

# Cryptographic Applications of Random Variables

Austin Chase Minor

November 25, 2016

## 1 Introduction

You are a budding cryptanalyst and have noticed a security flaw in your companies login system. You find that matching certain keys to values gives you access to peoples' accounts. You have no way of determining which key/value pairs match. Also, you only have so many retry counts. Your job is to determine the number of key/value matches you will get given a certain retry count. This is a mathematical hard problem to determine the exact probability of success. This is due to the fact that as a key/value is matched. The number of keys decrease, and thus the probability increases for a match. So remembering the theory of random variables, you decide code the problem statement and simulate to find the average number of matches for a given retry count.

We will start with the mathematical description of the problem. From there, we will give the problem statement. After running the test, we will analyze the problem and give a conclusion.

## 2 Mathematical Description

Let  $A, B$  be a set of keys and values respectively.

Let  $f : A \rightarrow B$  be the function relating keys to values that you are trying to discover.

Let  $\phi : A \times B \rightarrow \{0, 1\}$  be a truth function representing *true* = 1 if  $f(a) = b; a \in A, b \in B$  and *false* = 0 otherwise.

Let  $k \in \mathbb{N}$  represent the number of retry counts allowed (the number of tries for each individual  $b \in B$  to the  $\phi$  function).

## 3 Problem Statement

Through some thought, it can be shown that the best way to go about trying potential values is to try each  $b \in B$  with every  $a \in A$   $k$  times removing  $a$ 's as they are matched according to the  $\phi$  function. This guarantees a minimum of  $k$  matches. Furthermore, it maximizes the shared information between  $a$ 's

resulting in a higher probability. This is intuitively seen by noticing that every time an  $b$  is tried against a  $a$  the probability of not succeeding decreases.

By repeating this trial over  $k$  several times, we can generate a sample of the actual probability. This allows us to estimate the mean and standard deviation along with seeing the general shape of the function. The Matlab code for this along with the corresponding graphs are below.

```

1 PRECISION = 100;
2 RETRY_COUNT = 10000;
3
4 freq = zeros(PRECISION, PRECISION);
5 match_num = 0;
6
7 %Graph for each k as
8 for j = 1:PRECISION
9     for i = 1:RETRY_COUNT
10        match_num = match(PRECISION,j);
11        freq(j, match_num) = freq(j, match_num) + 1;
12    end
13 end
14
15 %variation and standard deviation
16 %removes nonzero values of freq for stddev and var
17 temp = zeros(PRECISION, RETRY_COUNT);
18 for i = 1:PRECISION
19     for j = 1:PRECISION
20         if freq(i,j) != 0
21             %t is [prev array, array of size number of matches with vals of match number]
22             t = [temp(i,temp(i,:) != 0) ones(1,freq(i,j))*j];
23             %pads array with zeros to help with previous assignment
24             temp(i,:) = padarray(t, [0 (RETRY_COUNT - size(t,2))], 'post');
25         end
26     end
27 end
28
29 variance = var(temp');
30 std_deviation = std(temp');
31 mean_val = mean(temp');
32
33 figure
34 surf(freq);
35 figure
36 plot(1:PRECISION, variance, 1:PRECISION, std_deviation);
37 figure
38 plot(1:PRECISION, mean_val);

```

Listing 1: CryptoSim Main

```

1  function successes = match(sz, k)
2      A = B = 1:sz;
3      A = randomize_array(A);
4      count = 0;
5
6      for i = 1:sz
7          for j = 1:k
8              if A(i) == B(j)
9                  B(j) = [];
10                 count = count + 1;
11                 break;
12             end
13         end
14     end
15     successes = count;
16 end

```

Listing 2: Match

```

1  function B = randomize_array(A)
2      sz = size(A);
3      sz = sz(2);
4      for i = 1:sz
5          r = floor(sz * rand(1)) + 1;
6          temp = A(i);
7          A(i) = A(r);
8          A(r) = temp;
9      end
10     B = A;
11 end

```

Listing 3: Randomize Array

## 4 Analysis of Problem

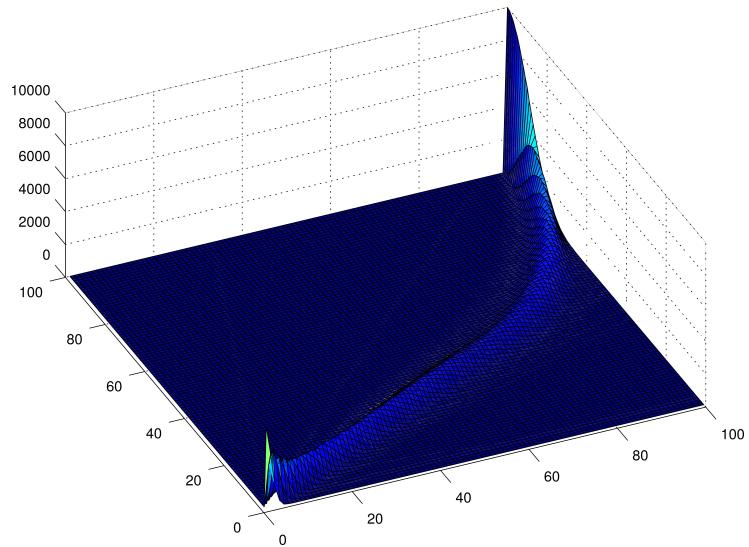


Figure 1: 3D Image of the match frequencies and numbers

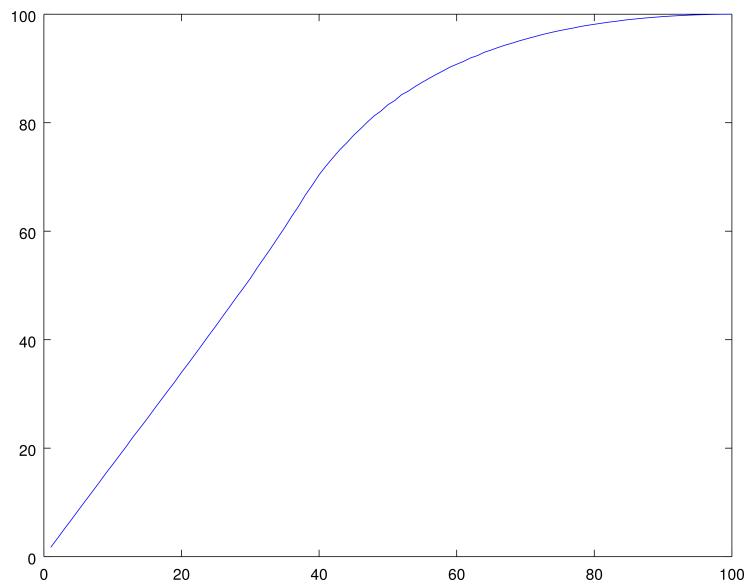


Figure 2: Average mathches over  $k$  – matches - y-axis;  $k$  - x-axis

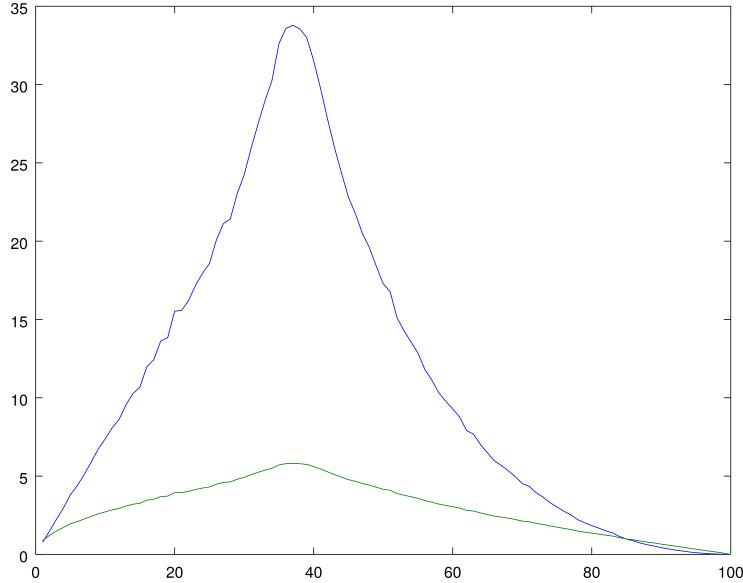


Figure 3: Standard Deviation and Variance over  $k$  – Standard Deviation - green; Variance - blue

It is logical to assume that at a certain point  $k$  there would be a tipping point. I.E. there would be a point where the number of matches being found would reduce the number left to find in a vicious cycle resulting in complete matches at a lower number. However, Figure 2 shows that this indeed is not what happens. Instead the graph increases linearly then levels off. This is surprising and means that in theory this problem is not as insecure as it may seem at first glance. Other than that, the mean shows that the number of matches to  $k$  is between (1:1 and 2:1). We have no theory for why the average matches level out instead of jumping to 100%. Further research needs to be done to determine a mathematical model of the problem to see why this is the case since it is not expected intuitively.

Overall, the matches you receive are fairly consistent. This is shown by the standard deviation in Figure 3. At max, the standard deviation is 5-6. This occurs around 38. We theorize that this is due to the leveling off of the graph in Figure 2 for high  $k$ . We theorize that for low  $k$  this is due to the lack of matches contributing to more matches, this phenomenon being described above. Thus according to Figure 3, an attacker would be able to consistently break the system at the rate given by Figure 2 (the mean).

## 5 Conclusion

We discovered that this problem is far from being as cryptographically insecure as at first glance. The growth rate does not hit a tipping point. Rather, it

actually levels off instead. Further investigation is needed to understand why this happens. This phenomenon moves the problem down from a high security problem to a medium security problem since their is not a tipping point for gaining 100% matches.