Suppose Alice wants to sign a message x, which is a bitstring of arbitrary length. She first constructs the message digest $z = h(x)$, and then computes the signature on z, namely, $y = sig_K(z)$. Then she transmits the ordered pair (x, y) over the channel. Now the verification can be performed (by anyone) by first reconstructing the message digest $z = h(x)$ using the public hash function h, and then checking that $ver_K(z, y) = true$.

We have to be careful that the use of a hash function h does not weaken the security of the signature scheme, for it is the message digest that is signed, not the message. It will be necessary for h to satisfy certain properties in order to prevent various attacks. The desired properties of hash functions were the ones that were already discussed in Section 5.2.

The most obvious type of attack is for Oscar to start with a valid signed message (x, y), where $y = sig_K(h(x))$. (The pair (x, y) could be any message previously signed by Alice.) Then he computes $z = h(x)$ and attempts to find $x' \neq x$ such that $h(x') = h(x)$. If Oscar can do this, (x', y) would be a valid signed message, so y is a forged signature for the message x'. This is an existential forgery using a known message attack. In order to prevent this type of attack, we require that h be second-preimage resistant.

Another possible attack is the following: Oscar first finds two messages $x \neq x'$ such that $h(x) = h(x')$. Oscar then gives x to Alice and persuades her to sign the message digest $h(x)$, obtaining y. Then (x', y) is a valid signed message and y is a forged signature for the message x'. This is an existential forgery using a chosen message attack; it can be prevented if h is collision resistant.

Here is a third variety of attack. It is often possible with certain signature schemes to forge signatures on random message digests z (we observed already that this could be done with the *RSA Signature Scheme*). That is, we assume that the signature scheme (without the hash function) is subject to existential forgery using a key-only attack. Now, suppose Oscar computes a signature on some message digest z, and then he finds a message x such that $z = h(x)$. If he can do this, then (x, y) is a valid signed message and y is a forged signature for the message x. This is an existential forgery on the signature scheme using a key-only attack. In order to prevent this attack, we desire that h be a preimage resistant hash function.

8.3 The ElGamal Signature Scheme

In this section, we present the *ElGamal Signature Scheme*, which was described in a 1985 paper. A modification of this scheme has been adopted as the *Digital Signature Algorithm* (or *DSA*) by the National Institute of Standards and Technology. The *DSA* also incorporates some ideas used in a scheme known as the *Schnorr Signature Scheme*. All of these schemes are designed specifically for the purpose of signatures, as opposed to the *RSA Cryptosystem*, which can be used both as a public-key cryptosystem and a signature scheme.

Cryptosystem 8.2: *ElGamal Signature Scheme*

Let p be a prime such that the discrete log problem in $\mathbb{Z}_p$ is intractable, 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 the public key, and a is the private key.

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

$$\mathbf{sig}_K(x, k) = (\gamma, \delta),$$

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

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

The *ElGamal Signature Scheme* is randomized (recall that the *ElGamal Public-key Cryptosystem* is also randomized). This means that there are many valid signatures for any given message, and the verification algorithm must be able to accept any of these valid signatures as authentic. The description of the *ElGamal Signature Scheme* is given as Cryptosystem 8.2.

We begin with a couple of preliminary observations. An ElGamal signature consists of two components, which are denoted γ and δ. The first component, γ, is obtained by raising α to a random power modulo p; it does not depend on the message (namely, x) that is being signed. The second component, δ, depends on the message x as well as the private key a. Verifying the signature is accomplished by checking that a certain congruence holds modulo p; this congruence does not involve the private key, of course.

We now show that, if the signature was constructed correctly, then the verification will succeed. This follows easily from the following congruences:

$$\begin{aligned} \beta^\gamma \gamma^\delta &\equiv \alpha^{a\gamma} \alpha^{k\delta} \pmod p \\ &\equiv \alpha^x \pmod p, \end{aligned}$$

where we use the fact that

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

Actually, it is probably less mysterious to begin with the verification equation, and then derive the corresponding signing function. Suppose we start with the congruence

$$\alpha^x \equiv \beta^\gamma \gamma^\delta \pmod{p}. \tag{8.1}$$

Then we make the substitutions

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

and

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

but we do not substitute for γ in the exponent of (8.1). We obtain the following:

$$\alpha^x \equiv \alpha^{a\gamma + k\delta} \pmod{p}.$$

Now, α is a primitive element modulo p; so this congruence is true if and only if the exponents are congruent modulo $p-1$, i.e., if and only if

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

Given x, a, γ, and k, this congruence can be solved for δ, yielding the formula used in the signing function of Cryptosystem 8.2.

Alice computes a signature using both the private key, a, 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 8.1 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 Alice wants to sign the message $x = 100$ and she 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 the signature $(29, 51)$ by checking that

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

and

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

Hence, the signature is valid. □

8.3.1 Security of the ElGamal Signature Scheme

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 given message x using this approach.

However, there is a method by which Oscar can sign a random message by choosing γ, δ, and x simultaneously. Thus an existential forgery is possible under a key-only attack (assuming a hash function is not used). We describe how to do this now.

Suppose i and j are integers such that $0 \leq i \leq p-2$, $0 \leq j \leq p-2$, and suppose we express γ in the form $\gamma = \alpha^i \beta^j \bmod p$. Then the verification condition is

$$\alpha^x \equiv \beta^\gamma (\alpha^i \beta^j)^\delta \pmod{p}.$$

This is equivalent to

$$\alpha^{x - i\delta} \equiv \beta^{\gamma + j\delta} \pmod{p}.$$

This latter congruence will be satisfied if

$$x - i\delta \equiv 0 \pmod{p-1}$$

and

$$\gamma + j\delta \equiv 0 \pmod{p-1}.$$

Given i and j, we can easily solve these two congruences modulo $p-1$ for δ and x, provided that $\gcd(j, p-1) = 1$. We obtain the following:

$$\begin{aligned} \gamma &= \alpha^i \beta^j \bmod p, \\ \delta &= -\gamma j^{-1} \bmod (p-1), \quad \text{and} \\ x &= -\gamma i j^{-1} \bmod (p-1). \end{aligned}$$

By the way in which we constructed (γ, δ), it is clear that it is a valid signature for the message x.

We illustrate with an example.

Example 8.2 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:

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

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

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

□

Here is a second type of forgery, in which Oscar begins with a message previously signed by Alice. This is an existential forgery under a known message attack. 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:

$$\begin{array}{rcl} \lambda & = & \gamma^h \alpha^i \beta^j \bmod p \\ \mu & = & \delta\lambda(h\gamma - j\delta)^{-1} \bmod (p-1), \quad \text{and} \\ x' & = & \lambda(hx + i\delta)(h\gamma - j\delta)^{-1} \bmod (p-1). \end{array}$$

Then, it is tedious but straightforward to check that the verification condition

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

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

Both of these methods are existential forgeries, but it does not appear that they can be modified to yield selective forgeries. Hence, they do not seem to represent a threat to the security of the *ElGamal Signature Scheme*, provided that a secure hash function is used as described in Section 8.2.1.

We also mention a couple of ways in which the *ElGamal Signature Scheme* can be broken if it is used carelessly (these are further instances of protocol failures, as introduced in the Exercises of Chapter 6). First, the random value k used in computing a signature should not be revealed. For, if k is known and $gcd(\gamma, p-1) = 1$, then it is a simple matter to compute

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

Once a is known, then the system is completely 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 will result in a repeated γ value, and it 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

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

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

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

which is equivalent to

$$x_1 - x_2 \equiv k(\delta_1 - \delta_2) \pmod{p-1}. \tag{8.2}$$

Let $d = \gcd(\delta_1 - \delta_2, p - 1)$. If $d = 1$, then we can immediately solve (8.2), obtaining

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

However, even if $d > 1$, we still might be able to determine k, provided d is not too large. Since $d \mid (p-1)$ and $d \mid (\delta_1 - \delta_2)$, it follows that $d \mid (x_1 - x_2)$. Define

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

Then the congruence (8.2) becomes:

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

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

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

The value of k is determined modulo p' to be

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

This yields d candidate values for k:

$$k = x'\epsilon + ip' \bmod (p-1)$$

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

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