Problem Set 1

Andrew Kaufman
SID: 998048873

April 6, 2016

PROBLEM 1

When a user *i* uses a password *K* in some cryptographic protocol, system flows will usually reveal a pair *(X, Y)* where *Y = F(K, X, i)*. Here F is some known, fixed function associated to the protocol. Consider a user *i*'s password to be *cracked* if an adversary, given *(X, Y)*, can find some key *K* for which *Y = F(K, X, i)*.

Attacker Ned has built a large password-cracking machine. Ned used a budget of $100 million for this project, spending half the money on custom chips (ASICs) that, given (X, Y, i), test, for a candidate *K*, if *Y = F(K, X, i)*. (The other half of Ned's budget was used for circuit boards, assembly, power, and cooling). The ASICs cost Ned $10 each and, for some particular *F*, each chip can test 10^9 different passwords per second.

(a) Hal, Leo, and Pat have passwords that are reasonably modelled as uniformly random strings of 8, 12, and 16 characters, respectively, each character being a lowercase letter (a–z). Attacker Ned gets hold of an *(X, Y)* pair for each user *i*. How long will it take Ned to crack each user's password?

$$\$50 \text{ million spent on ASIC chips} \rightarrow \frac{\$50 \cdot 10^6}{\$10} = 5{\cdot}10^6 \text{ chips}$$

$$(5{\cdot}10^6 \text{ chips})(1{\cdot}10^9 \ \frac{passwords}{second}) = 5{\cdot}10^{15} \ \frac{passwords}{sec}$$

Hal's password is 8 characters:

lowercase (a–z) = 26 possibilities for each character

$\rightarrow 26^8$ password possibilities

$$\frac{26^8}{5 \cdot 10^{15}} = .000041765 \text{ seconds}$$

Lee's password is 12 characters:

lowercase (a−z) = 26 possibilities for each character

$\rightarrow 26^{12}$ password possibilities

$$\frac{26^{12}}{5 \cdot 10^{15}} = 19.085791332 \text{ seconds}$$

Pat's password is 16 characters:

lowercase (a−z) = 26 possibilities for each character

$\rightarrow 26^{16}$ password possibilities

$$\frac{26^{16}}{5 \cdot 10^{15}} = 8.72 \cdot 10^6 \text{ seconds}$$

(b) A system administrator comes in and demands that users select passwords that include one uppercase letter (A−Z), one lowercase letter (a−z), and one digit (0−9). Hal, Leo, and Pat change their passwords so that they are now well modelled as having one uppercase letter, one digit, and the rest lowercase letters (same lengths as before). How effective was this change in stopping Ned? (Recompute the attack times.)

Hal's password:

Uppercase (A−Z): 26 possibilities · 8 locations

Digit (0-9): 10 possibilities · 7 locations

Lowercase (a−z): 26^6 possibilities · 6! arrangements

$$\frac{(26 \cdot 8) \cdot (10 \cdot 7) \cdot (26^6 \cdot 6!)}{5 \cdot 10^{15}} = 2.22 \cdot 10^{11} \text{ passwords}$$

$$\frac{2.22 \cdot 10^{11}}{5 \cdot 10^{15}} = 4.44 \cdot 10^{-5} \text{ seconds}$$

Lee's password:

Uppercase (A−Z): 26 possibilities · 12 locations

Digit (0-9): 10 possibilities · 11 locations

Lowercase (a−z): 26^{10} possibilities · 10! arrangements

$\frac{(26 \cdot 12) \cdot (10 \cdot 11) \cdot (26^{10} \cdot 10!)}{5 \cdot 10^{15}} = 5.12 \cdot 10^{20}$ passwords

$\frac{5.12 \cdot 10^{20}}{5 \cdot 10^{15}}$ = 102453.4313 seconds

Pat's password:

Uppercase (A−Z): 26 possibilities · 16 locations

Digit (0-9): 10 possibilities · 15 locations

Lowercase (a−z): 26^{14} possibilities · 14! arrangements

$\frac{(26 \cdot 16) \cdot (10 \cdot 15) \cdot (26^{14} \cdot 14!)}{5 \cdot 10^{15}} = 5.62 \cdot 10^{30}$ passwords

$\frac{5.62 \cdot 10^{30}}{5 \cdot 10^{15}} = 1.12^{15}$ seconds

These changes were more effective in stopping Neds as seen by the increase in times to crack the passwords. For the passwords of shorter lengths, such as Neds password, we do not see as much of an increase in difficulty as the longer passwords.

(c) A cryptographer comes in and redesigns the function *F* such that $10 ASICs will now need 100 msec to compute it. Hal, Leo, and Pat use their original passwords. How effective is this change in stopping Ned? (Recompute the attack times.)

100 msec computation time = 0.100 seconds per password

$\rightarrow$ computes 10 passwords per second

$(5 \cdot 10^6$ chips$)(100 \frac{passwords}{second}) = 5 \cdot 10^7 \frac{passwords}{second}$

Hal's password is 8 characters:

lowercase (a−z) = 26 possibilities for each character

$\rightarrow 26^8$ password possibilities

$\frac{26^8}{5 \cdot 10^7}$ = 4176.541292 seconds

Lee's password is 12 characters:

lowercase (a−z) = 26 possibilities for each character

$\rightarrow 26^{12}$ password possibilities

$\frac{26^{12}}{5 \cdot 10^7}$ = $1.90 \cdot 10^9$ seconds

Pat's password is 16 characters:

lowercase (a−z) = 26 possibilities for each character

$\rightarrow 26^{16}$ password possibilities

$\frac{26^{16}}{5 \cdot 10^7}$ = $8.72 \cdot 10^{14}$ seconds

This change is effective in stopping Ned since the ASICs are not able to test as many passwords per second. By testing fewer passwords per second, the time needed for Ned to crack the password takes significantly longer.

(d) Another cryptographers comes in and modifies the design so that, for every user, the key *K* wonâĂŹt be a password but a 128-bit random string. How effective is this change in stopping Ned? (Recompute the attack time.)

single chip computes $\rightarrow 10^9 \ \frac{passwords}{second}$

($5 \cdot 10^6$ chips)($10^9 \ \frac{passwords}{second}$) = $5 \cdot 10^{15} \ \frac{passwords}{second}$

2^{128} possible passwords

$\frac{2^{128}}{5 \cdot 10^{15}}$ = $6.80 \cdot 10^{22}$ seconds

This is effective in stopping Ned once again because although the rate at which passwords are tested is fast, the sample space is so large that a significant amount of time is needed for computation. This time needed for computation proved to be much larger than the previous designs.

PROBLEM 2

Alice has a pretty penny. Unfortunately, it may not be "fair" penny: it might, when flipped, land heads with some probability $p \neq 0.5$. Alice wants to generate a uniform random bit b: the bit should be 1 with probability 0.5 and zero with probability 0.5. Describe a strategy Alice can use to achieve this result using her possibly-biased coin.

PROBLEM 3

Alice and Bob have an infinite pile of pennies. They take turns placing their pennies on a perfectly round table, beginning with Alice. A penny may be placed anywhere on the table so long as all of the penny fits fully on top of the table and no part of the penny is on top of any other penny. Pennies must be placed flat on their heads or tails side. A party loses if he has nowhere to put his penny. Show that Alice can always win.

Alice can always win when she plays first against Bob. The approach she would need to take in order to win is to vary her placement of pennies depending on the available space left on the table. Since pennies cannot overlap one another, the placement and distance between pennies is crucial because once there is no longer enough space for a single penny, the following player loses. To accomplish this, Alice would begin by placing pennies as close to each other as possible and take note of the remaining space on the table following each of Bob's turns. On each of her turns, if there is enough room for ≥ 3 pennies, she will continue to place her pennies close enough to each other to maximize the remaining space. However, if there is room for ≤ 2 pennies, she will place her penny in the space such that it takes up enough room to where Bob has nowhere to place his penny. Since Bob cannot place his penny, Alice wins.