CS 165 Lab 6 Assignment

November 4, 2012

Shamir's Secret Sharing

1. Use Shamir's Secret Sharing scheme to share the secret $s = 7$, producing $n = 5$ shares such that any $k = 2$ shares are sufficient to recover the secret. You need to start by producing a degree $k - 1$ polynomial, setting one of the coefficients to s (usually a_0), and generating random values for the remaining coefficients.

2. Recover the secret s using exactly two of the five shares you generated above.

Birthday Paradox

3. Assume we have an ideal hash function h that hashes to one of $n = 2^b$ values, for some chosen $b \in \mathbb{Z}$. What is the smallest value of b that allows us to hash an arbitrary set M of 6 messages ($|M| = 6$) with a collision probability of ≤ 0.10?

OpenSSL

From the command line, do the following:

1. Create cleartext private and public 1024 bit RSA keys. The outputs are base 64 encoded and contain information about the encryption scheme details used, separate keys for signing, etc. The private key file contains the public key as well:

   ```
   openssl genrsa -out rsaprivatekey.pem 1024
   openssl rsa -in rsaprivatekey.pem -pubout -out rsapublickey.pem
   ```

2. Create a new plaintext file called Yourlastname.txt, and put the contents of some computer security article in it, with your name at the top. This will be the file you will send from server to client in your project.

3. Test the RSA keys by signing and verifying a hash of the file.

   (a) Produce ASCII hexadecimal hash/message digest using SHA1 (an old *secure hash algorithm*):

   ```
   openssl sha1 -out hash.txt Yourlastname.txt
   ```

   (b) Remove the prefix text "SHA1(Yourlastname.txt)= " (including the space) from hash.txt.

   (c) Convert the hash to binary before signing:

   ```
   xxd -r -ps hash.txt > hash.bin
   ```

   (d) Sign the hash:

   ```
   openssl rsautl -sign -inkey rsaprivatekey.pem -in hash.bin
                                                 -out hash-signature.bin
   ```

   (e) Verify the signature:

   ```
   openssl rsautl -verify -pubin -inkey rsapublickey.pem
                          -in hash-signature.bin -out hash-verify.bin
   ```

   (f) Convert the signature verification output back from binary to base 64 / plain text. Verify that the original hash and the signature verification output are the same. If so, you can assume that your RSA keys are working properly.

   ```
   xxd -ps hash-verify.bin > hash-verify.txt
   diff hash.txt hash-verify.txt
   ```