3 Betriebsarten bei Blockverschlüsselung

Die Anwendung einer Blockverschlüsselungsfunktion $f : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n$ auf längere (oder kürzere) Bitfolgen erfordert zwei Maßnahmen:

1. die Folge in n-Bit-Blöcke aufspalten,
2. den letzten Block auffüllen („padding“) mit
   - Nullen oder
   - Zufallswerten oder
   - Strukturinformationen.

Jeder Block wird dann im Prinzip einzeln verschlüsselt, wobei aber gewisse „Verkettungen“ üblich sind. Hierfür gibt es vier Standardverfahren, die „Betriebsarten“ oder „Modi“ genannt werden:

- ECB,
- CBC,
- CFB,
- OFB,

die zusammen mit dem DES-Verfahren normiert wurden, aber für beliebige Blockchiffren anwendbar sind. Fundstelle dieser Normen im Internet ist

`http://csrc.nist.gov/cryptval/`

Darüber hinaus gibt es natürlich auch Varianten und nicht standardisierte Verfahren; den aktuellen Diskussionsstand findet man in

`http://www.nist.gov/modes/`

Zur Beschreibung der Betriebsarten ist es meist sinnvoll, folgende allgemeine Situation zu betrachten: Σ sei ein „Blockalphabet“, etwa $\mathbb{F}_2^n$, versehen mit einer Gruppenoperation $*$. Ferner sei eine Verschlüsselungsfunktion

$$f : \Sigma \longrightarrow \Sigma$$

gegeben.

Die Abhängigkeit vom Schlüssel spielt hierbei keine Rolle und wird daher in der Notation weggelassen.

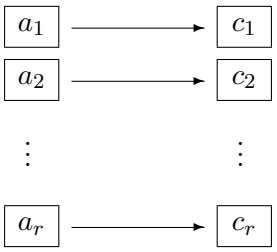
3.1 ECB = Electronic Code Book

Beschreibung

Sei r die Zahl der Blöcke, der Klartext also die Folge $(a_1, \ldots, a_r)$ von Blöcken.

Verschlüsselung: Beim ECB-Modus wird der Reihe nach einfach jeder Block für sich verschlüsselt:

$$a = (a_1, \ldots, a_r) \mapsto c = (c_1, \ldots, c_r) \in \Sigma^r \quad \text{mit } c_i = f(a_i).$$



Entschlüsselung: $a_i = f^{-1}(c_i)$.

Eigenschaften

Es handelt sich um eine monoalphabetische Substitution auf Σ. Falls $\#\Sigma$ sehr groß ist, ist das hinreichend sicher vor einem Geheimtextangriff. Nachteilig ist aber in jedem Fall:

- Information über identische Blöcke wird preisgegeben. Der Klartext ist zwar nicht zufällig, dennoch wird die Faustregel für das Geburtstagsphänomen hier oft interpretiert als: „Nach $2^{n/2}$ Bits beginnt beim ECB Information auszutreten.“ Durch die im folgenden behandelten Betriebsarten wird diese Grenze nach oben verschoben.

- Das Anlegen eines „Codebuchs“ aus bekannten Klartexten ist möglich. Bei strukturierten Nachrichten, z. B. Banktransaktionen, ist es ziemlich leicht, bekannte Klartextblöcke zu gewinnen.

- Ein aktiver Angriff durch Austausch oder Einschub einzelner Geheimtextblöcke (z. B. mit bekanntem, „sympathischen“ Klartext) ist möglich. Beispielsweise könnte ein Angreifer bei einer Banktransaktion, für die er weiß, in welchem Block der Empfänger definiert ist, diesen austauschen, um den Geldfluss umzuleiten. Was er dort hinschreiben muss, hat er aus einer früheren Transaktion als bekannten Klartextblock abgegriffen. Für diesen Angriff muss er den Schlüssel nicht kennen.

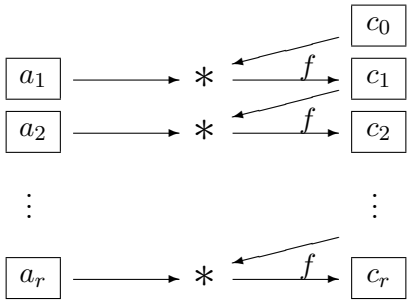
- Erlaubt die Situation einen Angriff mit gewähltem Klartext (Black-Box-Analyse), so ist Probeverschlüsselung bis hin zur Wörterbuch-Attacke möglich.

Besser ist es, eine Diffusion über die Klartextblöcke hinweg zu erzeugen. Das wird durch die im folgenden beschriebenen Betriebsarten erreicht.

3.2 CBC = Cipher Block Chaining

Beschreibung

Mit einem zufällig gewähltem Startwert c_0 (auch IV = „Initialisierungsvektor“ genannt) sieht das Verfahren so aus:



Verschlüsselung: Beim CBC-Modus wird nach folgender Formel verschlüsselt:

$$\begin{aligned} c_i &:= f(a_i * c_{i-1}) \quad \text{für } i = 1, \ldots, r \\ &= f(a_i * f(a_{i-1} * \cdots f(a_1 * c_0) \ldots)). \end{aligned}$$

Entschlüsselung: $a_i = f^{-1}(c_i) * c_{i-1}^{-1}$ für $i = 1, \ldots, r$.

Eigenschaften

- Jeder Geheimtextblock hängt von *allen vorhergehenden* Klartextblöcken ab (Diffusion).
- Ein Angreifer kann Textblöcke nicht unbemerkt ersetzen oder einfügen.
- Gleiche Klartextblöcke werden im allgemeinen verschieden chiffriert.
- Ein Angriff mit bekanntem Klartext ist hingegen, im Vergleich zum ECB, nicht erschwert.
- Jeder Klartextblock hängt von zwei Geheimtextblöcken ab.
- D. h., bei fehlerhafter Übermittlung eines Geheimtextblocks werden (nur) zwei Klartextblöcke unleserlich (≫Selbstsynchronisation≪ des Verfahrens).

Frage: *Kann der Startwert c_0 bei Geheimhaltung als zusätzlicher Schlüssel dienen?* (Das wären im Beispiel DES aus 56 Bits des Schlüssels und 64 Bits des Startwerts insgesamt 120 Bits.)

Antwort: Nein!

Begründung: Nur a_1 hängt beim Entschlüsseln von c_0 ab, d. h., es wird lediglich bekannter Klartext am Anfang etwas besser verschleiert, wenn c_0 geheim bleibt. Ist der zweite oder ein späterer Klartextblock bekannt, kann der Angreifer wie bei EBC den Schlüssel bestimmen (durch vollständige Suche oder einen anderen Angriff mit bekanntem Klartext.

Bemerkungen

1. CBC ist die Komposition $f\circ$ (Geheimtext-Autokey). Ist also $f = \mathbf{1}_\Sigma$, so bleibt das (völlig untaugliche) Geheimtext-Autokey-Verfahren mit Schlüssellänge 1 übrig.

2. (John Kelsey in der Mail-Liste `cryptography@c2.net`, 24 Nov 1999) Falls eine „Kollision“ $c_i = c_j$ für $i \neq j$ auftritt, folgt $f(a_i * c_{i-1}) = f(a_j * c_{j-1})$, also $a_i * c_{i-1} = a_j * c_{j-1}$ und daraus $a_j^{-1} * a_i = c_{j-1} * c_{i-1}^{-1}$. Der Gegner gewinnt also etwas Information über den Klartext.

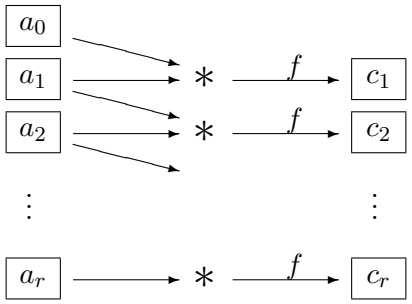
   Erwarten kann man diese Situation wegen des Geburtstagsphänomens nach ca. $\sqrt{\#\Sigma}$ Blöcken.

   Je länger der Text, desto mehr solcher Kollisionen sind zu erwarten. Auch dies bestätigt wieder die Faustregel über die Frequenz nötiger Schlüsselwechsel: rechtzeitig bevor $\sqrt{\#\Sigma}$ Blöcke erreicht sind.

3.3 Varianten des CBC

Klartext-Autokey

Ersetzt man beim CBC das Geheimtext-Autokey-Verfahren durch Klartext-Autokey, so erhält man folgendes Schema:



welches man PBC = Plaintext Block Chaining nennen könnte.

Verschlüsselung: Die Verschlüsselung folgt nach Wahl eines Startwertes a_0 der Formel:

$$c_i := f(a_i * a_{i-1}) \quad \text{für } i = 1, \ldots, r.$$

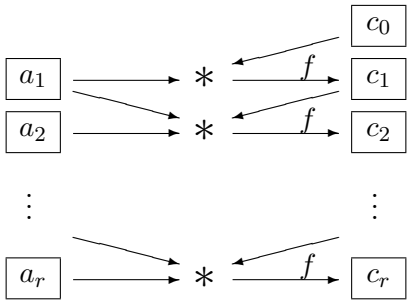
Entschlüsselung: Die Formel für die Entschlüsselung heißt:

$$a_i = f^{-1}(c_i) * a_{i-1}^{-1} \quad \text{für } i = 1, \ldots, r.$$

Dieses Verfahren ist allerdings völlig unüblich, und über seine Sicherheit ist anscheinend nichts bekannt.

PCBC = error-Propagating CBC

Dieses Verfahren ist ein Mix aus CBC und PBC und folgt dem Schema



Verschlüsselung: Die Verschlüsselung folgt (mit dem Startwert $a_0 =$ neutrales Element der Gruppe) der Formel:

$$c_i := f(a_i * a_{i-1} * c_{i-1}) \quad \text{für } i = 1, \ldots, r.$$

Für die Bitblock-Chiffrierung wird a_0 also als Nullblock gewählt.

Entschlüsselung: Die Formel für die Entschlüsselung heißt:

$$a_i = f^{-1}(c_i) * c_{i-1}^{-1} * a_{i-1}^{-1} \quad \text{für } i = 1, \ldots, r.$$

Dieses Verfahren wurde bei älteren Versionen von Kerberos verwendet, wegen gewisser Schwächen inzwischen jedoch aufgegeben.

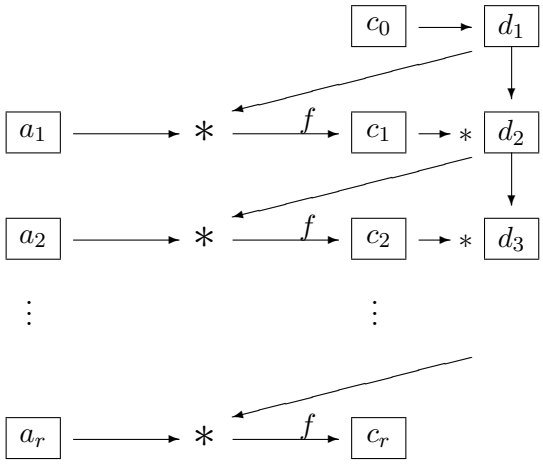
Verallgemeinerung nach MEYER/MATYAS

$$c_i := f(a_i * h(a_{i-1}, c_{i-1})) \quad \text{für } i = 1, \ldots, r,$$

wobei im Falle $\Sigma = \mathbb{F}_2^n$ z. B. für h die Addition modulo 2^n vorgeschlagen wird.

BCM = Block Chaining Mode

Diese Betriebsart folgt dem Schema:



Formel für die Verschlüsselung:

$$\begin{aligned} d_i &:= c_0 * \ldots * c_{i-1}, \\ c_i &:= f(a_i * d_i) \quad \text{für } i = 1, \ldots, r. \end{aligned}$$

Eine Anwendung des CBC

Der CBC-MAC (= „Message Authentication Code“) ist eine schlüsselabhängige „Hash-Funktion“, die zur Integritätsprüfung von Nachrichten verwendet wird. Sie ist in ISO/IEC 9797 normiert und mit dem DES-Verfahren im Bankenbereich verbreitet.

Sender und Empfänger einer Nachricht – die auch identisch sein können, wenn es sich um Nachrichtenspeicherung handelt – haben den Schlüssel k gemeinsam und verwenden die Verschlüsselungsfunktion $f = f_k$.

Der MAC eines Textes $a = (a_1, \ldots, a_r)$ ist der letzte Geheimtextblock, wenn a nach dem CBC verschlüsselt wird, also

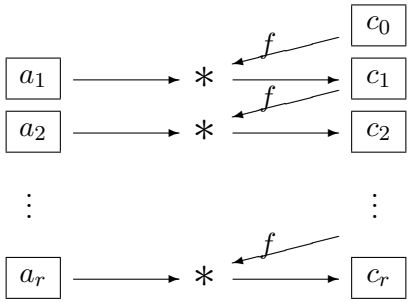
$$\mathrm{MAC}(a) = c_r = f(a_r * f(a_{r-1} * \cdots f(a_1 * c_0) \ldots)).$$

Wird $\mathrm{MAC}(a)$ zusammen mit a verschickt, kann der Empfänger die Echtheit von Absender und Inhalt prüfen, denn nur wer den Schlüssel hat, kann diesen Wert richtig berechnen.

Der Nachteil des geteilten Geheimnisses k ist allerdings, dass in einem Rechtsstreit zwischen den beiden Parteien jeder dem anderen eine Fälschung unterstellen kann.

3.4 CFB = Cipher Feedback

Beschreibung (in der einfachsten Version)



Verschlüsselung: Beim CFB-Modus wird nach folgender Formel verschlüsselt:

$$\begin{aligned} c_i &:= a_i * f(c_{i-1}) \quad \text{für } i = 1, \ldots, r \\ &= a_i * f(a_{i-1} * f(\cdots a_1 * f(c_0) \ldots)). \end{aligned}$$

Entschlüsselung: $a_i = c_i * f(c_{i-1})^{-1}$ für $i = 1, \ldots, r$.

Eigenschaften

- Der Startwert taugt auch hier nicht als zusätzlicher Schlüssel.
- Ein Angriff mit bekanntem Klartext wird auch durch diese Betriebsart nicht erschwert.
- Bemerkenswert ist, dass man auch zum Entschlüsseln nur f braucht, nicht etwa f^{-1}. Im Vorgriff sei hier schon vermerkt:
  - Der CFB-Modus ist für asymmetrische Chiffren ungeeignet.
  - Er kann aber mit einer echten (natürlich schlüsselabhängigen) Einweg- oder Hash-Funktion verwendet werden.
- Der CFB reduziert sich im Falle der identischen Abbildung $f = \mathbf{1}_\Sigma$ ebenfalls auf das Geheimtext-Autokey-Verfahren.
- (David Wagner) ECB $\circ$ CFB = CBC:

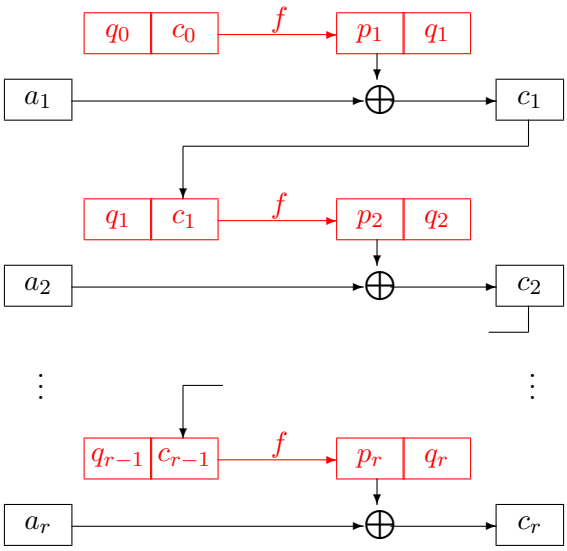
Dazu setzt man c_0 als Startwert für CFB und $c_0' := f(c_0)$ als Startwert für

CBC. Dann ist

$$\begin{aligned} c_1 &= \mathrm{CFB}(a_1) = a_1 * f(c_0), \\ c_1' &= \mathrm{ECB}(c_1) = f(a_1 * f(c_0)) = f(a_1 * c_0') = \mathrm{CBC}(a_1), \\ c_2 &= \mathrm{CFB}(a_2) = a_2 * f(c_1), \\ c_2' &= \mathrm{ECB}(c_2) = f(a_2 * f(c_1)) = f(a_2 * c_1') = \mathrm{CBC}(a_2), \\ &\text{usw.} \end{aligned}$$

Die genormte Version

... verwendet ein Schieberegister und ist daher nur im Falle $\Sigma = \mathbb{F}_2^n$ definiert. Hier ist $1 \leq t \leq n$, und verschlüsselt werden Blöcke $a_i \in \mathbb{F}_2^t$ der Länge t; der aktuelle Geheimtextblock c_i der Länge t wird von rechts in das (hier rot dargestellte) Schieberegister nachgeschoben:

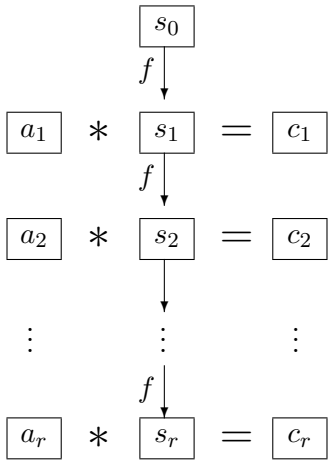


Die q_i sind dabei stets Bitblöcke der Länge $n - t$.

Diese allgemeinere Version ist weniger sicher als die mit $t = n$ und wird daher kaum verwendet.

3.5 OFB = Output Feedback

Beschreibung (in der einfachsten Version)



Auch diese Betriebsart war ursprünglich als Schieberegister-Version definiert; auch hier bedeutet die Verwendung eines $t <$ Blocklänge n eine Schwächung des Verfahrens [Jueneman, Crypto 82].

Verschlüsselung: Beim OFB-Modus wird nach folgender Formel verschlüsselt:

$$c_i := a_i * s_i, \quad s_i := f(s_{i-1}) \quad \text{für } i = 1, \ldots, r.$$

Entschlüsselung: nach der Formel

$$a_i = c_i * s_i^{-1}, \quad s_i := f(s_{i-1}) \quad \text{für } i = 1, \ldots, r.$$

Eigenschaften

- Es gibt keine Diffusion, aber gleiche Klartextblöcke werden im allgemeinen verschieden verschlüsselt.
- Im Falle $\Sigma = \mathbb{F}_2^s$ handelt es sich um eine Bitstrom-Chiffre mit f als „Zufallsgenerator“.
- Wird eine besonders schnelle Ver- und Entschlüsselung benötigt, so kann man den „Schlüsselstrom“ s_i auf beiden Seiten (beim Sender und beim Empfänger) vorausberechnen.
- Auch hier wird zum Entschlüsseln nur f benötigt, nicht f^{-1}.
- Falls $\Sigma = \mathbb{F}_2^s$, ist die Chiffre sogar involutorisch, d. h., Verschlüsselung = Entschlüsselung (als Funktion). Allgemeiner gilt das, wenn Σ eine Gruppe vom Exponenten 2 ist.

- Bei einem Angriff mit bekanntem Klartext liefert ein Paar (a_1, c_1) den Wert s_1, ein weiteres Paar (a_2, c_2) den Wert $f(s_1)$. Damit ist also ein Angriff mit bekanntem Klartext auf f selbst möglich.
- Das Geheimhalten des Startwerts s_0 bringt also auch hier praktisch keine zusätzliche Sicherheit.

Variante: Der Counter-Mode CTR

Hier ist im einfachsten Fall

$$c_i := a_i * f(i) \quad \text{für } i = 1, \ldots, r.$$