COMP 4109: Applied Cryptography
Assignment 4

March 3, 2005

Instructions:

- The answers to these questions are due on Monday, March 14th at 2:30 PM.
- Turn in paper copies **in class**. In addition, please email all program outputs to Mohammad Mannan (mannan@ccsl.carleton.ca), packaged as a zip file with the name "<*your name*>-<*your student ID*>-COMP4109-4.zip".
- You will need to refer to the PKCS #1 document, which is available through the assignment web page.
- When reporting octet values (e.g. for an RSA signature), encode the octets as pairs of lowercase hexadecimal characters, separated by a single space, with at most 24 octets per line. For example:

  00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15 16 17

- Good luck!

1. (a) Using the p, q, and e values supplied on the assignment website, generate an RSA public-private keypair.

   (b) Compute the SHA-1 hash value (in hexadecimal) of the test message on the assignment website. Be sure to treat the file as a binary input to the SHA-1 algorithm. (You may use the `sha1sum` available on most Linux systems. The `wget` command may also be useful.)

   (c) Manually format this hash value to create a PKCS #1 v1.5 encoded message EM. Refer to Section 9.2 of the the PKCS #1 specification for a precise definition of EM.

   (d) Use EM and your computed RSA public/private keypair to manually create a PKCS #1 v1.5-compliant signature. Be sure to use the appropriate octet-to-integer and integer-to-octet conversions!

   (e) Manually verify that your computed PKCS #1 v1.5 signature is correct. Follow the specified verification algorithm. Show your work. (You may skip the ASN.1 parsing step, however, and instead simply check for the hex values given in the document.)

(f) In a similar manner, create and verify a PSS signature using the assigment key and test document. Use "comp4109" as your salt value.

(g) Will the PKCS #1 v1.5 signature you create be identical to that created by your classmates? What about the PSS signature? Would they be identical if you and your classmates used a real application (e.g. GnuPG) to generate these signatures? Explain.

2. (a) Which inputs to the RSA key generation algorithm (Alg. 8.1) may be shared with others? Which ones must be kept secret?

   (b) Which inputs to the ElGamal key generation algorithm (Alg. 8.17) may be shared with others? Which ones must be kept secret?

3. You are building a secure messaging application similar in design to PGP that uses RSA (1024 bit public keys) to perform public key encryption and signatures and AES-128 for bulk encryption. You will use the Miller-Rabin probabilistic primality test (Algorithm 4.24) to determine whether randomly-generated numbers are prime.

   (a) Why might you want to use the same RSA key for signature and encryption operations? Why might you want to use different RSA keys for signature and encryption operations?

   (b) Let $t = 15$ for Miller-Rabin. If Miller-Rabin returns "prime" for a number x, what is the probability P(x is composite)?

   (c) What is approximately the maximum t value you could use for the Miller-Rabin when testing your candidate p and q values for RSA key generation, if you assume that you never choose a given a more than once? Base your value of t on the approximate magnitude of p and q, given that n is to be approximately 1024 bits in size.

   (d) You wish to ensure that the probability P(x is composite | Miller-Rabin(x)=prime) is no more than the probability of guessing a correct AES-256 key. What value of t should you use for Miller-Rabin?

   (e) You change your mind and decide that wish to be 99.999% sure that the prime candidates reported by Miller-Rabin are actually prime. What t value should you use?

4. What is the difference between the guarantees provided by unconditional security, provable security, computational security, and ad-hoc security? Identify what type of security each of the following cryptographic primitives provides: one-time pads, DES, AES, SHA-1, SHA-256, RSA, DSA. Explain your classifications.