Multi-Block Analysis of the Unix crypt Cipher

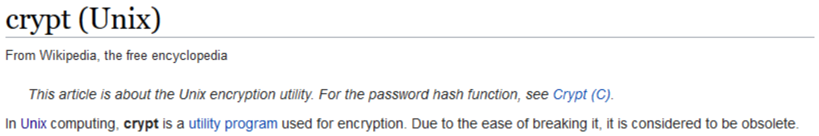
Charlie Gerrie

Dalhousie University

October 2019

History

Unix crypt first released in 1973.



crypt (Unix)

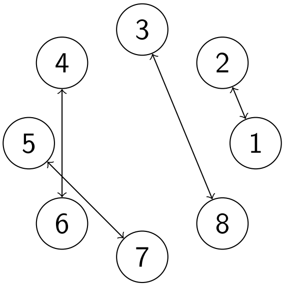
From Wikipedia, the free encyclopedia

This article is about the Unix encryption utility. For the password hash function, see Crypt (C).

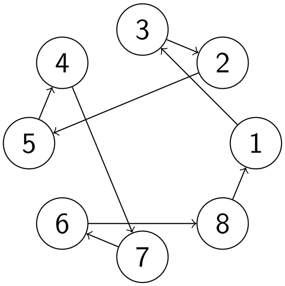
In Unix computing, **crypt** is a utility program used for encryption. Due to the ease of breaking it, it is considered to be obsolete.

The Key

The key is composed of two permutations acting on the integers modulo 256. The first, σ is fixed-point free and self-inverse, and the second, τ is a 256-cycle.



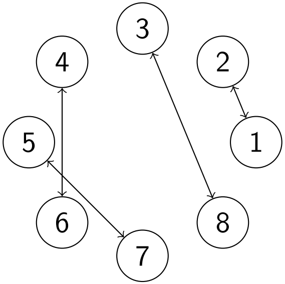
σ



τ

The Key

The key is composed of two permutations acting on the integers modulo 256. The first, σ is fixed-point free and self-inverse, and the second, τ is a 256-cycle.



σ

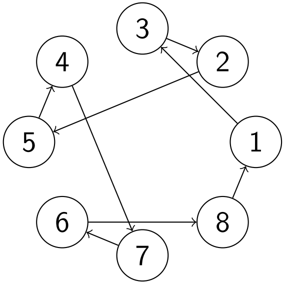
$$\sigma(x) \neq x$$

$$\sigma^2 = 1$$

The Key

The key is composed of two permutations acting on the integers modulo 256. The first, σ is fixed-point free and self-inverse, and the second, τ is a 256-cycle.

$$\mathtt{Ord}\tau = 256$$



τ

Encoding

- Let $i = 256j + k$
- We can then encode p_i into the ith character of the ciphertext using this equation:

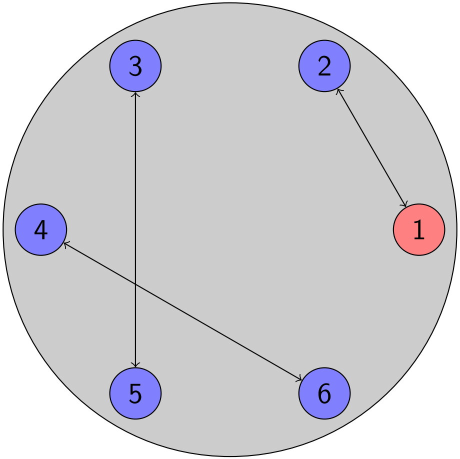
$$c_i = \kappa^{-k}\tau^{-j}\sigma\tau^{j}\kappa^{k}p_i$$

where κ is the successor function $\kappa(x) = x + 1 \mod 256$.

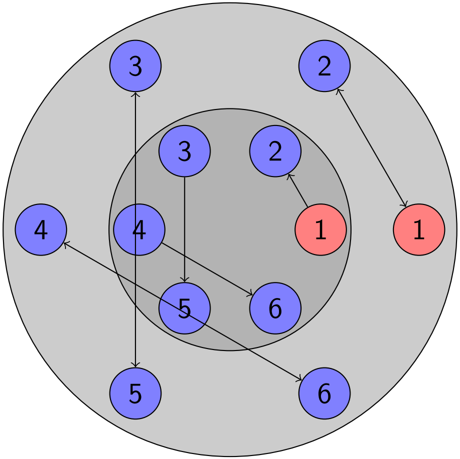
Rotary Cipher

$$\begin{aligned} c_i &= \kappa^{-k}\tau^{-j}\sigma\tau^{j}\kappa^{k}p_i \\ &= \kappa^{-k}(\rho^{-1}\kappa^{-j}\rho)(\rho^{-1}\pi\rho)(\rho^{-1}\kappa^{j}\rho)\kappa^{k}p_i \\ &= \kappa^{-k}\rho^{-1}(\kappa^{-j}\pi\kappa^{j})\rho\kappa^{k}p_i \end{aligned}$$

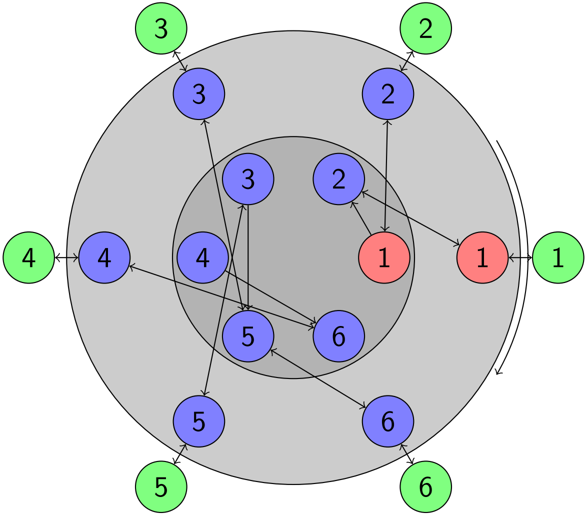
Rotary Cipher



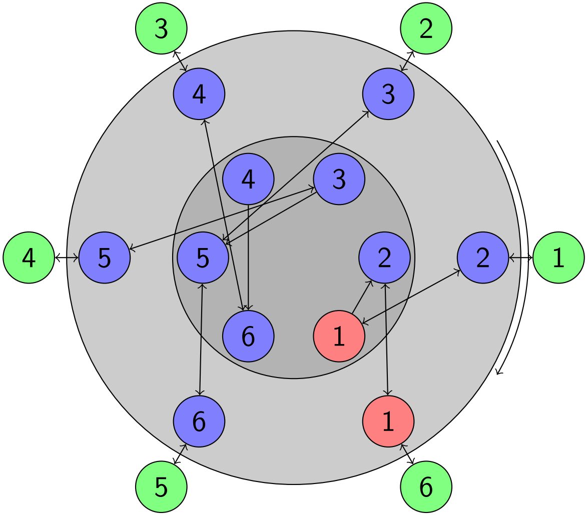
Rotary Cipher



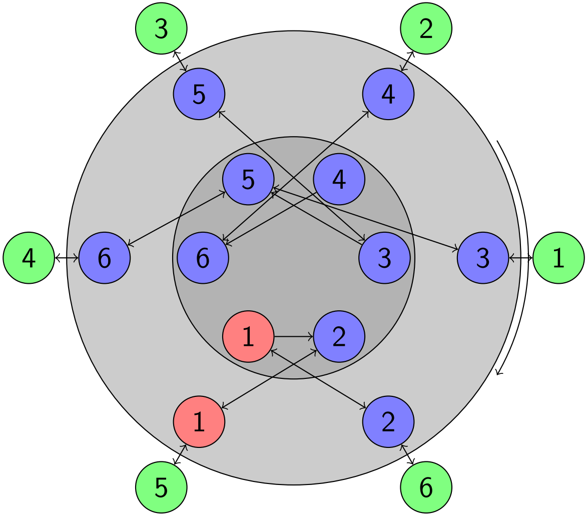
Rotary Cipher



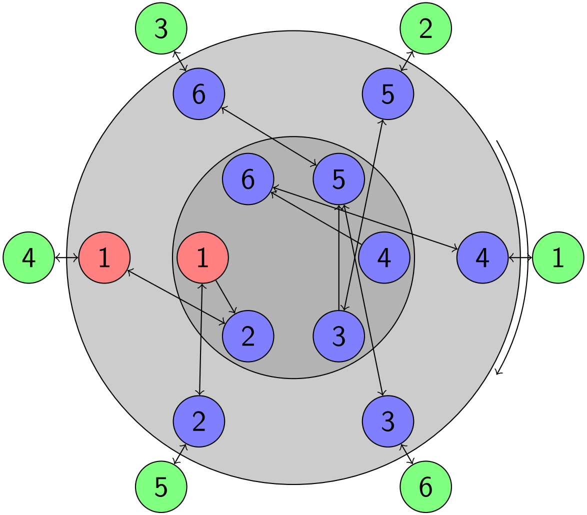
Rotary Cipher



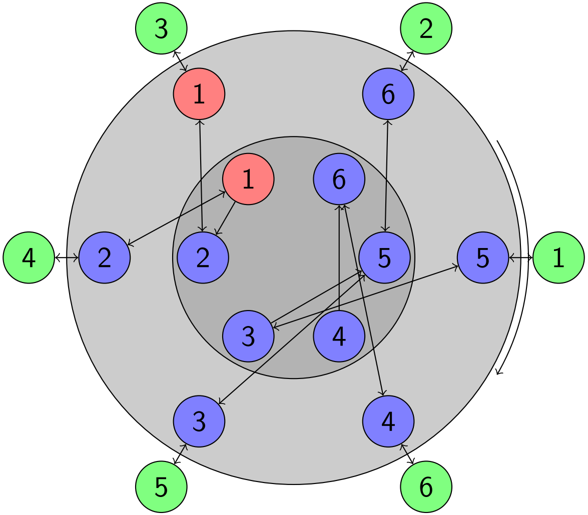
Rotary Cipher



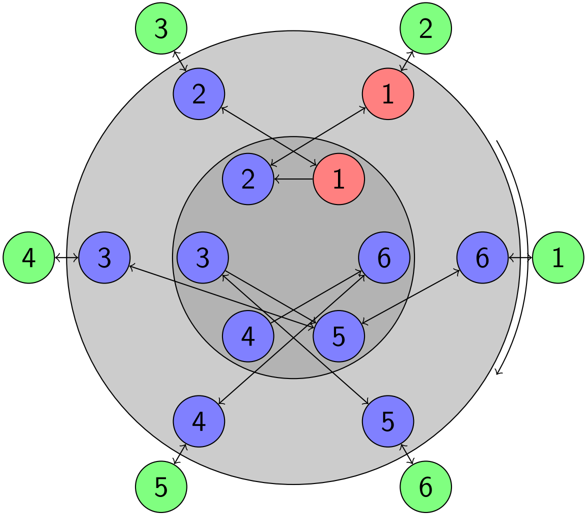
Rotary Cipher



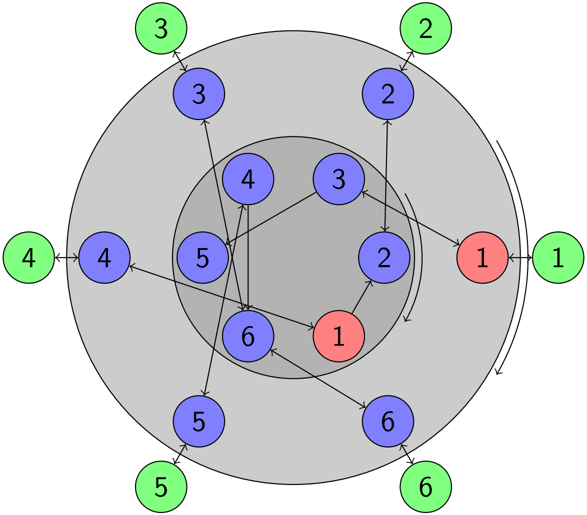
Rotary Cipher



Rotary Cipher



Rotary Cipher



Block Keys

- Unix crypt splits the plaintext to be encoded into blocks of 256 characters.
- Recall the equation for encoding the *i*th character, where $i = 256j + k$:

  $$c_i = \kappa^{-k}\tau^{-j}\sigma\tau^{j}\kappa^{k}p_i$$

  Notice that the power on τ only increases when j does, so every 256 characters. Thus we can define the *j*th block key as:

  $$\sigma_i = \tau^{-j}\sigma\tau^{j}$$

Single-block

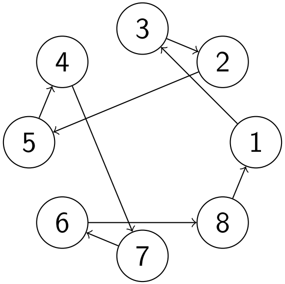
- **The single-block analysis gives us approximations of the block keys.**
- It does this by using *a priori* statistics about the probabilities of different characters being in proximity to other characters.
- We will denote these block key approximations by $\hat{\sigma}_i$.

Multi-block plan

Our plan is to take the block key approximations from the single-block analysis and "connect the dots". The hard part is reconstructing τ. Once we have τ it is relatively easy to recover the other half of the key, σ.

Wires

- For a given permutation σ, a *wire* is a pair (a, b) such that $\sigma(a) = b$.
- e.g. $(2, 5)$, $(7, 6)$



Multisets

- A multiset X of cardinality $|X|$ is the equivalence class of $|X|$-tuples up to permutations. Sorting the elements yields a unique representation.
- We say an element is *in* a multiset if it appears at least once in the tuple. The *multiplicity* of an element is the number of times it appears in the list.
- e.g. (1,1,2,2,2,3)

Some Algebra

- If we conjecture that (a, b) is a wire in τ then it must also be a wire in $\sigma_1 \tau \sigma_0^{-1}$.
- But the wire in $\sigma_1 \tau \sigma_0^{-1}$ is really three wires: a wire $(a, \sigma_1(a))$ in σ_1, a wire $(\sigma_1(a), \sigma_0(b))$ in τ, and a wire $(\sigma_0(b), b)$ in σ_0^{-1}. Thus, if (a, b) is a wire in τ then $(\sigma_1(a), \sigma_0(b))$ must also be a wire in τ. In general, all wires $(\sigma_{i+1}(a), \sigma_i(b))$ must be in τ.
- The *wire consequences* of (a, b) are all the wires that must also be in τ if (a, b) is.

(a) Commutative Diagram

$$\begin{tikzcd}
{} \arrow[r, "\sigma_0"] & {} \\
{} \arrow[u, "\tau"] \arrow[r, "\sigma_1"] & {} \arrow[u, "\tau"'] \\
{} \arrow[u, "\tau"] \arrow[r, "\sigma_2"] & {} \arrow[u, "\tau"'] \\
{} \arrow[u, "\tau"] \arrow[r, "\sigma_3"] & {} \arrow[u, "\tau"'] \\
{} \arrow[u, "\tau"] \arrow[r, "\sigma_4"] & {} \arrow[u, "\tau"']
\end{tikzcd}$$

Consequences

For a wire (a, b), we have

$$\mathtt{conseq}(a, b) = \{\!\{\hat{\sigma}_i(a), \hat{\sigma}_{i+1}(b)) : 0 \leq i < N - 1\}\!\} \tag{1}$$

where N is the number of $\hat{\sigma}$'s. This is a multiset.

Coincidences

We define an indicator function δ taking two arguments x and y

$$\delta(x,y) = \begin{cases} 1 & \texttt{if } x = y \\ 0 & \texttt{if } x \neq y \end{cases}$$

Then for any two multisets X and Y with elements x_i and y_j, where $0 \leq i < |X|$ and $0 \leq j < |Y|$, we define their *coincidence* as:

$$\texttt{coinc}(X;Y) = \sum_{i,j} \delta(x_i, y_j)$$

with the sum being over all possible pairs of i and j.

Coincidences

If we consider two wires (a, b) and (c, d), then we have

$$\texttt{coinc}(a, b, c, d) = \sum_{i=0}^{255} \sum_{j=0}^{255} \texttt{conseq}(a, b)_{i,j} \texttt{conseq}(c, d)_{i,j} \qquad (2)$$

where $\texttt{conseq}(a, b)_{i,j}$ is the multiplicity of the wire (i, j) in the consequences of (a, b).

Coincidences

$$\begin{tikzcd}
(1,20) \arrow[dr, leftrightarrow] & (2,4) \arrow[dr, leftrightarrow] \arrow[drr, leftrightarrow] & (2,4) \arrow[d, leftrightarrow] \arrow[dr, leftrightarrow] & (3,8) & (17,2) \arrow[d, leftrightarrow] & (17,2) \arrow[dl, leftrightarrow] \\
(0,4) & (1,20) & (2,4) & (2,4) & (17,2) & (18,21)
\end{tikzcd}$$

Statistics

Now that we can calculate the number of coincidences between the consequences of different wires, we will use statistics on this data to choose the correct wires in τ.

A Bag Example

- I have two bags filled with marked balls. One has 10 balls in it, the other 100.
- Just by iteratively removing balls from the bags and examining them, how can I tell which bag has the 10 balls and which the 100?

The Answer

We take a ball out, remember it, and place it back in the bag. Then we take another ball and note if we drew the same ball as last time. This will happen about 10% of the time for one bag and 1% of the time for the other. In other words, the probability of drawing the same ball twice in a row will be distributed differently for the two bags.

The Null Distribution

For each possible wire endpoint, we consider two hypotheses.

1. The alternative hypothesis: that this is the correct endpoint.
2. The null hypothesis: that this is the wrong endpoint.

The Null Distribution

In the null hypothesis case the endpoint is wrong. Thus none of the correlations from the conjugacies of the block keys apply, so the wires are chosen randomly.

X	0	1	2	...
$P(\texttt{coinc}(a,b,c,d) = X)$	0.965	0.0345	0.000618	...

Bayes' Law

Now we have two potential distributions and we would like to know which one our sample distribution follows. Let D_1 and D_2 be the events that the sample distribution follows the two distributions and S is the event of getting the samples that we did. Normally we would then use Bayes' theorem:

$$P(D_1|S) = \frac{P(S|D_1)P(D_1)}{P(S)}$$

But this would require us to know an *a priori* probability of the distribution and the samples. This we cannot know, so Bayes' theorem fails us.

Likelihood

- We will be using a concept from statistics and probability theory known as *likelihood*. If we consider two events A and B, then likelihood is related to conditional probability by the relation
$$\mathcal{L}(A|B) = P(B|A)$$
- It is easy to calculate $P(S|D_1)$ and $P(S|D_2)$. Assuming the samples are independent, they are simply the products of the pdf values for all the samples. Thus we can acquire $\mathcal{L}(D_1|S)$ and $\mathcal{L}(D_2|S)$.

Likelihood

- Now we can apply what is known as the Maximum Likelihood Method. What this means is that rather than choosing what the most *probable* wire endpoint for each wire, we choose the most *likely* one.
- The wire most likely to be correct is the wire least likely to be incorrect i.e. follow the null distribution.

Picking a wire

Here are some coincidences distributions for a wire:

X	0	1	2	...
$P(\texttt{coinc}(15, 1, c, d) = X)$	0.940	0.0580	0.00178	...
$P(\texttt{coinc}(15, 2, c, d) = X)$	0.940	0.0583	0.00147	...
$P(\texttt{coinc}(15, 3, c, d) = X)$	0.944	0.0515	0.00262	...
$P(\texttt{coinc}(15, 4, c, d) = X)$	0.940	0.0578	0.00167	...

Finding σ

- Recall that $\sigma_i = \tau^{-j}\sigma\tau^{j}$, so $\sigma = \tau^{j}\sigma_i\tau^{-j}$.
- Thus once we have recovered τ, we can construct as many guesses at σ as we have blocks. Then we simply pick the wires that occur in the majority of these.

Conclusion

Thus it is possible to algorithmically recover the key using only the ciphertext and the character-distribution data used by the single-block algorithm. Our work now is in trying to decrease the number of blocks we need to successfully recover the key.

Thank you

I would like to thank Peter Selinger, Dalhousie University and NSERC.