Methods of Cryptanalysis
Lecture 9

Lecture Notes for a course given by Alex Biryukov
at the Weizmann Institute of Science, Israel.
Notes taken by ....

May 27, 2000

Lecture 9

Properties of Random Functions. Hellman's Attack.

9.1 Birthday Paradox

How many people should gather in a classroom so that at least two of them have the same birthday? It turns out that with 23 people in a room, there is more than 50 percent chance that this will happen.

A simple calculation explains this apparent 'paradox'. We enumerate the people in the room from 1 to k, and denote by d_i the day of birth (a number from 1 to n, n being equal to 365) of person i. Assuming the days of birth to be independent, the probability that two persons i and j have the same birthday is equal to

$$\sum_{d=1}^{n} P(d_i = d) \cdot P(d_j = d) = n \cdot \frac{1}{n^2} = \frac{1}{n}.$$

So the probability that i and j have the same birthday is the same as the probability that one of them has his birthday in a fixed day of the year (this is a consequence of the independence assumption). Now we look for the probability that at least two out of the k people share the same day of birth. The probability that all k people have distinct birthdays is given by

$$P_{distinct\ birthdays} = 1 \cdot \Big(1 - \frac{1}{n}\Big) \cdot \Big(1 - \frac{2}{n}\Big) \cdots \Big(1 - \frac{k-1}{n}\Big).$$

In fact with two persons – say 1 and 2 – it happens with probability $\frac{1}{n}$ that 2 has the same day of birth as 1. So $1 - \frac{1}{n}$ is the probability that 1 and 2 don't share the same day of birth. Once it is known that 1 and 2 don't have the same day of birth, considering 1,2 and 3 there is a probability $1 - \frac{2}{n}$ that 3 doesn't share the same day of birth with 1 and 2, and so on, until we reach k for which there is a probability $1 - \frac{k-1}{n}$ of not having the same day of birth as $1, 2, \ldots, k-1$.

We would like to see when the probability that all k people have distinct birthdays is less than a half, so that at least two persons have the same birthday with probability greater or equal to a half. Using the inequality $1 + n \leq e^n$, we get

$$P_{distinct\ birthdays} \leq e^{-1/n} e^{-2/n} \ldots e^{-(k-1)/n} \;=\; e^{-\frac{k(k-1)}{n}}.$$

Demanding this probability to be less than a half, we get:

$$k \;\geq\; \frac{1}{2} + \sqrt{\frac{1}{4} + 2n\ log 2}.$$

If n is large, we may write

$$k \approx \sqrt{2n \log 2}.$$

9.2 Random Functions and Permutations

The above so-called Birthday Paradox is an example of "occupancy problems". It can be equivalently stated in terms of throwing k balls into n bins. Such problems can be formalized in terms of random functions from a finite set set of size k to a finite set of size n. In particular, we are interested in the case of random functions of a finite set into itself.

Definition 9.2.1. *Let us denote by F_n the set of all functions from the set $\{1, .., n\}$ to itself. Assuming all members of this space to be equiprobable, we get a probability space in the usual sense of the word (a set with a measure). The term 'random function' refers to a function selected randomly from this space.*

We can describe any $f \in F_n$ with the help of a table of the form:

$$\begin{bmatrix} 1 & 2 & \cdots & i & \cdots & n \\ f(1) & f(2) & \cdots & f(i) & \cdots & f(n) \end{bmatrix}$$

In general, there is a possibility of collisions - a collision occurs when $f(i) = f(j)$ and $i \neq j$. If f is a permutation, however, collisions do not occur. A 'random permutation' will, similarly to 'random functions', refer to a permutation on the set $\{1, .., n\}$ randomly chosen from the set of all permutations on $\{1, .., n\}$, assumed equiprobable.

It is often convenient to describe functions and permutations on the set $\{1, .., n\}$ in terms of their directed graphs, by which the following is meant. For any $f \in F_n$ we consider the directed graph, which has the points $\{1, .., n\}$ as nodes, and as edges the ordered pairs $(i, f(i))$ for all $1 \leq i \leq n$. The typical graph of a random function is a collection of cycles with trees attached. One of such cycles is scketched in Figure 9.1.

The graph of a random permutation looks like the one in Figure 9.2. Since collisions are not allowed, each node in the graph has a unique predecessor, and the graph is simply a collection of disjoint cycles.

We are interested in evaluating some parameters of random functions and permutations. In the sequel some results that had been obtained will be stated. Given a function from the set $\{1, .., n\}$ to itself, we define a point to be *cyclic* if it belongs to a cycle; a point is *terminal* if it has no preimage; an *image* point is a non terminal point; a k*-th iterate* image point is an image point of $f^{(k)} = f \circ f \circ ... \circ f$.

Theorem 9.2.1. *The expectations of parameters (1) to (5) of a random function from the set $\{1, .., n\}$ to itself, has the following asymptotic appearance as $n \to \infty$:*
(1) # connected components: $\frac{logn}{2}$
(2) # cyclic points: $\sqrt{\frac{\pi n}{2}}$
(3) # terminal points: $\frac{n}{e}$
(4) # image points: $(1 - \frac{1}{e})n$

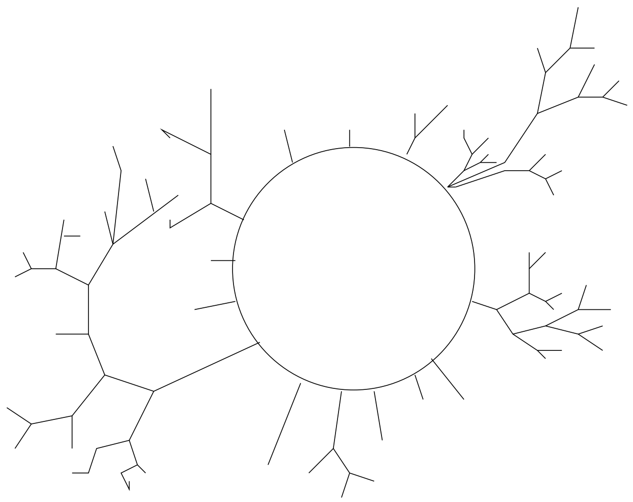

Figure 9.1: Component of a random function graph

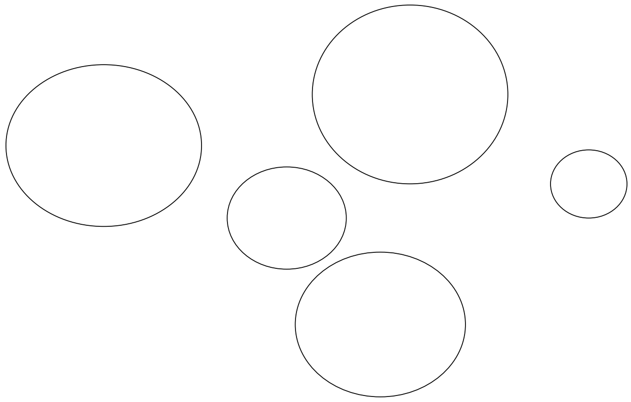

Figure 9.2: Random permutation graph

(5) # k-th iterate image points: $(1-\tau_k)n$, *where* τ_k *satisfies* $\tau_0 = 0$, $\tau_{k+1} = e^{-1+\tau_k}$.

Theorem 9.2.1 gives estimates of some specific 'global' characteristics in a random function. The next theorem, Theorem 9.2.2, will give estimates of characteristics of random functions as seen 'locally' from a random point. Let f be a function from the set $\{1, .., n\}$ to itself, and fix a node i_0 in its directed graph. Iterating the function f, which can be described as going along the edges of the graph starting from the node i_0, we get the sequence i_0, $i_1 = f(i_0)$, $i_2 = f(i_1) = f(f(i_0))$ etc. Since there are n nodes in our graph, at most after $n-1$ iterations we will find ourselves at a node which we have already visited. This means that from that point begins a cycle of the function f. The *tail length* of i_0, denoted by $\lambda(i_0)$ is the length of the path from i_0 to i_c where i_c is the first already encountered point. The *cycle length* of i_0 is denoted by $\mu(i_0)$ and is the length of the cycle starting at i_c. $\rho(i_0) = \lambda(i_0) + \mu(i_0)$ is called the ρ-length of i_0 and is the length of the nonrepeating trajectory of i_0. Considering the whole directed grapf of f, the *tree size* of i_0 is the size of the maximal tree (rooted on a cycle) that contains i_0; the *predecessor size* of i_0 is the size of the maximal tree with root i_0.

Theorem 9.2.2. *Seen from a random point on the directed graph of a random function, the expectations of parameters (1) to (6) have the following asymptotic forms:*
(1) Tail length (λ): $\sqrt{\frac{\pi n}{8}}$
(2) Cycle length (μ): $\sqrt{\frac{\pi n}{8}}$
(3) ρ*-length:* $\sqrt{\frac{\pi n}{2}}$
(4) Tree size: $\frac{n}{3}$
(5) Component size: $\frac{2n}{3}$
(6) Predecessor size: $\sqrt{\frac{\pi n}{8}}$

Some other results on the extremal statistics of random functions have been obtained, an example given in the following theorem.

Theorem 9.2.3. *We have the following expected values for the parameters below (asymptotic form):*
(1) Maximum cycle length, μ_{max}: $c_1\sqrt{n}$, $c_1 \approx 0.782$.
(2) Maximum tail length, λ_{max}: $c_2\sqrt{n}$, $c_2 = \sqrt{2\pi log 2} \approx 1.737$.
(3) Largest tree size: $d_1 n$, $d_1 \approx 0.48$.
(4) Largest component size: $d_2 n$, $d_2 \approx 0.76$.

Reasoning similar to the one used to explain the Birthday Paradox may be used to explain why the expected maximal cycle length of a random function is proportional to $\sqrt{n}$ (and is therefore quite small). In fact the following holds:

Proposition 9.2.1. *Let f be a random function from the set $\{1, .., n\}$ to itself. If f is evaluated $k > \sqrt{2cn}$ times for some constant c, the probability that two of the values collide is at least $1 - e^{-c}$; thus with $k \approx \sqrt{n}$ the probability of collision is $\geq 1/2$.*
Proof. Denote by C the event that there are no collisions in k-th iterate of f. By the Birthday Paradox arguments one gets:

$$Pr(C) \leq e^{-\frac{k(k-1)}{2n}}.$$

Thus for $Pr(C)$ to be less e^{-c} we need to have $k > \sqrt{2cn}$, implying the proposition.□

For some applications it is useful to know also the approximate parameters of random permutations. The directed graphs of the latter decompose, as had been already said, into disjoint cycles. We denote by α_i the number of cycles of length i. We have:

Theorem 9.2.4. *Given a random permutation on n points, we have (E denotes the expected value, Var the variance) (1)*

$$E\left(\sum_{i=1}^{n} \alpha_i\right) = \sum_{i=1}^{n} \frac{1}{i} = log\ n\ + \gamma + O(\frac{1}{n}),$$

where $\gamma \approx 0.577$ (Euler-Mascheroni constant).
(2)

$$Var\left(\sum_{i=1}^{n} \alpha_i\right) = \sum_{i=1}^{n} \frac{i-1}{i^2} = log(n) + \gamma - \frac{\pi^2}{6} + O(\frac{1}{n}).$$

(3)

$$lim_{n\to\infty} Pr(\alpha_i = 0) = \frac{1}{e}$$

(4)

$$lim_{n\to\infty} Pr(\alpha_i = k) = \frac{1}{k!} e^{-\frac{1}{i}} \left(\frac{1}{i}\right)^k$$

Denoting by $M(\alpha)$, $m(\alpha)$ the maximal and the minimal of α_i-s, we have:
(5)

$$lim_{n\to\infty} \frac{E(M(\alpha))}{n} = \lambda \approx 0.624$$

(6)

$$lim_{n\to\infty} \frac{E(m(\alpha))}{log\ n} = e^{-\gamma} \approx \frac{1}{2}$$

9.3 Hellman's tradeoff attack

Hellman's tradeoff attack is a method which optimizes to some extent the use of resources (time, memory) in the cryptanalysis of a cipher. Let us denote by T the time and by M the amount of memory which one needs to attack successfully a cipher with key space K of cardinality $|K| = N$. In what follows, the time unit is the maximal time needed to encrypt a message, and the memory unit is the maximal amount of memory needed to store the encrypted message. There are two obvious 'extreme' attacks against any cipher whatsoever:
· Exhaustive search: $T = N$, $M = 1$. It is a known plaintext attack, i.e. the cryptanalyst is assumed to know for some ciphertext the corresponding plaintext, finding the key used by encrypting the plaintext under different keys, until the given ciphertext is found.
· Table lookup: $T = 1$, $M = N$, which is a chosen plaintext attack. The cryptanalyst is assumed to have at his disposal a table of all possible ciphertexts for a chosen plaintext message, obtained by some preprocessing (i.e. the costs of those computations not taken into account), and ordered alphabetically by the ciphertext. Then given the ciphertext for the chosen message, the key is found from the table (the time it takes to search the table is neglected, being logarithmic in N).
Both of those attacks may require an unacceptably large amount of one of the resources, thus it makes sense to analyze time-memory tradeoffs, in which the number of computations may be reduced by using larger memory and vice-versa. An ideal tradeoff would be one for which $TM = N$, such have been found for many searching problems. However for attacking a cipher the tradeoff is not as good.

Hellman's attack, which makes time-memory tradeoff possible, is analogously to the table lookup attack a chosen plaintext attack, and likewise it assumes a precomputation stage. We denote by S_k the encryption function under the key k. The chosen plaintext block P_0 is then transformed into a ciphertext block C_0, $C_0 = S_k(P_0)$. It is assumed that the ciphertext blocks and the key have the same length; if not, one simply chops the excessive bits of the ciphertext. We can now view $S_k(P_0)$ as a function $f : K \to K$ from the set of keys to the set of keys. In Figure 9.3 the construction of the function f is depicted.

We describe the precomputation stage: f is iterated t times on each of the m randomly chosen keys $x_{10}, .., x_{m0}$, giving the points $x_{i,j} \in K$, $x_{i,j} = f(x_{i,j-1})$, $1 \le i \le m$, $0 \le j \le t$. The

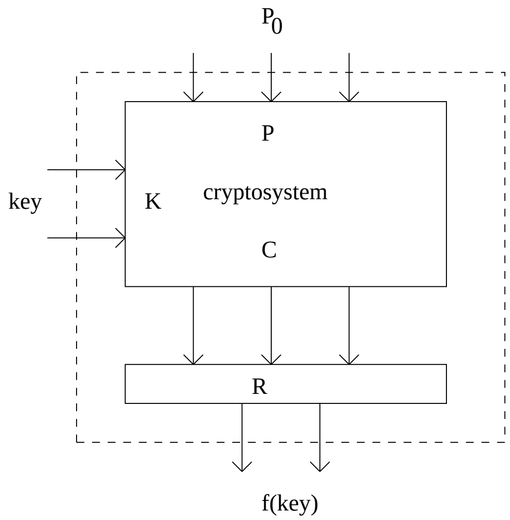

Figure 9.3: Construction of f

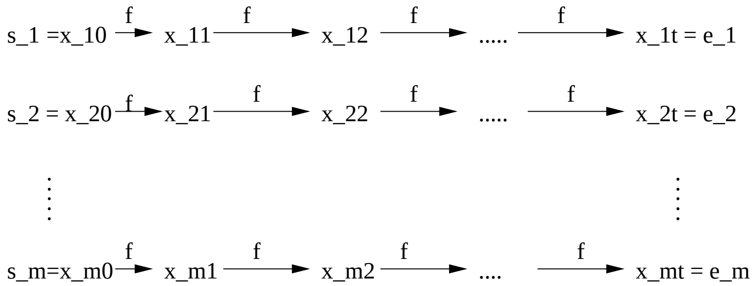

Figure 9.4: Table of images under f

iterative process gives the table of images under f depicted in Figure 9.4. The parameters m and t are chosen later, in order to set the appropriate time-memory tradeoff. To save memory, only the m starting points and their corresponding end points are kept (denoted by s_i,e_i respectively), and the resulting table is sorted by the end points, finishing the precomputation stage.
Let us now describe the attack. We are given the ciphertext C_0 of a chosen plaintext P_0 under an unknown key k. Chopping this ciphertext if necessary to the size of the key, we get Y_1. We may check if Y_1 is one of the end points previously stored. If this is the case, and $Y_1 = e_i = x_{i,t}$, $x_{i,t-1}$ is an obvious candidate for the unknown key k. It is only a candidate, since e_i may have more than one preimage under f; if indeed it is a wrong candidate, we refer to such a case as a 'false alarm'. $x_{i,t-1}$ is computed as $f^{t-1}(s_i)$, which is the reason for keeping the starting points in memory. To check if $x_{i,t-1}$ is the right candidate and not a false alarm, it suffices to encrypt under it the chosen message P_0, and check whether we obtain C_0 as a result.
Now, if Y_1 is not one of the stored end points, or if a false alarm was found, $Y_2 = f(Y_1)$ is computed. It is again checked whether Y_2 coincides with one of the end points. If yes, we again have a candidate for the key. If it turns to be a false alarm, or if there is no end point coinciding with Y_2, $Y_3 = f(Y_2)$ is computed, and so on.

We now analyze Hellman's attack. We say that a point $p \in K$ is *covered* by the table obtained in the precomputation stage, if it is equal to one of $x_{i,j}$, $1 \leq i \leq m$, $1 \leq j \leq t$. Note that a point does not have to appear in the table stored in memory in order to be covered by that table (recall that only starting and end points are actually stored in memory). When choosing at random a point from K, we call it a success, if the point chosen is covered by the table. In this case iterating f at most t times on that point will give us an end point. Thus success means that we can arrive, using Hellman' attack, at a candidate for the unknown key after at most t steps. The probability of success depends on how many distinct points are covered by the table. If the table covers $|A|$ keys, $A \subset K$, then the probability of success is $P(S) = |A|/N$. In the ideal case, when all $x_{i,j}$-s are distinct, we have $|A| = mt$, and the probability of success is mt/N. Since the amount of memory used is m (up to a constant factor), and the time to get to the first candidate is t (at most), we see that to maintain a fixed probability of success (in other words, the proportion of the keys which can be found using the information obtained in the precomputation stage), only the ratio mt has to be kept fixed - a clear tradeoff of time and memory. However, when mt is not small, we can no longer assume that the points covered by the table are all distinct. We have, however, the following theorem.

Theorem 9.3.1. *The probability of success is bounded from below by*

$$P(S) \;\geq\; \frac{1}{N} \sum_{i=1}^{m} \sum_{j=0}^{t-1} \left(\frac{N-it}{N}\right)^{j+1}.$$

Proof. We go over the points $x_{i,j}$ covered by the table, ordered such that $(i_1, j_1) \leq (i_2, j_2)$ if $i_1 < i_2$ or if $i_1 = i_2$ and $j_1 \leq j_2$. Let us denote by $I(X)$ the indicator function of the event X. The number of distinct points covered by the table is given by

$$\sum_{i=1}^{m} \sum_{j=0}^{t-1} I(x_{i,j} \textit{ is new}).$$

We have for the probability of success $P(S) = E(|A|/N)$, where A is the set of points covered by the table, and E is the expectation. Since the expectation of a sum of random variables is equal to the sum of their expectations, we have:

$$P(S) = E\frac{1}{N} \sum_{i=1}^{m} \sum_{j=0}^{t-1} I(x_{i,j} \textit{ is new}) = \frac{1}{N} \sum_{i=1}^{m} \sum_{j=0}^{t-1} Pr(x_{i,j} \textit{ is new}).$$

Estimating, we have:

$$Pr(x_{i,j} \textit{ is new}) \geq Pr(x_{i,0}, ..., x_{i,j} \textit{ are all new}) = Pr(x_{i,0} \textit{ is new}) Pr(x_{i,1} \textit{ is new} | x_{i,0} \textit{ is new})...$$

Note that each factor on the right cannot be less than

$$\frac{N-it}{N,}$$

since at most it distinct items can be chosen prior to choosing x_{ij}. Thus

$$Pr(x_{i,j} \textit{ is new}) \geq \left(\frac{N-it}{N}\right)^{j+1},$$

thus establishing the inequality. $\square$

Using the approximation

$$\left(\frac{N-it}{N}\right)^{j+1} \approx e^{-\frac{ijt}{N}},$$

it is possible to see that if $mt^2 \gg N$, most terms in the sum above will be small, implying there is no big gain to increase mt^2 much beyond N. When $mt^2 \ll N$, each term in the sum is close to 1, and we have a lower bound on $P(S)$ which is close to the upper bound mt/N. Thus it is reasonable to set $mt^2 \approx N$. Numerical evaluations show that in this case $P(S)$ is approximately $0.80mt/N$, and it is not sensitive to changes in m, t as long as $mt^2 = N$ is not altered.
Another indication given by the lower bound is that $P(S)$ is small. For example, taking $m = t = N^{1/3}$ gives $P(S) = N^{-1/3}$. This problem can be overcome as follows. Instead of just chopping the extra bits of the ciphertext C_0, we choose another way to produce a key-size string from the ciphertext. We name the resulting function the reduction function. The point is that minor changes in reduction functions, produce nearly independent random functions $f : K \to K$. Thus if we take $N^{1/3}$ different reduction functions, and produce for each of them a table of size $N^{1/3}$, allowing running time $N^{1/3}$, we get that the memory storage needed is $N^{1/3} \cdot N^{1/3} = N^{2/3}$. The running time is $N^{1/3}$ to get the first candidate for the key for each table; since there are $N^{1/3}$ tables, the total running time is of the order of $N^{1/3} \cdot N^{1/3} = N^{2/3}$. As for the probability of success, it is $N^{-1/3}$ for each table. Because different reduction functions produce almost independent random functions the total probability of success is the sum of probabilities of success for each of the tables, thus giving an overall probability of $N^{1/3} \cdot N^{-1/3} = O(1)$.

Thus we see that instead of the $TM = N$ tradeoff which would result in $T = N^{1/2}$, $M = N^{1/2}$, we get $T = N^{2/3}$, $M = N^{2/3}$. In the general case we have the following complexity bounds: $mt^2 = N$, and to produce non negligeable probability of success using tables each of which covers mt points, we have to have $N/mt = mt^2/mt = t$ tables. Each table requires memory m. Thus the total amount of memory required is $M = mt$. During the attack, the tables are scanned one by one, and for each one t steps are needed. Thus the total amount of time required is $T = t \cdot t = t^2$.

A troublesome point in the exposition above is that it was not shown that the rate of false alarms is small enough, and does not affect the complexity calculations. Indeed, the calculations were made under the assumption that in the majority of cases a candidate for the unknown key is indeed the unknown key. However there is the following theorem.

Theorem 9.3.2. *In each table the expected number of false alarms, $E(F)$, is of the order of mt^2/N.*

The time to check each false alarm in each table is t - the price for getting from the starting point to the candidate for the unknown key and encrypt under it the chosen plaintext. Since there are t tables, we see that the overall increase in the running time is of the order of

$$t \cdot \frac{mt^2}{N} \cdot t = t^2$$

which is exactly the running time computed earlier. Thus false alarms merely increase the running time by some constant factor, and the complexity calculations are thus justified.

Bibliography

[1] Martin Hellman, *A Cryptanalytic Time-Memory Trade-Off*, IEEE Transactions on Information Theory, Vol. IT-26, No.4, July 1980

[2] Philippe Flajolet, Andrew Odlyzko, *Random Mapping Statistics*, Lecture Notes in Computer Science 434, EUROCRYPT'89, pp.329-354,1989.

[3] Solomon Golomb, *Shift Register Sequences*, San Francisco, Holden-Day, 1967, pp.175-180.