# Randomisierung

https://iuk.one/DS-RAND.pdf

Clemens H. **Cap**
ORCID: 0000-0003-3958-6136

Department of Computer Science
University of **Rostock**
Rostock, Germany
clemens.cap@uni-rostock.de

2026-11-23 20:52



# Ziel

Zufall im Sinne von "Randomisierung" ist ein wesentliches Werkzeug der Kryptographie.

Wie wollen verstehen **warum** das so ist, und was daraus für die Datensicherheit **folgt**.

# Motivation

**Beispiel:**

Alice an Bob: `1214 9ACE FDDE`
Bob an Alice: `4444 7777 1214`
Alice an Bob: `5738 AE34`
Bob an Alice: `4444 7777 1214`
Alice an Bob: `EEF7 889C FF45 6676`
Bob an Alice: `4444 7777 1214`

**Fragen:**

- Zu welchem Plaintext gehört `4444 7777 1214`?
- Ist das immer derselbe Plaintext?
- Macht eine einfach Replay-Attacke vielleicht Sinn?
- Was kann der Angreifer tun, wenn er zusätzlich noch den Plaintext herausfindet?

# Angriffe auf deterministische Verschlüsselung (1)

## Deterministische Verschlüsselung

Klartext $p$ verschlüsselt immer in denselben Chiffretext $c$.

**Codebuch-Angriff:**

- Baue ein Wörterbuch aus Chiffretext – Klartext auf.
  Aus: Einzelnen abgefangenen (Chiffretext, Klartext) Paaren.
  Aus: Kontextuell abgeleiteten Bedeutungen.
- Bsp: `4444 7777 1214` bedeutet `Ja, gerne.`

**Korrelations-Attacke:** Beobachte Zusammenhang zwischen Chiffretext und Kontext.

- Bsp: Immer wenn Alice `4444 7777 1214` sendet kommt Tresorwagen zur Bank.
  `4444 7777 1214` senden und zeitgleich 10 Anrufe, um Polizei zu beschäftigen.

# Angriffe auf deterministische Verschlüsselung (2)

**Replay Attacke:**

- Angreifer sendet einen abgefangenen Chiffretext nochmals.
- Empfänger hält diesen für authentisch, entschlüsselt und reagiert auf ihn.

**Key Guessing Attacke:**

- Öffentlicher RSA Schlüssel enthält Produkt zweier großer Primzahlen: $n = p \cdot q$.
- Angreifer besorgt sich viele öffentliche Schlüssel.
- Wurden die Primzahlen nicht *sauber* randomisiert, enthalten vielleicht zwei Schlüssel denselben Primfaktor $p$: $n_i = p \cdot q$ und $n_j = p \cdot r$.
- Angreifer berechnet für alle Schlüsselpaare den ggT nach Euklid: $\mathbf{ggT}(n_i, n_j) = p$
- Damit kennt Angreifer beide Primfaktoren
  und kann den privaten Schlüssel $d$ aus dem öffentlichen $e$ bestimmen.

# Problem deterministischer Verschlüsselung

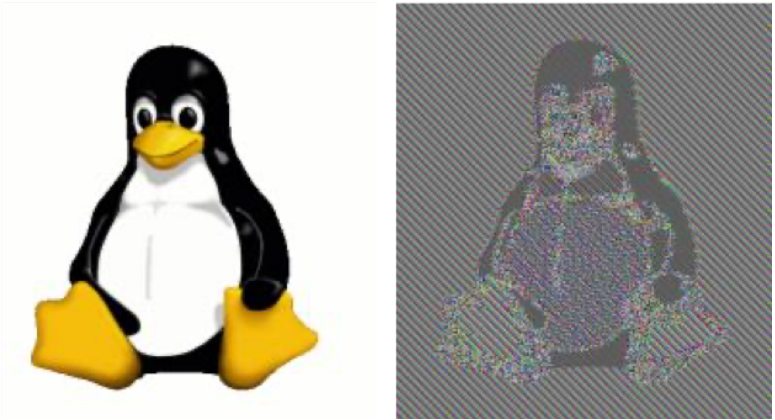
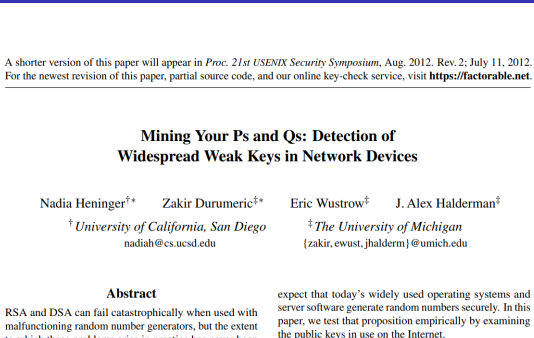


**Abb. 1:** Deterministische Verschlüsselung führt dazu, daß ein Klartext immer in denselben Chiffretext verschlüsselt. Daraus lassen sich je nach Situation weitgehende Angriffe auf Zusammenhänge ableiten. In der Abbildung wird eine Grafik ziemlich untauglich verschlüsselt, indem jeder Pixelwert deterministisch verschlüsselt wird. Das Beispiel ist eine Art Korrelations-Attacke, da ein Zusammenhang zwischen Chiffretext und Helligkeitwert hergestellt werden kann. Rechte s. Anhang.

# Key Guessing Attacke



A shorter version of this paper will appear in *Proc. 21st USENIX Security Symposium*, Aug. 2012. Rev. 2; July 11, 2012.
For the newest revision of this paper, partial source code, and our online key-check service, visit **https://factorable.net**.

**Mining Your Ps and Qs: Detection of Widespread Weak Keys in Network Devices**

Nadia Heninger[†*] Zakir Durumeric[‡*] Eric Wustrow[‡] J. Alex Halderman[‡]

[†] *University of California, San Diego*
nadiah@cs.ucsd.edu

[‡] *The University of Michigan*
{zakir, ewust, jhalderm}@umich.edu

**Abstract**

RSA and DSA can fail catastrophically when used with malfunctioning random number generators, but the extent to which these problems arise in practice has never been comprehensively studied at Internet scale. We perform the largest ever network survey of TLS and SSH servers and present evidence that vulnerable keys are surprisingly widespread. We find that 0.75% of TLS certificates share keys due to insufficient entropy during key generation, and we suspect that another 1.70% come from the same faulty implementations and may be susceptible to compromise. Even more alarmingly, we are able to obtain RSA private keys for 0.50% of TLS hosts and 0.03% of SSH hosts, because their public keys shared nontrivial common factors due to entropy problems, and DSA private keys for 1.03% of SSH hosts, because of insufficient signature randomness. We cluster and investigate the vulnerable keys and find that they are generated by a variety of implementations, and we explain how some of the most widely used cryptographic libraries fail to generate sufficiently random keys. Finally, we analyze behavioral properties of the weak keys and suggest that many of the vulnerabilities we observe are due to specific software failures. Our study calls into question the assumption that secure keys can be generated easily, and we conclude with recommendations for system designers. We expect that today's widely used operating systems and server software generate random numbers securely. In this paper, we test that proposition empirically by examining the public keys in use on the Internet.

The first component of our study is the most comprehensive Internet-wide survey to date of two of the most important cryptographic protocols, TLS and SSH (Section 3.1). By scanning the public IPv4 address space, we collected 5.8 million unique TLS certificates from 12.8 million hosts and 6.2 million unique SSH host keys from 10.2 million hosts. This is 67% more TLS hosts than the latest released EFF SSL Observatory dataset [20]. Our techniques take less than 24 hours to scan the entire address space for listening hosts and less than 96 hours to retrieve keys from them. The results give us a macroscopic perspective of the universe of keys.

Next, we analyze this dataset to find evidence of several

**Abb. 2:** Publikation über die erfolgreiche Key Guessing Attacke. Rechte s. Anhang.

# Lösung: Probabilistische Verschlüsselung

## Probabilistische Verschlüsselung

Klartext $p$ verschlüsselt (fast) immer in einen anderen Chiffretext.

**Praktisches Vorgehen:**

- Wähle eine Zufallszahl.
- Nutze diese als Präfix oder Suffix des Klartextes oder arbeite sie *sonst wie* in den Klartext ein.
- Das *sonst wie* ist eine eigene Wissenschaft!
- Bei der Entschlüsselung trennt man die Zufallsbits wieder ab!

# Wo brauchen wir Randomisierung?

1. **Probabilistische Verschlüsselung**.
2. **Probabilistische Unterschrift**.
3. **Erzeugung neuer Schlüssel** für asymmetrische Verschlüsselung.
   Generieren großer ganzer Zahlen als Kandidaten für große Primzahlen.
   Probabilistische Primzahlen-Tests auf diesen Kandidaten.
   $d$ des privaten Schlüssels.
4. **Erzeugung neuer Schlüssel** für symmetrische Verschlüsselung.
   Insbesondere beim OTP.
5. **Erzeugung neuer Schlüssel** für hybride Verschlüsselung.
   Regelmäßig neue symmetrische Schlüssel ausgehandelt.
6. **Erzeugung neuer Challenges** für Challenge-Response Protokolle.

# Umfrage: Welche Randomisierung ist sicher gegen Angriffe?

1. Aktuelle Uhrzeit auf Sekunden genau im Moment der Randomisierung.
2. Komplexe Folge von Rechenoperationen auf einen fix einprogrammierten Startwert.
3. Uhrzeit statt fixer Startwert.
4. Ein 4-stelliger Startwert, der vom Benutzer eingegeben wird.
5. Ankunftszeit des letzten Ethernet Pakets im Rechner.
6. Zahl der Dateien auf der Festplatte.
7. Aktuelle Uhrzeit, darauf dann SHA-512 als Hashfunktion angewendet.
8. Aktuelle Uhrzeit, darauf eine 4096-bit RSA Operation mit festem Schlüssel, um den Wert länger zu machen.

https://twbk.de/21z1

# Bessere Ansätze für das Randomisieren (1)

**(1) Quantenmechanische** Prozesse.
Gelten als *grundsätzlich* nicht vorhersagbare Zufallsvorgänge aufgrund der aktuell erfolgreichen Theorien der Quantenphysik.
**Beispiele:**

- Zeit zwischen zwei radioaktiven Zerfällen.
- Spannung an einer Zener-Diode.

**(2) Chaotische thermodynamische** Prozesse.
Gelten als *praktisch* nicht auf *lange* Zeit vorhersagbare Zufallsvorgänge aufgrund der Komplexität und des "chaotischen" Verhaltens des zugrundeliegenden Systems.
**Beispiele:**

- Thermisches Rauschen eines Verstärkers.
- Verhalten einer Lava-Lampe.
- `random.org`

# Lavalampe



**Abb. 3:** Lavalampe. Aufheizen und längere Wartezeiten ergeben einen sehr langsamen Strom von Zufallswerten.
Rechte s. Anhang.

# Bessere Ansätze für das Randomisieren (2)

**(3) Verhalten** eines komplexen Systems.
**Achtung:** Nicht wirklich zufällig. Erfordert Nachbearbeitung.
**Beispiele:**

1. Ankunftszeit des nächsten Ethernet-Pakets.
2. Anzahl der Dateien auf einer Festplatte.

**(4) Benutzerverhalten**
**Achtung:** Nicht wirklich zufällig. Erfordert Nachbearbeitung.
**Beispiele:**

1. Zeit zwischen zwei Tastatur-Anschlägen.
2. Tasten, die gedrückt werden bei der Bitte *"drücke beliebige Tasten"*.

# Sichere Randomisierung in der Praxis

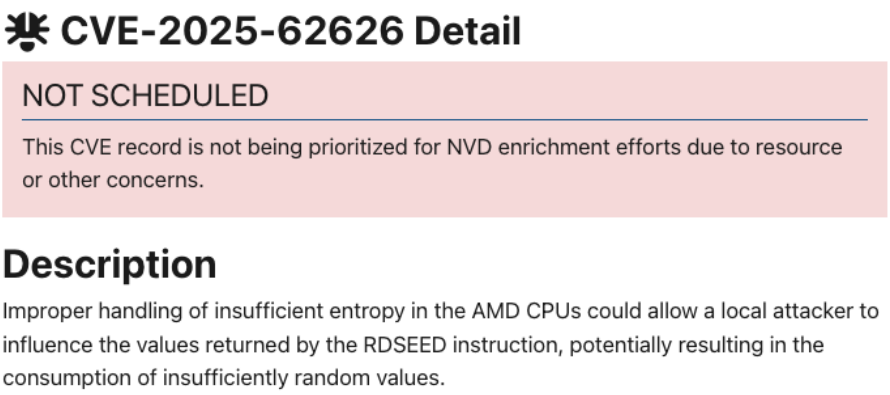
**Typische Quelle:** Quantenphysikalische Effekte an Zener-Diode

- **Variante 1:** Zener-Diode auf der CPU (Bsp: High-end Intel Prozessoren).
- **Variante 2:** Externes Device. (Bsp: USB-Stick)

**Am besten:**

- Aggregieren von Zufall aus **vielen verschiedenen** Quellen.
- Bsp: Hardware-Quelle, Benutzer, Uhrzeit, Datenverkehr, Festplattendaten uvm.
- Entropie rechnerisch aus diesen Quellen "destillieren".
- Auf Unix unter /dev/random und /dev/urandom gesammelt.
- Für Anwender: Vertrauenswürdige Crypto-Library nutzen!

# Fehler in Implementierung der Randomisierung



## CVE-2025-62626 Detail

NOT SCHEDULED

This CVE record is not being prioritized for NVD enrichment efforts due to resource or other concerns.

## Description

Improper handling of insufficient entropy in the AMD CPUs could allow a local attacker to influence the values returned by the RDSEED instruction, potentially resulting in the consumption of insufficiently random values.

**Abb. 4:** Gelegentlich geht die Implementierung in einem Prozessor auch schief, die CVE-2025-62626 zeigt. Rechte s. Anhang.

# Nonces und Challenge Response Protokolle

**Situation:** **P**eggy (**P**rover) will **V**iktor (**V**erifier, Bsp: Autotüre) beweisen, daß sie berechtigt ist, diese zu öffnen.

**Problem:** Das Senden eines Passworts kann abgehört werden.
Idee: Passwort verschlüsseln.
**Problem:** Auch verschlüsselt ist die Antwort immer gleich.
Idee: Randomisieren mit Zufall.
**Problem:** Angreifer wiederholt frühere Antwort ("zufällig" gleich randomisiert)

**Lösung:** Challenge-Response Protokoll.

- Viktor nennt eine Zufalls-Frage, Peggy muß richtig antworten.
- Bsp: Frage ist Zufallszahl $z$, Antwort ist $h(k \cdot z)$ mit Hash-Funktion $h$.
- $k$ ist Schlüssel, der nur Peggy und Viktor bekannt ist.
- **Achtung:** Viktor darf eine Zufallszahl nie zweimal als Challenge benutzen.

Eine **Nonce** ist eine Zufallszahl, die vorher noch nie verwendet wurde.

# Pseudo-Zufallszahlen

**Pseudo-Zufallszahlen** sind Ergebnisse mathematischer Operationen, die beinahe zufällige Werte ergeben.

**Typisches Beispiel:**

- Zufälliger Startwert $s$ (seed)
- Dann Iterieren einer Funktion, etwa: $f(x) = x \cdot b + c \bmod (n)$

**Hinweis:**

- Für Simulation & Spiele (gelegentlich) gut geeignet.
- Für ernsthafte kryptographische Anwendungen **völlig ungeeignet**.
- Warnung leider oftmals nicht beachtet!

# Reality Check

**Frage:** Ist schlampige Randomisierung wirklich so ein Problem?

**Beispiel 1:** Die NSA hat bewirkt, daß die NIST den Dual-EC-DBRG Pseudo-Randomisierungs-Algorithmus normiert, zu dem die NSA ein Backdoor hat.

**Beispiel 2:** Randomisierung in rund 1% aller Router ist nicht ausreichend groß und erlaubt eine erfolgreiche Key Guessing Attacke. Siehe Slide 7.

# Appendix

# Übersicht


Übersicht ..... 20

Verzeichnis aller Abbildungen Abb ..... 21

Rechtsnachweise © ..... 22

Rechtliche Hinweise § ..... 23

Zitierweise dieses Dokuments → ..... 25

Verzeichnis aller Folien ..... 26

# Verzeichnis aller Abbildungen

1 Deterministische Verschlüsselung eines Bildes . . . . . 6
2 Key Guessing Attack . . . . . 7
3 Lavalampe . . . . . 12
4 Fehler in Implementierung der Randomisierung . . . . . 15

# Rechtsnachweise

Abb. 1 Linux Pinguin, im public domain.

Abb. 2 Eigener Screenshot nach Zitaterecht.

Abb. 3 Nutzung nach CC BY-SA 3.0, Quelle: https://commons.wikimedia.org/wiki/Special:Redirect/page/242255

Abb. 4 Eigener Screenshot von https://nvd.nist.gov/vuln/detail/cve-2025-62626

# Rechtliche Hinweise (1)

Die hier angebotenen Inhalte unterliegen deutschem Urheberrecht. Inhalte Dritter werden unter Nennung der Rechtsgrundlage ihrer Nutzung und der geltenden Lizenzbestimmungen hier angeführt. Auf das Literaturverzeichnis wird verwiesen. Das Zitatrecht in dem für wissenschaftliche Werke üblichen Ausmaß wird beansprucht. Wenn Sie eine Urheberrechtsverletzung erkennen, so bitten wir um Hinweis an den auf der Titelseite genannten Autor und werden entsprechende Inhalte sofort entfernen oder fehlende Rechtsnennungen nachholen. Bei Produkt- und Firmennamen können Markenrechte Dritter bestehen. Verweise und Verlinkungen wurden zum Zeitpunkt des Setzens der Verweise überprüft; sie dienen der Information des Lesers. Der Autor macht sich die Inhalte, auch in der Form, wie sie zum Zeitpunkt des Setzens des Verweises vorlagen, nicht zu eigen und kann diese nicht laufend auf Veränderungen überprüfen.

Alle sonstigen, hier nicht angeführten Inhalte unterliegen dem Copyright des Autors, Prof. Dr. Clemens Cap, ©2020. Wenn Sie diese Inhalte nützlich finden, können Sie darauf verlinken oder sie zitieren. Jede weitere Verbreitung, Speicherung, Vervielfältigung oder sonstige Verwertung außerhalb der Grenzen des Urheberrechts bedarf der schriftlichen Zustimmung des Rechteinhabers. Dieses dient der Sicherung der Aktualität der Inhalte und soll dem Autor auch die Einhaltung urheberrechtlicher Einschränkungen wie beispielsweise Par 60a UrhG ermöglichen.

Die Bereitstellung der Inhalte erfolgt hier zur persönlichen Information des Lesers. Eine Haftung für mittelbare oder unmittelbare Schäden wird im maximal rechtlich zulässigen Ausmaß ausgeschlossen, mit Ausnahme von Vorsatz und grober Fahrlässigkeit. Eine Garantie für den Fortbestand dieses Informationsangebots wird nicht gegeben.

Die Anfertigung einer persönlichen Sicherungskopie für die private, nicht gewerbliche und nicht öffentliche Nutzung ist zulässig, sofern sie nicht von einer offensichtlich rechtswidrig hergestellten oder zugänglich gemachten Vorlage stammt.

**Use of Logos and Trademark Symbols:** The logos and trademark symbols used here are the property of their respective owners. The YouTube logo is used according to brand request 2-9753000030769 granted on November 30, 2020. The GitHub logo is property of GitHub Inc. and is used in accordance to the GitHub logo usage conditions `https://github.com/logos` to link to a GitHub account. The Tweedback logo is property of Tweedback GmbH and here is used in accordance to a cooperation contract.

# Rechtliche Hinweise (2)

**Disclaimer:** Die sich immer wieder ändernde Rechtslage für digitale Urheberrechte erzeugt ein nicht unerhebliches Risiko bei der Einbindung von Materialien, deren Status nicht oder nur mit unverhältnismäßig hohem Aufwand abzuklären ist. Ebenso kann den Rechteinhabern nicht auf sinnvolle oder einfache Weise ein Honorar zukommen, obwohl deren Leistungen genutzt werden.

Daher binde ich gelegentlich Inhalte nur als Link und nicht durch Framing ein. Lt EuGH Urteil 13.02.2014, C-466/12 (Pressemitteilung, Blog-Beitrag, Urteilstext). ist das unbedenklich, da die benutzten Links ohne Umgehung technischer Sperren auf im Internet frei verfügbare Inhalte verweisen.

Wenn Sie diese Rechtslage stört, dann setzen Sie sich für eine Modernisierung des völlig veralteten Vergütungs- und Anreizsystems für urheberrechtliche Leistungen ein. Bis dahin klicken Sie bitte auf die angegebenen Links und denken Sie darüber nach, warum wir keine für das digitale Zeitalter sinnvoll angepaßte Vergütungs- und Anreizsysteme digital erbrachter Leistungen haben.

Zu Risiken und Nebenwirkungen fragen Sie Ihren Rechtsanwalt oder Gesetzgeber.

Weitere Hinweise finden Sie im Netz hier und hier oder hier.

# Zitierweise dieses Dokuments

Wenn Sie Inhalte aus diesem Werk nutzen oder darauf verweisen wollen,
zitieren Sie es bitte wie folgt:

Clemens H. Cap: Randomisierung. Electronic document, 23. 11. 2026 https://iuk.one/DS-RAND.pdf

**Bibtex Information:** https://iuk.one/DS-RAND.bib

```
@misc{doc:DS-RAND,
    author       = {Clemens H. Cap},
    title        = {Randomisierung},
    year         = {2026},
    month        = {11},
    howpublished = {Electronic document},
    url          = {https://iuk.one/DS-RAND.pdf }
}
```

**Typographic Information:**
Typeset on 2026-11-23 at 20:52
This is pdfTeX, Version 3.141592653-2.6-1.40.29 (TeX Live 2026) kpathsea version 6.4.2
This is pgf in version 3.1.11a
This is preamble-slides.tex ©C.H.Cap
Jobtype flag=foreground
Active modes (if any): slides, biblio
Keywords: keywords
Subject: subject

# Verzeichnis aller Folien


Titelseite ..... 1
Ziel ..... 2
Motivation ..... 3
Angriffe auf deterministische Verschlüsselung (1) ..... 4
Angriffe auf deterministische Verschlüsselung (2) ..... 5
Problem deterministischer Verschlüsselung ..... 6
Key Guessing Attacke ..... 7
Lösung: Probabilistische Verschlüsselung ..... 8
Wo brauchen wir Randomisierung? ..... 9
Umfrage: Welche Randomisierung ist sicher gegen Angriffe? ..... 10
Bessere Ansätze für das Randomisieren (1) ..... 11
Lavalampe ..... 12
Bessere Ansätze für das Randomisieren (2) ..... 13
Sichere Randomisierung in der Praxis ..... 14
Fehler in Implementierung der Randomisierung ..... 15
Nonces und Challenge Response Protokolle ..... 16
Pseudo-Zufallszahlen ..... 17
Reality Check ..... 18


**Legende:**
Fortsetzungsseite
Seite ohne Überschrift
Bildseite