

---

## Problem Set 4

**Revised November 8, 2010.**

This problem set is due on *Monday, November 15, 2010* by 9AM. You should submit your homework via SPARK<sup>1</sup>.

You may write your solution in any program (plaintext, Word, L<sup>A</sup>T<sub>E</sub>X, etc.), but you must either (A) convert your document to PDF for attachment in SPARK or (B) use the textbox dialog box in SPARK to submit a plaintext response. While you may write your solution in any format for your own use, we will not accept attachments other than PDFs in SPARK. If you choose to instead submit a handwritten solution in class, you will not be eligible for the “typesetting” point. Please contact the TA well ahead of the deadline if you have a question about these procedures.

This is a team problem set. Separate teams may not collaborate. See Handout 1 (*Course Information*) for our policy on collaboration and late penalties. When appropriate, mark the top of each page with your name, cs466, the problem set number and question, and the date<sup>2</sup>. Please begin each solution on a new page. Points may be deducted if your TA has problems understanding your solution. In mathematical problems, **show all your work**.

### Problem 4-1. Typesetting and attribution(1 pt)

If you submit a nicely typeset solution via SPARK, you will earn an extra point. Handwritten solutions will not.

In one paragraph, explain the precise role each team member served in this problem set (problem by problem). The work may be divided up in any manner, but all team members should participate. All students will be expected to understand the material for the final exam. If the staff decides that a particular team member does not contribute sufficiently, that team member will have to write their own solutions separately for future homework.

### Problem 4-2. For the \* they are a-changin’ (7 pts)

Montgomery reductions allow for fast modular multiplication, especially when the modulus remains static for many multiplications and when the Montgomery residue of a multiplicand can be pre-computed and reused. Modular exponentiation produces exactly such an environment. In this problem, you will learn exactly how much faster Montgomery reductions can help the performance of modular multiplications.

Make sure to use avoid all trial division (only bit shifts and the % operator) in Montgomery. Also make sure to use the fast version of MM for  $\tilde{z} = MM(\tilde{x}, \tilde{y})$  as given by, for instance, Algorithm 14.36 on page 602 on <http://www.cacr.math.uwaterloo.ca/hac/about/chap14.pdf>. The version of `modexp` that integrates MM is explained in Algorithm 14.94 on page 620 in the same PDF. Note that  $R$  and  $R^2$  are generally computed once at startup and reused to save time.

The TA has implemented part of this system for you. Look for an email addendum that explains how to get the bonus code for the MSP430 simulator.

---

<sup>1</sup><http://spark.oit.umass.edu>

<sup>2</sup>We will distribute our favorite solution for each problem to the class as the “official” solution—this is your chance to become famous!

**(a) Implementation**

Implement four versions of the iterative version of modular exponentiation (see Stinson page 177) on the MSP430 simulator. The four versions are: the original `modexp` algorithm, `modexp` using the Chinese Remainder Theorem, original `modexp` algorithm using Montgomery residues and Montgomery reductions, `modexp` using Montgomery residues and Montgomery reductions and the Chinese Remainder Theorem. Submit your code and proof that the four algorithms produce consistent results on the following test cases where  $n = 881 * 883$  and  $a = 13$ .

- $a^{1023} \bmod n$
- $a^{1024} \bmod n$
- $a^{1025} \bmod n$

Note that 881 and 883 are both primes<sup>3</sup>.

**(b) Performance**

Run performance tests of your four algorithms on various modulus lengths (picking appropriately sized primes). For each of the modulus sizes you pick (we suggest a binary search to get started), run at least 3 trials for each algorithm where a trial means randomly selecting a base and exponent and performing `modexp`. Graph the performance with a line chart. The x-axis should be the bit length of the modulus, and the y-axis should be the running times of the four `modexp` algorithms running on the same input values. Pick a range of bit lengths such that you see “interesting” results that show crossover points in performance.

Submit your graph.

**(c) Good better best**

Under which conditions does each algorithm perform the best on the MSP430 and why?

**Problem 4-3. Side channels [4 pts]**

Consider a simple power analysis of RSA decryption. If the adversary can measure the power consumption (under constant voltage) of a microcontroller, then one might be able to determine the bits of the secret exponent  $d$ . Without countermeasures, an implementation of RSA decryption will create side channels that leak information. If the microcontroller computes a squaring or multiplication, the power consumption will increase. Because of slight delays between stages of the iterative loop of `modexp`, one can see power consumption temporarily drop between loop iterations. This reveals when a new iteration is about to begin. Furthermore, one can identify from the trace of power consumption whether an iteration has done just squaring (a short duration of high power consumption) or both squaring and multiplication (a longer duration of high power consumption).

Answer the following questions with simple power analysis—assuming the standard RSA setup ( $n = pq, ed = 1 \bmod \phi(n)$ ) and `modexp` using a left-to-right iterative algorithm as specified in Stinson page 177 in Algorithm 5.5.

[We will distribute a power trace online]

1. Identify the respective rounds in the power consumption figure and mark each round as either “S” for squaring only, or “SM” for both squaring and multiplication.

---

<sup>3</sup>AKA twin primes

2. Assume that the starting values for `modexp` have already been initialized and that the power trace is exactly of the `for` loop within `modexp`. How many bits are in the exponent  $d$ ?
3. What is the value of  $d$ ?
4. Discuss one countermeasure (you may search other literature, but you must cite any sources you use).

#### Problem 4-4. Hash functions and MACs [5 pts]

##### (a) Hash properties

Suppose a message  $m$  of length  $kl$  bits is divided into blocks of equal length  $k$  bits:  $m = M_1 || M_2 || \dots || M_l$ . Let  $h(x) = M_1 \oplus M_2 \oplus \dots \oplus M_l$ . Which of the three properties (PIR, CR, 2PIR) hold for the function  $h$  and why?

Now assume that the bit length message  $m$  is not a multiple of  $k$  bits. Thus, the last block will be partial. Consider the following approaches when  $m = M_1 || M_2 || \dots || m_r$  where  $m_r$  represents the remainder of bits that does not fill an entire block.

Do problem 4.9(b) from Stinson on page 157. Below we give one solution for problem 4.9(a) to serve as a learning exercise.

For the purpose of contradiction, let's assume that  $h_2$  is *not* collision resistant. Then, there exists an adversary who can find  $x$  and  $y$  such that  $h_2(x) = h_2(y)$  where  $x \neq y$ .

$$h_2(x) = h_1(h_1(x_1) || h_1(x_2))$$

$$h_2(y) = h_1(h_1(y_1) || h_1(y_2))$$

where  $x = x_1 || x_2$  and  $y = y_1 || y_2$ .

Let  $v = h_1(x_1) || h_1(x_2)$  and  $w = h_1(y_1) || h_1(y_2)$ .

Then,  $h_1(v) = h_1(w)$ , and we found collision for  $h_1$ . This contradicts the given condition that  $h_1$  is a collision resistant hash function, therefore  $h_2$  is a collision resistant hash function.

##### (b) Number theoretic hash functions

Consider a function  $G$  defined as  $G(x) = g^{h(x)} \pmod p$ , where  $p$  is a large prime,  $g$  is a generator modulo  $p$ , and  $h$  is a hash function that is CR and PIR. For the purposes of this exercise, let us assume that the size of the output of  $h$  is at least twice as small the size of  $p$ . Is  $G$  PIR? Is  $G$  2PIR? Make sure to prove your answers rigorously. Is  $G$  a good hash function? Explain.

##### (c) MAC-then-Encrypt or Encrypt-then-MAC (postponed until PS5)

##### POSTPONED!

This problem should illustrate an example of the failure of MAC-then-Encrypt authentication schemes against chosen plaintext attacks (the curious reader may see Krawczyk's *The Order of Encryption and Authentication For Protecting Communications* for a formal treatment). Consider a scheme where in order to obtain confidentiality and authentication a user first breaks his/her message  $m$  into a sequence of blocks  $m_1 || m_2 || \dots || m_l$  (each of the same size  $n$ ) and then attaches a MAC that is a hash of the  $\oplus$  (exclusive-or) of all the blocks of the message. The size of the output of the hash function is also  $n$ . The resulting sequence of blocks (together with the MAC

attached to the end of  $m$ ) is encrypted via a symmetric block cipher in CBC mode using a randomly generated initial vector which becomes the first block of the ciphertext  $c = IV || c_1 || c_2 || \dots c_l$ . For this exercise let us assume access to a *strong* cryptographic hash function. The recipient of the message must first decrypt the message and then check that the hash of the  $\oplus$  of all the blocks of the message (without the very last block) is equal to the decryption of the very last block of the ciphertext. The secret key is known only to the sender and the recipient. Design a chosen plaintext attack in which an adversary can request from the sender no more than a polynomial (in  $n$ ) number of plaintext queries in order to obtain their encryption. An attack is successful when a new ciphertext is produced, distinct from all that were once requested from the sender, that is accepted as authentic by the recipient. Now, do the same for a scheme where the MAC is simply  $\oplus$  of all the blocks of the message (no hash functions involved).

Hint: it is helpful to start by drawing a diagram of all the processes involved in MAC formation, encryption, decryption, and authentication.

#### Problem 4-5. DoppelB(1)ock Cipher [3 pts]

A German company is attempting to make an extra strong block cipher. The company decided that composing block cipher encryption multiple times in their DoppelBock Cipher product might increase security, so they first try running a conventional block cipher encryption algorithm E twice

$$c = E_{k_1}(E_{k_2}(m))$$

with  $n$ -bit keys  $k_1$  and  $k_2$ . Thus, the total key size is  $2n$  bits. Unfortunately, a highly paid consultant explained that the new cipher is not much more secure than the original algorithm E under a chosen plaintext attack.

- (a) Show how to break this scheme using a chosen plaintext attack that requires  $O(2^n)$  memory and  $O(2^n)$  encryptions/decryptions using AES.
- (b) Not to give up easily, the company fired back with Mark II of the DoppelBock Cipher.

$$c = \text{AES}_{k_1}(\text{AES}_{k_2}^{-1}(\text{AES}_{k_1}(m)))$$

where  $\text{AES}_k$  means AES encryption under key  $k$  and  $\text{AES}_k^{-1}$  means AES decryption under key  $k$ .

Describe a chosen plaintext attack on this two-key version of DoppelBock when  $n = 128$ , requiring roughly  $2^{128}$  steps and storage of  $2^{128}$  encryptions under single AES encryptions.