University of Maryland
CMSC456 — Introduction to Cryptography
Professor Jonathan Katz

---

# Problem Set 5

**Due at the *beginning* of class on Nov. 2**

1. Say you have an internet-disabled portable music device (iPoMd) that "shuffles" the songs it contains and plays them at random. (That is, the next song that is played is always chosen uniformly from all songs on the device.)

   (a) If you have 400 songs on your iPoMd, how many songs do you expect to listen to before, with probability roughly 1/2, you hear *some* song repeat?

   (b) Say you want to load enough songs onto you iPoMd so that, with probability 1/2, you can listen to 30 consecutive songs before hearing a repeat. How many songs do you need?

2. MD5 was a popular hash function that was broken a few years ago, but is still used in some software. Download the code for MD5 from http://www.fourmilab.ch/md5/.

   A *$t$-bit collision in MD5* is a pair of unequal strings whose MD5 hashes are equal in their initial $t$ bits. Write a program that takes as input an argument $t$ and finds $t$-bit collisions in MD5 using a "birthday attack". (Your program need only handle values $t \leq 40$.) Run your program 10 times for each of $t = 4, 8, 12, 16, 20, 30$ and tabulate, for each $t$, the average number of evaluations of MD5 until a collision was found. What do you expect, and how does this compare to what you observe?

   Turn in a collision for $t = 40$, a graph of your results, and a discussion of how this compares to the theoretical prediction.

3. For each of the following modifications to the Merkle-Damgård transformation (Construction 4.13), determine whether the resulting hash function is collision resistant or not. If yes, provide a proof sketch (a formal reduction is not needed, but you should provide enough detail that one could be constructed); if not, describe an attack.

   (a) Modify the construction so the input length is not used at all; i.e., output $z_B$ instead of $z_{B+1}$.

   (b) Instead of outputting $h^s(z_B \| L)$, output $z_B \| L$.

4. Let $F$ be a length-preserving, keyed function. Say that, given oracle access to $F_k$ for a randomly-chosen key $k$, it is possible to determine $k$ using only 100 queries to the oracle (and minimal computation). Prove formally that $F$ cannot be a pseudorandom permutation. (I.e., show and analyze an explicit attack.)

5. Consider a modified substitution-permutation network where instead of carrying out the key-mixing, substitution, and permutation steps in alternating order for $r$ rounds,

the cipher first applies $r$ rounds of key-mixing, then carries out $r$ rounds of substitution, and finally applies $r$ mixing permutations. Analyze the security of this construction as a function of the number of rounds.

6. On the webpage you will find code for a primitive block cipher `cipher` and a "wrapper" function `cipher_hiddenkey`.

   (a) Write a program `inverse` that takes as input a 16-byte key and an 8-byte block, and such that for any key $k$ and any $x$, `inverse`($k$, `cipher`$(k, x)) = x$.

   (b) Implement a key-recovery attack on the cipher. The program implementing your attack should call `cipher_hiddenkey` only; i.e., your program should not use any knowledge of the key. (That is, even though you will know the key — since you will need to hard-code it into `cipher_hiddenkey` — your attack should make no use of this information, and should work for any value of the key.)

   *Note:* Although the attack described in the book should work (with appropriate implementation), you can (and should!) apply further optimizations to make your attack run more quickly.

   In addition to submitting your working code, please describe your attack in pseudocode; provide a rough complexity analysis (as in class) of how long your attack takes; and write down how long (in hours/minutes) your attack takes.