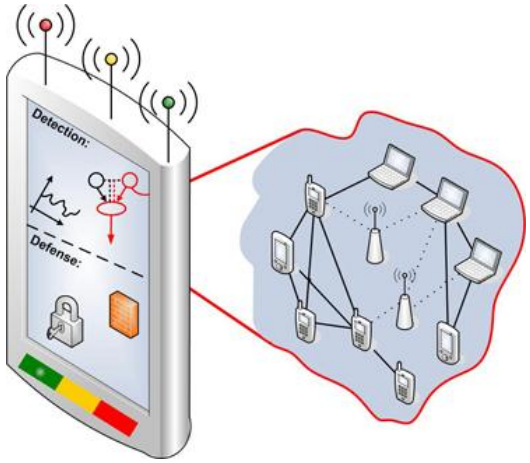
Automatic Detection of Embedded, Unwanted Binary Code for Mobiles

als Teil des SKIMS Projekts

Von Benjamin Jochheim

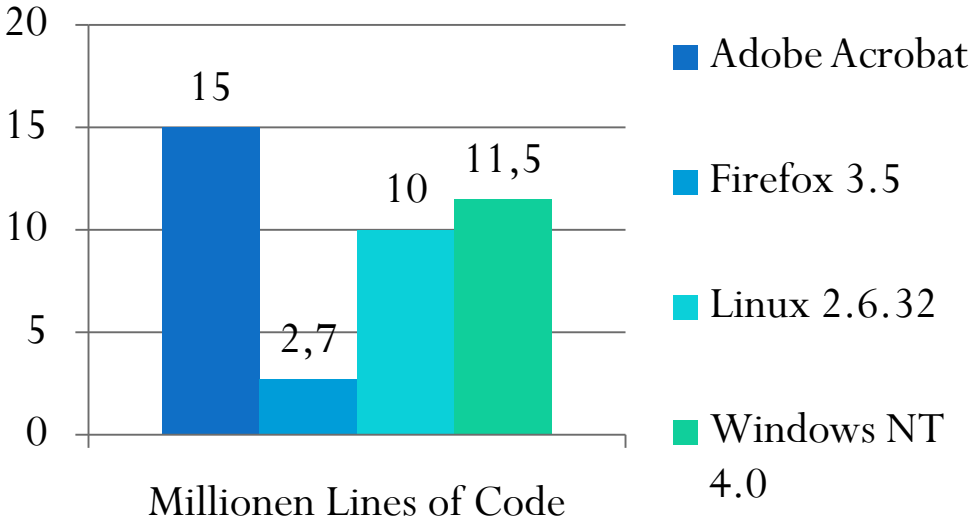


Gliederung

- Motivation am Beispiel PDF
- Herkömmliche Malware-Erkennung
- Ein Leichtgewichtiges Verfahren
- Merkmalsextraktion
- Klassifikator
- Testdatenbeschaffung
- Testdatenaufbereitung
- Optimierung des Verfahrens
- Ergebnisse
- Ausblicks

Einbettende Formate am Beispiel PDF

- ISO 32000-1:2008 Standard
- Adobe Acrobat kann mehr als erwartet
  - OpenGL Rendering innerhalb von Dokumenten
  - Eingebettetes Flash
  - Eingebetteten Sound und Videos abspielen
- Unterschiedliche Ausgabe pro Gerätetyp möglich



Quelle: [10,4]

Herkömmliche Malware-Erkennung

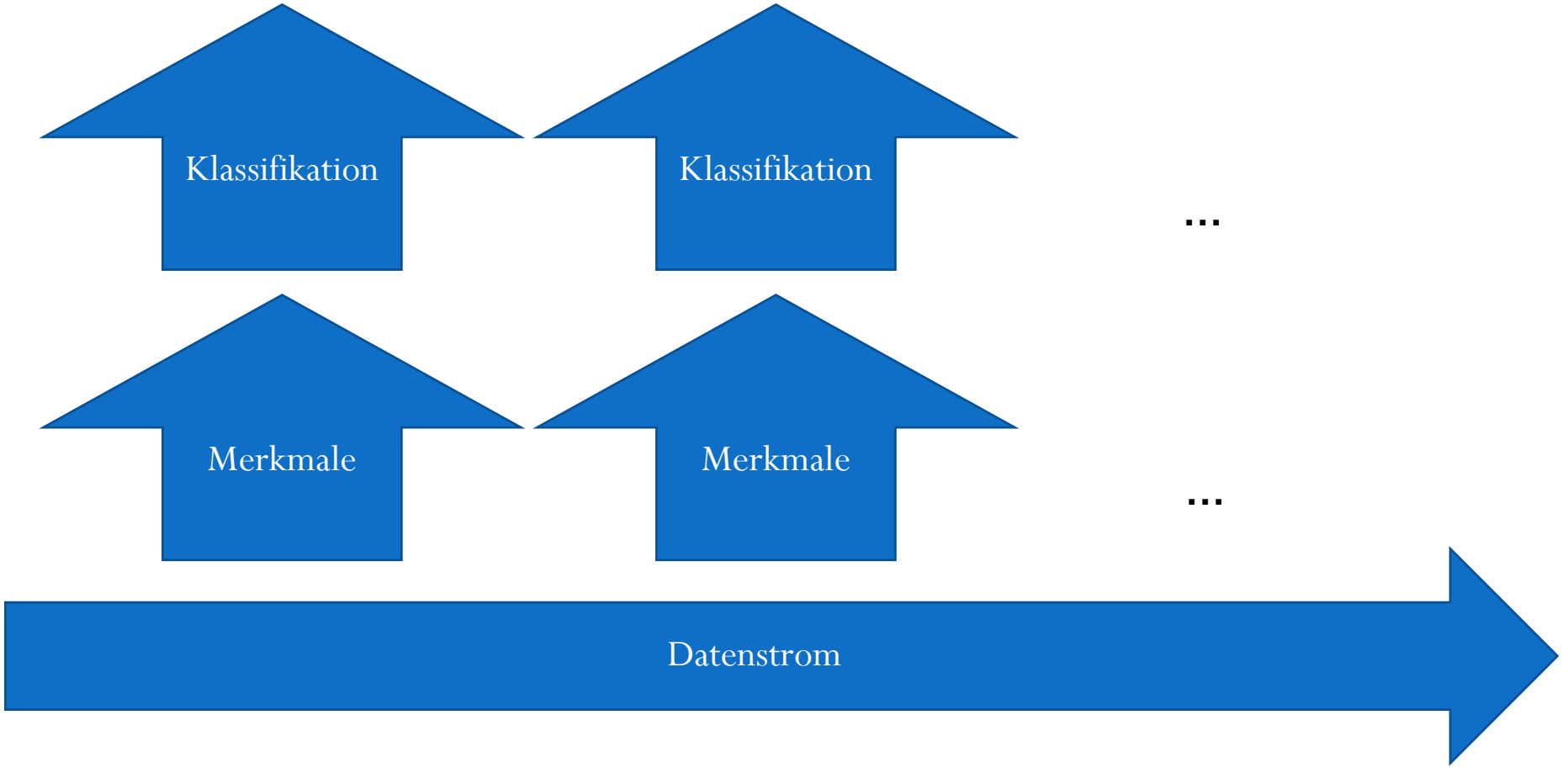
- Offline-Verfahren
  - Hashes/Signaturen
  - Heuristische Verfahren
  - Parser
- Online Verfahren
  - Verhaltensüberwachung
    - API-Aufrufe
    - Auffällige Nutzung der Systemresourcen
  - Sandbox

Ein Leichtgewichtiges Verfahren (1)

- Leichtgewichtig
- Zero Knowledge Verfahren
- Identifikation kleiner Abschnitte in Dateien
- Keine Dateiidentifikation sondern nur binary VS. nonbinary unterscheiden
- Kontextabhängigkeit
  - Verfahren (später) mit Kenntnis des Dateityps
- Zustandslos

Ein Leichtgewichtiges Verfahren (2)

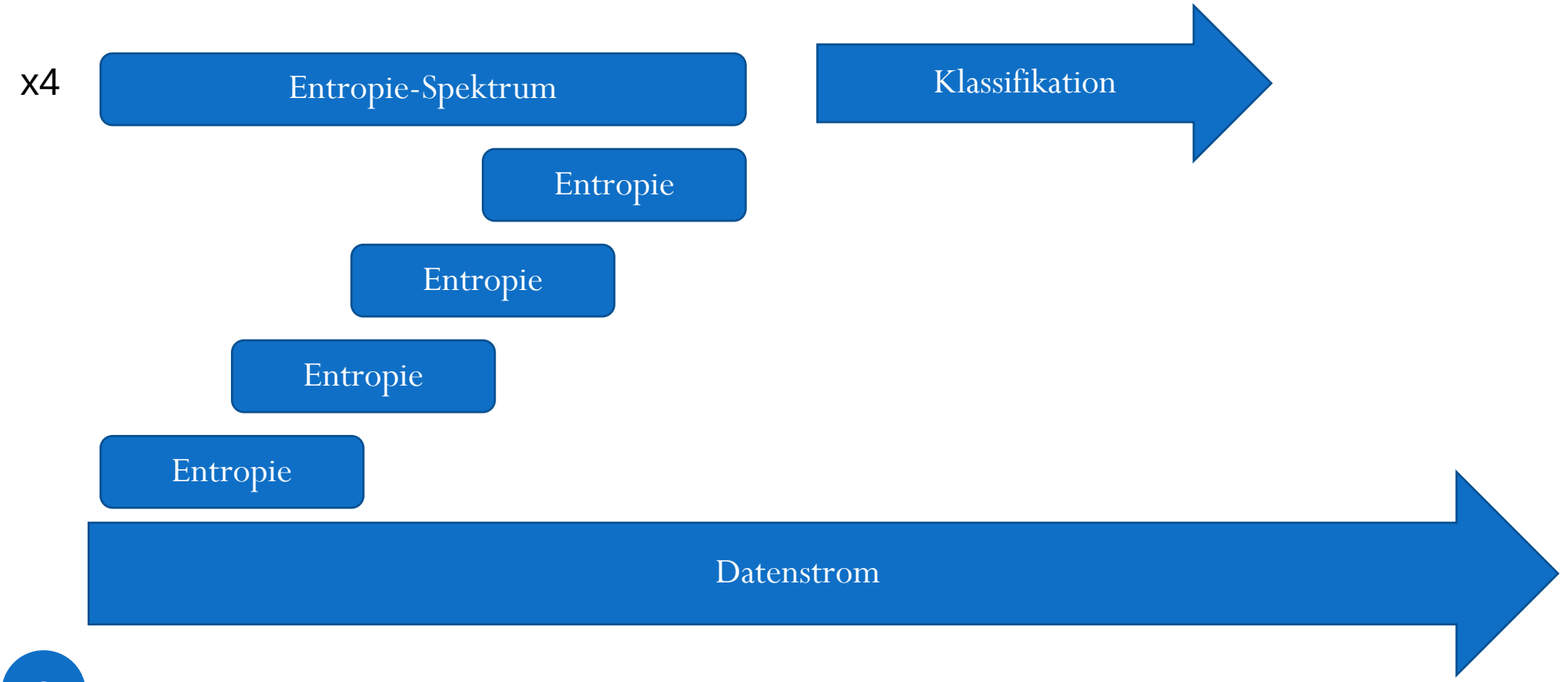
Binärcode oder reguläre Daten?



Ein Leichtgewichtiges Verfahren (3)

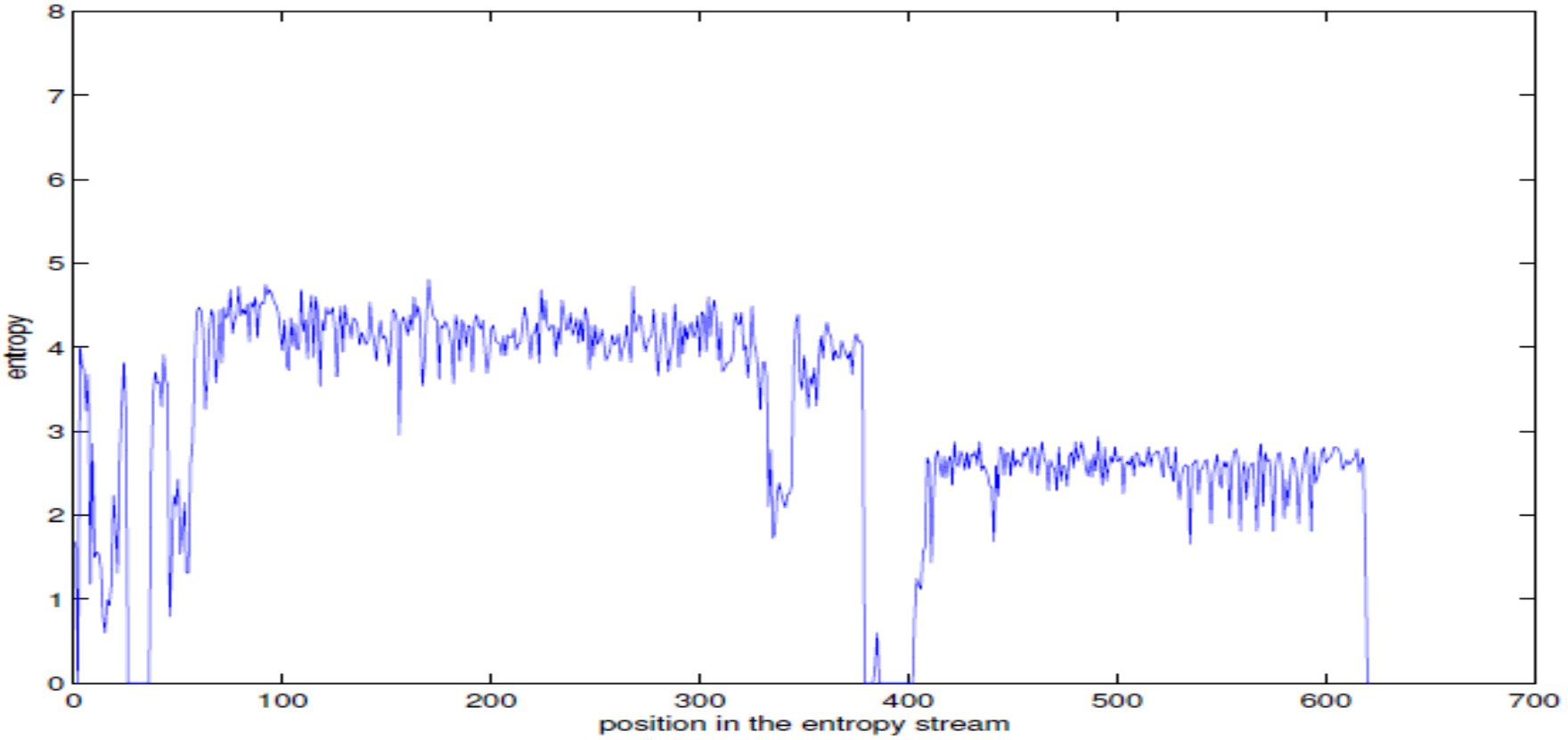
- Merkmale
  - Entropie [3]
  - Frequenzanalyse der Entropie
- Probleme der Fenstertechnik

Ein Leichtgewichtiges Verfahren (4)



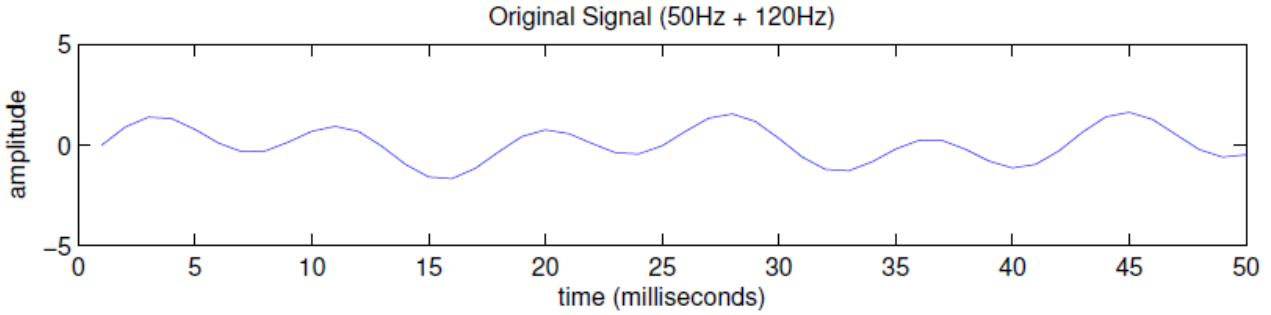
Entropie

- Maß für Informationsdichte
- Interne Struktur von Dateien wird sichtbar

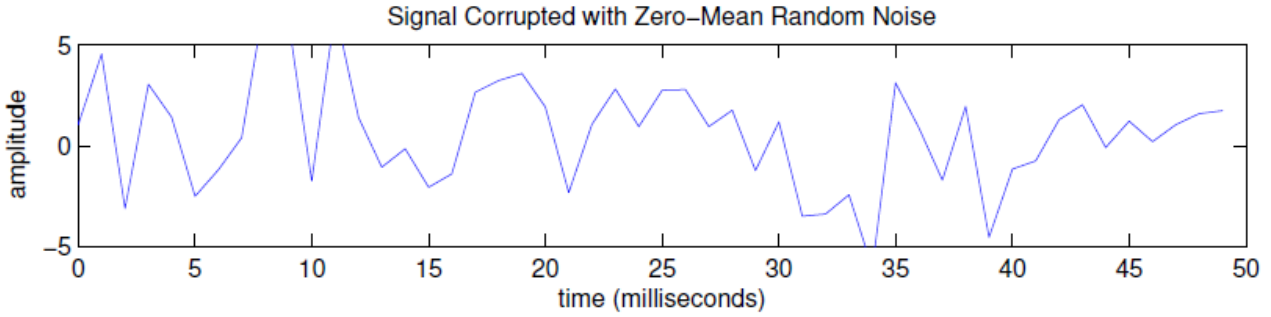


[3]

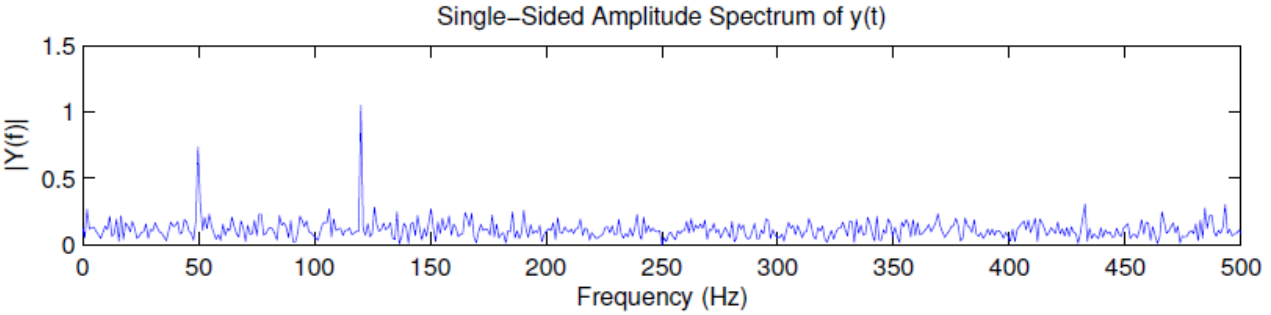
Fourier-Analyse



(a) Two sinusoidal signals of 50 Hz and 120 Hz.

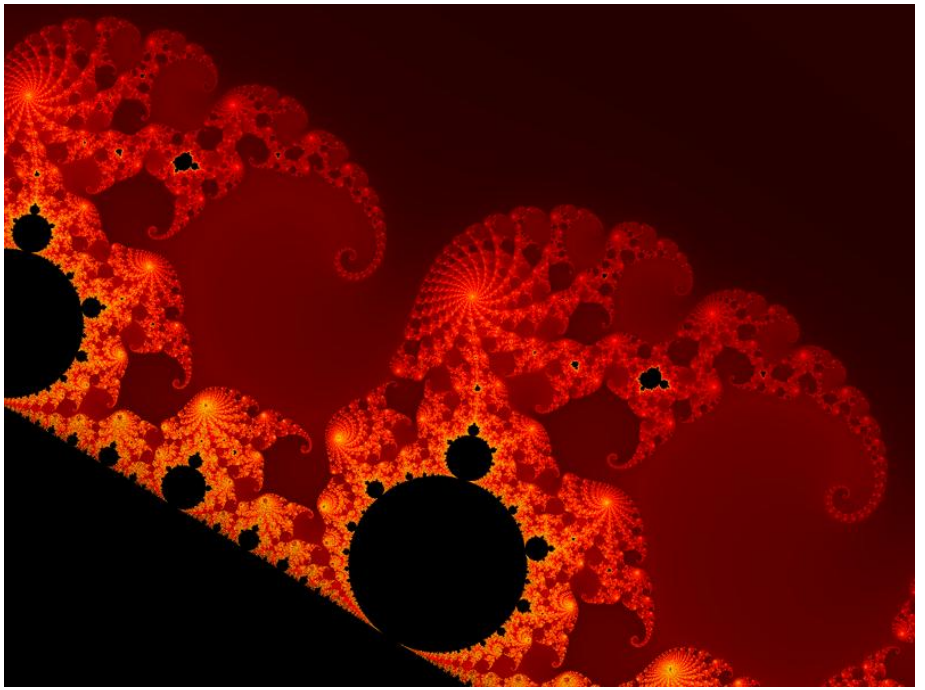


(b) Figure 1(a) with random noise added.



Kolmogorow-Komplexität

- Maß für Code-Komplexität
  - Kleinste Darstellungsform für gegebenen Code
- Vergleich zur Entropie
  - Alphabet: {A,B}
  - X=ABAB…



Quellen: [7,8]

Klassifikator: Neuronale Netze

- Überwachtes Lernen
- Manuelles Klassifizieren zum Lernen erforderlich
- Wenig Speicherbedarf während der Ausführung
- Gelerntes ist nicht „Exportierbar“
- Ergebnisse nicht immer einheitlich durch anfängliche Randomisierung

Quelle: [5]

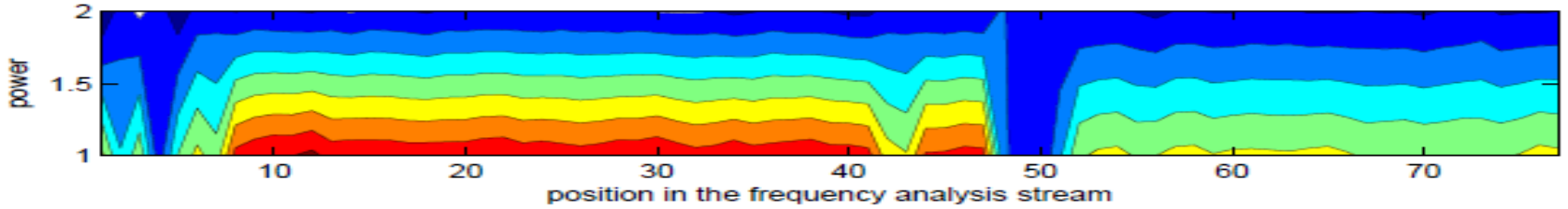
Testdatenbeschaffung

- Beschränkung auf ARM-Platform
- Testdaten aus möglichst vielen Quellen um systematische Fehler auszuschliessen. Z.B. Bias auf
  - Generatoren für Dateitypen
  - Verwendete Bibliotheken
- Quellen
  - ARM-Binaries aus Ubuntu-Installation
  - MS-DOC, HTML, ODT, PDF, PPT, MS-XLS, Text, JavaScript, JPEG durch suche über Google via Wörterbuch
- Kritik: Bias durch Google vorgegeben

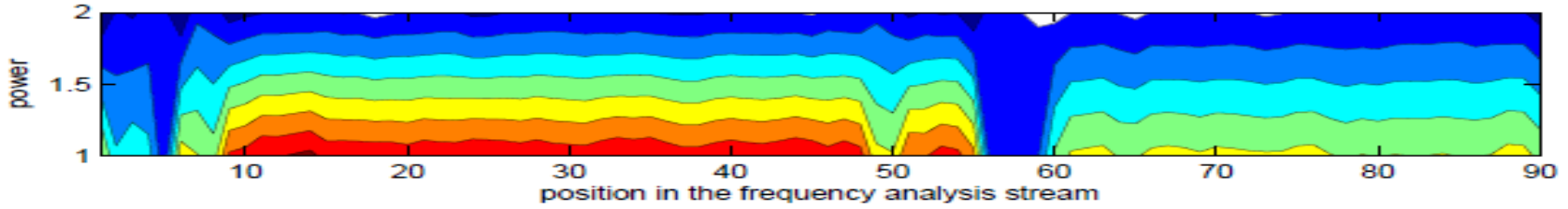
Testdatenaufbereitung

- Dateien > 1/10 der Testdatensatzgröße entfernen
- Zwei Testdatensätze pro Dateityp
  - 1MB Lerndatensatz
  - 10MB Verifikationssatz
- Binaries beinhalten u.a.
  - Headerinformationen
  - Statische Datensektion
  - Code-Section
  - Ressourcen

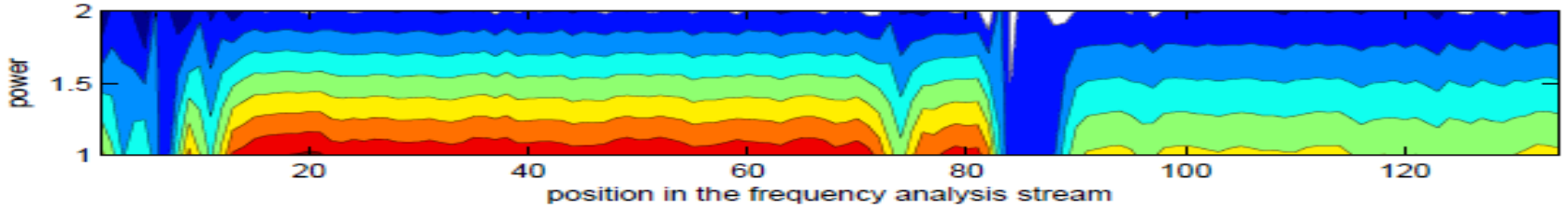
Optimierung der Fenstergrößen



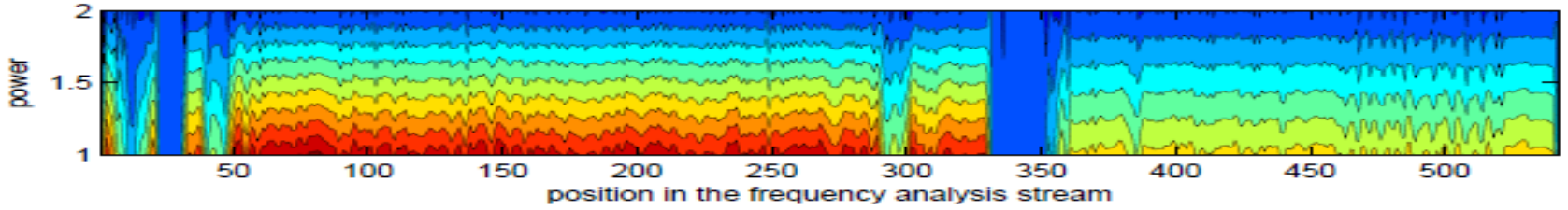
(a) $w_f = 16,\ o_f = 2$



(b) $w_f = 16,\ o_f = 4$

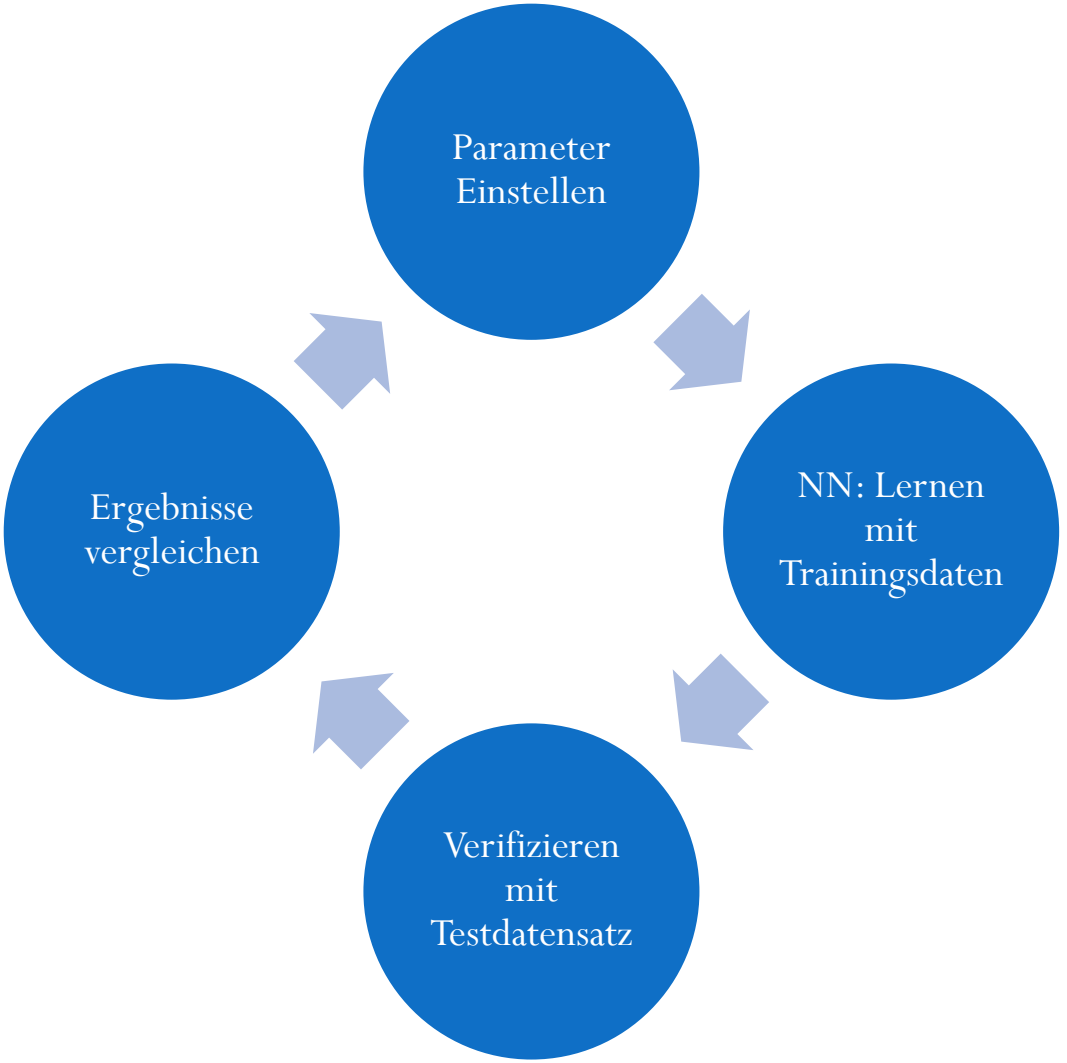


(c) $w_f = 16,\ o_f = 8$



(d) $w_f = 4,\ o_f = 2$

Optimieren des Verfahrens



Ergebnisse

- Erfolgs-Metrik: klassifizierte Fenster

Dateityp	Dateianzahl	Größe in MB	Falsch-Positive(%)	Falsch-Negative (%)
Htm	191	10	0,65	0
Text	7	10	0,31	0
JPEG	37	10	5,21	0
JavaScript	71	10	0,37	0
ELF-arm-32	141	3	0	4,89
ELF-x86-64	422	3	0	5,07
PE-EXE	148	7	0	6,35

Ausblick

- Geschwindigkeit des Verfahrens verbessern
  - Paninski Entropy Estimator [1,2]
  - Parallelisierbarkeit
- Implementierung des Verfahrens und Integration in das Framework von Theodor Nolte
  - Weitere Tests
    - Einzelfallanalyse
      - Shellcode manuell analysieren
  - Implementation auf der Android-Platform
  - Eventuell: Optimierung für Mehrkern-ARM Prozessoren (Pandaboard [9])

Fazit

- Grenzen des Verfahrens
- Risiken
  - Rooten der Smartphones ist notwendig
  - Performance
  - Schlechte Erkennungsrate

Vielen Dank für eure Aufmerksamkeit!
Fragen ?

Referenzen

- **[1]** IEEE TRANSACTIONS ON INFORMATION THEORY, VOL. 50, NO. 9, SEPTEMBER 2004, Liam Paninski, Estimating Entropy on Bins Given Fewer Than Samples, http://www.stat.columbia.edu/~liam/research/abstracts/nm_proof-abs.html
- **[2]** Jean Goubault-Larrecq and Julien Olivain, Detecting Subverted Cryptographic Protocols by Entropy Checking, 2006, http://www.lsv.ens-cachan.fr/Publis/RAPPORTS_LSV/PDF/rr-lsv-2006-13.pdf
- **[3]** Gregory A. Hall, Sliding Window Measurement for File Type Identification, Computer Forensics and Intrusion Analysis Group, ManTech Security and Mission Assurance, 2006.
- **[4]** Jeremy Z. Kolter and Marcus A. Maloof. 2004. Learning to detect malicious executables in the wild. In Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '04).
- **[5]** M.G. Schultz, E. Eskin, E. Zadok, S.J. Stolfo, "Data mining methods for detection of new malicious executables", IEEE Symposium on Security and Privacy, pp. 38-49, USA, IEEe Press, 2001.
- **[7]** A. Lempel and J. Ziv, \On the complexity of nite sequences," Information Theory, IEEE Transactions on, vol. 22, no. 1, pp. 75{81, 1976. [Online]. Available: http://ieeexplore.ieee.org/xpls/absn all.jsp?arnumber=1055501
- **[8]** Image taken from: http://en.wikipedia.org/wiki/Kolmogorov_complexity
- **[9]** Pandaboard ARM development Platform: http://www.pandaboard.org
- **[10]** Julia Wolf, OMG-WTF-PDF presentation slides, 27th Chaos Computer Congress, 2010: http://blog.fireeye.com/research/2011/02/omg-wtf-pdf-denouement.html