Introduction to Cryptography
Solution to Maman 14

Yehuda Lindell

January 3, 2008

Solution 1: *Technical computation, thus omitted.*

Solution 2: In order to compute a, we first compute $\phi(n) = 28 \cdot 10 = 280$. Then, we need $3a = 1 \bmod 280$. If we take $a = 187$, we have $3a = 561$ where $561 = 1 \bmod 280$. Thus, the value of a in the secret key is 187. An encryption of 100 is just $100^3 \bmod 319$. In order to compute this, calculate $100^3/319$ and you obtain 3134.796... Then, we have that $100^3 = 3134 \cdot 319 + 254$ and thus $100^3 = 254 \bmod 319$.

Solution 3:

1. In this case, it is possible to simply compute the 26 encryptions of each character of the alphabet. Then, a ciphertext can be decrypted by just looking at which of the 26 blocks appeared, and mapping it back to the appropriate plaintext character.

2. (Note that the question is to choose 26 strings $r_A, r_B, \ldots, r_Z$; one random string for each character.) In this case, the attacker cannot re-encrypt because it doesn't know the random strings. Nevertheless, it can carry out the statistical attack on a substitution cipher (details omitted).

3. In this case the encryption is secure because a different r is used each time. This is similar to the idea of random padding used in the PKCS #1 standard (v1.5).

Solution 4: For RSA: given $y_1 = x_1^b \bmod n$ and $y_2 = x_2^b \bmod n$ we have that $y_1 \cdot y_2 = (x_1^b \bmod n) \cdot (y_2 = x_2^b \bmod n) \bmod n$. But as we know, this equals $x_1^b \cdot x_2^b \bmod n$ which in turn equals $(x_1 \cdot x_2)^b \bmod n$. Thus the product of two ciphertexts yields an encryption of the product of the two plaintexts. This holds for all b, including $b = 2$, and thus for Rabin as well.

The more interesting part of the question relates to "boosting" the success of an inversion algorithm. Let A be an algorithm that succeeds in inverting 1% of the ciphertexts. Of course, we don't know which 1%. Given a key (b, n) and a ciphertext $y = x^b \bmod n$, we wish to compute x. We can first invoke A on input (b, n, y) and see if A returns the correct x (this can be checked because we can just compute $x^b \bmod n$ and see if it equals y). However, we have no guarantee as to A's success on this specific input. Rather, we choose a random r and compute $s = r^b \bmod n$ and $y' = y \cdot s \bmod n$. Then, we invoke A on input (b, n, y'). Let x' be the value returned by A. If $(x')^b \bmod n = y'$, then it follows $x' = x \cdot r$ and so we output $x = x'/r \bmod n$. Otherwise, we keep trying. We can try 5000 times.

What is the probability of success? Notice that the value y' is *uniformly distributed* among the number $1, \ldots, n-1$. Therefore, by the assumption on A's success, it succeeds in inverting 1% with probability 0.01. We have that in every attempt, the probability that A fails is $(1 - \frac{1}{100})$. Thus, the probability that we fail in 5000 attempts is:

$$\left(1 - \frac{1}{100}\right)^{5000} = \left(1 - \frac{1}{100}\right)^{100 \cdot 50} < \left(\frac{1}{e}\right)^{50}$$

which is an incredibly small fail rate. Of course, we can reduce or increase the number of attempts, depending on what fail rate we desire. We can also continue trying until we succeed in which case we will run in expected polynomial-time and always succeed.

Solution 5: Let the public key of a public-key encryption scheme be denoted PK. Furthermore, let c be a ciphertext computed by encrypting the plaintext string m. Now, since decryption is unique, it must hold that there is a single m for which $e_{PK}(m) = c$. Therefore it is possible to try to encrypt all possible strings m using all possible random coins in the encryption procedure until the ciphertext c is output. Again, since decryption is unique, there can only be one m for which this procedure succeeds. This attack is not efficient. However, for perfect secrecy the adversary need not be efficient. Thus, we prove that it is always possible to decrypt and so perfect secrecy cannot hold.