1 Kryptografie

Die Kryptografie wird in drei Hauptkategorien unterteilt:

1. Symmetrische Kryptografie
2. Hashfunktionen
3. Asymmetrische Kryptografie

1.1 Symmetrische Kryptografie

Symmetrische Kryptografie ist eine Menge von kryptografischen Protokollen, bei der derselbe geheime Schlüssel für die Ver- und Entschlüsselung von Daten verwendet wird.

Symmetrische Kryptosysteme

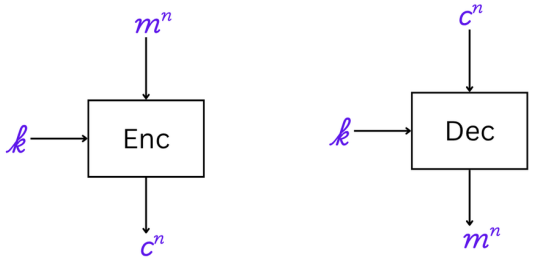
Ein symmetrisches Kryptosystem ist ein 5-Tupel (M, K, C, e, d) bestehend aus:
- einer Menge M von Klartexten,
- einer Menge K von Schlüsseln,
- einer Menge C von Chiffretexten,
- einer Verschlüsselungsfunktion $e : M \times K \to C$,
- einer Entschlüsselungsfunktion $d : C \times K \to M$,

so dass für alle Klartexte $m \in M$ und alle Schlüssel $k \in K$ gilt, dass $d(e(m, k), k) = m$.

1.1.1 Blockchiffren

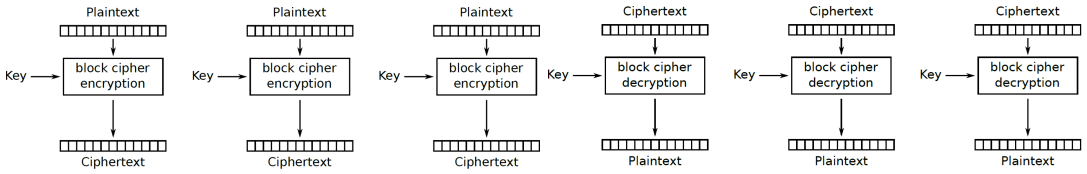
Definition

Blockchiffren sind Kryptosysteme, die nur Blöcke fester Länge verschlüsseln können.



- Ein Blockchiffre arbeitet auf einem Klartextblock der Länge b, um einen Chiffretextblock der Länge b zu erzeugen.
- Der gleiche Schlüssel kann mehrmals auf unterschiedliche Blöcke verwendet werden.
- Beispiele von Blockchiffren: AES, DES, 3DES, Serpent, Twofish, Blowfish, etc.

Electronic Code Book (ECB) Modus



Electronic Codebook (ECB) mode encryption

Electronic Codebook (ECB) mode decryption

Wenn die Blöcke nicht die Länge n haben, können trotzdem beliebige Nachrichten verschlüsselt werden, da eine **Auffüllfunktion** (Padding function) benutzt wird.

Bei vielen Padding-Verfahren wird *immer* ein Padding hinzugefügt, auch wenn die Nachricht bereits ein Vielfaches der Blocklänge n hat. Dies ist notwendig, damit die $unpad()$-Funktion eindeutig feststellen kann, wie viele Bytes entfernt werden müssen. Eine gute Auffüllfunktion sollte umkehrbar sein, d.h. es muss eine $unpad()$-Funktion geben mit $unpad(pad(x)) = x \quad \forall x \in M*$.

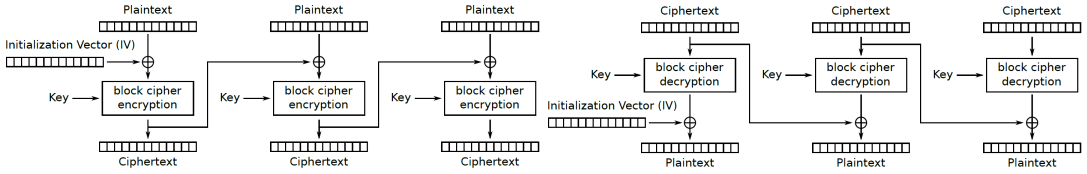
Vorteile:

- Unkomplizierte Bedienung. Jeder Block wird unabhängig bearbeitet.
- Parallelisierbarkeit von Ver- und Entschlüsselungsverfahren.
- Beschädigte Datenblöcke beeinflussen keine anderen Blöcke (Fehlertoleranz).

Nachteile:

- *Deterministisch:* Muster im Klartext sind sichtbar. Identische Klartextblöcke ergeben immer identische Chiffretextblöcke.
- *Keine Diffusion:* Kleine Änderungen im Klartext führen zu lokalisierten Änderungen im Geheimtext.

Cipher Block Chaining (CBC) Modus



Cipher Block Chaining (CBC) mode encryption

Cipher Block Chaining (CBC) mode decryption

Zur Formalisierung von CBC benötigen wir randomisierte Kryptosysteme. Der Zufallswert r wird hier als Initialisierungsvektor (IV) bezeichnet.

Randomisierte symmetrische Kryptosysteme

Ein randomisiertes symmetrisches Kryptosystem ist ein 5-Tupel (M, K, C, e, d) bestehend aus:

- einer Menge M von Klartexten,
- einer Menge K von Schlüsseln,
- einer Menge C von Chiffretexten,
- einer Menge R von Zufallswerten (z.B. IVs),
- einer Verschlüsselungsfunktion $e : M \times K \times R \to C$,
- einer Entschlüsselungsfunktion $d : C \times K \to M$,

(Anmerkung: Die Entschlüsselung d benötigt den IV r, dieser wird aber typischerweise als Teil des Chiffretextes C übermittelt und nicht als separater Zufallseingang für d selbst.)

Sei $r \in R$ der Initialisierungsvektor (IV). **Verschlüsselung**

$$e^*(x_0x_1 \dots x_n, k, r) = y_0y_1 \dots y_n \text{ mit } y_0 = e(x_0 \oplus r, k) \quad \text{und} \quad y_i = e(x_i \oplus y_{i-1}, k) \quad \text{für } i \geq 1$$

Entschlüsselung

$$d^*(y_0y_1 \dots y_n, k, r) = x_0x_1 \dots x_n \text{ mit } x_0 = d(y_0, k) \oplus r \quad \text{und} \quad x_i = d(y_i, k) \oplus y_{i-1} \quad \text{für } i \geq 1$$

- Zur Verschlüsselung muss ein Wert $r \in R$ (der IV) gewählt werden.
- Zufallswerte aus R (IVs) sind nicht geheim, sie können unverschlüsselt mit dem Chiffrat gespeichert und verschickt werden (meist als erster Block).
- Wir wollen $e(x, k, r^1) \neq e(x, k, r^2)$ für $r^1 \neq r^2$.
- Wichtig für die Sicherheit ist, dass der IV (Zufallswert r) **eindeutig** (nie doppelt für denselben Schlüssel) und **unvorhersagbar** ist.
- Muster im Klartext sind im Chiffrat nicht mehr erkennbar.
- Gleiche Klartextblöcke werden unterschiedlich verschlüsselt.
- Ein fehlerhafter Chiffreblock y_i führt nur zur fehlerhaften Entschlüsselung des aktuellen Blocks x_i und des unmittelbar nachfolgenden Blocks x_{i+1}.
- Verschlüsselung ist **nicht** parallelisierbar (sequenziell), Entschlüsselung ist parallelisierbar.

CBC Padding Oracle Attack CBC ist anfällig für Padding-Oracle-Angriffe

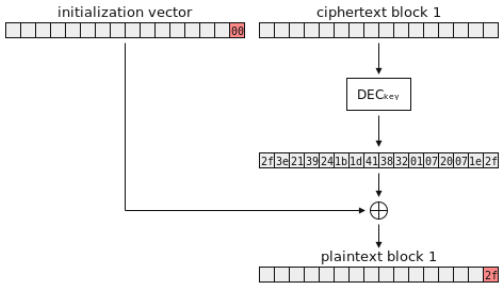
Ein solcher Angriff ermöglicht es, einen Geheimtext Schritt für Schritt zu entschlüsseln, ohne den Verschlüsselungsschlüssel zu kennen. Der Angreifer nutzt aus, wie ein Server auf fehlerhaftes Padding reagiert.

Der Angreifer:

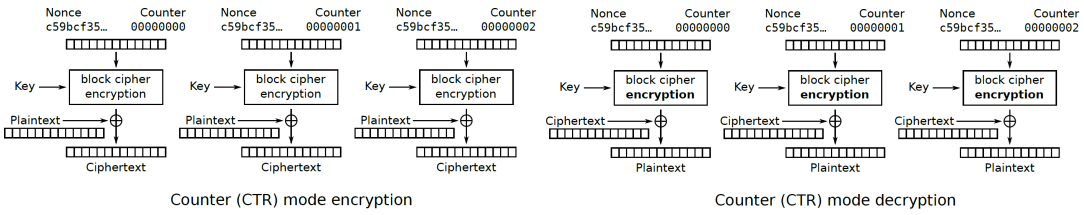
- hat keinen Zugriff auf den geheimen Schlüssel.
- ist in der Lage, gültige Chiffretexte abzufangen.
- ist in der Lage, modifizierte Versionen des Chiffretextes an das Orakel/Server zu senden und dessen Antworten zu beobachten.

Das Orakel (Server):

- muss ein überprüfbares Padding-Schema verwenden.
- muss dem Angreifer verraten, ob ein entschlüsselter Text ein gültiges (oder nicht) Padding hat. Dies kann geschehen durch:
  1. direkte Fehlermeldungen (z.B. HTTP 500) oder
  2. Side-Channel-Messungen (z.B. Unterschiede im Antwortverhalten).



Counter Mode (CTR) Modus



Um diesen Modus zu formalisieren, benötigen wir eine randomisierte Zählfunktion, die einen ”Nonce” (Number used once) r verwendet.

- Ein randomisierter Zähler (Funktion $ctr(r, i)$) bildet einen Zufallswert (Nonce r) und eine natürliche Zahl i (Zähler) auf eine Bitkette fester Länge ab.
- Eine einfache Implementierung benutzt die Binärdarstellung der natürlichen Zahl (LSB- oder MSB-Kodierung) mit 0-Padding, konkateniert mit der Nonce.
- Ein randomisierter Zähler $\text{ctr}(r, \cdot)$ sollte injektiv sein (für ein festes r). Man sollte die Periode (Wiederholung) so lang wie möglich wählen.
- Die Nonce r muss für denselben Schlüssel **nie** wiederverwendet werden.

Verschlüsselung:

$$e^*(x_0x_1 \dots x_n, k, r) = y_0y_1 \dots y_n \text{ mit } y_i = e(\text{ctr}(r, i), k) \oplus x_i$$

Entschlüsselung:

$$d^*(y_0y_1 \dots y_n, k, r) = x_0x_1 \dots x_n \text{ mit } x_i = e(\text{ctr}(r, i), k) \oplus y_i$$

Der CTR Modus unterscheidet sich stark von den vorher betrachteten Betriebsmodi:

- Ver- und Entschlüsselung nutzen beide die Verschlüsselungsfunktion e der Blockchiffre; die Entschlüsselungsfunktion d selbst wird nicht benötigt.
- Ver- und Entschlüsselung sind identisch (XOR mit dem Keystream).
- Die Berechnung des Keystreams $e(\text{ctr}(r, i), k)$ ist unabhängig vom zu verschlüsselnden Text.
- Ver- und Entschlüsselung können vollständig parallelisiert werden.
- CTR ist eine One-Time-Pad-Konstruktion, bei der die Blockchiffre als Pseudozufallsgenerator (Keystream-Generator) dient.

Advanced Encryption Standard (AES)

- Blocklänge ist 128 bereits
- AES-Schlüssel können $128, 192,$ oder256 bits lang sein

Sicherheit

- AES ist sicher solange die Implementierung und dazugehörige Systeme richtig konfiguriert sind (s. CBC Padding Attack)
- Schwache Schlüssel und IV-Generierung kann die Sicherheit von AES gefährden
- Side-channel Angriffe wie cache-timing und power analysis können verwendet werden, um den Schlüssel abzuleiten

Gegenmaßnahmen

- Konstantzeit-Implementierung (gegen Timing Angriffe): Ausführungszeit von Code soll unabhängig von den verarbeiteten geheimen Daten sein
-

Stromchiffren Stromchiffren können beliebig lange Bitketten verschlüsseln. Dabei sind Klar- und Chiffretexte beliebiger Länge, nur der Schlüssel hat eine feste Länge. Aus dem Schlüssel wird ein pseudozufälliger Schlüsselstrom erzeugt. Pseudozufallszahlen hängen von ihren Startparametern ab (seed) ab - gleiche Parameter liefern gleiche Zufallszahlen Ver- und Entschlüsselung ist ein bitweise exklusives oder (XOR) mit dem Schlüsselstrom. Ein Kryptosystem heißt Stromchiffre, wenn es eine Funktion $keystream(x, z) = |x|$ gibt, so dass $e(x, y) = d(x, z) = x \oplus keystream(x, y)$. Die Funktion keystream nenn wir Schlüsselstromgenerator und ihren Funktionswert Schlüsselstrom.

- Keystream sollte ein Pseudozufallszahlengenerator sein
- Keystream kann unabhängig vom INhalt der ersten Variable sein, also $keystream(x_1, k) = keystream(x_2, k)$ für beliebige x_1 und x_2 mit $|x_1| = |x_2|$
- Falls der Schlüsselstrom sich wieder, ist die Stromchiffre nicht mehr sicher

ChaCha20 ist eine moderne Stromchiffre, die als Alternative zu AES entwickelt wurde.

1.2 Kryptografische Hashfunktionen

Eine Hashfunktion ist ein Algorithmus, der eine Eingabe beliebiger Größe in einen Hashwert mit einer festen Länge umwandelt. Hashfunktionen sind deterministisch, erlauben eine schnelle Berechnung und bieten Integritätsschutz (Änderung der Eingabe ändert den Hash) Eigenschaften einer Hashfunktion:

1. **Pre-Image Resistance**
   Bei gegebenem Hashwert h ist es rechnerisch unmöglich, die ursprüngliche Nachricht m zu finden, so dass $H(m) = h$.
2. **Second-Image Resistance**
   Bei gegebener Nachricht m_1 ist es rechnerisch unmöglich, eine andere Nachricht m_2 zu finden, die denselben Hashwert erzeugt, so dass $H(m_1) = H(m_2)$
3. **Collision Resistance**
   Es ist rechnerisch unmöglich irgendwelche zwei unterschiedlichen Nachrichten m_1 und m_2 zu finden, die denselben Hashwert erzeugen, so dass $H(m_1) = H(m_2)$

Hashfunktion	**Output**	**Sicherheit**	**Anwendung**
MD5	128 Bits	Unsicher	X
SHA-1	160 Bits	Unsicher seit 2017	X
SHA-256	256 Bits	Sicher	TLS/SSL, hashing, Blockchain
SHA-3/Keccak	224/256/384/512 Bits	Sicher	Ähnlich wie SHA-2 (aber langsamer ohne Hardware Unterstützung)

Vergleich von Hashfunktionen