University of Amsterdam
System and Network Engineering

Known plaintext attack on encrypted ZIP files

Barosan Dragos Laurentiu

Supervisor: Armijn Hemel

Why?

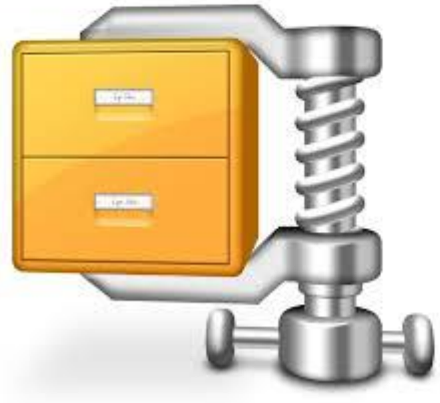
- There is no open source implementation
  - Source Code available for PkCrack by Peter Conrad

- Interesting to study a successful attack on encryption

Research Questions

- How feasible is to obtain plaintext?
- What implementation options are for the attack?

Is it still used ?



- Winzip
  - default AES, can use classic ZIP encryption
- Winrar
  - only classic ZIP encryption for ZIP format
- 7ZIP
  - default classic ZIP encryption, can use AES
- PKZIP
  - default AES, can use classic ZIP encryption
- ZIP utility on Linux
  - Only classic ZIP encryption

Zip encryption

- Stream cipher
- Internal state represented by 3 variables on 32 bits each
  - Key0, key1, key2
- Default known internal state updated by the password
  - Afterwards updated by plaintext
- key3 is the actual encryption key and is derived from key2
- 12 bytes encryption header prepended

Internal state keys dependency

- $key0_i$ = $f(key0_{i-1}$, char)
- $Key1_i = g(key0_i, key1_{i-1})$
- $Key2_i = f(key2_{i-1}, key1_i)$
- $Key3_i = h(key2_i)$
- $Ciphertext_i$= $key3_i$ XOR $plaintext_i$

Attacks

- Original attack
  - Eli Biham and Paul C. Kocher
  - Requires 13 plaintext bytes

- ZIP Attacks with Reduced Known Plaintext
  - Michael Stay
  - Requires only 2 plaintext bytes at the cost of complexity
  - Can exploit the PRNG used by InfoZIP

- Yet another plaintext attack to ZIP encryption scheme
  - Mike Stevens and Elisa Flanders
  - Exploit in the PRNG from IBDL32.dll used by WinZip

Compressed ?

- Some files are not compressed
  - Even with maximum compression level
- Because the compression algorithm needs redundancy
- In the table: maximum size of a file so it is not compressed

	Deflate level 1-9	Bzip2
One letter	8	43
Lorem Ipsum	56	129
Kafka	64	140
Pangram	78	162
Random symbols	127	237

Values are in bytes

Plaintext

- The last byte of the encryption header
    - MSB of the data CRC
- File headers
    - Executable files
    - ZIP files
- Known files from the Internet
    - Pictures
    - Setup files

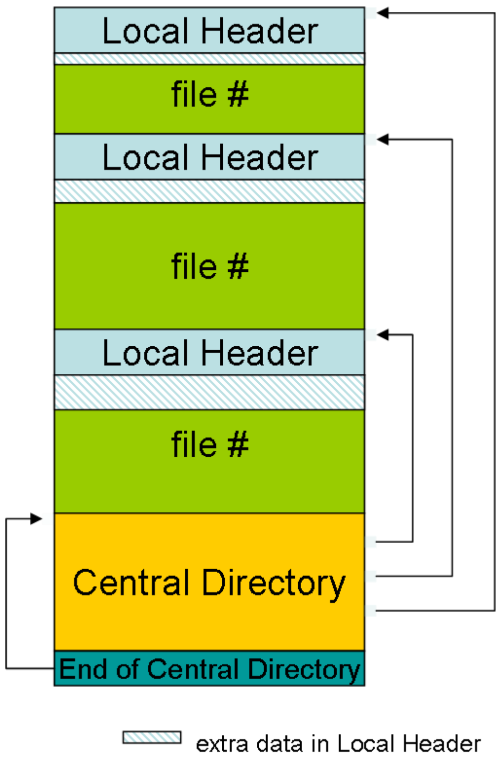
Chosen plaintext attack

- We have a list of key3's from the plaintext
- The goal is to find internal state ($key2_i$, $key1_i$, $key0_i$) for some i

1. From the list of key3's find possible key2 lists
2. For each key2 list find possible key1 lists
3. For each key1 list find one key0 list
4. Discover true key0 list

Locate data

Zip archive format



http://www.codeproject.com/Articles/8688/Extracting-files-from-a-remote-ZIP-archive

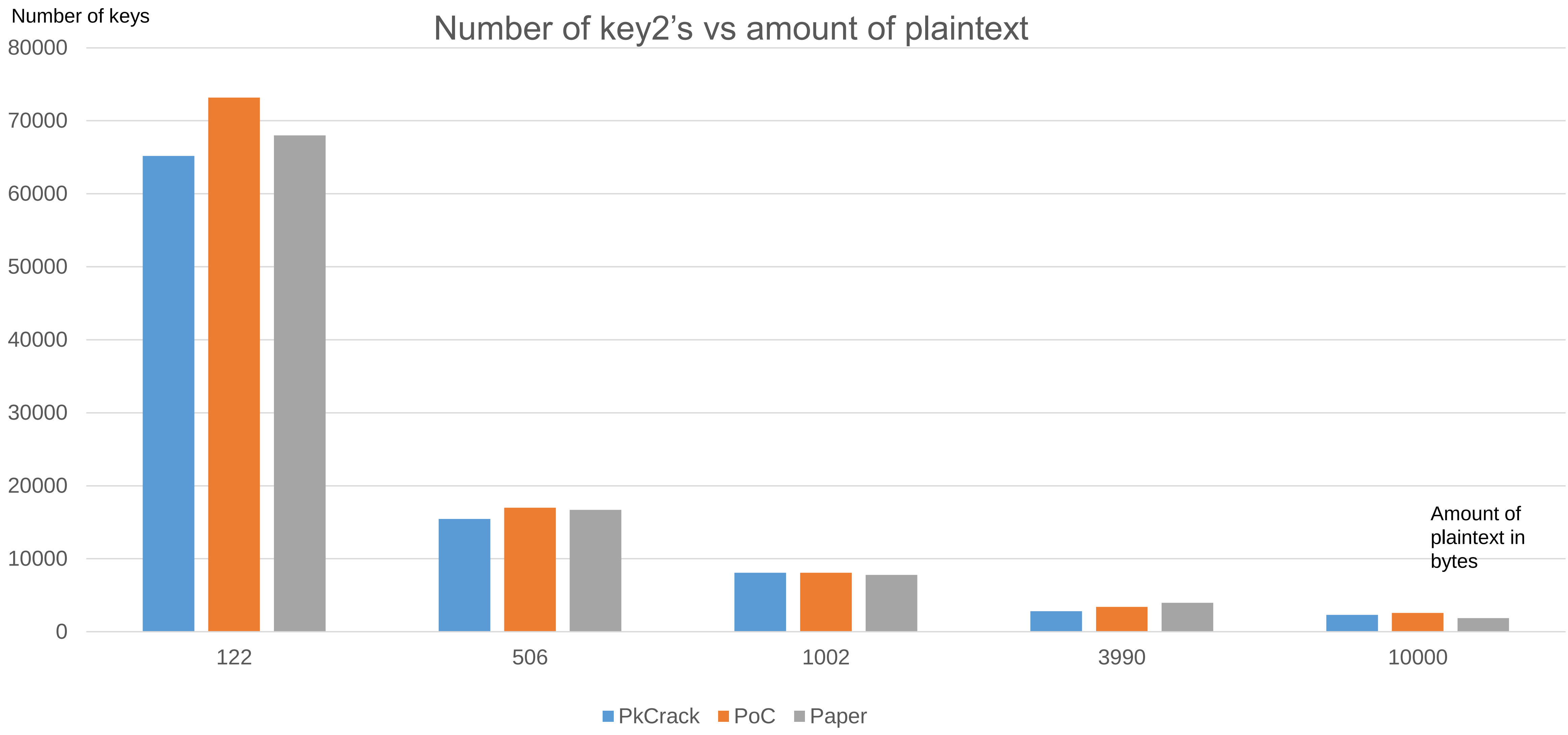
ZIP local file header

Offset	Bytes	Description[5]
0	4	Local file header signature = 0x04034b50 (read as a little-endian number)
4	2	Version needed to extract (minimum)
6	2	General purpose bit flag
8	2	Compression method
10	2	File last modification time
12	2	File last modification date
14	4	CRC-32
18	4	Compressed size
22	4	Uncompressed size
26	2	File name length (*n*)
28	2	Extra field length (*m*)
30	*n*	File name
30+*n*	*m*	Extra field

en.Wikipedia.org/wiki/Zip_(file_format)

Stage 1

- From the list of key3’s find possible key2 lists
- For efficiency precompute a number of tables as hash maps
  - The inverse of the CRC function
- Using the equations in the paper we come to 2^{22} possible $key2_n$
  - trim the plaintext and select n as 13
  - use extra plaintext to reduce the number of key2’s


Number of keys
Number of key2's vs amount of plaintext
80000
70000
60000
50000
40000
30000
20000
10000
0
122
506
1002
3990
10000
Amount of plaintext in bytes
PkCrack
PoC
Paper

Implementation

- key2 reduction is computation heavy in this stage
  - The function iterates for the number of extra plaintext bytes and returns the reduced list of keys
- Serial
- Parallel
  - Python Global Interpreter Lock does not allow use of threads in parallel
  - Use parallel processes
  - Creating new processes at every iteration
  - Using shared data between processes

Parallel

- Parallel with new processes every iteration
  - The parallel reduction computation runs 4 times faster then the serial one
  - The program as a whole runs slower as the amount of plaintext becomes larger
  - The cost of managing new processes stays constant while the gains of running parallel become smaller

- Parallel with shared data
  - 80 times slower than previous solution

Measurements

Plaintext (bytes)	Execution time Parallel (minutes)	Execution time Serial (minutes)	System/User time Parallel	System/User time Serial
40	0:34.44	1:03.6	0.0647	0.0026
122	1:08.5	1:38	0.1648	0.0017
309	1:49.3	1:56	0.3411	0.0014
506	2:29	2:07.2	0.5066	0.0012
1002	3:28	2:22	0.7455	0.0011
3990	10:07	3:02.1	1.4550	0.0009

Conclusions

- While difficult there are ways of obtaining the necessary amount of plaintext
  - Using the newer attacks, in some cases it is not even necessary
- The attack can be implemented by taking advantage of multiple cores
  - Python makes it difficult because processes must be used instead of threads

Future work

- Implement full attack and release under open source license
  - In C to take advantage of the parallel sections of the algorithm
- Compare performance with PkCrack
- Detailed analysis of the other attacks

Questions ?

Contact: Barosan.dragos@gmail.com