Zufall und Randomisierung

https://iuk.one/1010-1022

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

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

Version 1

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`

Naive Fragen:

- Zu welchem Plaintext gehört `4444 7777 1214`?
- Ist das immer derselbe Plaintext?
- Was kann der Angreifer tun, wenn er das herausfindet?

Ernsthafte Frage: Welche konkreten Angriffe wären denkbar?

Angriffe auf deterministische Verschlüsselung (1)

Deterministische Verschlüsselung

Klartext p verschlüsselt immer in denselben Chiffretext c.

Codebuch-Angriff:

- Baue ein Wörterbuch aus Chiffretext – Klartext auf.
- Bsp: `4444 7777 1214` bedeutet `Ja, gerne.`

Korrelations-Attacke: Beobachte Zusammenhang zwischen Chiffretext und Kontext.

- Bsp: Immer wenn Alice `4444 7777 1214` sendet kommt ein Tresorwagen zur Bank. Was sind nun für Angriffe möglich?
- Bsp: Wenn Alice 4444 7777 1214 von Bob erhält, dann sendet sie 1315 zu Carol. Angreifer sendet 1315 zu Carol.

Angriffe auf deterministische Verschlüsselung (2)

Replay Attacke: Angreifer sendet Chiffretext nochmals.

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 die privaten Schlüssel bestimmen.

Problem deterministischer Verschlüsselung

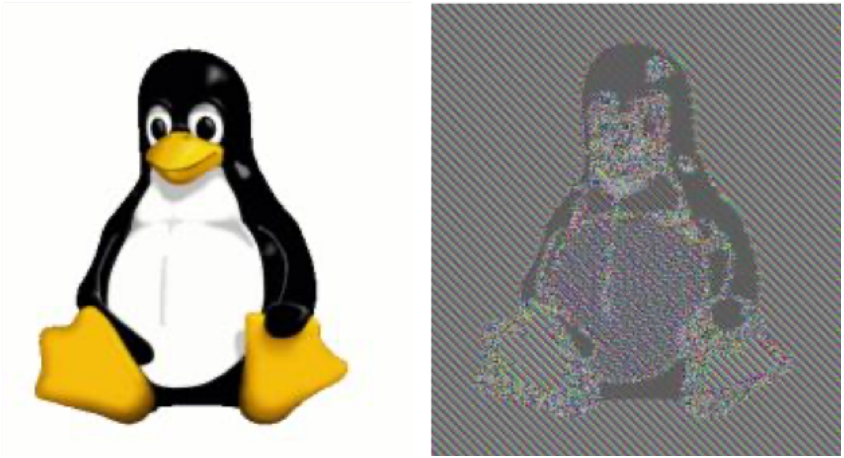

Abb. 1: Deterministische Verschlüsselung führt dazu, daß ein Klartext immer in denselben Chiffretext verschlüsselt. Daraus lassen sich je nach Situation weitgehende Angriffe auf Zusammenhänge ableiten. In der Abbildung wird eine Grafik ziemlich untauglich verschlüsselt, indem jeder Pixelwert deterministisch verschlüsselt wird.

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 vul-

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.

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 noch Randomisierung?

Probabilistische Verschlüsselung.

Probabilistische Unterschrift.

Erzeugung neuer Schlüssel für asymmetrische Verschlüsselung.

Erzeugung neuer Schlüssel für hybride Verschlüsselung.

Erzeugung neuer Challenges für Challenge-Response Protokolle.

Probleme praktischer Randomisierung

Viele gut gemeinte Ansätze der Randomisierung versagen:

1. Aktuelle Uhrzeit im Moment der Randomisierung.
2. Komplexe Folge von Rechenoperationen auf einen fix einprogrammierten Startwert.
3. Startwert ist Uhrzeit
4. Ein 4-stelliger Startwert, der vom Benutzer eingegeben wird.
5. (Nur die) Ankunftszeit des letzten Ethernet Pakets im Rechner.
6. (Nur die) Zahl der Dateien auf der Festplatte.
7. Aktuelle Uhrzeit, darauf eine 4096-bit RSA Operation mit festem Schlüssel, um den Wert länger zu machen.

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) Thermodynamische Prozesse.
Gelten als *praktisch* nicht vorhersagbare Zufallsvorgänge aufgrund der Komplexität und des "chaotischen Verhaltens"des zugrundeliegenden Systems.
Beispiele:

- Thermisches Rauschen eines Verstärkers.
- Verhalten einer Lava-Lampe.

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 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!

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.

Anhang

Übersicht

Verzeichnis aller Abbildungen

Rechtliche Hinweise

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.

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.

Zitierweise dieses Dokuments

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

Clemens H. Cap: Zufall und Randomisierung. Electronic document. https://iuk.one/1010-1022
17. 1. 2021.

Bibtex Information: https://iuk.one/1010-1022.bib

```
@misc{doc:1010-1022,
   author       = {Clemens H. Cap},
   title        = {Zufall und Randomisierung},
   year         = {2021},
   month        = {1},
   howpublished = {Electronic document},
   url          = {https://iuk.one/1010-1022}
}
```

Typographic Information:
Typeset on January 17, 2021
This is pdfTeX, Version 3.14159265-2.6-1.40.21 (TeX Live 2020) kpathsea version 6.3.2
This is pgf in version 3.1.5b
This is preamble-slides.tex myFormat©C.H.Cap

Verzeichnis aller Folien

Legende:
Fortsetzungsseite
Seite ohne Überschrift
Bildseite