CS4410 Spring 2009 Homework 7 Solutions

1. Consider the following ways of sending your credit card information to a vendor. Could an attacker who doesn't actually work at the vendor's store steal your credit card data? If so, how would he do it? If not, why?
    a. A browser window with an https URL and only "secure content" on the page. You fill in the boxes and click "Confirm my purchase"

    Yes, but he would probably need to slip some form of spyware (virus) onto your machine that would install itself as a keystroke logger, or break into the vendor's computer system and steal the vendor's secret key. Lacking that key, the messages transmitted between your system and the vendor's website would be pretty secure in this scenario.
    b. Same browser window, but you saw a warning "This page mixes secure and insecure content. Display the insecure content too?". You clicked "OK" (e.g., "yes")

    Yes, either using the same tactics as in (a) or by managing to trick you into putting your credit card data into an insecure frame on the web page.
    c. You sent email to the vendor

    Easily: email is sent in plain text so any credit card information in an email can be read by anyone who is in a position to watch the bits go by on the Internet.
    d. The vendor provided a phone number. You call and give the vendor the credit card information by phone.

    This is probably as secure as (f) provided that you aren't using a faked phone number
    e. You made the purchase in a store and swiped your credit card.

    Yes, if the store saves the credit card information and doesn't adequately protect itself against intrusions that aim at stealing that data.
2. A brilliant PhD student from Mathematics takes cs4410. The class is discussing the RSA algorithm, in which the public key is a pair *(e,n),* where *n* is a product of two large, randomly selected prime numbers. Being bored, he starts thinking about factoring and invents a brilliant new algorithm for factoring large numbers that are a product of exactly two prime numbers. The algorithm runs in time polynomial in the number of digits in *n.*
    a. True or False (why?): This algorithm has the potential to defeat the security of the RSA scheme and eliminate security for much of the modern Internet.

    True. Although there are public key schemes that don't use RSA, RSA is by far the most widely deployed solution. If someone can break RSA, they can potentially break into a wide range of interactions that <u>should</u> be secured.
    b. True or False (why?): The answer to (a) depends on the polynomial.

    Also true, but here things get tricky. Obviously a very high degree polynomial corresponds to very long run times and the algorithm might not turn out to be practical. But <u>any</u> polynomial-time scheme for breaking RSA could turn out to have special cases where it runs way faster. So even a high-degree polynomial solution could be very worrying. On the other hand, it would be so cool to have a cs4410 student solve one of the most important open questions in mathematics, and the Internet is so insecure anyhow, that I think I would be "ok" with such a situation!

3. The Cornell chapter of the Skull and Bones society is developing a new security architecture for calling top-secret meetings, sending each other recipes and announcing the birth of children, weddings, and other life-and-death purposes. They've decided to send cryptographically encoded messages using email.
    a. Invent a reasonably simple scheme that ensures that (i) a message can be sent as a single email with a list of destinations, (ii) any legitimate Skull and Bones member can decode these messages, but (iii) no intruder who intercepts such a message can read it. Keep in mind that new members are added every year (once a member, though, a member for life).

    Suppose that there was a very secret Skull and Bones "key" and that only members have a copy. Than when swearing a new member in, you could give him or her the key to keep in some sensible way (hopefully, not a way that involves anything really gross). Now you can encrypt the body of a message using the S&B key and send the resulting random bit string to anyone you like – post it on the wall of the NYC subway if you wish (or use email if you prefer). You can even include fake destinations that aren't true members. Only legitimate S&B members would have the key to read it.
    b. Over time, some Skull and Bones members die, and when this happens, you must assume that outsiders might gain possession of their laptops, passwords, USB keys, and so forth. Can your proposal from part (a) protect against such problems?
        i. First, assume that the last act of a Skull and Bones member, about to pass away, is to send out a message "Goodbye, cruel world". How would you solve the problem of "cutting off access" to future emails?

        Sounds like we should think in terms of changing the key each time the membership of S&B changes. One way to do that would be to have an S&B website where S&B members can visit. The site would have a list on a per-member basis with the current S&B key encrypted using that member's public key. So an S&B member who is still alive would look up his or her key by finding the list entry with his or her name and then decoding it. The list would just have the identical current key encrypted many times, once per living member. So then you could have, say, an S&B key committee responsible for watching for "goodbye" messages and rekeying each time one is sent out. (There are even trickier solutions in which a quorum of S&B committee members could function as a single fault-tolerant virtual member, to avoid situations in which the key committee itself "fails" by having its members die before there is time to elect new ones). Of course the S&B web site would need to be secure and S&B members might even consider using an anonymous browser to access it (these bounce your requests around in a set of routers worldwide – there are several free ones; the effect is to make it hard to know who accessed the S&B site).
        ii. Next, assume that some members die without sending a final goodbye. Eventually, some living Skull and Bones member finds out, but it may take a while, at which point a "Poor John is gone" message would be sent by the member. How does this change the situation?

        In this case it might be wise to update the key frequently, say once per day. A call to a meeting would use the current key and S&B members would need a

way to access the list of keys that applied over the past few hours to find the "right key" for a given message, but this seems like a simple generalization of the scheme from (i). Once the "Poor John" message reaches the key committee, they would stop encoding the new keys with John's secret key and might even erase past ones on the web site. This way by the time John's kids realize the treasure in his USB, it might already be useless! On the other hand during the period before word spreads about John's demise, the secrecy of the meetings clearly could be compromised.

c. A new society, Blood and Gore, is competing with Skull and Bones. Sometimes they try and infiltrate their members into the Skull and Bones community. To reduce this risk, Skull and Bones starts to require signed "vouchers" from a minimum of five current members before a new member can be trusted. How would you implement these vouchers? What issues are introduced by the scenario of part b/ii?

I would recommend using digital signatures: "I'm S&B member George W. Bush, and I vouch for my good pal Vladimir Putin." When Vladimir presents this, the S&B key committee can confirm that it is a legitimate, unmodified voucher from a legitimate living member of S&B. Vladimir can't get onto the S&B members list without their approval. So it seems pretty easy to support.

4. You are developing a honeyfarm that Citigroup will employ to detect virus attacks on its computer. Your boss, however, is balking at the expense of purchasing 5000 Xen VMM licenses for each honey server: she thinks that 1 O/S should be just fine for 1 machine. How would you explain to her the benefits of using virtualization in a honeyfarm?

The key thing is to make sure she understands that with virtualization, we're creating one computer that "looks like" 5000 machines, not buying 5000 licenses but then using just one or two of them. I would explain that because the VMMs in the honeyfarm aren't working very hard (since virus attacks aren't very common), a single hardware system can potentially "support" a vast number of virtual ized nodes, offering a very cost-effective way to trap virus attacks: when the virus scans Citigroup instead of seeing perhaps 2500 machines on the network, it will see 7500, 2/3 of which are in the honeyfarm. This makes it likely that the virus will attack one of the VMM fakes and trigger an alarm. I might also show her that the cost of a VMM license is pretty low compared to putting a true operating system on a real computer: Xen wants to encourage companies like Citigroup to go crazy thinking of cool uses for Xen and presumably their pricing will reflect this.