c corresponds to exactly q ordered pairs (e_1, e_2) (this is because y and β are both elements of the multiplicative group G of prime order q). Now, when Bob receives the challenge c, he has no way of knowing which of the q possible ordered pairs (e_1, e_2) Alice used to construct c. We claim that, if $y \not\equiv x^a \pmod{p}$, then any possible response $d \in G$ that Bob might make is consistent with exactly one of the q possible ordered pairs (e_1, e_2).

Since α generates G, we can write any element of G as a power of α, where the exponent is defined uniquely modulo q. So write $c = \alpha^i$, $d = \alpha^j$, $x = \alpha^k$, and $y = \alpha^\ell$, where $i, j, k, \ell \in \mathbb{Z}_q$ and all arithmetic is modulo p. Consider the following two congruences:

$$c \equiv y^{e_1}\beta^{e_2} \pmod{p}$$
$$d \equiv x^{e_1}\alpha^{e_2} \pmod{p}.$$

This system is equivalent to the following system:

$$i \equiv \ell e_1 + a e_2 \pmod{q}$$
$$j \equiv k e_1 + e_2 \pmod{q}.$$

Now, we are assuming that

$$y \not\equiv x^a \pmod{p},$$

so it follows that

$$\ell \not\equiv ak \pmod{q}.$$

Hence, the coefficient matrix of this system of congruences modulo q has nonzero determinant, and thus there is a unique solution to the system. That is, every $d \in G$ is the correct response for exactly one of the q possible ordered pairs (e_1, e_2). Consequently, the probability that Bob gives Alice a response d that will be verified is exactly $1/q$, and the theorem is proved. ∎

We now turn to the disavowal protocol. This protocol consists of two runs of the verification protocol and is presented in Figure 6.8.

Steps 1–4 and steps 5–8 comprise two unsuccessful runs of the verification protocol. Step 9 is a "consistency check" that enables Alice to determine if Bob is forming his responses in the manner specified by the protocol.

FIGURE 6.8
Disavowal protocol

1. Alice chooses e_1, e_2 at random, $e_1, e_2 \in \mathbb{Z}_q^*$
2. Alice computes $c = y^{e_1}\beta^{e_2} \bmod p$ and sends it to Bob
3. Bob computes $d = c^{a^{-1} \bmod q} \bmod p$ and sends it to Alice
4. Alice verifies that $d \not\equiv x^{e_1}\alpha^{e_2} \pmod p$
5. Alice chooses f_1, f_2 at random, $f_1, f_2 \in \mathbb{Z}_q^*$
6. Alice computes $C = y^{f_1}\beta^{f_2} \bmod p$ and sends it to Bob
7. Bob computes $D = C^{a^{-1} \bmod q} \bmod p$ and sends it to Alice
8. Alice verifies that $D \not\equiv x^{f_1}\alpha^{f_2} \pmod p$
9. Alice concludes that y is a forgery if and only if

$$(d\alpha^{-e_2})^{f_1} \equiv (D\alpha^{-f_2})^{e_1} \pmod p.$$

The following example illustrates the disavowal protocol.

Example 6.6
As before, suppose $p = 467$, $\alpha = 4$, $a = 101$ and $\beta = 449$. Suppose the message $x = 286$ is signed with the (bogus) signature $y = 83$, and Bob wants to convince Alice that the signature is invalid.

Suppose Alice begins by choosing the random values $e_1 = 45$, $e_2 = 237$. Alice computes $c = 305$ and Bob responds with $d = 109$. Then Alice computes

$$286^{45}4^{237} \bmod 467 = 149.$$

Since $149 \neq 109$, Alice proceeds to step 5 of the protocol.

Now suppose Alice chooses the random values $f_1 = 125$, $f_2 = 9$. Alice computes $C = 270$ and Bob responds with $D = 68$. Alice computes

$$286^{125}4^{9} \bmod 467 = 25.$$

Since $25 \neq 68$, Alice proceeds to step 9 of the protocol and performs the consistency check. This check succeeds, since

$$(109 \times 4^{-237})^{125} \equiv 188 \pmod{467}$$

and

$$(68 \times 4^{-9})^{45} \equiv 188 \pmod{467}.$$

Hence, Alice is convinced that the signature is invalid. □

We have to prove two things at this point:

1. Bob can convince Alice that an invalid signature is a forgery.
2. Bob cannot make Alice believe that a valid signature is a forgery except with a very small probability.

THEOREM 6.2
If $y \not\equiv x^a \pmod{p}$, *and Alice and Bob follow the disavowal protocol, then*

$$(d\alpha^{-e_2})^{f_1} \equiv (D\alpha^{-f_2})^{e_1} \pmod{p}.$$

PROOF Using the facts that

$$d \equiv c^{a^{-1}} \pmod{p},$$

$$c \equiv y^{e_1}\beta^{e_2} \pmod{p}$$

and

$$\beta \equiv \alpha^a \pmod{p},$$

we have that

$$\begin{aligned}(d\alpha^{-e_2})^{f_1} &\equiv \left((y^{e_1}\beta^{e_2})^{a^{-1}}\alpha^{-e_2}\right)^{f_1} \pmod{p} \\ &\equiv y^{e_1 f_1}\beta^{e_2 a^{-1} f_1}\alpha^{-e_2 f_1} \pmod{p} \\ &\equiv y^{e_1 f_1}\alpha^{e_2 f_1}\alpha^{-e_2 f_1} \pmod{p} \\ &\equiv y^{e_1 f_1} \pmod{p}.\end{aligned}$$

A similar computation, using the facts that $D \equiv C^{a^{-1}} \pmod{p}$, $C \equiv y^{f_1}\beta^{f_2} \pmod{p}$ and $\beta \equiv \alpha^a \pmod{p}$, establishes that

$$(D\alpha^{-f_2})^{e_1} \equiv y^{e_1 f_1} \pmod{p},$$

so the consistency check in step 9 succeeds. ▮

Now we look that the possibility that Bob might attempt to disavow a valid signature. In this situation, we do not assume that Bob follows the protocol. That is, Bob might not construct d and D as specified by the protocol. Hence, in the following theorem, we assume only that Bob is able to produce values d and D which satisfy the conditions in steps 4, 8, and 9 of the protocol presented in Figure 6.8.

THEOREM 6.3
Suppose $y \equiv x^a \pmod{p}$ *and Alice follows the disavowal protocol. If*

$$d \not\equiv x^{e_1}\alpha^{e_2} \pmod{p}$$

and

$$D \not\equiv x^{f_1}\alpha^{f_2} \pmod{p},$$

then the probability that

$$(d\alpha^{-e_2})^{f_1} \not\equiv (D\alpha^{-f_2})^{e_1} \pmod{p}$$

is $1 - 1/q$.

PROOF Suppose that the following congruences are satisfied:

$$y \equiv x^a \pmod{p}$$
$$d \not\equiv x^{e_1}\alpha^{e_2} \pmod{p}$$
$$D \not\equiv x^{f_1}\alpha^{f_2} \pmod{p}$$
$$(d\alpha^{-e_2})^{f_1} \equiv (D\alpha^{-f_2})^{e_1} \pmod{p}.$$

We will derive a contradiction.

The consistency check (step 9) can be rewritten in the following form:

$$D \equiv {d_0}^{f_1}\alpha^{f_2} \pmod{p},$$

where

$$d_0 = d^{1/e_1}\alpha^{-e_2/e_1} \bmod p$$

is a value that depends only on steps 1–4 of the protocol.

Applying Theorem 6.1, we conclude that y is a valid signature for d_0 with probability $1 - 1/q$. But we are assuming that y is a valid signature for x. That is, with high probability we have

$$x^a \equiv {d_0}^a \pmod{p},$$

which implies that $x = d_0$.

However, the fact that

$$d \not\equiv x^{e_1}\alpha^{e_2} \pmod{p}$$

means that

$$x \not\equiv d^{1/e_1}\alpha^{-e_2/e_1} \pmod{p}.$$

Since

$$d_0 \equiv d^{1/e_1}\alpha^{-e_2/e_1} \pmod{p},$$

we conclude that $x \neq d_0$ and we have a contradiction.

Hence, Bob can fool Alice in this way with probability $1/q$. ∎

6.6 Fail-stop Signatures

A fail-stop signature scheme provides enhanced security against the possibility that a very powerful adversary might be able to forge a signature. In the event that Oscar is able to forge Bob's signature on a message, Bob will (with high probability) subsequently be able to prove that Oscar's signature is a forgery.

In this section, we describe a fail-stop signature scheme constructed by van Heyst and Pedersen in 1992. This is a one-time scheme (only one message can be signed with a given key). The system consists of signing and verification algorithms, as well as a "proof of forgery" algorithm. The description of the signing and verification algorithms of the **van Heyst and Pedersen Fail-stop Signature Scheme** is presented in Figure 6.9.

It is straightforward to see that a signature produced by Bob will satisfy the verification condition, so let's turn to the security aspects of this scheme and how the fail-safe property works. First we establish some important facts relating to the keys of the scheme. We begin with a definition. Two keys $(\gamma_1, \gamma_2, a_1, a_2, b_1, b_2)$ and $(\gamma_1', \gamma_2', a_1', a_2', b_1', b_2')$ are said to be *equivalent* if $\gamma_1 = \gamma_1'$ and $\gamma_2 = \gamma_2'$. It is easy to see that there are exactly q^2 keys in any equivalence class.

We establish several lemmas.

LEMMA 6.4
Suppose K and K' are equivalent keys and suppose that $ver_K(x, y) =$ true. *Then $ver_{K'}(x, y) =$* true.

PROOF Suppose $K = (\gamma_1, \gamma_2, a_1, a_2, b_1, b_2)$ and $K' = (\gamma_1, \gamma_2, a_1', a_2', b_1', b_2')$, where

$$\gamma_1 = \alpha^{a_1}\beta^{a_2} \bmod p = \alpha^{a_1'}\beta^{a_2'} \bmod p$$

and

$$\gamma_2 = \alpha^{b_1}\beta^{b_2} \bmod p = \alpha^{b_1'}\beta^{b_2'} \bmod p.$$

Suppose x is signed using K, producing the signature $y = (y_1, y_2)$, where

$$y_1 = a_1 + xb_1 \bmod q,$$

$$y_2 = a_2 + xb_2 \bmod q.$$

Now suppose that we verify y using K':

$$\alpha^{y_1}\beta^{y_2} \equiv \alpha^{a_1'+xb_1'}\beta^{a_2'+xb_2'} \pmod p$$

FIGURE 6.9
van Heyst and Pedersen Fail-stop Signature Scheme

Let $p = 2q + 1$ be a prime such that q is prime and the discrete log problem in $\mathbb{Z}_p$ is intractible, Let $\alpha \in \mathbb{Z}_p^*$ be an element of order q. Let $1 \leq a_0 \leq q - 1$ and define $\beta = \alpha^{a_0} \bmod p$. The values p, q, α, β, and a_0 are chosen by a central (trusted) authority. p, q, α, and β are public and will be regarded as fixed. The value of a_0 is kept secret from everyone (even Bob).

Let $\mathcal{P} = \mathbb{Z}_q$ and $\mathcal{A} = \mathbb{Z}_q \times \mathbb{Z}_q$. A key has the form

$$K = (\gamma_1, \gamma_2, a_1, a_2, b_1, b_2),$$

where $a_1, a_2, b_1, b_2 \in \mathbb{Z}_q$,

$$\gamma_1 = \alpha^{a_1}\beta^{a_2} \bmod p,$$

and

$$\gamma_2 = \alpha^{b_1}\beta^{b_2} \bmod p.$$

For $K = (\gamma_1, \gamma_2, a_1, a_2, b_1, b_2)$ and $x \in \mathbb{Z}_q$, define

$$sig_K(x) = (y_1, y_2),$$

where

$$y_1 = a_1 + xb_1 \bmod q$$

and

$$y_2 = a_2 + xb_2 \bmod q.$$

For $y = (y_1, y_2) \in \mathbb{Z}_q \times \mathbb{Z}_q$, we have

$$ver_K(x, y) = \text{true} \Leftrightarrow \gamma_1\gamma_2{}^x \equiv \alpha^{y_1}\beta^{y_2} \pmod p.$$

$$\equiv \alpha^{a_1'}\beta^{a_2'}(\alpha^{b_1'}\beta^{b_2'})^x \pmod{p}$$
$$\equiv \gamma_1\gamma_2^x \pmod{p}.$$

Thus, y will also be verified using K'. ∎

LEMMA 6.5
Suppose K is a key and $y = sig_K(x)$. Then there are exactly q keys K' equivalent to K such that $y = sig_{K'}(x)$.

PROOF Suppose γ_1 and γ_2 are the public components of K. We want to determine the number of 4-tuples (a_1, a_2, b_1, b_2) such that the following congruences are satisfied:

$$\gamma_1 \equiv \alpha^{a_1}\beta^{a_2} \pmod{p}$$
$$\gamma_2 \equiv \alpha^{b_1}\beta^{b_2} \pmod{p}$$
$$y_1 \equiv a_1 + xb_1 \pmod{q}$$
$$y_2 \equiv a_2 + xb_2 \pmod{q}.$$

Since α generates G, there exist unique exponents $c_1, c_2, a_0 \in \mathbb{Z}_q$ such that

$$\gamma_1 \equiv \alpha^{c_1} \pmod{p},$$

$$\gamma_2 \equiv \alpha^{c_2} \pmod{p}$$

and

$$\beta \equiv \alpha^{a_0} \pmod{p}.$$

Hence, it is necessary and sufficient that the following system of congruences be satisfied:

$$c_1 \equiv a_1 + a_0a_2 \pmod{q}$$
$$c_2 \equiv b_1 + a_0b_2 \pmod{q}$$
$$y_1 \equiv a_1 + xb_1 \pmod{q}$$
$$y_2 \equiv a_2 + xb_2 \pmod{q}.$$

This system can, in turn, be written as a matrix equation in $\mathbb{Z}_q$, as follows:

$$\begin{pmatrix} 1 & a_0 & 0 & 0 \\ 0 & 0 & 1 & a_0 \\ 1 & 0 & x & 0 \\ 0 & 1 & 0 & x \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ b_1 \\ b_2 \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ y_1 \\ y_2 \end{pmatrix}.$$

Now, the coefficient matrix of this system can be seen to have rank [1] three: Clearly,

[1] the *rank* of a matrix is the maximum number of linearly independent rows it contains

the rank is at least three since rows 1, 2 and 4 are linearly independent over $\mathbb{Z}_q$. And the rank is at most three since

$$r_1 + xr_2 - r_3 - a_0 r_4 = (0, 0, 0, 0),$$

where r_i denotes the ith row of the matrix.

Now, this system of equations has at least one solution, obtained by using the key K. Since the rank of the coefficient matrix is three, it follows that the dimension of the solution space is $4 - 3 = 1$, and there are exactly q solutions. The result follows. ▌

By similar reasoning, the following result can be proved. We omit the proof.

LEMMA 6.6

Suppose K is a key, $y = sig_K(x)$, and $ver_K(x', y') =$ true, where $x' \neq x$. Then there is at most one key K' equivalent to K such that $y = sig_{K'}(x)$ and $y' = sig_{K'}(x')$.

Let's interpret what the preceding two lemmas say about the security of the scheme. Given that y is a valid signature for message x, there are q possible keys that would have signed x with y. But for any message $x' \neq x$, these q keys will produce q different signatures on x'. Thus, the following theorem results.

THEOREM 6.7

Given that $sig_K(x) = y$ and $x' \neq x$, Oscar can compute $sig_K(x')$ with probablity $1/q$.

Note that this theorem does not depend on the computational power of Oscar: the stated level of security is obtained because Oscar cannot tell which of q possible keys is being used by Bob. So the security is unconditional.

We now go on to look at the fail-stop concept. What we have said so far is that, given a signature y on message x, Oscar cannot compute Bob's signature y' on a different message x'. It is still conceivable that Oscar can compute a forged signature $y'' \neq sig_K(x')$ which will still be verified. However, if Bob is given a valid forged signature, then with probability $1 - 1/q$ he can produce a "proof of forgery." The proof of forgery is the value $a_0 = \log_\alpha \beta$, which is known only to the central authority.

So we assume that Bob possesses a pair (x', y'') such that $ver(x', y'') =$ true and $y'' \neq sig_K(x')$. That is,

$$\gamma_1 \gamma_2^{x'} \equiv \alpha^{y_1''} \beta^{y_2''} \pmod{p},$$

where $y'' = (y_1'', y_2'')$. Now, Bob can compute his own signature on x', namely $y' = (y_1', y_2')$, and it will be the case that

$$\gamma_1 \gamma_2^{x'} \equiv \alpha^{y_1'} \beta^{y_2'} \pmod{p}.$$

Hence,

$$\alpha^{y_1''} \beta^{y_2''} \equiv \alpha^{y_1'} \beta^{y_2'} \pmod{p}.$$

Writing $\beta = \alpha^{a_0} \bmod p$, we have that

$$\alpha^{y_1'' + a_0 y_2''} \equiv \alpha^{y_1' + a_0 y_2'} \pmod{p},$$

or

$$y_1'' + a_0 y_2'' \equiv y_1' + a_0 y_2' \pmod{q}.$$

This simplifies to give

$$y_1'' - y_1' \equiv a_0 (y_2' - y_2'') \pmod{q}.$$

Now, $y_2' \not\equiv y_2'' \pmod{q}$ since y' is a forgery. Hence, $(y_2' - y_2'')^{-1} \bmod q$ exists, and

$$a_0 = \log_\alpha \beta = (y_1'' - y_1')(y_2' - y_2'')^{-1} \bmod q.$$

Of course, by accepting such a proof of forgery, we assume that Bob cannot compute the discrete logarithm $\log_\alpha \beta$ by himself. This is a computational assumption.

Finally, we remark that the scheme is a one-time scheme since Bob's key K can easily be computed if two messages are signed using K.

We close with an example illustrating how Bob can produce a proof of forgery.

Example 6.7
Suppose $p = 3467 = 2 \times 1733 + 1$. The element $\alpha = 4$ has order 1733 in $\mathbb{Z}_{3467}^*$. Suppose that $a_0 = 1567$, so

$$\beta = 4^{1567} \bmod 3467 = 514.$$

(Recall that Bob knows the values of α and β, but not a_0.) Suppose Bob forms his key using $a_1 = 888$, $a_2 = 1024$, $b_1 = 786$ and $b_2 = 999$, so

$$\gamma_1 = 4^{888} 514^{1024} \bmod 3467 = 3405$$

and

$$\gamma_2 = 4^{786} 514^{999} \bmod 3467 = 2281.$$

Now, suppose Bob is presented with the forged signature $(822, 55)$ on the message 3383. This is a valid signature since the verification condition is satisfied:

$$3405 \times 2281^{3383} \equiv 2282 \pmod{3467}$$

and

$$4^{822}514^{55} \equiv 2282 \pmod{3467}.$$

On the other hand, this is not the signature Bob would have constructed. Bob can compute his own signature to be

$$(888 + 3383 \times 786 \bmod 1733, 1024 + 3383 \times 999 \bmod 1733) = (1504, 1291).$$

Then, he proceeds to calculate the secret discrete log

$$a_0 = (822 - 1504)(1291 - 55)^{-1} \bmod 1733 = 1567.$$

This is the proof of forgery. □

6.7 Notes and References

For a nice survey of signature schemes, we recommend Mitchell, Piper, and Wild [MPW92]. This paper also contains the two methods of forging **ElGamal** signatures that we presented in Section 6.2.

The **ElGamal Signature Scheme** was presented in ElGamal [EL85]. The **Digital Signature Standard** was first published by NIST in August 1991, and it was adopted as a standard in December 1994 [NBS94]. There is a lengthy discussion of **DSS** and the controversy surrounding it in the July 1992 issue of the *Communications of the ACM*. For a response by NIST to some of the questions raised, see [SB93].

The **Lamport Scheme** is described in the 1976 paper by Diffie and Hellman [DH76]; the modification by Bos and Chaum is in [BC93]. The undeniable signature scheme presented in Section 6.5 is due to Chaum and van Antwerpen [CVA90]. The fail-stop signature scheme from Section 6.6 is due to van Heyst and Pedersen [VHP93].

Some examples well-known "broken" signature schemes include the **Ong-Schnorr-Shamir Scheme** [OSS85] (broken by Estes *et al.* [EAKMM86]); and the **Birational Permutation Scheme** of Shamir [SH94] (broken by Coppersmith, Stern, and Vaudenay [CSV94]). Finally, **ESIGN** is a signature scheme due to Fujioka, Okamoto, and Miyaguchi [FOM91]. Some versions of the scheme were broken, but the variation in [FOM91] has not been broken.

Exercises

6.1 Suppose Bob is using the **ElGamal Signature Scheme**, and he signs two messages x_1 and x_2 with signatures (γ, δ_1) and (γ, δ_2), respectively. (The same value for γ occurs in both signatures.) Suppose also that $\gcd(\delta_1 - \delta_2, p-1) = 1$.

(a) Describe how k can be computed efficiently given this information.
(b) Describe how the signature scheme can then be broken.
(c) Suppose $p = 31847$, $\alpha = 5$ and $\beta = 25703$. Perform the computation of k and a, given the signature $(23972, 31396)$ for the message $x = 8990$ and the signature $(23972, 20481)$ for the message $x = 31415$.

6.2 Suppose I implement the **ElGamal Signature Scheme** with $p = 31847$, $\alpha = 5$ and $\beta = 26379$. Write a computer program which does the following.

(a) Verify the signature $(20679, 11082)$ on the message $x = 20543$.
(b) Determine my secret exponent, a, using the Shanks time-memory tradeoff. Then determine the random value k used in signing the message x.

6.3 Suppose Bob is using the **ElGamal Signature Scheme** as implemented in Example 6.1: $p = 467$, $\alpha = 2$ and $\beta = 132$. Suppose Bob has signed the message $x = 100$ with the signature $(29, 51)$. Compute the forged signature that Oscar can then form by using $h = 102$, $i = 45$ and $j = 293$. Check that the resulting signature satisfies the verification condition.

6.4 Prove that the second method of forgery on the **ElGamal Signature Scheme**, described in Section 6.2, also yields a signature that satisfies the verification condition.

6.5 Here is a variation of the **ElGamal Signature Scheme**. The key is constructed in a similar manner as before: Bob chooses $\alpha \in \mathbb{Z}_p^*$ to be a primitive element, a is a secret exponent ($0 \leq a \leq p-2$) such that $\gcd(a, p-1) = 1$, and $\beta = \alpha^a \bmod p$. The key $K = (\alpha, a, \beta)$, where α and β are public and a is secret. Let $x \in \mathbb{Z}_p$ be a message to be signed. Bob computes the signature $sig(x) = (\gamma, \delta)$, where

$$\gamma = \alpha^k \bmod p$$

and

$$\delta = (x - k\gamma)a^{-1} \bmod (p-1).$$

The only difference from the original **ElGamal Scheme** is in the computation of δ. Answer the following questions concerning this modified scheme.

(a) Describe how a signature (γ, δ) on a message x would be verified using Bob's public key.
(b) Describe a computational advantage of the modified scheme over the original scheme.
(c) Briefly compare the security of the original and modified scheme.

6.6 Suppose Bob uses the **DSS** with $q = 101$, $p = 7879$, $\alpha = 170$, $a = 75$ and $\beta = 4567$, as in Example 6.3. Determine Bob's signature on the message $x = 5001$ using the random value $k = 49$, and show how the resulting signature is verified.

6.7 In the **Lamport Scheme**, suppose that two $k-$tuples, x and x', are signed by Bob. Let $\ell = d(x, x')$ denote the number of coordinates in which x and x' differ. Show that Oscar can now sign $2^\ell - 2$ new messages.

6.8 In the **Bos-Chaum Scheme** with $k = 6$ and $n = 4$, suppose that the messages $x = (0, 1, 0, 0, 1, 1)$ and $x' = (1, 1, 0, 1, 1, 1)$ are signed. Determine the new messages that be signed by Oscar, knowing the signatures on x and x'.

6.9 In the **Bos-Chaum Scheme**, suppose that two $k-$tuples x and x' are signed by Bob. Let $\ell = |\phi(x) \cup \phi(x')|$. Show that Oscar can now sign $\binom{\ell}{n} - 2$ new messages.

6.10 Suppose Bob is using the **Chaum-van Antwerpen Undeniable Signature Scheme** as in Example 6.5. That is, $p = 467, \alpha = 4, a = 101$ and $\beta = 449$. Suppose Bob is presented with a signature $y = 25$ on the message $x = 157$ and he wishes to prove it is a forgery. Suppose Alice's random numbers are $e_1 = 46, e_2 = 123, f_1 = 198$ and $f_2 = 11$ in the disavowal protocol. Compute Alice's challenges, c and d, and Bob's responses, C and D, and show that Alice's consistency check will succeed.

6.11 Prove that each equivalence class of keys in the **Pedersen-van Heyst Fail-stop Signature Scheme** contains q^2 keys.

6.12 Suppose Bob is using the **Pedersen-van Heyst Fail-stop Signature Scheme**, where $p = 3467, \alpha = 4, a_0 = 1567$ and $\beta = 514$ (of course, the value of a_0 is not known to Bob).

(a) Using the fact that $a_0 = 1567$, determine all possible keys

$$K = (\gamma_1, \gamma_2, a_1, a_2, b_1, b_2)$$

such that $sig_K(42) = (1118, 1449)$.

(b) Suppose that $sig_K(42) = (1118, 1449)$ and $sig_K(969) = (899, 471)$. Without using the fact that $a_0 = 1567$, determine the value of K (this shows that the scheme is a one-time scheme).

6.13 Suppose Bob is using the **Pedersen-van Heyst Fail-stop Signature Scheme** with $p = 5087, \alpha = 25$ and $\beta = 1866$. Suppose the key is

$$K = (5065, 5076, 144, 874, 1873, 2345).$$

Now, suppose Bob finds the signature $(2219, 458)$ has been forged on the message 4785.

(a) Prove that this forgery satisfies the verification condition, so it is a valid signature.

(b) Show how Bob will compute the proof of forgery, a_0, given this forged signature.