Project

Wednesday, March 12, 2014 18:08

(The content below is very preliminary, under construction, and may be changed during the semester, possibly based on student feedback, which is encouraged via e-mail or at the appropriate Forum area.)

Recent updates:
Feb 21: allowing teams of 4 students (read below for details)
Feb 21: report should include what tasks were performed by which student (read below for details)
Mar 4: clarified Part 1,2,3 scope and attached file with details about primitives and schemes to be implemented in Part 1,2,3
Mar 6: confirming project teams
The project consists of a software implementation of a set of cryptographic primitives (i.e., block ciphers, block cipher modes of operation) and cryptographic schemes (i.e., symmetric encryption schemes, asymmetric encryption schemes, message authentication codes, signature schemes, etc.) and possibly higher-level protocols using them. The implementation has to be in C or C++ on any one of the following C++ programming environments (to be decided, but almost surely: Visual Studio Express and Eclipse, which are freely available from the Internet). You will be allowed and encouraged to use software libraries from the Internet, and will have to produce a report detailing implementation approach, software documentation and performance analysis. A project should be realized by a team of 3 students, and comes with a minimal assignment divided into three parts, each part requiring a submission of a software and report from each team and being graded by the grader and myself; any additional work you perform will be considered extra credit work. Teams of 4 students are allowed but their submissions will be judged according to proportionally higher standards than teams of 3 students.
Please confirm your project team status by emailing ASAP to the instructor the students' names, with cc to all team members. If your team has 3 or 4 members, it will be quickly confirmed. If your team has 1 or 2 members, it will be quickly merged with other teams into a team of 3 or 4 members.
Part 1 will be about symmetric encryption. You will have to implement 3 symmetric encryption schemes chosen among those in the attached file.
Part 2 will be about message authentication and improved symmetric encryption. You will have to pick one of the symmetric encryption schemes implemented in part 1 and then augment it with 3 message authentication schemes chosen among those in the attached file.
Part 3 will be about asymmetric encryption, hybrid encryption, and signature schemes. You will have to pick one of the symmetric encryption schemes implemented in part 2 and then augment it with 2 asymmetric encryption schemes (using the hybrid encryption paradigm) chosen among those in the attached file, and then augment the resulting scheme with 2 signature schemes (certifying the decryption public key) chosen among those in the attached file
For each part, your (properly commented) submitted software should allow anyone to execute the programs: KeyGeneration, Encrypt, Decrypt, capable of reading input from and/or writing output on files key.txt, encryptedplaintext.txt, ciphertext.txt, and decryptedplaintext.txt, following the appropriate syntax, and allowing processing of arbitrary-length messages; specifically, the latter three files should start with an integer on line 1 denoting the length of the plaintext/ciphertext (i.e., the number of symbols in whichever format you chose; binary or exadecimal; see below) file and then contain plaintext/ciphertext starting from line 2. Specifically, your executable files KeyGeneration, Encrypt, Decrypt, should be able to run with command line inputs as follow:

```
./keygen  > key.txt
./encrypt  key.txt  encryptedplaintext.txt  > ciphertext.tex
./decrypt  key.txt  ciphertext.txt  > decryptedplaintext.txt
```

For part 3, you will also produce KeySign and KeyVerify programs that run as follows:

```
./keysign  key.txt  validityparameters.txt  >  keycertificate.txt
./keyverify  key.txt  keycertificate.txt  validityparameters.txt  > yesno_output.txt
```

Your files can be in one of two formats: binary (i.e., all symbols from {0,1}) or exadecimal (i.e., all symbols from {0,1,...,9,A,B,...,E,F}); let us know which one you picked both in the code and in the report.

These programs should make, whenever possible, calls to software taken from libraries freely available on the internet (e.g., OpenSSL and/or others). After producing your software, your goal is to compare the performance

of the 3 or 4 primitives or schemes for varying message lengths, and discuss the relative efficiencies of the 3 or 4 cryptographic primitives or schemes within the same set.
Your accompanying **report** should at least include the following sections:

1. A brief introductory section describing the set of primitives or schemes that you have chosen, and the list of project tasks performed by each student in the team
2. A detailed explanation of how to run all programs to be able to generate keys, encrypt messages and decrypt ciphertexts
3. A detailed explanation of which software you selected from internet libraries and your choice criteria
4. Performance analysis of the 3 or 4 cryptographic primitives or schemes on different inputs, including at least one graph plotting the relative performance, with discussion of lessons learned

Your software and report in part 2 should be built incrementally to the one built in part 1.
Your software and report in part 3 should be built incrementally to the ones built in parts 1 and 2.
Preferred extensions (to be considered as **extra credit**) include the following:

1. creating a client and a server and allowing the capability to run encryption at the client and decryption at the server
2. repeating performance analysis for more than 1 key length
3. anything else you want to add

Each of your 3 submission will be judged based on the following **project grading criteria**:

1. technical correctness (i.e., if your implemented primitives or schemes, after some amount of testing, seem to satisfy decryption correctness)
2. software usability (i.e., if you followed all of the above instructions, especially on how to run the programs, and if software is easy to use)
3. primitive/scheme speeds (i.e., how fast is your implemented primitive/scheme)
4. insightful performance analysis (i.e., generality of the analysis, valid interpretation of it and insightful learned lessons).

(Tentative) **timeline and due dates** is on the syllabus.

No late submissions can be accepted and early submissions are encouraged. You are strongly recommended to submit any questions to the grader and the instructor, and to use feedback obtained with your grading on part 1 and 2 before your submission of part 3. Links to submit your team choice, and part 1,2,3 project, will appear here.

Here is the text content of the File that is mentioned:
Remember:

if possible, you should take online (and not implement from scratch) code for as many methods as you can, especially the following: DES, AES, SHA1, RSA, El Gamal, some modes of operation, etc.

Sets of cryptographic primitives / schemes:
Part 1. Symmetric encryption schemes:
(Choose 3 among the following schemes)
1. Scheme based on pseudo-random generators as in Lecture 5
(use pseudo-random generator G(s) = (F(k,0),F(k,1),F(k,2),…,) where key k is set =s and F is AES)
2. Scheme based on pseudo-random functions as in Lecture 5
(use AES in CBC mode and IV=0 as a pseudo-random function)
3. Scheme based on a block cipher (use AES) and a block cipher mode of operation (use CBC)
4. Scheme based on a block cipher (use AES) and a block cipher mode of operation (use Counter mode)
5. Scheme based on a block cipher (use Triple DES) and a block cipher mode of operation (use CBC)
6. Scheme based on a block cipher (use Triple DES) and a block cipher mode of operation (use Counter mode)

Part 2. Message authentication and improved symmetric encryption:

Pick one of the encryption schemes implemented under part 1.
Augment this scheme with all three of the following timestamping + message authentication schemes:

1. timestamping and F-CBC-MAC (use F = AES)
2. timestamping and HMAC (choose H = one among MD5, SHA1, SHA2 or SHA3)
3. timestamping and the MAC based on pseudo-random functions (use AES in CBC mode and IV=0 as a pseudo-random function)

Part 3. Asymmetric encryption, hybrid encryption, and signature schemes:

Pick one of the schemes implemented under part 2.
Augment this scheme with two of the following asymmetric encryption schemes:

1. Textbook RSA
2. Padded RSA
3. OAEP RSA
4. OAEP+ RSA
5. El Gamal
6. Cramer-Shoup lite
7. Cramer-Shoup general

and with two of the following signature schemes:
1. Hashed RSA
2. DSS

References:	Lecture	Slides (may be slightly incorrect)
Scheme based on pseudo-random generators	6	10 - 12
Scheme based on pseudo-random functions	6	14 - 16
DES	7	9 - 11
Double DES	7	12 - 13
Triple DES	7	12
AES	7	14 - 16
ECB mode	7	16
CBC mode	7	17
OFB mode	7	18
Counter mode	7	19 - 21
Timestamping	8	18 - 19
MAC based on pseudo-random functions	8	8
F-CBC-MAC	8	10
HMAC	8	15
MD5, SHA1, SHA2, SHA3	8	16
Textbook RSA	9	11
Padded RSA	9	12
OAEP RSA	9	12
OAEP+ RSA		www.shoup.net/papers/oaep.pdf
El Gamal	9	13 - 16
Cramer-Shoup lite	10	11 - 13
Cramer-Shoup general	10	11 - 12, 14
Hashed RSA	11	9
DSS	11	13