Chewing on the 0xDEADBEEF (redacted 26th March ‘08)

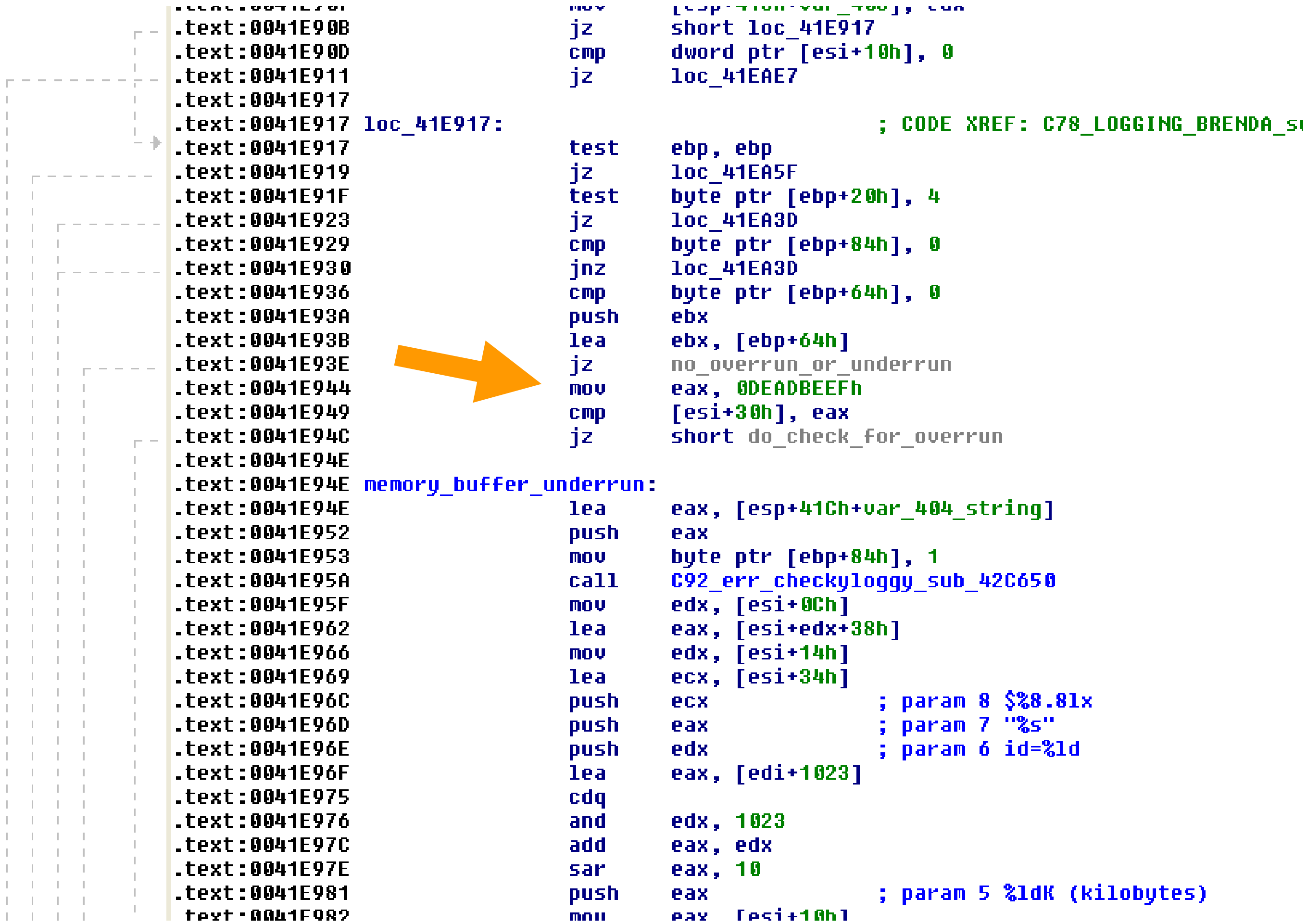
Mike Bond

10th August ‘07

Why redact this talk?

- Surely the crooks know this stuff already? Well maybe not, this is not crooks we are up against, just amateur hackers who want to cheat at games (little money in cheating at FPS yet), and its probably not illegal.
- True, Joint Ops is already suffering from cheats, and almost dead, but I don’t want to be putting the final nail in the coffin on what was an exceptionally good game in its time.
- Another 6 months or so and there should be no harm in releasing full detail of this talk.
- Some redaction in the screenshots was done to protect privacy of testers (and unwitting testers)
- **Any bona fide researchers in game cheating/network effects are welcome to take a copy of the full talk, plus source code etc, so long as they can satisfy me it will be put to good use. Email Mike.Bond@cl.cam.ac.uk, Phone +44 7890 171913**

The 0xDEADBEEF



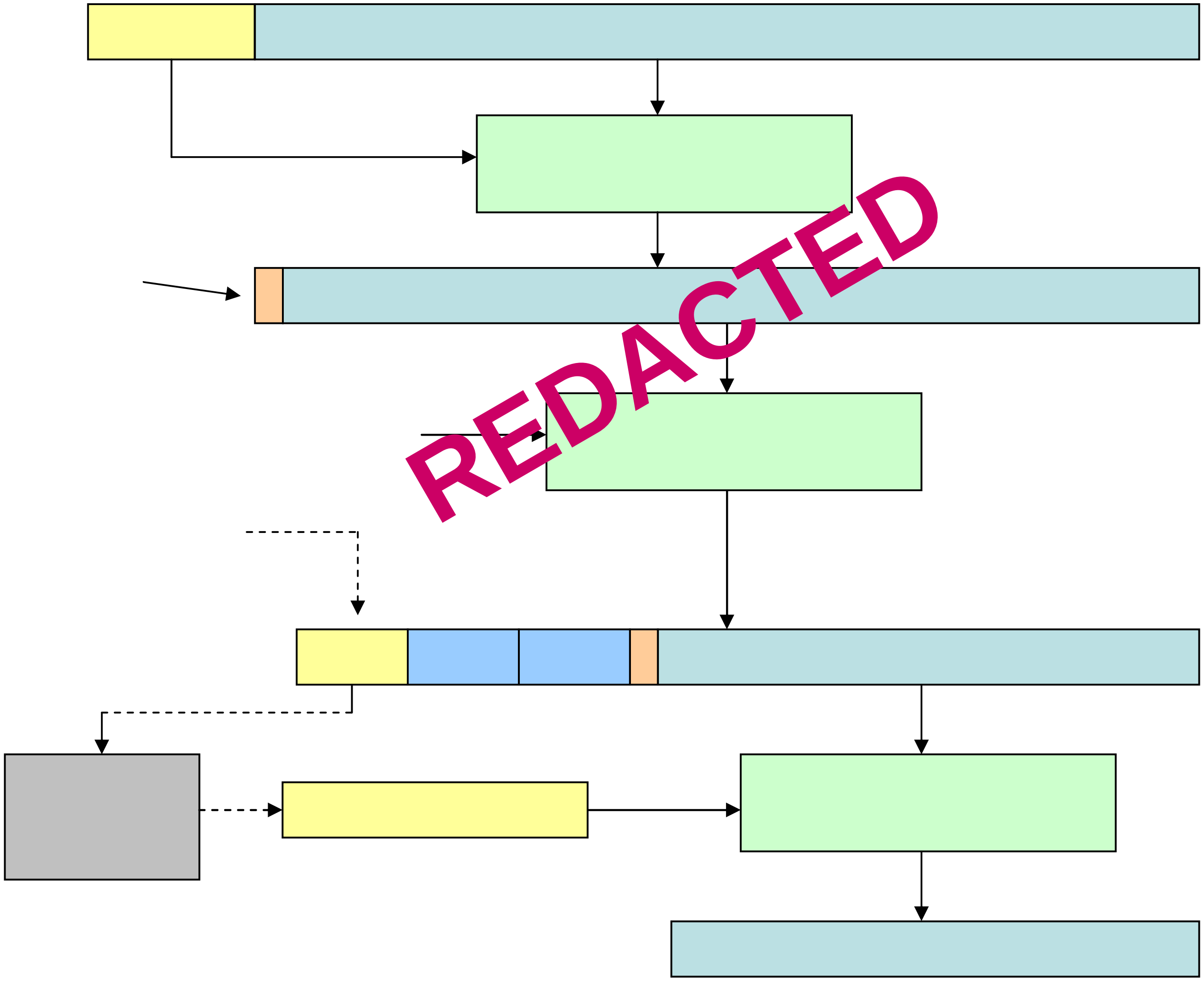
```
.text:0041E90B                 jz      short loc_41E917
.text:0041E90D                 cmp     dword ptr [esi+10h], 0
.text:0041E911                 jz      loc_41EAE7
.text:0041E917
.text:0041E917 loc_41E917:                             ; CODE XREF: C78_LOGGING_BRENDA_s
.text:0041E917                 test    ebp, ebp
.text:0041E919                 jz      loc_41EA5F
.text:0041E91F                 test    byte ptr [ebp+20h], 4
.text:0041E923                 jz      loc_41EA3D
.text:0041E929                 cmp     byte ptr [ebp+84h], 0
.text:0041E930                 jnz     loc_41EA3D
.text:0041E936                 cmp     byte ptr [ebp+64h], 0
.text:0041E93A                 push    ebx
.text:0041E93B                 lea     ebx, [ebp+64h]
.text:0041E93E                 jz      no_overrun_or_underrun
.text:0041E944                 mov     eax, 0DEADBEEFh
.text:0041E949                 cmp     [esi+30h], eax
.text:0041E94C                 jz      short do_check_for_overrun
.text:0041E94E
.text:0041E94E memory_buffer_underrun:
.text:0041E94E                 lea     eax, [esp+41Ch+var_404_string]
.text:0041E952                 push    eax
.text:0041E953                 mov     byte ptr [ebp+84h], 1
.text:0041E95A                 call    C92_err_checkyloggy_sub_42C650
.text:0041E95F                 mov     edx, [esi+0Ch]
.text:0041E962                 lea     eax, [esi+edx+38h]
.text:0041E966                 mov     edx, [esi+14h]
.text:0041E969                 lea     ecx, [esi+34h]
.text:0041E96C                 push    ecx             ; param 8 $%8.8lx
.text:0041E96D                 push    eax             ; param 7 "%s"
.text:0041E96E                 push    edx             ; param 6 id=%ld
.text:0041E96F                 lea     eax, [edi+1023]
.text:0041E975                 cdq
.text:0041E976                 and     edx, 1023
.text:0041E97C                 add     eax, edx
.text:0041E97E                 sar     eax, 10
.text:0041E981                 push    eax             ; param 5 %ldK (kilobytes)
.text:0041E982                 mov     eax, [esi+10h]
```

Contents

This work discusses reverse engineering and cryptanalysis of the encrypted data packets transmitted by an online multiplayer tactical shooter computer game “Joint Operations”.

- How it does the encryption
- How I cracked it
- Sturgeon’s Razor
- What if?
- Future Work

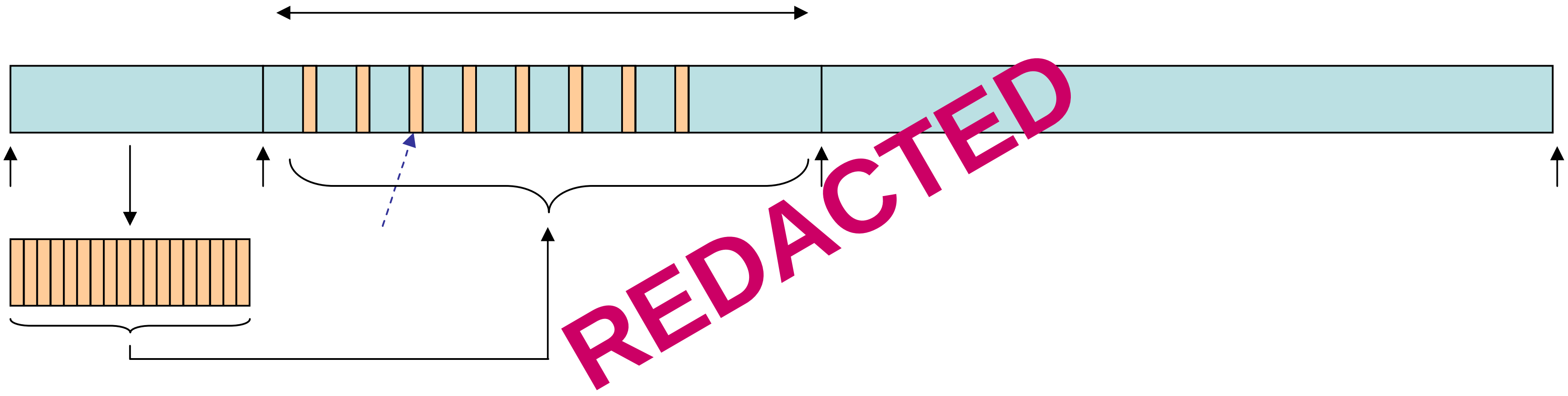
Packet “Encryption” Overview



REDACTED

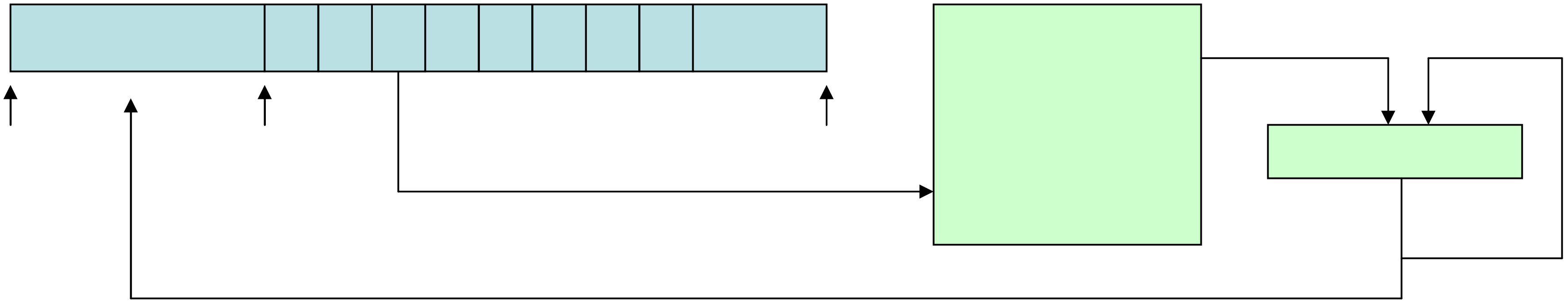
Henry Decrypt

Packets longer than X bytes encoded as follows:



REDACTED

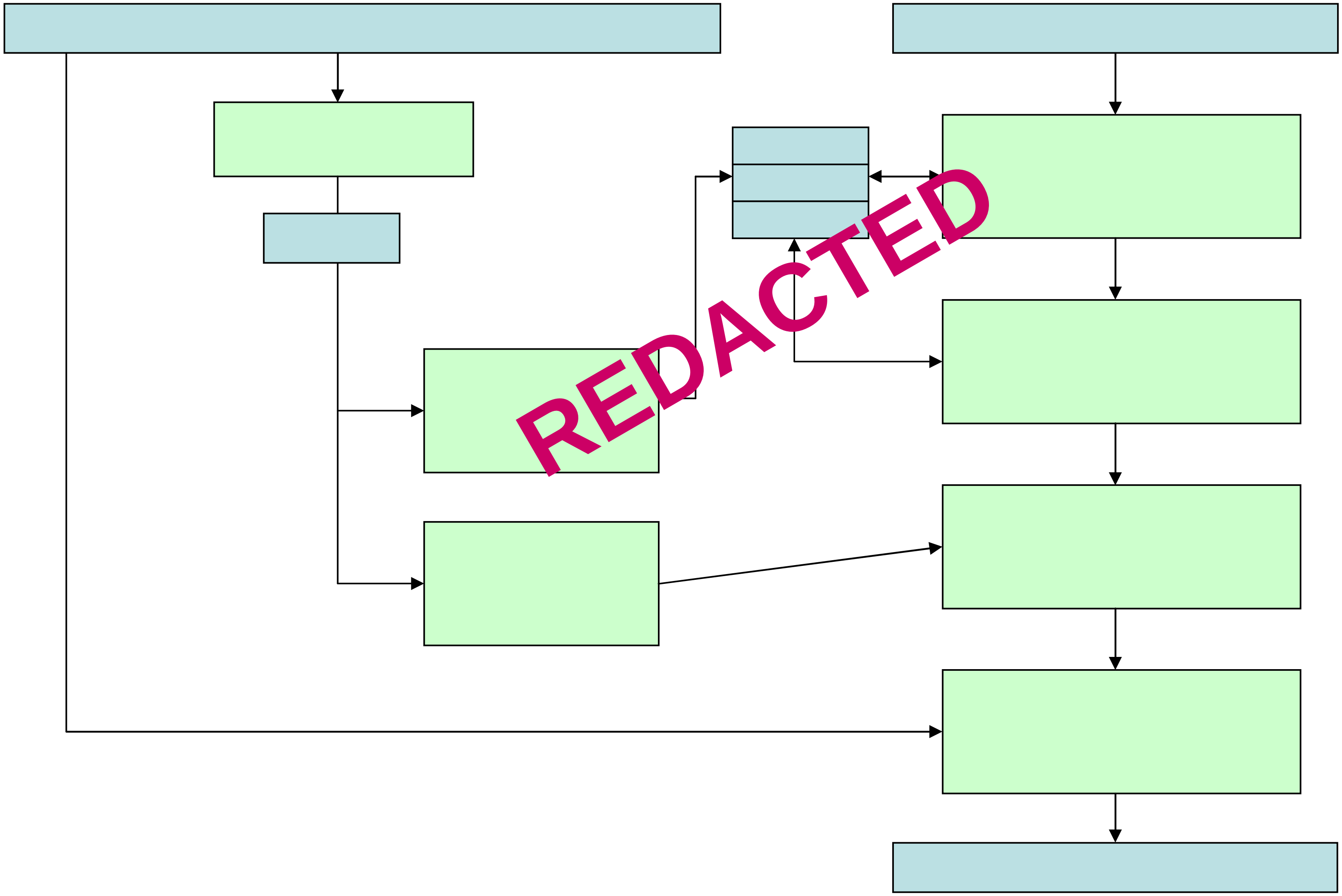
Packets less than X bytes with X



Main Decrypt Routine

- Takes null terminated array (usually ASCII string) as key input
- Always discards first byte without use
- Consists of key schedule derivation plus four base routines, all operating with byte-wise modulo addition
  - first quadratic equation
  - second quadratic equation
  - conditional string reverse (diffusion!)
  - Vigenère cipher

Main Decrypt Routine



decrypt_mask2

```
void decrypt_mask2(ref byte[] message, UInt32 len,ref UInt32[] keyschedule)
       {
       <snip>
       }
```

REDACTED

nb. code looks weird because it is trans-literated from dissasembly

decrypt_mask1

```
void decrypt_mask1(ref byte[] message, UInt32 len, UInt32 magic, UInt32 k3)
       {
       <snip>
       }
```

REDACTED

nb. code looks weird because it is trans-literated from dissasembly

decrypt_loopy

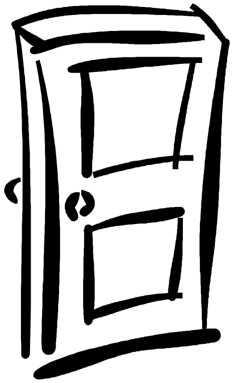
```
void decloopy(ref byte[] message, UInt32 len, byte[] key)
      {
      <snip>

      }
```

REDACTED

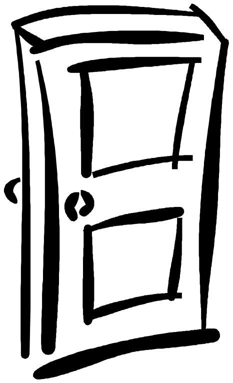
nb. code looks weird because it is trans-literated from dissasembly

Three Doors

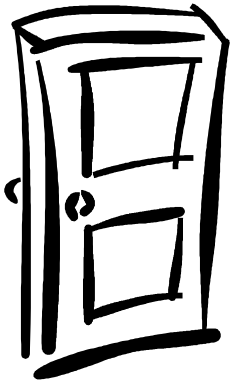
Behind one is a car, behind one freedom, and behind the other, certain death! Pick a door. Now I take away a door. Do you change your mind, or stick with your door?



Cryptanalysis



Static
Reverse
Engineering



Debugging

How I cracked it

1. intercepted packets using Ethereal: noted apparent encryption
2. created chosen chat messages, analysed by length of packet: detected individual packets corresponding to chat message
3. took differential between chosen plaintexts ‘aaaaaaaaaaa’ and ‘bbbbbbbbbb’, looking for evidence of stream cipher
4. stream cipher theory validated, began reverse engineering to locate stream cipher

How I cracked it (2)

5. no evidence of stream cipher from examining XOR calls
6. worked upwards from the UDP sendto system call, found static hard-coded keys at 6 layers up (later henry at 3 layers also)
7. from static keys located crypto, discovered stream combined using byte-level addition
8. reverse engineered crypto algorithms (but not their calling structure)

How I cracked it (3)

9. studied packet ciphertexts and differentials between packets looking for evidence of crypto algorithm identified
10. found good evidence, but also evidence of another algorithm
11. went back looking and found “henry”
12. implemented decryption of henry

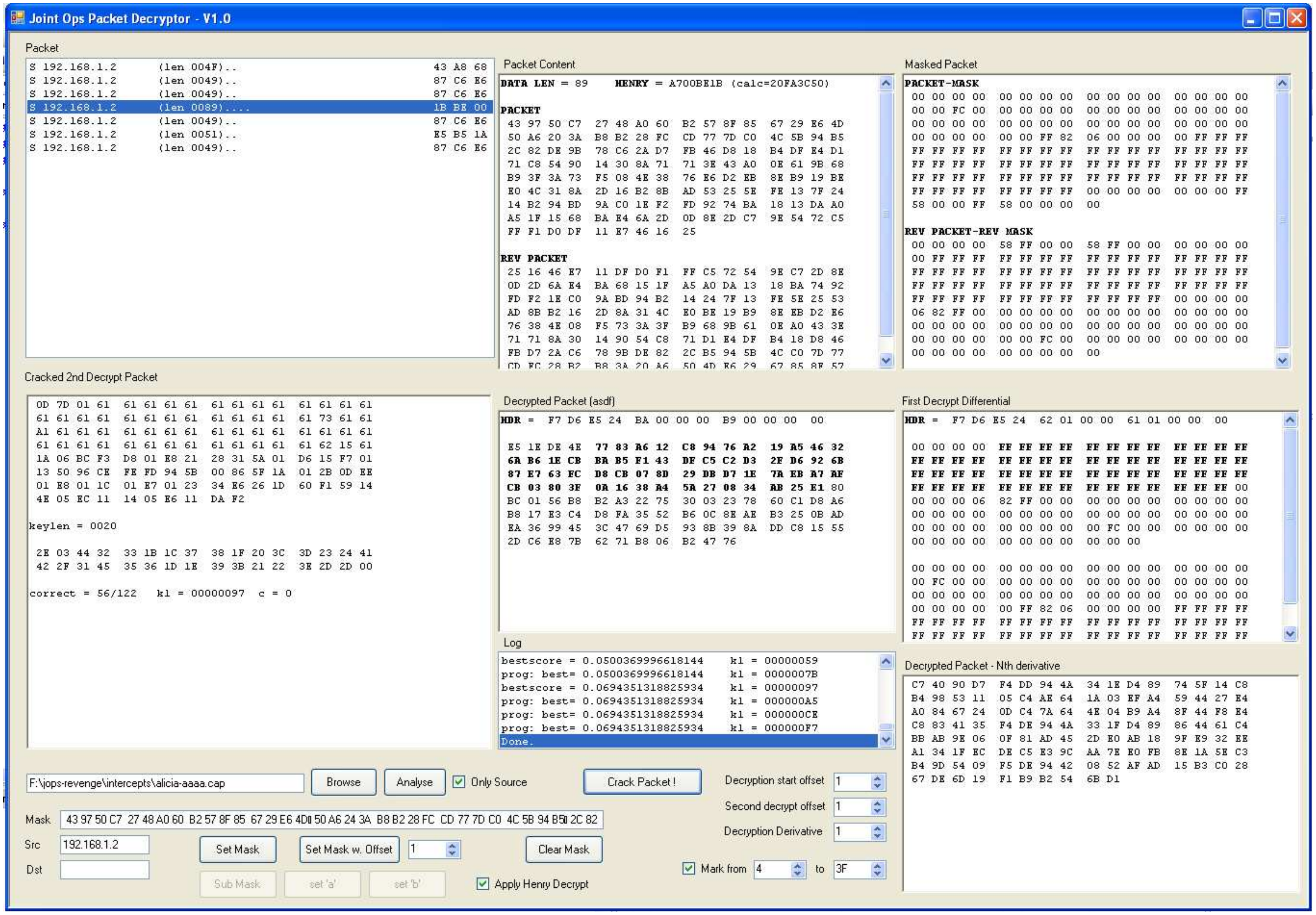
How I cracked it (4)

13. implemented decryption using static hard-coded key; by luck applying XXX yielded success.. a low entropy header
14. analysed packets looking for size and meaning of header
15. analysed differentials after first decrypt, looking for second decrypt
16. after much thought concluded second decrypt was indeed same algorithm, starting with XXX

How I cracked it (5)

17. Upgraded analysis tool to crack final key using chosen plaintext from chat messages.
18. Cracking algorithm uses brute-force to crack quadratic-equation based keys
19. calulcates optimal Vignere cipher key using a tuned fitness function

My Analysis Tool



Joint Ops Packet Decryptor - V1.0

Packet

```
S 192.168.1.2        (len 004F)..                        43 A8 68
S 192.168.1.2        (len 0049)..                        87 C6 E6
S 192.168.1.2        (len 0049)..                        87 C6 E6
S 192.168.1.2        (len 0089)....                      1B BE 00
S 192.168.1.2        (len 0049)..                        87 C6 E6
S 192.168.1.2        (len 0051)..                        E5 B5 1A
S 192.168.1.2        (len 0049)..                        87 C6 E6
```

Packet Content

```
DATA LEN = 89     HENRY = A700BE1B (calc=20FA3C50)

PACKET
 43 97 50 C7  27 48 A0 60  B2 57 8F 85  67 29 E6 4D
 50 A6 20 3A  B8 B2 28 FC  CD 77 7D C0  4C 5B 94 B5
 2C 82 DE 9B  78 C6 2A D7  FB 46 D8 18  B4 DF E4 D1
 71 C8 54 90  14 30 8A 71  71 3E 43 A0  0E 61 9B 68
 B9 3F 3A 73  F5 08 4E 38  76 E6 D2 EB  8E B9 19 BE
 E0 4C 31 8A  2D 16 B2 8B  AD 53 25 5E  FE 13 7F 24
 14 B2 94 BD  9A C0 1E F2  FD 92 74 BA  18 13 DA A0
 A5 1F 15 68  BA E4 6A 2D  0D 8E 2D C7  9E 54 72 C5
 FF F1 D0 DF  11 E7 46 16  25

REV PACKET
 25 16 46 E7  11 DF D0 F1  FF C5 72 54  9E C7 2D 8E
 0D 2D 6A E4  BA 68 15 1F  A5 A0 DA 13  18 BA 74 92
 FD F2 1E C0  9A BD 94 B2  14 24 7F 13  FE 5E 25 53
 AD 8B B2 16  2D 8A 31 4C  E0 BE 19 B9  8E EB D2 E6
 76 38 4E 08  F5 73 3A 3F  B9 68 9B 61  0E A0 43 3E
 71 71 8A 30  14 90 54 C8  71 D1 E4 DF  B4 18 D8 46
 FB D7 2A C6  78 9B DE 82  2C B5 94 5B  4C C0 7D 77
 CD FC 28 B2  B8 3A 20 A6  50 4D E6 29  67 85 8F 57
```

Masked Packet

```
PACKET-MASK
 00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
 00 00 FC 00  00 00 00 00  00 00 00 00  00 00 00 00
 00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
 00 00 00 00  00 00 FF 82  06 00 00 00  00 FF FF FF
 FF FF FF FF  FF FF FF FF  FF FF FF FF  FF FF FF FF
 FF FF FF FF  FF FF FF FF  FF FF FF FF  FF FF FF FF
 FF FF FF FF  FF FF FF FF  FF FF FF FF  FF FF FF FF
 FF FF FF FF  FF FF FF FF  00 00 00 00  00 00 00 FF
 58 00 00 FF  58 00 00 00  00

REV PACKET-REV MASK
 00 00 00 00  58 FF 00 00  58 FF 00 00  00 00 00 00
 00 FF FF FF  FF FF FF FF  FF FF FF FF  FF FF FF FF
 FF FF FF FF  FF FF FF FF  FF FF FF FF  FF FF FF FF
 FF FF FF FF  FF FF FF FF  FF FF FF FF  FF FF FF FF
 FF FF FF FF  FF FF FF FF  FF FF FF FF  00 00 00 00
 06 82 FF 00  00 00 00 00  00 00 00 00  00 00 00 00
 00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
 00 00 00 00  00 00 FC 00  00 00 00 00  00 00 00 00
 00 00 00 00  00 00 00 00  00
```

Cracked 2nd Decrypt Packet

```
 0D 7D 01 61  61 61 61 61  61 61 61 61  61 61 61 61
 61 61 61 61  61 61 61 61  61 61 61 61  61 73 61 61
 A1 61 61 61  61 61 61 61  61 61 61 61  61 61 61 61
 61 61 61 61  61 61 61 61  61 61 61 61  61 62 15 61
 1A 06 BC F3  D8 01 E8 21  28 31 5A 01  D6 15 F7 01
 13 50 96 CE  FE FD 94 5B  00 86 5F 1A  01 2B 0D EE
 01 E8 01 1C  01 E7 01 23  34 E6 26 1D  60 F1 59 14
 4E 05 EC 11  14 05 E6 11  DA F2

keylen = 0020

 2E 03 44 32  33 1B 1C 37  38 1F 20 3C  3D 23 24 41
 42 2F 31 45  35 36 1D 1E  39 3B 21 22  3E 2D 2D 00

correct = 56/122   kl = 00000097  c = 0
```

Decrypted Packet (asdf)

```
HDR =  F7 D6 E5 24  BA 00 00 00  B9 00 00 00  00

 E5 1E DE 4E  77 83 A6 12  C8 94 76 A2  19 A5 46 32
 6A B6 1E CB  BA B5 F1 43  DF C5 C2 D3  2F D6 92 6B
 87 E7 63 FC  D8 CB 07 8D  29 DB D7 1E  7A EB A7 AF
 CB 03 80 3F  0A 16 38 A4  5A 27 08 34  AB 25 E1 80
 BC 01 56 B8  B2 A3 22 75  30 03 23 78  60 C1 D8 A6
 B8 17 E3 C4  D8 FA 35 52  B6 0C 8E AE  B3 25 0B AD
 EA 36 99 45  3C 47 69 D5  93 8B 39 8A  DD C8 15 55
 2D C6 E8 7B  62 71 B8 06  B2 47 76
```

First Decrypt Differential

```
HDR =  F7 D6 E5 24  62 01 00 00  61 01 00 00  00

 00 00 00 00  FF FF FF FF  FF FF FF FF  FF FF FF FF
 FF FF FF FF  FF FF FF FF  FF FF FF FF  FF FF FF FF
 FF FF FF FF  FF FF FF FF  FF FF FF FF  FF FF FF FF
 FF FF FF FF  FF FF FF FF  FF FF FF FF  FF FF FF 00
 00 00 00 06  82 FF 00 00  00 00 00 00  00 00 00 00
 00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
 00 00 00 00  00 00 00 00  00 FC 00 00  00 00 00 00
 00 00 00 00  00 00 00 00  00 00 00

 00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
 00 FC 00 00  00 00 00 00  00 00 00 00  00 00 00 00
 00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
 00 00 00 00  00 FF 82 06  00 00 00 00  FF FF FF FF
 FF FF FF FF  FF FF FF FF  FF FF FF FF  FF FF FF FF
 FF FF FF FF  FF FF FF FF  FF FF FF FF  FF FF FF FF
```

Log

```
bestscore = 0.0500369996618144    kl = 00000059
prog: best= 0.0500369996618144    kl = 0000007B
bestscore = 0.0694351318825934    kl = 00000097
prog: best= 0.0694351318825934    kl = 000000A5
prog: best= 0.0694351318825934    kl = 000000CE
prog: best= 0.0694351318825934    kl = 000000F7
Done.
```

Decrypted Packet - Nth derivative

```
 C7 40 90 D7  F4 DD 94 4A  34 1E D4 89  74 5F 14 C8
 B4 98 53 11  05 C4 AE 64  1A 03 EF A4  59 44 27 E4
 A0 84 67 24  0D C4 7A 64  4E 04 B9 A4  8F 44 F8 E4
 C8 83 41 35  F4 DE 94 4A  33 1F D4 89  86 44 61 C4
 BB AB 9E 06  0F 81 AD 45  2D E0 AB 18  9F E9 32 EE
 A1 34 1F EC  DE C5 E3 9C  AA 7E E0 FB  8E 1A 5E C3
 B4 9D 54 09  F5 DE 94 42  08 52 AF AD  15 B3 C0 28
 67 DE 6D 19  F1 B9 B2 54  6B D1
```

F:\jops-revenge\intercepts\alicia-aaaa.cap

☑ Only Source

Mask: 43 97 50 C7 27 48 A0 60 B2 57 8F 85 67 29 E6 4D▯ 50 A6 24 3A B8 B2 28 FC CD 77 7D C0 4C 5B 94 B5▯ 2C 82

Src: 192.168.1.2

Dst:

Set Mask w. Offset: 1

☑ Apply Henry Decrypt

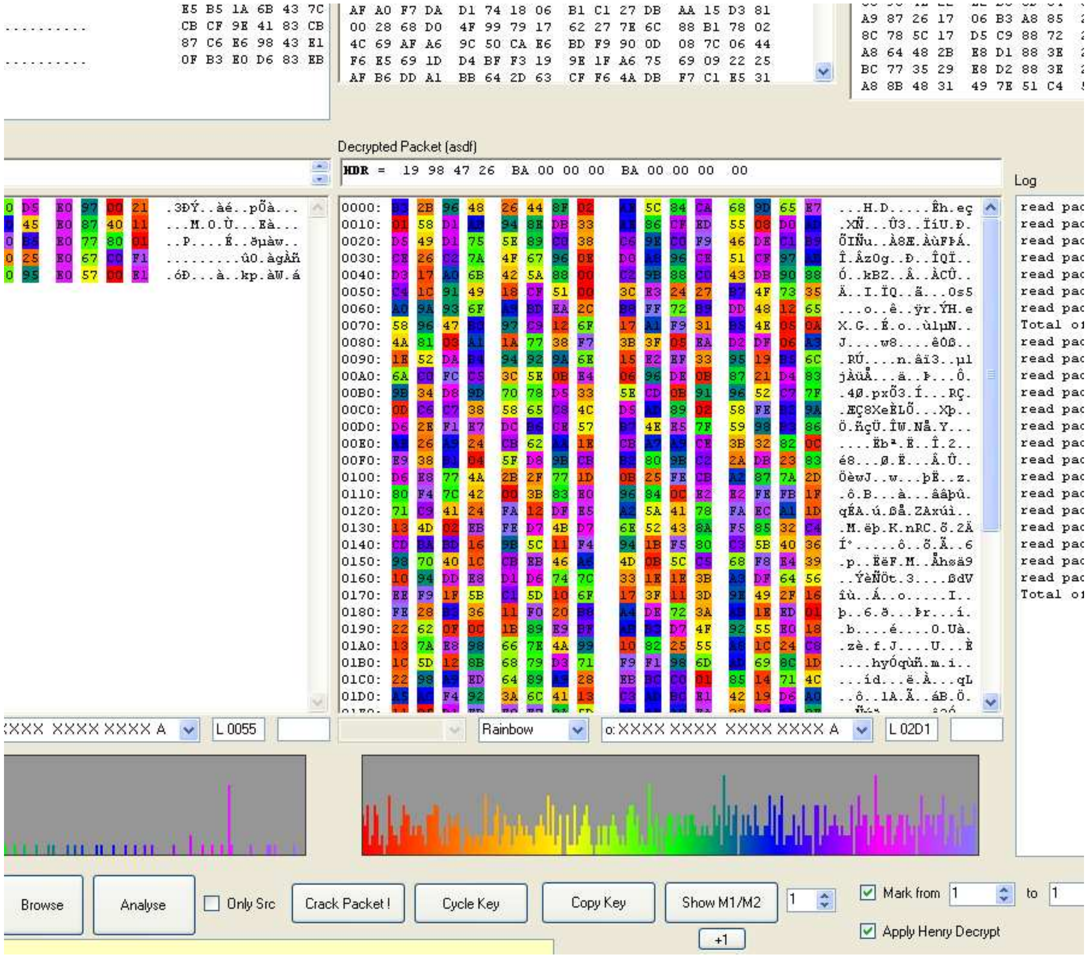
Decryption start offset: 1

Second decrypt offset: 1

Decryption Derivative: 1

☑ Mark from 4 to 3F

My Analysis Tool (2)



```
          E5 B5 1A 6B 43 7C
.......... CB CF 9E 41 83 CB
          87 C6 E6 98 43 E1
.......... 0F B3 E0 D6 83 EB
```

```
AF A0 F7 DA  D1 74 18 06  B1 C1 27 DB  AA 15 D3 81
00 28 68 D0  4F 99 79 17  62 27 7E 6C  88 B1 78 02
4C 69 AF A6  9C 50 CA E6  BD F9 90 0D  08 7C 06 44
F6 E5 69 1D  D4 BF F3 19  9E 1F A6 75  69 09 22 25
AF B6 DD A1  BB 64 2D 63  CF F6 4A DB  F7 C1 E5 31
```

```
A9 87 26 17  06 B3 A8 85
8C 78 5C 17  D5 C9 88 72
A8 64 48 2B  E8 D1 88 3E
BC 77 35 29  E8 D2 88 3E
A8 8B 48 31  49 7E 51 C4
```

```
0 D5  E0 97 00 21   .3ÐÝ..àé..pÕà...
  45  E0 87 40 11   ...M.0.Ù...Eà...
0     E0 77 80 01   ..P....É..ðµàw..
0 25  E0 67 C0 F1   ...........û0.àgÀñ
0 95  E0 57 00 E1   .óÐ...à..kp.àW.á
```

Decrypted Packet (asdf)

```
HDR =  19 98 47 26  BA 00 00 00  BA 00 00 00  00
```

```
0000: [illegible] 2B 96 48  26 44 8F 02  [illegible] 5C 84 CA  68 9D 65 E7   ...H.D.....Êh.eç
0010: 01 58 D1 [illegible]  94 8E DB 33  [illegible] 86 CF ED  55 08 D0 [illegible]   .XÑ...Û3..ÏíU.Ð.
0020: D5 49 D1 75  5E 89 C0 38  C6 9E C0 F9  46 DE C1 B9   ÕIÑu..À8Æ.ÀùFÞÁ.
0030: CE 26 C2 7A  4F 67 96 0E  D0 A8 96 CE  51 CF 97 AD   Î.ÂzOg..Ð..ÎQÏ..
0040: D3 17 A0 6B  42 5A 88 00  C2 9B 88 C0  43 DB 90 88   Ó..kBZ..Â..ÀCÛ..
0050: C4 1C 91 49  18 CF 51 00  3C E3 24 27  [illegible] 4F 73 35   Ä..I.ÏQ..ã...Os5
0060: A0 9A 93 6F  [illegible] 8D EA 2C  [illegible] FF 72 [illegible]  DD 48 12 65   ...o..ê..ÿr.ÝH.e
0070: 58 96 47 [illegible]  97 C9 12 6F  17 A1 F9 31  B5 4E 05 0A   X.G..É.o..ù1µN..
0080: 4A 81 03 A1  1A 77 38 F7  3B 3F 05 EA  D2 DF 06 A3   J....w8....êÒß..
0090: 1E 52 DA [illegible]  94 92 9A 6E  15 E2 EF 33  95 19 B5 6C   .RÚ....n.âï3..µl
00A0: 6A C0 FC C5  3C 5E 0B E4  06 96 DE 0B  87 21 D4 83   jÀüÅ...ä..Þ...Ô.
00B0: 9B 34 D8 9D  70 78 D5 33  5E CD 0B 91  96 52 C7 7F   .4Ø.pxÕ3.Í...RÇ.
00C0: 0D C6 C7 38  58 65 C8 4C  D5 [illegible] 89 02  58 FE [illegible] 9A   .ÆÇ8XeÈLÕ...Xþ..
00D0: D6 2E F1 E7  DC B6 CE 57  [illegible] 4E E5 7F  59 98 [illegible] 86   Ö.ñçÜ.ÎW.Nå.Y...
00E0: [illegible] 26 A9 24  CB 62 AA 1E  CB A7 A9 CE  3B 32 82 0C   ....Ëbª.Ë..Î.2..
00F0: E9 38 [illegible] 04  5F D8 9B CB  [illegible] 80 9B C2  2A DB 23 83   é8...Ø.Ë...Â.Û..
0100: D6 E8 77 4A  2B 2F 77 1D  0B 25 FE CB  A2 87 7A 2D   ÖèwJ..w...þË..z.
0110: 80 F4 7C 42  00 3B 83 E0  96 84 0C E2  E2 FE FB 1F   .ô.B...à...ââþû.
0120: 71 C9 41 24  FA 12 DF E5  A2 5A 41 78  FA EC A1 1D   qÉA.ú.ßå.ZAxúì..
0130: 13 4D 02 EB  FE D7 4B D7  6E 52 43 8A  F5 85 32 C4   .M.ëþ.K.nRC.õ.2Ä
0140: CD BA BD 16  9B 5C 11 F4  94 1B F5 80  C3 5B 40 36   Íº.....ô..õ.Ã..6
0150: 98 70 40 1C  CB EB 46 A5  4D 0B 5C C5  68 F8 E4 39   .p..ËëF.M..Åhøä9
0160: 10 94 DD E8  D1 D6 74 7C  33 1E 1E 3B  A3 DF 64 56   ..ÝèÑÖt.3....ßdV
0170: EE F9 1F 5B  C1 5D 10 6F  17 3F 11 3D  9E 49 2F 16   îù..Á..o.....I..
0180: FE 28 B3 36  11 F0 20 B8  A4 DE 72 3A  A8 1E ED 01   þ..6.ð...Þr...í.
0190: 22 62 0F 0C  1B 89 E9 BF  A8 B3 D7 4F  92 55 E0 18   .b....é....O.Uà.
01A0: 13 7A E8 98  66 7E 4A 99  10 82 25 55  A8 1C 24 C8   .zè.f.J....U...È
01B0: 1C 5D 12 8B  68 79 D3 71  F9 F1 98 6D  AD 69 8C 1D   ....hyÓqùñ.m.i..
01C0: 22 98 A9 ED  64 89 A9 28  EB BC C0 01  85 14 71 4C   ...íd...ë.À...qL
01D0: A5 AC F4 92  3A 6C 41 13  C3 A8 BC E1  42 19 D6 A0   ..ô..lA.Ã..áB.Ö.
```

Log

```
read pac
read pac
read pac
read pac
read pac
read pac
read pac
Total of
read pac
read pac
read pac
read pac
read pac
read pac
read pac
read pac
read pac
read pac
read pac
read pac
read pac
read pac
read pac
Total of
```

XXXX XXXX XXXX A | L 0055 | Rainbow | o: XXXX XXXX XXXX XXXX A | L 02D1

What if?

- Suppose they’d used stronger key stream generator based on proper crypto algorithm (e.g. 3DES)?
  – Easier. Only need to reverse engineer to identify algorithm, not to re-implement
  – Easier. Crypto with proper diffusion characteristics makes it easier to determine when you have got it right. Still stuck with 95% accurate crypto reimplementations which occasionally get a byte wrong
  – Easier to locate in disassembly. Look for crypto-code characteristics
- Supose they changed key every packet instead of every session?
  – Much easier to exhaust key space of weak cipher by sending a repeated message under many different keys

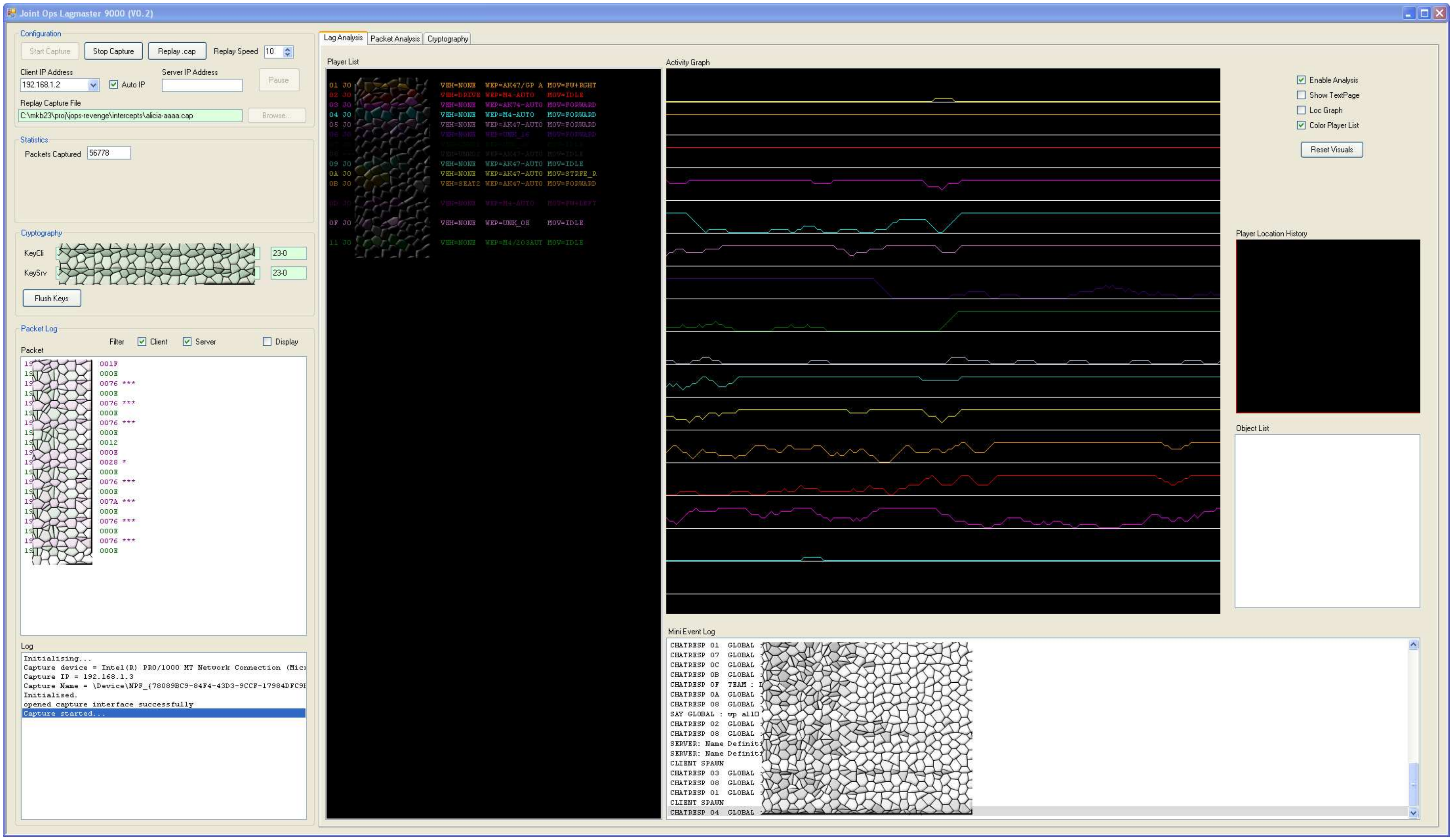
Sturgeon’s Razor

- My previous reverse-engineering rules
  - do what you can
  - give everything a name
- Occam and Sturgeon together…
  - “90% of everything is crap”
  - “All things being equal, the simplest explanation tends to be the best one."
- Yields the new rules (used when explaining weird function behaviour) :
  - “the simplest explanation is that its just a load of crap”

Future Work

- This work just a pre-requisite to experiments on “Neo-Tactics”
- I want to find out how the update rates of players in-game vary. You have 150 players in game, but you only have 600 bytes of data in your packet. What do you send?
  - Does perception of unfairness/cheating correspond to real bandwidth problems, or inadequacies/anomalies in *use of available bandwidth*?
- This work will also enable (study of) many sorts of undetectable cheating based on packet interception/analysis/rewriting
  - What sorts of hack are achievable if you can mess with the packets in tactical shooters? I hope to make a taxonomy.

Working Lag Analysis Tool



Joint Ops Lagmaster 9000 (V0.2)

Configuration

Replay Speed: 10

Client IP Address: 192.168.1.2 ☑ Auto IP

Server IP Address:

Replay Capture File: C:\mkb23\proj\jops-revenge\intercepts\alicia-aaaa.cap

Statistics

Packets Captured: 56778

Cryptography

KeyCli: [redacted] 23-0

KeySrv: [redacted] 23-0

Packet Log

Filter ☑ Client ☑ Server □ Display

Packet

```
19[redacted] 001F
19 000E
19 0076 ***
19 000E
19 0076 ***
19 000E
19 0076 ***
19 000E
19 0012
19 000E
19 0028 *
19 000E
19 0076 ***
19 000E
19 007A ***
19 000E
19 0076 ***
19 000E
19 0076 ***
19 000E
```

Log

```
Initialising...
Capture device = Intel(R) PRO/1000 MT Network Connection (Mic
Capture IP = 192.168.1.3
Capture Name = \Device\NPF_{78089BC9-84F4-43D3-9CCF-17984DFC9
Initialised.
opened capture interface successfully
Capture started...
```

Player List

```
01 JO [redacted]  VEH=NONE  WEP=AK47/GP A MOV=FW+RGHT
02 JO             VEH=DRIVE WEP=M4-AUTO   MOV=IDLE
03 JO             VEH=NONE  WEP=AK74-AUTO MOV=FORWARD
04 JO             VEH=NONE  WEP=M4-AUTO   MOV=FORWARD
05 JO             VEH=NONE  WEP=AK47-AUTO MOV=FORWARD
06 JO             VEH=NONE  WEP=UNK_16    MOV=FORWARD

08 --             VEH=UNKOI WEP=AK47-AUTO MOV=IDLE
09 JO             VEH=NONE  WEP=AK47-AUTO MOV=IDLE
0A JO             VEH=NONE  WEP=AK47-AUTO MOV=STRFE_R
0B JO             VEH=SEAT2 WEP=AK47-AUTO MOV=FORWARD

0D JO             VEH=NONE  WEP=M4-AUTO   MOV=FW+LEFT

0F JO             VEH=NONE  WEP=UNK_0E    MOV=IDLE

11 JO             VEH=NONE  WEP=M4/203AUT MOV=IDLE
```

Activity Graph

☑ Enable Analysis
□ Show TextPage
□ Loc Graph
☑ Color Player List

Player Location History

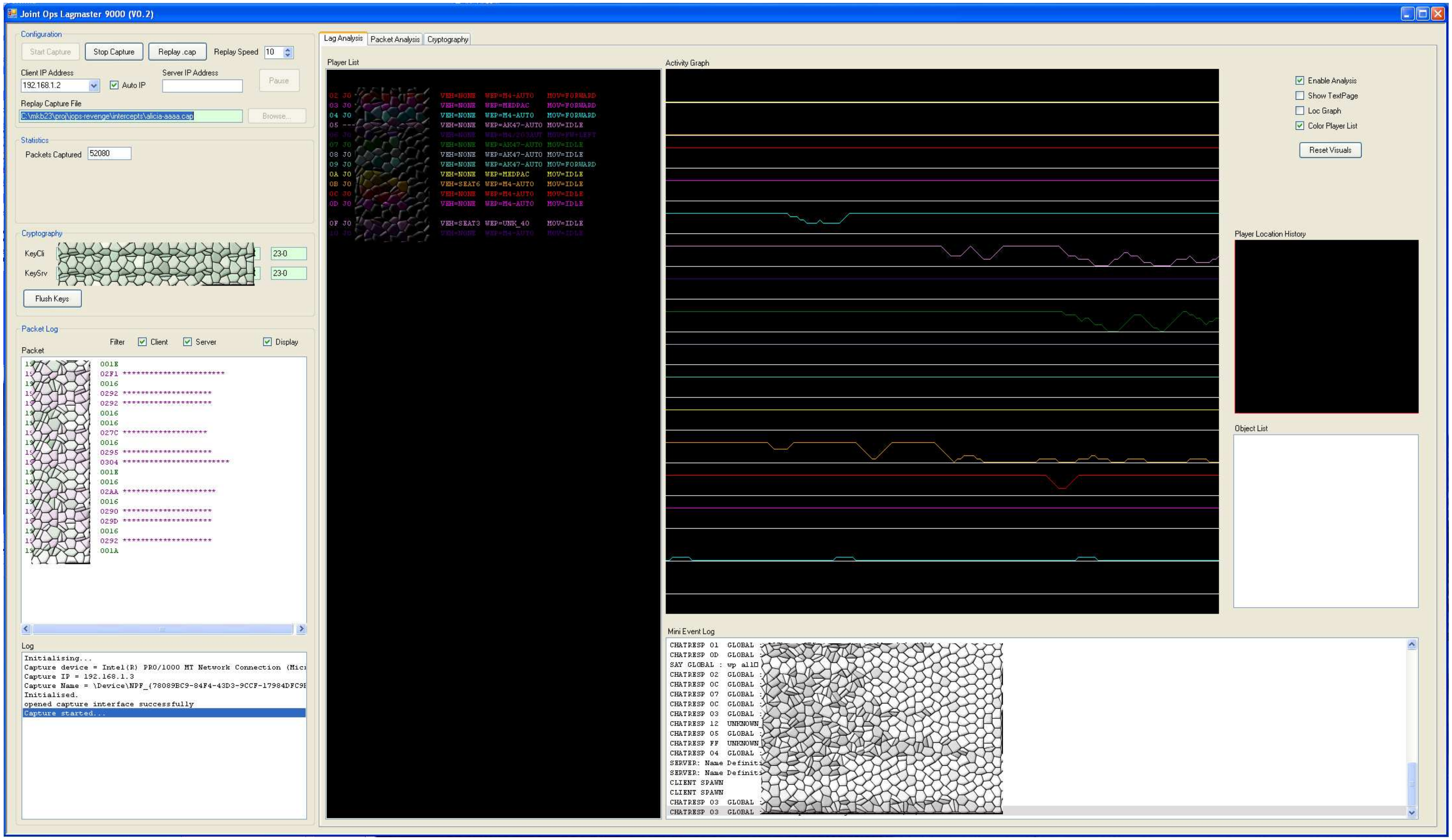
Object List

Mini Event Log

```
CHATRESP 01  GLOBAL :[redacted]
CHATRESP 07  GLOBAL :
CHATRESP 0C  GLOBAL :
CHATRESP 0B  GLOBAL :
CHATRESP 0F  TEAM : L
CHATRESP 0A  GLOBAL :
CHATRESP 08  GLOBAL :
SAY GLOBAL : wp all□
CHATRESP 02  GLOBAL :
CHATRESP 08  GLOBAL :
SERVER: Name Definiti
SERVER: Name Definiti
CLIENT SPAWN
CHATRESP 03  GLOBAL :
CHATRESP 08  GLOBAL :
CHATRESP 01  GLOBAL :
CLIENT SPAWN
CHATRESP 04  GLOBAL :
```

Hi res picture: see http://www.cl.cam.ac.uk/~mkb23/jopsdec/lag1-censor.png

Working Lag Analysis Tool (2)



Joint Ops Lagmaster 9000 (V0.2)

Configuration

Start Capture | Stop Capture | Replay .cap | Replay Speed 10

Client IP Address 192.168.1.2 ☑ Auto IP | Server IP Address | Pause

Replay Capture File C:\mkb23\proj\jops-revenge\intercepts\alicia-aaaa.cap | Browse...

Statistics

Packets Captured 52080

Cryptography

KeyCli [redacted] 23-0

KeySrv 23-0

Flush Keys

Packet Log

Filter ☑ Client ☑ Server ☑ Display

Packet

```
19 [redacted]  001E
19   02F1 ************************
19   0016
19   0292 ********************
19   0292 ********************
19   0016
19   0016
19   027C *******************
19   0016
19   0295 ********************
19   0304 ************************
19   001E
19   0016
19   02AA *********************
19   0016
19   0290 ********************
19   029D ********************
19   0016
19   0292 ********************
19   001A
```

Log

```
Initialising...
Capture device = Intel(R) PRO/1000 MT Network Connection (Mic
Capture IP = 192.168.1.3
Capture Name = \Device\NPF_{78089BC9-84F4-43D3-9CCF-17984DFC9
Initialised.
opened capture interface successfully
Capture started...
```

Lag Analysis | Packet Analysis | Cryptography

Player List

```
02 JO [redacted]  VEH=NONE   WEP=M4-AUTO    MOV=FORWARD
03 JO   VEH=NONE   WEP=MEDPAC     MOV=FORWARD
04 JO   VEH=NONE   WEP=M4-AUTO    MOV=FORWARD
05 ---  VEH=NONE   WEP=AK47-AUTO  MOV=IDLE
06 JO   VEH=NONE   WEP=M4/203AUT  MOV=FW+LEFT
07 JO   VEH=NONE   WEP=AK47-AUTO  MOV=IDLE
08 JO   VEH=NONE   WEP=AK47-AUTO  MOV=IDLE
09 JO   VEH=NONE   WEP=AK47-AUTO  MOV=FORWARD
0A JO   VEH=NONE   WEP=MEDPAC     MOV=IDLE
0B JO   VEH=SEAT6  WEP=M4-AUTO    MOV=IDLE
0C JO   VEH=NONE   WEP=M4-AUTO    MOV=IDLE
0D JO   VEH=NONE   WEP=M4-AUTO    MOV=IDLE

0F JO   VEH=SEAT3  WEP=UNK_40     MOV=IDLE
10 JO   VEH=NONE   WEP=M4-AUTO    MOV=IDLE
```

Activity Graph

☑ Enable Analysis
☐ Show TextPage
☐ Loc Graph
☑ Color Player List

Reset Visuals

Player Location History

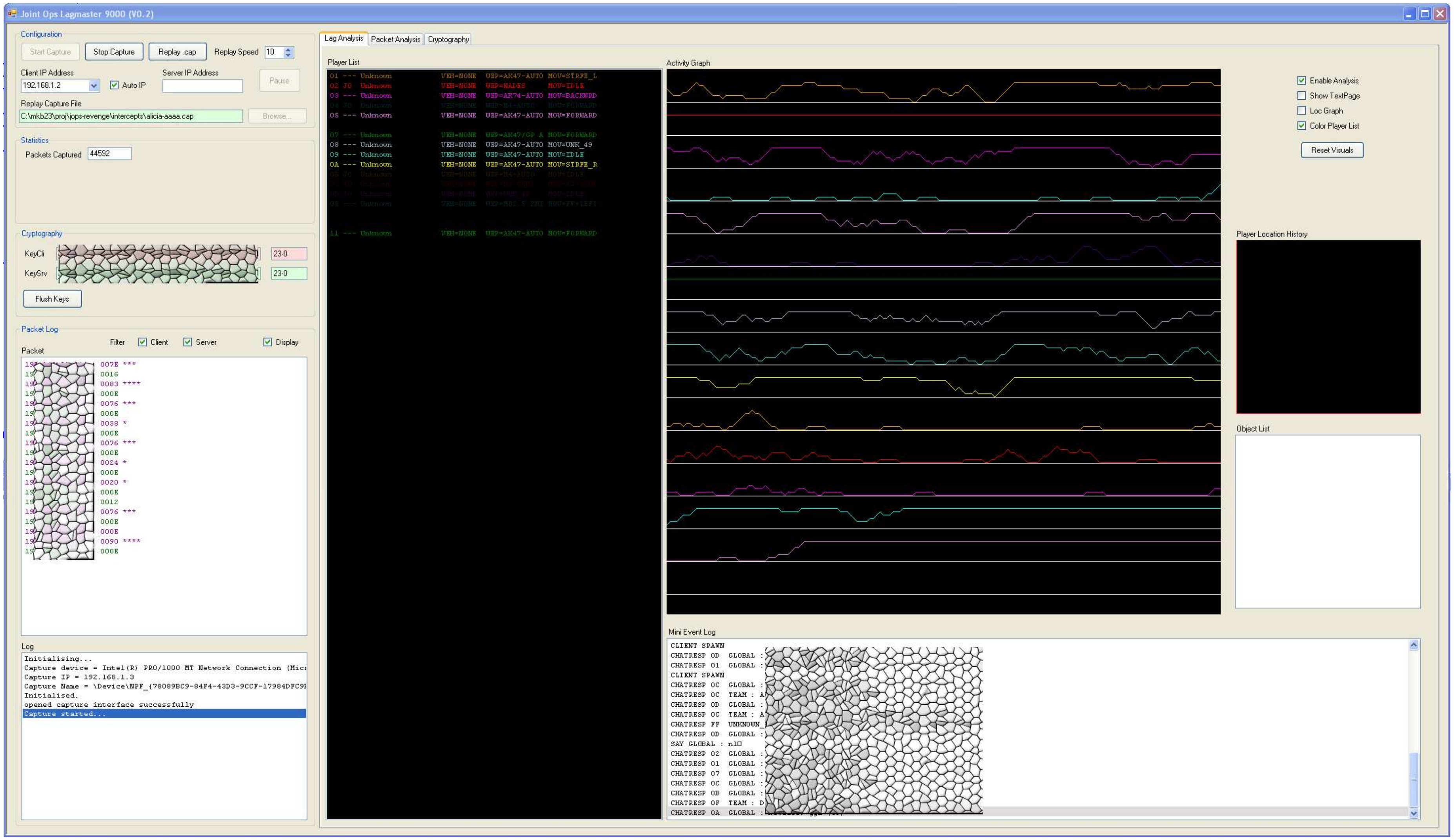
Object List

Mini Event Log

```
CHATRESP 01  GLOBAL : [redacted]
CHATRESP 0D  GLOBAL :
SAY GLOBAL : wp all□
CHATRESP 02  GLOBAL :
CHATRESP 0C  GLOBAL :
CHATRESP 07  GLOBAL :
CHATRESP 0C  GLOBAL :
CHATRESP 03  GLOBAL :
CHATRESP 12  UNKNOWN_
CHATRESP 05  GLOBAL :
CHATRESP FF  UNKNOWN_
CHATRESP 04  GLOBAL :
SERVER: Name Definiti
SERVER: Name Definiti
CLIENT SPAWN
CLIENT SPAWN
CHATRESP 03  GLOBAL :
CHATRESP 03  GLOBAL :
```

Hi res picture: see http://www.cl.cam.ac.uk/~mkb23/jopsdec/lag2-censor.png

Working Lag Analysis Tool (3)



Replay Speed 10

Client IP Address 192.168.1.2 ☑ Auto IP

Server IP Address

Replay Capture File C:\mkb23\proj\jops-revenge\intercepts\alicia-aaaa.cap

Packets Captured 44592

KeyCli [redacted] 23-0

KeySrv [redacted] 23-0

Filter ☑ Client ☑ Server ☑ Display

Packet

```
19[redacted] 007E ***
19           0016
19           0083 ****
19           000E
19           0076 ***
19           000E
19           0038 *
19           000E
19           0076 ***
19           000E
19           0024 *
19           000E
19           0020 *
19           000E
19           0012
19           0076 ***
19           000E
19           000E
19           0090 ****
19           000E
```

Log

```
Initialising...
Capture device = Intel(R) PRO/1000 MT Network Connection (Mic
Capture IP = 192.168.1.3
Capture Name = \Device\NPF_{78089BC9-84F4-43D3-9CCF-17984DFC9
Initialised.
opened capture interface successfully
Capture started...
```

Player List

```
01 --- Unknown          VEH=NONE  WEP=AK47-AUTO MOV=STRFE_L
02 J0  Unknown          VEH=NONE  WEP=NADES     MOV=IDLE
03 --- Unknown          VEH=NONE  WEP=AK74-AUTO MOV=BACKWRD
04 J0  Unknown          VEH=NONE  WEP=M4-AUTO   MOV=FORWARD
05 --- Unknown          VEH=NONE  WEP=AK47-AUTO MOV=FORWARD

07 --- Unknown          VEH=NONE  WEP=AK47/GP A MOV=FORWARD
08 --- Unknown          VEH=NONE  WEP=AK47-AUTO MOV=UNK_49
09 --- Unknown          VEH=NONE  WEP=AK47-AUTO MOV=IDLE
0A --- Unknown          VEH=NONE  WEP=AK47-AUTO MOV=STRFE_R
0B J0  Unknown          VEH=NONE  WEP=M4-AUTO   MOV=IDLE
0C J0  Unknown          VEH=NONE  [illegible]
0D J0  Unknown          VEH=NONE  WEP=UNK_48    MOV=IDLE
0E --- Unknown          VEH=NONE  WEP=M82.5 SHT MOV=FW+LEFT


11 --- Unknown          VEH=NONE  WEP=AK47-AUTO MOV=FORWARD
```

☑ Enable Analysis
☐ Show TextPage
☐ Loc Graph
☑ Color Player List

```
CLIENT SPAWN
CHATRESP 0D  GLOBAL : [redacted]
CHATRESP 01  GLOBAL :
CLIENT SPAWN
CHATRESP 0C  GLOBAL :
CHATRESP 0C  TEAM : A
CHATRESP 0D  GLOBAL :
CHATRESP 0C  TEAM : A
CHATRESP FF  UNKNOWN_
CHATRESP 0D  GLOBAL :
SAY GLOBAL : n1□
CHATRESP 02  GLOBAL :
CHATRESP 01  GLOBAL :
CHATRESP 07  GLOBAL :
CHATRESP 0C  GLOBAL :
CHATRESP 0B  GLOBAL :
CHATRESP 0F  TEAM : D
CHATRESP 0A  GLOBAL :
```

Hi res picture: see http://www.cl.cam.ac.uk/~mkb23/jopsdec/lag3-censor.png