THIS IS NOT AN OPEN-BOOK EXAMINATION
– CANDIDATES MAY NOT CONSULT ANY
REFERENCE MATERIAL DURING THE
SITTING

Calculators may be used in this examination
provided they are not capable of being used to
store alphabetical information other than
hexadecimal numbers.

THE UNIVERSITY OF BIRMINGHAM

Degree of MEng with Honours
Computer Science/Software Engineering, Final Examination

Undergraduate Occasional
Computer Science/Software Engineering
Economics

06 17417

Computer Security

Tuesday 1st June 2004  1400 hrs – 1600 hrs

[Answer ALL Questions]
1. Symmetric-key encryption.

Alice and Bob wish to communicate using symmetric key encryption. They intend to use a substitution cipher: the secret key is a permutation of bytes. (For example, the key may be represented as an array [0...255] each cell of which is a number in the range 0—255, without duplicates.)

(a) What attacks are possible on this scheme, under each of the following assumptions:
   (i) The attacker knows the plain text is an HTML page; [4%]
   (ii) The attacker knows the plain text is a compressed page of ascii (the compression algorithm is not known). [4%]

(b) How many possible keys are there? (Your answer can be an expression, if you are not able to evaluate it as a number.) [4%]

(c) How does this encryption method compare with DES, in terms of
   (i) a cryptanalytic attack [4%]
   (ii) a brute-force attack? [4%]

2. Secure hash functions.

(a) What are the important properties of secure hash functions such as MD5? [4%]

(b) Explain in what way MD5 is thought to be weak. [4%]

(c) A software company uses the following method to distribute software to its clients.
   • It puts the software on an insecure web server.
   • The client downloads the software, and computes an MD5 hash of the downloaded file. Then they telephone the software company, and read out the first four bytes of the hash (which consists of 16 bytes).
   • If the four bytes verify, the client accepts the software as securely transmitted.

   The software company approaches you and asks your opinion on whether they should cease using MD5 in this way, in view of its weakness.

   (i) Advise the company. [4%]
   (ii) Would your advice be different if they verified all 16 bytes of the MD5 hash? [4%]
3. Public-key cryptography.

(a) Fred has implemented RSA. Part of his code is shown at the end of this examination paper. A company has approached you to ask your opinion of whether they should accept Fred's implementation. Advise them. 

(b) Alice and Bob wish to issue a lengthy (10 MB) joint statement to newspaper N. N has asked for the statement to be signed by each of A and B. They also want to prevent eavesdropper E from having prior access to the statement. Alice proposes the following protocol:

Alice writes the statement, encrypts it with Bob's public key, and sends it to him. If he accepts her version of the statement, he signs a secure hash $h$ of it with his private key, resulting in the message $m = \{h\}_{KB^{-1}}$. He then encrypts his signature $m$ with Alice's public key, resulting in $\{m\}_{KA^{-1}}$ and sends it to her. Alice decrypts Bob's message to recover $m$, and signs it with her private key. The result is $\{m\}_{KA^{-1}}$, i.e. $\{\{h\}_{KB^{-1}}\}_{KA^{-1}}$. She puts the statement together with the doubly-signed hash, encrypts them with N's public key, and sends it to N.

Assume that all public keys are reliably known, and that the cryptography is secure.

(i) What cryptographic computations is N supposed to perform, in order to verify the signatures? 

(ii) Comment on the proposed protocol.

(1) Is the protocol secure for A? For B? For N? 

(2) Can you optimise it without weakening its security?

A company called Mocrosift wishes to check that its customers are using its web browser “Internet Deplorer” (ID), and not the rival product “Melizza”. It consults two security companies to suggest an appropriate method. The companies respond as follows.

- **Protocol A.** When ID access a web page from Mocrosift's server, the server responds with a challenge consisting of Mocrosift's normal public key. ID replies with a message consisting of
  - a digital hash #A of the ID executable,
  - a secret, which is embedded in ID's executable in a way resistant to tamper or extraction.

  The message is encrypted with Mocrosift's public key before it is sent off. Mocrosift checks that the digital hash #A is the correct hash for ID, and that the secret is correct. It sends the web page.

- **Protocol B.** When ID access a web page from Mocrosift's sever, the server responds with a challenge consisting of a nonce. ID replies with a message consisting of
  - the nonce, signed with a secret key which is embedded in ID's executable in a way resistant to tamper or extraction.

  Mocrosift verifies that the nonce is the one it sent. It sends the web page.

(a) What attacks are available to Melizza on each of the proposals? [4%]

(b) Advise Mocrosift on whether it should accept either of them. [4%]

(c) How can Trusted Computing enable the deployment of a protocol, which is more secure than either of the proposals above? [4%]

5. Short notes.

Write short notes (maximum 100 words) on each of the following:

(a) The meaning of *assets, threats, attacks, vulnerabilities, risk, and impact.* [6%]

(b) How can companies and legislators prevent phishing? [6%]

(c) The desired properties of a digital cash system. [6%]

(d) Zero-knowledge protocols. [6%]

(e) Problems with logins by fingerprint recognition. [6%]

(f) Fair exchange protocols. [6%]
Fragment of Java code for question on public key encryption:

class Encrypter {

    private Key publicKey;
    private Key privateKey;

    public Encrypter () {
        // generate keys

        final BigInteger p = new BigInteger("65669205018189751363824155419918192392295592176092883676630\4161790553989228223793461834703506872747071705167995972707253\94009946986951642289363357693");

        final BigInteger q = new BigInteger("55026006850391381679477575674873495721752555710127735261793\9473192429642846467658900405045171852174534967724674742228789\190327015236678532191013890529");

        final BigInteger n = p.multiply(q);
        final BigInteger e = new BigInteger("7");
        final BigInteger d = findInverse(p, q, e);
        // finds d such that \( e \times d = 1 \mod (p-1)(q-1) \)

        publicKey = new Key(e, n);
        privateKey = new Key(d, n);
    }

    public encrypt (Msgblock m) {
        // returns m transformed with publicKey
    }

    public decrypt (Msgblock m) {
        // returns m transformed with privateKey
    }
}