암호학 스터디 3주차

여러분 죄송합니다ㅠㅠ

못난 암호학 스터디장이 건망증이 심해서 스터디 자료 제작이 늦었습니다.ㅠㅠ 용서해주세요

이번시간에는 지난시간 문제풀이하고, 현대암호를 들어가기에 앞서 간단한 내용만 알아가려고 합니다.

이번에도 과제가 있습니다! (한문제)

문제 못푸시겠으면, 어떻게 풀어야할 것 같다라는 생각만 짧게 남겨주셔도 됩니다!

자 그러면 시작합니다.

1. 지난 시간

에는 무슨 문제를 풀었을까요?

Gur Mra bs Clguba, ol Gvz Crgref Ornhgvshy vf orggre guna htyl. Rkcyvpvg vf orggre guna vzcyvpvg. Fvzcyr vf orggre guna pbzcyrk. Pbzcyrk vf orggre guna pbzcyvpngrq. Syng vf orggre guna arfgrq. Fcnefr vf orggre guna qrafr. Ernqnovyvgl pbhagf. Fcrpvny pnfrf nera'g fcrpvny rabhtu gb oernx gur ehyrf. Nygubhtu cenpgvpnyvgl orngf chevgl. Reebef fubhyq arire cnff fvyragyl. Hayrff rkcyvpvgyl fvyraprq. Va gur snpr bs nzovthvgl, ershfr gur grzcgngvba gb thrff. Gurer fubhyq or bar-- naq cersrenoyl bayl bar --boivbhf jnl gb qb vg. Naq bs pbhefr, Synt Fubhyq or urer! Vg vf ZnxrClgu0ater4gntn1a Nygubhtu gung jnl znl abg or boivbhf ng svefg hayrff lbh'er Qhgpu. Abj vf orggre guna arire. Nygubhtu arire vf bsgra orggre guna evtug abj. Vs gur vzcyrzragngvba vf uneq gb rkcynva, vg'f n onq vqrn. Vs gur vzcyrzragngvba vf rnfl gb rkcynva, vg znl or n tbbq vqrn. Anzrfcnprf ner bar ubaxvat terng vqrn -- yrg'f qb zber bs gubfr!

가 첫 문제였네요.

이 암호문이 암호화된 방식은 Additive Cipher 라고 문제 처음에 언급이 되어있습니다.

아무것도 모르면 딱 25번만 돌려보면 됩니다.

복호화 코드는 1, 2주차를 참고하면 됩니다.

그렇게 무차별적으로 키를 대입하다 보면

K = 13일 때

The Zen of Python, by Tim Peters Beautiful is better than ugly. Explicit is better than implicit. Simple is better than complex. Complex is better than complicated. Flat is better than nested. Sparse is better than dense. Readability counts. Special cases aren't special enough to break the rules. Although practicality beats purity. Errors should never pass silently. Unless explicitly silenced. In the face of ambiguity, refuse the temptation to guess. There should be one-- and preferably only one --obvious way to do it. And of course, Flag Should be here! It is MakePyth0ngre4taga1n Although that way may not be obvious at first unless you're Dutch. Now is better than never. Although never is often better than right now. If the implementation is hard to explain, it's a bad idea. If the implementation is easy to explain, it may be a good idea. Namespaces are one honking great idea -- let's do more of those!

라는 평문을 얻을 수 있습니다.

쓸데없는 사실이지만 실제로 Python 키고 import this치면 저 평문이 나와요

두번째 문제는

Msnjsp jf qms Pqjng (MjqP) ap h gudqakdhysn jidais ehqqds hnsih vaosj thgs osvsdjkso hio kuedapmso ey Edazzhno Siqsnqhaigsiq fjn Garnjpjfq Waiojwp hio ghrJP qmhq whp nsdshpso ji Buis 2, 2015. Qms thgs fshqunsp msnjsp fnjg Edazzhno'p fnhirmapsp airduoait Whnrnhfq, Oahedj, PqhnRnhfq, Qms Djpq Vacaitp, hio Jvsnwhqrm. Qms thgs upsp ejqm fnss-qj-kdhy hio fnssgaug gjosdp hio ap pukkjnqso ey garnjkhygsiqp, wmarm rhi es upso qj kunrmhps msnjsp, vapuhd hdqsnhqajip fjn qms msnjsp ai qms thgs, hio gjuiqp. Edazzhno ojsp ijq rhdd qms thgs h "gudqakdhysn jidais ehqqds hnsih" jn hi "hrqaji nshd-qags pqnhqsty" esrhups qmsy fssd aq ap pjgsqmait oaffsnsiq waqm h enjhosn kdhypqyds; qmsy nsfsn qj aq hp hi jidais "msnj enhwdsn". Msnjsp jf qms Pqjng nsvjdvsp hnjuio jidais 5-vsnpup-5 ghqrmsp, jksnhqso qmnjutm Edazzhno'p jidais thgait psnvars Ehqqds.isq. Kdhysnp rhi rmjjps fnjg oaffsnsiq thgs gjosp, wmarm airduos kdhyait hthaipq rjgkuqsn-rjiqnjddso msnjsp jn jqmsn kdhysnp. Aiaqahddy, ij msnjsp hns hvhadheds fjn ksnghisiq ups; mjwsvsn, kdhysnp ghy rmjjps fnjg h dapq jf msnjsp qmhq hns fnss qj ups fnjg h wsscdy njqhqaji. Ey upait tjdo rjaip, qms ai-thgs runnsiry, jn qmnjutm garnjqnhiphrqajip, qmsy rhi thai ksnghisiq hrrspp qj h msnj. Hp jf Ghy 2017, qmsns hns runnsiqdy 67 msnjsp ai qms thgs, oavaoso aiqj favs pskhnhqs njdsp: Hpphppai, Whnnajn, Pukkjnq, Pksrahdapq hio jis Gudqardhpp msnj. Qmsns hns runnsiqdy 13 ghkp hvhadheds qj kdhy, hdd jf wmarm mhvs oaffsnsiq jebsrqavsp qj psruns, waqm pjgs mhvait oaffsnsiq varqjny rjioaqajip. Sxksnasirs kjaiqp, wmarm rhi es thaiso ey esait ishney sisgy uiaqp wmsi qmsy'ns caddso, hns pmhnso hrnjpp qms siqans qshg. Wmsi h qshg nshrmsp h rsnqhai sxksnasirs kjaiq qmnspmjdo, svsny msnj ji qmhq qshg dsvsdp uk, hrluanait pdatmqdy hgkdafaso kjwsnp. Svsny fsw dsvsdp, kdhysnp ghy psdsrq h qhdsiq wmarm jffsnp h isw headaqy, jn hutgsiqp hi sxapqait jis. Qmap dsvsdait pypqsg sgkmhpazsp qms agkjnqhirs jf qshgwjnc hio kdhiiait, pairs h kdhysn'p hrqaji rhi hffsrq qms wmjds qshg. Kdhysnp rhi hdpj gjuiq oaffsnsiq hiaghdp, purm hp mjnpsp, dazhnop, jn uiarjnip, qj airnshps qmsan gjvsgsiq pksso, huqjghqarhddy oapgjuiqait wmsi oshdait/nsrsavait ohghts jn upait hi headaqy. Hio Kdhysnp jf CUARP whnthgs rhi tsq h fdht qmnjutmq dsqqsn fnslusiry hihdypap. Aq ap KjcKuitjfPatjitApQmsEspq!

였네요. 저번시간에 Subtitution Cipher를 배웠으니, 그거겠죠. 물론 Statistical Attack에 어느정도 취약하다고 이야기도 했습니다.

빈도수 구하는 방법은 파이썬 돌려서 딕셔너리를 잘 사용하시면 됩니다.

하지만 코드를 남겨놓을게요

```
>>> dict = {}  #딕셔너리 선언

>>> string = """Msnjsp jf qms Pqjng (MjqP) ap h gudqakdhysn jidais ehqqds hnsih vaosj thgs osvsdjkso hio kuedapmso ey Edazzhno Siqsnqhaigsiq fjn Garnjpjfq Waiojwp hio ghrJP qmhq whp nsdshpso ji Buis 2, 2015. Qms thgs fshqunsp msnjsp fnjg Edazzhno'p fnhirmapsp airduoait Whnrnhfq, Oahedj, PqhnRnhfq, Qms Djpq Vacaitp, hio Jvsnwhqrm. Qms thgs upsp ejqm fnss-qj-kdhy hio fnssgaug gjosdp hio ap pukkjnqso ey garnjkhygsiqp, wmarm rhi es upso qj kunrmhps msnjsp, vapuhd hdqsnhqajip fjn qms msnjsp ai qms thgs, hio gjuiqp. Edazzhno ojsp ijq rhdd qms thgs h "gudqakdhysn jidais ehqqds hnsih" jn hi "hrqaji nshd-qags pqnhqsty" esrhups qmsy fssd aq ap pjgsqmait oaffsnsiq waqm h enjhosn kdhypqyds; qmsy nsfsn qj aq hp hi jidais "msnj enhwdsn". Msnjsp jf qms Pqjng nsvjdvsp hnjuio jidais 5-vsnpup-5 ghqrmsp, jksnhqso qmnjutm Edazzhno'p jidais thgait psnvars Ehqqds.isq. Kdhysnp rhi rmjjps fnjg oaffsnsiq thgs gjosp, wmarm airduos kdhyait hthaipq rjgkuqsn-rjiqnjddso msnjsp jn jqmsn kdhysnp. Aiaqahddy, ij msnjsp hns hvhadheds fjn ksnghisiq ups; mjwsvsn, kdhysnp ghy rmjjps fnjg h dapq jf msnjsp qmhq hns fnss qj ups fnjg h wsscdy njqhqaji. Ey upait tjdo rjaip, qms ai-thgs runnsiry, jn qmnjutm garnjqnhiphrqajip, qmsy rhi thai ksnghisiq hrrspp qj h msnj. Hp jf Ghy 2017, qmsns hns runnsiqdy 67 msnjsp ai qms thgs, oavaoso aiqj favs pskhnhqs njdsp: Hpphppai, Whnnajn, Pukkjnq, Pksrahdapq hio jis Gudqardhpp msnj. Qmsns hns runnsiqdy 13 ghkp hvhadheds qj kdhy, hdd jf wmarm mhvs oaffsnsiq jebsrqavsp qj psruns, waqm pjgs mhvait oaffsnsiq varqjny rjioaqajip. Sxksnasirs kjaiqp, wmarm rhi es thaiso ey esait ishney sisgy uiaqp wmsi qmsy'ns caddso, hns pmhnso hrnjpp qms siqans qshg. Wmsi h qshg nshrmsp h rsnqhai sxksnasirs kjaiq qmnspmjdo, svsny msnj ji qmhq qshg dsvsdp uk, hrluanait pdatmqdy hgkdafaso kjwsnp. Svsny fsw dsvsdp, kdhysnp ghy psdsrq h qhdsiq wmarm jffsnp h isw headaqy, jn hutgsiqp hi sxapqait jis. Qmap dsvsdait pypqsg sgkmhpazsp qms agkjnqhirs jf qshgwjnc hio kdhiiait, pairs h kdhysn'p hrqaji rhi hffsrq qms wmjds qshg. Kdhysnp rhi hdpj gjuiq oaffsnsiq hiaghdp, purm hp mjnpsp, dazhnop, jn uiarjnip, qj airnshps qmsan gjvsgsiq pksso, huqjghqarhddy oapgjuiqait wmsi oshdait/nsrsavait ohghts jn upait hi headaqy. Hio Kdhysnp jf CUARP whnthgs rhi tsq h fdht qmnjutmq dsqqsn fnslusiry hihdypap. Aq ap KjcKuitjfPatjitApQmsEspq!"""  # 암호문

>>> for i in string:  # 암호문을 돌면서

...     if i not in dict:  # 딕셔너리에 없으면

...         dict[i] = 1  # 추가하고

...     else:  # 있으면

...         dict[i] += 1  # 카운트를 늘린다

...

>>> dict  # 결과
```

{'M': 3, 's': 250, 'n': 137, 'j': 123, 'p': 118, ' ': 354, 'f': 44, 'q': 135, 'm': 79, 'P': 9, 'g': 56, '(': 1, ')': 1, 'a': 122, 'h': 168, 'u': 42, 'd': 89, 'k': 36, 'y': 44, 'i': 127, 'e': 20, 'v': 24, 'o': 50, 't': 42, 'E': 7, 'z': 10, 'S': 3, 'G': 3, 'r': 62, 'W': 4, 'w': 21, 'J': 2, 'B': 1, '2': 3, ',': 34, '0': 2, '1': 3, '5': 3, '.': 17, 'Q': 6, '"': 4, 'O': 1, 'R': 2, 'D': 1, 'V': 1, 'c': 5, '-': 7, '"': 6, ';': 2, 'K': 5, 'A': 4, 'H': 3, '7': 2, '6': 1, ':': 1, '3': 1, 'b': 1, 'x': 3, 'l': 2, '/': 1, 'C': 1, 'U': 1, '!': 1}

코드를 돌리면 빈도수를 확인할 수 있습니다.

영어 빈도수 통계를 볼 때, s가 원래 e였다고 생각할 수 있겠군요.

이렇게 하나씩 맞춰가면 됩니다.

그러면 답은

```
Heroes of the Storm (HotS) is a multiplayer online battle arena video game developed and published by Blizzard Entertainment for Microsoft Windows and macOS that was released on June 2, 2015. The game features heroes from Blizzard's franchises including Warcraft, Diablo, StarCraft, The Lost Vikings, and Overwatch. The game uses both free-to-play and freemium models and is supported by micropayments, which can be used to purchase heroes, visual alterations for the heroes in the game, and mounts. Blizzard does not call the game a "multiplayer online battle arena" or an "action real-time strategy" because they feel it is something different with a broader playstyle; they refer to it as an online "hero brawler". Heroes of the Storm revolves around online 5-versus-5 matches, operated through Blizzard's online gaming service Battle.net. Players can choose from different game modes, which include playing against computer-controlled heroes or other players. Initially, no heroes are available for permanent use; however, players may choose from a list of heroes that are free to use from a weekly rotation. By using gold coins, the in-game currency, or through microtransactions, they can gain permanent access to a hero. As of May 2017, there are currently 67 heroes in the game, divided into five separate roles: Assassin, Warrior, Support, Specialist and one Multiclass hero. There are currently 13 maps available to play, all of which have different objectives to secure, with some having different victory conditions. Experience points, which can be gained by being nearby enemy units when they're killed, are shared across the entire team. When a team reaches a certain experience point threshold, every hero on that team levels up, acquiring slightly amplified powers. Every few levels, players may select a talent which offers a new ability, or augments an existing one. This leveling system emphasizes the importance of teamwork and planning, since a player's action can affect the whole team. Players can also mount different animals, such as horses, lizards, or unicorns, to increase their movement speed, automatically dismounting when dealing/receiving damage or using an ability. And Players of KUICS wargame can get a flag throught letter frequency analysis. It is PokPungofSigongIsTheBest!
```

이 되는 것을 확인할 수 있군요

그런데 직접 손으로 하기 귀찮을 겁니다.

그럴때는 quipqiup.com에 가면 알아서 해줍니다. <- 이번시간 내용중에 가장 중요한 내용입니다.

2. 다시 mod 로

Mod 에 대해서는 제대로 알고 넘어가야 합니다. 계속 나올거거든요.

하지만 필요한 것만 알고 갑니다

2.1.

페르마의 소정리라는 게 있습니다.

p 가 소수이고 a 가 p 의 배수가 아니면

$a^{p-1} \equiv 1 \, mod \, p$ 입니다.

이를 이용하면 123^460 === ? mod 461 이 주어졌을 때

? = 1 임을 쉽게 찾을 수 있죠

2.2.

비슷한 정리를 알아보기 전에 pi-function 이라는 것을 알아봅시다.

$\varphi(n)$ = (1 부터 n-1 까지, n 과 서로소인 수의 개수)입니다.

우리가 여기서 특히 알아야 할 성질은

p, q 가 소수고, n = p * q 면,

$\varphi(n) = \varphi(q) * \varphi(p)$라는 것입니다.

쉽게 말해서 $\varphi(10) = \varphi(2) * \varphi(5)$니

$\varphi(2) = 2 - 1 = 1, \varphi(5) = 5 - 1 = 4$니

$\varphi(10)$ = 1 * 4 = 4 가 되네요

p 가 소수일 때, $\varphi(p) = p - 1$임은 잘 생각하면 당연하다는 것을 알 수 있습니다.

2.3.

이 φ함수로부터 새로운 정리가 나옵니다.

오일러의 정리라는 것인데요,

a 와 n 이 서로소일 때,

$a^{\varphi(n)} \equiv 1 \, mod \, n$이 성립합니다.

그냥 아무생각이 안들겠지만, 이 공식이 RSA 의 기초가 됩니다.

3. XOR

C 배울 때, ^라고 다들 알고 있을겁니다.

그런데 어디에 쓰이는지는 잘 모를겁니다.

여기서 자주 쓸겁니다.

1 ^ 1 = 0

1 ^ 0 = 1

0 ^ 1 = 1

0 ^ 0 = 0

입니다.

더 쉽게 생각해보죠. 그냥 xor 의 연산자가 1 이면 비트를 뒤집는다고 생각하면 되는데,

0 ^ 1 일 때, 0 인 비트를 뒤집어서 1 이 나온다고 생각하면 되고

1 ^ 0 일 때, 1 인 비트를 뒤집지 않는다고 생각하면 그냥 1 이 됩니다.

쉽죠?

예시를 봅시다.

0x01 ^ 0xff 를 해봅시다.

0x01 -> 0000 0001

0xff -> 1111 1111 입니다.

위의 느낌을 가지고 가면

0x01 ^ 0xff 는 1111 1110 -> 0xfe 가 되는 것을 알 수 있습니다.

xor 의 가장 중요한 점은 똑같은 수를 xor 을 두번하면 사라진다는 것입니다

a ^ b ^ b = a 라는 것이죠.

0xff ^ 0x12 ^ 0x12 는 무엇일까요?

0x12 ^ 0x12 = 0 이 되어버리니

0xff ^ 0x12 ^ 0x12 = 0xff ^ 0x00 = 0xff 가 되는것이죠.

잘 감이 오지 않을 것 같아서 문제를 하나 드릴게요.

이번주의 문제입니다.

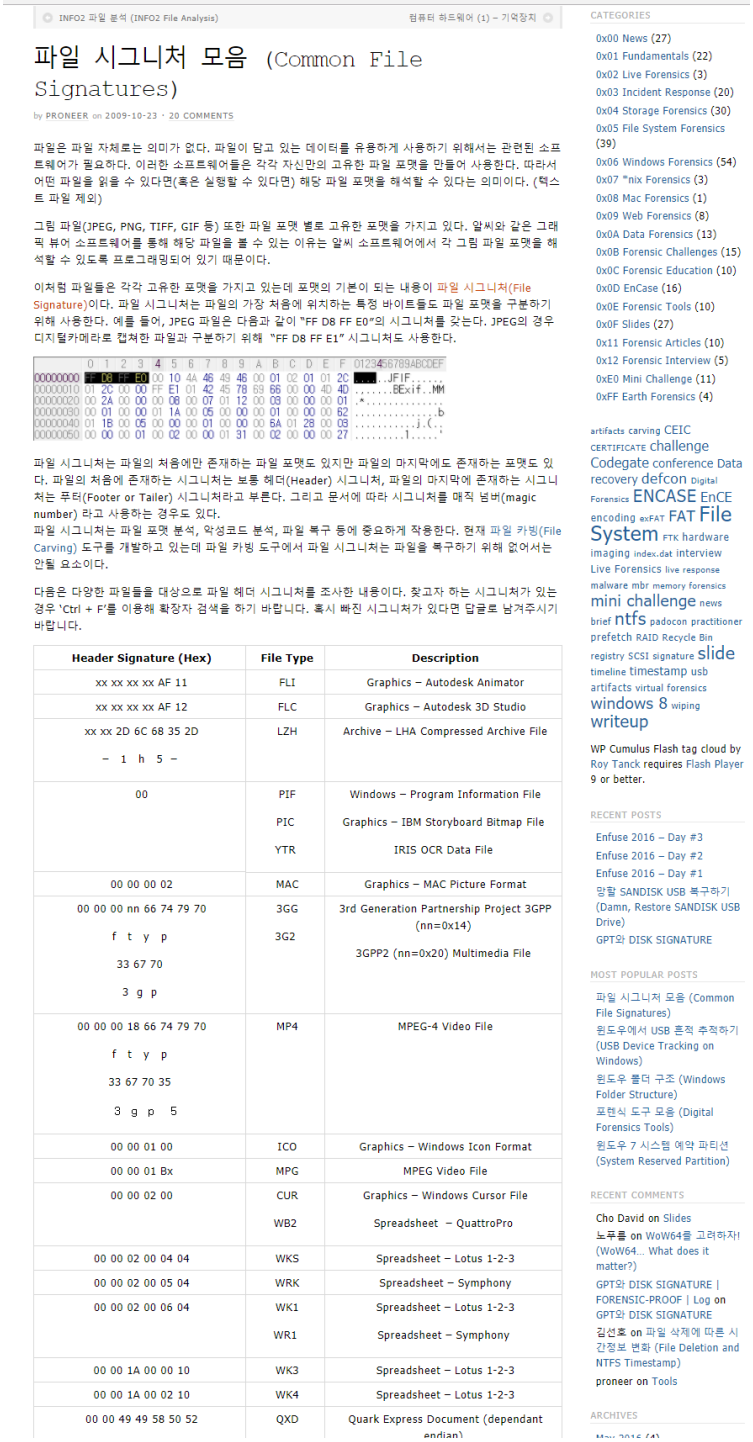
실제 CTF 에 나왔던 문제를 쓰겠습니다.

Hxd 를 설치하셔야 합니다. 인터넷에 검색하면 바로 찾을 수 있어요.

문제

기초지식

대부분의 파일은 file signature 라는 것을 가집니다. 이 파일이 어떠한 파일인지를 나타내는 정보가 맨 앞에 붙어있다고 생각하면 됩니다.



INFO2 파일 분석 (INFO2 File Analysis) 컴퓨터 하드웨어 (1) – 기억장치

파일 시그니처 모음 (Common File Signatures)

by PRONEER on 2009-10-23 · 20 COMMENTS

파일은 파일 자체로는 의미가 없다. 파일이 담고 있는 데이터를 유용하게 사용하기 위해서는 관련된 소프트웨어가 필요하다. 이러한 소프트웨어들은 각각 자신만의 고유한 파일 포맷을 만들어 사용한다. 따라서 어떤 파일을 읽을 수 있다면(혹은 실행할 수 있다면) 해당 파일 포맷을 해석할 수 있다는 의미이다. (텍스트 파일 제외)

그림 파일(JPEG, PNG, TIFF, GIF 등) 또한 파일 포맷 별로 고유한 포맷을 가지고 있다. 알씨와 같은 그래픽 뷰어 소프트웨어를 통해 해당 파일을 볼 수 있는 이유는 알씨 소프트웨어에서 각 그림 파일 포맷을 해석할 수 있도록 프로그래밍되어 있기 때문이다.

이처럼 파일들은 각각 고유한 포맷을 가지고 있는데 포맷의 기본이 되는 내용이 파일 시그니처(File Signature)이다. 파일 시그니처는 파일의 가장 처음에 위치하는 특정 바이트들도 파일 포맷을 구분하기 위해 사용한다. 예를 들어, JPEG 파일은 다음과 같이 "FF D8 FF E0"의 시그니처를 갖는다. JPEG의 경우 디지털카메라로 캡쳐한 파일과 구분하기 위해 "FF D8 FF E1" 시그니처도 사용한다.

```
          0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
00000000  FF D8 FF E0 00 10 4A 46 49 46 00 01 02 01 01 2C  ....JFIF.....,
00000010  01 2C 00 00 FF E1 01 42 45 78 69 66 00 00 4D 4D  .,...áBExif..MM
00000020  00 2A 00 00 00 08 00 07 01 12 00 03 00 00 00 01  .*..............
00000030  00 01 00 00 01 1A 00 05 00 00 00 01 00 00 00 62  ...............b
00000040  01 1B 00 05 00 00 00 01 00 00 00 6A 01 28 00 03  ...........j.(..
00000050  00 00 00 01 00 02 00 00 01 31 00 02 00 00 00 27  .........1.....'
```

파일 시그니처는 파일의 처음에만 존재하는 파일 포맷도 있지만 파일의 마지막에도 존재하는 포맷도 있다. 파일의 처음에 존재하는 시그니처는 보통 헤더(Header) 시그니처, 파일의 마지막에 존재하는 시그니처는 푸터(Footer or Tailer) 시그니처라고 부른다. 그리고 문서에 따라 시그니처를 매직 넘버(magic number) 라고 사용하는 경우도 있다.
파일 시그니처는 파일 포맷 분석, 악성코드 분석, 파일 복구 등에 중요하게 작용한다. 현재 파일 카빙(File Carving) 도구를 개발하고 있는데 파일 카빙 도구에서 파일 시그니처는 파일을 복구하기 위해 없어서는 안될 요소이다.

다음은 다양한 파일들을 대상으로 파일 헤더 시그니처를 조사한 내용이다. 찾고자 하는 시그니처가 있는 경우 'Ctrl + F'를 이용해 확장자 검색을 하기 바랍니다. 혹시 빠진 시그니처가 있다면 답글로 남겨주시기 바랍니다.

Header Signature (Hex)	File Type	Description
xx xx xx xx AF 11	FLI	Graphics – Autodesk Animator
xx xx xx xx AF 12	FLC	Graphics – Autodesk 3D Studio
xx xx 2D 6C 68 35 2D – l h 5 –	LZH	Archive – LHA Compressed Archive File
00	PIF PIC YTR	Windows – Program Information File Graphics – IBM Storyboard Bitmap File IRIS OCR Data File
00 00 00 02	MAC	Graphics – MAC Picture Format
00 00 00 nn 66 74 79 70 f t y p 33 67 70 3 g p	3GG 3G2	3rd Generation Partnership Project 3GPP (nn=0x14) 3GPP2 (nn=0x20) Multimedia File
00 00 00 18 66 74 79 70 f t y p 33 67 70 35 3 g p 5	MP4	MPEG-4 Video File
00 00 01 00	ICO	Graphics – Windows Icon Format
00 00 01 Bx	MPG	MPEG Video File
00 00 02 00	CUR WB2	Graphics – Windows Cursor File Spreadsheet – QuattroPro
00 00 02 00 04 04	WKS	Spreadsheet – Lotus 1-2-3
00 00 02 00 05 04	WRK	Spreadsheet – Symphony
00 00 02 00 06 04	WK1 WR1	Spreadsheet – Lotus 1-2-3 Spreadsheet – Symphony
00 00 1A 00 00 10	WK3	Spreadsheet – Lotus 1-2-3
00 00 1A 00 02 10	WK4	Spreadsheet – Lotus 1-2-3
00 00 49 49 58 50 52	QXD	Quark Express Document (dependant endian)

CATEGORIES
0x00 News (27)
0x01 Fundamentals (22)
0x02 Live Forensics (3)
0x03 Incident Response (20)
0x04 Storage Forensics (30)
0x05 File System Forensics (39)
0x06 Windows Forensics (54)
0x07 *nix Forensics (3)
0x08 Mac Forensics (1)
0x09 Web Forensics (8)
0x0A Data Forensics (13)
0x0B Forensic Challenges (15)
0x0C Forensic Education (10)
0x0D EnCase (16)
0x0E Forensic Tools (10)
0x0F Slides (27)
0x11 Forensic Articles (10)
0x12 Forensic Interview (5)
0xE0 Mini Challenge (11)
0xFF Earth Forensics (4)

artifacts carving CEIC CERTIFICATE challenge Codegate conference Data recovery defcon Digital Forensics ENCASE EnCE encoding exFAT FAT File System FTK hardware imaging index.dat interview Live Forensics live response malware mbr memory forensics mini challenge news brief ntfs padocon practitioner prefetch RAID Recycle Bin registry SCSI signature slide timeline timestamp usb artifacts virtual forensics windows 8 wiping writeup

WP Cumulus Flash tag cloud by Roy Tanck requires Flash Player 9 or better.

RECENT POSTS
Enfuse 2016 – Day #3
Enfuse 2016 – Day #2
Enfuse 2016 – Day #1
망할 SANDISK USB 복구하기 (Damn, Restore SANDISK USB Drive)
GPT와 DISK SIGNATURE

MOST POPULAR POSTS
파일 시그니처 모음 (Common File Signatures)
윈도우에서 USB 흔적 추적하기 (USB Device Tracking on Windows)
윈도우 폴더 구조 (Windows Folder Structure)
포렌식 도구 모음 (Digital Forensics Tools)
윈도우 7 시스템 예약 파티션 (System Reserved Partition)

RECENT COMMENTS
Cho David on Slides
노루룹 on WoW64를 고려하자! (WoW64... What does it matter?)
GPT와 DISK SIGNATURE | FORENSIC-PROOF | Log on GPT와 DISK SIGNATURE
김선호 on 파일 삭제에 따른 시간정보 변화 (File Deletion and NTFS Timestamp)
proneer on Tools

ARCHIVES

http://forensic-proof.com/archives/300

실제 문제

https://ctf.kaspersky.com/contests/1/tasks/16/

이 링크에 문제가 있습니다.

힌트

문제의 링크에 있는 파일을 hxd 로 열어보면 뭔가 반복되는 부분이 있습니다. 저런 경우에는 한바이트씩 암호화 했을 가능성이 있죠.

xor 을 이번에 가르친 이유와 파일 시그니처가 기초지식에 올라와있는 것을 참고하시면 문제를 푸는 방향을 어느정도 잡을 수 있을 것입니다.