COMP 4109: Applied Cryptography
Assignment 4 Solutions

March 3, 2005

1. *(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.*

   The signature you generate will be identical to that created by your other classmates for both PKCS #1 v.1.5 and PSS. In a real application, PKCS #1 v.1.5 signatures would also be identical; because the program would choose a random salt value (instead of "comp4109"), though, each PSS signature would be unique, even when an identical key and document are used.

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

   RSA key generation takes three inputs: two primes p and q, and an encryption exponent e. The input e (the encryption exponent) may be shared with other key pairs, and in fact it may be set to a system-wide value. p and q must not be shared with anyone and in fact should be discarded after key generation. n (the product of p and q) may be made public; however, it not be used to generate other key pairs, even if a different e and d value is used.

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

   ElGamal key generation takes three inputs: p, α, and a (also called x). p and α may be shared, and in fact may be used to generate multiple key sets. Every key pair, however, should use its own a value that is kept secret (and thus is not shared).

3. (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?*

   You might wish to use the same RSA key for signatures and encryption operations if storage space is at a premium, or if it is significantly more difficult to manage two key pairs rather than one.

   On the other hand, the biggest advantage to using separate keys for encryption and signatures is that it enables different key management strategies for the two roles. In particular, by changing encryption keys frequently (say, once a month), you can ensure that the compromise of an encryption key (whether voluntary or not) will have limited impact. On the other hand, to maintain

continuity of identity (and to facilitate the authentication of encryption keys), it is advantageous to have much longer lived signature keys.

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.*

   Unconditional security means that the attacker does not have enough information with which to compromise security. **Provable security** means that it can be proved that the difficulty of breaking the system is equivalent to a known-hard problem (typically, a hard number-theoretic problem). **Computational security** means that the work required to break a system significantly exceeds the resources of a hypothetical opponent using conceivable methods. **Ad-hoc security** means that the system can be argued to be secure, but there are no rigorous estimates of the amount of work that would be required by an adversary other than it exceeding a fixed bound.

   One-time pads have unconditional security in the form of perfect secrecy because an attacker with the ciphertext has no more information (in an information-theoretic sense) than one without the ciphertext. (Here we ignore the fact that the attacker does, in fact, know that a message was sent and that it has a known size.)

   RSA offers provable security since a proof exists that its difficulty is equivalent to that of factoring, a problem that is generally believed to be hard (Fact 8.6).

   The security of DSA (as an ElGamal signature variant) is based on the discrete logarithm problem; however, to my knowledge it has not been proven to be equivalent to the discrete logarithm problem. Thus, it is probably best characterized as having computational security, because the discrete logarithm problem takes a long time to solve (given suitable values) and there are no other known methods for attacking DSA.

   DES offers computational security if 2^{56} operations is beyond the reach of an attacker; given that such computations can be done nowadays, though, it does not offer any security in a cryptographic sense. (2 key triple DES, however, does still offer computational security).

   AES, SHA-1, and SHA-256 all offer computational security because the work required to break them still exceeds that possible through known methods. Given advances in attack technology, though, this situation could change.

   (Ad-hoc security typically is applied to protocols and not cryptographic primitives.)