Assignment 1
Computer and Network Security

Matt Barrie
matt.barrie@sydney.edu.au
Department of Electrical Engineering
University of Sydney

May 23, 2011

Due: Beginning of class, Friday June 3rd
You are to work on this assignment in groups of 1 or 2. These groups do not have to be the same as those for you project. This assignment is to be handed to the tutors at the beginning of class, with your answers typed (not handwritten). Late assignments will not be accepted. Make sure you understand each of your answers well - you may be asked to explain your answer in class. Please make sure the names and SIDs of each person in the group are provided.

1 DES (25 marks)

Before DESX was invented, the researchers at RSA Labs came up with DESV and DESW, defined by

$$\begin{aligned} DESV_{kk_1}(M) &= DES_k(M) \oplus k_1 \\ DESW_{kk_1}(M) &= DES_k(M \oplus k_1) \end{aligned}$$

Z

As with DESX, $\|k\| = 56$ bits and $\|k_1\| = 64$ bits. Show that both these proposals do not increase the work needed to break the cryptosystem using brute-force key search. That is, show how to break these schemes using on the order of 2^{56} DES encryptions/decryptions. You may assume that you have a moderate number of plaintext-ciphertext pairs.

2 Hash Functions (25 marks)

2.1 Cyphers using Hash Functions (5 marks)

Show with reference to a diagram how a block cypher can be created using a hash function given that a hash function is one-way and a cypher must be reversible.

2.2 Cyphers as Hash Functions (10 marks)

Let us assume that we create a keyed hash function from DES by breaking the message m into 64-bit blocks $m_1, m_2, \ldots m_n$ and using each m_i as they key in repeatedly encrypting an initial constant IV until we result in the final 64-bit digest output d (ie. $d = E_{m_n}(\ldots E_{m_2}(E_{m_1}(\text{IV})))$). Remembering the birthday paradox, how many messages would you have to try before you found two such that $d = d'$?

2.3 Cyphers as Hash Functions 2 (10 marks)

Now how many iterations would be required to find a digest $d' = d$ for a given value of d (Hint: consider your answer to 'Multiple Encryptions' above)

3 Protocols (25 marks)

Consider the following protocol for performing authentication.

Set Up.

1. Alice chooses random primes p and q and computes $n = pq$.
2. Alice then chooses random $s < n$ and computes $v = s^2 \ mod \ n$.
3. Alice publishes $\{n, v\}$, and keeps s secret. She can discard p and q.

Authentication.

1. Alice chooses k random numbers $r_1, r_2, \ldots, r_k$. For each r_i she sends $r_i^2 \ mod \ n$ to Bob.
2. Bob chooses a random subset of the r_i^2 and tells Alice which subset he has selected to be known as subset 1. The others will be known as subset 2.
3. Alice sends $sr_i \ mod \ n$ for each r_i^2 of subset 1, and sends $r_i \ mod \ n$ for each r_i^2 of subset 2.

4. Bob squares Alices's replies *mod* n. For those r_i^2 in subset 1 he checks that the square of the reply is vr_i^2 *mod* n. For those r_i^2 in subset 2 he checks that the square of the reply is r_i^2 *mod* n.

Why does this authenticate Alice? In your answer be sure to include reasons why someone cannot impersonate Alice. In particular, what is the purpose of subsets 1 and 2?

4 Question 4: Protocol Design (25 marks)

Commitment schemes allow Alice to commit a value x to Bob. The scheme is *secure* if the commitment does not reveal any information about the committed value to Bob. At a later time, Alice can *open* the commitment and convince Bob that the committed value is x. The commitment is *binding* if Alice cannot convince Bob that some other value $x' \neq x$ is the committed value.

Consider the following commitment scheme:

Commitment: Alice chooses a random r the same size as x and calculates $y = x \oplus r$. She sends the value y to Bob.

Open: Alice sends r to Bob, and Bob calculates $x = y \oplus r$.

A. Show that the proposed scheme is secure.

B. Show that the proposed scheme is **not** binding. That is, how can Alice defeat the protocol to reveal a fake committed value.

C. Suggest a modification to the scheme that will allow Bob to detect cheating. Remember that *freshness* is an issue with proving the scheme is binding, i.e. some guarantee that Alice has not found a value ahead of time which may opened two different ways to reveal two different committment values.