Messung der Schwierigkeit von Programmieraufgaben zur Kryptologie in Java

ABP-Workshop 29.10.2021

Konstantin Knorr

Informatik
Hauptcampus

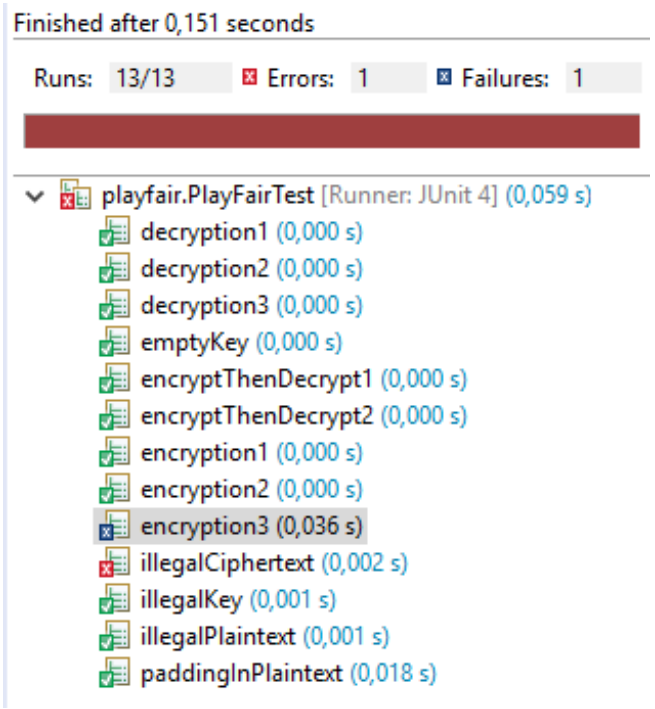
HOCHSCHULE TRIER

Gliederung

- Hintergrund
- Grundlagen
  - JUnit Tests & Kryptologie
  - Das verwendete ABP-System ASB
- Studienaufbau
- Ergebnisse
- Diskussion
- Ausblick

- Anwendung eines bestehendes ABP-Systems auf Aufgaben zur Kryptologie
- Studierende verbessern Verständnis der Kryptologie durch ihre programmatische Umsetzung
- „Schwierigkeit" im Wesentlichen gleich Fehlerrate
- Auswertung der Fehlerrate
  - innerhalb einer Aufgabe: Welche Testfälle sind besonders schwer?
  - Aufgaben-übergreifend: Substitutions- fallen leichter als Transpositionschiffren und
  - pro Testfall-Kategorie: Tests zu Konstruktoren, Exceptions und Padding besser gelöst wurden als Tests zu Signaturen und deren Verifikation
- Identifikation „schwerer" Teile der Kryptologie und bessere didaktische Darstellung dieser in der Zukunft

JUnit-Testfälle zur Kryptologie und deren Ergebnisse in Eclipse



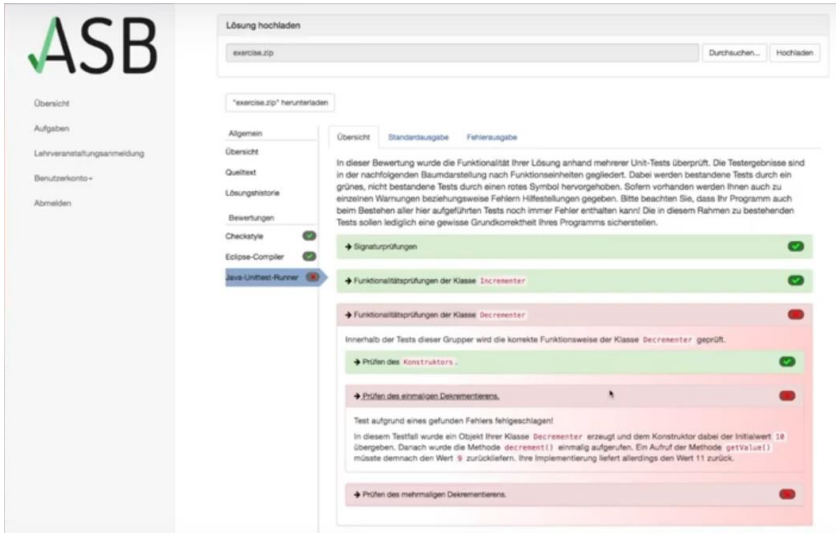

```
public class PlayFairTest extends TestCase{

@Test
@TestDescription("Check correct encryption of
Playfair cipher")
@DependsOnCorrectnessOf("SignatureTests.testSigna
ture")
@TestFailureMsg("The correctly encrypted
ciphertext …)
public void encryption1() {
    PlayFair pf = new PlayFair("CRYPTOLOGY",
    "EDUCATION", true);
    String ciphertext = pf.getCiphertext();
    assertEquals("FECOBPDBQW", ciphertext);
}

@Test(expected = IllegalCharacterException.class)
@TestDescription("Check illegal keys")
@DependsOnCorrectnessOf("SignatureTests.testSigna
ture")
@TestFailureMsg("Only chars A-Z are allowed as
keys.")
public void illegalKey() throws
IllegalCharacterException {
    new PlayFair("CRYPTO*LOGY", "SUPPER", true);
}
```

Das verwendete ASB-System

- ASB = Automatische Software-Bewertung
- Konzipiert von Rainer Oechsle
- Seit 2006 im Einsatz an der Hochschule Trier in mehreren Informatik-Lehrveranstaltungen
- Unterstützte Programmier-sprachen: **Java**, C++, Python
- Zugriff über Browser



- Studierende haben innerhalb des Bearbeitungszeitraums <u>beliebig viele Lösungsversuche</u> pro Aufgabe
- Direktes Feedback über das Ergebnis der einzelnen Testfälle (Bestanden / Nicht bestanden)

- Durchgeführt im Bachelor Wahlpflichtfach „Kryptologisches Programmierpraktikum“ (KPP) im WS 20/21, 5 ECTS, Vorlesung/Screencast: Erläuterung der Theorie, Übung: Vorbesprechung der Aufgaben und Besprechung der Lösung
- Vorkenntnisse der Studierenden:
  - IT-Sicherheit
  - Einführung in die Programmierung (Java)
- 20 Studierende, 15 haben regelmäßig Aufgaben eingereicht
- 10 Wochen lang ca. 2-3 ASB-Aufgaben pro Woche, danach eigenes „freies“ Projekt
- Von den 28 Aufgaben wurden 20 über ASB ausgewertet, der Rest manuell
- ~15 JUnit-Tests pro Aufgabe ➔ ~300 JUnit-Testfälle
- Datensatz + Zusatzinfos stehen anonymisiert zum Download bereit: https://seafile.rlp.net/d/1776901730d14aefa869/

Über ASB bewertete Themen und Aufgaben zur Kryptologie

Themen	Aufgaben
Klassische Chiffren	FourSquare, Playfair, Porta (Substitutionschiffren) Jägerzaun, Skytale (Transpositionschiffren) Bifid (beides)
Asymmetrische Chiffren	BigInteger, RSA, Rabin, Elgamal, Paillier, NTRU
Signaturen	Lamport, RabinSig
Elliptische Kurven	myEC, ECDH, ECDSA
Sonstiges	Shamir Secret Sharing, Java Key Store, Padding Oracle

Kategorien von kryptologischen Testfällen

1. Encryption
2. Decryption
3. EncDec: Encryption and subsequent Decryption
4. Signature: Erstellen der Signatur
5. Verify: Verifikation der Signatur
6. SignAndVerify: Sign and subsequent Signature Verification
7. Exceptions, e.g.
   - invalid input or encoding for keys, plaintext or ciphertext
   - invalid parameters like wrong key length, wrong block size, or wrong length of an initialization vector
   - invalid padding
8. Padding
9. Constructor
10. Misc.

ASB liefert das Ergebnis/Result *r*.

$$r: A \times S \times T \times Z \rightarrow \{true, false\}$$

r(Rabin, 6137, testEncryption2, 2020-10-29T13:21:35) = *true*.

Dabei sind

- A die Menge der Aufgaben
- *S* ist die Menge der Studierenden-IDs
- *T* ist die Menge der Testfälle
- Z steht für die Zeit

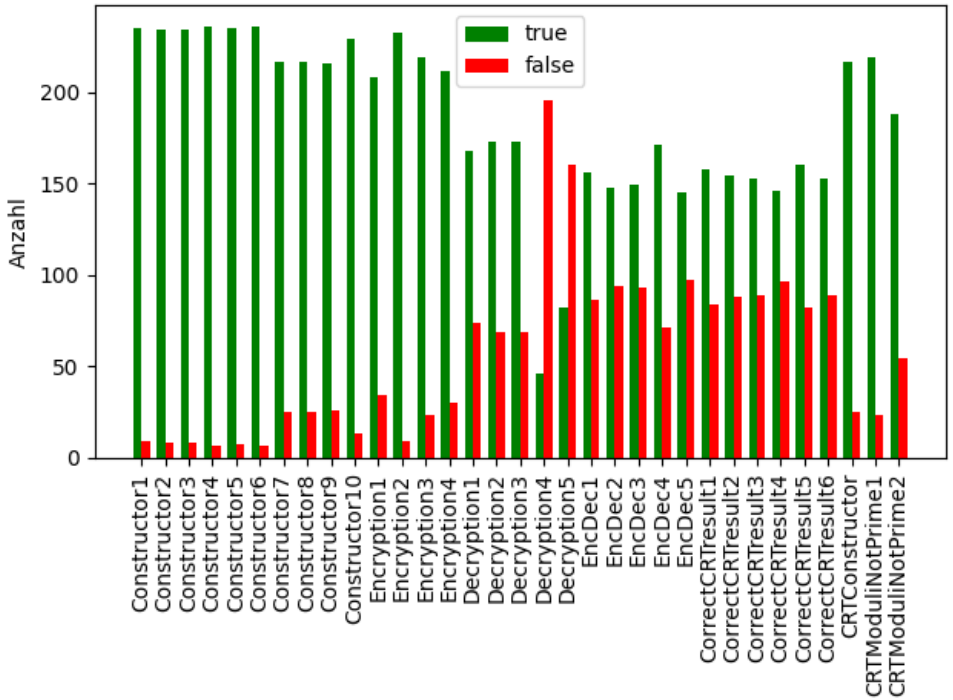
Die **Fehlerrate *e*** gibt den Prozentsatz nicht bestandener Testfälle an:

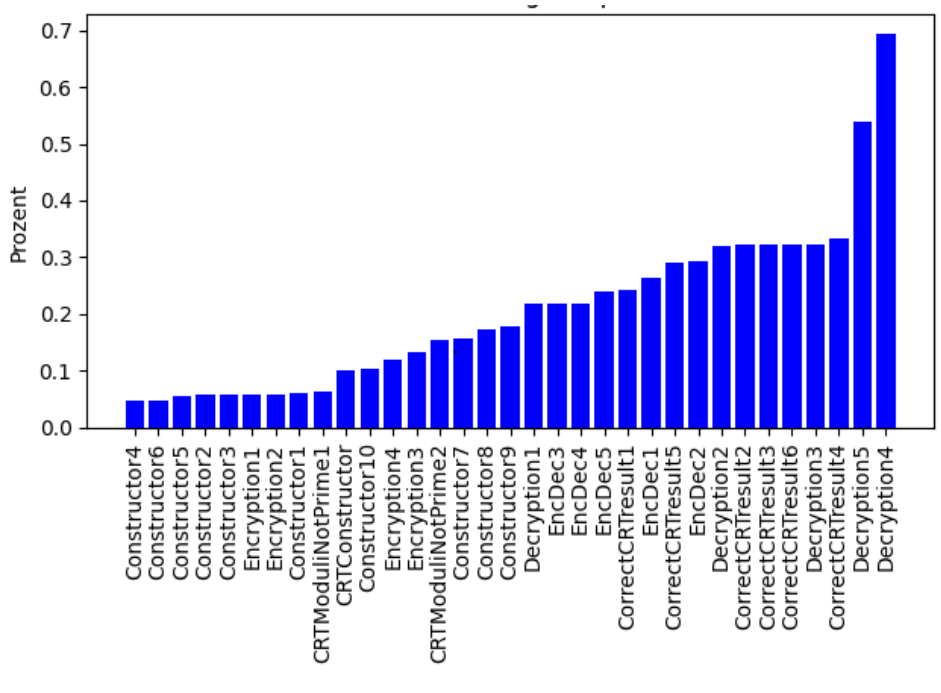
$$e: A \times S \times T \rightarrow [0,1]$$

Die **Bearbeitungszeit (evtl. besser „Bestehenszeitpunkt“) *t*** gibt den normierten ersten Zeitpunkt des Bestehens für einen Testfall an:

$$t: A \times S \times T \rightarrow [0,1]$$

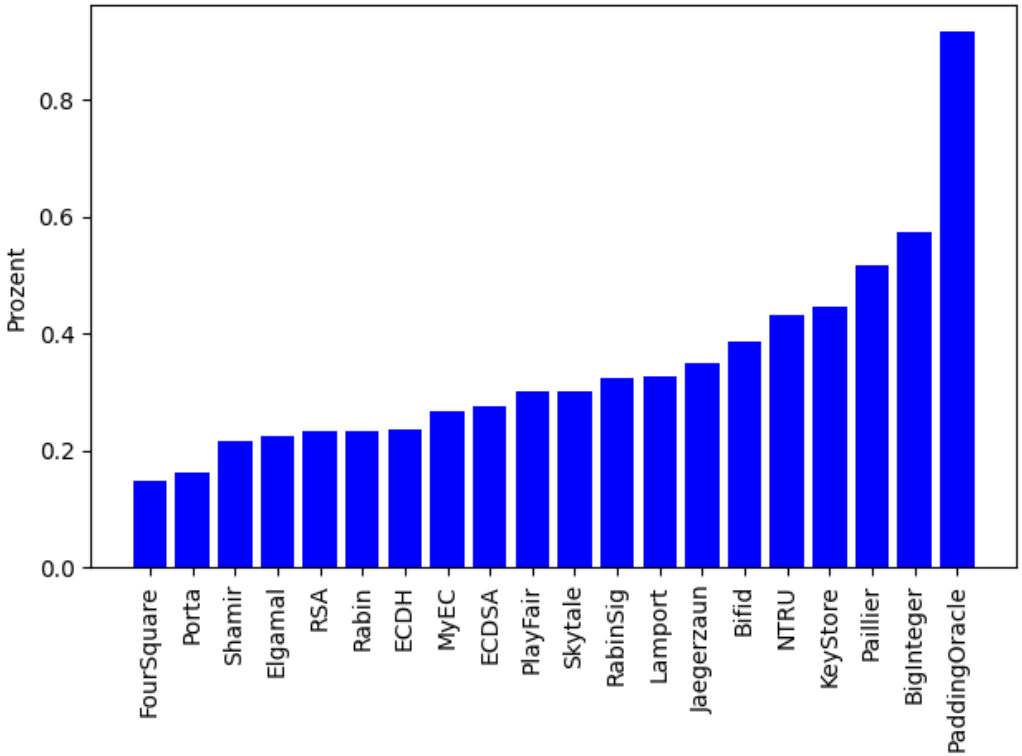
Fehlerraten absolut (links) und Bearbeitungszeit *t* (rechts) in der Aufgabe zur Rabin-Verschlüsselung





Ergebnis: Die Verschlüsselungs- und Constructor-Tests weisen deutlich niedrigere Werte für *e* und *t* auf, als die Tests zur Entschlüsselung, zur Ver- und anschließenden Entschlüsselung und zum chinesischen Restsatzes.

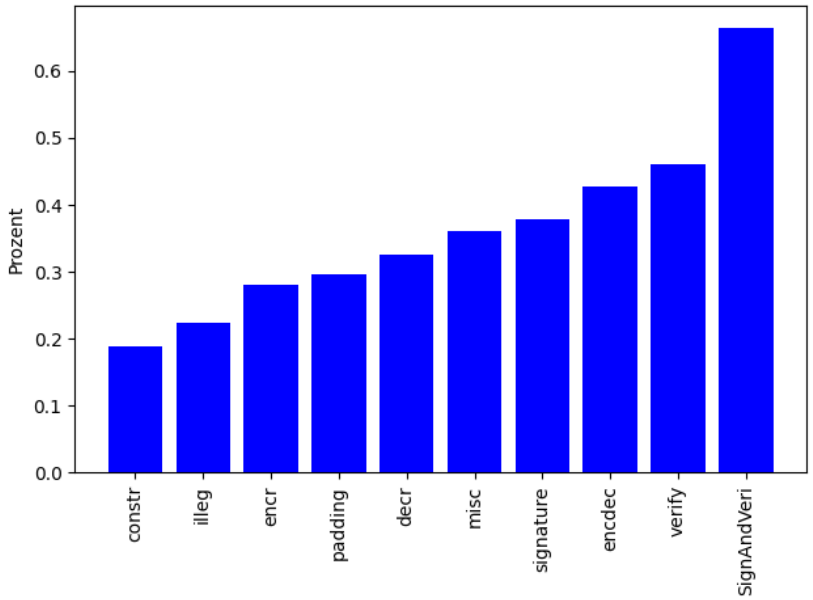
Durchschnittliche Fehlerrate *e* pro Aufgabe



Ergebnisse:

- Substitutionschiffren FourSquare und Porta besser gelöst als Transpositonschiffren Jägerzaun und Skytale
- Asymmetrische Verschlüsselungsverfahren wie RSA und Elgamal besser gelöst als Signaturverfahren (RabinSig, Lamport)
- Moderne asym. Verfahren wie NTRU und Paillier fallen schwerer als klassische asym. Verfahren wie RSA und Rabin
- Aufgaben zu elliptischen Kurven rangieren im Mittelfeld

Fehlerrate *e* über alle Aufgaben gruppiert nach Testkategorie



Ergebnisse:

- Tests zu den Konstruktoren und Exceptions am leichtesten
- Encryption leichter als Decryption
- Signieren leichter als die Signaturverifikation
- Vertraulichkeitsschutz leichter als Integritätsschutz
- „Kombinations-Tests“ EncDec und SignAndVerify schwerer als Einzeltests

Einschränkungen

- Unterschiedliche Voraussetzung bei den Studierenden
- Plagiate / Teamarbeit
- Chronologie der Testfälle
- Difficulty vs. Complexity

Ausblick

- Aufgaben in Python
- Neue Aufgaben z.B. für Timing Angriffe, symmetrische und Postquanten-Verfahren
- Erweiterung von ASB z.B. um Auswertung zur Fehlerrate
- Bessere Kategorisierung von Themen und Testkategorien

Vielen Dank für Ihre Aufmerksamkeit!

knorr@hochschule-trier.de