



# Kryptographie in CTFs

Eine Einführung (Basierend auf Folien von Leonard Schönborn)

Benedikt Waibel | 19.05.2022

```
sc[] = "\x6a\x0b" // push byte +0xb
// pop eax
// cdq
// push edx
"\x2f\x73\x68" // push dword 0x6
"\x62\x69\x6e" // push dword 0x6
// mov ebx, esp
// xor ecx, ecx
// int 0x80
```

# Klassische Kryptographie

## ■ Caesar-Chiffre

- Jeder Buchstabe wird um einen festen Wert  $k$  verschoben
- Brechen durch Ausprobieren oder durch Häufigkeitsanalyse einfach möglich
- Wird heutzutage manchmal noch verwendet (spiegel.de Paywall)

Klartext: 

|   |   |   |   |   |
|---|---|---|---|---|
| A | B | C | D | E |
|---|---|---|---|---|

 .. 

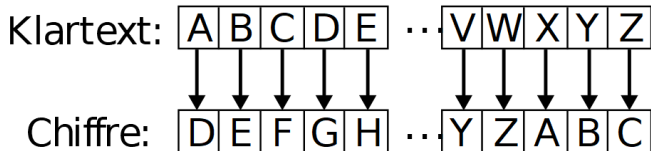
|   |   |   |   |   |
|---|---|---|---|---|
| V | W | X | Y | Z |
|---|---|---|---|---|

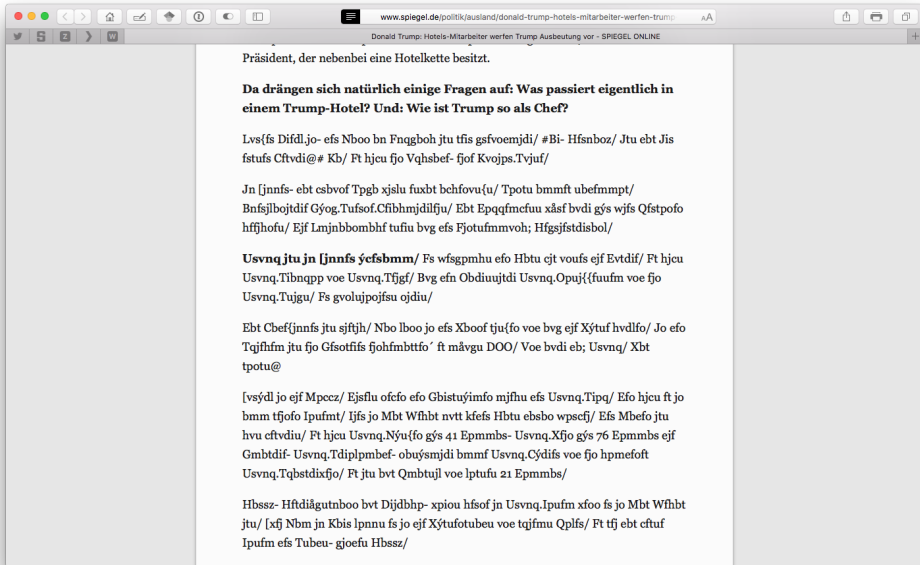
Chiffre: 

|   |   |   |   |   |
|---|---|---|---|---|
| D | E | F | G | H |
|---|---|---|---|---|

 .. 

|   |   |   |   |   |
|---|---|---|---|---|
| Y | Z | A | B | C |
|---|---|---|---|---|

A diagram illustrating the Caesar cipher. It shows the mapping of plaintext letters to ciphertext letters. For the first five letters (A-E), each is shifted three positions forward in the alphabet (A to D, B to E, C to F, D to G, E to H). For the last five letters (V-Z), the shift continues from the end of the alphabet (V to Y, W to Z, X to A, Y to B, Z to C). Arrows point from each plaintext letter to its corresponding ciphertext letter.



# Klassische Kryptographie

## ■ Vigenère-Chiffre

- Wähle Schlüsselwort und verschiebe jeden Buchstaben entsprechend dem Schlüsselbuchstaben
- Schlüssellänge bestimmen und dann Caesar-Chiffre für jede Schlüsselposition einzeln brechen.

|                     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|---------------------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Klartext Nachricht: | W | H | I | T | E | H | A | T | B | L | A | C | K | H | A | T |
|                     | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ |
| Schlüssel:          | S | E | C | U | R | I | T | Y | S | E | C | U | R | I | T | Y |
|                     | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ |
| Chiffretext:        | O | L | K | N | V | P | T | R | T | P | C | W | B | P | T | R |



# Klassische Kryptographie

- **XOR-Chiffre:**

- Verschlüssele Klartext durch XOR mit Key
- ähnlich zu Vigenère

- **One Time Pad (OTP):**

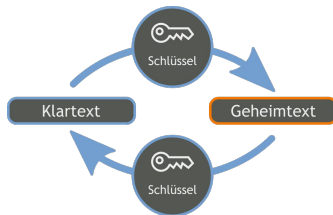
- Verwende Schlüssel mit gleicher Länge wie Klartext
- Perfekte Sicherheit, wenn der Schlüssel gleichverteilt zufällig generiert ist und nur einmal verwendet wird



# Zufall

- Zufällige Eingaben an vielen Stellen für Chiffren benötigt
- unsichere Pseudozufallszahlengeneratoren (PRNGs) Standardquelle für Zufall in vielen Sprachen
- Cryptographisch sichere RNGs:
  - `/dev/urandom`
  - Hardware RNGs
  - RNGs der meisten Crypto-Bibliotheken (z.B. `secrets` in python)

# Symmetrische Verschlüsselungen



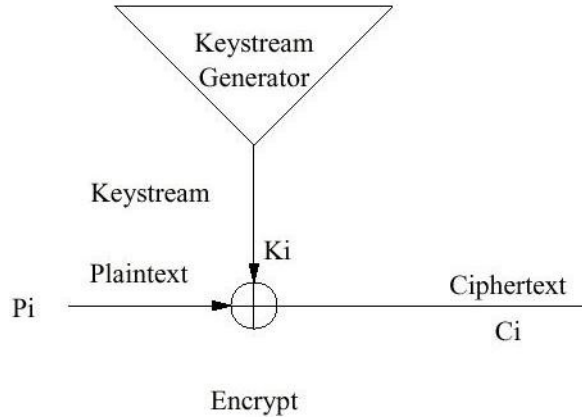
## ■ Blockchiffren

- Verschlüsselt Blöcke fester Länge
- Betriebsmodus wird zur Verschlüsselung längerer Daten verwendet

## ■ Stromchiffren

- Pseudozufälliger Schlüsselstrom wird aus Schlüssel abgeleitet
- Schlüsselstrom wird mit Klartext kombiniert

# Stromchiffren





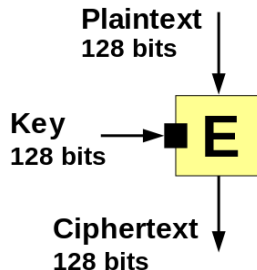


# Stromchiffren

- RC4, SEAL, Salsa, CryptMT
- **Mögliche Angriffe:**
  - Bekannter Klartext: Aus einem bekannten Klartext  $m$  mit zugehörigem Chifftrat  $c$  kann der Schlüsselstrom  $K$  rekonstruiert werden
  - Key-Reuse: Sind  $c_1$  und  $c_2$  mit dem gleichen Schlüssel verschlüsselt worden, dann kann man  $m_1$  XOR  $m_2$  wie folgt berechnen:

# Blockchiffren

- DES, IDEA, RC5, AES, Blowfish, ...
- Block- und Schlüssellänge
- Padding: Erweitern der Nachricht auf Blocklänge
- Betriebsmodi
  - Electronic Code Book (ECB)
  - Cipher Block Chaining (CBC)
  - Counter Mode (CTR)
  - Galois Counter Mode (GCM)
  - ...



# Electronic Code Book

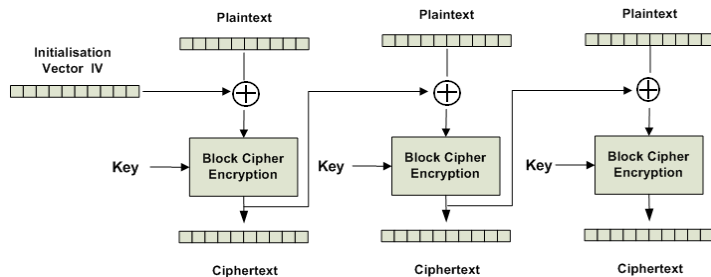


- Verschlüsselt jeden Block einzeln
- **Probleme:**
  - Daten Einfügen möglich
  - Deterministisch

# Cipher Block Chaining

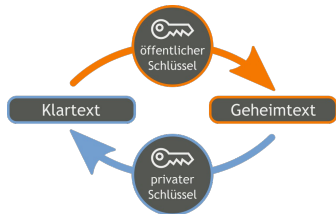
Für Klartextblöcke  $P_i$ , Chiffatblöcke  $C_i, i \in \{1, \dots, n\}$ ,  $C_0 = IV$

- Verschlüsseln:  $C_i = Enc(P_i \oplus C_{i-1})$
- Entschlüsseln:  $P_i = Dec(C_i) \oplus C_{i-1}$
- Initialisierungsvektor zufällig
- **Probleme:**
  - Verlust eines Chiffatblocks führt zu Verlust 2er Klartextblöcke

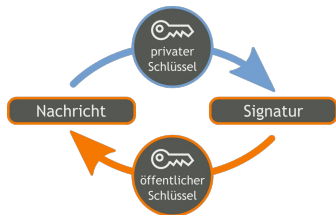


# Asymmetrische Kryptosysteme

## ■ Verschlüsselung:



## ■ Signatur:





# RSA

- Wähle zwei Primzahlen  $p$  und  $q$
- Bestimme  $N = p * q$
- Bestimme  $\Phi(N) = (p - 1) * (q - 1)$
- Wähle  $e$  so, dass  $ggT(e, \Phi(N)) = 1 \wedge 1 < e < \Phi(N)$  gilt
- Bestimme  $d$  so, dass  $e * d \equiv 1(mod \Phi(N))$  gilt. (Erweiterter Euklidischer Algorithmus)
- Öffentlicher Schlüssel:  $N, e$
- Privater Schlüssel:  $d$
- $ggT(a, b)$ : größter gemeinsamer Teiler von  $a$  und  $b$
- $\Phi(N) = |\{a \in \mathbb{N} | 1 \leq a \leq n \wedge ggT(a, N) = 1\}|$   
Anzahl aller Zahlen, die zu  $n$  teilerfremd sind bzw. Gruppenordnung (Eulersche Funktion)



# RSA

- **Encryption:**

$$c = m^e \bmod N$$

- **Decryption:**

$$\begin{aligned} & c^d \bmod N \\ \iff & m^{ed} \bmod N \\ \iff & m^{ed \bmod \Phi(N)} \bmod N \\ \iff & m^1 \bmod N \end{aligned}$$

Mit dem kleinen fermatschen Satz

- **Homomorphie:**

$c_1 = m_1^e \bmod N$  und  $c_2 = m_2^e \bmod N$ , so gilt

$$c_1 \cdot c_2 = m_1^e \cdot m_2^e \bmod N = (m_1 \cdot m_2)^e \bmod N.$$

Es gilt also  $Enc(m_1, pk) \cdot Enc(m_2, pk) = Enc(m_1 \cdot m_2, pk)$



# Angriffe auf RSA

| Bedingung  | Angriff   | Komplexität   |
|--|---|---|
| Keine<br>Kleines $d$ ( $d < \frac{1}{3}N^{\frac{1}{4}}$ )<br>$m < N^{\frac{1}{e}}$<br>Senden der gleichen Nachricht<br>an viele Empfänger mit selbem $e$ | Faktorisierung<br>Wiener's Attack<br>Wurzel ziehen<br>Hastad's Broadcast Attack | $\sim \exp(\log(N)^{\frac{1}{3}}(\log\log N)^{\frac{2}{3}})$<br>Polynomiell<br>Polynomiell<br>Polynomiell |

Und viele mehr!





# Aufgaben

- <https://github.com/kitctf/www/tree/files/crypto.zip>
- <https://cryptopals.com>
- <https://overthewire.org/wargames/krypton>
- <https://picoctf.com>