

IT-Security Tutorübung 06

Dorian Zedler

27. November 2023

Technische Universität München

Aufgabe 1

Aufgabe 2

Aufgabe 3

- Nachrichten Authentifizierungs-Codes (MACs)
- AEAD Chiffren
- Digitale Signaturen
- Pseudo-Zufallszahlengeneratoren (PRNGs)
- Diffie-Hellman Schlüsselaustausch

Aufgabe 1

Aufgabe 1a - MAC-Then-Encrypt

- a) In der Vorlesung haben Sie gelernt, dass bei der Generierung von MACs immer das Prinzip „Encrypt-Then-MAC“ eingehalten werden sollte. Warum ist dies der Variante „MAC-Then-Encrypt“ vorzuziehen? (Hinweis: Erinnern Sie sich an die Padding Oracle Aufgabe!)
- Bei **MAC-Then-Encrypt** wird der MAC über den Klartext gebildet.
Schritte um den MAC zu prüfen:
 - 1) Nachricht entschlüsseln
 - 2) MAC prüfen→ Padding Oracle Angriff möglich!
 - Bei **Encrypt-Then-MAC** wird der MAC über den Ciphertext gebildet.
Schritte um den MAC zu prüfen:
 - 1) MAC prüfen
 - 2) Nachricht entschlüsseln→ Padding Oracle Angriff nicht möglich!

- b) Was sind Vorteile einer AEAD Chiffre im Vergleich zu traditioneller, separater Verschlüsselung und Integritätsschutz mittels MACs?
- Geringere Fehleranfälligkeit bei der Implementierung
 - Alle CIA Schutzziele werden abgedeckt
 - Nur ein Schlüssel wird benötigt
 - Potenziell effizienter als separate Verschlüsselung und MACs
 - Beispiel: **AES-GCM**

c) Grenzen Sie digitale Signaturen und MACs voneinander ab!

- **Digitale Signaturen:**

- Asymmetrisch, Verifikation mit öffentlichem Schlüssel des Senders
- Öffentlicher Schlüssel ist einer Person zugeordnet → signierte Nachricht kann eindeutig dem Absender zugeordnet werden!
- Schutzziele: Authentizität, Integrität, **Verbindlichkeit**
- Beispiel: Unterschreiben eines Vertrages

- **MACs:**

- Symmetrisch, Verifikation mit gemeinsamem, geheimem Schlüssel
- Geheimer Schlüssel ist jedem Teilnehmer bekannt → Nachricht kann nicht eindeutig zugeordnet werden!
- Schutzziele: Integrität, Authentizität

Aufgabe 2

Aufgabe 2a - PRNGs

- a) Für welche kryptografischen Anwendungen benötigen Sie zufällige Werte, die ein Angreifer nicht erraten darf?
- Schlüsselgenerierung
 - Initialisierungsvektoren
 - Seeds für sicheres Padding

Aufgabe 2 - PRNGs, was soll das?

- Zufallszahlen sind wichtig!
- Beispiele für Zufall:
 - Würfeln
 - Rauschender Pin
 - Lavalampe



- Quantenphänomene, z.B. halbdurchlässiger Spiegel



- Zufall ist schwer und teuer zu erzeugen!
- Lösung: **Pseudo-Zufallszahlengeneratoren (PRNGs)**

Aufgabe 2 - PRNGs, wie geht das?

- PRNGs generieren, basierend auf einem **zufälligen Seed**, **deterministisch** Pseudo-Zufallszahlen.
- Beispiel: **Xorshift32** (32-Bit)

```
1  state = ... # Seed the rng
2
3  def u32(x):
4      return x & ((1 << 32) - 1)
5
6  def randgen_xorshift32():
7      global state
8      x = state
9      x ^= u32(x << 13);
10     x ^= u32(x >> 17);
11     x ^= u32(x << 5);
12     state = x
13     return x
```

- Wird in einer erweiterten Form im v8 JavaScript Engine für `Math.random()` verwendet

b) Berechnen Sie die nächsten vier ausgegebenen Zufallszahlen für den Seed: 1.

- 270369
- 67634689
- 2647435461
- 307599695

Aufgabe 2c - Sicher oder nicht sicher, das ist hier die Frage!

- c) Bewerten Sie nun: Ist dies ein kryptografisch sicherer PRNG (CSPRNG) und wäre somit geeignet für die gerade genannten Anwendungen? Prüfen Sie hierfür die Anforderungen aus der Vorlesung!
- Voraussetzungen:
 - Zahlen dürfen keine Hinweise auf Nachfolger geben:
 - Zahlen dürfen keine Hinweise auf Vorgänger geben:
 - Zahlen müssen statistisch gleichverteilt sein:

Aufgabe 2c - Sicher oder nicht sicher, das ist hier die Frage!

- c) Bewerten Sie nun: Ist dies ein kryptografisch sicherer PRNG (CSPRNG) und wäre somit geeignet für die gerade genannten Anwendungen? Prüfen Sie hierfür die Anforderungen aus der Vorlesung!
- Voraussetzungen:
 - Zahlen dürfen keine Hinweise auf Nachfolger geben:
 - Zahlen dürfen keine Hinweise auf Vorgänger geben:
 - Zahlen müssen statistisch gleichverteilt sein:

- c) Bewerten Sie nun: Ist dies ein kryptografisch sicherer PRNG (CSPRNG) und wäre somit geeignet für die gerade genannten Anwendungen? Prüfen Sie hierfür die Anforderungen aus der Vorlesung!
- Voraussetzungen:
 - Zahlen dürfen keine Hinweise auf Nachfolger geben: ✘
→ Da der gesamte innere Zustand des PRNGs bekannt ist, kann mit einer bekannten Zahl jede folgende berechnet werden.
 - Zahlen dürfen keine Hinweise auf Vorgänger geben:
 - Zahlen müssen statistisch gleichverteilt sein:

Aufgabe 2c - Sicher oder nicht sicher, das ist hier die Frage!

- c) Bewerten Sie nun: Ist dies ein kryptografisch sicherer PRNG (CSPRNG) und wäre somit geeignet für die gerade genannten Anwendungen? Prüfen Sie hierfür die Anforderungen aus der Vorlesung!
- Voraussetzungen:
 - Zahlen dürfen keine Hinweise auf Nachfolger geben: ✘
→ Da der gesamte innere Zustand des PRNGs bekannt ist, kann mit einer bekannten Zahl jede folgende berechnet werden.
 - Zahlen dürfen keine Hinweise auf Vorgänger geben: ✘
→ Eine niedrige Zahl deutet darauf hin, dass die vorherige Zahl auch niedrig war.
 - Zahlen müssen statistisch gleichverteilt sein:

Aufgabe 2c - Sicher oder nicht sicher, das ist hier die Frage!

- c) Bewerten Sie nun: Ist dies ein kryptografisch sicherer PRNG (CSPRNG) und wäre somit geeignet für die gerade genannten Anwendungen? Prüfen Sie hierfür die Anforderungen aus der Vorlesung!
- Voraussetzungen:
 - Zahlen dürfen keine Hinweise auf Nachfolger geben: ✘
→ Da der gesamte innere Zustand des PRNGs bekannt ist, kann mit einer bekannten Zahl jede folgende berechnet werden.
 - Zahlen dürfen keine Hinweise auf Vorgänger geben: ✘
→ Eine niedrige Zahl deutet darauf hin, dass die vorherige Zahl auch niedrig war.
 - Zahlen müssen statistisch gleichverteilt sein: ✔

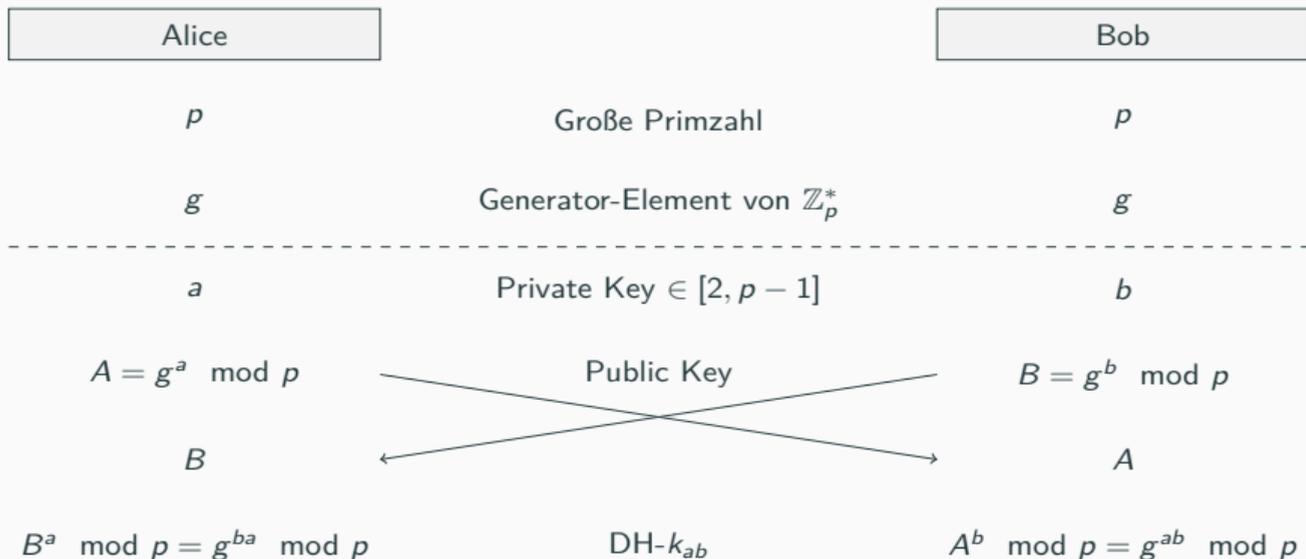
d) Würde es die kryptographische Sicherheit verbessern, wenn nicht x sondern $x \bmod 16777217$ ausgegeben wird?

- **Nein!**
- Wenn der interne Zustand **kleiner** als 16777217 ist
→Die Ausgabe ist gleich dem internen Zustand!
- Wenn der interne Zustand **größer** als 16777217 ist
 - Der interne Zustände ist gleich der Ausgabe $+x \cdot 16777217$
 - Wenn noch weitere Zahlen bekannt sind, kann man x berechnen!
 - Sobald man x kennt, kann man alle folgenden Zahlen berechnen!

Aufgabe 3

Aufgabe 3 - Diffie-Hellman

- Problem: Austausch eines gemeinsamen Schlüssels über einen unsicheren Kanal
- Lösung: Diffie-Hellman Schlüsselaustausch



Aufgabe 3a - Diffie-Hellman

- Gegeben:

- Öffentliche Primzahl $p = 89$
- Generator-Element der zyklischen Gruppe \mathbb{Z}_{89}^* : $g = 28$

a) Führen Sie das DH-Verfahren mit ihrem Tischnachbarn durch!

	Alice
Private Key	
Public Key	
DH-Secret	
	Bob
Private Key	
Public Key	
DH-Secret	

Aufgabe 3a - Diffie-Hellman

- Gegeben:

- Öffentliche Primzahl $p = 89$
- Generator-Element der zyklischen Gruppe \mathbb{Z}_{89}^* : $g = 28$

a) Führen Sie das DH-Verfahren mit ihrem Tischnachbarn durch!

	Alice
Private Key	
Public Key	
DH-Secret	
	Bob
Private Key	
Public Key	
DH-Secret	

Aufgabe 3a - Diffie-Hellman

- Gegeben:

- Öffentliche Primzahl $p = 89$
- Generator-Element der zyklischen Gruppe \mathbb{Z}_{89}^* : $g = 28$

a) Führen Sie das DH-Verfahren mit ihrem Tischnachbarn durch!

	Alice
Private Key	$a = 15$
Public Key	
DH-Secret	
	Bob
Private Key	$b = 47$
Public Key	
DH-Secret	

Aufgabe 3a - Diffie-Hellman

- Gegeben:

- Öffentliche Primzahl $p = 89$
- Generator-Element der zyklischen Gruppe \mathbb{Z}_{89}^* : $g = 28$

a) Führen Sie das DH-Verfahren mit ihrem Tischnachbarn durch!

	Alice
Private Key	$a = 15$
Public Key	$A = g^a \bmod p = 28^{15} \bmod 89 = 13$
DH-Secret	
	Bob
Private Key	$b = 47$
Public Key	$B = g^b \bmod p = 28^{47} \bmod 89 = 31$
DH-Secret	

Aufgabe 3a - Diffie-Hellman

- Gegeben:
 - Öffentliche Primzahl $p = 89$
 - Generator-Element der zyklischen Gruppe \mathbb{Z}_{89}^* : $g = 28$

a) Führen Sie das DH-Verfahren mit ihrem Tischnachbarn durch!

	Alice
Private Key	$a = 15$
Public Key	$A = g^a \bmod p = 28^{15} \bmod 89 = 13$
DH-Secret	$(g^b)^a \bmod p = B^a \bmod p = 31^{15} \bmod 89 = 28$
	Bob
Private Key	$b = 47$
Public Key	$B = g^b \bmod p = 28^{47} \bmod 89 = 31$
DH-Secret	$(g^a)^b \bmod p = A^b \bmod p = 13^{47} \bmod 89 = 28$

- b) Nach der Durchführung haben Sie nun ein gemeinsames DH-Secret in Form eines Integers. Wie können Sie daraus einen AES Schlüssel generieren?
- Umwandeln in Byte-Array
 - Verwenden als Input für eine Key-Derivation-Function (KDF), z.B. pbkdf2

Aufgabe 3c - Perfect Forward Secrecy

c) Erklären Sie das Konzept von Perfect Forward Secrecy (PFS)!

Aufgabe 3c - Perfect Forward Secrecy

- c) Erklären Sie das Konzept von Perfect Forward Secrecy (PFS)!
- Beispiel: Schlüsselaustausch mit RSA
 - Alice sendet Bob ihren öffentlichen Schlüssel
 - Bob verschlüsselt damit den gemeinsamen Schlüssel und sendet ihn an Alice
 - Eve zeichnet alle Nachrichten auf
 - Eve bricht in Alice Computer ein und stiehlt ihren privaten Schlüssel
 - Eve kann nun den gemeinsamen Schlüssel und damit auch alle folgenden Nachrichten entschlüsseln
 - PFS ist nur dann erfüllt, wenn Eve nach dem Einbruch in Alice Computer nicht in der Lage ist, alle aufgezeichneten Nachrichten zu entschlüsseln

- c) Erklären Sie das Konzept von Perfect Forward Secrecy (PFS)!
- Lösung: Schlüsselaustausch mit DH
 - Alice und Bob generieren für jeden Schlüsselaustausch ein neues DH-Secret
 - Nach dem Schlüsselaustausch werden die privaten DH-Parameter verworfen
 - Eve zeichnet alle Nachrichten auf
 - Eve bricht in Alice Computer ein und stiehlt ihr aktuelles DH-Secret
 - Eve kann mit dem gestohlenen DH-Secret nur die aktuelle Nachricht entschlüsseln
 - Alle älteren und zukünftigen Nachrichten bleiben geheim!
→ PFS ist erfüllt!

- d) Unter welchen Voraussetzungen bietet der DH-Schlüsselaustausch PFS?
- Alice und Bob müssen für jeden Schlüsselaustausch ein neues DH-Secret generieren
 - Die privaten DH-Parameter müssen nach dem Schlüsselaustausch verworfen werden