Information Security:

Theory vs. Reality

Exercise 2

Tel Aviv University

0368-4474-01, Winter 2015-2016

Lecturer: Eran Tromer

Teaching assistant: Tzvika Geft

Submit a ZIP file containing all your work to istvr1516.course@gmail.com by 29 December 2015.

Include your name and ID in the subject and the submitted ZIP file's name.

Submissions are individual and must be done independently of other students and any course-specific reference material. Using reference material that is not course-specific is OK, if you state so explicitly in your code and the body of the submission email; provide a pointer to its origin; and adapt it to precisely and elegantly answer the actual question.

Question 1

You found a device that performs AES encryption using an unknown 128-bit key that you would like to discover.

The device implements AES using a microcontroller, and stores the AES state bytes in the 8-bit registers of the microcontroller. There is a specific byte register which is always used as index into the S-box lookup table in the SubBytes stage (stage 3.1 in the AES overview below), i.e., every time the SubByte operation substitutes a state byte b by Sbox[b], the register contains b.

You decide to measure and analyze the power consumption of the device (by connecting a small resistor between the device and its power supply and using an oscilloscope to measure the voltage fluctuations on the resistor during operation). You tell this to Doc Brown. He asks you to sit down and wait, and runs into his laboratory with your device. He returns after a while holding a paper sheet. "Here's one of the key bytes! And I only had to encrypt 200 plaintexts to get it", he says, hands you the sheet and wanders off.

On the sheet are the Doc's findings: a vector of size 200 with integers between 0 and 255 – titled "plaintext slices", and a matrix is of size 200x525 – titled "traces".

A. Explain the likely meaning of the Doc's sheets.

The content of the measurements done by Doc Brown is attached to this exercise (see below).

B. Write a program which performs the analysis and outputs the value of the attacked key byte. The program should receive two command-line arguments: the name of a file containing "plaintext slices", followed by the name of a file containing "traces". The input files are in textual CSV format, as in the attached example. The program should output a single decimal integer: its guess of the key byte value.

Suddenly, the Doc returns with the device, with an awkward gadget strapped on top. "Here! I replaced the clock crystal with this Time Circuit, and now it's secure!", he exclaims as he puts it in your hand and wanders off again. You measure power consumption (these measurements are attached), but your previous program doesn't succeed on these.

C. Write another program which outputs the value of the attacked key byte for this newer device. The input and output format should be the same as in the previous section. (Hint: See what happens when you plot several traces on top of each other.)

The following files are attached to the exercise:

"`plaintextSlicesB`.csv" and "`plaintextSlicesC.csv`" contain the plaintexts slices for the two devices attacked in sections B and C, respectively.

"`tracesB.csv`" and "`tracesC.csv`" are the trace files for the two devices attacked in sections B and C, respectively.

"`sbox.py`" contains the S-box lookup table used in AES encryption, included for your convenience.

For these specific files, the correct key byte value is 119 for the first device (section B) – so this is what your first program should output given these files. For the "fixed" device in section C, the correct value given the sample files is 53.

Implementation notes:

- Your code should be clear and well documented.
- Implement your solution in Python (or request the teaching assistant's permission, in advance, to submit in another language).
- You may use the "numpy" library (supported in our testing environment, and packaged by all Linux distributions). You can find (unofficial) Windows installations here, and Mac/Linux here.
- The runtime of your attacks should be conveniently low (less than 30min on a modern PC). Aim for the lowest time.

AES Overview:

For your convenience, we include a high level description of AES (courtesy of Wikipedia[1]), in a representation suitable for 8-bit microcontrollers. To further understand how AES works, see also the AES Flash animation[2]. Refer also to the key schedule description[3] (hint: note that the round key in the initial round is simply the 128-bit encryption key).

Initial state – the 128-bit plaintext block.

1. KeyExpansion—round keys are derived from the cipher key using Rijndael's key schedule
2. Initial AddRoundKey: each byte of the state is combined with the round key using bitwise xor.
3. Round (repeated 9 times):
    1. SubBytes: a non-linear substitution step where each byte of the state is replaced with another according to a S-box lookup table (see attached "sbox.py" file).
    2. ShiftRows: a transposition step where each row of the state is shifted cyclically a certain number of steps.
    3. MixColumns: a mixing operation which operates on the columns of the state, combining the four bytes in each column.
    4. AddRoundKey: each byte of the state is combined with the round key using bitwise xor.
4. Final Round (no MixColumns)
    1. SubBytes
    2. ShiftRows
    3. AddRoundKey

[1] http://en.wikipedia.org/wiki/Advanced_Encryption_Standard
[2] http://www.cs.bc.edu/~straubin/cs381-05/blockciphers/rijndael_ingles2004.swf
[3] http://en.wikipedia.org/wiki/Rijndael_key_schedule

Question 2

Tom, Leroy and Kate use a voice chat software to talk to each other. Tom suspects Kate and Leroy are talking about him behind his back and decides to listen in on their conversations

A. Tom and Leroy are roommates, so they are both using the same router to connect to the internet. How can Tom find Leroy's network utilization during his conversations with Kate?

B. Tom studies the protocol used by the voice chat software. He finds that the software separately encodes (and sends) every second of sound by compressing the recorded voice and then encrypting it. The compression ratio (and the size of the sent data) depends on the sound recorded in that second. How can Tom use this knowledge to find out what Kate is telling Leroy during their voice chat conversations? If you invoke an algorithm described in class, explicitly specify all the inputs and how you would choose them, as well as the outputs and how you would use them.

Question 3

Bob has a 2048-bit RSA private key (*p*,*q*) that he uses to decrypt ciphertexts (encrypted under plain RSA) that he receives.

Bob's RSA decryption software uses the Chinese Remainder Theorem as shown in class. That algorithm requires multiplication of big integers, but Bob's computer directly supports arithmetic only on 32-bit words. Thus, the modular multiplication of big integers *A* and *B* modulo a big integer *p* is implemented by "elementary-school multiplication", using 32-bit words as the basic unit (instead of decimal digits):

```
multiply(A, B, p):
   A = A mod p
   B = B mod p
   result = 0
   for (i = 0; i < size(A); i++)
      for (j = 0; j < size(B); j++)
         result = ( result + A_i*B_j << (i*32 + j*32) ) mod p
   return result
```

where:
If X is an unsigned integer, size(X) is the number of 32-bit words in the binary representation of X, and X_i denotes the unsigned integer value of the i'th least-significant word.
<< denotes logical left shift of big integers
+ denotes addition big integers
* denotes integer multiplication, and works only on 32-bit unsigned integers

Unknown to Bob, his CPU model has a hardware bug. In the CPU hardware, there is a multiplier unit which can multiply two 32-bit unsigned integers. There exist two fixed 32-bit unsigned integers *X* and *Y* for which the multiplier returns an incorrect result which does not equal *X***Y*.

You have just discovered this bug, and found *X* and *Y*. You would like to exploit this knowledge to steal Bob's private decryption key. How can you do so? You can ask Bob to decrypt messages of your choice (as in a chosen-ciphertext attack).

Hint: Consider the case $p<C<q$.