Problem Set 1 Solutions

Today's LaTeX fun-fact: quotes are done "like this" and not "like this." (The difference is that you use two back-ticks (`) for open quotes and two single-quotes (') for close quotes. The double-quote mark is never used in LaTeX.) See `http://www.maths.tcd.ie/~dwilkins/LaTeXPrimer/` for a good introductory LaTeX tutorial. *No points were deducted for failure to use the correct LaTeX quotes.*

Problem 1-1. PGP

TA Notes: The biggest mistake made on this problem was confusing (or merging) the separate concepts of *authenticity* and *trust*. When Alice signs Bob's key, she is asserting that the key belongs to Bob, i.e. that the key is *authentic*. She is *not* making any declaration about Bob's inclination to be a responsible keysigner; that would be *trust*. To make an extreme example: suppose Chris is a malicious PGP user who deliberately signs bogus keys, like those of `president@whitehouse.gov`. It is still appropriate for Simson to sign Chris's key (after verifying his identity), even though he is fully aware of Chris's evil ways. It would *not* be appropriate, however, for Simson (or anyone else) to *trust* any of Chris's signatures. For this reason, trust is generally not "transitive" — indeed, it can stop after just one link. PGP is capable of using your *trust* of certain users to infer *authenticity* of new keys, via signatures made by those trusted users.

Many students made the mistake of assuming that the only way to verify the trust on a key was by having the key signed by someone that they trust.

- Simson's PGP 2.6.2 RSA key, downloaded from the PGP key servers, matches the fingerprint on page 248 of the book *PGP: Pretty Good Privacy*, which is available in the library and some bookstores.[1] Simson's PGP IDEA key was signed with the RSA key.

  One student wrote "I believe Simson's key, because under his key there is his photograph. The chances of someone impersonating him to that extent I believe to be very slim." In fact, anybody uploading a PGP key to the PGP keyserver is free to put any photograph they wish on the key. Unless the photograph is signed by a key you trust, it is meaningless.

- Chris's public key is on his website, at `http://theory.lcs.mit.edu/~cpeikert/pubkey.asc`. Although it is certainly possible that somebody could have modified the fingerprint in the *PGP* book and replaced the key on Chris' website, these attacks would most likely have been detected and overcome (by reprinting the book or replacing the incorrect file on the `theory.lcs.mit.edu` webserver).

A few students came to us during our office hours and had us verify our PGP key fingerprints. In-person verification is, of course, the best way to verify a key.

A note on mailers. A few students sent PGP ciphertext or keys by pasting them into Microsoft Outlook and then sending them as HTML text. This effectively broke our ability to import the keys or decrypt the ciphertext. In general, you should be aware of the transformations caused by email clients; Microsoft's clients are especially bad in this respect.

The following solution was submitted by Lucy Jin, Xian Ke, Ryan Manuel, and Matt Wilkerson:

(a) Alice's signature of Bob's key declares Alice's belief that the key really does belong to Bob. It is useful for people to sign each other's keys as a way of attesting to the authenticity of the keys. A third party who has never met Bob can authenticate Bob's key if the third party trusts Alice's signature on that key. Before signing someone's key, one must verify the identity of the keyholder and trust that

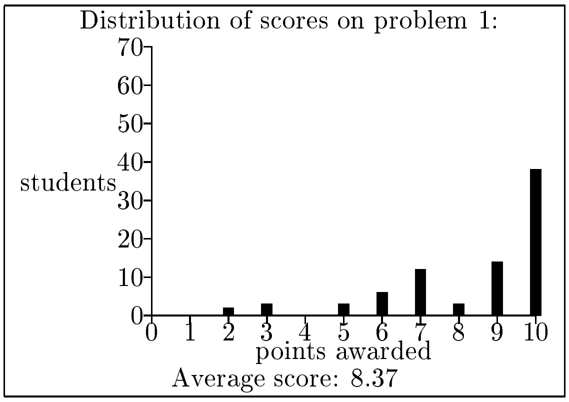
[1] One student noted that the date of Simson's key on the PGP key server was prior to the publication date of his book, but that student didn't bother to check out the fingerprint in the book. This isn't enough to ensure authenticity, though: an imposter could easily upload a key with any date he wished to the keyserver simply by changing the date on his computer when he made the key.

the key has not been tampered with or switched from the one originally generated for the keyholder. Authentication of the keyholder can occur through various means. For example, Bob can hand his key to Alice in person. Verification of the secure transmission of the key can be performed by matching some identifying information (e.g. hash of key) prior to and after transmission. A certificate authority such as Verisign may search credit records or establish phone or mailing contact with the requestor of the signature to verify the keyholder and the key. These verification measures are appropriate to prevent adversaries from impersonating other individuals.

(b) Individual work.

(c) Individual work.

(d) One useful feature of PGP key servers is as a place to store and search for the public keys of individuals. Without such a centralized location, keyholders would need to find their own location to store their public key for retrieval (e.g. web site). Finding a key then becomes quite unwieldy (i.e., web searches may not yield direct results). The drawback with PGP key servers is that forgeries make it difficult to obtain the correct public key. PGP key servers do not require any authentication steps before uploading public keys to the server. Authentication is instead the responsibility of any signers of the key. However, this means that anyone can upload a key claiming that it belongs to anyone else, and have these false keys show up in the search results for the named individual. The search on `president@whitehouse.gov` yielded no fewer than 9 keys with that exact email address on the pgpkeys.mit.edu server, many of them obviously forged ("Cody" is not the first name of any U.S. president in history). Of course, any distributed scheme is also subject to tampering and potential for forgery (e.g. make another web site holding a forged key). In most cases, the benefits of having a PGP key server outweigh the potential for misuse and confusion.



Problem 1-2. Password Sniffing.

Here are some of the answers that were proposed for this problem:

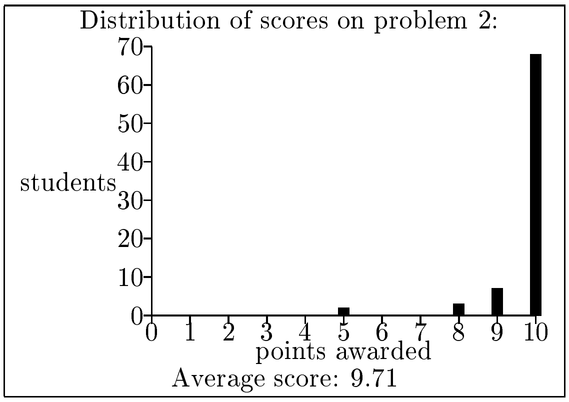
- Ask the user for his or her password. (Asking nicely helps.)
- Shoulder surfing.
- Hardware key logger inside keyboard or between keyboard and host computer.
- Key logger software on the client.
- Operating system modification to capture keystrokes.
- Camera captures passwords.
- Rogue 802.11 access point.
- Capture 802.11 wireless packets.

- Capture Ethernet wired packets.
- Capture RF emissions and decode them.
- The web browser could be rigged to capture Alice's password, either using its legitimate "remember password" feature or using a hidden back door.
- Exploit (possible) flaws in SSL implementation.
- Find a flaw in RSA or another cryptographic algorithm that is used.
- Man-in-the-middle attack between the web browser and the web server.
- A fake web server could capture Alice's password.
- The real web server could capture Alice's password.
- Call the ISP and pretend to be Eve; ask for her password.
- Hack the ISPs authentication server.
- Hack the ISPs authentication database.
- Drug the user and demand the password.

The key thing to realize is that the TELNET protocol is vulnerable to sniffing, whereas the SSL protocol is not. TELNET is further vulnerable to DNS spoofing; SSL is only vulnerable to DNS spoofing if a "valid" certificate for the unspoofed name is at the server. This can be done by loading a fake CA certificate into the SSL client and then issuing a new certificate to the spoof server.

In general, any group that came up with more than five different avenues for attack got 10 points. Groups lost points if they only provided five attacks and if there was significant overlap between those attacks.

So groups that submitted six or more vulnerabilities generally got 10 points. Groups that submitted precisely five had a lower chance of getting 10 points.



Problem 1-3. Exercises in Hashing

TA Notes: The most common mistake in this problem was in arguing that a psuedorandom function was *not* collision-resistant. A typical "proof" used a pseudorandom hash-function with a one-bit output, for which it is obviously easy to find a collision (just try three different inputs). The problem with this reasoning is that, when we use the word "infeasible," it's in reference to how the work required to do a task (e.g., find a collision) *increases as the output length of the hash function grows*. Therefore it isn't meaningful to talk about "feasibility vs. infeasibility" when referring to hash functions with very short outputs. For all practical purposes in this class, it's enough to consider hash functions which have pretty long outputs (e.g., 128 or 160 bits) when talking about infeasibility. Because this is a relatively subtle issue that we didn't go into, not many points were deducted for this kind of mistake.

Also, in the Bitdiddle authentication scheme, the groups of Alexandros Kyriakides, Saad Shakhsir, and Ioannis Tsoukalidis, as well as Rui Viana, Conor Murray, and Richard Hansen pointed out that the actual implementation of MD4 is slightly more complicated than that presented in the book. In particular, *every message* (even one that is an exact multiple of 16 words long) receives some padding before going through the MD4 algorithm. For this reason, the exact attack envisioned by Alyssa P. Hacker probably would not work in practice. However, in principle there still may be message-extension vulnerabilities in certain applications using iterative hash functions like those in the MD4 family (MD4, MD5, SHA-1, etc.).

The following solution was submitted by Siddique Khan, Andrew Tsai, and Dian Chen:

(a) We claim that a collision-resistant hash function is not necessarily pseudorandom. We prove this by demonstrating the following example:

Assume that we have a collision-resistant hash function g, that outputs bit strings of length $n-1$. We define the hash function h that outputs bit strings of length n as follows: $h(x) = 0 \circ x$ if $|x| = n-1$; $h(x) = 1 \circ g(x)$ otherwise.

Now, h is collision resistant because all output strings with first bit 0 map one to one with the inputs x of length $n-1$, and if it was feasible to find a collision on any output strings with first bit 1 this would contradict the fact that g is collision-resistant.

But h is not pseudorandom because it is feasible to tell it apart from a random oracle (RO) using the following test: Given a black box, B, pass some x of length $n-1$ as the input to B. If the output $B(x)$ is $0 \circ x$, then B is very unlikely to be an RO because the outputs of an RO are random and independent of each other. Thus it is feasible to determine whether B implements a Random Oracle or the hash function h. Therefore h is not pseudorandom.

We conclude from this example that a collision resistant hash function is not necessarily pseudorandom.

We claim that a pseudorandom function is collision resistant. We proceed to prove this by proving the contrapositive:

Let g be a hash function that is not collision resistant. Then it is feasible to find inputs x, x' such that $x \neq x'$ and $g(x) = g(x')$.

Now, it is feasible to distinguish g from an RO by the following test: first, find unique x and x' such that $g(x) = g(x')$ since this is feasible. Given a black box B, pass x and x' as inputs to B and observe whether $B(x) = B(x')$. If these outputs are equal, then B is unlikely to be an RO because finding a collision in an RO is infeasible (by a birthday argument). Therefore g is not pseudorandom.

So we conclude the contrapositive: a pseudorandom function is collision resistant.

(b) MD4 is an iterative hash function that takes the data stream input in 16 word chunks (i.e. it first takes s, then r_1, then r_2, etc.). It first passes s and a fixed initial string, I, into its algorithm (denote this algorithm as g). Then it passes the output $g(I, s)$ and r_1 back into g to obtain $g(g(I, s), r_1)$, etc. So we see that $\text{MD4}(s) = g(I, s)$, $\text{MD4}(s \circ r_1) = g(\text{MD4}(s), r_1)$, $\text{MD4}(s \circ r_1 \circ r_2) = g(\text{MD4}(s \circ r_1), r_2)$, etc.

Now in the given authentication scheme, an adversary can always see each of the challenge strings $r_1, r_2, r_3, \ldots, r_k$ as well as each of the previous hash values, $\text{MD4}(s), \text{MD4}(s \circ r_1), \ldots, \text{MD4}(s \circ r_1 \circ \cdots \circ r_{k-1})$. Since the adversary can also know the MD4 algorithm, he can easily compute $\text{MD4}(s \circ r_1 \circ \cdots \circ r_{k-1} \circ r_k) = g(\text{MD4}(s \circ r_1 \circ \cdots \circ r_{k-1}), r_k)$ and send this value to the server. The server will recognise this as the correct hash value and accept him as the client on the kth authentication.

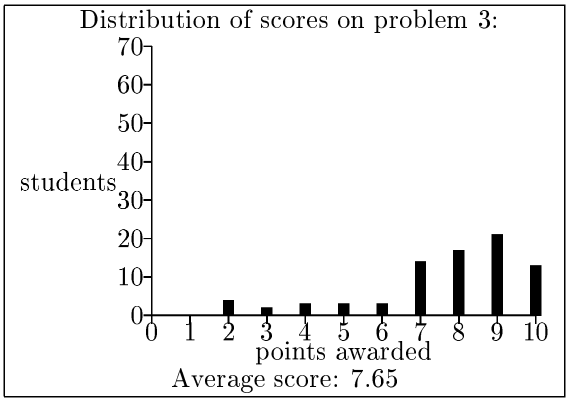
However, if h were an RO instead of MD4, $h(s \circ r_1 \circ \cdots \circ r_{k-1} \circ r_k)$ would be random and independent of all the previous hash values and therefore the adversary cannot feasibly compute this value even though he knows all the previous r_i and hash values.

All the other algorithms in the MD4 family are iterative and are just more complicated forms of MD4, according to the text, so they are all susceptible to the attack outlined above while a random oracle is not. This is what Alyssa meant by her comment.

(c) 1. $h(x) = f(x) \circ g(x)$. Proposition: h is definitely collision resistant.
Proof: Assume h is not collision resistant. Then it is feasible to find unique x, x' such that $h(x) = h(x')$. Suppose f is collision resistant (if g is the collision resistant hash, the argument follows from symmetry): $h(x) = f(x) \circ g(x)$ and $h(x') = f(x') \circ g(x')$ implies $f(x) \circ g(x) = f(x') \circ g(x')$. Then $f(x) = f(x')$ which contradicts our statement that f is collision resistant. (Note: f and g are hash functions, which must therefore have fixed length outputs. This is why we can compare the substrings $f(x)$ and $f(x')$.)

2. $h(x) = f(g(x))$. We provide the following counterexample:
In the case that g is not collision resistant: then it is possible to find unique x, x' such that $g(x) = g(x')$. Then $f(g(x)) = f(g(x'))$ and $h(x) = h(x')$. Therefore, in this case h is not collision resistant.

3. $h(x) = f(g(x)) \circ g(f(x))$. Consider the counterexample:
Let f be collision-resistant and g be some dumb function that returns the same constant-bit output, c, no matter the input.
Then, choose some arbitrary unique pair of inputs x, x': $h(x) = f(g(x)) \circ g(f(x))$, and $h(x') = f(g(x')) \circ g(f(x'))$.
Now $g(f(x)) = g(f(x')) = c$ so last part of concatenation is equal (by the definition of g). And $f(g(x)) = f(g(x'))$ since $g(\cdot) = c$. Therefore $h(x) = h(x')$.
We conclude that h is not definitely collision resistant.



Problem 1-4. Security Policies.

TA Notes: All but 3 groups did the MIT card policy. The other groups did the Apple iTunes policy.

Grading this problem was very difficult. Because of the variance of essays, we tried to implement a uniform scoring system, specifically:

- One point given for an objective.
- One point given for explaining how cards are created.
- One point given for explaining students obligations and restrictions.
- One point given for discussing the role of the MIT in validating who proper students are (e.g., the Registrar tells the Card office who is a student).
- One point given for discussing lost cards.
- One point given for discussing security requirements of other uses (like opening doors, Tech Cash, etc.).
- Four points given for general writing quality, flow and logical consistency.

The following are some common errors that we saw:

- Many students asserted that valid students would be given cards, without explaining what a valid student actually is, or how a determination of validity would be made.
- Many students made students absolutely liable for lost cards. Come on, guys! You don't want to live in that world, do you? Try making security policies that both make sense and that are humane. Don't create a world that you wouldn't want to live in.
- Many students discussed the mechanism of assuring that each card is unique (e.g., giving a card a 128-bit code that is distinct from the student ID number), rather than discussing the motivation for having each card being unique.
- Some students required that a person reporting a stolen or lost card present ID to prove their identity. (What do you do if the whole wallet is stolen? Why not just use the photo that MIT card office has on file?)

(a) The following security policy for the MIT ID Card was turned in by Luxiou Chen, Jason Hung, and Erik Nordlander:

I. Objective An MIT student ID is utilized for accessing doors, checking out books and making purchases on campus. The objective of this policy is to ensure a secure environment for usage of the ID and the privacy of all students. Each student is responsible for following the guidelines outlined in this policy and should not attempt to misuse the ID in any way.

II. MIT Card Creation In order for an MIT student to receive an ID, he must be present at the MIT card office, with a valid proof of identification such as a driver's license or passport. Before receiving a card, the registrar's list of current students will be checked to ensure the proper status of the person as an MIT student. A picture is taken of the student at this time and is placed on the ID. The student is issued one unique card, along with a copy of this security policy. Each card is marked with a date of expiration, after which the account is deactivated.

III. General rules of usage The following is a list of guidelines for the use of the MIT student ID:

1. Use only the student ID that belongs to you — Since the cards are used for making purchases and accessing doors on campus, you should not lend your student ID to another student nor should you use the ID of another student.
2. Protect the MIT card from theft, and should be treated in a similar fashion to a house key.
3. Do not infringe on the integrity of the MIT ID system by doing any of the following:
   (a) Duplicating any student's ID card, including your own.
   (b) Altering data on ID card in order to gain unauthorized access on campus
   (c) Altering data on another student's ID card.
   (d) Attempt to use the ID card following its expiration
4. In the event of a lost ID, immediately report it so the account may be deactivated so as to prevent another student from using it. You will be issued a new unique ID card.
5. In the event of a broken ID card, report it to the card office so as to be issued a new unique ID card.
6. The card is the property of MIT, and must be surrendered upon request such as at the expiration of the card.

IV. Role of MIT card office The role of the MIT card office is to issue each student an ID card. It must also ensure that they do not assign a student more than one ID card. In the event of a lost or broken ID, the office should deactivate the old card and should issue a new unique ID card. The office should also guarantee proper functionality of the card (i.e. the student has the appropriate privileges to access doors on campus and is able to make purchases). A master list of privileges is maintained by the card office. The office is responsible for the deactivation of the MIT card once a person is no longer an MIT student. In order to ensure the privacy of all students, the office should not track the door accesses made by the card.

V. Role of Registrar The role of the registrar is to ensure the MIT card office has a current list of all MIT students and expected time of graduation, in order to determine an appropriate expiration date of the ID card. It should not be involved in the functionality of the card nor should it track any of the accesses or purchases made in order to ensure privacy.

(b) The following security policy for Apple's iPod was submitted by Jamie Hope, Catherine Havasi, and Thomas Lin:

Apple's iPod: Apple's goal regarding the iPod/iTunes system should not be to incur the wrath of the RIAA by leaving the door wide open for piracy, but at the same time should leave the door open enough to keep users interested in the products. The "Don't steal music." sticker is a start, but probably not good enough for the RIAA. We must proceed under the RIAA's assumption that the User is the Enemy.

The traditional security policy of an Internet-based sales system must apply. Transactions at the iTunes Music Store should all be encrypted (via SSL, for example) so that the User feels safe purchasing music. Apple already has experience in this area with the Apple Store. The User should be able to purchase and download music easily, but the system should not facilitate music piracy. The music file should be playable on the User's hardware (or on machines to which he has access) [1], but should not be playable on the computer of another user who had acquired the file via Kazaa or whatever other illegal file sharing utility we assume the User will use. To this end, the music file must be encrypted or otherwise locked such that it may be played only on a computer having a key that is available for download from Apple.

This key should be downloadable only by the User, and the User should not be able to directly manipulate the key or make it available to others. All key acquisitions must be controlled by Apple. Additionally, it should be in the User's best interest not to tell others how they may acquire his key from Apple [2]. Apple should keep a record of which computers are authorized to play music from a particular account using the computer's serial number or some other metric which may be automatically obtained through a network connection to the computer (i.e. does not depend on the User to accurately and honestly enter the information).

The iPod should have access to all the music on the User's computer, whether it be from the iTunes Music Store or imported from the User's CD collection or obtained through other channels, but it should not be a way to transfer this music from one computer to another at Firewire speed. So, music transfer should be allowed from computer to iPod but not from iPod to computer [3]. This is, of course, not possible when the iPod is being used as a removeable hard disk, but hopefully the RIAA won't notice or will realize that it is infeasible to bring a lawsuit against the removeable media industry.

[1] Apple's documentation states that up to three computers at a time may be authorized to play music corresponding to a particular Music Store ID. If playback on a fourth computer is desired, one of the original three must be deauthorized first (The User must have physical access to a computer to deauthorize it.). A single computer may be authorized to play music from multiple IDs. Computers are authorized and deauthorized by selecting a menu item in the iTunes application and then entering an ID and passphrase.

[2] This is the reasoning behind the three-computer limit, or at least why the number of computers on which the file is playable must be limited; if the User publishes instructions on how to authorize a computer to play his music, at most two other computers can take advantage of the instructions. Furthermore, now the User cannot authorize a second computer of his own. Since he does not have physical access to the other two machines with his authorization, he cannot revoke their authorizations, which he would have to do to authorize another machine of his own.

[3] Apple's solution to this is to put the music files in a hidden directory on the iPod's hard disk; see Version Tracker `http://www.versiontracker.com` for a number of utilities which circumvent this block.

References:

Apple Help Viewer files for iPod, iTunes, and the iTunes Music Store, `http://www.info.apple.com`.
Apple iPod product page, `http://www.apple.com/ipod/`.
Apple iTunes product page, `http://www.apple.com/itunes/`.