Assignment: Due Wednesday, 11:59pm

Chris Betsill

October 19, 2016

1. How many bits change?

Use the `openssl` package to create hashes of the following two strings:

```
library(openssl)
m1 <- "Oh, say, can you see,
By the dawn's early light,
What so proudly we hailed
At the twilight's last gleaming?
Whose broad stripes and bright stars,
Through the perilous fight,
O'er the ramparts we watched,
Were so gallantly streaming?"
m2 <- "Oh, say, can you see,
By the dawn's early light,
What so proudly we hailed
At the twilight's last gleaming?
Whose broad stripes and bright stars,
Through the perilous night,
O'er the ramparts we watched,
Were so gallantly streaming?"

lyrics <- c(m1, m2)

compare <- function(list){
  methods <- c(md5(list), sha1(list),sha256(list))
  raw <- sapply(methods, charToRaw)
  bits <- sapply(raw, rawToBits)
  for(i in seq(1, length(bits), 2)){
    len <- length(bits[[i]])
    cat(100*(len-sum(bits[[i]]==bits[[i+1]]))/len,"%, ")
  }

}
compare(lyrics)
```

```
33.98438 %, 28.75 %, 30.27344 %,
```

Use MD5, SHA-1, and SHA-256. Express each pair of hashes in bits, and for each pair, determine how many bits are different (i.e., how many binary digits of the hash of `m1` are different from the corresponding digit of the hash of `m2`.) Express your answers as percentages of the length of the hash. Describe the property of hash functions that your calculations illustrate.

This illustrates the hash functions propertiy of diffusion, showing how a single character difference results in hashes that are around 30% different.

2. Dictionary attack, with and without salt

Use the dictionary of common passwords below to determine what passwords the following two hashes came from. The first is a plain `sha256` hash; the second is an `sha256` hash with a 16-bit salt (i.e., a `raw` vector of length 2) used as an HMAC key. Show your work. See the `openssl` package documentation: the parameter `key` is how the salt was added.

```
# NOTE: If you don't want to execute this code block every time you knit (e.g., while you are
# working on other problems) you can replace "{r}" with "{r, eval=FALSE}" at the top of this
# code block. However, make sure you evaluate it when you knit your final PDF to hand in.
unsaltedHash <- "059a00192592d5444bc0caad7203f98b506332e2cf7abb35d684ea9bf7c18f08"
saltedHash <- "cc94120c42e2c4b64fb10464f565b57039886771d5dd06a3f79b34d440c36eaf"
dictionary <- c("123456", "password", "12345678", "qwerty", "dragon", "baseball", "abc123",
                "football", "monkey", "letmein", "shadow", "master", "qwertyuiop", "mustang",
                "michael", "654321", "superman", "1qaz2wsx", "7777777")
unsalted <- function(dict, hash){
  for(pass in dict){
    if(sha256(pass) == hash){
      print(pass)
      break
    }
  }
}

salted <- function(dict, hash){
  for(i in 0:255){
    keyone <-  as.raw(i)
    for(j in 0:255){
      keytwo <- as.raw(j)
      salt <- c(keyone,keytwo)
      for(pass in dict){
        if(sha256(pass, salt) == hash){
          print(pass)
          break
        }
      }
    }
  }
}
system.time(unsalted(dictionary, unsaltedHash))[3]
```

```
[1] "1qaz2wsx"

elapsed
      0
```

```
system.time(salted(dictionary, saltedHash))[3]
```

```
[1] "letmein"

elapsed
 45.559
```

How does adding a 16-bit salt affect the running time of a dictionary attack? (Either calculate or measure to find your answer.)