Preparing for Midterm #1

MATH 481/581 — Cryptography
Wednesday, Feb 21

Please print your name:

Problem 1.

(a) Do the practice problems that were compiled from the examples from lectures. (Solutions to these can be found in the corresponding lecture sketches.) In particular, fill in all the conceptual empty boxes. To save time, you don't need to work through all details. However, make sure that you know how to do each problem.

(b) A collection of online homework questions that are particularly relevant for the midterm is posted on our website. (Recall that redoing problems can never hurt your scores.)

(c) Do the problems below. (Solutions are posted separately.)

Bonus challenge. Let me know about any typos you spot in the posted solutions (or lecture sketches). Any typo, that is not yet fixed by the time you send it to me, is worth a bonus point.

Problem 2. Eve intercepts the ciphertext $c = (1111\ 1011\ 0000)_2$ from Alice to Bob. She knows that the plaintext begins with $m = (1100\ 0\ldots)_2$.

(a) Eve suspects that a stream cipher with PRG $x_{n+1} \equiv 5x_n + 1 \pmod{16}$ was used for encryption. If that's the case, break the cipher and determine the plaintext. What is your verdict on Eve's suspicion?

(b) On second thought, Eve thinks a stream cipher using a LFSR with $x_{n+3} \equiv x_{n+2} + x_n \pmod 2$ was used. If that's the case, what would be the plaintext?

(c) If a nonce was used, how would that affect Eve's attack?

(d) What should Alice learn from this? (Obviously, apart from the fact that the key space is too small.)

Solution.

(a) Since $c = m \oplus \mathrm{PRG}$, we learn that the initial piece of the keystream is $\mathrm{PRG} = m \oplus c = (1100\ 0\ldots)_2 \oplus (1111\ 1\ldots)_2 = (0011\ 1\ldots)_2$.

Since each x_n has 4 bits, we learn that $x_1 = (0011)_2 = 3$. Using $x_{n+1} \equiv 5x_n + 1 \pmod{16}$, we find $x_2 = 0$, $x_3 = 1$, ... In other words, $\mathrm{PRG} = 3, 0, 1, \ldots = (0011\ 0000\ 0001\ \ldots)_2$.

Hence, the plaintext would be $m = c \oplus \mathrm{PRG} = (1111\ 1011\ 0000)_2 \oplus (0011\ 0000\ 0001)_2 = (1100\ 1011\ 0001)_2$.

However, note that the 5th bit does not agree with Eve's knowledge about m. This means that the message could not have been encrypted using this particular stream cipher.

(b) Again, the initial piece of the keystream is $\mathrm{PRG} = m \oplus c = (1100\ 0\ldots)_2 \oplus (1111\ 1\ldots)_2 = (0011\ 1\ldots)_2$.

Now, each x_n is a single bit, and we have $x_1 = 0$, $x_2 = 0$, $x_3 = 1$. The given LFSR produces $x_4 = x_3 + x_1 = 1$, $x_5 = x_4 + x_2 = 1$, $x_6 = 0$, $x_7 = 1$, and so on. Continuing, we obtain $\mathrm{PRG} = x_1x_2\ldots = (0011\ 1010\ 0111)_2$.

Hence, the plaintext would be $m = c \oplus \mathrm{PRG} = (1111\ 1011\ 0000)_2 \oplus (0011\ 1010\ 0111)_2 = (1100\ 0001\ 0111)_2$.

(c) Whether or not a nonce was used has absolutely no effect on Eve's attack.

Armin Straub
straub@southalabama.edu

(Recall that the nonce is combined with the key in order that a different seed is used each time the stream cipher is used to encrypt a message.)

(d) Alice should learn that, when using a stream cipher, we need a PRG which is unpredictable. Linear congruential generators and LFSRs, by themselves, are completely predictable: knowing a few terms, one can predict (even with certainty), as we did in the first two parts, all future (even past) terms.

For instance, Alice could use the Blum-Blum-Shub PRG, which is believed to be unpredictable. (However, let us note that B-B-S is usually considered too slow for practical applications.) □

Problem 3.

(a) Evaluate $850^{6677} \pmod{77}$.

(b) Evaluate $100^{7300} \pmod{91}$.

(c) Determine all solutions to $x^2 \equiv 9 \pmod{91}$.

Solution.

(a) Obviously, $850^{6677} \equiv 3^{6677} \pmod{77}$. We note that $\gcd(3, 77) = 1$, so that we may apply Euler's theorem.

Since $\phi(77) = \phi(7)\phi(11) = 6 \cdot 10 = 60$, we conclude that $3^{6677} \equiv 3^{17} \pmod{77}$.

Using binary exponentiation, $3^2 = 9$, $3^4 \equiv 4$, $3^8 \equiv 16$, $3^{16} \equiv 25$.

Hence, $3^{17} = 3^{16} \cdot 3^1 \equiv 25 \cdot 3 \equiv 75 \pmod{77}$. In summary, $850^{6677} \equiv 75 \pmod{77}$.

(b) Obviously, $100^{7300} \equiv 9^{7300} \pmod{91}$. We note that $\gcd(9, 91) = 1$, so that we may apply Euler's theorem.

Since $\phi(91) = \phi(7)\phi(13) = 6 \cdot 12 = 72$, we conclude that $9^{7300} \equiv 9^{28} \pmod{91}$.

Using binary exponentiation, $9^2 \equiv -10$, $9^4 \equiv (-10)^2 \equiv 9$, $9^8 \equiv -10$, $9^{16} \equiv 9$.

Hence, $9^{28} = 9^{16} \cdot 9^8 \cdot 9^4 \equiv 9 \cdot (-10) \cdot 9 \equiv 9 \pmod{91}$. In summary, $100^{7300} \equiv 9 \pmod{91}$.

Comment. Actually, to save time, we should have paused when we found $9^4 \equiv 9 \pmod{91}$. As discussed in the second part, this means that $9^3 \equiv 1 \pmod{91}$. This means that we can reduce the exponent in $9^{28} \pmod{91}$ modulo 3. Since $28 \equiv 1 \pmod{3}$, we conclude that $9^{28} \equiv 9^1 = 9 \pmod{91}$.

(c) By the CRT,

$$x^2 \equiv 9 \pmod{91} \iff \begin{matrix} x^2 \equiv 9 \pmod{7} \\ x^2 \equiv 9 \pmod{13} \end{matrix} \iff \begin{matrix} x \equiv \pm 3 \pmod{7} \\ x \equiv \pm 3 \pmod{13} \end{matrix}.$$

In particular, there are four solutions to $x^2 \equiv 9 \pmod{91}$. Two of these are obvious: $x \equiv \pm 3 \pmod{91}$. To find the other two, we apply the CRT to solve $x \equiv 3 \pmod{7}$ and $x \equiv -3 \pmod{13}$. This produces

$$x \equiv 3 \cdot 13 \cdot \underbrace{13^{-1}_{\bmod 7}}_{-1} - 3 \cdot 7 \cdot \underbrace{7^{-1}_{\bmod 13}}_{2} = -39 - 42 \equiv 10 \pmod{91}.$$

Hence, we conclude that the solutions to $x^2 \equiv 9 \pmod{91}$ are ± 3, ± 10.

□

Problem 4.

(a) Using the Chinese remainder theorem, solve $x \equiv 3 \pmod{4}$, $x \equiv 1 \pmod{7}$, $x \equiv 2 \pmod{11}$.

(b) Using the Chinese remainder theorem, find all solutions to $x^3 \equiv 1 \pmod{70}$.

(c) Determine the number of solutions to $x^3 \equiv 1 \pmod{182}$.

If you wish additional practice using the CRT, find all (or just a select few) solutions.

Solution.

(a) $x \equiv 3\cdot 7\cdot 11 \cdot \underbrace{(7\cdot 11)^{-1}_{\bmod 4}}_{1} + 1\cdot 4\cdot 11 \cdot \underbrace{(4\cdot 11)^{-1}_{\bmod 7}}_{4} + 2\cdot 4\cdot 7 \cdot \underbrace{(4\cdot 7)^{-1}_{\bmod 11}}_{2} = 231 + 176 + 112 \equiv 211 \pmod{308}$

(b) By the CRT:

$$x^3 \equiv 1 \pmod{70} \iff \begin{matrix} x^3 \equiv 1 \pmod 2 \\ x^3 \equiv 1 \pmod 5 \\ x^3 \equiv 1 \pmod 7 \end{matrix} \iff \begin{matrix} x \equiv 1 \pmod 2 \\ x \equiv 1 \pmod 5 \\ x \equiv 1, 2, 4 \pmod 7 \end{matrix}$$

Here, for the last equivalence, we just go through all residues (there's only $1, 4$ and 6 many invertible residues in each case) to find the possible values for x. Note that, in the end, there is $1\cdot 1\cdot 3 = 3$ possible combinations for x.

Using the CRT, we then construct the corresponding values modulo 70:

$$\begin{aligned}
&x\equiv 1 \pmod 2, \quad x\equiv 1 \pmod 5, \quad x\equiv 1 \pmod 7 \iff x \equiv 1 \pmod{70} \\
&x\equiv 1 \pmod 2, \quad x\equiv 1 \pmod 5, \quad x\equiv 2 \pmod 7 \iff x \equiv 1\cdot 5\cdot 7 \cdot \underbrace{(5\cdot 7)^{-1}_{\bmod 2}}_{1} + 1\cdot 2\cdot 7 \cdot \underbrace{(2\cdot 7)^{-1}_{\bmod 5}}_{-1} + \\
&\qquad\qquad 2\cdot 2\cdot 5 \cdot \underbrace{(2\cdot 5)^{-1}_{\bmod 7}}_{-2} = 35 - 14 - 40 \equiv 51 \pmod{70} \\
&x\equiv 1 \pmod 2, \quad x\equiv 1 \pmod 5, \quad x\equiv 4 \pmod 7 \iff x \equiv 1\cdot 5\cdot 7 \cdot \underbrace{(5\cdot 7)^{-1}_{\bmod 2}}_{1} + 1\cdot 2\cdot 7 \cdot \underbrace{(2\cdot 7)^{-1}_{\bmod 5}}_{-1} + \\
&\qquad\qquad 4\cdot 2\cdot 5 \cdot \underbrace{(2\cdot 5)^{-1}_{\bmod 7}}_{-2} = 35 - 14 - 80 \equiv 11 \pmod{70}
\end{aligned}$$

Hence, we conclude that $x^3 \equiv 1 \pmod{70}$ has the three solutions $1, 11, 51 \pmod{70}$.

(c) Note that $182 = 2\cdot 7\cdot 13$. As in the previous part, by the CRT:

$$x^3 \equiv 1 \pmod{182} \iff \begin{matrix} x^3 \equiv 1 \pmod 2 \\ x^3 \equiv 1 \pmod 7 \\ x^3 \equiv 1 \pmod{13} \end{matrix} \iff \begin{matrix} x \equiv 1 \pmod 2 \\ x \equiv 1, 2, 4 \pmod 7 \\ x \equiv 1, 3, 9 \pmod{13} \end{matrix}$$

Again, since there is only 12 invertible residues modulo 13 we just checked all of them, and found that $1, 3, 9$ are the only ones satisfying $x^3 \equiv 1 \pmod{13}$. (Of course, if we have to do more such problems, we should spend a little extra time thinking about better ways. The comment below might give you one idea for optimization.)

In the end, there is $1 \cdot 3 \cdot 3 = 9$ possible combinations for x modulo 182. In other words, there is exactly 9 solutions to $x^3 \equiv 1 \pmod{182}$.

As in the previous part, you can work out all nine solutions using the CRT. The solutions are $1, 9, 29, 53, 79, 81, 107, 113, 165 \pmod{182}$.

Comment. By the way, it is no coincidence that the solutions modulo 7 and 13 are of the form $1, a, a^2$. This follows from $a^3 - 1 = (a-1)(1+a+a^2)$. Can you see how?

□

Problem 5.

(a) When using a stream cipher, why must we not use the same keystream a second time?

(b) Explain how a nonce makes it possible to use the same key in a stream cipher multiple times.

(c) During a conversation you hear the statement that "the one-time pad is perfectly secure". What is your reaction?

(d) Your company is implementing measures for secure internal communication. As part of that, a random secret key is to be generated for each employee. A colleague says: "That's easy, let me do it! Java has a built-in class called `Random`. It shouldn't be more than a few lines of code." What is your reaction?

(e) We observed that many programming languages use linear congruential generators when producing pseudo-random numbers. If these are predictable, why are they still used?

Solution.

(a) In short, using the same keystream twice is the same as using a one-time pad twice. As indicated by the name, this is a terrible sin.

In more detail, recall that a message m is encrypted as $c = m \oplus \text{KEYSTREAM}$. Using the same keystream twice, we can combine the ciphertexts $c_1 = m_1 \oplus \text{KEYSTREAM}$ and $c_2 = m_2 \oplus \text{KEYSTREAM}$ to obtain

$$c_1 \oplus c_2 = (m_1 \oplus \text{KEYSTREAM}) \oplus (m_2 \oplus \text{KEYSTREAM}) = m_1 \oplus m_2.$$

This is information about the plaintexts! (In fact, if the plaintexts are text encoded in ASCII this is usually enough to obtain both m_1 and m_2 from $m_1 \oplus m_2$.)

(b) Without a nonce, we use the key as the seed for the PRG. This means that the same key with result in the same keystream. As discussed in the previous item, we therefore cannot use the same key twice.

The idea of a nonce is to combine it with the key when determining the seed for the PRG (for instance, nonce and key could be simply concatenated). Hence, as long as we change the nonce, we can keep the same key, but will have a different seed each time, which results in a different keystream. The nonce is not kept secret but instead passed along with the ciphertext (so that the person on the other hand, who already knows the secret key, now can also compute the seed and the resulting keystream).

(c) The statement is too bold, and it only focuses on one aspect of security. The one-time pad provides perfect confidentiality, but it is not secure against tampering.

(d) You should stop your colleague! The typical random number generators built into programming languages are not meant for cryptographic applications. They are often slight variations of linear congruential generators or LFSRs. These are terribly predictable. As a result, if Eve gains information on the secret key of one or several employees, she might be able to predict other secret keys. A malicious employee could even try to extrapolate from the secret key he already has.

Comment. In fact, Java also provides a class `SecureRandom`, which might fit your bill. However, you should look into how that is implemented on your platform and whether any security concerns have been raised since then.
http://stackoverflow.com/questions/11051205/difference-between-java-util-random-and-java-security-securerandom

(e) Because they are fast. It helps that they are very simple to implement, too. Also, we need to be mindful of the vastly different applications for pseudorandom numbers. For instance, casual computer games will need random numbers, in which case our cryptographic concerns do not apply. In fact, the (insecure) simple PRGs often do have very pleasant statistical properties and "look" nice and random from that point of view.

One could argue that one could certainly aim at finding something that's almost as fast and "more" secure than the typical standard implementations. However, that might be even more dangerous because it could lull some people into using them for crypto when they still are not secure enough.

One needs to be aware of the requirements. □

Problem 6.

(a) If you can only do a single modular computation, how would you check whether a huge randomly selected number N is prime or not?

(b) Which flaw of the Fermat primality test renders it unsuitable as a general primality test? How can this flaw be fixed?

(c) Despite the flaw in the previous item, in which scenario is it fine to use the Fermat primality test regardless?

Solution.

(a) Compute $2^{N-1} \pmod N$ (using binary exponentiation). If this is $2^{N-1} \not\equiv 1 \pmod N$, then N is not a prime. [There's nothing special about 2, by the way.]

Otherwise, N is a prime or 2 is a Fermat liar modulo N (but the latter is exceedingly unlikely for a huge randomly selected number N; a bonus challenge from class indicates that this is almost as unlikely as randomly running into a factor of N).

(b) There exist composite numbers n such that every residue a is either a Fermat liar or $\gcd(a, n) > 1$ (in which case, a reveals a factor of n, which is as unlikely as finding a divisor of n by trial division). For these numbers (called absolute pseudoprimes) the Fermat primality test would usually suggest the wrong conclusion that the number is a prime.

The issue is fixed by the Miller–Rabin primality test, an extension of the Fermat primality test.

(c) When testing a large randomly generated number for primality. The reason is that Fermat liars are extremely rare among large numbers. □

Problem 7.

(a) Express 123 in base 2 (and then in base 7).

(b) Predict the number of solutions to $x^2 \equiv 4 \pmod{1001}$.

(c) After how many terms must the LFSR $x_{n+7} \equiv x_{n+3} + x_n \pmod 2$ repeat?

Solution.

(a) $123 = (1111011)_2 = (234)_7$.

(b) Observe that $1001 = 7 \cdot 11 \cdot 13$. (Of course, you are not expected to factor numbers of this size on the exam without calculators.)

By the CRT:

$$x^2 \equiv 4 \pmod{1001} \iff \begin{matrix} x^2 \equiv 4 \pmod 7 \\ x^2 \equiv 4 \pmod{11} \\ x^2 \equiv 4 \pmod{13} \end{matrix} \iff \begin{matrix} x \equiv \pm 2 \pmod 7 \\ x \equiv \pm 2 \pmod{11} \\ x \equiv \pm 2 \pmod{13} \end{matrix}$$

Again, using the CRT, we can combine each of these $2^3 = 8$ possibilities for x to a solution modulo 1001. In particular, there are 8 different solutions to $x^2 \equiv 4 \pmod{1001}$.

(c) The LFSR computes the next value x_{n+7} from the past 7 values $(x_n, x_{n+1}, ..., x_{n+6})$. Hence, it has $2^7 = 128$ many states. Since the state $(0, 0, 0, 0, 0, 0, 0)$ remains zero forever, 127 states remain. This means that the generated sequence must be periodic, with period at most 127.

Indeed, for any initial condition (except the one where everything is 0), the period is 127. □