Chapter 13 Digital Signatures

For years, people have been using various types of signatures to associate their identities to documents. In the Middle Ages, a nobleman sealed a document with a wax imprint of his insignia. The assumption was that the noble was the only person able to reproduce the insignia. In modern transactions, credit card slips are signed. The salesperson is supposed to verify the signature by comparing with the signature on the card. With the development of electronic commerce and electronic documents, these methods no longer suffice.

For example, suppose you want to sign an electronic document. Why can't you simply digitize your signature and append it to the document? Anyone who has access to it can simply remove the signature and add it to something else, for example, a check for a large amount of money. With classical signatures, this would require cutting the signature off the document, or photocopying it, and pasting it on the check. This would rarely pass for an acceptable signature. However, such an electronic forgery is quite easy and cannot be distinguished from the original.

Therefore, we require that digital signatures cannot be separated from the message and attached to another. That is, the signature is not only tied to the signer but also to the message that is being signed. Also, the digital signature needs to be easily verified by other parties. Digital signature schemes therefore consist of two distinct steps: the signing process, and the verification process.

In the following, we first present two signature schemes. We also discuss the important "birthday attacks" on

signature schemes.

Note that we are not trying to encrypt the message m. In fact, often the message is a legal document, and therefore should be kept public. However, if necessary, a signed message may be encrypted after it is signed. (This is done in PGP, for example. See Section 15.6.)

13.1 RSA Signatures

Bob has a document m that Alice agrees to sign. They do the following:

1. Alice generates two large primes p, q, and computes $n = pq$. She chooses e_A such that $1 < e_A < \phi(n)$ with $\gcd(e_A,\ \phi(n)) = 1,$ and calculates d_A such that $e_A d_A \equiv 1 \pmod{\phi(n)}$. Alice publishes $(e_A,\ n)$ and keeps private d_A, p, q.
2. Alice's signature is

   $$s \equiv m^{d_A} \pmod{n}.$$

3. The pair $(m,\ s)$ is then made public.

Bob can then verify that Alice really signed the message by doing the following:

1. Download Alice's $(e_A,\ n)$.
2. Calculate $z \equiv s^{e_A} \pmod{n}$. If $z = m$, then Bob accepts the signature as valid; otherwise the signature is not valid.

Suppose Eve wants to attach Alice's signature to another message m_1. She cannot simply use the pair $(m_1,\ s)$, since $s^{e_A} \not\equiv m_1 \pmod{n}$. Therefore, she needs s_1 with $s_1^{e_A} \equiv m_1 \pmod{n}$. This is the same problem as decrypting an RSA "ciphertext" m_1 to obtain the "plaintext" s_1. This is believed to be hard to do.

Another possibility is that Eve chooses s_1 first, then lets the message be $m_1 \equiv s_1^{e_A} \pmod{n}$. It does not appear that Alice can deny having signed the message m_1 under the present scheme. However, it is very unlikely that m_1 will be a meaningful message. It will probably be a random sequence of characters, and not a message committing her to give Eve millions of dollars. Therefore, Alice's claim that it has been forged will be believable.

There is a variation on this procedure that allows Alice to sign a document without knowing its contents. Suppose Bob has made an important discovery. He wants to record publicly what he has done (so he will have priority when it comes time to award Nobel prizes), but he does not want anyone else to know the details (so he can make a lot of money from his invention). Bob and Alice do the following. The message to be signed is m.

1. Alice chooses an RSA modulus n ($n = pq$, the product of two large primes), an encryption exponent e, and decryption exponent d. She makes n and e public while keeping p, q, d private. In fact, she can erase p, q, d from her computer's memory at the end of the signing procedure.

2. Bob chooses a random integer $k \pmod n$ with $\gcd(k, n) = 1$ and computes $t \equiv k^e m \pmod n$. He sends t to Alice.

3. Alice signs t by computing $s \equiv t^d \pmod n$. She returns s to Bob.

4. Bob computes $s/k \pmod n$. This is the signed message m^d.

Let's show that s/k is the signed message: Note that $k^{ed} \equiv (k^e)^d \equiv k \pmod n$, since this is simply the encryption, then decryption, of k in the RSA scheme. Therefore,

$$s/k \equiv t^d/k \equiv k^{ed} m^d/k \equiv m^d \pmod n,$$

which is the signed message.

The choice of k is random, so $k^e \pmod n$ is the RSA encryption of a random number, and hence random. Therefore, $k^e m \pmod n$ gives essentially no information about m (however, it would not hide a message such as $m = 0$). In this way, Alice knows nothing about the message she is signing.

Once the signing procedure is finished, Bob has the same signed message as he would have obtained via the standard signing procedure.

There are several potential dangers with this protocol. For example, Bob could have Alice sign a promise to pay him a million dollars. Safeguards are needed to prevent such problems. We will not discuss these here.

Schemes such as these, called **blind signatures**, have been developed by David Chaum, who has several patents on them.

13.2 The ElGamal Signature Scheme

The ElGamal encryption method from Section 10.5 can be modified to give a signature scheme. One feature that is different from RSA is that, with the ElGamal method, there are many different signatures that are valid for a given message.

Suppose Alice wants to sign a message. To start, she chooses a large prime p and a primitive root α. Alice next chooses a secret integer a such that $1 \leq a \leq p-2$ and calculates $\beta \equiv \alpha^a \pmod{p}$. The values of p, α, and β are made public. The security of the system will be in the fact that a is kept private. It is difficult for an adversary to determine a from (p, α, β) since the discrete log problem is considered difficult.

In order for Alice to sign a message m, she does the following:

1. Selects a secret random k with $1 \leq k \leq p-2$ such that $\gcd(k, p-1) = 1$.
2. Computes $r \equiv \alpha^k \pmod{p}$ (with $0 < r < p$).
3. Computes $s \equiv k^{-1}(m - ar) \pmod{p-1}$.

The signed message is the triple (m, r, s).

Bob can verify the signature as follows:

1. Download Alice's public key (p, α, β).
2. Compute $v_1 \equiv \beta^r r^s \pmod{p}$, and $v_2 \equiv \alpha^m \pmod{p}$.
3. The signature is declared valid if and only if $v_1 \equiv v_2 \pmod{p}$.

We now show that the verification procedure works. Assume the signature is valid. Since $s \equiv k^{-1}(m - ar) \pmod{p-1}$, we have $sk \equiv m - ar \pmod{p-1}$, so $m \equiv sk + ar \pmod{p-1}$. Therefore (recall that a congruence mod $p-1$ in the exponent yields an overall congruence mod p),

$$v_2 \equiv \alpha^m \equiv \alpha^{sk+ar} \equiv (\alpha^a)^r(\alpha^k)^s \equiv \beta^r r^s \equiv v_1 \pmod{p}.$$

Suppose Eve discovers the value of a. Then she can perform the signing procedure and produce Alice's signature on any desired document. Therefore, it is very important that a remain secret.

If Eve has another message m, she cannot compute the corresponding s since she doesn't know a. Suppose she tries to bypass this step by choosing an s that satisfies the verification equation. This means she needs s to satisfy

$$\beta^r r^s \equiv \alpha^m \pmod{p}.$$

This can be rearranged to $r^s \equiv \beta^{-r}\alpha^m \pmod{p}$, which is a discrete logarithm problem. Therefore, it should be hard to find an appropriate s. If s is chosen first, the equation for r is similar to a discrete log problem, but more complicated. It is generally assumed that it is also difficult to solve. It is not known whether there is a way to choose r and s simultaneously, though this seems to be unlikely. Therefore, the signature scheme appears to be secure, as long as discrete logs mod p are difficult to compute (for example, $p-1$ should not be a product of small primes; see Section 10.2).

Suppose Alice wants to sign a second document. She must choose a new random value of k. Suppose instead that she uses the same k for messages m_1 and m_2. Then the same value of r is used in both signatures, so Eve will see that k has been used twice. The s values are different; call them s_1 and s_2. Eve knows that

$$s_1 k - m_1 \equiv -ar \equiv s_2 k - m_2 \pmod{p-1}.$$

Therefore,

$$(s_1 - s_2)k \equiv m_1 - m_2 \pmod{p-1}.$$

Let $d = \gcd(s_1 - s_2,\ p-1)$. There are d solutions to the congruence, and they can be found by the procedure given in Subsection 3.3.1. Usually d is small, so there are not very many possible values of k. Eve computes α^k for each possible k until she gets the value r. She now knows k. Eve now solves

$$ar \equiv m_1 - ks_1 \pmod{p-1}$$

for a. There are $\gcd(r,\ p-1)$ possibilities. Eve computes α^a for each one until she obtains β, at which point she has found a. She now has completely broken the system and can reproduce Alice's signatures at will.

Example

Alice wants to sign the message $m_1 = 151405$ (which corresponds to *one*, if we let $01 = a$, $02 = b$, ...). She chooses $p = 225119$. Then $\alpha = 11$ is a primitive root. She has a secret number a. She computes $\beta \equiv \alpha^a \equiv 18191 \pmod{p}$. To sign the message, she chooses a random number k and keeps it secret. She computes $r \equiv \alpha^k \equiv 164130 \pmod{p}$. Then she computes

$$s_1 \equiv k^{-1}(m_1 - ar) \equiv 130777 \pmod{p-1}.$$

The signed message is the triple $(151405, 164130, 130777)$.

Now suppose Alice also signs the message $m_2 = 202315$ (which is *two*) and produces the signed message $(202315, 164130, 164899)$. Immediately, Eve recognizes that Alice used the same value of k, since

the value of r is the same in both signatures. She therefore writes the congruence

$$-34122k \equiv (s_1 - s_2)k \equiv m_1 - m_2 \equiv -50910 \pmod{p-1}.$$

Since $\gcd(-34122, p-1) = 2$, there are two solutions, which can be found by the method described in Subsection 3.3.1. Divide the congruence by 2:

$$-17061k \equiv -25455 \pmod{(p-1)/2}.$$

This has the solution $k \equiv 239 \pmod{(p-1)/2}$, so there are two values of $k \pmod{p}$, namely 239 and $239 + (p-1)/2 = 112798$. Calculate

$$\alpha^{239} \equiv 164130, \ \alpha^{112798} \equiv 59924 \pmod{p}.$$

Since the first is the correct value of r, Eve concludes that $k = 239$. She now rewrites $s_1 k \equiv m_1 - ar \pmod{p-1}$ to obtain

$$164130a \equiv ra \equiv m_1 - s_1 k \equiv 187104 \pmod{p-1}.$$

Since $\gcd(164130, p-1) = 2$, there are two solutions, namely $a = 28862$ and $a = 141421$, which can be found by the method of Subsection 3.3.1. Eve computes

$$\alpha^{28862} \equiv 206928, \ \alpha^{141421} \equiv 18191 \pmod{p}.$$

Since the second value is β, she has found that $a = 141421$.

Now that Eve knows a, she can forge Alice's signature on any document.

The ElGamal signature scheme is an example of a **signature with appendix**. The message is not easily recovered from the signature (r, s). The message m must be included in the verification procedure. This is in contrast to the RSA signature scheme, which is a **message recovery scheme**. In this case, the message is readily obtained from the signature s. Therefore, only

s needs to be sent since anyone can deduce m as $s^{e_A} \pmod{n}$. It is unlikely that a random s will yield a meaningful message m, so there is little danger that someone can successfully replace a valid message with a forged message by changing s.

13.3 Hashing and Signing

In the two signature schemes just discussed, the signature can be longer than the message. This is a disadvantage when the message is long. To remedy the situation, a hash function is used. The signature scheme is then applied to the hash of the message, rather than to the message itself.

The hash function h is made public. Starting with a message m, Alice calculates the hash $h(m)$. This output $h(m)$ is significantly smaller, and hence signing the hash may be done more quickly than signing the entire message. Alice calculates the signed message $\text{sig}(h(m))$ for the hash function and uses it as the signature of the message. The pair $(m, sig(h(m)))$ now conveys basically the same knowledge as the original signature scheme did. It has the advantages that it is faster to create (under the reasonable assumption that the hash operation is quick) and requires less resources for transmission or storage.

Is this method secure? Suppose Eve has possession of Alice's signed message $(m, \text{sig}(h(m))$. She has another message m' to which she wants to add Alice's signature. This means that she needs $\text{sig}(h(m'))$ = $\text{sig}(h(m))$; in particular, she needs $h(m') = h(m)$. If the hash function is one-way, Eve will find it hard to find any such m'. The chance that her desired m' will work is very small. Moreover, since we require our hash function to be strongly collision-resistant, it is unlikely that Eve can find two messages $m_1 \neq m_2$ with the same signatures. Of course, if she did, she could have Alice sign m_1, then transfer her signature to m_2. But Alice would get suspicious since m_1 (and m_2) would very likely be meaningless messages.

In the next section, however, we'll see how Eve can trick Alice if the size of the message digest is too small (and we'll see that the hash function will not be strongly collision-resistant, either).

13.4 Birthday Attacks on Signatures

Alice is going to sign a document electronically by using one of the signature schemes to sign the hash of the document. Suppose the hash function produces an output of 50 bits. She is worried that Fred will try to trick her into signing an additional contract, perhaps for swamp land in Florida, but she feels safe because the chance of a fraudulent contract having the same hash as the correct document is 1 out of 2^{50}, which is approximately 1 out of 10^{15}. Fred can try several fraudulent contracts, but it is very unlikely that he can find one that has the right hash. Fred, however, has studied the birthday attack and does the following. He finds 30 places where he can make a slight change in the document: adding a space at the end of a line, changing a wording slightly, etc. At each place, he has two choices: Make the small change or leave the original. Therefore, he can produce 2^{30} documents that are essentially identical with the original. Surely, Alice will not object to any of these versions. Now, Fred computes the hash of each of the 2^{30} versions and stores them. Similarly, he makes 2^{30} versions of the fraudulent contract and stores their hashes. Consider the generalized birthday problem with $r = 2^{30}$ and $N = 2^{50}$. The probability of a match is approximately

$$1 - e^{-r^2/N} = 1 - e^{-1024} \approx 1.$$

Therefore, it is very likely that a version of the good document has the same hash as a version of the fraudulent contract. Fred finds the match and asks Alice to sign the good version. He plans to append her signature to the fraudulent contract, too. Since they have the same hash, the signature would be valid for the

fraudulent one, so Fred could claim that Alice agreed to buy the swamp land.

But Alice is an English teacher and insists on removing a comma from one sentence. Then she signs the document, which has a completely different hash from the document Fred asked her to sign. Fred is foiled again. He now is faced with the prospect of trying to find a fraudulent contract that has the same hash as this new version of the good document. This is essentially impossible.

What Fred did is called the birthday attack. Its practical implication is that you should probably use a hash function with output twice as long as what you believe to be necessary, since the birthday attack effectively halves the number of bits. What Alice did is a recommended way to foil the birthday attack on signature schemes. Before signing an electronic document, make a slight change.

13.5 The Digital Signature Algorithm

The National Institute of Standards and Technology proposed the Digital Signature Algorithm (DSA) in 1991 and adopted it as a standard in 1994. Later versions increased the sizes of the parameters. Just like the ElGamal method, DSA is a digital signature scheme with appendix. Also, like other schemes, it is usually a message digest that is signed. In this case, let's say the hash function produces a 256-bit output. We will assume in the following that our data message m has already been hashed. Therefore, we are trying to sign a 256-bit message.

The generation of keys for DSA proceeds as follows. First, there is an initialization phase:

1. Alice finds a prime q that is 256 bits long and chooses a prime p that satisfies $q|p-1$ (see Exercise 15). The discrete log problem should be hard for this choice of p. (In the initial version, p had 512 bits. Later versions of the standard require longer primes, for example, 2048 bits.)

2. Let g be a primitive root mod p and let $\alpha \equiv g^{(p-1)/q} \pmod{p}$. Then $\alpha^q \equiv 1 \pmod{p}$.

3. Alice chooses a secret a such that $1 \leq a < q-1$ and calculates $\beta \equiv \alpha^a \pmod{p}$.

4. Alice publishes (p, q, α, β) and keeps a secret.

Alice signs a message m by the following procedure:

1. Select a random, secret integer k such that $0 < k < q-1$.

2. Compute $r = (\alpha^k \pmod{p}) \pmod{q}$.

3. Compute $s \equiv k^{-1}(m + ar) \pmod{q}$.

4. Alice's signature for m is (r, s), which she sends to Bob along with m.

For Bob to verify, he must

1. Download Alice's public information (p, q, α, β).
2. Compute $u_1 \equiv s^{-1}m \pmod q$, and $u_2 \equiv s^{-1}r \pmod q$.
3. Compute $v = (\alpha^{u_1}\beta^{u_2} \pmod p) \pmod q$.
4. Accept the signature if and only if $v = r$.

We show that the verification works. By the definition of s, we have

$$m \equiv (-ar + ks) \pmod q,$$

which implies

$$s^{-1}m \equiv (-ars^{-1} + k) \pmod q.$$

Therefore,

$$\begin{aligned} k &\equiv s^{-1}m + ars^{-1} \pmod q \\ &\equiv u_1 + au_2 \pmod q. \end{aligned}$$

So $\alpha^k = \alpha^{u_1+au_2} = (\alpha^{u_1}\beta^{u_2} \pmod p) \pmod q$. Thus $v = r$.

As in the ElGamal scheme, the integer a must be kept secret. Anyone who has knowledge of a can sign any desired document. Also, if the same value of k is used twice, it is possible to find a by the same procedure as before.

In contrast to the ElGamal scheme, the integer r does not carry full information about k. Knowing r allows us to find only the mod q value. There are approximately $2^{2048-256} = 2^{1792}$ numbers mod p that reduce to a given number mod q.

What is the advantage of having $\alpha^q \equiv 1 \pmod p$ rather than using a primitive root? Recall the Pohlig-Hellman attack for solving the discrete log problem. It could find information mod small prime factors of $p - 1$, but it was useless mod large prime factors such as q. In

the ElGamal scheme, an attacker could determine $a \pmod{2^t}$, where 2^t is the largest power of 2 dividing $p-1$. This would not come close to finding a, but the general philosophy is that many little pieces of information collectively can often be useful. The DSA avoids this problem by removing all but the mod q information for a.

In the ElGamal scheme, three modular exponentiations are needed in the verification step. This step is modified for the DSA so that only two modular exponentiations are needed. Since modular exponentiation is one of the slower parts of the computation, this change speeds up the verification, which can be important if many signatures need to be verified in a short time.

13.6 Exercises

1. Show that if someone discovers the value of k used in the ElGamal signature scheme, then a can also be determined if $\gcd(r, p-1)$ is small.

2. Alice signs the hash of a message. Suppose her hash function satisfies $h(x) \equiv 2^x \pmod{101}$ and $1 \leq h(x) \leq 100$ for all x. Suppose m is a valid signed message from Alice. Give another message m_1 for which the same signature is also valid.

3. Alice says that she is willing to sign a petition to save koala bears. Alice's signing algorithm uses a hash function that has an output of 60 bits (and she signs the hash of the document). Describe how Eve can trick Alice into signing a statement allowing Eve unlimited withdrawals from Alice's bank account.

4. Alice uses RSA signatures (without a hash function).

    1. Eve wants to produce Alice's signature on the document $m = 123456789$. Why is this difficult? Explain this by saying what difficult cryptographic problem must be solved. (Do not say that it's because Eve does not know the decryption exponent d. Why isn't there another way to produce the signature?)

    2. Since part (a) is too hard for Eve, she decides to produce Alice's valid RSA signature on a document so that the signature is $s = 112090305$ (= Alice, when $A = 01$, $L = 12$, etc.). How does Eve find an appropriate message?

5. Nelson thinks he has a new version of the signature scheme. He chooses RSA parameters n, e, and d. He signs by computing $s \equiv m^d \pmod{n}$. The verification equation is $(s-m)(s+m) \stackrel{?}{=} s^2 - m^2$.

    1. Show that if Nelson correctly follows the signing procedure, or if he doesn't, then the signature is declared valid.

    2. Show that Eve can forge Nelson's signature on any document m, even though she does not know d.

    (The point of this exercise is that the verification equation is important. All Eve needs to do is satisfy the verification equation.

She does not need to follow the prescribed procedure for producing the signature.)

6. Alice has a long message m. She breaks m into blocks of 256 bits: $m = m_1||m_2||\cdots||m_N$. She regards each block as a number between 0 and $2^{256}-1$, and she signs the sum $t = m_1 + m_2 + \cdots + m_N$. This means her signed message is $(m,\ \text{sig}(t))$, where *sig* is the signing function. Is this a good idea? Why or why not?

7. Suppose that (m, r, s) is a message signed with the ElGamal signature scheme. Choose h with $\gcd(h, p-1) = 1$ and let $r_1 \equiv r^h \pmod{p}$. Let $s_1 \equiv s r_1 h^{-1} r^{-1} \pmod{p-1}$.

    1. Find a message m_1 for which (m_1, r_1, s_1) is a valid signature.

    2. This method allows Eve to forge a signature on the message m_1. Why is it unlikely that this causes problems?

8. Let $p = 11$, $q = 5$, $\alpha = 3$, and $k = 3$. Show that $(\alpha^k \pmod{p}) \pmod{q} \neq (\alpha^k \pmod{q}) \pmod{p}$. This shows that the order of operations in the DSA is important.

9. There are many variations to the ElGamal digital signature scheme that can be obtained by altering the signing equation $s \equiv k^{-1}(m - ar) \pmod{p-1}$. Here are some variations.

    1. Consider the signing equation $s \equiv a^{-1}(m - kr) \pmod{p-1}$. Show that the verification $\alpha^m \equiv (\alpha^a)^s r^r \pmod{p}$ is a valid verification procedure.

    2. Consider the signing equation $s \equiv am + kr \pmod{p-1}$. Show that the verification $\alpha^s \equiv (\alpha^a)^m r^r \pmod{p}$ is a valid verification procedure.

    3. Consider the signing equation $s \equiv ar + km \pmod{p-1}$. Show that the verification $\alpha^s = (\alpha^a)^r r^m \pmod{p}$ is a valid verification procedure.

10. Consider the following variant of the ElGamal Signature Scheme: Alice chooses a large prime p, a primitive root α, and a secret integer a. She computes $h \equiv \alpha^a \pmod{p}$. The numbers p, α, h are made public and a is kept secret. If $m < p-1$, Alice signs m as follows: She chooses a random integer k and computes $r \equiv \alpha^k \pmod{p}$, with $0 < r < p-1$, and $s \equiv am - kr \pmod{p-1}$. The signed message is (m, r, s). Bob verifies the signature by checking that $\alpha^s r^r \equiv h^m \pmod{p}$. If $m \geq p-1$, she breaks m into blocks and signs each block.

1. Show that if Alice signs correctly then the verification congruence is satisfied.

2. Suppose Eve has a document m_1 and she wants to forge Alice's signature on m_1. That is, she wants to find r_1 and s_1 such that (m_1, r_1, s_1) is valid. Eve chooses $r_1 = 2015$ and tries to find a suitable s_1. Why will it probably be hard to find s_1?

3. Suppose Alice has a very long message m and wants to decrease the size of the signature. How can she use a hash function to do this? Explicitly give the modifications of the above equations that must be done to accomplish this.

11. The ElGamal signature scheme presented is weak to a type of attack known as existential forgery. Here is the basic existential forgery attack. Choose u, v such that $\gcd(v, p-1) = 1$. Compute $r \equiv \beta^v \alpha^u \pmod{p}$ and $s \equiv -rv^{-1} \pmod{p-1}$.

    1. Prove the claim that the pair (r, s) is a valid signature for the message $m = su \pmod{p-1}$ (of course, it is likely that m is not a meaningful message).

    2. Suppose a hash function h is used and the signature must be valid for $h(m)$ instead of for m (so we need to have $h(m) = su$). Explain how this scheme protects against existential forgery. That is, explain why it is hard to produce a forged, signed message by this procedure.

12. Alice's RSA public key is (n, e) and her private key is d. Recall that a document with an RSA signature (m, s) is valid if $m \equiv s^e \pmod{n}$. Bob wants Alice to sign a document m but he does not want Alice to read the document. Assume $m < n$. They do the following:

    1. Bob chooses a random integer k with $\gcd(k, n) = 1$. He computes $m_1 \equiv k^e m \pmod{n}$.

    2. Alice signs m_1 by computing $s_1 \equiv m_1^d \pmod{n}$.

    3. Bob divides s_1 by k mod n to obtain $s \equiv k^{-1} s_1 \pmod{n}$.

    1. Show that (m, s) is valid.

    2. Why is it assumed that $\gcd(k, n) = 1$?

13. Alice wants to sign a document using the ElGamal signature scheme. Suppose her random number generator is broken, so she uses $k = a$ in the signature scheme. How will Eve notice this and

how can Eve determine the values of k and a (and thus break the system)?

14. Suppose Alice signs contracts using a 30-bit hash function h (and h is known to everyone). If m is the contract, then $(m, sig(h(m)))$ is the signed contract (where *sig* is some public signature function). Eve has a file of 2^{20} fraudulent contracts. She finds a file with 2^{20} contracts with valid signatures (by Alice) on them. Describe how Eve can accomplish her goal of putting Alice's signature on at least one fraudulent document.

15.
    1. In several cryptographic protocols, one needs to choose a prime p such that $q = (p-1)/2$ is also prime. One way to do this is to choose a prime q at random and then test $2q+1$ for primality. Suppose q is chosen to have approximately 300 decimal digits. Assume $2q+1$ is a random odd integer of 300 digits. (This is not quite accurate, since $2q+1$ cannot be congruent to 1 mod 3, for example. But the assumption is good enough for a rough estimate.) Show that the probability that $2q+1$ is prime is approximately $1/345$ (use the prime number theorem, as in Section 9.3). This means that with approximately 345 random choices for the prime q, you should be able to find a suitable prime p.

    2. In a version of the Digital Signature Algorithm, Alice needs a 256-bit prime q and a 2048-bit prime p such that $q|p-1$. Suppose Alice chooses a random 256-bit prime q and a random 1792-bit even number k such that $qk+1$ has 2048 bits. Show that the probability that $qk+1$ is prime is approximately 1/710. This means that Alice can find a suitable q and p fairly quickly.

16. Consider the following variation of the ElGamal signature scheme. Alice chooses a large prime p and a primitive root α. She also chooses a function $f(x)$ that, given an integer x with $0 \le x < p$, returns an integer $f(x)$ with $0 \le f(x) < p-1$. (For example, $f(x) = x^7 - 3x + 2 \pmod{p-1}$ for $0 \le x < p$ is one such function.) She chooses a secret integer a and computes $\beta \equiv \alpha^a \pmod{p}$. The numbers p, α, β and the function $f(x)$ are made public.

    Alice wants to sign a message m:

    1. Alice chooses a random integer k with $\gcd(k, p-1) = 1$.
    2. She computes $r \equiv \alpha^k \pmod{p}$.
    3. She computes $s \equiv k^{-1}(m - f(r)a) \pmod{p-1}$.

    The signed message is (m, r, s).

Bob verifies the signature as follows:

1. He computes $v_1 \equiv \beta^{f(r)} r^s \pmod{p}$.
2. He computes $v_2 \equiv \alpha^m \pmod{p}$.
3. If $v_1 \equiv v_2 \pmod{p}$, he declares the signature to be valid.

1. Show that if all procedures are followed correctly, then the verification equation is true.
2. Suppose Alice is lazy and chooses the constant function satisfying $f(x) = 0$ for all x. Show that Eve can forge a valid signature on every message m_1 (for example, give a value of k and of r and s that will give a valid signature for the message m_1).

17. Alice wants to sign a long message $m = m_1||m_2||\cdots||m_L$ that she has broken into blocks $m_1, \ldots, m_L$. She knows that signing each block m_i individually is wasteful, so she computes $g(m) = m_1 \oplus m_2 \oplus \cdots \oplus m_L$ and signs $g(m)$. Her signed message is $(m, sig_A(g(m)))$. Suppose Eve has a message that Alice signed. How can Eve put Alice's signature on fraudulent messages?

18. In some scenarios, it is necessary to have a digital document signed by multiple participants. For example, a contract issued by a company might need to be signed by both the Issuer and the Supervisor before it is valid. To accomplish this, a trusted entity chooses $n = pq$ to be the product of two large, distinct primes and chooses integers $k_1, k_2, k_3 > 1$ with $k_1 k_2 k_3 \equiv 1 \pmod{(p-1)(q-1)}$. The pair (n, k_1) is given to the Issuer, the pair (n, k_2) is given to the Supervisor, and the pair (n, k_3) is made public.

    1. Under the assumption that RSA is hard to decrypt, why should it be difficult for someone who knows at most one of k_1, k_2 to produce s such that $s^{k_3} \equiv m \pmod{n}$?
    2. Devise a procedure where the Issuer signs the contract first and gives the signed contract to the Supervisor, who then signs it in such a way that anyone can verify that the document was signed by both the Issuer and the Supervisor. Use part (a) to show why the verification convinces someone that both parties signed the contract.

13.7 Computer Problems

1. Suppose we use the ElGamal signature scheme with $p = 65539$, $\alpha = 2$, $\beta = 33384$. We send two signed messages (m, r, s):

   $(809, 18357, 1042)$ $(= \text{hi})$ and $(22505, 18357, 26272)$ $(= \text{bye})$.

   1. Show that the same value of k was used for each signature.

   2. Use this fact to find this value of k and to find the value of a such that $\beta \equiv \alpha^a \pmod{p}$.

2. Alice and Bob have the following RSA parameters:

   $$\begin{aligned} n_A &= 171024704183616109700818066925197841516671277, \\ n_B &= 839073542734369359260871355939062622747633109, \\ e_A &= 1571, \quad e_B = 87697. \end{aligned}$$

   Bob knows that

   $$p_B = 9876345769783456893461 3, \quad q_B = 8495789457893457345793$$

   (where $n_B = p_B q_B$). Alice signs a document and sends the document and signature (m, s) (where $s \equiv m^{d_A} \pmod{n_A}$) to Bob. To keep the contents of the document secret, she encrypts using Bob's public key. Bob receives the encrypted signature pair $(m_1, s_1) \equiv (m^{e_B}, s^{e_B}) \pmod{n_B}$, where

   $$\begin{aligned} m_1 &= 418726553997094258577980055061305150940547956 \\ s_1 &= 749142649641548101520133634736865752883277237. \end{aligned}$$

   Find the message m and verify that it came from Alice. (The numbers m_1, s_1, n_A, n_B, p_B, q_B are stored as *sigpairm1, sigpairs1, signa, signb, sigpb, sigqb* in the downloadable computer files (`bit.ly/2JbcS6p`).)

3. In problem 2, suppose that Bob had primes $p_B = 7865712896579$ and $q_B = 8495789457893457345793$. Assuming the same encryption exponents, explain why Bob is unable to verify Alice's signature when she sends him the pair (m_2, s_2) with

   $$\begin{aligned} m_2 &= 148237652324987123445124187171309 30, \\ s_2 &= 431761216284654413401124186720650 63. \end{aligned}$$

   What modifications need to be made for the procedure to work? (The numbers m_2 and s_2 are stored as *sigpairm2, sigpairs2* in the downloadable computer files (`bit.ly/2JbcS6p`).)