

Problem Set 5 — Solutions

1. I wasn't looking for a numerically-exact answer here. I was just looking for an application of the "birthday problem". Specifically, if you choose q items out of a set of size N , you expect to have a repeat with probability roughly $q^2/2N$. In the first problem $N = 400$, and we are looking for q such that $q^2/2N \approx 0.5$. This works out to $q = 20$. In the second case we have $q = 30$ and are looking for N such that $q^2/2N \approx 0.5$. This works out to $N = 900$.
2. The basic birthday attack applies the hash to N distinct inputs, *and then checks to find a collision among all pairs of outputs*. When using this to find a collision in a t -bit hash, you expect it to succeed with probability roughly half when $N = 2^{t/2}$ (this is just another application of the birthday problem).

Remark: If your program does a naive comparison among all pairs of N hash values, then the running time of your program will be $N^2 \approx 2^t$, even though you still perform only $N \approx 2^{t/2}$ hash evaluations (which is what I asked you to measure. As noted in the book, to improve this you would sort the hash values in $O(N \log N)$ time, and then check for a match in linear time.

3. (Not required.)
4. Let \mathcal{A} be an algorithm that, given oracle access to F_k , outputs k (in polynomial time). The "natural" distinguisher would run \mathcal{A} , answering the queries of \mathcal{A} using its own oracle, and then output 1 if \mathcal{A} outputs *some* key, and 0 otherwise. The problem with this is that we have no idea what happens when \mathcal{A} is given oracle access to a random function — maybe it always outputs some key in that case also! So we have to be a little more careful.

Consider the following distinguisher D given access to an oracle $\mathcal{O} : \{0, 1\}^n \rightarrow \{0, 1\}^n$:

$D^{\mathcal{O}(\cdot)}(1^n)$
Run $\mathcal{A}(1^n)$, answering the oracle queries of \mathcal{A} using \mathcal{O}
Eventually, \mathcal{A} outputs some key k
Choose an arbitrary $x \in \{0, 1\}^n$ that was not previously queried to \mathcal{O}
If $\mathcal{O}(x) = F_k(x)$, output 1; else output 0

(We assume without loss of generality that \mathcal{A} always outputs a key. This is easy to ensure by, for example, taking k to be the all-0 key if \mathcal{A} fails to output anything.) In the last line, above, $\mathcal{O}(x)$ is obtained by querying the oracle, while $F_k(x)$ is computed using the given values k and x and the fact that D knows the description of F .

Let us analyze the behavior of D . When $\mathcal{O} = F_k$ for a randomly-chosen key k , then (by assumption) \mathcal{A} outputs this key. So the test in the last line of D will always succeed, and we have

$$\Pr[D^{F_k(\cdot)}(1^n) = 1] = 1.$$

(In the above probability, k is chosen uniformly at random.) On the other hand, if \mathcal{O} is a completely random function, then *no matter what key k is output by \mathcal{A}* , the probability that $F_k(x) = \mathcal{O}(x)$ is 2^{-n} . (This is true because x was not previously queried to \mathcal{O} , and so $\mathcal{O}(x)$ is a random n -bit string.) Therefore:

$$\Pr[D^{f(\cdot)}(1^n) = 1] = 1,$$

where f is chosen uniformly at random from the set of all functions on n bits. We conclude that $|\Pr[D^{F_k(\cdot)}(1^n) = 1] - \Pr[D^{f(\cdot)}(1^n) = 1]|$ is not negligible, and so F is not a pseudorandom function.

5. Recall that key mixing just involves XORing a sub-key with the current state. So r rounds of key mixing using the r sub-keys k_1, \dots, k_r is equivalent to one round of key mixing using the single key $k_1 \oplus \dots \oplus k_r$.

Similarly, r sequential rounds of substitution are equivalent to one round of substitution. To see this, assume for concreteness that S -boxes operate on 4 bits at a time, and look at what happens to the 4 left-most bits. These bits are passed through the S -boxes S_1^1, \dots, S_1^r (where S_1^i represents the left-most S -box in round i). But this is just equivalent to passing the 4 left-most bits through the single S -box $S_1^r \circ \dots \circ S_1^1$ (where “ \circ ” is just function composition).

Finally, the same sort of argument applies to the mixing permutation: applying r mixing permutations in a row is equivalent to applying a single mixing permutation.

Thus, this approach is equivalent to a 1-round SPN, which we know is insecure.