Problem Set 2 Solutions

Problem 2-1. Primes

Write an efficient program (i.e. probabilistic polynomial time) that takes as input a string s and produces a number likely to be a prime and whose high order bytes are s. The input s is a byte string, and the output number should be printed in decimal.

For example, the string "rivest+abhi+fubob" is

```
0x72 69 76 65 73 74 2B 61 62 68 69 2B 66 75 62 6F 62 00 00
```

when each ASCII character is converted to its hexadecimal representation and two zero bytes are appended. The two extra zero bytes provide the flexibility needed to modify this number into a prime without having to change the original message. In decimal form, our starting number is 2551472030826120444243689867576938224490250240. By flipping some of the bits in the last two bytes of this number, we can generate the following number which is likely prime: 2551472030826120444243689867576938224490250261.

This solution comes from Muhammad Zia Hydari, Alexander Yip, and Sophia Yuditskaya:

We used Miller-Rabin randomized primality testing algorithm for our program. Some precomputation was done to avoid unnecessary checking. We made sure that the number was odd by checking the last appended byte. Moreover, we tested our candidate number to make sure it was not divisible by 3, 5 or 7. According to Bruce Scheier, (*Applied Crytography*) this eliminates 54 percent of the odd numbers.

We also tested `isPrime()` with 23 Carmichael numbers [1]. All of these numbers were rejected in just one call to `isPrime()` (the base used was 7). The Carmichael numbers we used were: 561, 1105, 1729, 2465, 2821, 6601, 8911, 10585, 15841, 29341, 41041, 46657, 52633, 62745, 63973, 75361, 101101, 115921, 126217, 162401,146843929, 161242705, 2023528501.

The number of times that `isPrime()` is called with a random base each time can be set based on the level of confidence that the user desires. The algorithm is false-biased Monte Carlo so it is alway right when it declares that a number is not prime. However, when the algorithm declares the number as prime, the probability of it being wrong is at most $(1/4)^n$ where n is the number of times the `isPrime()` is run. Knuth proves in the solution to Exercise 22 of section 4.5.4 (*Volume 2: Seminumerical Algorithms*) that the probability of failure of `isPrime()` is at most 1/4. This upper bound is true regardless of the value of the number being tested. In practise however, the algorithm rarely fails. So if we repeat the test a number of times with a random base the upper bound on the probability of failure becomes negligible. (e.g., it would be less than $(1/4)^{23}$ if we run `isPrime()` 23 times)

If we express our candidate prime as $p = 1 + 2^t * u$ and if the random chosen base is a, we know that $a^u \pmod p$ can be calculated in $O(\log u)$ multiplication operations. Then a further t multiplications will generate a^{p-1}. We did not brute-force the last two bytes - multiples of 2, 3, 5, and 7 were sifted and not fed into Miller-Rabin testing.

The source code for this solution is on the 6.857 Web page.

[1] obtained from ftp.dpmms.cam.ac.uk/pub/Carmichael/carmichael-14.gz

Problem 2-2. Reuse, reduce, collide

This solution comes to you from David Bailey, Eric J. Cholankeril, Dennis Kwon, and Misha Zitser.

Let $g : \{0,1\}^* \rightarrow \{0,1\}^n$ be a hash function mapping arbitrarily long binary strings to n-bit strings. Define the hash function $h(x) = g(g(x))$.

(a) Argue that if g is one-way (pre-image resistant) then h is one-way.

Suppose g is one-way. We will prove by contradiction that h is also one-way.

Proof. Suppose h is not one-way. Then there exists an adversary with a magic box A that can invert h on a non-negligible fraction of inputs, i.e given an input y, A outputs x such that $h(x) = g(g(x)) = y$. We now construct a magic box B that can invert g on a non-negligible fraction of its inputs. We let $B(y) = g(A(y))$. So, given an input y for B, we first feed it into A. A outputs some x. We take x and output $g(x)$. For a non-negligible fraction of inputs y, A will output an x such that $h(x) = g(g(x)) = y$. Whenever A outputs such an x, $g(x)$ will be a correct pre-image of y under g. Therefore, for a non-negligible fraction of inputs y, B successfully inverts g. This contradicts the fact that g is one-way!

(b) Argue that if g is collision-resistant, then h is collision-resistant.

Suppose g is CR. We will prove by contradiction that h is also CR.

Proof. Assume h is not CR. So, we can find distinct inputs x_1 and x_2 s.t $h(x_1) = h(x_2)$, namely $g(g(x_1)) = g(g(x_2))$. Let $y = g(x_1)$ and $y_2 = g(x_2)$. If $y_1 = y_2$, then x_1 and x_2 form a collision pair for g because $g(x_1) = g(x_2)$. If $y_1 \neq y_2$, then y_1 and y_2 form a collision pair for g because $g(y_1) = g(y_2)$. In either case, we contradict the fact that g is CR.

(c) Argue that if g is weakly collision-resistant (i.e. second preimage-resistant), then h need not be weakly collision-resistant. Although a counter-example would be splendid, it is sufficient to sketch how it might be possible to have a counter-example.

Proof. Suppose g is WCR, but not CR. Presumably there are many such functions g. If h is to be WCR, then given a random x_1 it should be hard to find an x_2 such that $x_2 \neq x_1$ and $h(x_1) = g(g(x_1)) = h(x_2) = g(g(x_2))$. However, finding an x_2 satisfying the latter conditions does seem possible. If x_2 is such that $g(x_1) = g(x_2)$, then we will have found a collision with x_1 under g, and thus we will have contradicted the fact that g is WCR. On the other hand, it might well be possible that give a random x_1, we can easily find an $x_2 \neq x_1$ such that $g(x_1) \neq g(x_2)$ and $h(x_1) = g(g(x_1)) = g(g(x_2)) = h(x_2)$. If the latter were to be true, we would not in any way contradict the fact that g is WCR. We would simply have found a collision pair for g, namely $g(x_1)$ and $g(x_2)$, but since g is not CR, this is completely acceptable. We will also not have contradicted the fact that g is WCR because given a random x we still can't easily find a y such that $g(x) = g(y)$. Therefore, we have no reason to rule out that a function g, as just described, exists. If such a g exists, h would not be WCR.

The answer above was sufficient for this problem set. However, one group (Steven Chan, Samitha Samaranayake, Buddhika Kottahachchi, and Hoeteck Wee) constructed the following WCR $g(x)$ such that $h(x)$ is not WCR. Their definfition of WCR is not exactly the same as presented in class. But it's close enough.

If weakly collision-resistant hash functions do not exist, then the statement holds vacuously. Suppose on the other hand that weakly collision-resistant hash functions do exist; then, let f be one such function, where $f : \{0,1\}^* \to \{0,1\}^n$, and consider $g : \{0,1\}^* \to \{0,1\}^n$ defined as follows:

$$g(x) = \begin{cases} 0^n & \text{if } x \in \{0,1\}^n \\ f(x) & \text{otherwise} \end{cases}$$

We claim that g is also a weakly collision-resistant function. The intuition is as follows: suppose on the contrary that g is not weakly collision-resistant, so there exists some efficient adversary $\mathcal{A}$ that for a randomly chosen input x could return $x' \neq x$ such that $g(x') = g(x)$ with some non-negligible probability p. Note that the probability ϵ that a randomly chosen x is in $\{0,1\}^n$ is negligible, since $\{0,1\}^n$ is finite, and the input space is infinite. Therefore, furthermore, $\mathcal{A}$ does in fact find a collision for f if $x \notin \{0,1\}^n$. Hence, $\mathcal{A}$ constitutes an efficient adversary that could find a collision for f on a randomly chosen input x with probability at least $p - \epsilon$, which is non-negligible (since p is non-negligible and ϵ is negligible). Thus f is not weakly collision-resistant, a contradiction.

For this choice of g, $h(x) = g(g(x)) = 0^n$ for all $x \in \{0,1\}^*$, since $g(x) \in \{0,1\}^n$. Clearly, h is not weakly collision-resistant since any choice of $x' \neq x$ yields a collision.

Adam Smith, a former 6.875 TA, provided a similar solution. Let D denote $\{0,1\}^n$. Assume we have a WCR function f mapping D^2 to D. (e.g., for x and y in D, $f(x,y)$ is also in D).

Define g mapping D^2 to D^2 via:

$$g(x,y) = \begin{cases} (0, f(x,y)) & \text{if } x \neq 0^n \\ (0,0) & \text{if } x = 0^n \end{cases}$$

Claim: g is WCR. Well, this depends on the definition of WCR, but if an adversary is given a randomly chosen (x,y) and also $g(x,y)$, then he should have a negligible chance of finding (x',y') such that $g(x',y') = g(x,y)$, if f is WCR.

Definition: $h(x,y) = g(g(x,y))$

Claim: h is not WCR. *Proof.* $h(x,y) = (0,0)$ for all x, y; any (x,y) is a pre-image of $(0,0)$.

Problem 2-3. Voting

Evaluate the following voting proposal. Note advantages and deficiencies and suggest improvements or remedies as appropriate. Or, if you think the scheme can not be improved sufficiently to be usable, say why. Limit your discussion to one page.

The election officials for a county with n voters generate n random RSA keys and place each public/private key pair on a separate floppy disk. The officials also keep a record of $h(PK)$ for each public key PK, for some suitable hash function h. Each floppy has a random two-digit number R stuck on it with a post-it note; this number is also recorded with the corresponding $h(PK)$. During the month before the election, each voter goes to the county courthouse, identifies and authenticates herself, and grabs a random floppy from the box of floppies. (The election official should not be able to correlate the voter with the floppy he took.) The voter memorizes the two-digit number and discards the post-it note before leaving with the floppy.

The voter creates her ballot B on her home computer (or, if the voter does not have a PC, on one at the local high school). He uses the private key SK from the floppy to sign the pair (B,R), and sends B, R, PK, and the signature through an anonymizing email channel over the Internet to the county courthouse.

The county computer receives everything, computes $h(PK)$, confirms it is on the list, checks that R is correct, verifies the digital signature, and if everything checks, adds (B, R, PK) to a list of "validated ballots." When the election is over, the validated ballots are tallied.

There were many excellent reports describing the advantages and shortcomings of this voting system. There was no single right answer – just answers that had justification. Here is the report from Ivan Davtchev, Kalin Rahnev, Shane Cruz, and Jang Kim:

After carefully evaluating the proposed voting architecture, we realized that there are many problems with the system that would need to be addressed in order to achieve the voting requirements we have discussed in class.

One of the most compelling reasons that we determined this scheme is not satisfactory is the fact that there is no assurance given to each voter that their vote has been successfully conveyed to the county courthouse and tallied. After a user sends their signed ballot from their home computer, the message could get lost in the network, eaten-up by the anonymous email channel, or altered and rendered invalid by an adversary and therefore the vote would never get counted. The voter would never get notification that this vote was not received and therefore would not know that their vote was not counted. Due to the need to keep the votes anonymous and the fact that the public-private key pairs are random, we determined that it would be almost infeasible to change this system to allow for voter assurance. Along these lines, the scheme for accepting ballots received over the anonymous email channel only seems to verify that (B, R, PK) is valid and does not record that the vote is received. Therefore, an adversary could successfully use replay attacks to re-send a ballot multiple times in order to vote more than once.

Another large problem we found with this system is the difficulties that arise with the use of floppy disks to store the random key pairs. What happens if a voter picks up their disk at the county courthouse and then loses it during the month of the election? That voter would lose their ability to vote since they have already been checked off at the courthouse as receiving a floppy disk. Also, if some third-party adversary finds the lost disk, he would be able to "steal" the person's vote by using her floppy disk. The random two-digit R helps to deter this, but the adversary would easily be able to try the 100 different possible R's and therefore "steal" the vote. Another weakness associated with the floppies is that the post-it note may have the wrong number, it may also fall off, be switched by someone, or the user may forget the two-digit number associated with their disk. It may also be possible for a voter to grab more than one disk from the box of floppies. These floppies also bring up the problem of compatibility issues, disk errors, and other general problems that are associated with the use of floppy disks. Requiring a floppy disk to vote also makes absentee ballots and provisional ballots almost impossible to implement.

The third major category with which we found problems is with the overall ease-of-use for the public. If a person does not have a computer or Internet connection at home, they need to make two trips (one to the courthouse and one to the library) in order to place a vote. This may be too much of a hassle for some people. Also, people who are unfamiliar with cryptography and authentication may not feel comfortable with the system as a whole. By making the system more complex, you may make citizens less comfortable. This added complexity also requires more trust to be placed in the system, and more possible holes in security. With this new architecture, we must trust the people who generate the key pairs, the courthouse officials who distribute the disks, the software used to create the ballots, the anonymous email channel, and the courthouse officials who receive the votes.

Overall, we found this voting proposal to have some interesting ideas, but we determined it to be too weak to implement. Due to all the inherent weaknesses associated with the use of floppy disks, we decided that we could not remedy this system without completely changing the architecture.