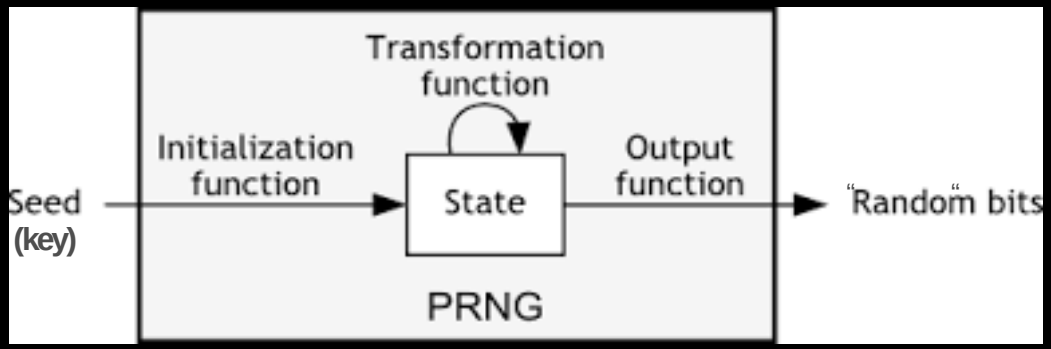
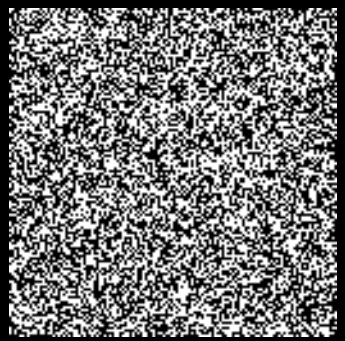
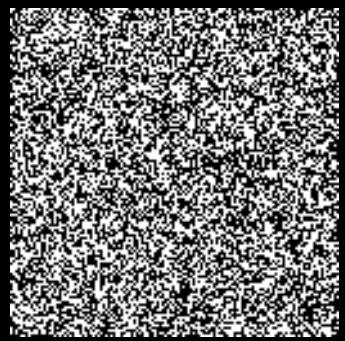
Finding the Weak Crypto Needle in a Byte Haystack

Ben Herzog
Check Point Malware Research Team | benhe@checkpoint.com

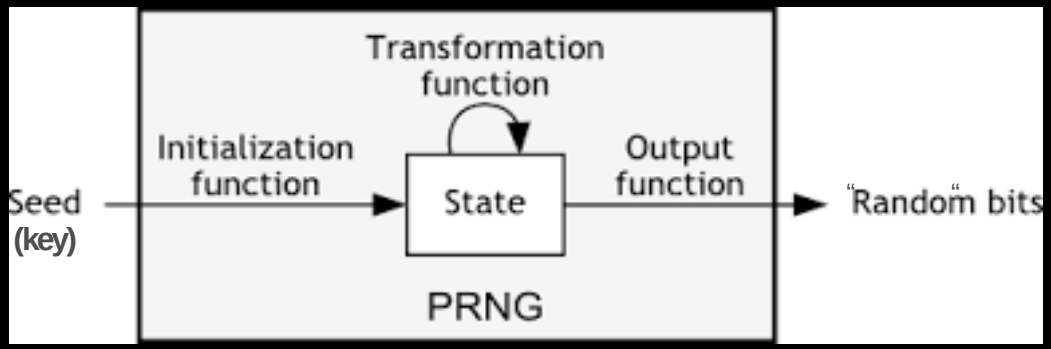
What's a Stream Cipher?



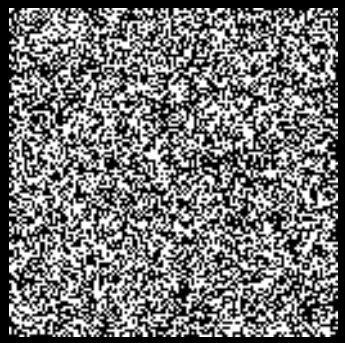
p ⊕ keystream = c

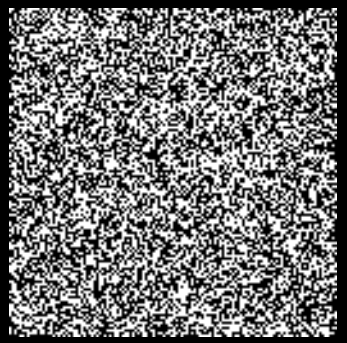




What's a Stream Cipher?

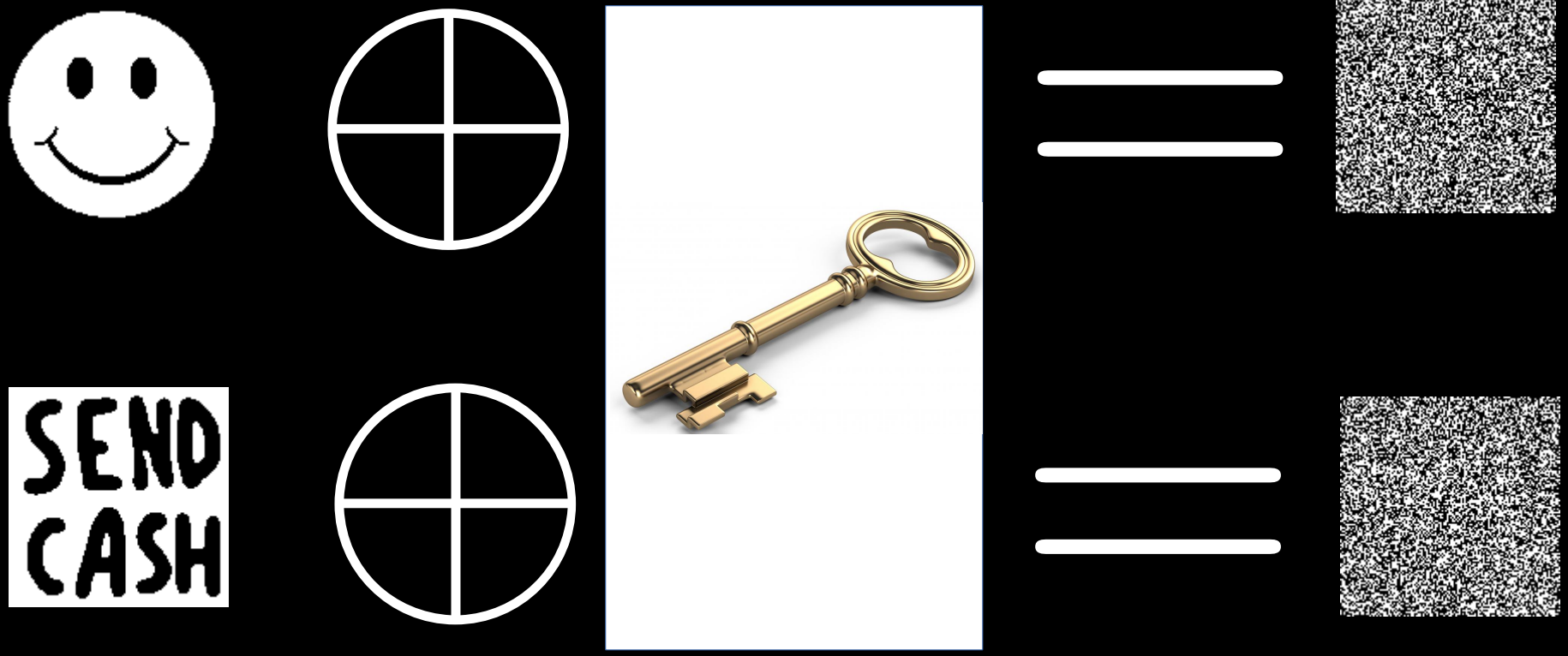


c ⊕ keystream = p



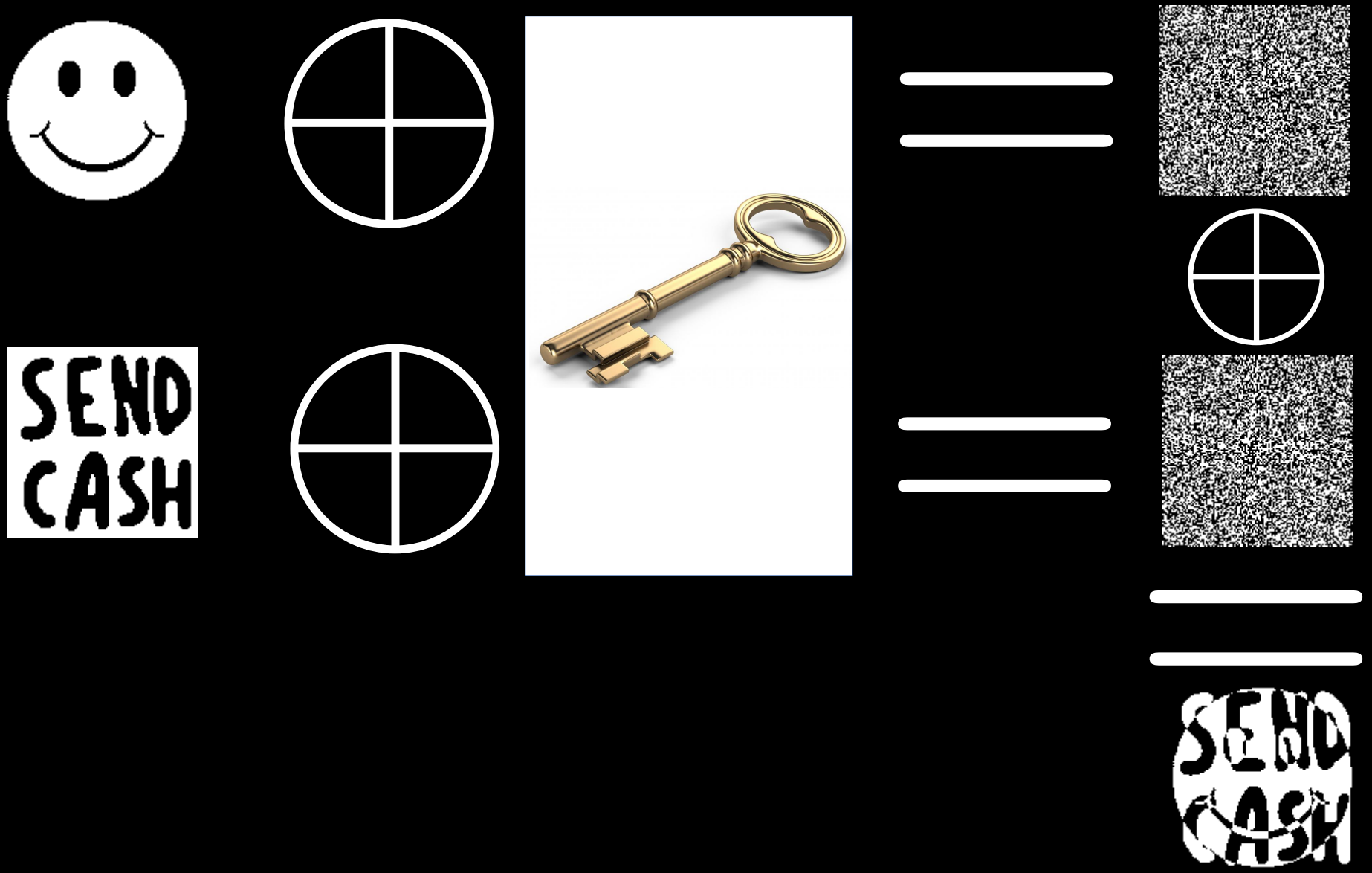


The Problem with Key Reuse

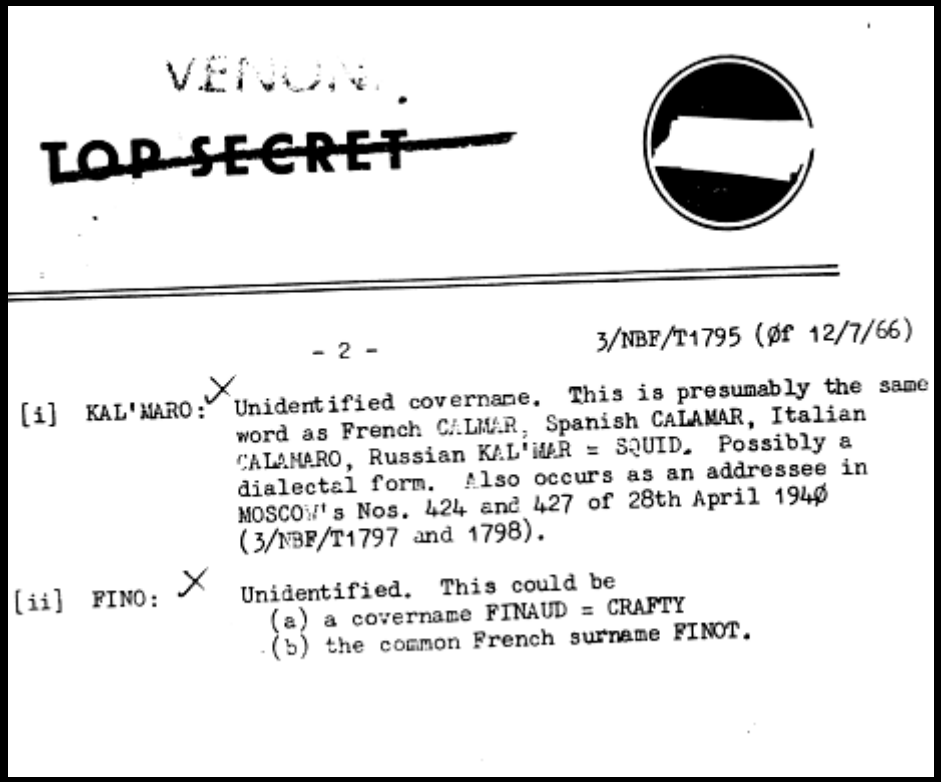


The Problem with Key Reuse

$$c_1 \oplus c_2 = (p_1 \oplus k) \oplus (p_2 \oplus k) = (p_1 \oplus p_2) \oplus (k \oplus k) = p_1 \oplus p_2$$



Why Detect Key Reuse? (1)



VENONA

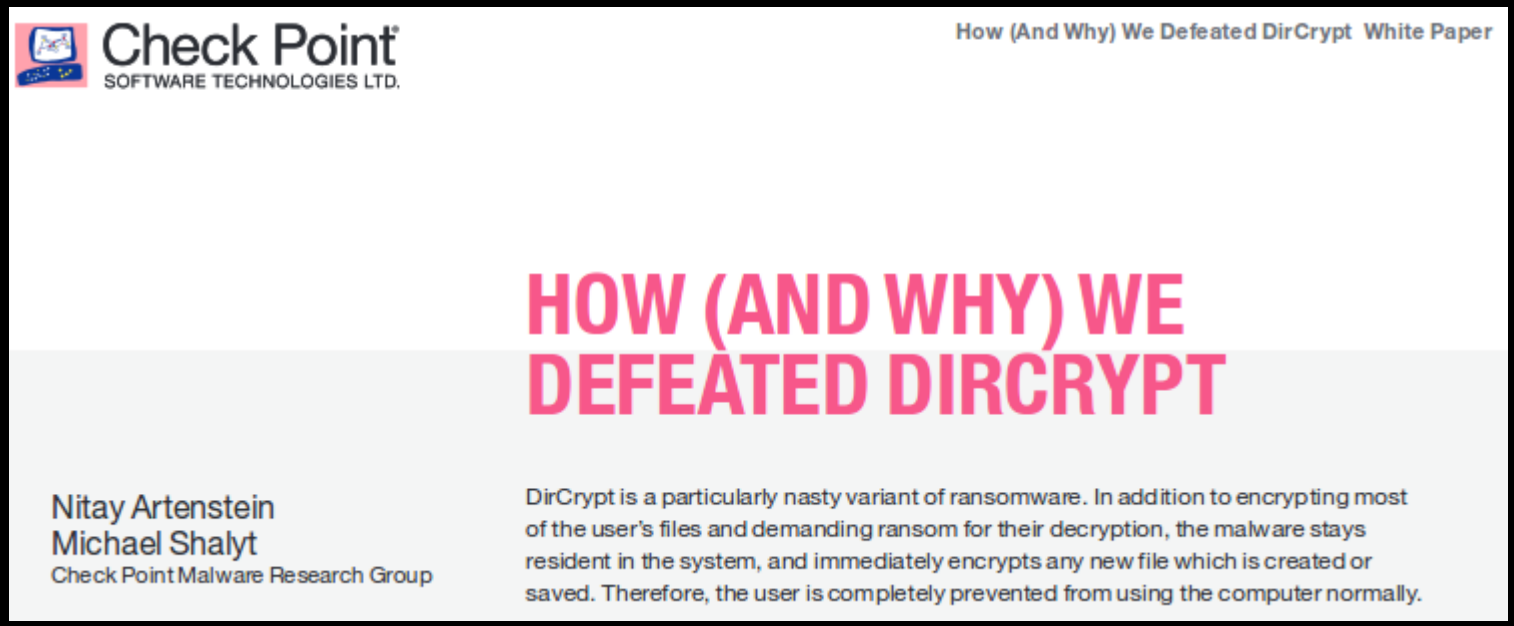
~~TOP SECRET~~

- 2 - 3/NBF/T1795 (øf 12/7/66)

[i] KAL'MARO: Unidentified covername. This is presumably the same word as French CALMAR, Spanish CALAMAR, Italian CALAMARO, Russian KAL'MAR = SQUID. Possibly a dialectal form. Also occurs as an addressee in MOSCOW's Nos. 424 and 427 of 28th April 194Ø (3/NBF/T1797 and 1798).

[ii] FINO: Unidentified. This could be
(a) a covername FINAUD = CRAFTY
(b) the common French surname FINOT.

Why Detect Key Reuse? (2)



Check Point
SOFTWARE TECHNOLOGIES LTD.

How (And Why) We Defeated DirCrypt White Paper

HOW (AND WHY) WE DEFEATED DIRCRYPT

Nitay Artenstein
Michael Shalyt
Check Point Malware Research Group

DirCrypt is a particularly nasty variant of ransomware. In addition to encrypting most of the user's files and demanding ransom for their decryption, the malware stays resident in the system, and immediately encrypts any new file which is created or saved. Therefore, the user is completely prevented from using the computer normally.

Why Detect Key Reuse? (3)

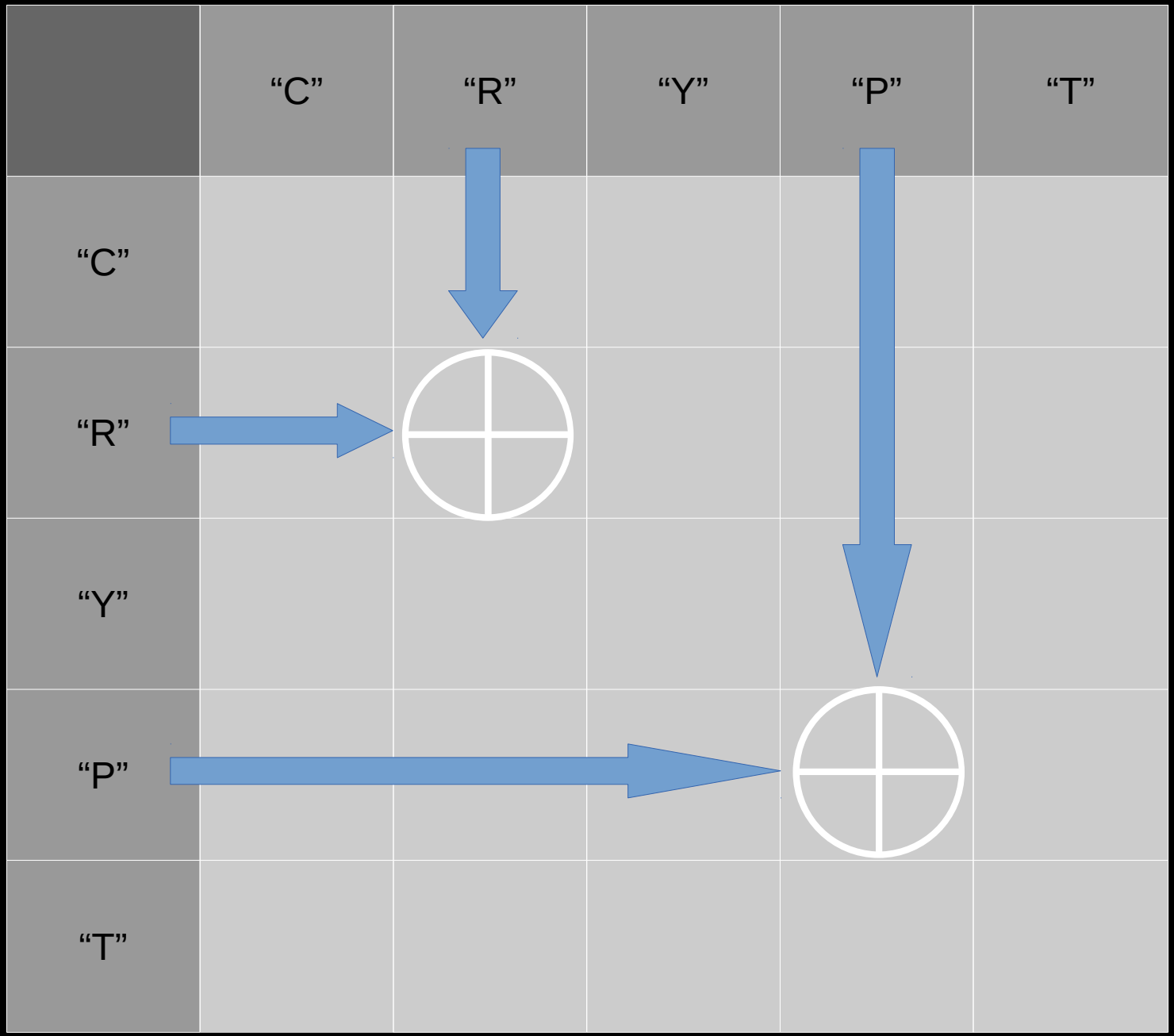
Ramnit traffic, recorded by TCPdump

```
    00000000  00 ff 0f 00 00 00 01 00  09 00 00 00 f8 a7 72 26 ........ ......r&
    00000010  16 81 98 fa bb                                    .....
00000000  00 ff 4c 00 00 00                                    ..L...
00000006  e2 00 21 00 00 00 ca 81  42 6a 16 c0 c4 bc 8e db ..!..... Bj......
00000016  b8 50 c1 f5 96 1d d3 e2  0d 62 53 ef fa 66 f5 42 .P...... .bS..f.B
00000026  8e c1 a7 8d 11 23 93 00  20 00 00 00 bf a4 30 60 .....#.. .....0`
00000036  10 93 9c eb ff 88 e3 0a  94 f7 98 1c d7 e6 04 63 ........ .......c
00000046  51 e8 a6 6a f7 44 83 cf  a6 8a 1b 20 00 ff 07 01 Q..j.D.. ... ....
00000056  00 00 f0 01 00 00 00 00  01 00 00 00 00 01 00 00 ........ ........
00000066  00 00 01 00 00 00 00 01  00 00 00 00 01 e9 13 19 ........ ........
00000076  00 00 a7 00 00 00 10 c2  01 52 22 f5 fd 89 ce e9 ........ .R"..... 
00000086  80 68 d8 ca af 28 e4 d0  35 5a 31 bf b1 25 ad 13 .h...(.. 5Z1..%..
00000096  d3 d9 c6 d9 4b 7d 82 cd  c1 86 5f b4 5a 6c e2 69 ....K}.. .._.Zl.i
000000A6  18 86 22 a7 c7 c3 b8 4a  ac 3d 6b 7d 0c e8 b0 69 .."....J .=k}...i
000000B6  81 b0 fd 7e 7f 8a 14 12  00 e1 66 d7 3d e9 3e 3a ...~.... ..f.=.>:
000000C6  d1 54 99 71 56 dd 75 9e  41 58 a1 20 b9 9c 6b 46 .T.qV.u. AX. ..kF
000000D6  5f a2 52 e6 03 aa 78 34  dc 65 ca 93 56 af c6 e2 _.R...x4 .e..V...
000000E6  54 47 f1 57 d5 e7 0a b4  e2 50 90 03 c5 a6 87 a7 TG.W.... .P......
000000F6  b8 2c d9 e6 f0 2b d8 09  64 43 88 ab 0d bc 0d 9b .,...+.. dC......
00000106  be 99 c5 dd 1e b5 f1 d0  73 a8 7a dc 7f 04 2f d4 ........ s.z.../.
00000116  20 61 1d 2b dc d4 fc 59  e1 f6 c5 81 c7 00 20 00  a.+...Y ......  .
00000126  00 00 bc fa 64 67 11 cd  cd be fc d1 b5 51 95 f8 ....dg.. .....Q..
00000136  99 4e 82 e8 51 63 53 bb  fa 35 f1 47 8f 9c f5 8d .N..QcS. .5.G....
00000146  18 23 00 03 00 00 00 cd  8e 4d 01 05 00 00 00 01 .#...... .M......
00000156  00 00 00 00 01 0d 00 00  00                     ........ .
0000015F  00 ff 0e 00 00 00                               ......
00000165  11 01 00 00 00 00 00 03  00 00 00 cd 8e 4d       ........ .....M
```

Why Detect Key Reuse? (4)

Let's Get Our Hands Dirty

The XORspace



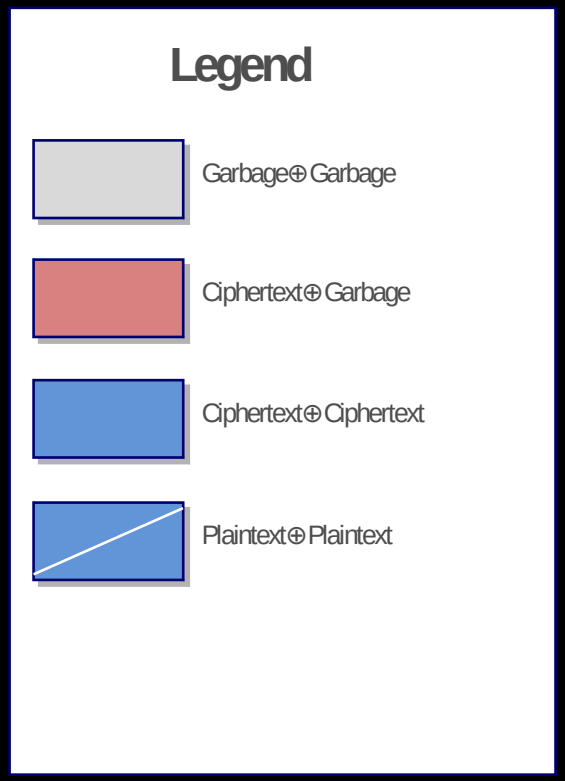
The XORspace

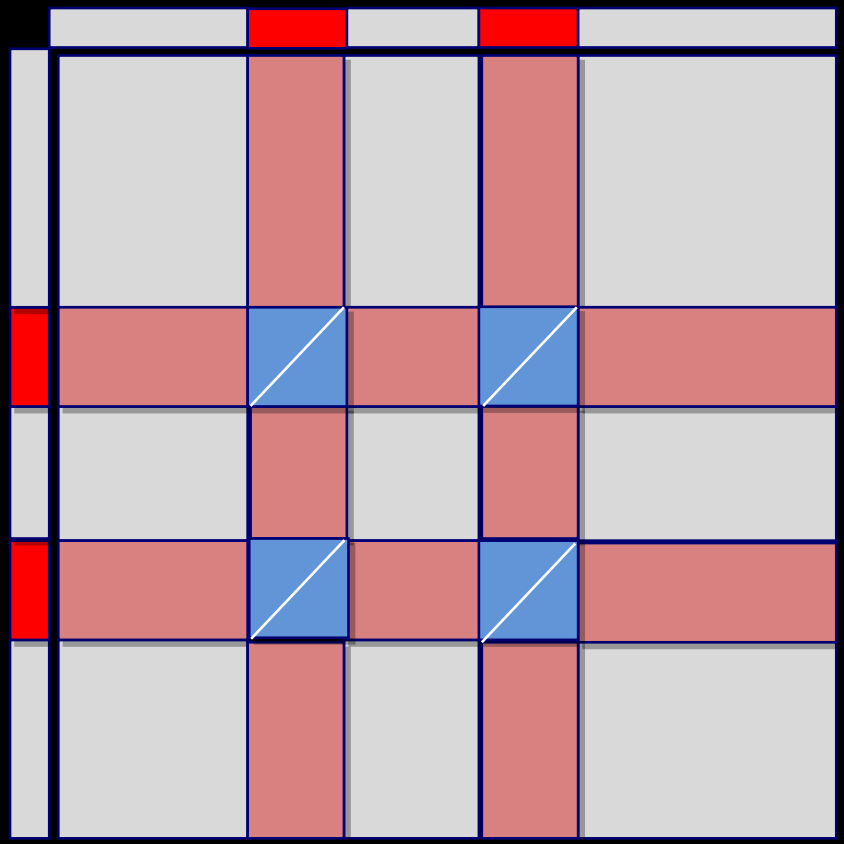
	“C”	“R”	“Y”	“P”	“T”
“C”	\x00	\x11	\x1A	\x13	\x17
“R”	\x11	\x00	\x0B	\x02	\x06
“Y”	\x1A	\x0B	\x00	\x09	\x0D
“P”	\x13	\x02	\x09	\x00	\x04
“T”	\x17	\x06	\x0D	\x04	\x00

The XORspace

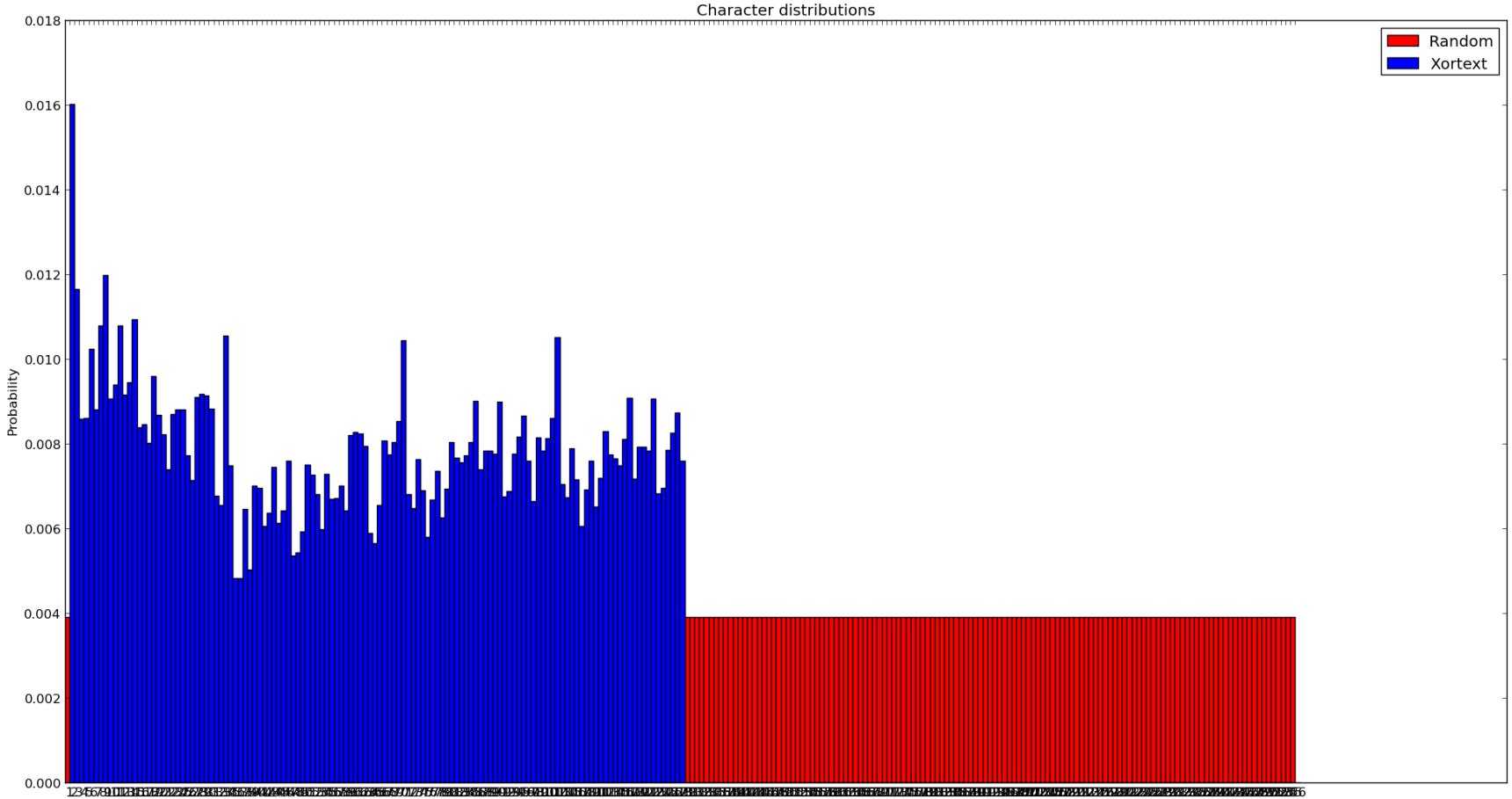
Input:

XOR Every Byte with Every Other Byte:



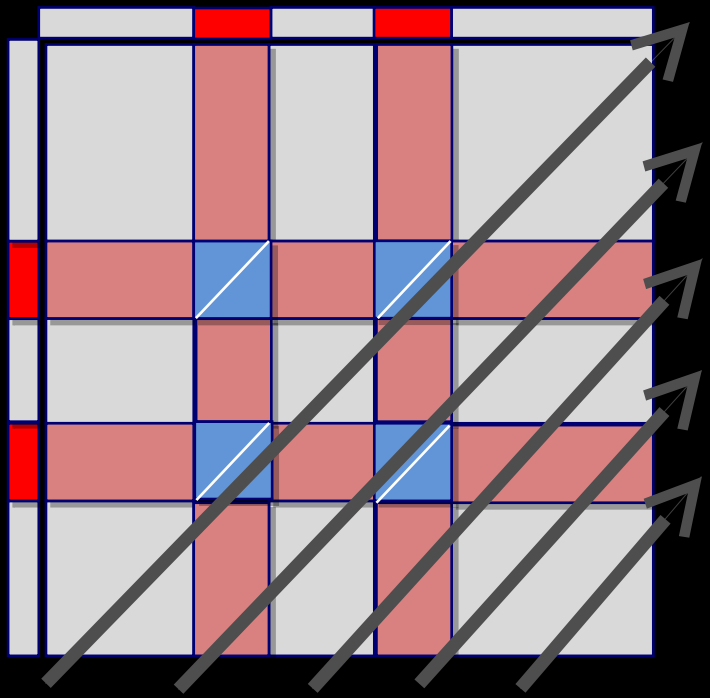


Byte Frequencies: Random vs. XorText



The Algorithm in a Nutshell

- Scan xorspace along the diagonals
- Look for “streaks” of positive evidence
- If positive evidence is “enough” - raise alarm

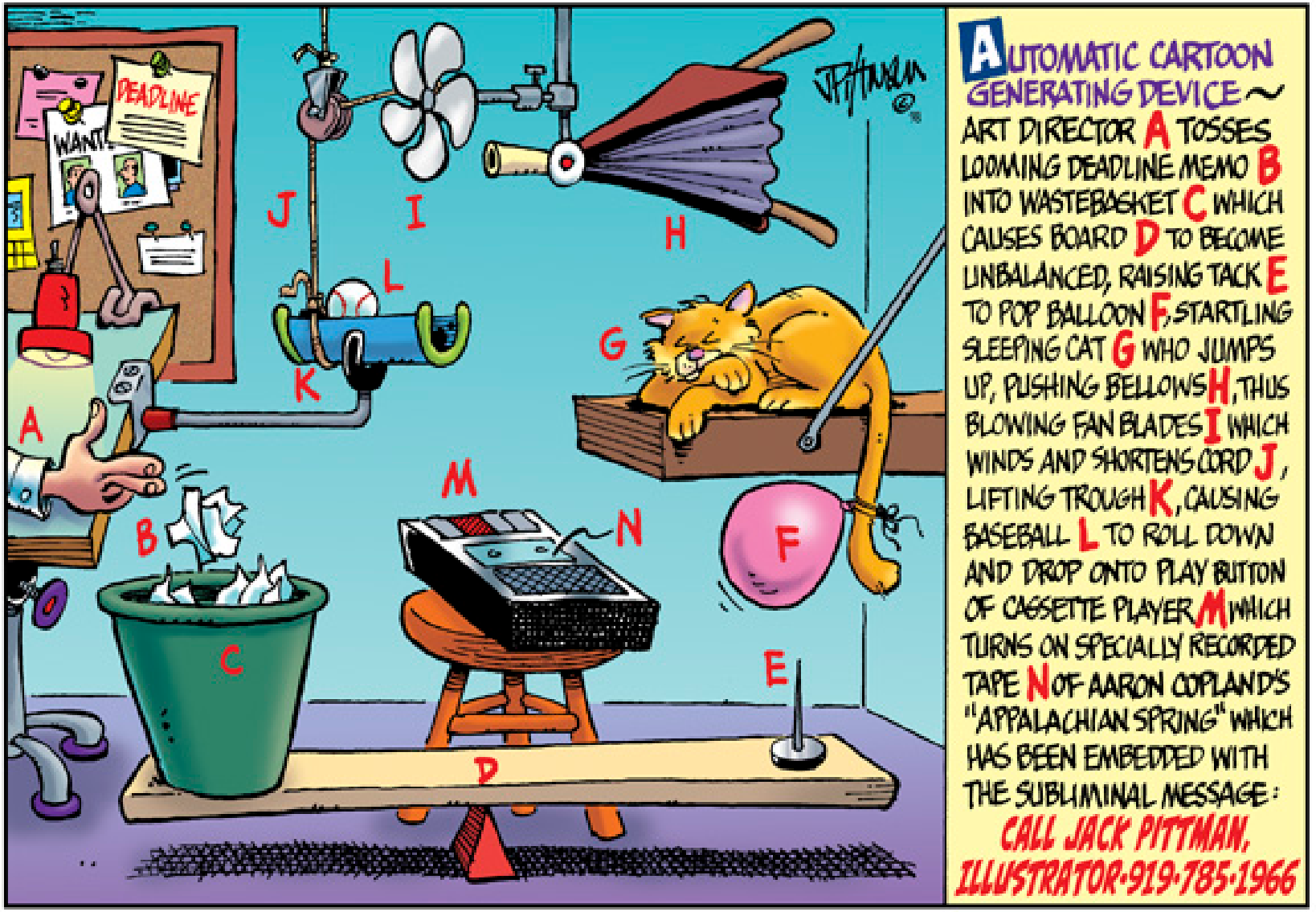


Math tl;dr

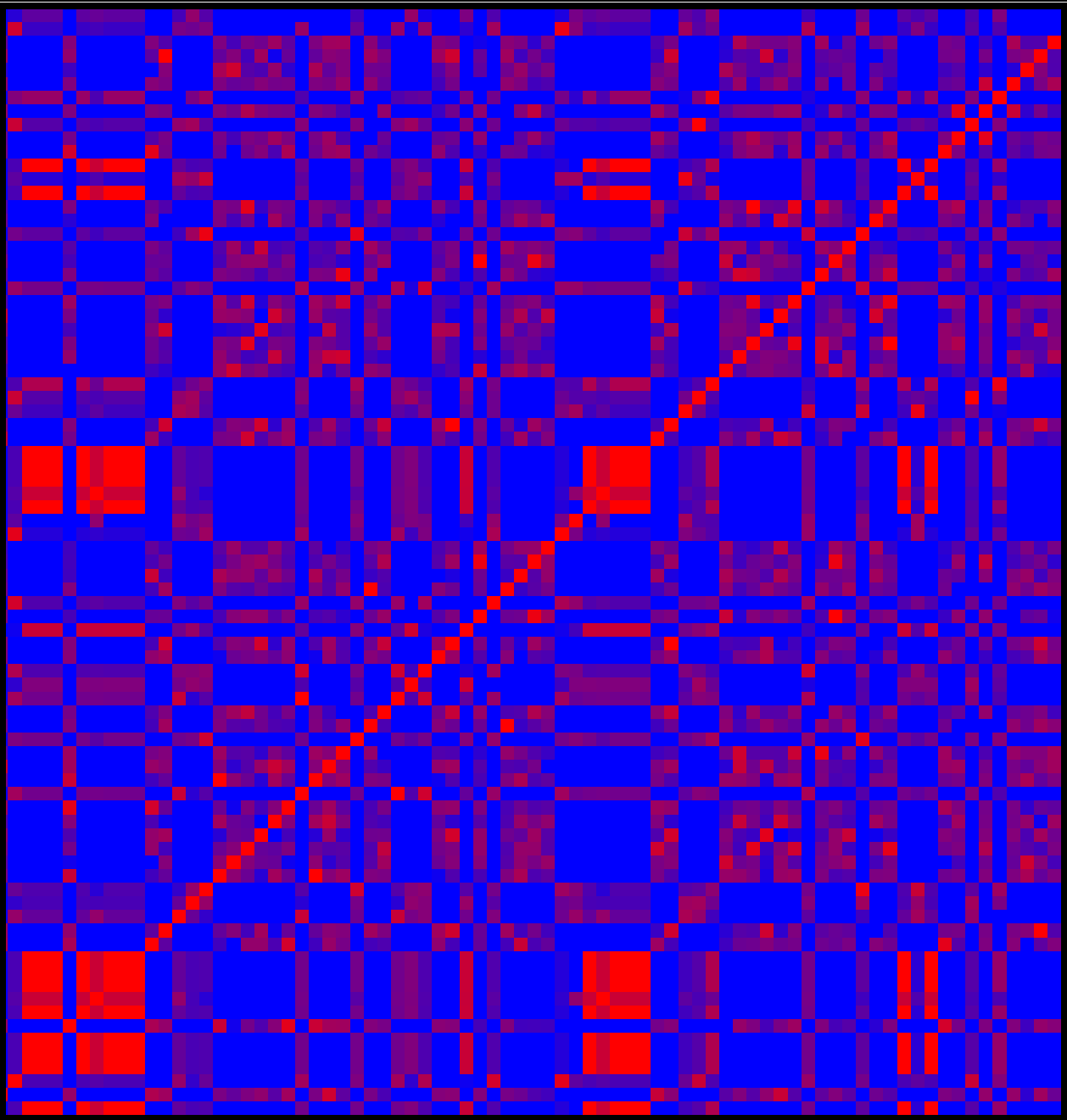
- How do you quantify “evidence”?
  - Naive Bayesian log-odds
- How much evidence is “enough”?
  - Enough that we can expect at most about 1 FP along the whole xortext
- If we set the bar that high, can we actually detect stuff?
  - It depends on the parameters of the problem, but generally yes

$$pr(fail) \leq \frac{\sigma^2}{M \left| \frac{2\log(N)-1}{M} - \mu \right|^2}$$

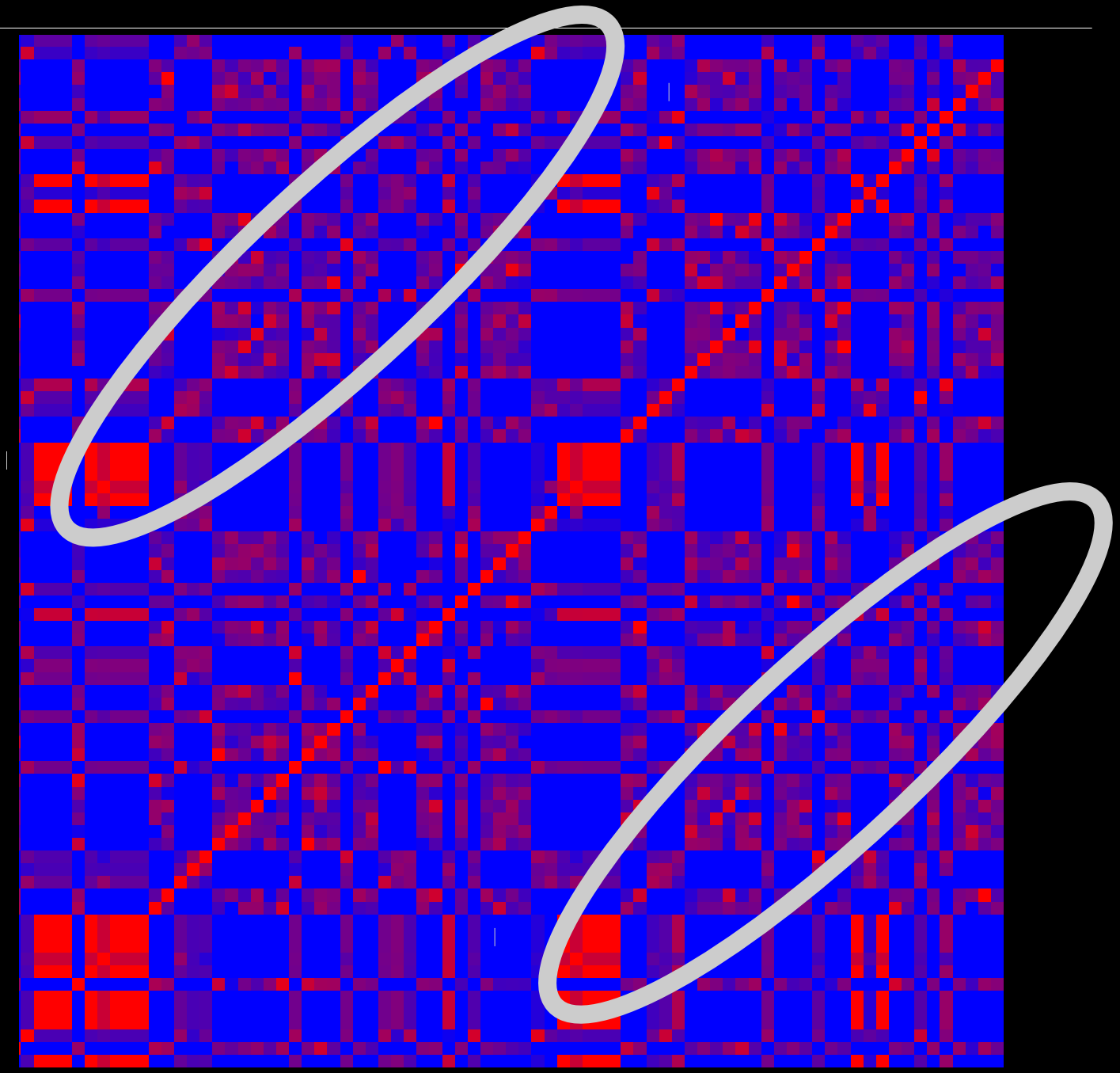
Demo


DEADLINE
WANT!
A
B
C
D
E
F
G
H
I
J
K
L
M
N
AUTOMATIC CARTOON GENERATING DEVICE ~
ART DIRECTOR A TOSSES LOOMING DEADLINE MEMO B INTO WASTEBASKET C WHICH CAUSES BOARD D TO BECOME UNBALANCED, RAISING TACK E TO POP BALLOON F, STARTLING SLEEPING CAT G WHO JUMPS UP, PUSHING BELLOWS H, THUS BLOWING FAN BLADES I WHICH WINDS AND SHORTENS CORD J, LIFTING TROUGH K, CAUSING BASEBALL L TO ROLL DOWN AND DROP ONTO PLAY BUTTON OF CASSETTE PLAYER M WHICH TURNS ON SPECIALLY RECORDED TAPE N OF AARON COPLAND'S "APPALACHIAN SPRING" WHICH HAS BEEN EMBEDDED WITH THE SUBLIMINAL MESSAGE:
CALL JACK PITTMAN, ILLUSTRATOR·919·785·1966

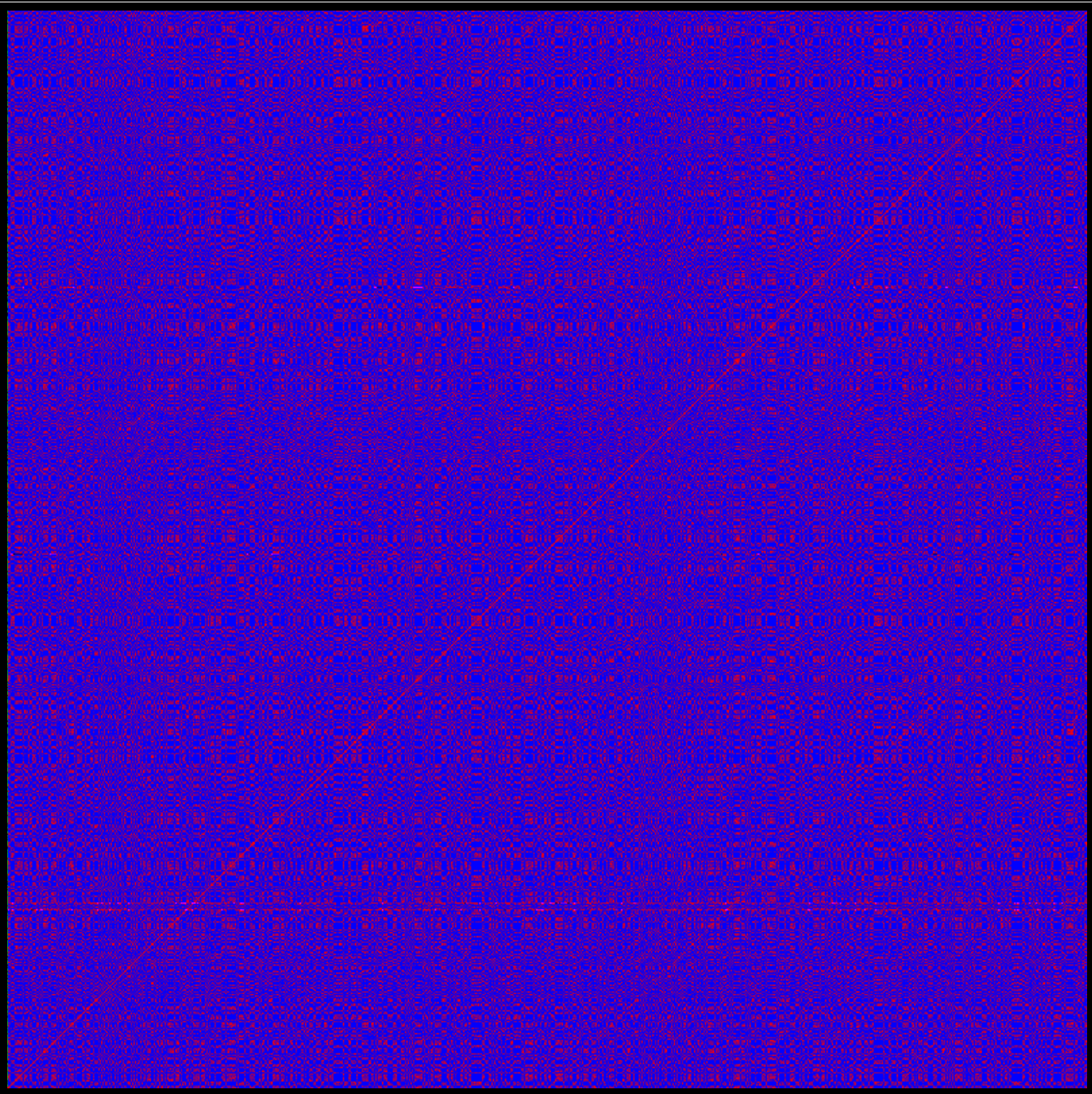
Evidence “Heat Map” - Ramnit



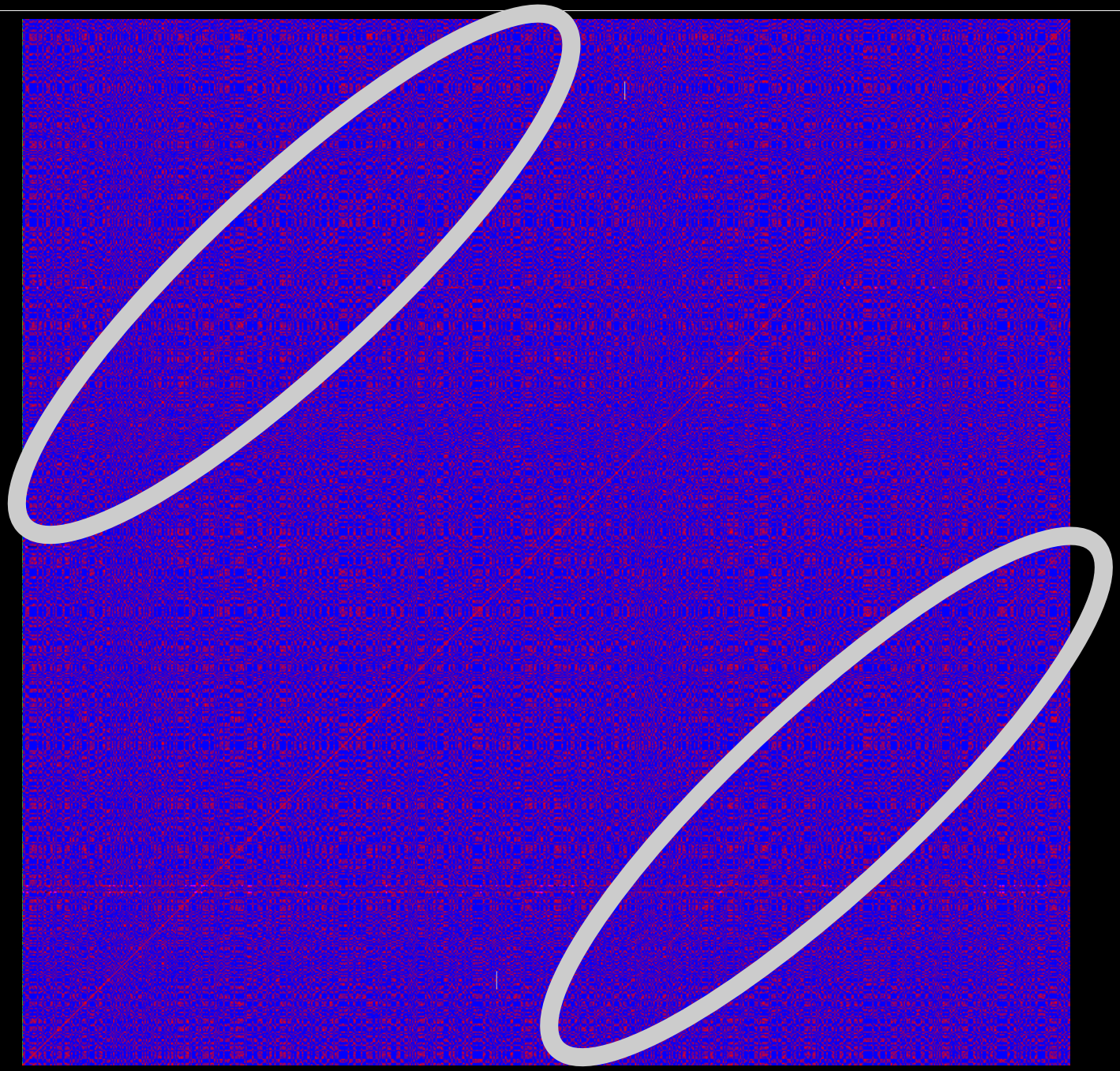
Evidence “Heat Map” - Ramnit



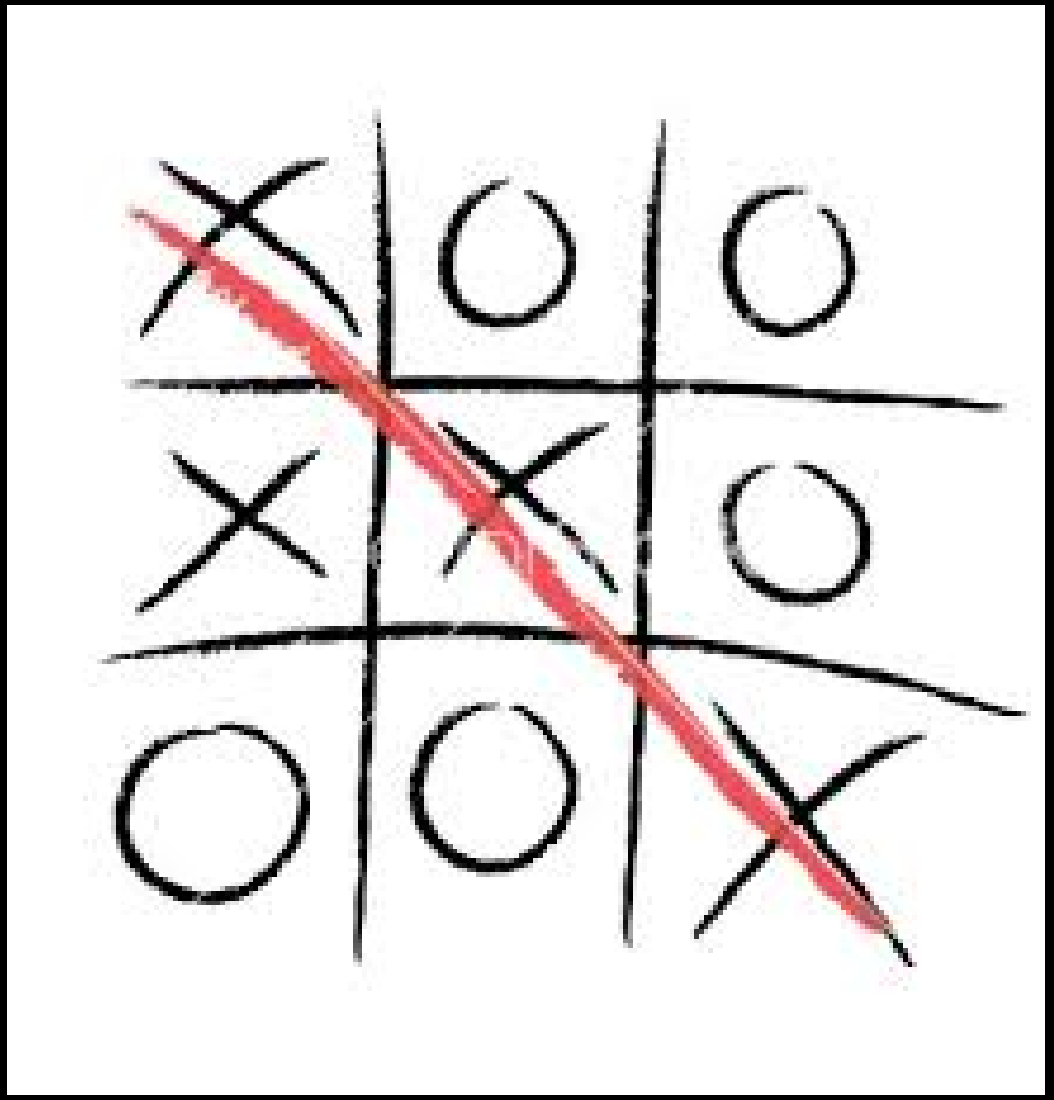
Evidence “Heat Map” - DirCrypt



Evidence “Heat Map” - DirCrypt



Success



Questions?