Chapter 22 Pairing-Based Cryptography

As we pointed out in the previous chapter, elliptic curves have various advantages in cryptosystems based on discrete logs. However, as we'll see in this chapter, they also open up exciting new vistas for cryptographic applications. The existence of a bilinear pairing on certain elliptic curves is what makes this possible.

First, we'll describe one of these pairings. Then we'll give various applications, including identity-based encryption, digital signatures, and encrypted keyword searches.

22.1 Bilinear Pairings

Although most of this chapter could be done in the context of cyclic groups of prime order, the primary examples of pairings in cryptography are based on elliptic curves or closely related situations. Therefore, for concreteness, we use only the following situation.

Let p be a prime of the form $6q - 1,$ where q is also prime. Let E be the elliptic curve $y^2 \equiv x^3 + 1 \bmod p$. We need the following facts about E.

1. There are exactly $p + 1 = 6q$ points on E.
2. There is a point $P_0 \neq \infty$ such that $qP_0 = \infty$. In fact, if we take a random point $P,$ then, with very high probability, $6P \neq \infty$ and $6P$ is a multiple of P_0.
3. There is a function $\tilde{e}$ that maps pairs of points (aP_0, bP_0) to qth roots of unity for all integers a, b. It satisfies the **bilinearity property**

   $$\tilde{e}(aP_0, bP_0) = \tilde{e}(P_0, P_0)^{ab}$$

   for all a, b. This implies that

   $$\tilde{e}(aP, bQ) = \tilde{e}(P, Q)^{ab}$$

   for all points P, Q that are multiples of P_0. (See Exercise 2.)
4. If we are given two points P and Q that are multiples of $P_0,$ then $\tilde{e}(P, Q)$ can be computed quickly from the coordinates of P and Q.
5. $\tilde{e}(P_0, P_0) \neq 1,$ so it is a nontrivial qth root of unity.

We also make the following assumption:

1. If we are given a random point $P \neq \infty$ on E and a random qth root of unity $\omega \neq 1,$ it is computationally infeasible to find a point Q on E with $\tilde{e}(Q, P) = \omega$ and it is computationally infeasible to find Q' with $\tilde{e}(P, Q') = \omega$.

Remarks

Properties (1) and (2) are fairly easy to verify (see Exercises 4 and 5). The existence of $\tilde{e}$ satisfying (3), (4), (5) is deep. In fact, $\tilde{e}$ is a modification of what is known as the Weil pairing in the theory of elliptic curves. The usual Weil pairing e satisfies $e(P_0, P_0) = 1,$ but the present version is modified using special properties of E to obtain (5). It is generally assumed that (A) is true if the prime p is large enough, but this is not known. See Exercise 10.

The fact that $\tilde{e}(P, Q)$ can be computed quickly needs some more explanation. The two points P, Q satisfy $P = aP_0$ and $Q = bP_0$ for some a, b. However, to find a and b requires solving a discrete log problem, which could take a long time. Therefore, the obvious solution of choosing a random qth root of unity for $\tilde{e}(P_0, P_0)$ and then using the bilinearity property to define $\tilde{e}$ does not work, since it cannot be computed quickly. Instead, $\tilde{e}(P, Q)$ is computed directly in terms of the coordinates of the points P, Q.

Although we will not need to know this, the qth roots of unity lie in the finite field with p^2 elements (see Section 3.11).

For more about the definition of $\tilde{e}$, see [Boneh-Franklin] or [Washington].

The curve E is an example of a **supersingular** elliptic curve, namely one where the number of points is congruent to 1 mod p. (See Exercise 4.) For a while, these curves were regarded as desirable for cryptographic purposes because computations can be done quickly on them. But then it was shown that the discrete logarithm problem for them is only slightly more difficult than the classical discrete logarithm mod p (see Section 22.2), so they fell out of favor (after all, they are slower

computationally than simple multiplication mod p, and they provide no security advantage). Because of the existence of the pairing $\tilde{e}$, they have become popular again.

22.2 The MOV Attack

Let E be the elliptic curve from Section 22.1. Suppose $Q = kP_0,$ where k is some integer and P_0 is the point from Section 22.1. The Elliptic Curve Discrete Log Problem asks us to find k. Menezes, Okamoto, and Vanstone showed the following method of reducing this problem to the discrete log problem in the field with p^2 elements. Observe that

$$\tilde{e}(Q, P_0) = \tilde{e}(kP_0, P_0) = \tilde{e}(P_0, P_0)^k.$$

Therefore, solving the discrete log problem $\tilde{e}(Q, P_0) = \tilde{e}(P_0, P_0)^k$ for k yields the answer. Note that this latter discrete log problem is not on the elliptic curve, but instead in the finite field with p^2 elements. There are analogues of the index calculus for this situation, so usually this is an easier discrete log problem.

For a randomly chosen (not necessarily supersingular) elliptic curve, the method still works in theory. But the values of the pairing usually lie in a field much larger than the field with p^2 elements. This slows down the computations enough that the MOV attack is infeasible for most non-supersingular curves.

The MOV attack shows that cryptosystems based on the elliptic curve discrete log problem for supersingular curves gives no substantial advantage over the classical discrete log problem mod a prime. For this reason, supersingular curves were avoided for cryptographic purposes until these curves occurred in applications where pairings needed to be computed quickly, as in the next few sections.

22.3 Tripartite Diffie-Hellman

Alice, Bob, and Carlos want to agree on a common key (for a symmetric cryptosystem). All communications among them are public. If there were only two people, Diffie-Hellman could be used. A slight extension of this procedure works for three people:

1. Alice, Bob, and Carlos agree on a large prime p and a primitive root α.
2. Alice chooses a secret integer a, Bob chooses a secret integer b, and Carlos chooses a secret integer c.
3. Alice computes $A \equiv \alpha^a \pmod{p}$, Bob computes $B \equiv \alpha^b \pmod{p}$, and Carlos computes $C \equiv \alpha^c \pmod{p}$.
4. Alice sends A to Bob, Bob sends B to Carlos, and Carlos sends C to Alice.
5. Alice computes $A' \equiv C^a$, Bob computes $B' \equiv A^b$, and Carlos computes $C' \equiv B^c$.
6. Alice sends A' to Bob, Bob sends B' to Carlos, and Carlos sends C' to Alice.
7. Alice computes $A'' \equiv C'^a$, Bob computes $B'' \equiv A'^b$, and Carlos computes $C'' \equiv B'^c$. Note that $A'' = B'' = C'' \equiv \alpha^{abc} \pmod{p}$.
8. Alice, Bob, and Carlos use some agreed-upon method to obtain keys from A'', B'', C''. For example, they could use some standard hash function and apply it to $\alpha^{abc} \pmod{p}$.

This protocol could also be used with p and α replaced by an elliptic curve E and a point P_0, so Alice computes aP_0, etc., and the final result is $abcP_0$.

In 2000, Joux showed how to use pairings to obtain a more efficient protocol, one in which there is only one round instead of two:

1. Alice, Bob, and Carlos choose a supersingular elliptic curve E and a point P_0, as in Section 22.1.

2. Alice chooses a secret integer a, Bob chooses a secret integer b, and Carlos chooses a secret integer c.

3. Alice computes $A = aP_0$, Bob computes $B = bP_0$, and Carlos computes $C = cP_0$.

4. Alice makes A public, Bob makes B public, and Carlos makes C public.

5. Alice computes $\tilde{e}(B, C)^a$, Bob computes $\tilde{e}(A, C)^b$, and Carlos computes $\tilde{e}(A, B)^c$. Note that each person has computed $\tilde{e}(P_0, P_0)^{abc}$.

6. Alice, Bob, and Carlos use some agreed-upon method to obtain keys from $\tilde{e}(P_0, P_0)^{abc}$. For example, they could apply some standard hash function to this number.

The eavesdropper Eve sees E and the points P_0, aP_0, bP_0, cP_0 and needs to compute $\tilde{e}(P_0, P_0)^{abc}$. This computation is called the **Bilinear Diffie-Hellman Problem**. It is not known how difficult it is. However, if Eve can solve the Computational Diffie-Hellman Problem (see Section 10.4), then she uses E, P_0, aP_0, bP_0 to obtain abP_0 and computes $\tilde{e}(abP_0, cP_0) = \tilde{e}(P_0, P_0)^{abc}$. Therefore, the Bilinear Diffie-Hellman Problem is no harder than the Computational Diffie-Hellman Problem.

Joux's result showed that pairings could be used in a constructive way in cryptography, rather than only in a destructive method such as the MOV attack, and this led to pairings being considered for applications such as those in the next few sections. It also meant that supersingular curves again became useful in cryptography, with the added requirement that when a curve mod p is used, the prime p must be chosen large enough that the classical discrete logarithm problem (solve $\alpha^x \equiv \beta \pmod{p}$ for x) is intractable.

22.4 Identity-Based Encryption

In most public key systems, when Alice wants to send a message to Bob, she looks up his public key in a directory and then encrypts her message. However, she needs some type of authentication – perhaps the directory has been modified by Eve, and the public key listed for Bob was actually created by Eve. Alice wants to avoid this situation. It was suggested by Shamir in 1984 that it would be nice to have an identity-based system, where Bob's public identification information (for example, his email address) serves as his public key. Such a system was finally designed in 2001 by Boneh and Franklin.

Of course, some type of authentication of each user is still needed. In the present system, this occurs in the initial setup of the system during the communications between the Trusted Authority and the User. In the following, we give the basic idea of the system. For more details and improvements, see [Boneh-Franklin].

We need two public hash functions:

1. H_1 maps arbitrary length binary strings to multiples of P_0. A little care is needed in defining H_1, since no one should be able, given a binary string b, to find k with $H_1(b) = kP_0$. See Exercise 7.

2. H_2 maps qth roots of unity to binary strings of length n, where n is the length of the messages that will be sent. Since H_2 must be specified before the system is set up, this limits the lengths of the messages that can be sent. However, the message could be, for example, a DES key that is used to encrypt a much longer message, so this length requirement is not a severe restriction.

To set up the system we need a Trusted Authority. Let's call him Arthur. Arthur does the following.

1. He chooses, once and for all, a secret integer s. He computes $P_1 = sP_0$, which is made public.

2. For each User, Arthur finds the user's identification *ID* (written as a binary string) and computes

$$D_{\text{User}} = s\, H_1(ID).$$

   Recall that $H_1(ID)$ is a point on E, so D_{User} is s times this point.

3. Arthur uses a secure channel to send D_{User} to the user, who keeps it secret. Arthur does not need to store D_{User}, so he discards it.

The system is now ready to operate, but first let's review what is known:

Public: E, p, P_0, P_1, H_1, H_2

Secret: s (known only to Arthur), D_{User} (one for each User; it is known only by that User)

Alice wants to send an email message m (of binary length n) to Bob, who is one of the Users. She knows Bob's address, which is `bob@computer.com`. This is his *ID*. Alice does the following.

1. She computes $g = \tilde{e}(H_1(\texttt{bob@computer.com}), P_1)$. This is a q th root of unity.

2. She chooses a random $r \not\equiv 0 \pmod{q}$ and computes

$$t = m \oplus H_2(g^r).$$

3. She sends Bob the ciphertext

$$c = (rP_0,\ t).$$

   Note that rP_0 is a point on E, and t is a binary string of length n.

If Bob receives a pair (U, v), where U is a point on E and v is a binary string of length n, then he does the following.

1. He computes $h = \tilde{e}(D_{\text{Bob}},\ U)$, which is a qth root of unity.

2. He recovers the message as

$$m = v \oplus H_2(h).$$

Why does this yield the message? If the encryption is performed correctly, Bob receives $U = rP_0$ and $v = t = m \oplus H_2(g^r)$. Since $D_{\text{Bob}} = s\,H_1$ `(bob@computer.com)`,

$$h = \tilde{e}(D_{\text{Bob}},\, rP_0) = \tilde{e}(H_1,\, P_0)^{sr} = \tilde{e}(H_1,\, sP_0)^r = g^r. \tag{22.1}$$

Therefore,

$$t \oplus H_2(h) = t \oplus H_2(g^r) = m \oplus H_2(g^r) \oplus H_2(g^r) = m,$$

as desired. Note that the main step is Equation (22.1), which removes the secret s from the D_{Bob} in the first argument of $\tilde{e}$ and puts it on the P_0 in the second argument. This follows from the bilinearity property of the function $\tilde{e}$. Almost all of the cryptographic uses of pairings have a similar idea of moving from one masking of the secret to another. The pairing allows this to be done without knowing the value of s.

It is very important that s be kept secret. If Eve obtains s, then she can compute the points D_{User} for each user and read every email. Since $P_1 = sP_0$, the security of s is compromised if Eve can compute discrete logs on the elliptic curve. Moreover, the ciphertext contains rP_0. If Eve can compute a discrete log and find r, then she can compute g^r and use this to find $H_2(g^r)$ and also m. Therefore, for the security of the system, it is vital that p be chosen large enough that discrete logs are computationally infeasible.

22.5 Signatures

22.5.1 BLS Signatures

Alice wants to sign a document m. In earlier chapters, we have seen how to do this with RSA and ElGamal signatures. The BLS method, due to Boneh, Lynn, and Schacham, uses pairings.

We use a supersingular elliptic curve E_0 and point P_0, as in Section 22.1. To set up the signature scheme, we'll need a public hash function H that maps arbitrary length binary strings to multiples of P_0. A little care is needed in defining H, since no one should be able, given a binary string b, to find k with $H(b) = kP_0$. See Exercise 7.

To set up the system, Alice chooses, once and for all, a secret integer a and computes $K_{\text{Alice}} = aP_0$, which is made public.

Alice's signature for the message m is $S = aH(m)$, which is a point on E.

To verify that (m, S) is a valid signed message, Bob checks

$$\tilde{e}(S, P_0) \stackrel{?}{=} \tilde{e}(H(m), K_{\text{Alice}}).$$

If this equation holds, Bob says that the signature is valid.

If Alice signs correctly,

$$\tilde{e}(S, P_0) = \tilde{e}(aH(m), P_0) = \tilde{e}(H(m), P_0)^a = \tilde{e}(H(m), aP_0) = \tilde{e}(H(m), K_{\text{Alice}}),$$

so the verification equation holds.

Suppose Eve wants to forge Alice's signature on a document m. The values of $H(m)$, K_{Alice}, and P_0 are then already determined, so the verification equation says that Eve needs to find S satisfying $\tilde{e}(S, P_0) =$ a known quantity. Assumption (A) is Section 22.1 says that (we hope) this is computationally infeasible.

22.5.2 A Variation

The BLS signature scheme uses a hash function whose values are points on an elliptic curve. This might seem less natural than using a standard hash function with values that are binary strings (that is, numbers). The following method of Zhang, Safavi-Naini, and Susilo remedies this situation. Let H be a standard hash function such as SHA-3 or SHA-256 that maps binary strings of arbitrary length to binary strings of fixed length. Regard the output of H as an integer. Alice's key is the same as in BLS, namely, $K_{\text{Alice}} = aP_0$. But the signature is computed as

$$S = (H(m) + a)^{-1} P_0,$$

where $(H(m) + a)^{-1}$ is the modular multiplicative inverse of $(H(m) + a)$ mod q (where q is the order of P_0, as in Section 22.1).

The verification equation for the signed message (m, S) is

$$\tilde{e}(H(m)P_0 + K_{\text{Alice}}, S) \stackrel{?}{=} \tilde{e}(P_0, P_0).$$

Since $H(m)P_0 + K_{\text{Alice}} = (H(m) + a)P_0$, the left side of the verification equation equals the right side when Alice signs correctly. Again, assumption (A) from Section 22.1 says that it should be hard for Eve to forge Alice's signature.

22.5.3 Identity-Based Signatures

When Alice uses one of the above methods or uses RSA or ElGamal signatures to sign a document, and Bob wants to verify the signature, he looks up Alice's key on a web-page, for example, and uses it in the verification process. This means he must trust the web-page to be correct, not a fake one made by Eve. It would be preferable to use something closely associated with Alice such as her email address as the public key. This is of course the same problem that was solved in the previous section for encyprytion, and similar techniques work in the present situation.

The following method of Hess gives an identity-based signature scheme.

We use a supersingular elliptic curve E and point P_0 as in Section 22.1. We need two public hash functions:

1. H_1 maps arbitrary length binary strings to multiples of P_0. A little care is needed in defining H_1, since no one should be able, given a binary string b, to find k with $H_1(b) = kP_0$. See Exercise 7.

2. H_2 maps binary strings of arbitrary length to binary strings of fixed length; for example, H_2 can be a standard hash function such as SHA-3 or SHA-256.

To set up the system, we need a Trusted Authority. Let's call him Arthur. Arthur does the following.

1. He chooses, once and for all, a secret integer s. He computes $P_1 = sP_0$, which is made public.

2. For each User, Arthur finds the user's identification *ID* (written as a binary string) and computes

$$D_{\text{User}} = s\, H_1(ID).$$

   Recall that $H_1(ID)$ is a point on E, so D_{User} is s times this point.

3. Arthur uses a secure channel to send D_{User} to the user, who keeps it secret. Arthur does not need to store D_{User}, so he discards it.

The system is now ready to operate, but first let's review what is known:

Public: E, p, P_0, P_1, H_1, H_2

Secret: s (known only to Arthur), D_{User} (one for each User; it is known only by that User)

To sign m, Alice does the following.

1. She chooses a random point $P \neq \infty$ on E.
2. She computes $r = \tilde{e}(P, P_0)$.
3. She computes $h = H_2(m \,||\, r)$.
4. She computes $U = hD_{\text{Alice}} + P$.
5. The signed message is (m, U, h).

If Bob receives a triple (m, U, h), where U is a point on E and h is a binary string, then he does the following.

1. He computes $v_1 = \tilde{e}(H_1(ID_{\text{Alice}}), P_1)^{-h}$, which is a qth root of unity.
2. He computes $v_2 = v_1 \cdot \tilde{e}(U, P_0)$.
3. If $h = H_2(m \,||\, v_2)$, then Bob says the signature is valid.

Let's suppose Alice signed correctly. Then, writing H_1 for $H_1(ID_{\text{Alice}})$, we have

$$\begin{aligned} v_1 = \tilde{e}(H_1, P_1)^{-h} &= \tilde{e}(H_1, P_0)^{-hs} = \tilde{e}(sH_1, P_0)^{-h} \\ &= \tilde{e}(D_{\text{Alice}}, P_0)^{-h} = \tilde{e}(-h\, D_{\text{Alice}}, P_0). \end{aligned}$$

Also,

$$\begin{aligned} v_2 = v_1 \cdot \tilde{e}(U, P_0) &= \tilde{e}(-h\, D_{\text{Alice}}, P_0) \cdot \tilde{e}(U, P_0) \\ &= \tilde{e}(-h\, D_{\text{Alice}} + U, P_0) \\ &= \tilde{e}(P, P_0) = r. \end{aligned}$$

Therefore,

$$H_2(m \,||\, v_2) = H_2(m \,||\, r) = h,$$

so the signature is declared valid.

Suppose Eve has a document m and she wants to forge Alice's signature so that (m, U, h) is a valid signature. She cannot choose h arbitrarily since Bob is going to compute v_2 and see whether $h = H_2(m \,||\, v_2)$. Therefore, if H_2 is a good hash function, Eve's best strategy is to choose a value $v_2^{'}$ and compute $h = H(m \,||\, v_2^{'})$. Since H_2 is collision resistant and $H_2(m \,||\, v_2^{'}) = h = H_2(m \,||\, v_2),$ this $v_2{}'$ should equal $v_2 = v_1 \cdot \tilde{e}(U, P_0)$. But v_1 is completely determined by Alice's ID and h. This means that in order to satisfy the verification equation, Eve must find U such that $\tilde{e}(U, P_0)$ equals a given quantity. Assumption (A) says this should be hard to do.

22.6 Keyword Search

Alice runs a large corporation. Various employees send in reports containing secret information. These reports need to be sorted and routed to various departments, depending on the subject of the message. Of course, each report could have a set of keywords attached, and the keywords could determine which departments receive the reports. But maybe the keywords are sensitive information, too. Therefore, the keywords need to be encrypted. But if the person who sorts the reports decrypts the keywords, then this person sees the sensitive keywords. It would be good to have the keywords encrypted in a way that an authorized person can search for a certain keyword in a message and determine either that the keyword is present or not, and receive no other information about other keywords. A solution to this problem was found by Boneh, Di Crescenzo, Ostrovsky, and Persiano.

Start with a supersingular elliptic curve E and point P_0 as in Section 22.1. We need two public hash functions:

1. H_1 maps arbitrary length binary strings to multiples of P_0. A little care is needed in defining H_1, since no one should be able, given a binary string b, to find k with $H_1(b) = kP_0$. See Exercise 7.

2. H_2 maps the qth roots of unity (in the finite field with p^2 elements) to binary strings of fixed length; for example, if the roots of unity are expressed in binary, H_2 can be a standard hash function.

Alice sets up the system as follows. Incoming reports (encrypted) will have attachments consisting of encrypted keywords. Each department will have a set of keyword searches that it is allowed to do. If it finds one of its allotted keywords, then it saves the report.

1. Alice chooses a secret integer a and computes $P_1 = aP_0$.

2. Alice computes

$$T_w = a\, H_1(w).$$

The point T_w is sent via secure channels to each department that is authorized to search for w.

When Daphne writes a report, she attaches the relevant encrypted keywords to the documents. These encrypted keywords are produced as follows:

1. Let w be a keyword. Daphne chooses a random integer $r \not\equiv 0 \pmod{q}$ and computes

$$t = \tilde{e}(H_1(w), rP_1).$$

2. The encryption of the keyword w is the pair

$$[rP_0,\, H_2(t)].$$

A searcher looks at the encrypted keywords $[A,\, b]$ attached to a document and checks

$$H_2(\tilde{e}(T_w,\, A)) \stackrel{?}{=} b.$$

If yes, then the searcher concludes that w is a keyword for that document. If no, then w is not a keyword for it.

Why does this work? If $[A,\, b]$ is the encrypted form of the keyword w, then

$$A = rP_0 \text{ and } t = \tilde{e}(H_1(w), rP_1)$$

for some r. Therefore,

$$\tilde{e}(T_w,\, A) = \tilde{e}(a\, H_1(w), rP_0) = \tilde{e}(H_1(w), rP_0)^a = \tilde{e}(H_1(w), rP_1) = t,$$

so $H_2(\tilde{e}(T_w,\, A)) = H_2(t) = b.$

Suppose conversely, that w' is another keyword. Is it possible that $[A,\, b]$ corresponds to both w and w'? Since the same value of r could be used in the encryptions of w and w', it is possible that A occurs in encryptions of both w and w'. However, we'll show that in this case the number b comes from at most one of w and w'. Since H_1 is collision resistant and $T_{w'} = a\, H_1(w')$ and

$T_w = a\, H_1(w),$ we expect that $T_{w'} \neq T_w$ and $\tilde{e}(T_{w'}, A) \neq \tilde{e}(T_w, A)$. (See Exercise 1.) Since H_2 is collision resistant, we expect that

$$H_2(\tilde{e}(T_{w'}, A)) \neq H_2(\tilde{e}(T_w, A)).$$

Therefore, $[A, b]$ passes the verification equation for at most one keyword w.

Each time that Daphne encrypts the keyword, a different r should be used. Otherwise, the encryption of the keyword will be the same and this can lead to information being leaked. For example, someone could notice that certain keywords occur frequently and make some guesses as to their meanings.

There are many potential applications of this keyword search scheme. For example, suppose a medical researcher wants to find out how many patients at a certain hospital were treated for disease X in the previous year. For privacy reasons, the administration does not want the researcher to obtain any other information about the patients, for example, gender, race, age, and other diseases. The administration could give the researcher T_X. Then the researcher could search the encrypted medical records for keyword X without obtaining any information other than the presence or absence of X.

22.7 Exercises

1. Let E be the supersingular elliptic curve from Section 22.1.

    1. Let $P \neq \infty$ and $Q \neq \infty$ be multiples of P_0. Show that $\tilde{e}(P, Q) \neq 1$. (Hint: Use the fact that $\tilde{e}(P_0, P_0) = \omega$ is a qth root of unity and that $\omega^x = 1$ if and only if $x \equiv 0 \pmod{q}$.)

    2. Let $Q \neq \infty$ be a multiple of P_0 and let P_1, P_2 be multiples of P_0. Show that if $\tilde{e}(P_1, Q) = \tilde{e}(P_2, Q)$, then $P_1 = P_2$.

2. Let E be the supersingular elliptic curve from Section 22.1.

    1. Show that

    $$\tilde{e}(aP, bQ) = \tilde{e}(P, Q)^{ab}$$

    for all points P, Q that are multiples of P_0.

    2. Show that

    $$\tilde{e}(P + Q, R) = \tilde{e}(P, R)\,\tilde{e}(Q, R)$$

    for all P, Q, R that are multiples of P_0.

3. Let E be the supersingular elliptic curve from Section 22.1. Suppose you have points A, B on E that are multiples of P_0 and are not equal to ∞. Let a and b be two secret integers. Suppose you are given the points aA and bB. Find a way to use $\tilde{e}$ to decide whether or not $a \equiv b \pmod{q}$.

4. Let $p \equiv -1 \pmod{3}$ be prime.

    1. Show that there exists d with $3d \equiv 1 \pmod{p-1}$.

    2. Show that if $a^3 \equiv b \pmod{p}$ if and only if $a \equiv b^d \pmod{p}$. This shows that every integer $\bmod p$ has a unique cube root.

    3. Show that $y^2 \equiv x^3 + 1 \pmod{p}$ has exactly $p + 1$ points (including the point ∞). (Hint: Apply part (b) to $y^2 - 1$.) (Remark: A curve mod p whose number of points is congruent to 1 mod p is called *supersingular*.)

5. (for those who know some group theory)

1. In the situation of Exercise 4, suppose that $p = 6q - 1$ with q also prime. Show that there exists a point $P_0 \neq \infty$ such that $qP_0 = \infty$.

2. Let $Q = (2, 3)$, as in Exercise 9 in Chapter 21. Show that if $P \not\in \{\infty, Q, 2Q, 3Q, 4Q, 5Q\}$, then $6P \neq \infty$ and $6P$ is a multiple of P_0. (For simplicity, assume that $q > 3$.)

6. Let H_0 be a hash function that takes a binary string of arbitrary length as input and then outputs an integer mod p. Let $p = 6q - 1$ be prime with q also prime. Show how to use H_0 to construct a hash function H_1 that takes a binary string of arbitrary length as input and outputs a point on the elliptic curve $y^2 \equiv x^3 + 1 \pmod{p}$ that is a multiple of the point P_0 of Section 22.1. (Hint: Use the technique of Exercise 4 to find y, then x. Then use Exercise 5(b).)

7.
    1. In the identity-based encryption system of Section 22.4, suppose Eve can compute k such that H_1 `(bob@computer.edu)` $= k\,P_0$.. Show that Eve can compute g^r and therefore read Bob's messages.

    2. In the BLS signature scheme of Section 22.5.1, suppose Eve can compute k such that $H(m) = k\,P_0$. Show that Eve can compute S such that (m, S) is a valid signed document.

    3. In the identity-based signature scheme of Section 22.5.1, suppose Eve can compute k such that $H_1(ID_{\text{Alice}}) = k\,P_0$. Show that Eve can compute D_{Alice} and therefore forge Alice's signature on documents.

    4. In the keyword search scheme of Section 22.6, suppose Eve can compute k such that $H_1(w) = k\,P_0$. Show that Eve can compute T_w and therefore find the occurrences of encrypted w on documents.

8. Let E and P_0 be as in Section 22.1. Show that an analogue of the Decision Diffie-Hellman problem can be solved for E. Namely, if we are given aP_0, bP_0, cP_0, show how we can decide whether $abP_0 = cP_0$.

9. Suppose you try to set up an identity-based cryptosystem as follows. Arthur chooses large primes p and q and forms $n = pq$, which is made public. For each User, he converts the User's identification ID to a number e_{User} by some public method and then computes d with $de_{\text{User}} \equiv 1 \pmod{\phi(n)}$. Arthur gives d to the User. The integer n is the same for all users. When Alice wants to send an email to Bob, she uses the public method to convert his email address to e_{Bob} and then uses this to encrypt messages with

RSA. Bob knows d, so he can decrypt. Explain why this system is not secure.

10. You are given a point $P \neq \infty$ on the curve E of Section 22.1 and you are given a qth root of unity ω. Suppose you can solve discrete log problems for the qth roots of unity. That is, if $\alpha \neq 1$ and β are qth roots of unity, you can find k so that $\alpha^k = \beta$. Show how to find a point Q on E with $\tilde{e}(Q, P) = \omega$.