Privacy-Preserving Email Forensics

By

Frederik Armknecht, Andreas Dewald and Michael Gruhn

Presented At

The Digital Forensic Research Conference

DFRWS 2015 USA Philadelphia, PA (Aug 9th - 13th)

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG
FACULTY OF ENGINEERING

Privacy-Preserving Email Forensics (PPEF)

Frederik Armknecht (University of Mannheim), Andreas Dewald (Friedrich-Alexander University)

Sven Kälber (Speaker, Friedrich-Alexander University)

Agenda

- Idea, motivation & contributions
- The big picture / overall scheme design
- Own implementation details
  - Protection mechanism
  - Extraction mechanism
- Cryptographic building blocks
- Practical implementation and evaluation
- Summary & conclusion
- Limitations & future work

Idea of PPEF

- Privacy protection of employees in (large-scale) digital forensic investigations
- Revealing of only case relevant information
  - Achieved through strong cryptographic standards
- Operation principle:

1. Extraction of mailboxes
2. Encryption of all emails by applying our introduced scheme
3. Hand over of only encrypted mailboxes to third-party investigators
4. Decryption of individual emails only possible on *t* matching keywords

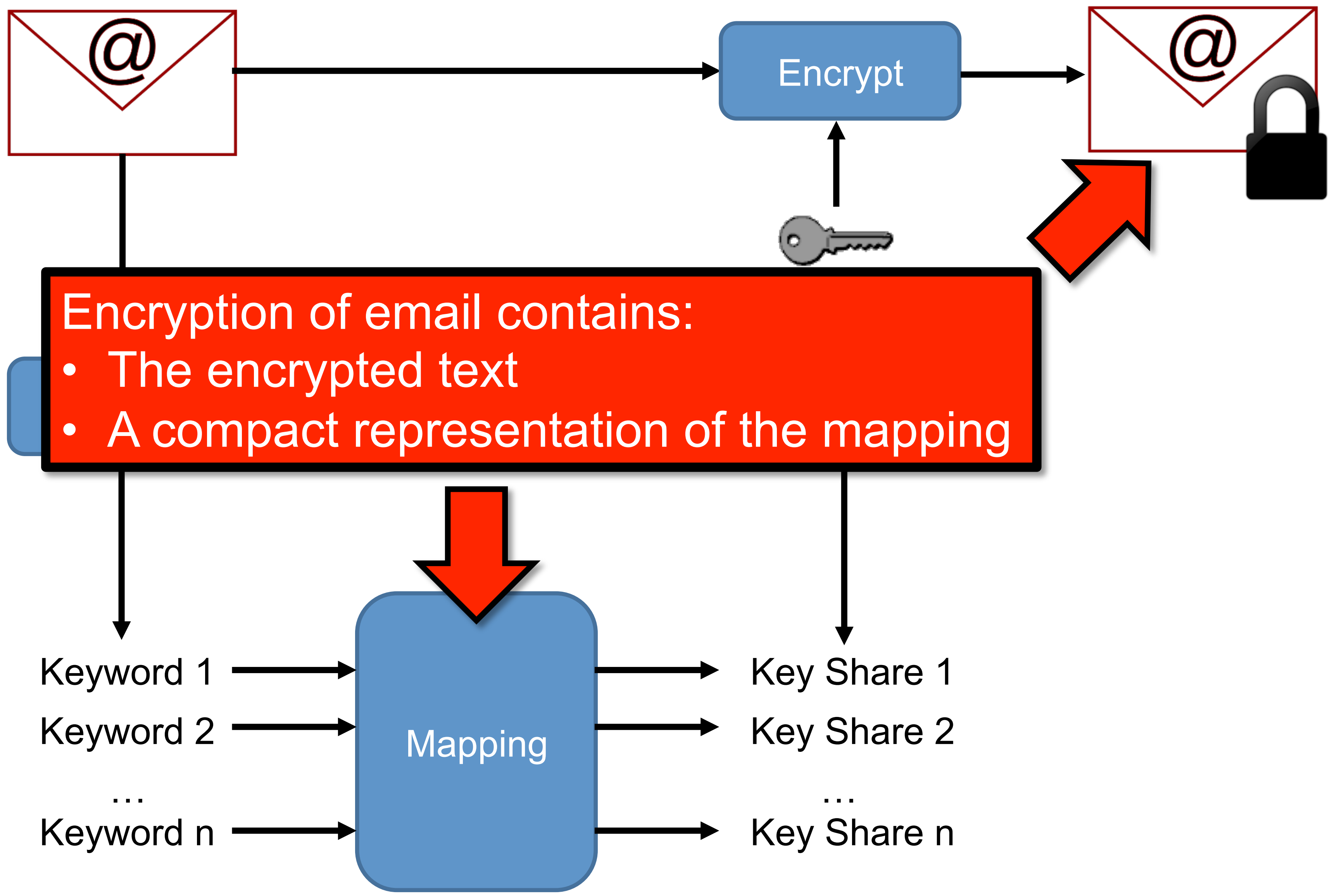
Motivation

- Private use of corporate e-mail accounts
  - Private e-mails typically contain private and very sensitive data
  - This information is often highly protected by local data protection laws
  - Typically case irrelevant information in private e-mails
- Todays approaches and tools are often limited to filtering, which is not enforced
  - Investigators might read private e-mails by accident or on purpose
- Case „United States v. Carey“ (1999)
- Problems of leaving the e-mails at the company‘s IT
  - Leaks search queries of investigators
  - Is costly and time consuming because of the high degree of interaction needed
  - Trust issues

Contributions

- Novel approach for privacy-preserving email forensics allowing for **non-interactive** theshold keyword search on **encrypted** emails
- Proof-of-concept implementation in Python and as a Autopsy v3 plug-in
- An evaluation of the practical applicability in terms of:
  - en- / decryption runtime performance
  - introduced storage overhead
  - brute-force / dicitonary attack vulnerability

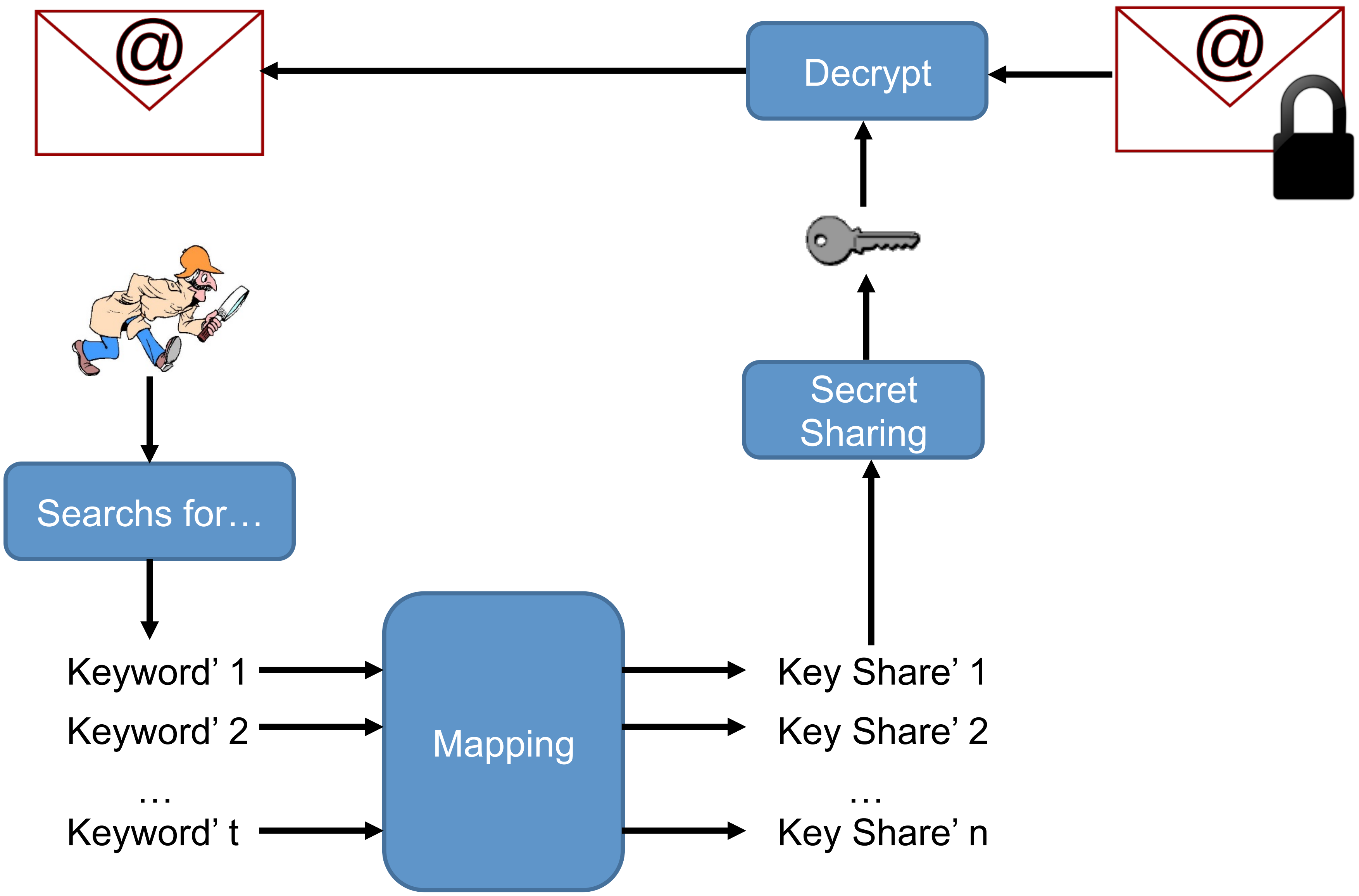
Encryption



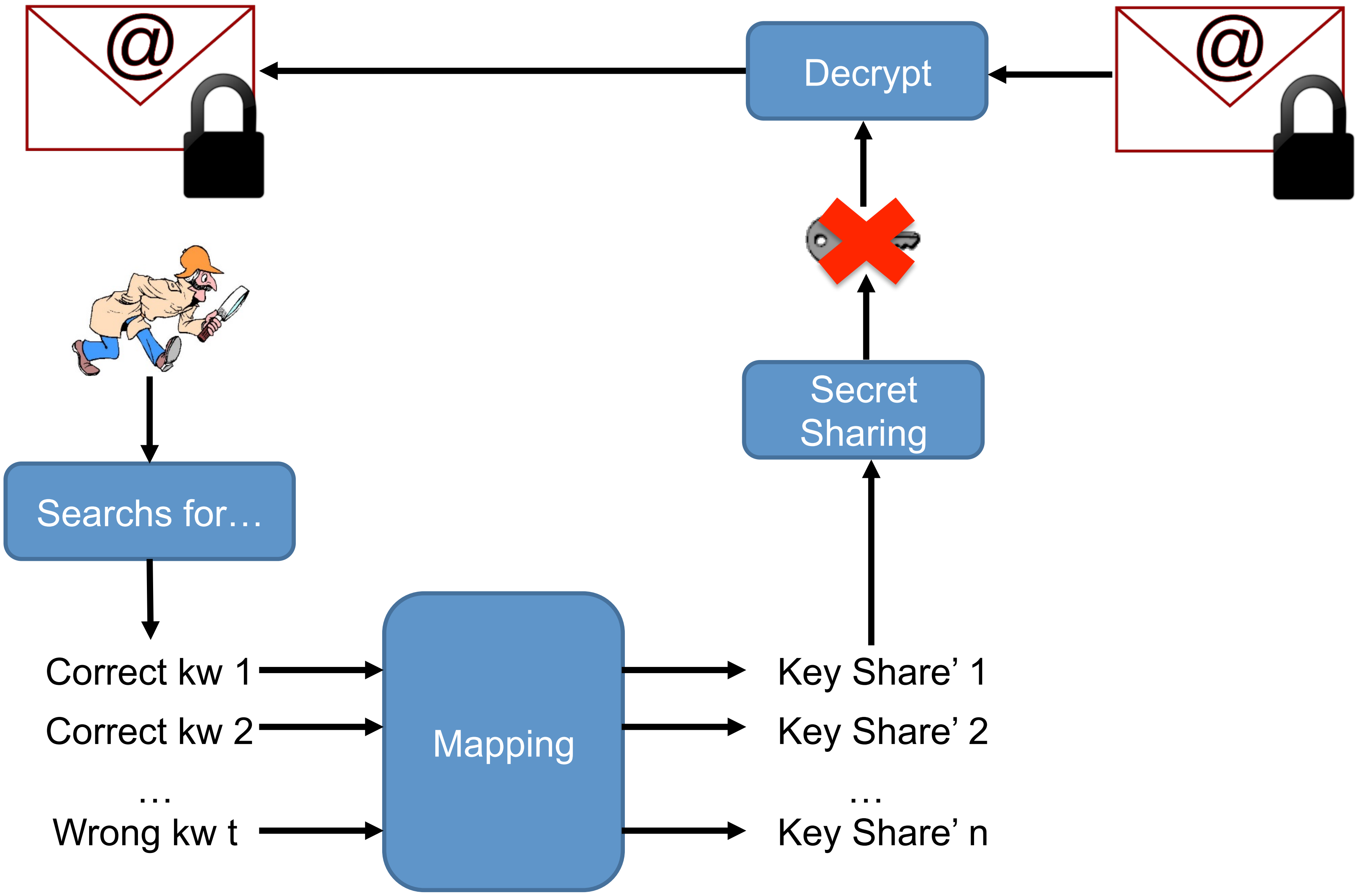
Encryption of email contains:

- The encrypted text
- A compact representation of the mapping

Decryption success



Decryption fail



Details

- Encryption of e-mails (protection mechanism):
  - Each e-mail plaintext ***P*** is encrypted to a cyphertext ***C*** with an individual secret key ***k***.
  - ***k*** gets split up in shares and might later be reconstructed
  - Support for blacklisting of commonly used words (e.g. „the“)
  - Support for whitelisting of investigation keywords (e.g. „fraud“)

- Decryption of e-mails (extraction mechanism):
  - Only possible when the e-mail in question contains at least ***t*** keywords.
  - Investigator learns nothing about the secret key of other e-mails upon successfully decrypting one e-mail.
  - Investigator learns nothing about the content of the mail if ***t-1 or less keywords*** match the content of the e-mail.

Cryptographic Building Blocks

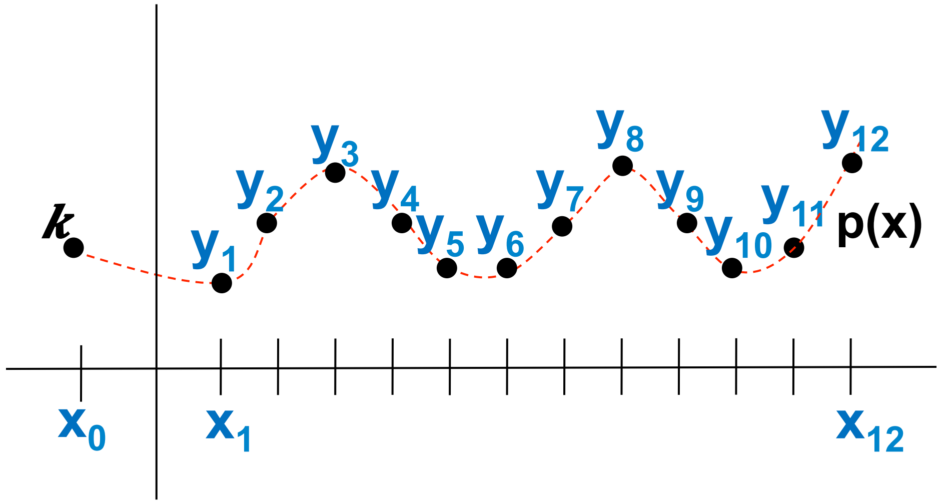
1. Encryption function:
   - AES-128 in CBC mode used for the encryption of individual e-mails
   - Add characteristic padding ***p*** as the first block to be decrypted ***(e.g. [0,...,0])***
2. Shamir's Secrect Sharing
   - Used for splitting the secret key ***k*** into shares
   - Details follow on the next slide
3. Mapping
   - Hash function: SHA-256 part of the mapping function
   - Further tweaks for efficiency reasons

Shamir‘s Secret Sharing

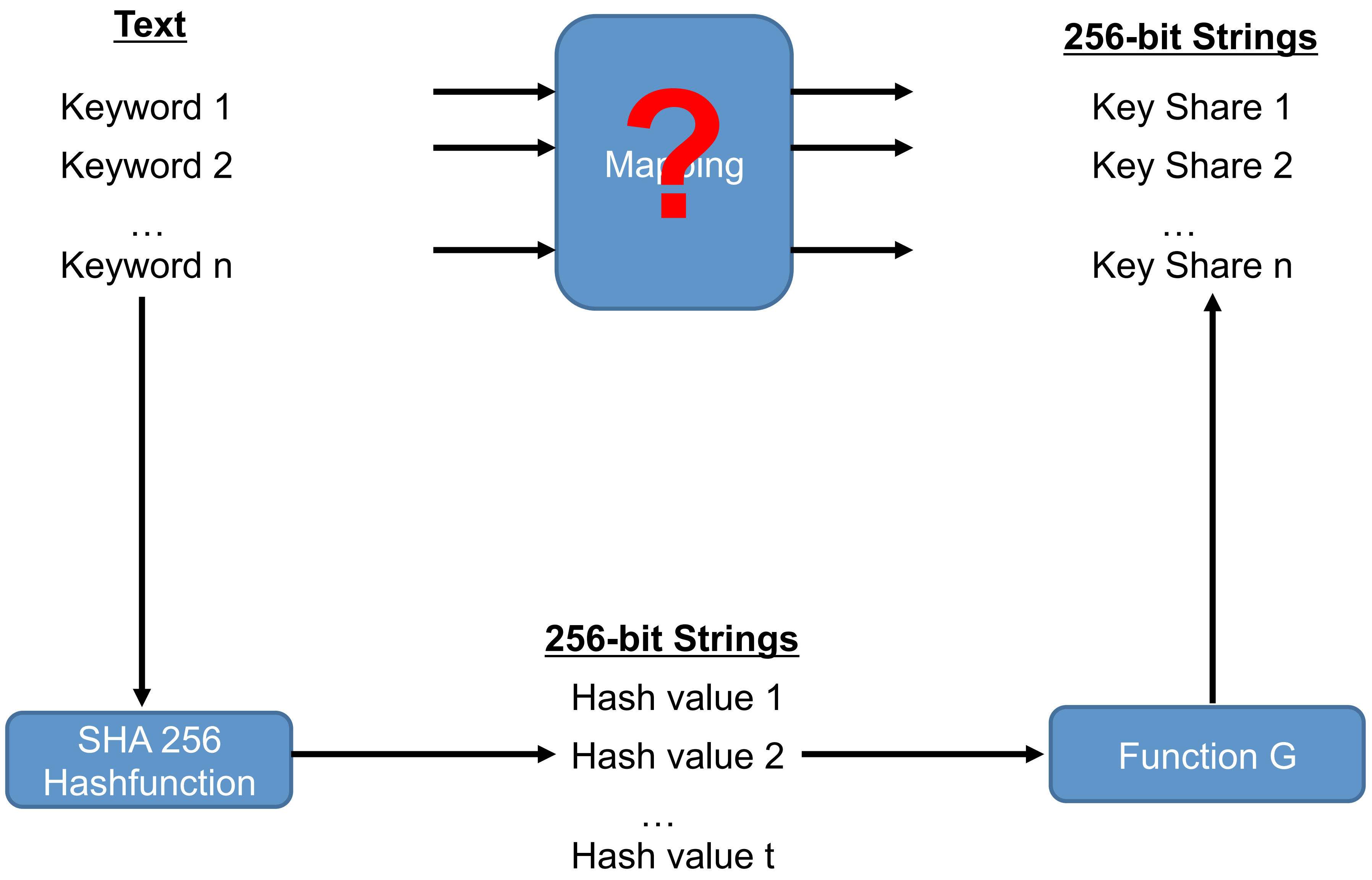
- Functionality:
  - Input: Two integers $t \leq n$, a secret k
  - Output: n shares $k_1, \dots, k_n$
- Security
  - Given at least t shares, one can reconstruct the secret
  - If less than t shares are known, reconstruction not possible
- Realization
  - Polynomial interpolation of a polynomial of degree t-1

Working Principle

- Input: Two integers t ≤ n, Secret k
- Choose polynomial $p(x)$ of degree t-1
- Compute shares: $(x_1, y_1), \ldots, (x_n, y_n)$ (here: $n=12$) with $y_i = p(x_i)$
- Reconstruction from t shares:
  - Interpolate $p(x)$
  - Compute $k = p(x_0)$



The Mapping Function



Function G

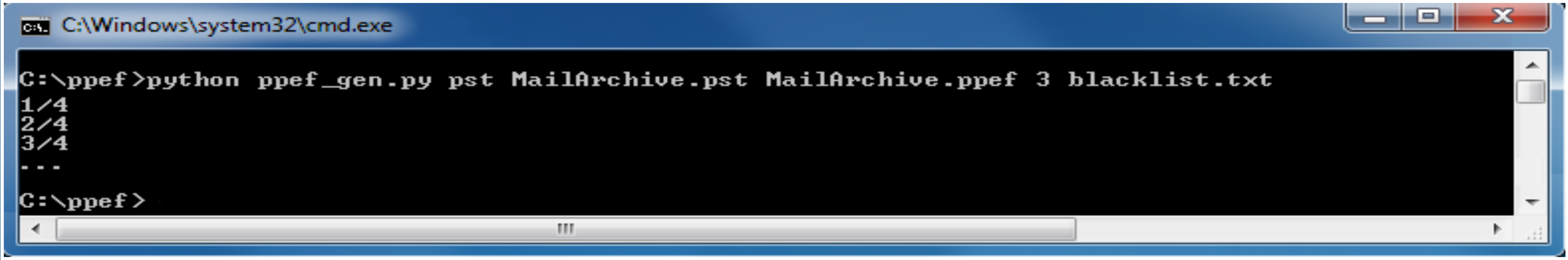
- Task: Map 256-bit hash values to 256-bit shares (x_i, y_i)
- Approach:
  - Interpret hash values as (x_i, z_i) (128-bit + 128-bit)
  - Use the values x_i to compute shares $(x_i, y_i) = (x_i, p(x_i))$
  - Find mapping $g(x)$ such that $g(x_i) = y_i \text{ XOR } z_i$
  - Function $G(h_i) = G(x_i,z_i) = (x_i, g(x_i) \text{ XOR } z_i) \rightarrow (x_i, y_i)$
- Getting the mapping:
  - Core idea: compute polynomial $g(x)$ such that $g(x_i)=y_i \text{ XOR } z_i$
  - Problem: requires to interpolate polynomial of degree n → effort is $O(n^3)$, too slow
  - Idea: Split range of x into l subsets, e.g. determined by the l last bits
  - Interpolate polynomials $g_j(x)$ for each subset
  - Effort: interpolate l polynomials, each of degree $\approx n/l$
  - Overall effort: $l \cdot (n / l)^3 = n^3 / l^2$

Practical implementation

1. Python en- / decrytion of mailboxes

   Supported mailbox formats:

   - mbox
   - pst
   - MH (RFC 822)
   - Maildir

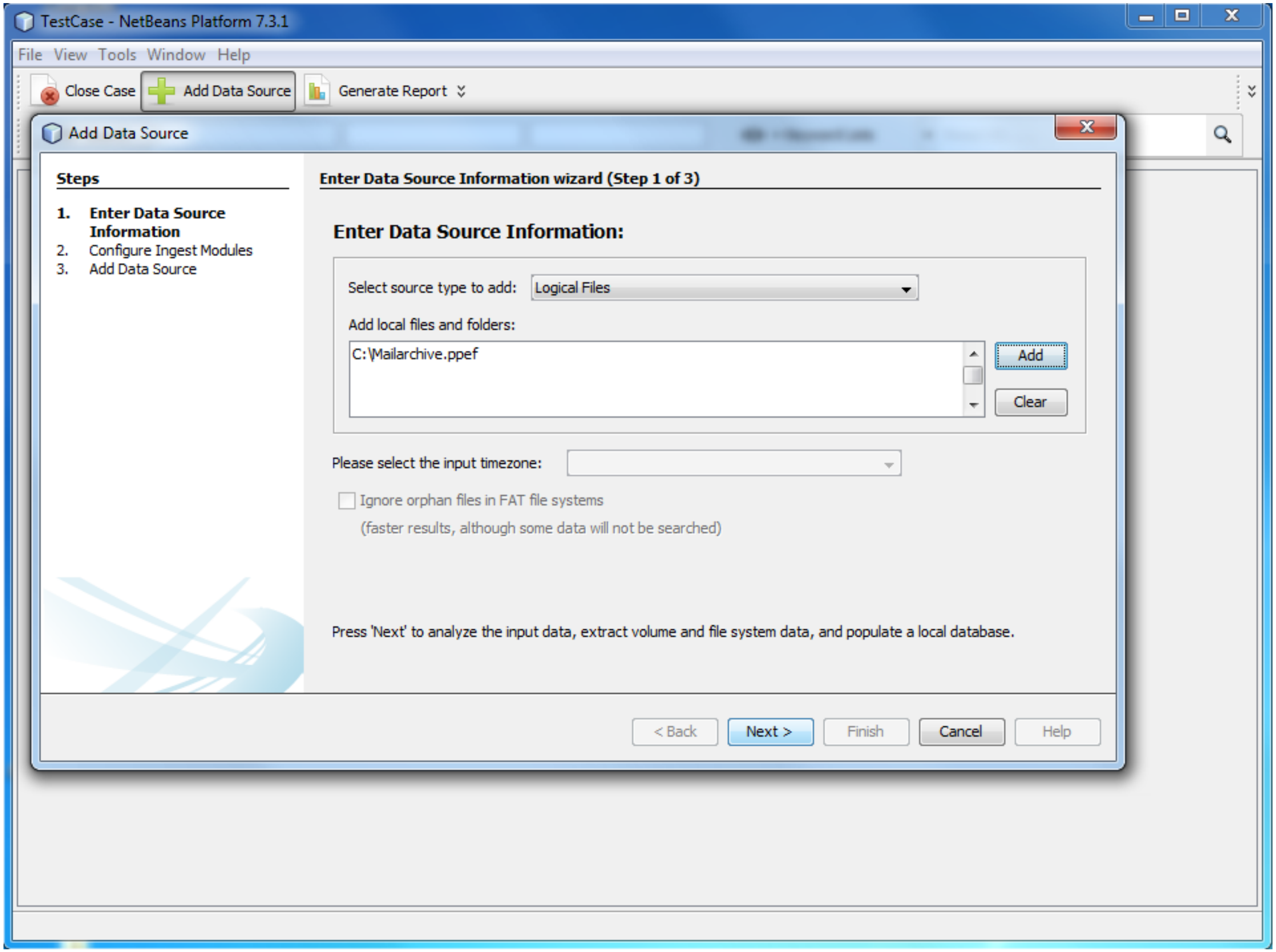


```
C:\ppef>python ppef_gen.py pst MailArchive.pst MailArchive.ppef 3 blacklist.txt
1/4
2/4
3/4
...

C:\ppef>
```

2. PPEF plugin for Autopsy v3

PPEF Autopsy plug-in


TestCase - NetBeans Platform 7.3.1
File View Tools Window Help
Close Case
Add Data Source
Generate Report
Add Data Source
Steps
1. Enter Data Source Information
2. Configure Ingest Modules
3. Add Data Source
Add
Clear
< Back
Next >
Finish
Cancel
Help

Enter Data Source Information wizard (Step 1 of 3)

Enter Data Source Information:

Select source type to add: Logical Files

Add local files and folders:

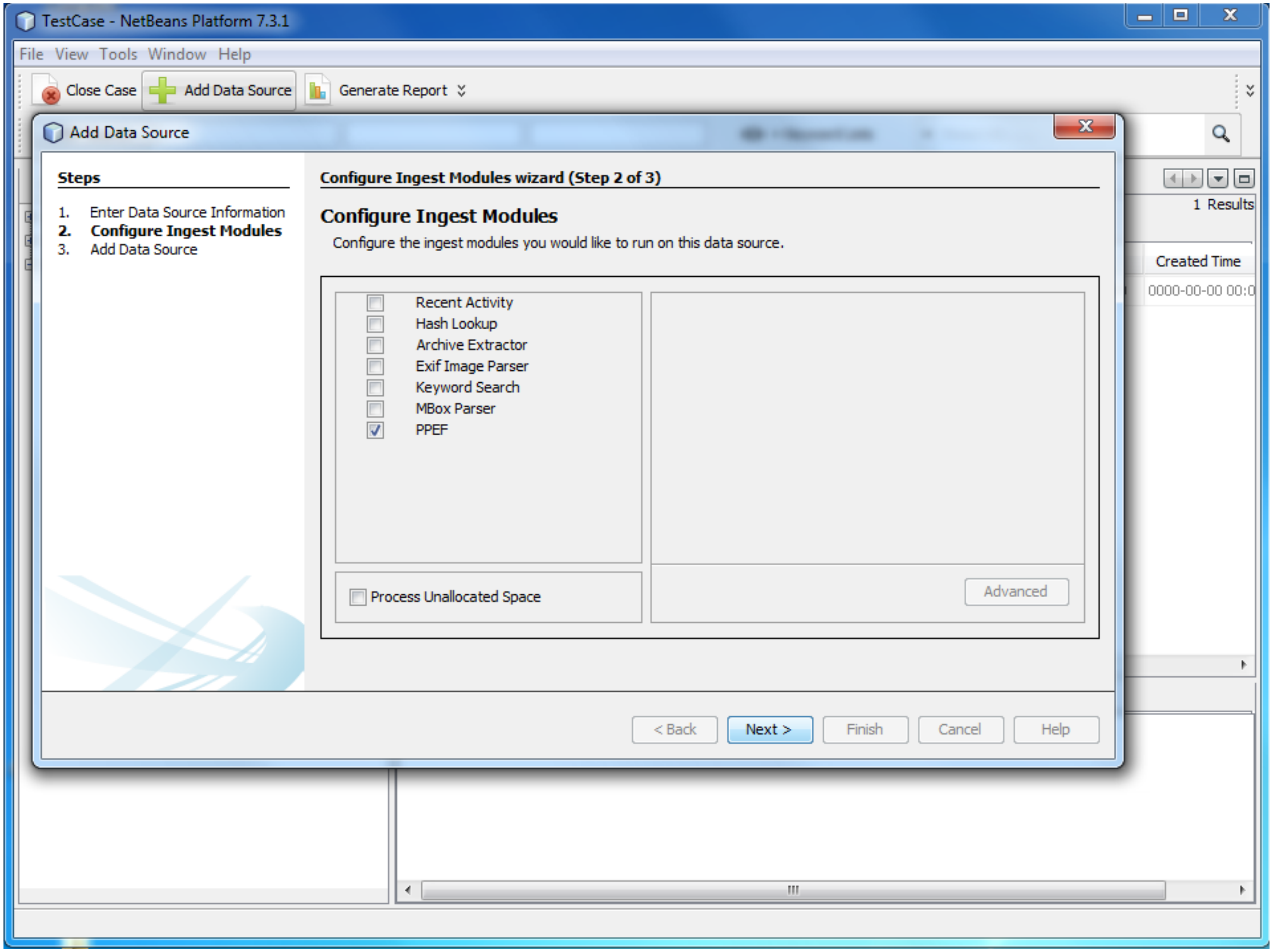
C:\Mailarchive.ppef

Please select the input timezone:

Ignore orphan files in FAT file systems
(faster results, although some data will not be searched)

Press 'Next' to analyze the input data, extract volume and file system data, and populate a local database.

PPEF Autopsy plug-in



Steps

1. Enter Data Source Information
2. **Configure Ingest Modules**
3. Add Data Source

Configure Ingest Modules wizard (Step 2 of 3)

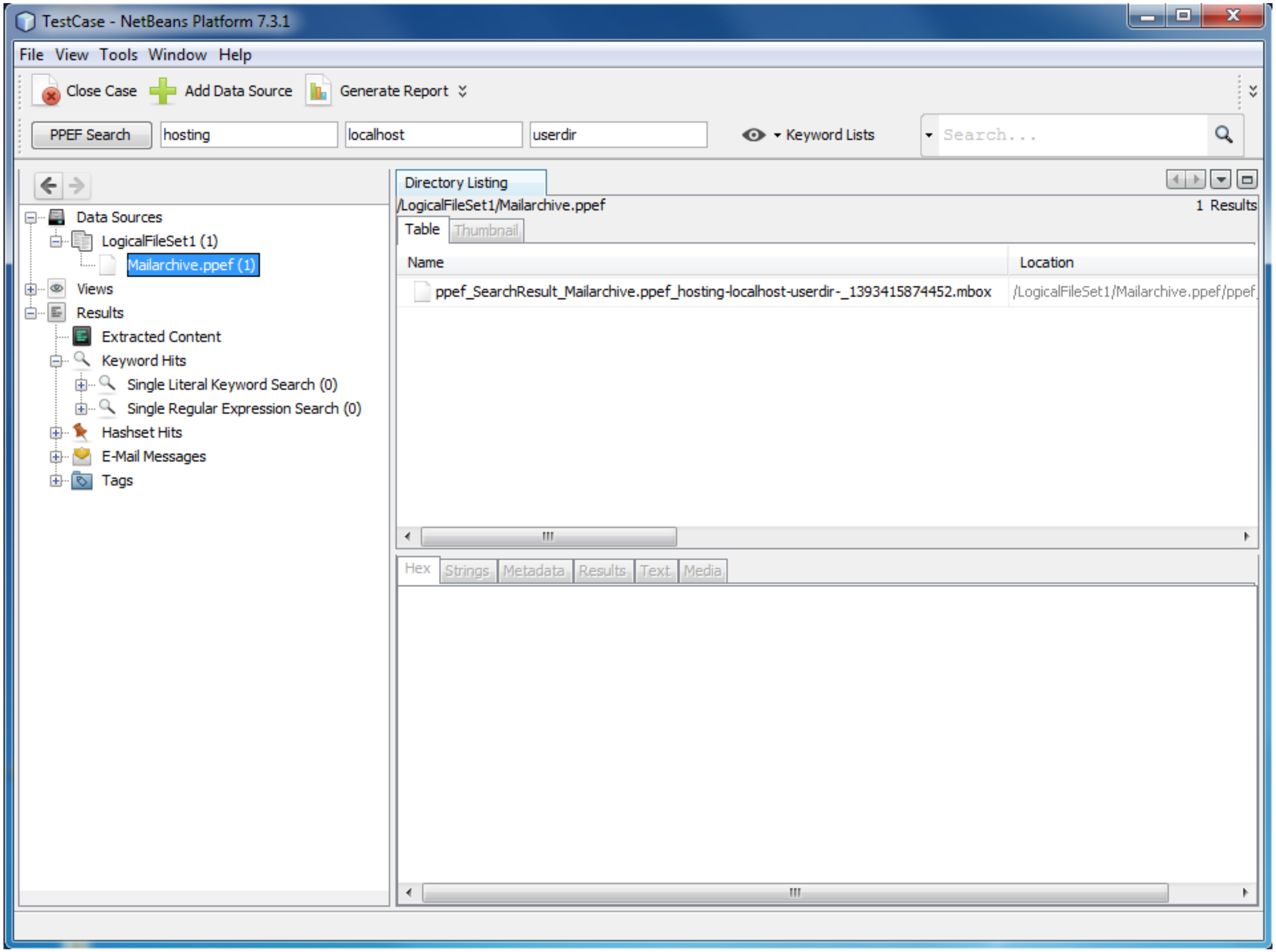
Configure Ingest Modules

Configure the ingest modules you would like to run on this data source.

- [ ] Recent Activity
- [ ] Hash Lookup
- [ ] Archive Extractor
- [ ] Exif Image Parser
- [ ] Keyword Search
- [ ] MBox Parser
- [✓] PPEF

- [ ] Process Unallocated Space

PPEF Autopsy plug-in



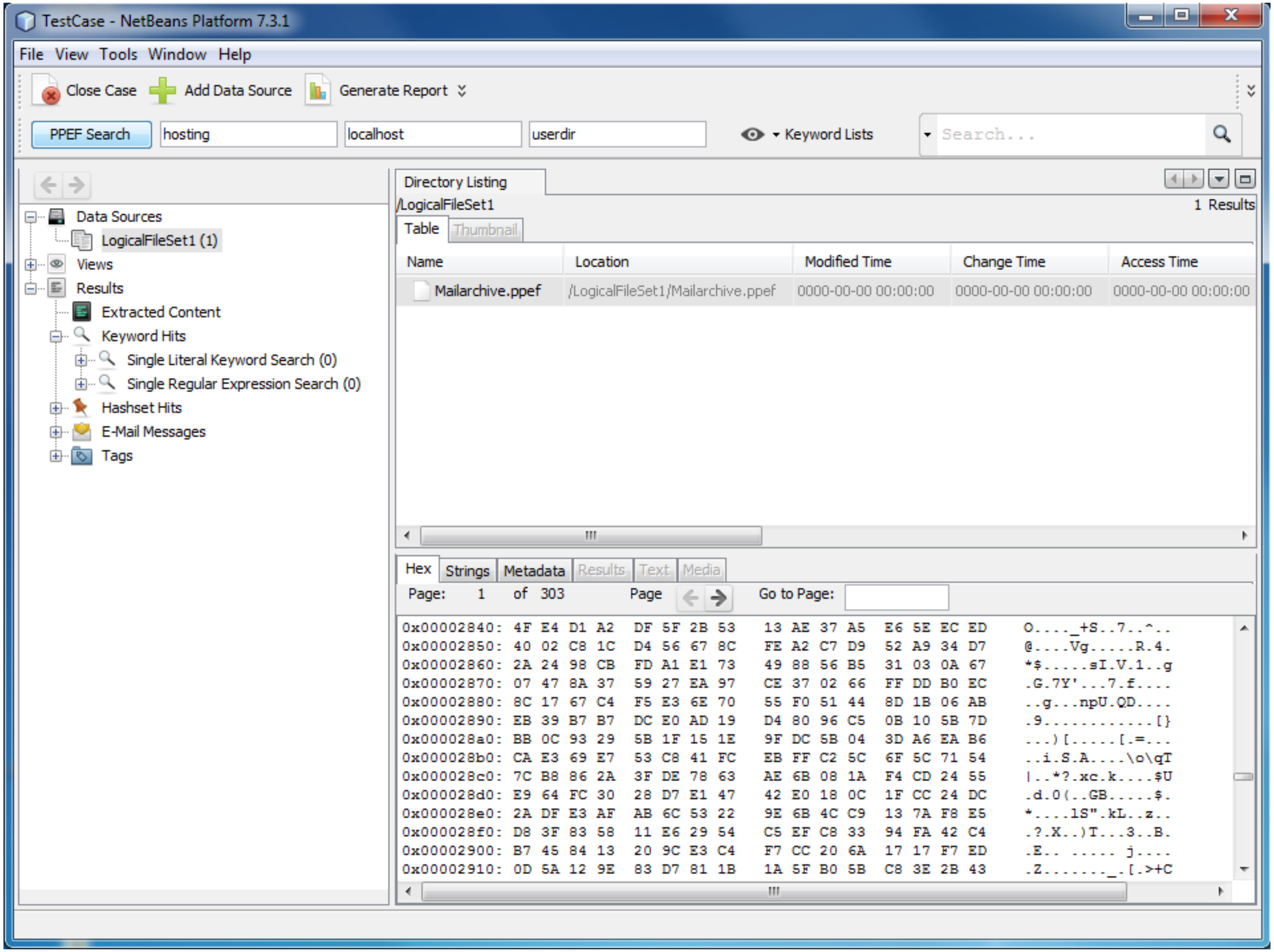
TestCase - NetBeans Platform 7.3.1

/LogicalFileSet1/Mailarchive.ppef

1 Results

Name	Location
ppef_SearchResult_Mailarchive.ppef_hosting-localhost-userdir-_1393415874452.mbox	/LogicalFileSet1/Mailarchive.ppef/ppef

PPEF Autopsy plug-in



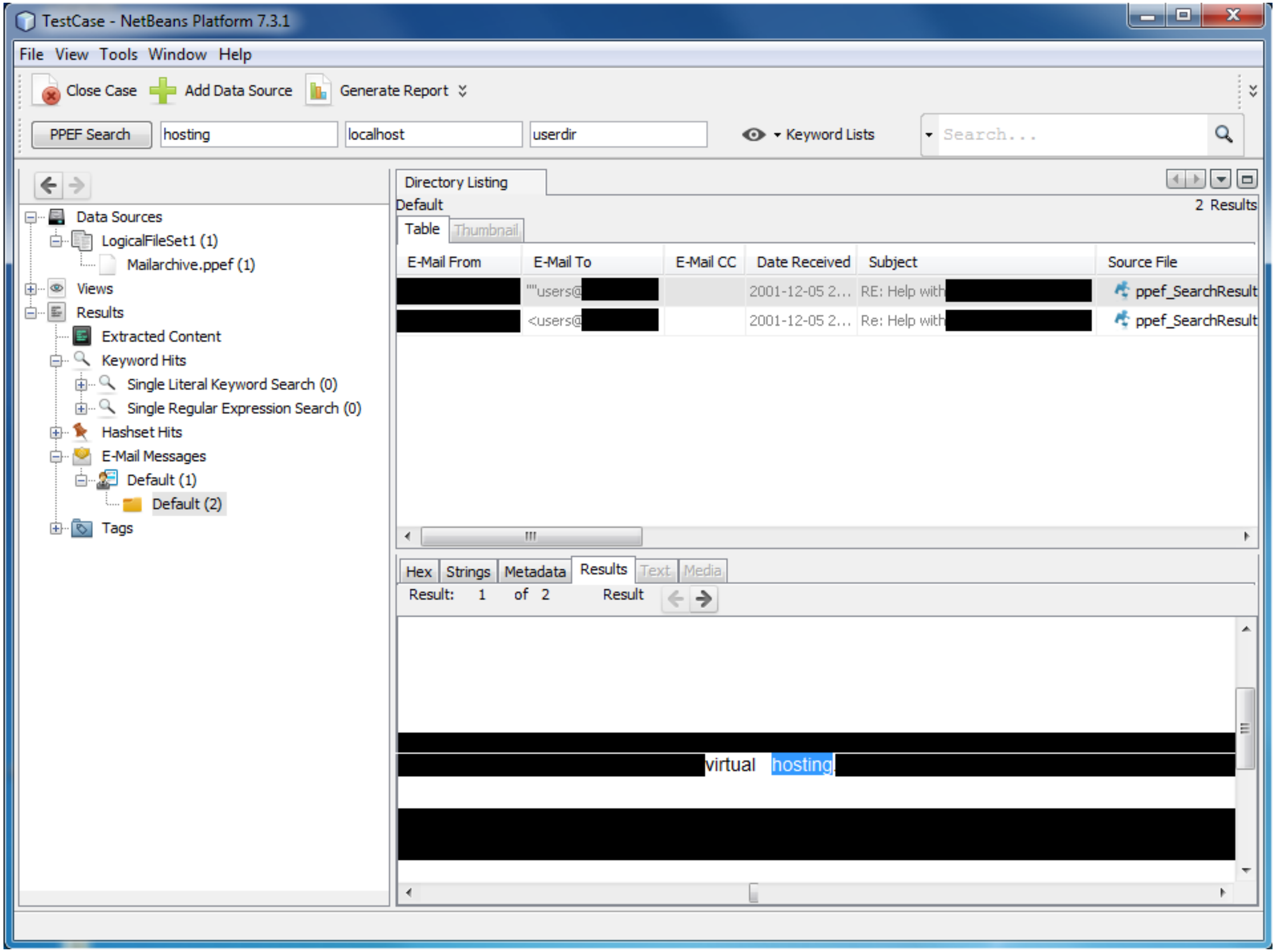
PPEF Search: hosting | localhost | userdir

Name	Location	Modified Time	Change Time	Access Time
Mailarchive.ppef	/LogicalFileSet1/Mailarchive.ppef	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00

Page: 1 of 303 Page Go to Page:

```
0x00002840: 4F E4 D1 A2  DF 5F 2B 53  13 AE 37 A5  E6 5E EC ED   O...._+S..7..^..
0x00002850: 40 02 C8 1C  D4 56 67 8C  FE A2 C7 D9  52 A9 34 D7   @....Vg.....R.4.
0x00002860: 2A 24 98 CB  FD A1 E1 73  49 88 56 B5  31 03 0A 67   *$.....sI.V.1..g
0x00002870: 07 47 8A 37  59 27 EA 97  CE 37 02 66  FF DD B0 EC   .G.7Y'...7.f....
0x00002880: 8C 17 67 C4  F5 E3 6E 70  55 F0 51 44  8D 1B 06 AB   ..g...npU.QD....
0x00002890: EB 39 B7 B7  DC E0 AD 19  D4 80 96 C5  0B 10 5B 7D   .9............[}
0x000028a0: BB 0C 93 29  5B 1F 15 1E  9F DC 5B 04  3D A6 EA B6   ...)[.....[.=...
0x000028b0: CA E3 69 E7  53 C8 41 FC  EB FF C2 5C  6F 5C 71 54   ..i.S.A....\o\qT
0x000028c0: 7C B8 86 2A  3F DE 78 63  AE 6B 08 1A  F4 CD 24 55   |..*?.xc.k....$U
0x000028d0: E9 64 FC 30  28 D7 E1 47  42 E0 18 0C  1F CC 24 DC   .d.0(..GB.....$.
0x000028e0: 2A DF E3 AF  AB 6C 53 22  9E 6B 4C C9  13 7A F8 E5   *....lS".kL..z..
0x000028f0: D8 3F 83 58  11 E6 29 54  C5 EF C8 33  94 FA 42 C4   .?.X..)T...3..B.
0x00002900: B7 45 84 13  20 9C E3 C4  F7 CC 20 6A  17 17 F7 ED   .E.. ..... j....
0x00002910: 0D 5A 12 9E  83 D7 81 1B  1A 5F B0 5B  C8 3E 2B 43   .Z......._.[.>+C
```

PPEF Autopsy plug-in



TestCase - NetBeans Platform 7.3.1

File View Tools Window Help

Close Case | Add Data Source | Generate Report

PPEF Search | hosting | localhost | userdir | Keyword Lists | Search...

Data Sources
- LogicalFileSet1 (1)
  - Mailarchive.ppef (1)

Views

Results
- Extracted Content
- Keyword Hits
  - Single Literal Keyword Search (0)
  - Single Regular Expression Search (0)
- Hashset Hits
- E-Mail Messages
  - Default (1)
    - Default (2)
- Tags

Directory Listing

Default — 2 Results

Table | Thumbnail

E-Mail From	E-Mail To	E-Mail CC	Date Received	Subject	Source File
[redacted]	""users@[redacted]		2001-12-05 2...	RE: Help with[redacted]	ppef_SearchResult
[redacted]	<users@[redacted]		2001-12-05 2...	Re: Help with[redacted]	ppef_SearchResult

Hex | Strings | Metadata | Results | Text | Media

Result: 1 of 2 Result

[redacted]

[redacted] virtual hosting [redacted]

[redacted]

Evaluation

The data set used in our evaluation consists of 5 different mailboxes:

- *Apache* – httpd user mailing list (75724 e-mails)
- *Work* – personal work e-mails (1590 e-mails)
- *A, B, C* – private e-mail accounts (511, 349, 83 e-mails)

Evaluations:

1. Encryption runtime performance
2. Encryption storage overhead
3. Search / decryption runtime performance
4. Brute-force attack performance

Encryption Performance

- Time (in seconds) to encrypt the corresponding emails of each account
- Average encryption rate: 13.5 emails/sec
  - Encryption of large mailboxes might take several hours (< 2h for 75724 e-mails) , but only needs to be done once!

	Apache [s]	Work	A	B	C
Min	0.004	0.005	0.005	0.005	0.005
Max	31.745	1.403	1.932	1.117	0.460
Avg	0.082	0.136	0.122	0.110	0.173
Med	0.072	0.115	0.101	0.074	0.150
σ	0.133	0.120	0.132	0.153	0.071
Σ	6243.511	217.242	62.842	38.535	14.367

Encryption storage overhead

- Encryption with AES does not add much storage overhead (33 – 48 bytes per mail)
- Main storage overhead factor is the mapping function (on average 582.4 bytes per mail)
- Average storage overhead: 5.2 %

	Apache	Work	A	B	C
Size Raw [KB]	376, 551	418, 680	16, 386	47, 486	6, 676
Size PPEF [KB]	418, 870	420, 418	16, 885	47, 821	6, 806
Overhead	11.2 %	0.4 %	3.0 %	0.7 %	1.9 %

Search / decryption performance

- Time (in seconds) to search each e-mail for 3 keywords and decrypt matching e-mails
- Average search and decryption rate: 98 mails/sec
  - Searches on large mailboxes take time (< 15min) but are still feasible

	Apache [s]	Work	A	B	C
Min	0.0090	0.0096	0.0098	0.0097	0.0098
Max	0.0598	0.1645	0.0139	0.1508	0.0148
Avg	0.0115	0.0137	0.0114	0.0123	0.0117
Med	0.0115	0.0117	0.0113	0.0113	0.0116
σ	0.0007	0.0103	0.0007	0.0086	0.0009
Σ	876.8591	21.7977	5.8650	4.2982	0.9750

Attack performance evaluation

- Brute-force attacks to decrypt the whole mailbox (π=0.99) or a random half of the mailbox (π=0.5)
- Using 4 different vocabularies
  - Oxford English Dictionary (171,476 words)
  - 50 % of the Oxford English Dictionary (85,738 words)
  - Vocabulary in daily speech edu. person (20,000 words)
  - Vocabulary of uneducated person (10,000 words)

π	N	Apache	Work	A	B	C
0.99	171,476	$1.15 \cdot 10^8$	$3.26 \cdot 10^5$	$1.23 \cdot 10^5$	$1.17 \cdot 10^5$	5,373.15
0.50	171,476	$1.73 \cdot 10^7$	49,072.84	18,565.34	17,638.94	808.74
0.99	85,738	$1.44 \cdot 10^7$	40,753.36	15,418.00	14,648.51	671.63
0.50	85,738	$2.17 \cdot 10^6$	6,133.99	2,320.64	2,204.82	101.09
0.99	20,000	$1.83 \cdot 10^5$	517.23	195.68	185.91	8.52
0.50	20,000	27,510.37	77.85	29.45	27.98	1.28
0.99	10,000	22,843.44	64.64	24.46	23.24	1.07
0.50	10,000	3,438.28	9.73	3.68	3.50	0.16

Summary / Conclusion

- We proposed a novel approach for privacy-preserving email forensics allowing for **non-interactive** theshold keyword search on **encrypted** e-mails.
- We developed a **prototype-implementation** in Python and an **Autopsy plug-in** that supports multiple well-known mailbox formats.
- We **evaluated the practical applicability** in terms of en- / decryption performance, storage overhead and brute-force vulnerability.
  - Sufficiently large mailboxes are well protected against dictionary (brute-force) attacks

Limitations / Future work

Limitations:

- Scheme based on keyword searches, therefore prone to spelling errors
- No wildcard operator or regular expression possible that allows for more advanced search queries
- Brute-force / dictionary attacks possible

Future work:

- Support for wildcard usage within the search keywords

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG
FACULTY OF ENGINEERING

Thank you for your attention!

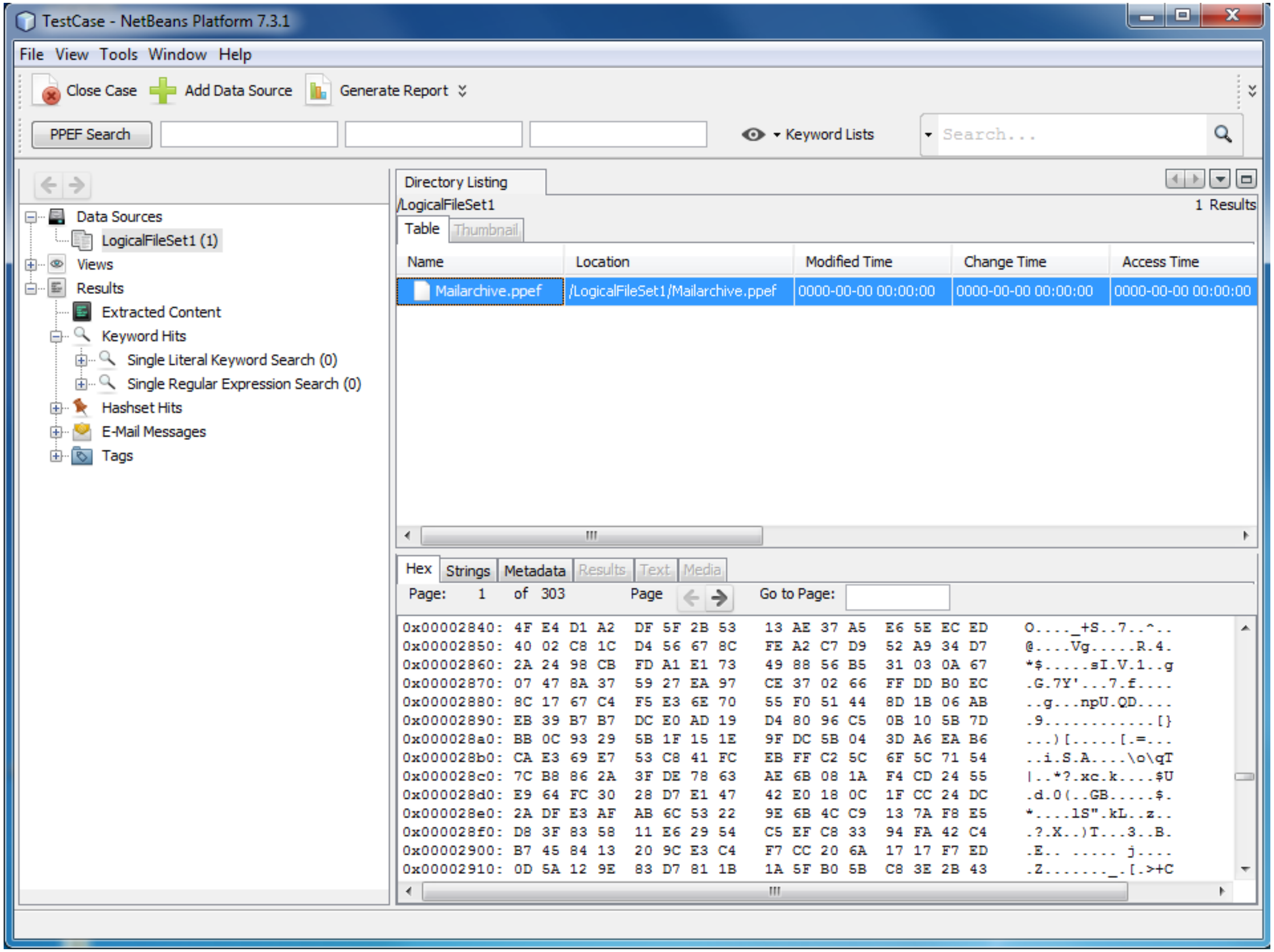
Speaker:

Sven Kälber - sven.kaelber@cs.fau.de

Authors:

Frederik Armknecht – armknecht@uni-mannheim.de

Andreas Dewald – andreas.dewald@fau.de



/LogicalFileSet1 — 1 Results

Name	Location	Modified Time	Change Time	Access Time
Mailarchive.ppef	/LogicalFileSet1/Mailarchive.ppef	0000-00-00 00:00:00	0000-00-00 00:00:00	0000-00-00 00:00:00

Page: 1 of 303 Page Go to Page:

```
0x00002840: 4F E4 D1 A2  DF 5F 2B 53   13 AE 37 A5  E6 5E EC ED   O...._+S..7..^..
0x00002850: 40 02 C8 1C  D4 56 67 8C   FE A2 C7 D9  52 A9 34 D7   @....Vg.....R.4.
0x00002860: 2A 24 98 CB  FD A1 E1 73   49 88 56 B5  31 03 0A 67   *$.....sI.V.1..g
0x00002870: 07 47 8A 37  59 27 EA 97   CE 37 02 66  FF DD B0 EC   .G.7Y'...7.f....
0x00002880: 8C 17 67 C4  F5 E3 6E 70   55 F0 51 44  8D 1B 06 AB   ..g...npU.QD....
0x00002890: EB 39 B7 B7  DC E0 AD 19   D4 80 96 C5  0B 10 5B 7D   .9............[}
0x000028a0: BB 0C 93 29  5B 1F 15 1E   9F DC 5B 04  3D A6 EA B6   ...)[.....[.=...
0x000028b0: CA E3 69 E7  53 C8 41 FC   EB FF C2 5C  6F 5C 71 54   ..i.S.A....\o\qT
0x000028c0: 7C B8 86 2A  3F DE 78 63   AE 6B 08 1A  F4 CD 24 55   |..*?.xc.k....$U
0x000028d0: E9 64 FC 30  28 D7 E1 47   42 E0 18 0C  1F CC 24 DC   .d.0(..GB.....$.
0x000028e0: 2A DF E3 AF  AB 6C 53 22   9E 6B 4C C9  13 7A F8 E5   *....lS".kL..z..
0x000028f0: D8 3F 83 58  11 E6 29 54   C5 EF C8 33  94 FA 42 C4   .?.X..)T...3..B.
0x00002900: B7 45 84 13  20 9C E3 C4   F7 CC 20 6A  17 17 F7 ED   .E.. ..... j....
0x00002910: 0D 5A 12 9E  83 D7 81 1B   1A 5F B0 5B  C8 3E 2B 43   .Z......._.[.>+C
```

Function G

- Task: Map 256-bit hash values to 256-bit key shares
- Approach
  - Interpret hash values as (128 bit + 128 bit)
  - Use the values to compute key shares
  - Find a mapping such that
  - Function
- Getting the Mapping
  - Core idea: compute polynomial such that
  - Problem: requires to interpolate polynomial of degree n effort is in , too
  - Idea: Split range of into subsets, e.g., determined by the first bits
  - Interpolate polynomials for each subset
  - Effort: interpolate polynomials, each of degree .
  - Overall effort: