Problem 1  An HTTP server can set a cookie in a browser through the Set-Cookie header. This header looks like this:

HTTP/1.1 200 OK
...
Set-Cookie: name=value; Domain=.foo.com; Max-Age=3600

Other attributes can be applied to the cookie, including the Secure attribute considered below.

The cookie will be sent on all subsequent connections (for up the next Max-Age seconds) to servers for which Domain is a suffix; for example, www.intranet.foo.com but not yafoo.com. If the Domain attribute is omitted, most browsers will send the cookie only to the host that set it. The cookie name-value pair is sent by the browser unaccompanied by any attributes such as Domain.

A server is allowed to set the Domain attribute of a Set-Cookie header to any suffix of its own hostname.¹

Browsers store cookies until they expire or the user’s cookie file grows too large; in the latter case, older cookies are expired to make room for newer ones.

Since cookies are used for state management on the Web, we would like the cookie mechanism to provide both confidentiality and integrity. Confidentiality here means that unauthorized servers should not be able to read a cookie set by foo.com; integrity here means that unauthorized servers should not be able to set cookies that will be sent by the browser to foo.com.

a. Against a Web attacker, do cookies set using the Set-Cookie header provide confidentiality? Do they provide integrity?

b. Suppose the Web attacker is additionally allowed to control servers whose hostname is related to the victim’s. For example, suppose that the attacker targeting www.foo.com controls the server at analytics.foo.com. Against this related-domain attacker, do cookies provide confidentiality? Do they provide integrity?

¹But not to a “public suffix” shared by multiple users, such as .com.
c. If the `Set-Cookie` header specifies the `Secure` attribute for a cookie, that cookie will only be sent from the browser to the server over the `https:` scheme, never over the `http:` scheme.

Against a network attacker, do cookies set using the `Set-Cookie` header with the `Secure` attribute provide confidentiality? Do they provide integrity?

Problem 2 The Like button is a widely deployed feature of the Facebook platform. A user who is logged in to Facebook can click on the Like button on a page she visits to express approval of the page or its content. This approval is reflected on that user’s news feed on the Facebook site. The suggested code for implementing a Like button on a page is:

```
<iframe
  scrolling="no" frameborder="0"
  style="border:none; width:450px; height:80px"></iframe>
```

a. Why is the Like button implemented in an iframe, rather than within the enclosing page itself?

b. Is clickjacking a concern for the Like button? What is the worst attack you can think of that is enabled by clickjacking Facebook Like buttons?

c. Should the JavaScript code making up the Like button include framebusting code to prevent clickjacking attacks?

Problem 3 The most commonly used keyed lock is a pin tumbler lock, shown in Figure 1. This type of lock features $k$ pin stacks (typically 4 to 7, with the lock in the figure having 6), each cut at one of $n$ possible depths (typically 4 to 10), which are held in the plug by springs.

A key has bitings at a certain depth for each pin stack. When the key is inserted into the keyway, its bitings push each pin stack up by a certain amounts. The correct key will cause each to be pushed so that its cut aligns with the shear line that separates the plug from the shell, allowing the plug to rotate, as shown in Figure 2.

In many institutional settings a master-keying system is used, where each of several locks (say, one for each room in a dorm) has a different change key, but all share a single master key. This is most commonly implemented by including two cuts in each pin stack, one for the change key, the other for the master key, as shown in Figure 3. The locks will each have different change cuts, but all share the same master cuts, so the same master key will open all of them. The change key and the master key never share a cut: in each pin stack there is one cut for the change key, and another at a different depth (either above or below) for the master key.

a. Suppose that a lock is keyed with the change key at 11111 (five pins, all at depth 1), and the master key at 44444 (five pins again, all at depth 4 this time). Will a key cut at 11411 open the lock? Explain.
Figure 1: On the left, the lock face. On the right, a cutaway view showing six pin stacks and (horizontal) shear line. The pin stacks keep the plug from rotating within the shell and opening the door.

Figure 2: On the left, a cutaway view showing the correct key inserted in the lock. Note that the cuts in the pin stacks are aligned with the shear line. This allows the plug to rotate, as shown on the right.

Figure 3: On the left, a cutaway view of a master-keyed lock. Note that each pin stack has two cuts. On the right, the lock with the change (non-master) key inserted. One set of cuts is at the shear line; the other cut for each stack is either above or below.
b. Suppose you have access to a master-keyed lock, a change key that opens it, but not the master key. In addition, you have a supply of blanks that can be filed down to specific bitings of your choice. Explain how to use the observation from part a. to recover the master key.

As a function of \( k \) and \( n \), how many blanks will you need? How many tries in the lock?

c. Suppose you have access to the change keys for many locks all keyed for the same master key (but, again, no access to the master key). For example, suppose that you measure the biting on the keys given to all the residents in a dorm. Explain how you can (with high probability) recover the master key without any blanks — and even without access to any of the locks!