3 Uneingeschränkte symmetrische Verschlüsselung

Szenarium 3: Alice möchte Bob *mehrere* Klartexte *beliebiger* Länge schicken. Sie verwendet dafür immer denselben Schlüssel. Eva hört die Chiffretexte mit und kann sich *einige wenige* Klartexte mit dem verwendeten Schlüssel verschlüsseln lassen.

Erweiterungen im Vergleich zu vorher:

(1) Beliebig lange Texte sind erlaubt.

(2) Mehrfaches Senden desselben Textes ist möglich; Eva sollte dies aber nicht erkennen.

Wir müssen die bisher benutzten Konzepte anpassen, um auch das mehrfache Senden derselben Nachricht zu erfassen. Ein grundlegender Ansatz, um mit identischen Botschaften umzugehen, ist Folgendes: Alices Verschlüsselungsalgorithmus ist *randomisiert*, liefert also zufallsabhängig unterschiedliche Chiffretexte für ein und denselben Klartext. Allerdings ist normalerweise die Anzahl der Zufallsexperimente fest und nicht von der Klartextlänge abhängig, so dass wir wie vorher davon ausgehen können, dass nur ein Experiment am Anfang ausgeführt wird. Auch der Sicherheitsbegriff und die Rechenzeitbeschränkungen müssen verändert werden.

Klartexte und Chiffretexte sind nun endliche Folgen von Bitvektoren („Blöcken“) der festen Länge ℓ. Die Menge aller dieser Folgen bezeichnen wir mit $(\{0,1\}^\ell)^*$ oder kürzer mit $\{0,1\}^{\ell\cdot *}$. Es gibt also unendlich viele Klartexte und Chiffretexte. Die Menge der Schlüssel heißt K. Der Verschlüsselungsalgorithmus E ist randomisiert und transformiert einen Klartext x und einen Schlüssel k in einen Chiffretext. Der Entschlüsselungsalgorithmus D ist deterministisch und transformiert einen Chiffretext y und einen Schlüssel k in einen Klartext. Sicher muss man den Algorithmen eine Rechenzeit zugestehen, die von der Länge der zu verarbeitenden Texte abhängt. Als effizient werden Algorithmen angesehen, die eine polynomielle Rechenzeit haben. Es muss eine verallgemeinerte Dechiffrierbedingung gelten, die die Randomisierung der Verschlüsselung berücksichtigt.

Definition 3.1 *Ein* symmetrisches ℓ-Kryptoschema *ist ein Tupel*

$$\mathcal{S} = (K, E, D)\,,$$

wobei

- $K \subseteq \{0,1\}^s$ *eine endliche Menge ist (für ein* $s \in \mathbb{N}$*),*
- $E(x\colon \{0,1\}^{\ell\cdot *}, k\colon K)\colon \{0,1\}^{\ell\cdot *}$ *ein randomisierter Algorithmus und*
- $D(y\colon \{0,1\}^{\ell\cdot *}, k\colon K)\colon \{0,1\}^{\ell\cdot *}$ *ein deterministischer Algorithmus*

sind, so dass gilt:

- *Die Laufzeiten von E und D sind polynomiell beschränkt in der Länge von x bzw. y.*
- *Für jedes $x \in \{0,1\}^{\ell \cdot *}$, $k \in K$ und jedes $m \in M_1 \times \cdots \times M_r$ (die Ausgänge der flip-Anweisungen in E) gilt:*

$$D(E^m(x,k),k) = x \qquad \textit{(Dechiffrierbedingung).}$$

Die Elemente von

- K heißen „Schlüssel“
- $\{0,1\}^{\ell \cdot *}$ heißen „Klartexte“ bzw. „Chiffretexte“, je nachdem, welche Rolle sie gerade spielen.

E ist der *Chiffrieralgorithmus*, D der *Dechiffrieralgorithmus*.

Bemerkungen:

- Zentral: Die Nachrichtenlänge ist unbestimmt.
- Wir werden immer davon ausgehen, dass der Schlüssel k uniform zufällig aus K gewählt wurde und beiden Parteien bekannt ist. Die Angreiferin Eva kennt k natürlich nicht.
- Etwas allgemeiner ist es, wenn man den Schlüssel nicht uniform zufällig aus einer Menge wählt, sondern von einem randomisierten Schlüsselerzeugungsalgorithmus $G(s\colon \text{integer})\colon \{0,1\}^*$ generieren lässt. Konzeptuell besteht zwischen den beiden Situationen aber kein großer Unterschied.
- Jeder Klar- und jeder Chiffretext ist ein Bitvektor der Länge $\ell \cdot h$ für ein $h \in \mathbb{N}$. (Um auf ein Vielfaches der Blocklänge zu kommen, muss man die Texte notfalls mit Nullen auffüllen.)
- Um einen Klartext x zu verschicken, wird der Algorithmus E mit einem *neuen, uniform zufällig gewählten* Element m abgearbeitet und es wird $E^m(x,k)$ als Chiffretext verschickt. Insbesondere entsteht bei wiederholter Verschlüsselung eines Textes x (mit sehr großer Wahrscheinlichkeit) jedesmal ein anderer Chiffretext.
- In der Literatur finden sich auch Kryptoschemen, bei denen auch die Dechiffrierung randomisiert ist. Wir betrachten dies hier nicht.

Der Standardansatz zur Konstruktion eines Kryptoschemas besteht darin, von einer Blockchiffre wie in Kapitel 2 auszugehen und sie zu einem Kryptoschema auszubauen. Dies wird im Folgenden beschrieben.

3.1 Betriebsarten

Symmetrische Kryptoschemen erhält man z. B. aus Blockchiffren. Durch einfache Regeln wird erklärt, wie ein Klartext, der aus einer Folge von Blöcken besteht, zu chiffrieren ist. Für alle folgenden Konstruktionen sei $\mathcal{B} = (\{0,1\}^\ell, K_\mathcal{B}, \{0,1\}^\ell, e_\mathcal{B}, d_\mathcal{B})$ ein ℓ-Block-Kryptosystem für Blöcke einer Länge ℓ. (Man darf sich hier zum Beispiel AES mit $\ell = 128$ vorstellen, oder eine Variante von DES.)

3.1.1 ECB-Betriebsart („*Electronic Code Book mode*“)

Dies ist die nächstliegende Methode. Ein Schlüssel ist ein Schlüssel k von $\mathcal{B}$. Man verschlüsselt einfach die einzelnen Blöcke von x mit $\mathcal{B}$, jedesmal mit demselben Schlüssel k.

Definition: Das zu $\mathcal{B}$ gehörende ***ℓ-ECB-Kryptoschema*** $\mathcal{S} = \mathrm{ECB}(\mathcal{B}) = (K_\mathcal{B}, E, D)$ ist gegeben durch die folgenden Algorithmen:

$\underline{E(x\colon \{0,1\}^{\ell\cdot *}, k\colon K_\mathcal{B})\colon \{0,1\}^{\ell\cdot *}}$
zerlege x in Blöcke der Länge ℓ: $x = x_0 x_1 \dots x_{m-1}$;
für $0 \le i < m$ setze $y_i \leftarrow e_\mathcal{B}(x_i, k)$;
gib $y = y_0 \dots y_{m-1}$ zurück.

$\underline{D(y\colon \{0,1\}^{\ell\cdot *}, k\colon K_\mathcal{B})\colon \{0,1\}^{\ell\cdot *}}$
zerlege y in Blöcke der Länge ℓ: $y = y_0 y_1 \dots y_{m-1}$;
für $0 \le i < m$ setze $x_i \leftarrow d_\mathcal{B}(y_i, k)$;
gib $x = x_0 \dots x_{m-1}$ zurück.

Die Verschlüsselung verzichtet auf die Option, Randomisierung zu verwenden. (Sie hat den großen Vorteil, parallel ausführbar zu sein.) Es ist klar, dass die Dechiffrierbedingung erfüllt ist. Jedoch hat dieses Kryptoschema ein ziemlich offensichtliches Problem, nämlich, dass ein Block $x \in \{0,1\}^\ell$ immer gleich verschlüsselt wird, Eva also ganz leicht nichttriviale Informationen aus dem Chiffretext erhalten kann. Zum Beispiel kann sie sofort sehen, ob der Klartext die Form $x = x_1 x_1$, mit $x_1 \in \{0,1\}^\ell$, hat oder nicht.

Das Problem wird augenfällig zum Beispiel bei der Verschlüsselung von Bildern. Ein Bild ist dabei ein rechteckiges Schema (eine Matrix) von „Pixeln“, denen jeweils ein Farbcode (z. B. 1 Byte) zugeordnet ist. Man könnte dabei die Pixel in Gruppen (quadratische Blöcke oder Zeilensegmente) unterteilen. Das Farbmuster jeder solchen Gruppe liefert einen Klartextblock. Wenn viele Bildteile identisch aussehen, etwa gleich konstant gefärbt sind,

ergibt sich jeweils derselbe Klartext für den entsprechenden Block, und damit auch derselbe Chiffretext. Wenn man den Chiffretext bildlich darstellt, ergibt sich leicht ein grober Eindruck des Originalbildes. Als Beispiel betrachte diese Bilder aus Wikipedia (auch in der Vorlesung gezeigt): `https://de.wikipedia.org/wiki/Electronic_Code_Book_Mode` .

Fazit: Obwohl er so naheliegend ist, sollte der ECB-Modus *niemals* benutzt werden!

3.1.2 CBC-Betriebsart („*Cipher Block Chaining mode*“)

Diese Betriebsart weicht dem zentralen Problem von ECB aus, indem man die Blöcke in Runden $i = 0, 1, \ldots, m-1$ nacheinander verschlüsselt und das Ergebnis einer Runde zur Modifikation des Klartextblocks der nächsten Runde benutzt. Konkret: Es wird nicht x_i mit $\mathcal{B}$ verschlüsselt, sondern $x_i \oplus_\ell y_{i-1}$ (bitweises XOR). Man benötigt dann einen Anfangsvektor y_{-1} für die erste Runde. Dieser ist Teil des Schlüssels des Kryptoschemas (nicht von $\mathcal{B}$), ein Schlüssel des Schemas ist also ein Paar (k, v) mit $k \in K_\mathcal{B}$ und $v \in \{0,1\}^\ell$.

Definition: Das zu $\mathcal{B}$ gehörende ***ℓ-CBC-Kryptoschema*** $\mathcal{S} = \mathrm{CBC}(\mathcal{B}) = (K_\mathcal{B} \times \{0,1\}^\ell, E, D)$ ist durch die folgenden Algorithmen gegeben:

$\underline{E(x\colon \{0,1\}^{\ell *}, (k,v)\colon K_\mathcal{B} \times \{0,1\}^\ell)\colon \{0,1\}^{\ell *}}$
zerlege x in Blöcke der Länge ℓ: $x = x_0 x_1 \ldots x_{m-1}$;
$y_{-1} \leftarrow v$;
für $i = 0, \ldots, m-1$ nacheinander: $y_i \leftarrow e_\mathcal{B}(x_i \oplus_\ell y_{i-1}, k)$;
gib $y = y_0 \ldots y_{m-1}$ zurück.

$\underline{E(x\colon \{0,1\}^{\ell *}, (k,v)\colon K_\mathcal{B} \times \{0,1\}^\ell)\colon \{0,1\}^{\ell *}}$
zerlege y in Blöcke der Länge ℓ: $y = y_0 y_1 \ldots y_{m-1}$
$y_{-1} \leftarrow v$;
für $i = 0, \ldots, m-1$ nacheinander: $x_i \leftarrow d_\mathcal{B}(y_i, k) \oplus_\ell y_{i-1}$;
gib $x = x_0 \ldots x_{m-1}$ zurück.

Der Vektor v wird *Initialisierungsvektor* genannt. Man versteht recht gut, was beim Chiffrieren passiert, wenn man sich das Bild auf Seite 104 im Buch von Küsters/Wilke ansieht. Beim Dechiffrieren geht man den umgekehrten Weg: Entschlüssele einen Block y_i mittels $\mathcal{B}$, *dann* addiere y_{i-1}, um den Klartextblock x_i zu erhalten. Es ist klar, dass die Dechiffrierbedingung erfüllt ist.

Interessant ist die folgende Eigenschaft der Entschlüsselung im CBC-Modus: Wenn bei der Übertragung des Chiffretextes ein einzelner Block y_i durch einen Fehler zu y_i' verfälscht wird, dann ist die Entschlüsselung ab Block y_{i+2} trotzdem wieder korrekt.

In diesem Kryptoschema ist der zentrale Nachteil der ECB-Betriebsart verschwunden: Identische Klartextblöcke führen nun praktisch immer zu verschiedenen Chiffretextblöcken. Die Verschlüsselung von $x = x_1x_1$ mit $x_1 \in \{0,1\}^\ell$ liefert i. a. keinen Chiffretext der Form $y = y_1y_1$. Die oben erwähnte Wikipedia-Seite zeigt auch, dass bei der Verschlüsselung des Bildes mit CBC keine offensichtlichen Probleme mehr auftauchen.

Ein Problem bleibt aber bestehen: Wird zweimal der Klartext x verschlüsselt, so geschieht dies immer durch denselben Chiffretext $y = E(x,(k,v))$. Dies ist eine Folge der Eigenschaft von CBC, *deterministisch* zu sein. Auch CBC wird man in der Praxis daher nicht benutzen.

3.1.3 R-CBC-Betriebsart („*Randomized CBC mode*“)

Um das Problem der identischen Verschlüsselung identischer Klartexte zu beseitigen, muss in die Verschlüsselung eine Zufallskomponente eingebaut werden. Beispielsweise kann man dazu CBC leicht modifizieren. Der Initialisierungsvektor $y_{-1} = v \in \{0,1\}^\ell$ ist nicht mehr Teil des Schlüssels, sondern wird vom Verschlüsselungsalgorithmus einfach zufällig gewählt, und zwar für jeden Klartext immer aufs Neue. Damit der Empfänger entschlüsseln kann, benötigt er v. Daher wird y_{-1} als Zusatzkomponente dem Chiffretext vorangestellt. Damit ist der Chiffretext um einen Block länger als der Klartext, und Eva kennt auch $v = y_{-1}$.

Definition: Das zu $\mathcal{B}$ gehörende ***ℓ-R-CBC-Kryptoschema*** $S = \text{R-CBC}(\mathcal{B}) = (K_\mathcal{B}, E, D)$ ist gegeben durch die folgenden Algorithmen:

$E(x\colon \{0,1\}^{\ell\cdot *}, k\colon K_\mathcal{B})\colon \{0,1\}^{\ell\cdot *}$;
zerlege x in m Blöcke der Länge ℓ: $x = x_0x_1\dots x_{m-1}$
setze $y_{-1} = \text{flip}(\{0,1\}^\ell)$;
für $i = 0,\dots,m-1$ nacheinander: $y_i \leftarrow e_\mathcal{B}(x_i \oplus_\ell y_{i-1}, k)$;
gib $y = y_{-1}y_0\dots y_{m-1}$ zurück. // Länge: $m+1$ Blöcke

$D(y\colon \{0,1\}^{\ell\cdot *}, k\colon K_\mathcal{B})\colon \{0,1\}^{\ell\cdot *}$;
zerlege y in $m+1$ Blöcke der Länge ℓ: $y = y_{-1}y_0y_1\dots y_{m-1}$
für $i = 0,\dots,m-1$ nacheinander: $x_i \leftarrow d_\mathcal{B}(y_i,k) \oplus_\ell y_{i-1}$;
gib $x = x_0\dots x_{m-1}$ zurück. // Länge: m Blöcke

Es gibt zwei Unterschiede zu CBC:

1. Für jede Verschlüsselung eines Klartextes wird ein neuer zufälliger Initialisierungsvektor verwendet. Dadurch wird ein Klartext x bei mehrfachem Auftreten mit hoher Wahrscheinlichkeit immer verschieden verschlüsselt.

2. Der Initialisierungsvektor ist nicht Teil des geheimen Schlüssels, sondern ist dem Angreifer bekannt, da er Teil des Chiffretextes ist. (Intuitiv würde man vielleicht sagen, dass dies „die Sicherheit verringert“.)

Tatsächlich und etwas unerwartet kann man nach einer sorgfältigen Formulierung eines Sicherheitsbegriffs für Kryptoschemen beweisen, dass die Betriebsart R-CBC zu „sicheren“ Verfahren führt, wenn die zugrundeliegende Blockchiffre „sicher“ ist. (Der Beweis ist aufwendig.)

Warnung, als Beispiel für harmlos erscheinende Modifikationen, die Kryptoschemen unsicher machen:

1. Man könnte meinen, dass es genügt, bei jeder Verschlüsselung einen neuen Initialisierungsvektor zu benutzen, also zum Beispiel nacheinander v, $v+1$, $v+2$, ... Dies führt jedoch zu einem unsicheren Kryptoschema.

2. Um Kommunikation zu sparen, könnte man auf die Idee kommen, dass Alice und Bob sich von einer Kommunikationsrunde zur nächsten den letzten Chiffretextblock merken und ihn bei der nächsten Runde als Initialisierungsvektor benutzen. Dieses Verfahren heißt „chained CBC“ und wurde in SSL 3.0 und TLS 1.0 verwendet. Es stellte sich heraus, dass dieses Verfahren mit einem Angriff mit gewähltem Klartext erfolgreich attackiert werden kann!

3.1.4 OFB-Betriebsart („*Output FeedBack mode*“)

Wir kommen nun zu zwei Betriebsarten, die einen ganz anderen Ansatz für die Verschlüsselung der einzelnen Blöcke von x verfolgen. Es wird dazu nämlich *nicht* $\mathcal{B}$ mit Schlüssel k benutzt, sondern der Mechanismus des Vernam-Systems (One-Time-Pads, siehe Beispiel 1.6): $y_i = x_i \oplus_\ell k_i$, wobei $k_i \in \{0,1\}^\ell$ ein „Rundenschlüssel“ für Block x_i ist. Das Kryptosystem $\mathcal{B}$ wird nur dafür benutzt, diese Rundenschlüssel herzustellen, die bei Verschlüsselung und bei Entschlüsselung identisch sind. Die Dechiffrierbedingung folgt dann daraus, dass das Vernam-System korrekt dechiffriert. Der Ansatz führt dazu, dass die Entschlüsselungsfunktion $d_\mathcal{B}$ des Block-Kryptosystems gar nicht benötigt wird.

Zuerst betrachten wir die Betriebsart „*Output Feedback*“. Dabei wird ein zufälliger Startvektor v aus $\{0,1\}^\ell$ gewählt. Man setzt $k_{-1} = v$, und konstruiert die Rundenschlüssel $k_0, \ldots, k_{m-1}$ dadurch, dass man iteriert den letzten Rundenschlüssel durch die Verschlüsselungsfunktion von $\mathcal{B}$ schickt: $k_i = e_\mathcal{B}(k_{i-1}, k)$, für $i = 0, 1, \ldots, m-1$. (Der Name “Output Feedback” rührt daher, dass das Ergebnis einer Verschlüsselung durch $e_\mathcal{B}$ wieder als Input des nächsten Aufrufs von $e_\mathcal{B}$ benutzt wird.) Der Empfänger benötigt v, um seinerseits die Rundenschlüssel zu berechnen; daher wird v als y_{-1} dem Chiffretext vorangestellt, wie beim R-CBC-Modus.

Definition: Das zu $\mathcal{B}$ gehörende ***ℓ-OFB-Kryptoschema*** $S = (K_\mathcal{B}, E, D) = \mathrm{OFB}(\mathcal{B}) = (K_\mathcal{B}, E, D)$ ist gegeben durch die folgenden Algorithmen:

$\underline{E(x\colon \{0,1\}^{\ell\cdot *}, k\colon K_\mathcal{B})\colon \{0,1\}^{\ell\cdot *}}$;
zerlege x in m Blöcke der Länge ℓ: $x = x_0 x_1 \dots x_{m-1}$;
$k_{-1} \leftarrow y_{-1} \leftarrow \mathrm{flip}(\{0,1\}^\ell)$;
für $i = 0, \dots, m-1$ nacheinander: $k_i \leftarrow e_\mathcal{B}(k_{i-1}, k)$ und $y_i \leftarrow x_i \oplus_\ell k_i$;
gib $y = y_{-1} y_0 \dots y_{m-1}$ zurück.

$\underline{D(y\colon \{0,1\}^{\ell\cdot *}, k\colon K_\mathcal{B})\colon \{0,1\}^{\ell\cdot *}}$;
zerlege y in $m+1$ Blöcke der Länge ℓ: $y = y_{-1} y_0 y_1 \dots y_{m-1}$;
$k_{-1} \leftarrow y_{-1}$;
für $i = 0, \dots, m-1$ nacheinander: $k_i \leftarrow e_\mathcal{B}(k_{i-1}, k)$ und $x_i \leftarrow y_i \oplus k_i$;
gib $x = x_0 \dots x_{m-1}$ zurück.

Dieses Verfahren hat einen interessanten Vorteil. Oft werden die Blöcke des Chiffretextes beim Empfänger nacheinander eintreffen. Die Hauptarbeit, nämlich die iterierte Verschlüsselung mit $e_\mathcal{B}$ zur Ermittlung der Rundenschlüssel k_i, kann unabhängig von der Verfügbarkeit der Klartextblöcke erfolgen, sobald y_{-1} eingetroffen ist.

Man kann beweisen, dass die Betriebsarten R-CBC und OFB „sicher“ sind, wenn die zugrundeliegende Blockchiffre „sicher“ ist. Dazu weiter unten mehr.

3.1.5 R-CTR-Betriebsart („*Randomized CounTeR mode*“)

Dies ist die zweite Betriebsart, bei der das Kryptosystem $\mathcal{B}$ nur zur Herstellung von m „Rundenschlüsseln“ benutzt wird, mit denen dann die Blöcke per $\oplus_\ell$ verschlüsselt werden. Anstatt iteriert zu verschlüsseln, was bei OFB eine sequentielle Verarbeitung erzwingt, werden hier die mit $\mathcal{B}$ zu verschlüsselnden Strings anders bestimmt. Man fasst $\{0,1\}^\ell$ als äquivalent zur Zahlenmenge $\{0, 1, \dots, 2^\ell - 1\}$ auf, interpretiert einen ℓ-Bit-String also als Block oder als Zahl, wie es passt. In dieser Menge wählt man eine Zufallszahl r. Man „zählt“ von r ausgehend nach oben und berechnet die Rundenschlüssel $k_0, \dots, k_{m-1}$ durch Verschlüsseln von $r, r+1, \dots, r+m-1$ (modulo 2^ℓ gerechnet) mittels $e_\mathcal{B}(\,.\,, k)$. Rundenschlüssel k_i ist also $e_\mathcal{B}((r+i) \bmod 2^\ell, k)$, und Chiffretextblock y_i ist $k_i \oplus_\ell x_i$. Interessant ist, dass hier die Berechnung der Rundenschlüssel und die Ver- bzw. Entschlüsselung der Blöcke parallel erfolgen kann, also sehr schnell, falls mehrere Prozessoren zur Verfügung stehen.

Definition: Das zu $\mathcal{B}$ gehörende ***ℓ-R-CTR-Kryptoschema*** $\mathcal{S} = \text{R-CTR}(\mathcal{B}) = (K_\mathcal{B}, E, D)$ ist gegeben durch die folgenden Algorithmen:

$\underline{E(x\colon \{0,1\}^{\ell\cdot *}, k\colon K_{\mathcal{B}})\colon \{0,1\}^{\ell\cdot *}};$
zerlege x in m Blöcke der Länge ℓ: $x = x_0 x_1 \dots x_{m-1}$;
$r \leftarrow \mathrm{flip}(\{0, \dots, 2^\ell - 1\})$;
für $0 \le i < m$: $y_i \leftarrow e_{\mathcal{B}}((r+i) \bmod 2^\ell, k) \oplus_\ell x_i$;
gib $y = r y_0 \dots y_{m-1}$ zurück.

$\underline{D(y\colon (\{0,1\}^\ell)^+, k\colon K_{\mathcal{B}})\colon \{0,1\}^{\ell\cdot *}};$
zerlege y in $m+1$ Blöcke der Länge ℓ: $y = y_{-1} y_0 y_1 \dots y_{m-1}$;
$r \leftarrow y_{-1}$;
für $0 \le i < m$: $x_i \leftarrow e_{\mathcal{B}}((r+i) \bmod 2^\ell, k) \oplus_\ell y_i$;
gib $x = x_0 \dots x_{m-1}$ zurück.

Es ist offensichtlich, dass die Dechiffrierbedingung erfüllt ist.

Bemerkungen:

- Wie bei R-CBC und OFB wird hier ein zufälliger Initialisierungswert r verwendet, der als Teil des Chiffretextes dem Angreifer bekannt ist.
- Wie bei OFB wird die Entschlüsselungsfunktion $d_{\mathcal{B}}$ gar nicht verwendet, man kann also anstelle der Verschlüsselungsfunktion $e_{\mathcal{B}}$ eine beliebige Funktion $e_{\mathcal{B}}\colon \{0,1\}^\ell \times \{0,1\}^\ell \to \{0,1\}^\ell$ benutzen, bei der die „Chiffren“ $e_{\mathcal{B}}(\,.\,, k)$ nicht einmal injektiv sein müssen.
- Man kann dieses Kryptoschema auch wie folgt verstehen: Zu einem gegebenen Klartext $x \in \{0,1\}^{\ell m}$ wird aus einem zufälligen Initialwert r ein langer Bitstring

$$k' = E_{\mathcal{B}}(r,k)\, E_{\mathcal{B}}((r+1) \bmod 2^\ell, k)\ \dots\ E_{\mathcal{B}}((r+m-1) \bmod 2^\ell, k)$$

  berechnet und der Klartext x dann mittels Vernamsystem und diesem Schlüssel verschlüsselt. Der Empfänger erhält r und den Chiffretext, kann also ebenfalls k' berechnen und damit entschlüsseln. Ist $\mathcal{B}$ ein sicheres Block-Kryptosystem, so kann ein Angreifer aus r den Vernam-Schlüssel k' nicht so einfach berechnen, da er k nicht kennt. Die R-CTR-Betriebsart liefert also intuitiv einen hohen Grad an Sicherheit.

3.2 Sicherheit von symmetrischen Kryptoschemen

Wir werden hier ein Sicherheitsmodell definieren, das es gestattet, Aussagen wie die folgende zu formulieren (und zu beweisen): Wenn $\mathcal{B}$ ein „sicheres“ ℓ-Block-Kryptosystem ist (bzgl. einer Reihe von Parametern), und das Kryptoschema $\mathcal{S}$ wird aus $\mathcal{B}$ konstruiert, indem man einen geeigneten Betriebsmodus verwendet, dann ist $\mathcal{S}$ ebenfalls „sicher“ (bzgl.

einer variierten Reihe von Parametern). Ziel ist dabei, Betriebsmodi zu identifizieren, die keine unnötigen neuen Unsicherheitskomponenten ins Spiel bringen, die nicht im Block-KS $\mathcal{B}$ schon vorhanden waren.

Wir beschränken uns hier auf den Fall, wo Eva begrenzte Ressourcen (Zeit, Orakelaufrufe) hat. Damit müssen wir uns bei Überlegungen zu Kryptoschemen auf das Verhalten auf Klartexten begrenzter Länge und auf feste Rechenzeiten beschränken. Nach dem Kerckhoffs-Prinzip nehmen wir an, dass Eva das Kryptoschema kennt, also zum Beispiel das verwendete Block-Kryptosystem und den Betriebsmodus. Sie kann sich einige Klartexte verschlüsseln lassen („known-plaintext attack“) und sieht einen Chiffretext y. Ihr Ziel ist, aus y Information über den zugrunde liegenden Klartext x zu gewinnen. Wir können nicht verhindern, dass Eva aus der Länge des Chiffretextes y auf die Länge von x schließt. (Bei allen Betriebsmodi, die wir gesehen haben, ergibt sich die Länge von x direkt aus der Länge von y.) Abgesehen hiervon soll sie mit hoher Wahrscheinlichkeit „keine signifikante Information“ über den Klartext erhalten können.

Wir skizzieren zunächst eine Idee für ein Sicherheitsmodell, das die Fähigkeit, in so einer Situation „Information zu ermitteln“, formalisiert. Eva behauptet, sie könne „aus y Information über x ermitteln, die über die Länge von x hinausgeht“. Um das zu überprüfen, stellt ihr ein „Herausforderer“ Charlie folgende Aufgabe: Eva darf sich zunächst eine Reihe von Klartexten ihrer Wahl verschlüsseln lassen. Dann wählt sie selbst zwei verschiedene, *gleich lange* Klartexte z_0 und z_1. Charlie verschlüsselt einen von diesen; der Chiffretext ist y. Eva bekommt y. Sie soll herausfinden, ob y von z_0 oder von z_1 kommt. Für diese Entscheidung darf sie sich weitere Klartexte verschlüsseln lassen und weiter rechnen. Wir betrachten ein Kryptoschema als unsicher, wenn Eva eine signifikant von „purem Raten“ abweichende Erfolgswahrscheinlichkeit hat, sie also mit guten Chancen unterscheiden kann, ob z_0 oder z_1 zu y verschlüsselt wurde.

Wie in Abschnitt 2.4 formulieren wir den Vorgang wieder als *Spiel*. Gegeben ist also $\mathcal{S} = (K, E, D)$.[30] Akteure sind Eva, hier „Angreifer“ (engl.: *adversary*) genannt, und Charlie („Herausforderer“, engl.: *challenger*). Die Parameter, mit denen wir die Erfolgschancen von Eva unter Ressourcenbeschränkungen messen, sind der „Vorteil“ in Analogie zu Definition 2.13, die Rechenzeit (inklusive Speicherplatz, wie vorher), Anzahl der Orakelaufrufe und Anzahl der bei der Verschlüsselung bearbeiteten Blöcke.

- Charlie wählt zufällig einen Schlüssel k aus K und legt damit die Chiffre $H = E(\,.\,, k)$ fest, die Klartexte aus $\{0,1\}^{\ell \cdot *}$ in Chiffretexte aus $\{0,1\}^{\ell \cdot *}$ transformiert.
- Eva wählt einige Klartexte und lässt sie sich von Charlie mit H verschlüsseln.
- Eva wählt zwei Klartexte z_0 und z_1 gleicher Länge und gibt sie an Charlie.

[30] Wie das Kryptoschema aufgebaut ist, also zum Beispiel ob es durch Anwendung einer Betriebsart auf ein Block-Kryptosystem entsteht, spielt hier keine Rolle.

- Verdeckt vor Eva: Charlie wirft eine Münze, um zufällig einen der beiden Klartexte zu wählen. Er verschlüsselt ihn mit H, das Ergebnis ist y. Charlie gibt y an Eva.
- Eva kann sich weitere Klartexte verschlüsseln lassen und (mit beschränkten Ressourcen) rechnen und muss schließlich sagen (raten?), ob Charlie z_0 oder z_1 zu y verschlüsselt hat.

Wenn die Wahrscheinlichkeit, dass Eva richtig antwortet, weit von zufälligem Raten abweicht, wollen wir das Kryptoschema als unsicher ansehen.

Man beachte: Unter den vorher oder nachher verschlüsselten Klartexten können auch z_0 und z_1 sein. Ein *deterministisches* Kryptoschema, also eines, das zu gegebenem Schlüssel k einen Klartext stets gleich verschlüsselt, ist damit sofort disqualifiziert. Wenn aber bei der Verschlüsselung Randomisierung im Spiel ist, liefern wiederholte Verschlüsselungsanforderungen mit Klartexten z_0 und z_1 (wahrscheinlich) lauter unterschiedliche Antworten, so dass dieser direkte Weg zur Ermittlung des verschlüsselten Klartextes nicht funktioniert.

Wir beschreiben den Part von Eva in diesem Spiel als Algorithmenpaar. Der erste Algorithmus AF (der „Finder“, „*find*“) ist für das Erzeugen von z_0 und z_1 zuständig. Als Argument erhält dieser Teil eine Chiffre H, die er im Sinne eines Orakels benutzen kann. Außerdem werden Aufzeichnungen über die angeforderten Verschlüsselungen und ihre Ergebnisse gemacht. Diese Aufzeichnungen sind als Element v einer endlichen Menge V kodiert. Der zweite Algorithmus AG (der „Rater“, „*guess*“) ist dafür zuständig, herauszufinden, ob z_0 oder z_1 verschlüsselt wurde. Dieser Algorithmus bekommt H als Orakel, die Aufzeichnungen v von AF und den Chiffretext y als Input.

Definition 3.2 *Ein ℓ-Angreifer A ist ein Paar von randomisierten Algorithmen*

$$\begin{aligned} &AF(H(\{0,1\}^{\ell\cdot *})\colon \{0,1\}^{\ell\cdot *})\colon (\{0,1\}^{\ell}\times\{0,1\}^{\ell})^{+}\times V\\ AG(v\colon V, H(\{0,1\}^{\ell\cdot *})\colon \{0,1\}^{\ell\cdot *}, &y\colon \{0,1\}^{\ell\cdot *})\colon \{0,1\}\end{aligned}$$

Hierbei ist H ein randomisierter Algorithmus (und nicht als Funktion zu verstehen).

Der „*Finder*“ AF bekommt also eine Chiffre $H = E(\,.\,,k)$ für einen zufälligen Schlüssel k gegeben. (Er darf diese Chiffre nur zum Verschlüsseln benutzen; k kennt er nicht.) Daraus berechnet er zwei verschiedene Klartexte (z_0, z_1) gleicher Länge und „Notizen“ $v \in V$. Das Ausgabeformat $(\{0,1\}^{\ell}\times\{0,1\}^{\ell})^{+}$ sagt, dass eine Folge

$$((z_0^{(0)}, z_1^{(0)}), (z_0^{(1)}, z_1^{(1)}), \ldots, (z_0^{(m-1)}, z_1^{(m-1)}))$$

von Blockpaaren ausgegeben werden soll, die wir dann als Paar

$$(z_0, z_1) = ((z_0^{(0)}, z_0^{(1)}, \ldots, z_0^{(m-1)}), (z_1^{(0)}, z_1^{(1)}, \ldots, z_1^{(m-1)}))$$

von zwei Folgen gleicher Länge lesen. Danach wird zufällig z_0 oder z_1 zu y verschlüsselt. Im zweiten Schritt verwendet der „*Rater*“ AG die Notizen v, die Chiffre $H = E(\,.\,, k)$ und die „Probe“ y, um zu bestimmen, ob z_0 oder z_1 verschlüsselt wurde.

Definition 3.3 *Sei $\mathcal{S} = (K, E, D)$ ein symmetrisches Kryptoschema, und sei $A = (AF, AG)$ ein ℓ-Angreifer. Das zugehörige* Experiment *(oder* Spiel*) ist der folgende Algorithmus $G_A^{\mathcal{S}}\colon \{0,1\}^*$:*

1. $k \leftarrow \mathrm{flip}(K)$
   $H \leftarrow E(\,.\,, k)$

   (In diesem Schritt wählt Charlie zufällig eine Chiffre des Kryptoschemas $\mathcal{S}$.)

2. $(z_0, z_1, v) \leftarrow AF(H)$

   (In dieser Phase berechnet der Finder ein Paar von Klartexten gleicher Länge, von denen er annimmt, ihre Chiffretexte unterscheiden zu können.)

3. $b \leftarrow \mathrm{flip}(\{0,1\})$ *und* $y \leftarrow E(z_b, k)$

   (In dieser Phase wählt Charlie zufällig einen der beiden Klartexte und verschlüsselt ihn zu y.)

4. $b' \leftarrow AG(v, H, y)$

   (In dieser Phase versucht der Rater herauszubekommen, ob z_0 oder z_1 verschlüsselt wurde.)

5. *falls $b = b'$, so gib* 1 *zurück, sonst* 0.

   (Charlies Auswertung: Hat AG recht oder nicht?)

Das verkürzte Experiment *oder* Spiel $S_A^{\mathcal{S}}$ *gibt im 5. Schritt einfach b' aus.*

Dann ist $\Pr(G_A^{\mathcal{S}} = 1)$ die Wahrscheinlichkeit dafür, dass der Angreifer A sich für den korrekten Klartext entscheidet. Der Wahrscheinlichkeitsraum entsteht durch die expliziten Zufallsexperimente in Schritt 1. und 3. in Kombination mit den Zufallsexperimenten, die bei der Verwendung von H ausgeführt werden. Man kann jetzt wie in Abschnitt 2.4 (Sicherheit von ℓ-Block-Kryptosystemen) den Vorteil

$$\mathrm{adv}(A, \mathcal{S}) = 2\Big(\Pr(G_A^{\mathcal{S}} = 1) - \frac{1}{2}\Big)$$

und die Größen

$\mathrm{suc}(A, \mathcal{S}) = \Pr(S_A^{\mathcal{S}}\langle b = 1\rangle = 1)$ („Erfolg“) und $\mathrm{fail}(A, \mathcal{S}) = \Pr(S_A^{\mathcal{S}}\langle b = 0\rangle = 1)$ („Misserfolg“)

definieren. Allerdings haben die beiden letzten Werte eine etwas andere Semantik. Der Wert $\mathrm{suc}(A, \mathcal{S})$ ist die bedingte Wahrscheinlichkeit, dass *richtig* erkannt wird, dass z_1 verschlüsselt wurde, $\mathrm{fail}(A, \mathcal{S})$ ist die bedingte Wahrscheinlichkeit, dass *nicht* erkannt wird, dass z_0 verschlüsselt wurde. Lemma 2.14 gilt wörtlich. Das heißt:

$$\mathrm{adv}(A, \mathcal{S}) = \mathrm{suc}(A, \mathcal{S}) - \mathrm{fail}(A, \mathcal{S}).$$

Wenn ein Angreifer A mit „nicht zu großem Rechenaufwand“ einen Vorteil erzielen kann, der deutlich über 0 liegt, wird man das Kryptoschema als unsicher einstufen.

Beispiel 3.4 Ziel ist es, die ECB-Betriebsart anzugreifen, d. h. sei $\mathcal{S} = \mathrm{ECB}(\mathcal{B})$ für ein ℓ-Block-Kryptosystem $\mathcal{B}$. Wir wollen zeigen, dass es einen ℓ-Angreifer A mit $\Pr(G_A^{\mathcal{S}} = 1) = 1$, also $\mathrm{adv}(A, \mathcal{S}) = 1$, gibt. Dieser ist wie folgt aufgebaut.

ℓ-Angreifer A mit $V = \{1\}$:[31]

$AF(H)$ arbeitet wie folgt: $z_0 \leftarrow 0^{2\ell}$; $z_1 \leftarrow 0^\ell 1^\ell$; Ausgabe: $(z_0, z_1, 1)$

$AG(v, H, y)$ tut Folgendes:
falls $y = y_1 y_1$ für ein $y_1 \in \{0,1\}^\ell$, gib 0 aus, sonst 1.

Im Ablauf des Spiels $G_A^{\mathcal{S}}$ wird der Rater AG mit $y = E(0^{2\ell}, k) = e_{\mathcal{B}}(0^\ell, k)\, e_{\mathcal{B}}(0^\ell, k)$ oder mit $y = E(0^\ell 1^\ell, k) = e_{\mathcal{B}}(0^\ell, k)\, e_{\mathcal{B}}(1^\ell, k)$ gestartet. Im ersten Fall ist $y = y_1 y_1$ für ein $y_1 \in \{0,1\}^\ell$, im zweiten Fall ist dies nicht so, wegen der Dechiffrierbedingung. Daher gilt $\Pr(G_A^{\mathcal{S}} = 1) = 1$, d. h. $\mathrm{adv}(A, \mathcal{S}) = 1$.

Die Ressourcen, die A benötigt, sind sehr klein: Zwei Aufrufe des Verschlüsselungsverfahrens $e_{\mathcal{B}}$ des Block-Kryptosystems. Wir können schließen, dass es einen effizienten Angreifer A gibt, dem das Sicherheitsmodell Vorteil 1 gibt. Damit gilt das Kryptoschema $\mathrm{ECB}(\mathcal{B})$ als komplett unsicher (ganz egal was $\mathcal{B}$ ist).

[31] $V = \{1\}$ bedeutet, dass stets $v = 1$ gilt, dass die Aufzeichnung v also keinerlei Information übermittelt.

Beispiel 3.5 Ziel ist es, die CBC-Betriebsart anzugreifen, d. h. es sei $\mathcal{S} = \mathrm{CBC}(\mathcal{B})$ für ein Block-Kryptosystem $\mathcal{B}$. Das Problem mit dieser Betriebsart ist, dass ein Klartext bei Wiederholung identisch verschlüsselt wird. Um dies auszunutzen, verwenden wir den folgenden ℓ-Angreifer A mit $V = \{0,1\}^\ell$, der zwei verschiedene Klartexte benutzt, die nur einen Block enthalten:

$AF(H)$ arbeitet wie folgt:
$z_0 \leftarrow 0^\ell$; $v \leftarrow H(z_0)$; $z_1 \leftarrow 1^\ell$; Ausgabe: (z_0, z_1, v)

(A merkt sich den Chiffretext zu $x = 0^\ell$.)

$AG(v, H, y)$ tut Folgendes:
falls $v = y$, so gib 0 aus, sonst 1.

Im Ablauf des Spiels $G_A^{\mathcal{S}}$ wird der Rater AG mit $E(0^\ell, k)$ oder mit $E(1^\ell, k)$ gestartet. Wegen $e_{\mathcal{B}}(0^\ell, k) \neq e_{\mathcal{B}}(1^\ell, k)$ (wegen der Dechiffrierbedingung) gilt also $\Pr(G_A^{\mathcal{S}} = 1) = 1$, d. h. $\mathrm{adv}(A, \mathcal{S}) = 1$.

Dieses Beispiel lässt sich verallgemeinern:

Lemma 3.6 *Es gibt einen ℓ-Angreifer A, so dass für jedes ℓ-Kryptoschema $\mathcal{S}$ mit deterministischer Verschlüsselungsfunktion gilt:* $\mathrm{adv}(A, \mathcal{S}) = 1$.

Wir benutzen einfach den in Beispiel 3.5 beschriebenen Angreifer. Damit zeigt sich, dass das beschriebene Spiel in der Lage ist, alle deterministischen Kryptoschemen als unsicher einzustufen (und damit die intuitive Einschätzung zu bestätigen).

Nun bringen wir die Ressourcen ins Spiel, die der Angreifer benutzen darf. Komponenten dabei sind die Laufzeit des gesamten Experiments, die Anzahl der durchgeführten H-Verschlüsselungen von Chiffretexten und die Anzahl der dabei bearbeiteten Blöcke.[32]

Definition 3.7 *Sei $n, q, t, \ell \in \mathbb{N}$, A ein ℓ-Angreifer, $\mathcal{S}$ ein symmetrisches ℓ-Kryptoschema. Dann heißt A* (n, q, t)-beschränkt*, wenn die Laufzeit des Experiments $G_A^{\mathcal{S}}$ durch t beschränkt ist, der Algorithmus H (als Orakel) höchstens q mal aufgerufen wird und bei diesen Aufrufen höchstens n Blöcke verwendet werden.*

Sei $\varepsilon > 0$. Dann heißt $\mathcal{S}$ (n, q, t, ε)-sicher*, wenn für jeden (n, q, t)-beschränkten ℓ-Angreifer A gilt*

$$\mathrm{adv}(A, \mathcal{S}) \leq \varepsilon\,.$$

[32] $\mathcal{S}$ kann eine beliebige Struktur haben, muss also nicht notwendigerweise auf einem ℓ-Block-Kryptosystem beruhen. Dennoch sind die Klartexte in Blöcke eingeteilt, und die Gesamtlänge aller betrachteten Blöcke ist eine sinnvolle Maßzahl.

Nach obigem Lemma gibt es (kleine) Konstanten c_1, c_2 und c_3, so dass kein deterministisches ℓ-Kryptoschema $(c_1, c_2, c_3, 1-\delta)$-sicher ist, für jedes noch so kleine $\delta > 0$.

Wir stellen nun fest, dass man aus sicheren Block-Kryptosystemen mit der R-CTR-Betriebsart sichere Kryptoschemen erhält – wenn man die Parameter richtig wählt, das heißt im Wesentlichen, wenn die Blocklänge genügend groß ist.

Diese „relative Sicherheit“ kann man folgendermaßen „rückwärts“ ausdrücken: Wenn das Kryptoschema $\mathcal{S} = \text{R-CTR}(\mathcal{B})$ unsicher ist, es also einen Angreifer mit großem Vorteil $\text{adv}(A, \mathcal{S})$ gibt, bei beschränkten Ressourcen, dann ist schon $\mathcal{B}$ unsicher, das heißt, es gibt einen Unterscheider U für $\mathcal{B}$ mit großem Vorteil $\text{adv}(U, \mathcal{B})$, bei beschränkten Ressourcen. Technisch wird dies folgendermaßen formuliert, unter Einbeziehung gewisser Fehlerterme und genauer Benennung der Ressourcenschranken.

Satz 3.8 *Es gibt eine kleine Konstante c, so dass für alle $t, n, q, \ell > 0$, alle ℓ-Block-Kryptosysteme $\mathcal{B}$ und alle (n, q, t)-beschränkten ℓ-Angreifer A ein $(n, t + c(q\log(q) + n)\cdot\ell)$-beschränkter ℓ-Unterscheider U existiert, so dass*

$$\text{adv}(A, \mathcal{S}) \leq 2 \cdot \text{adv}(U, \mathcal{B}) + \frac{2qn + n^2}{2^\ell},$$

wobei $\mathcal{S}$ das symmetrische ℓ-Kryptoschema ist, das $\mathcal{B}$ in der R-CTR-Betriebsart verwendet.

Den Fehlerterm $(2qn + n^2)/2^\ell$ kann man vernachlässigen, wenn ℓ genügend groß gewählt wird. Die Zahlen q und n entsprechen der Verarbeitung von Blöcken, Werte für q und n von mehr als 10^{12} sind also eher unrealistisch. Mit $\ell = 128$ ist $2^\ell > 3 \cdot 10^{38}$. Damit kann man ohne Weiteres $\frac{2qn+n^2}{2^\ell} < 10^{-14}$ annehmen. Im Wesentlichen besagt der Satz also, dass aus der Existenz eines effizienten ℓ-Angreifers mit einem gewissen Vorteil $a > 0$ gegen $\text{R-CTR}(\mathcal{B})$ folgt, dass es einen effizienten ℓ-Unterscheider U mit Vorteil $a/2$ gegen $\mathcal{B}$ gibt. Wenn also $\text{R-CTR}(\mathcal{B})$ unsicher ist (nicht ε-sicher für relativ großes ε), dann muss schon $\mathcal{B}$ unsicher gewesen sein (nicht $\varepsilon/2$-sicher). Oder noch kürzer: Wenn $\mathcal{B}$ „sicher“ ist, dann auch $\text{R-CTR}(\mathcal{B})$. Durch die R-CTR-Betriebsart wird keine neue Unsicherheitskomponente ins Spiel gebracht.

Mit den folgenden Definitionen lässt sich diese Aussage vielleicht noch griffiger formulieren.

Definition 3.9 *Die* Unsicherheit *eines Block-Kryptosystems $\mathcal{B} = (X, K, Y, e, d)$ zu Parametern q, t ist*

$$\text{insec}(q, t, \mathcal{B}) := \max\{\text{adv}(U, \mathcal{B}) \mid U \textit{ ist } (q, t)\textit{-beschränkter Unterscheider}\}.$$

Man beachte: Weil mit t auch die Programmtextlänge beschränkt ist, gibt es nur endlich viele solche Unterscheider. Damit ist das Maximum wohldefiniert. Offensichtlich gilt, nach den Definitionen aus Abschnitt 2.4: $\mathcal{B}$ ist (q, t, ε)-sicher $\Leftrightarrow \text{insec}(q, t, \mathcal{B}) \leq \varepsilon$.

Analog definiert man:

Definition 3.10 *Die* Unsicherheit *eines Kryptoschemas* $\mathcal{S} = (K, E, D)$ *zu Parametern* n, q, t *ist*

$$\mathrm{insec}(n, q, t, \mathcal{S}) := \max\{\mathrm{adv}(A, \mathcal{S}) \mid A \textit{ ist } (n, q, t)\textit{-beschränkter Angreifer}\}.$$

Satz 3.8 liest sich dann wie folgt: Wenn $\mathcal{S}$ das symmetrische ℓ-Kryptoschema ist, das $\mathcal{B}$ in der R-CTR-Betriebsart verwendet, dann gilt für beliebige n, t, q:

$$\mathrm{insec}(n, q, t, \mathcal{S}) \leq 2 \cdot \mathrm{insec}(n, t + c(q \log(q) + n) \cdot \ell, \mathcal{B}) + \frac{2qn + n^2}{2^\ell}.$$

$\mathcal{S}$ „erbt“ also die obere Schranke für die Unsicherheit von $\mathcal{B}$, bezüglich n Orakelanfragen und einer vergrößerten Rechenzeit von $t + c(q \log(q) + n) \cdot \ell$, verschlechtert nur um einen Faktor 2 und einen additiven Term $\frac{2qn+n^2}{2^\ell}$. Die Unsicherheit kommt also nicht durch die Verwendung der Betriebsart R-CTR ins Spiel, sondern steckt gegebenenfalls schon in $\mathcal{B}$.

Bemerkung: Für die Betriebsarten R-CBC und OFB gelten Aussagen, die zu Satz 3.8 analog sind. Die Beweise sind allerdings noch aufwendiger. Einen vollständigen Beweis von Satz 3.8 findet man in „Küsters und Wilke, Moderne Kryptographie“ (S. 114–121), und im Anhang.

Im Buch werden auch die folgenden konkreten Parameter diskutiert: $\ell = 128$. Nehmen wir an, die zugelassene Laufzeit t für den Angreifer ist $2^{60} > 10^{18}$ Rechenschritte (das ist so groß, dass es nicht wirklich realisierbar ist), und wir gestatten $q = 2^{30} \approx 10^9$ Orakelanfragen, wobei die gesamte betroffene Textlänge $n = 2^{36} \approx 64 \cdot 10^9$ Blöcke ist (etwa 2^{40} Byte, also ein Terabyte). Wenn die Konstante aus dem Satz etwa $c = 10$ ist, erhalten wir:

$$\mathrm{insec}(2^{36}, 2^{30}, 2^{60}, \mathcal{S}) \leq 2 \cdot \mathrm{insec}(2^{36}, 2^{60} + 10(30 \cdot 2^{30} + 2^{36}) \cdot 128, \mathcal{B}) + \frac{2^{66} + 2^{72}}{2^{128}}.$$

Man sieht, dass der additive Term $\frac{2^{66}+2^{72}}{2^{128}}$ kleiner als 2^{-55} ist, und die für den Unterscheider zugelassene Zeitschranke mit $2^{60} + 10(30 \cdot 2^{30} + 2^{36}) \cdot 128 < 2^{61}$ kaum größer ist als die für den Angreifer. Wenn man für $\mathrm{insec}(2^{36}, 2^{61}, \mathcal{B})$ eine Schranke $\leq 2^{-55}$ hätte, wäre $\mathrm{insec}(2^{36}, 2^{30}, 2^{60}, \mathcal{S})$ auch kleiner als 2^{-54}. (Solche konkreten Schranken sind allerdings für kein konkretes Block-Kryptosystem bewiesen.)

A Anhang: Beweis von Satz 3.8 (Stand: 26.11.2018)

Wir beweisen Satz 3.8. Gegeben ist also ein ℓ-Block-KS $\mathcal{B}$, aus dem wir mit dem R-CTR-Betriebsmodus ein Kryptoschema $\mathcal{S} = \text{R-CTR}(\mathcal{B})$ gewinnen. Gegeben ist weiter ein (n, q, t)-beschränkter ℓ-Angreifer $A = (AF, AG)$. Wir wollen zeigen, dass man aus A einen Unterscheider für $\mathcal{B}$ gewinnen kann, der im Wesentlichen die Hälfte des Vorteils von A gegen $\mathcal{S} = \text{R-CTR}(\mathcal{B})$ hat.

Man erinnere sich an Definition 2.11. Das Spiel (Experiment) $G_U^{\mathcal{B}}$ wählt in Form eines Zufallsbits b die „Welt“, in der gespielt wird, und stellt dem Unterscheider U eine entsprechende Blockchiffre F zur Verfügung.

- Wenn $b = 1$, ist F eine Chiffre $e(\,.\,, k)$ zu $\mathcal{B}$ („Realwelt“), mit zufälligem Schlüssel k.
- Wenn $b = 0$, ist F eine zufällige Permutation von $\{0,1\}^\ell$ („Idealwelt“).

U soll eine Vermutung abgeben, welcher der beiden Fälle zutrifft. Die Idee der nun folgenden Konstruktion ist, dass U im Wesentlichen das Spiel $G_A^{\mathcal{S}}$ für A ausführt und dabei den Verschlüsselungsalgorithmus H_F benutzt, der sich aus der Chiffre F mit der Betriebsart R-CTR ergibt. Dazu startet U den Findungsteil AF von A, der ein Paar (z_0, z_1) von Klartexten erzeugt. Dann wählt U zufällig (mit einem Zufallsbit c aus $\{0,1\}$) einen der beiden aus: z_c. Der Chiffretext $y = H_F(z_c)$ wird AG vorgelegt, der ein Antwortbit c' liefert. (Dieses Bit sollte, wenn der Angreifer gut ist und F von $\mathcal{B}$ her stammt, eher gleich c' als verschieden von c' sein.) Wenn $c = c'$, vermutet U „Realwelt“, gibt also $b' = 1$ zurück, sonst vermutet U „Idealwelt“ und antwortet mit $b' = 0$. U gewinnt das Spiel, wenn das Ausgabebit b' mit b übereinstimmt.

Im Fall $b = 0$ liegen die Erfolgschancen für U sehr nahe an $\frac{1}{2}$, wie wir sehen werden, da H_F sich selbst unter Berücksichtigung der Informationen, die AF und AG sammeln, fast wie eine Vernam-Chiffre verhält, und daher der Chiffretext zu z_0 ebenso wie der zu z_1 einfach wie ein Zufallsstring aussieht, so dass U keine Möglichkeit hat, die beiden zu unterscheiden. Im Fall $b = 1$ hingegen ist der Ablauf von $G_U^{\mathcal{B}}$ genau der des Spiels $G_A^{\mathcal{S}}$ von A mit $\mathcal{S} = \text{R-CTR}(\mathcal{B})$, und die Erfolgswahrscheinlichkeit von U entspricht dem dieses Spiels. Kombination der beiden Fälle ergibt, dass der Vorteil von U bis auf Fehlerterme die Hälfte des Vorteils von A ist. Details folgen.

Wir beschreiben zunächst die Arbeitsweise von U im Detail. U bekommt eine Blockchiffre $F\colon \{0,1\}^\ell \to \{0,1\}^\ell$ als Verschlüsselungsorakel zur Verfügung gestellt.

U benutzt den Angreifer $A = (AF, AG)$ als Unterprogramm. Von den Programmteilen AF und AG von A werden Orakelanfragen gestellt, die fordern, dass Texte $x \in \{0,1\}^{\ell \cdot *}$ verschlüsselt werden. Diese Anfragen beantwortet U, indem es die Chiffre F in R-CTR-

Betriebsart, also den Verschlüsselungsalgorithmus H_F einsetzt. Das entsprechende Unterprogramm von U heißt SIM:

SIM($x\colon \{0,1\}^{\ell\cdot *}, F\colon \{0,1\}^\ell \to \{0,1\}^\ell$)$\colon \{0,1\}^{\ell\cdot *}$

// Input x: Klartext, F: Blockchiffre (als Orakel); Output y: Chiffretext.

1. $r \leftarrow \text{flip}(\ell)$ // Zufallszahl in $[2^\ell] = \{0,1\}^\ell$

2. $t \leftarrow |x|/\ell$; // Anzahl der Blöcke in x, also $x = (x_0, \ldots, x_{t-1})$

3. $y \leftarrow (r, F(r) \oplus_\ell x_0, F((r+1) \bmod 2^\ell) \oplus_\ell x_1, \ldots, F((r+t-1) \bmod 2^\ell) \oplus_\ell x_{t-1})$

4. return y

Wenn wir im Folgenden $AF(\text{SIM}(\,.\,, F))$ schreiben, meinen wir, dass der Findungsteil von A aufgerufen wird, wobei Orakelanfragen von AF für x mit $\text{SIM}(x, F)$ beantwortet werden. Analog ist $AG(\text{SIM}(\,.\,, F))$ zu interpretieren.

U arbeitet mit gegebenem Verschlüsselungsorakel $F\colon \{0,1\}^\ell \to \{0,1\}^\ell$ wie folgt. Die Idee ist, dass U das Spiel $G_A^{\cdot\cdot}$ durchführt, wobei als Verschlüsselungsverfahren für das Kryptoschema die R-CTR-Betriebsart von F benutzt wird. Je nachdem ob A dabei das richtige oder das falsche Ergebnis erhält, tippt U auf „Realwelt“ bzw. „Idealwelt“.

1. *Findungsphase*
   $(z_0, z_1, v) \leftarrow AF(\text{SIM}(\,.\,, F))$.
   // Findungsteil von A; Orakelanfragen werden mit $\text{SIM}(\,.\,, F)$ beantwortet.
   // v sind die Notizen von A.

2. *Auswahl* // Wähle zufällig z_0 oder z_1, verschlüssle, Ergebnis ist y
   $c \leftarrow \text{flip}()$, $y \leftarrow \text{SIM}(z_c, F)$.

3. *Ratephase*
   $c' \leftarrow AG(v, \text{SIM}(\,.\,, F), y)$.
   // Rateteil von A; Orakelanfragen werden mit $\text{SIM}(\,.\,, F)$ beantwortet.

4. *Auswertung*
   return $[c = c']$

Nach den Regeln für Szenario 2 darf die Blockverschlüsselung F nicht mehrfach für denselben Block aufgerufen werden. Bei dem angegebenen Verfahren kann dies aber (aufgrund

der zufälligen Wahl des Startwerts r bei den Aufrufen von SIM) durchaus vorkommen. Wir zählen eine solche Abfrage *nicht* als neuen Aufruf. Ohne dass dies im Programmtext ausgeführt ist, sollte man sich vorstellen, dass U bzw. das Spiel $G_U^{\mathcal{B}}$ eine Datenstruktur für alle bisher benannten Blöcke s und die schon abgefragten Werte $F(s)$ mitführt und bei der Ausführung von SIM immer nachsieht, ob $F(s)$ für einen Block s schon bekannt ist. Der Aufwand hierfür wird in der Rechenzeit von U berücksichtigt. Wir diskutieren dies ganz am Ende des Beweises.

Wir erinnern uns an die Definition von $\mathrm{suc}(U,\mathcal{B})$ (Wahrscheinlichkeit für Antwort 1 in der Realwelt) und $\mathrm{fail}(U)$ (Wahrscheinlichkeit für Antwort 1 in der Idealwelt) und Lemma 2.14 mit der Formel

$$\mathrm{adv}(U,\mathcal{B}) = \mathrm{suc}(U,\mathcal{B}) - \mathrm{fail}(U). \tag{1.5}$$

Ziel ist, eine untere Schranke für $\mathrm{adv}(U,\mathcal{B})$ zu beweisen. Wir analysieren die beiden Terme separat. Dazu geben wir $\mathrm{suc}(U,\mathcal{B})$ an und beweisen eine obere Schranke für $\mathrm{fail}(U)$. Das erste ist einfach, das zweite technisch etwas aufwendiger.

„Realwelt“: Hier ist $b = 1$ und $F\colon \{0,1\}^\ell \to \{0,1\}^\ell$ eine durch den zufällig gewählten Schlüssel k bestimmte Chiffre $e(\,.\,,k)$ von $\mathcal{B}$. Wir analysieren $\mathrm{suc}(U,\mathcal{B}) = \Pr(S_U^{\mathcal{B}}\langle b=1\rangle = 1)$. Man sieht sofort, dass U genau das in Definition 3.3 beschriebene Spiel für A und $\mathcal{S}$ ausführt, und dass daher die Wahrscheinlichkeit für das Ergebnis „1“ genau $\Pr(G_A^{\mathcal{S}} = 1)$ ist. Wegen $\mathrm{adv}(A,\mathcal{S}) = 2\Pr(G_A^{\mathcal{S}} = 1) - 1$ gilt

$$\mathrm{suc}(U,\mathcal{B}) = \Pr(G_A^{\mathcal{S}} = 1) = \frac{1 + \mathrm{adv}(A,\mathcal{S})}{2}. \tag{1.6}$$

„Idealwelt“: Nun wenden wir uns der Analyse von $\mathrm{fail}(U) = \Pr(S_U^{\mathcal{B}}\langle b=0\rangle = 1)$ zu. Dies ist die Wahrscheinlichkeit, dass U in der Situation „Idealwelt“, in der F eine Zufallspermutation ist, das richtige Bit $c' = c$ ausgibt. Wir werden sehen, dass diese Wahrscheinlichkeit nur unwesentlich von $\frac{1}{2}$ abweichen kann, wenn ℓ genügend groß ist. (Dann ergibt sich mit (1.6), dass $\mathrm{adv}(U,\mathcal{B}) \approx \frac{1}{2}\mathrm{adv}(A,\mathcal{S})$ gilt.) Wir müssen eine obere Schranke für $\mathrm{fail}(U)$ angeben. Wir bemerken zunächst, dass das Superskript $\mathcal{B}$ in $S_U^{\mathcal{B}}\langle b=0\rangle$ redundant ist, da sich dieser Algorithmus auf die Idealwelt bezieht, in der $\mathcal{B}$ keine Rolle spielt. Wir schreiben daher im Folgenden dafür $S_U\langle b=0\rangle$.

Wir wissen, dass der Angreifer A insgesamt nicht mehr als q Orakelanfragen stellt und dass dabei die Anzahl der betroffenen ℓ-Bit-Blöcke nicht größer als n ist. Wir können sogar o. B. d. A. annehmen, dass die Anzahl der verschiedenen beim Ablauf von U insgesamt verschlüsselten Blöcke exakt n ist – falls nicht, lassen wir U einfach, außerhalb von SIM, die Verschlüsselung der fehlenden Anzahl von neuen Blöcken anfordern, ohne dass das Ergebnis eine Auswirkung auf die Ausgabe von U hat.

Ersetzung von Zufallspermutationen durch Zufallsfunktionen Um die weitere Arbeit zu vereinfachen, betrachten wir einen leicht veränderten Wahrscheinlichkeitsraum für F. Wir nehmen nämlich an, dass als Orakel für die Blockchiffre nicht eine Zufallspermutation F, sondern eine rein zufällige Funktion F' von $\{0,1\}^\ell$ nach $\{0,1\}^\ell$, verwendet wird.[33] Hier ist es also erlaubt, dass $F'(s_1) = F'(s_2)$ für Blöcke $s_1 \neq s_2$ ist. Der neue Wahrscheinlichkeitsraum heißt Ω', die Wahrscheinlichkeitsfunktion darin $\Pr'$. Es stellt sich heraus, dass es keinen großen Unterschied macht, welchen der beiden Räume man benutzt, solange n nicht zu groß und die Blocklänge ℓ nicht zu klein ist.

Behauptung 1: $|\Pr(S_U\langle b=0\rangle = 1) - \Pr'(S_U\langle b=0\rangle = 1)| \leq \dfrac{n^2}{2^{\ell+1}}$.

Bemerkung: Diese Behauptung liefert offenbar einen Teil des Fehlerterms in Satz 3.8.

Beweis von Behauptung 1: Definiere Ereignisse (für beide Wahrscheinlichkeitsräume in gleicher Weise), unter Bezugnahme auf den Ablauf des Spiels für U mit $b = 0$. In diesem Ablauf gibt es genau n Anfragen zu verschiedenen Blöcken an das Orakel F. (Der Ablauf selber, und um welche Blöcke es sich handelt, hängt auch vom Zufall ab.) Betrachte die folgenden Ereignisse:

$$D_i := \{\text{die } i\text{-te Anfrage an } F \text{ liefert ein Resultat in } \{0,1\}^\ell, \text{ das vorher schon vorkam}\},$$

für $i = 1, \ldots, n$, und $D := \bigcup_{1\leq i\leq n} D_i$. Dann ist D die *disjunkte* Vereinigung der Ereignisse $D_i \cap \overline{D}_1 \cap \overline{D}_2 \cap \cdots \cap \overline{D}_{i-1}$. Wir haben

$$\Pr'(D_i \cap \overline{D}_1 \cap \overline{D}_2 \cap \cdots \cap \overline{D}_{i-1}) \leq \Pr'(D_i \mid \overline{D}_1 \cap \overline{D}_2 \cap \cdots \cap \overline{D}_{i-1}) = \frac{i-1}{2^\ell}, \tag{1.7}$$

(die i-te Anfrage liefert einen der $i-1$ vorher ausgegebenen Werte), also

$$\Pr'(D) \leq \sum_{1\leq i\leq n} \frac{i-1}{2^\ell} = \frac{n(n-1)}{2\cdot 2^\ell} < \frac{n^2}{2^{\ell+1}}. \tag{1.8}$$

Bedingt auf den Teilwahrscheinlichkeitsraum $\overline{D} = \Omega' - D$ ist die Folge der n verwendeten F'-Werte eine rein zufällige Folge in $\{0,1\}^\ell$ mit der Zusatzeigenschaft, dass die Werte alle verschieden sind. Dies ist aber genau dieselbe Situation, die wir erhalten würden, wenn wir im Originalwahrscheinlichkeitsraum mit einer Zufallspermutation F arbeiten würden. Das heißt:

$$\Pr(S_U\langle b=0\rangle = 1) = \Pr'(S_U\langle b=0\rangle = 1 \mid \overline{D}).$$

Wegen (1.7) genügt es für den Beweis der Behauptung, mit der Abkürzung C für das Ereignis $\{S_U\langle b=0\rangle = 1\}$ Folgendes zu zeigen:

$$|\Pr'(C \mid \overline{D}) - \Pr'(C)| \leq \Pr'(D). \tag{1.9}$$

[33] Dies ist keine wirkliche Verschlüsselungsfunktion, sondern nur ein mathematisches Konstrukt.

Wir zeigen, dass (1.9) immer gilt, gleichgültig was C ist. Es gilt

$$\Pr'(C) \geq \Pr'(C \cap \overline{D}) = \Pr'(C \mid \overline{D})\Pr'(\overline{D}) = \Pr'(C \mid \overline{D})(1 - \Pr'(D)) \geq \Pr'(C \mid \overline{D}) - \Pr'(D).$$

Andererseits ist

$$\Pr'(C) = \Pr'(C \mid \overline{D})\Pr'(\overline{D}) + \Pr'(C \mid D)\Pr'(D) \leq \Pr'(C \mid \overline{D}) + \Pr'(D).$$

Damit ist (1.9) und damit die Behauptung gezeigt. Ab hier arbeiten wir also in Ω' mit einer Zufallsfunktion F'.

Wir überlegen nun, wie die Argumente bestimmt werden, für die die Zufallsfunktion F' aufgerufen wird. Von AF, bei der Verschlüsselung von z_c und von AG werden dem Unterprogramm $\textsc{Sim}(\,.\,, F')$ (bis zu) q Klartexte $x^{(1)}, \ldots, x^{(q)}$ mit $t_1, \ldots, t_q$ Blöcken zur Verschlüsselung vorgelegt, in dieser Reihenfolge, mit $t_1 + \cdots + t_q \leq n$. Dabei werden in $\textsc{Sim}(x^{(i)}, F')$ die t_i Argumente $r^{(i)}, \ldots, (r^{(i)} + t_i - 1) \bmod 2^\ell$ für F' benutzt, für eine rein zufällige Zahl $r^{(i)} \in [2^\ell]$. Wir betrachten das Ereignis

$$W := \{\text{unter den als Argument für } F' \text{ benutzten Blöcken gibt es Wiederholungen}\},$$

und schätzen $\Pr'(W)$ ab. Dazu definieren wir Intervalle

$$I_i := \{r^{(i)}, \ldots, (r^{(i)} + t_i - 1) \bmod 2^\ell\}$$

(gegebenenfalls mit Wraparound, wenn $r^{(i)} + t_i > 2^\ell$), für $1 \leq i \leq q$. Für beliebige Werte $i < i'$ betrachten wir das Ereignis

$$W_{i,i'} := \{I_i \cap I_{i'} \neq \emptyset\}.$$

Dann ist $\Pr'(W_{i,i'}) \leq (t_i + t_{i'})/2^\ell$. (Um dies zu sehen, darf man sich I_i und $t_{i'}$ als durch den Ablauf bis vor der Wahl von $r^{(i')}$ schon festgelegt vorstellen. Nur $r^{(i')}$ wird zufällig gewählt.) Mit

$$W \subseteq \bigcup_{2 \leq i' \leq t} \bigcup_{1 \leq i < i'} W_{i,i'}$$

und $\sum_{1 \leq i < i'} t_i \leq n$ erhalten wir

$$\begin{aligned}\Pr'(W) &\leq \sum_{1 \leq i < i' \leq q} \frac{t_i + t_{i'}}{2^\ell} \leq \frac{1}{2^{\ell+1}} \sum_{1 \leq i, i' \leq q} (t_i + t_{i'}) \\ &= \frac{1}{2^{\ell+1}} \left(\sum_{1 \leq i' \leq q} \sum_{1 \leq i \leq q} t_i + \sum_{1 \leq i \leq q} \sum_{1 \leq i' \leq q} t_{i'} \right) = \frac{2qn}{2^{\ell+1}} = \frac{qn}{2^\ell}.\end{aligned}$$

Mit $C = \{S_U\langle b = 0\rangle = 1\}$ wie oben können wir konstatieren:

$$\Pr'(C) = \Pr'(C \mid \overline{W})\Pr'(\overline{W}) + \Pr'(C \mid W)\Pr'(W) \leq \Pr'(C \mid \overline{W}) + \frac{qn}{2^\ell}. \tag{1.10}$$

Bemerkung: Hier sieht man, wie der zweite Teil des Fehlerterms durch die Möglichkeit einander überschneidender Intervalle I_i zustandekommt. Wir sehen als Nächstes, dass die Fehlerwahrscheinlichkeit des Spiels genau $\frac{1}{2}$ ist, wenn diese Möglichkeit ausgeschlossen ist.

Wir zeigen nun: $\Pr'(C \mid \overline{W}) = \frac{1}{2}$. Die Idee dabei ist, dass ohne Wiederholungen bei den verschlüsselten Blöcken der Angreifer A keinerlei Möglichkeit hat, die Situationen $c = 0$ und $c = 1$ zu unterscheiden. Die Idee ist, dass die Blöcke, die in $\textsc{Sim}(z_c, F')$ aufgerufen werden, in AF und AG sonst überhaupt nicht vorkommen, also $\textsc{Sim}(z_c, F')$ ein rein zufälliger String der Länge $|z_0| = |z_1|$ ist, selbst unter der Bedingung, dass A die Verschlüsselung der anderen in AF und AG vorkommenden Klartextblöcke kennt. Für den Beweis gehen wir sorgfältig vor.

Es genügt, Folgendes zu beweisen:

Behauptung 2: $\Pr'(\{c = c'\} \cap \overline{W}) = \Pr'(\{c \neq c'\} \cap \overline{W})$.

Um dies zu beweisen, zerlegen wir $\overline{W}$ in disjunkte Teile, die Abläufen von AF und AG (Schritte 1 und 3) entsprechen, und definieren eine wahrscheinlichkeitserhaltende Bijektion zwischen diesen Teilen. Dabei wird ein Ablauf mit $c = c'$ auf einen mit $c \neq c'$ abgebildet und umgekehrt, so dass die Wahrscheinlichkeiten jeweils gleich sind. Daraus folgt dann Behauptung 2.

Die Idee ist recht einfach: Durch die Bijektion wird ein Ablauf, in dem in Schritt 2 Text z_0 verschlüsselt wird, auf einen abgebildet, in dem z_1 verschlüsselt wird, und umgekehrt, ohne dass der Rateteil AG des Angreifers einen Unterschied erkennen kann. Er wird also in beiden Fällen dasselbe Ergebnisbit c' zurückgeben. Damit liegt AG mit seinem Ergebnis c' genauso oft richtig wie falsch. Details folgen.

Betrachte eine Festlegung der Ergebnisse aller Zufallsexperimente (Wahl von F' und von c, Wahl der Offsets $r^{(i)}$ bei den Aufrufen von $\textsc{Sim}(\,.\,, F')$), die in $\overline{W}$ liegt, bei der also kein Block in $\{0,1\}^\ell$ zweimal verschlüsselt wird. In Schritt 1 (Findungsphase) berechnet AF zwei Texte $z_0 = (z_0^{(0)}, \ldots, z_{t-1}^{(0)})$ und $z_1 = (z_0^{(1)}, \ldots, z_{t-1}^{(1)})$ und einen Datensatz v. In Schritt 2 (Auswahl) wird der Text z_c verschlüsselt; hierzu wird $r \in 2^\ell$ gewählt und es wird

$$y' = (r, z_0^{(c)} \oplus_\ell F'(r), z_1^{(c)} \oplus_\ell (F'((r+1) \bmod 2^\ell)), \ldots, z_{t-1}^{(c)} \oplus_\ell (F'((r+t-1) \bmod 2^\ell)))$$

an AG übergeben. Die Berechnung $AG(v, \textsc{Sim}(\,.\,, F'), y')$ in Schritt 3 (Ratephase) liefert ein Bit c'.

Wir ändern die Zufallsergebnisse geschickt ab. Unverändert bleiben die Ergebnisse der Zufallswahl von Offsets $r^{(i)}$ bei den Aufrufen von $\textsc{Sim}(\,.\,, F')$. Zufallsbit c wird durch sein Komplement $1 - c$ ersetzt. F' wird durch eine Funktion F'' ersetzt, die wie folgt definiert ist (mit bitweiser $\oplus_\ell$-Operation auf Blöcken der Länge ℓ):

$$F''(r) := z_0^{(0)} \oplus_\ell z_0^{(1)} \oplus_\ell F'(r),$$

$\vdots$

$F''((r+t-1) \bmod 2^\ell) := z_{t-1}^{(0)} \oplus_\ell z_{t-1}^{(1)} \oplus_\ell F'((r+t-1) \bmod 2^\ell)$;

Außerhalb von $\{r, (r+1) \bmod 2^\ell, \ldots, (r+t-1) \bmod 2^\ell\}$ sind die Werte wie bei F'.

Das so beschriebene Elementarereignis $((r^{(i)})_{1\le i\le q}, 1-c, F'')$ hat dieselbe Wahrscheinlichkeit wie $((r^{(i)})_{1\le i\le q}, c, F')$. Man sieht auch sofort, dass dieselbe Transformation $(1-c, F'')$ wieder in (c, F') zurückverwandelt. Wie läuft U ab, wenn man die veränderten Ergebnisse benutzt? Ablauf und Ergebnis (z_0, z_1, v) von AF (Schritt 1) ändern sich nicht, da die Blöcke, in denen sich F' und F'' unterscheiden, nicht abgefragt werden. In Schritt 2 wird nun z_{1-c} verschlüsselt, also wird $y'' = \textsc{Sim}(z_{1-c}, F'')$ sowie v an AG übergeben. Aus der Definition von F'' folgt

$$\begin{aligned} y'' &= (r, z_0^{(1-c)} \oplus_\ell F''(r), \ldots, z_{t-1}^{(1-c)} \oplus_\ell F''((r+t-1) \bmod 2^\ell)) \\ &= (r, z_0^{(c)} \oplus_\ell F'(r), \ldots, z_{t-1}^{(c)} \oplus_\ell F'((r+t-1) \bmod 2^\ell)) \\ &= y'. \end{aligned}$$

Für den dritten Teil erhält AG also den Input $(v, \textsc{Sim}(\,.\,, F''), y')$. Die dadurch ausgelöste Berechnung läuft genauso ab wie die beim Aufruf $AG(v, \textsc{Sim}(\,.\,, F'), y')$, da in den Anfragen an F'' nur Blöcke vorkommen, auf denen sich F' und F'' nicht unterscheiden. Also kommt AG zu demselben Ergebnis c'. Daher sind die Resultate $[c = c']$ und $[1 - c = c']$ in Schritt 4 unterschiedlich (eines ist 0 und eines ist 1), und beide Berechnungen haben dieselbe Wahrscheinlichkeit.

Durch Summation über alle diese Paare von Ereignissen folgt Behauptung 2. Zusammen mit Behauptung 1 und (1.10) erhalten wir die gewünschte Ungleichung für $\mathrm{fail}(U)$:

$$\begin{aligned} \mathrm{fail}(U) = \Pr(S_U\langle b = 0\rangle = 1) &\le \Pr'(C) + \frac{n^2}{2^{\ell+1}} \\ &\le \Pr'(C \mid \overline{W}) + \frac{qn}{2^\ell} + \frac{n^2}{2^{\ell+1}} = \frac{1}{2} + \frac{2qn + n^2}{2^{\ell+1}}. \end{aligned}$$

Mit (1.5) erhalten wir:

$$\mathrm{adv}(U, \mathcal{B}) \ge \frac{1 + \mathrm{adv}(A, \mathcal{S})}{2} - \left(\frac{1}{2} + \frac{2qn + n^2}{2^{\ell+1}}\right) = \frac{\mathrm{adv}(A, \mathcal{S})}{2} - \frac{2qn + n^2}{2^{\ell+1}},$$

wie gewünscht.

Schließlich machen wir noch Bemerkungen zur Rechenzeit von U. Es geht dabei um die gesamte Rechenzeit für das Spiel $G_U^{\mathcal{B}}$. Dabei muss U sämtliche Orakelanfragen an F notieren, also alle schon benutzten Argumente s und die zugehörigen Werte $F(s)$, um bei erneutem Auftauchen des Arguments s den schon einmal gewählten Wert ausgeben zu können

(Funktionseigenschaft von F). Diese Argumente sind als q Intervalle I_i angeordnet, die sich eventuell überschneiden. Um diese zu speichern, benutzt man zum Beispiel einen balancierten Binärbaum, in dem Anfangs- und Endpunkte der Intervalle $[a, b]$ gespeichert sind, für die die Werte $F(a), F(a+1), \ldots, F(b)$ alle schon gewählt worden sind. Zu einem solchen Intervall speichert man dann die zugehörigen F-Werte in einem Array ab. Damit lassen sich diese Operationen für die q Orakelanfragen der Form $\textsc{Sim}(x, F)$ in Zeit $O(q \cdot \log(q) \cdot \ell)$ abarbeiten. (Der log-Faktor rührt von den Suchkosten im Baum her.) Insgesamt ergibt sich für $G_U^{\mathcal{B}}$ über die Rechenzeit t von A hinaus ein Zusatzaufwand von $O((n + q \cdot \log(q))\ell)$.