Level 0x02

XOR, what is it good for…

Topics

- PRNG Review and application: ps3
- A bit about digital logic
- Xor operations / rules
- Uses in assembly
- One time pad

Upcoming CTFs

- Cyber Security Rumble CTF (Germany)
  - Open to students and beginners
  - Sat Oct 8th, 1PM - Sun Oct 9th 1PM
  - https://cybersecurityrumble.de
- REPLY Cyber Security Challenge (Italy)
  - Students and Professionals
  - Fri Oct 14th, 1:30PM - Sat Oct 15th 1:30PM
  - https://challenges.reply.com/tamtamy/challenges/category/cybersecurity

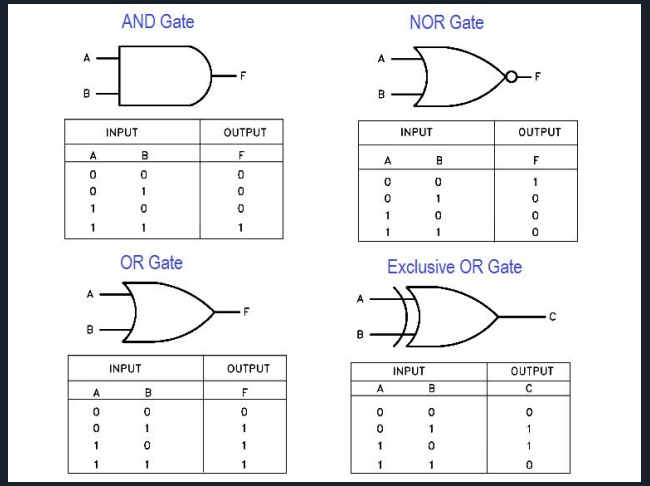
PRNG security - Playstation 3

- First 2 Star Wars challenges in Wildcat Practice CTF
  - Seeding a PRNG with a fixed value
  - Seeding a PRNG with time
- Playstation 3 security
  - Lots of effort put into securing platform
  - Epic pwnage after 4 years
  - Leaked the most private important key on the PS3…
    - Bad PRNG implementation

	Xbox	Wii	360	PS3
On-die bootROM	✓	✓	✓	✓
On-die key storage		✓	✓	
Public-key crypto	✓	✓	✓	✓
Chain of trust	✓		✓	✓
Per-console keys		✓	✓	✓
Signed executables	✓		✓	✓
Security coprocessor		✓		✓
Full media encryption and signing		✓		
Encrypted storage		✓		✓
Self-signed storage		✓		
Memory encryption/hashing			✓	
Hypervisor			✓	✓
User/kernelmode				✓
Anti-downgrade eFUSEs			✓	

Digital Logic

- CS, CpE, and EE will usually have a class on Digital Logic Design and Systems
  - Low voltage = OFF = False = 0
  - High voltage (3.3V) = ON = True = 1
- Boolean Operators
  - AND &&
  - OR ||



INPUT		OUTPUT
A	B	F
0	0	0
0	1	0
1	0	0
1	1	1

INPUT		OUTPUT
A	B	F
0	0	1
0	1	0
1	0	0
1	1	0

INPUT		OUTPUT
A	B	F
0	0	0
0	1	1
1	0	1
1	1	1

INPUT		OUTPUT
A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

```
if ( (var < 0) || (var > 100) ) {
     // print error
}
if ( (var >= 0) && (var <= 100) ) {
     // add to database
}
```

Bitwise Operators

- Similar to boolean, but evaluate each bit separately

```
var1 = 0x33; // 00110011
var2 = 0x58; // 01011000
```

```
// bitwise AND
var3 = var1 & var2;

  00110011
  01011000
& -------------
  00010000 = 0x10
```

```
// bitwise OR
var3 = var1 | var2;

  00110011
  01011000
| -------------
  01111011 = 0x7B
```

```
// bitwise XOR
var3 = var1 ^ var2;

  00110011
  01011000
^ -------------
  01101011 = 0x6B
```

XOR Rules / Quirks

X ^ X = zeros

X ^ zeros = X

X ^ ones = ~X

X ^ A ^ X = A

(X ^ A) ^ (X ^ B) = A ^ B

RandomData/HighEntropy ^ NormalData = RandomLookingData/HighEntropy

X86 Assembly: XORs everywhere!

godbolt.org

COMPILER EXPLORER | Add... | More | Templates | Listen to the Cpp.chat podcast | Backtrace | intel | Share | Policies | Other

C++ source #1 — C++

```cpp
#include <cstdlib>
#include <cstdio>



int main()
{
  int a = 0;
  printf("val = %d\n", a);

  int b = 1;
  printf("val = %d\n", b);

  return EXIT_SUCCESS;
}
```

x86-64 gcc 10.1 (C++, Editor #1, Compiler #1) — x86-64 gcc 10.1 — -O2

Output... | Filter... | Libraries (1) | Add new... | Add tool...

```asm
.LC0:
        .string "val = %d\n"
main:
        sub     rsp, 8
        xor     esi, esi
        mov     edi, OFFSET FLAT:.LC0
        xor     eax, eax
        call    printf
        mov     esi, 1
        mov     edi, OFFSET FLAT:.LC0
        xor     eax, eax
        call    printf
        xor     eax, eax
        add     rsp, 8
        ret
```

Output (0/0) x86-64 gcc 10.1 - 2104ms (35091B) ~2574 lines filtered | Compiler License

Output of x86-64 gcc 10.1 (Compiler #1) — Wrap lines | Select all

```
ASM generation compiler returned: 0
Execution build compiler returned: 0
Program returned: 0
  val = 0
  val = 1
```

X86 Assembly

godbolt.org

COMPILER EXPLORER | Add... | More | Templates | Listen to the Cpp.chat podcast | Backtrace | intel | Share | Policies | Other

C++ source #1 — C++

```
1  #include <cstdlib>
2  #include <cstdio>
3
4
5
6  int main()
7  {
8    int a = 0;
9    printf("val = %d\n", a);
10
11   int b = 1;
12   printf("val = %d\n", b);
13
14   return EXIT_SUCCESS;
15 }
16
```

x86-64 gcc 10.1 (C++, Editor #1, Compiler #1) — x86-64 gcc 10.1 — -O2

Output... | Filter... | Libraries (1) | + Add new... | Add tool...

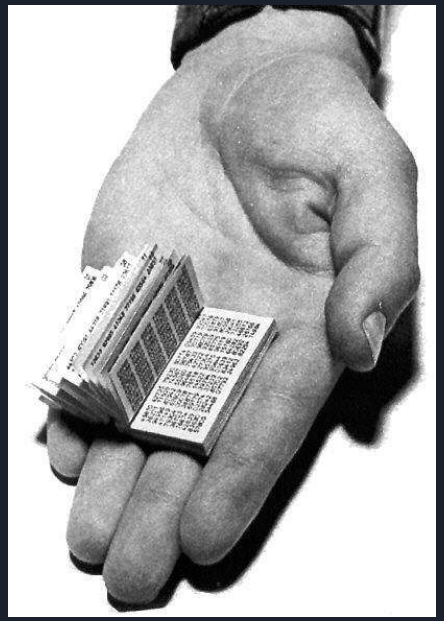
```
main:
        48 83 ec 08
401040  sub    rsp,0x8
        31 f6
401044  xor    esi,esi
        bf 04 20 40 00
401046  mov    edi,0x402004
        31 c0
40104b  xor    eax,eax
        e8 de ff ff ff
40104d  call   401030 <printf@plt>
        be 01 00 00 00
401052  mov    esi,0x1
        bf 04 20 40 00
401057  mov    edi,0x402004
        31 c0
40105c  xor    eax,eax
        e8 cd ff ff ff
40105e  call   401030 <printf@plt>
        31 c0
401063  xor    eax,eax
```

Output (0/0) x86-64 gcc 10.1 - 3303ms (22752B) Compiler License

Output of x86-64 gcc 10.1 (Compiler #1) — Wrap lines | Select all

```
ASM generation compiler returned: 0
Execution build compiler returned: 0
Program returned: 0
  val = 0
  val = 1
```

Encryption / One Time Pad



- One Time Pad (OTP) Encryption
- Generate a long file with random bytes
  - Give a copy of it to your friend
  - Keep this file secret (it's your encryption key)
- XOR your msg aka plaintext (PT) with OTP to get ciphertext (CT)
- Send ciphertext to your friend
- XOR ciphertext and OTP to get back plaintext

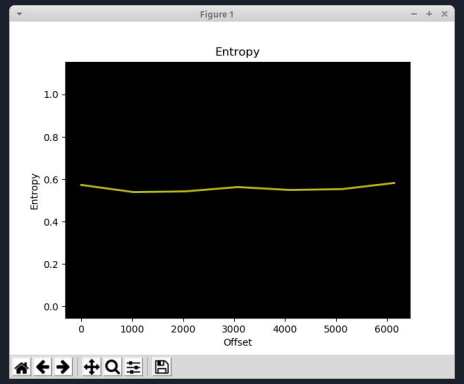
PT ^ OTP => CT

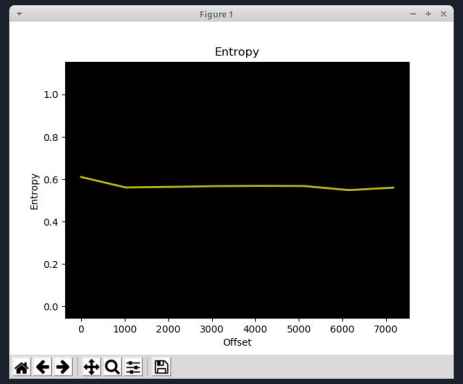
CT ^ OTP => PT

Don’t reuse OTP

- Once a byte from OTP is used, it is never used for future messages
  - Forces OTPs to be really huge if used frequently
  - Difficult to use / keep track of position in OTP

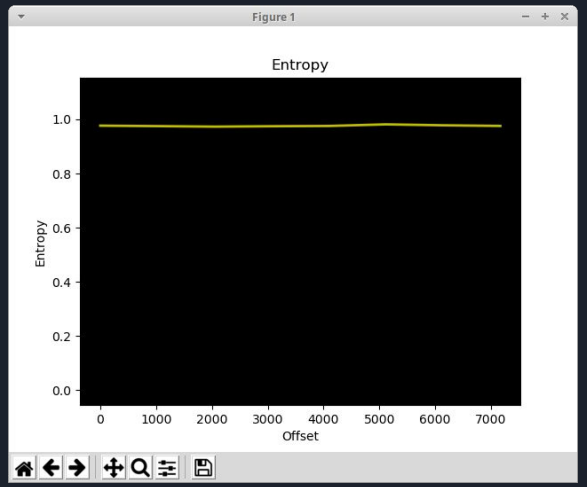
```
mwales@Metroid:~/checkouts/WildcatCSClub/slides/level2/xor$ hexdump -C plaintext
.txt | head -n 10
00000000  0a 4f 6e 65 2d 54 69 6d  65 2d 50 61 64 20 28 56  |.One-Time-Pad (V|
00000010  65 72 6e 61 6d 27 73 20  43 69 70 68 65 72 29 20  |ernam's Cipher) |
00000020  46 72 65 71 75 65 6e 74  6c 79 20 41 73 6b 65 64  |Frequently Asked|
00000030  20 51 75 65 73 74 69 6f  6e 73 0a 0a 20 20 20 20  | Questions..    |
00000040  22 41 20 6f 6e 65 2d 74  69 6d 65 20 70 61 64 20  |"A one-time pad |
00000050  69 73 6e 27 74 20 61 20  63 72 79 70 74 6f 73 79  |isn't a cryptosy|
00000060  73 74 65 6d 3a 20 69 74  27 73 20 61 20 73 74 61  |stem: it's a sta|
00000070  74 65 20 6f 66 20 6d 69  6e 64 2e 22 20 2d 20 4d  |te of mind." - M|
00000080  61 72 63 75 73 20 52 61  6e 75 6d 0a 0a 0a 41 20  |arcus Ranum...A |
00000090  52 75 73 73 69 61 6e 20  4f 6e 65 2d 74 69 6d 65  |Russian One-time|
mwales@Metroid:~/checkouts/WildcatCSClub/slides/level2/xor$ hexdump -C 2nd_pt.tx
t | head -n 10
00000000  0a 48 6f 6d 65 70 61 67  65 0a 43 72 79 70 74 6f  |.Homepage.Crypto|
00000010  0a 49 6e 64 65 78 0a 47  6c 6f 73 73 61 72 79 0a  |.Index.Glossary.|
00000020  45 6e 69 67 6d 61 0a 48  61 67 65 6c 69 6e 0a 46  |Enigma.Hagelin.F|
00000030  69 61 6c 6b 61 0a 52 6f  74 6f 72 0a 50 69 6e 2d  |ialka.Rotor.Pin-|
00000040  77 68 65 65 6c 0a 56 6f  69 63 65 0a 44 61 74 61  |wheel.Voice.Data|
00000050  0a 48 61 6e 64 0a 4f 54  50 0a 45 4d 55 0a 4d 69  |.Hand.OTP.EMU.Mi|
00000060  78 65 72 73 0a 50 68 6f  6e 65 73 0a 42 75 6c 6b  |xers.Phones.Bulk|
00000070  0a 46 49 4c 4c 0a 43 6f  64 65 62 6f 6f 6b 73 0a  |.FILL.Codebooks.|
00000080  41 6c 67 6f 72 69 74 68  6d 73 0a 43 72 79 70 74  |Algorithms.Crypt|
00000090  61 6e 61 6c 79 73 69 73  0a 43 6f 75 6e 74 72 69  |analysis.Countri|
```



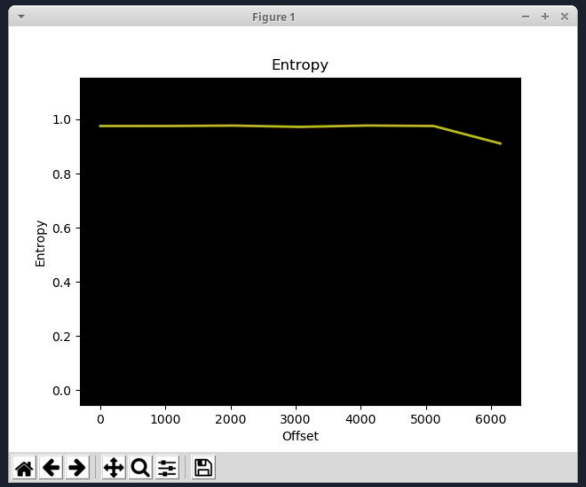


Entropy of ciphertext

One time pad entropy (random data)



Entropy of CT (PT ^ OTP)



One Time Pad Reuse

(msg1 ^ OTP) = ct1

(msg2 ^ OTP) = ct2

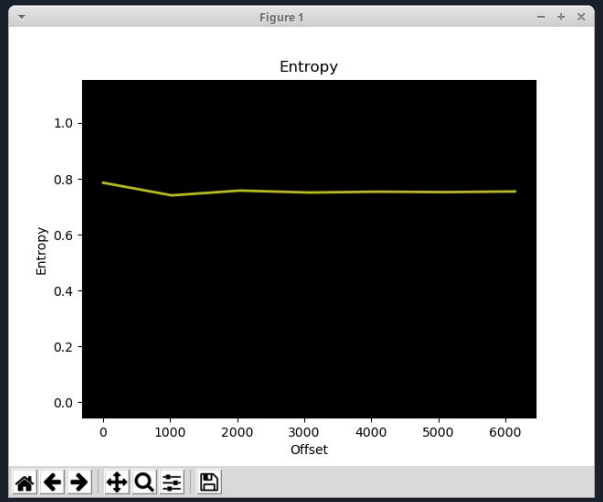
ct1 ^ ct2 = (msg1 ^ OTP) ^ (msg2 ^ OTP)

= (OTP ^ OTP) ^ (msg1 ^ msg2)

= (zeros) ^ (msg1 ^ msg2)

= msg1 ^ msg2

Entropy of msg1 ^ msg2



Links

- Failoverflow CCC 2010 presentation on Console Hacking
  - https://fahrplan.events.ccc.de/congress/2010/Fahrplan/attachments/1780_27c3_console_hacking_2010.pdf
- https://instrumentationtools.com/logic-gates-and-truth-tables/
- https://godbolt.org/
- http://www.ranum.com/security/computer_security/papers/otp-faq/