# Grundkurs Codierung

Lösungsvorschläge zu den Fragen in den Unterkapiteln „Was blieb?“
Stand 23.04.2007

## Unterkapitel 5.15, Seite 322

### Zu Frage 1:

Die Grundidee des McEliece-Verfahrens ist es, Codes für Fehlerkorrektur zur Verschlüsselung zu verwenden. Tatsächlich werden bei einem nicht systematischen Code die Info-Bits ( = Klartext) so in ein Codewort abgebildet, dass sie nicht mehr direkt lesbar sind. Wenn es nun möglichst viele verschiedene Generatormatrizen G gibt, mit denen sich die Codewörter berechnen lassen, hat man in der geheimen Auswahl einer solchen Matrix G bereits den Grundstock für ein Verschlüsselung. Die gezielte Verfälschung des Codewortes durch künstliche Fehler trägt überdies zusätzlich zur Erschwerung der Entschlüsselung mit “brute force”-Angriffen bei.

Der Goppa-Code eignet sich als Baustein besonders, da sich die Generatormatrix G für einen Code zur t-Fehlerkorrektur aus einer Vielzahl verschiedener Polynome c(z) über dem GF($2^m$) vom Grad t bilden lässt. Einzige Bedingung für c(z) ist es, dass dieses Polynom keine Nullstellen im GF($2^m$) besitzt. Nehmen wir das Beispiel eines Goppa-Codes über dem GF($2^4$) gemäß Unterkapitel 3.8.1, Seiten 161 ff, so hat der Klartext k die Länge m = 4 und jedes Codewort v zur Korrektur von t = 3 Fehlern die Länge n = $2^4$ = 16, die Abbildung ist

$$v = k \cdot G.$$

Die Anzahl möglicher Polynome c(z) vom Grad t = 3 als Schlüssel besteht nach Unterkapitel 5.7, Seite 296, aus 1360 Kandidaten – für die nennenswerte Erschwerung eines “brute force”-Angriffs natürlich noch viel zu wenig. Um die Schlüsselmenge zu vergrößern, kann man im nächsten Schritt eine zufällige, geheime, invertierbare (4·4)-Matrix wählen, siehe S. 298, Bild 5.8, und den Klartext hiermit multiplizieren:

$$k' = k \cdot S.$$

Wenn S als obere Dreiecksmatrix aufgebaut wird, ist sie auch mit geringem Aufwand invertierbar. Es gibt dann 63 Varianten, wegen

$$v' = k' \cdot G = k \cdot S \cdot G = k \cdot (S \cdot G)$$

steigt die Menge der Schlüssel auf 1360·63 = 85680 (S und G sind geheim!). Eine nochmalige, bedeutende Vergrößerung des “Schlüsselraums” gelingt, wenn v' mit einer zufällig gewählten, geheimen (16·16)-Permutationsmatrix P multipliziert wird (in Bild 5.8, Seite 298 ist eine solche Matrix der Größe (7·7) zur Veranschaulichung des Prinzips dargestellt, benötigt wird im vorliegenden Fall wegen der Länge 16 von v' allerdings eine (16·16)-Matrix). P ist wie S sehr einfach herstellbar, man muss nur dafür sorgen, dass die 1-Elemente in jeder Zeile immer an anderen Stellen stehen, so dass keine linearen Abhängigkeiten entstehen. Dann ist eine solche Matrix wegen $P^{-1} = P^T$ immer auch besonders einfach invertierbar.

Als Ergebnis hat man bis hierher

$$v'' = v' \cdot P = k' \cdot G \cdot P = k \cdot S \cdot G \cdot P = k \cdot (S \cdot G \cdot P)$$

erhalten. Da es für die Wahl von P eine Anzahl von $(2^4)!$ = 20922789888000 Möglichkeiten gibt (das Ausrufungszeichen steht für die Funktion “Fakultät”), steigt die Mächtigkeit des Schlüsselraums auf

$$1360 \cdot 63 \cdot 20922789888000 = 1792664637603840000 \approx 1.7 \cdot 10^{18},$$

also etwa auf 1.7 Trillionen. Im letzten Schritt wird auf v'' noch ein zufällig gewählter 3-Bit-Fehler e addiert,

wodurch schließlich der eigentliche Geheimtext g entsteht:

$$g = v'' + e = k \cdot (S \cdot G \cdot P) + e.$$

Hierfür gibt es “16 über 3” = 560 Varianten, die zwar nicht mehr Bestandteil des Schlüssels sind, aber “brute force”-Angriffe nochmals aufwändiger machen und die Anzahl auf

$$1360 \cdot 63 \cdot 20922789888000 \cdot 560 = 1003892197058150400000 \approx 1.0 \cdot 10^{21}$$

oder eine Trilliarden erhöhen. Der Fehler e interessiert im weiteren Verlauf nicht mehr.

Die genannte Zahl erscheint hoch, ist aber für “brute force”-Angriffe noch kein ernst zu nehmender Gegner. Wir haben, um die Übersichtlichkeit zu wahren, allerdings ein sehr “kleines” Beispiel gewählt. Prinzipiell kann man beliebig lange Codes verwenden, bei denen dann auch die Schlüsselräume die notwendigen astronomischen Umfänge annehmen.

Wenn man das Matrizenprodukt $(S \cdot G \cdot P)$ als öffentlichen und die 3 Matrizen S, G, und P für sich als geheimen Schlüssel einsetzt, hat man zugleich ein asymmetrisches Verfahren mit seinen Vorzügen bei der Schlüsselübergabe. Die Kenntnis des Produktes allein gestattet ohne zusätzliche Informationen keine schnelle Zerlegung in S, G und P. Die “normale” Entschlüsselung mit Hilfe der Schlüssel selbst ist auf Seite 299 beschrieben.

Zum Abschluss der Behandlung der vorliegenden Frage wird noch auf den Unterschied in der Wirkungsweise von S und P verwiesen: Mit S werden einzelne Bits des Klartextes durch additive Verknüpfungen in ihren Werten verändert, mit P werden einzelne Bits von v' auf andere Positionen gebracht.

### Zu Frage 2:

Die Sicherheit des Diffie-Hellmann-Schlüsseltausch-Verfahrens liegt in der Tatsache, dass man bisher nicht über schnelle Algorithmen zur Berechnung des diskreten Logarithmus verfügt. Der Einsatz von “brute force”-Methoden zum “Knacken” des der Schlüssel ist vergleichbar aufwändig wie bei der Zerlegung von Zahlen in Primfaktoren.

### Zu Frage 3:

Die Schwachstelle beim “normalen” Passwortverfahren besteht in erster Linie darin, dass das Passwort auf einem ungesicherten Kanal übertragen werden muss und daher abgehört werden kann. Damit besteht auch keinerlei Sicherheit. Beim Fiat-Shamir-Protokoll dagegen wird nicht das Passwort selbst übertragen, sondern nur Informationen darüber, ob der zu authentifizierende Partner (z. B. Bankkunde) im Besitz des richtigen Passwortes ist, ein entscheidender Unterschied. Dafür kann Einschränkung der Sicherheit jeder beliebige ungesicherte Kanal verwendet werden.

### Zu Frage 4:

Es ist zweckmäßig, die Authentifizierungsschritte des Fiat-Shamir-Protokolls oft zu durchlaufen, weil sich mit jedem Schritt die Wahrscheinlichkeit eines erfolgreichen Betrugsversuches halbiert. Bei 40 Schritten wäre die Wahrscheinlichkeit also bereits auf

$$0.5^{40} = 0.9\ 10^{-12}$$

oder 1 Billionstel gesunken.

## Zu Frage 5:

Das CRC-Verfahren gestattet über die aus der Nachricht u - oder u(x) in Polynomdarstellung - berechneten Prüfbits eine sichere Erkennung vor allem kleiner Veränderungen (alle 1-Bit-Fehler, alle 2-Bit-Fehler, alle Bit-Fehler ungerader Anzahl, alle Fehlerbündel von der Breite des Generatorpolynom-Grades und viele weitere, siehe Unterkapitel 3.7.10, Seite 155 ff). Wie bei allen Prüfbit-Algorithmen, bei denen die Anzahl der Prüfbits üblicherweise wesentlich geringer ist als die der Nachricht, gibt es alternative Nachrichten, welche die gleichen Prüfbits erzeugen. Man muss also zumindest für eine sichere Verschlüsselung der Prüfbits sorgen, um eine nicht erkennbare Manipulation von Nachrichten zu verhindern.

Der Weg bei Einsatz des CRC-Verfahrens wäre: Der Autor berechnet die Prüf-Bits, verschlüsselt sie mit seinem geheimen Schlüssel und veröffentlicht die Nachricht zusammen mit diesen verschlüsselten Prüf-Bits sowie dem Bildungsalgorithmus für die Prüf-Bits. Jeder Interessent der Nachricht kann sich dann die zugehörigen Prüf-Bits aus der der erhaltenen Nachricht bestimmen und mit den über den öffentlichen Schlüssel des Autors entschlüsselten Prüf-Bits vergleichen.

Zusätzlich kann der Autor zu den verschlüsselten Prüfbits noch die Anzahl der Nachrichten-Bits verschlüsselt angeben, damit zumindest eine Längenmanipulation erkennbar wird.

Das CRC-Verfahren ist dennoch nicht optimal, da jedes additive Nachrichten-Polynom $u^{*}(x)$, welches ein Vielfaches $q(x)\cdot g(x)$ des Generatorpolynoms darstellt, als Prüfbits nur “0”-Elemente ergibt. Addiert man also auf die Nachricht u(x) einen Anteil $u^{*}(x) = q(x)\cdot g(x)$ drauf, würde sich bei den originalen oder verschlüsselten Prüf-Bits keine Änderungen zeigen. Da es bei langen Nachrichten sehr viele solcher Nachrichtenzusätze $u^{*}(x) = q(x)\cdot g(x)$ gibt, könnte auch einer passen, der die originale Nachricht im Sinne des Betrügers unverfänglich verfälscht.

Der Betrüger könnte so vorgehen: Er verändert zunächst die Original-Nachricht u(x) in $u^{**}(x)$ und bildet die Differenz

$$d(x) = u(x) - u^{**}(x).$$

Diese Differenz muss nun durch Variation von $u^{**}(x)$ so gestaltet werden, dass

$$d(x) = u(x) - u^{**}(x) = q(x)\cdot g(x)$$

gilt. Sehr viele Nachrichten u**(x) werden diese Bedingung nicht erfüllen, aber die Wahrscheinlichkeit, mit Rechnerhilfe etwas passendes zu finden, ist nicht unrealistisch. Der Betrüger fängt also die Nachricht vor der technischen Veröffentlichung als “man in the middel” ab, manipuliert und veröffentlicht sie. Ein Leser hat nach Vergleich der Prüfsummen keinen Anlass, Verdacht zu schöpfen.

Besser eignen sich zum Nachweis der Unversehrtheit einer Nachricht Hash-Funktionen, wie sie in Unterkapitel 5.11, Seite 311 ff, beschrieben sind. Ein wesentliches Element ist hierbei das Verketten der Zwischenergebnisse im Ablauf der Hash-Summen-Berechnung (= Cipher Block Chaining). Der Wert der Hash-Summe hängt dann auch von der Reihenfolge der Klartextzeichen ab. Damit wird schon einmal verhindert, dass eine Vertauschung von Zeichen unentdeckt bleibt, siehe Beispiel auf Seite 312, “20:00 Uhr” und “02:00 Uhr”. Häufig angewendet werden z. B. die Hash-Summen nach dem Berechnungsverfahren mit der Bezeichnung MD5, wie etwa bei den frei ladbaren Installationsfiles der OpenOffice-Suite.

## Zu Frage 6:

Die eindeutige Zuordnung einer Hash-Summe zu einem Klartext ist dann unmöglich, wenn der Klartext mindestens 1 Bit länger als die die Hash-Summe ist. Eine eindeutige Zuordnung wäre nur bei gleichlangen Hash-Summen möglich, was aber den Aufwand stark erhöht und diese Variante uninteressant macht.

### Zu Frage 7:

PGP und GnuPP sind Hybrid-Verfahren aus symmetrischer und asymmetrischer Verschlüsselung, siehe Unterkapitel 5.13, Seite 317 ff. Für die eigentliche Verschlüsselung nutzen sie ein schnelles symmetrisches Verfahren, z. B. den DES- oder heute besser AES-Algorithmus. Hierzu ist jedoch der Austausch des symmetrischen Schlüssels notwendig, der nur über einen sicheren Kanal erfolgen darf. Da zu Anfang der Session der Kanal offen und damit unsicher ist, erzeugt der eine Partner für sich ein asymmetrisches Schlüsselpaar und schickt seinen öffentlichen Schlüssel an den Gegenüber. Der unsichere Kanal stellt hierbei keine Gefahr dar.

Nun wählt der Gegenüber einen geheimen Schlüssel für das symmetrische Verfahren, verschlüsselt diesen mit dem gerade erhaltenen öffentlichen Schlüssel des Partners und schickt ihm diesen zu. Auch jetzt stellt der offene Kanal kein Sicherheitsrisiko dar. Nach Entschlüsselung mit seinem geheimen asymmetrischen Schlüssel haben beide einen nur ihnen bekannten, geheimen Schlüssel für das symmetrische Verfahren. Damit ist der Kanal für die Dauer der Session sicher.

### Zu Frage 8:

Beim Verfahren der Quantenkryptographie wird sendet der eine Teilnehmer seinem Partner über einen öffentlichen, unsicheren Kanal eine zufällig gewählte, gleich verteilte Bit-Folge, die den Schlüssel für ein symmetrisches Verfahren enthält (aber nicht darstellt). Hierbei spielt auch die zufällig gewählte, gleich verteilte Polarisation der gesendeten Photonen eine zentrale Rolle. Die beiden Beispiel in Unterkapitel 5.14, Seiten 320 und 321, Bilder 5.8 und 5.9, können helfen, das besser zu verfolgen.

Nach Abschluss der Übertragung stellen beide Partner gemeinsam und öffentlich fest, an welchen Positionen der Folge sich die Schlüsselbits befunden haben, ohne deren Wert zu nennen. Da die Bits vom Sender zufällig und gleich verteilt gewählt wurden, muss auch die Schlüsselbitfolge gleich verteilt sein. Hat ein Dritter die Folge abgehört, so kann er aufgrund der quantenmechanischen Eigenschaften der Photonen Kopien der abgehörten Bits nur mit einer 50%-prozentigen Wahrscheinlichkeit richtig an den eigentlichen Empfänger weiter schicken.

Ein Abhören und Durchreichen der Photonen ist nicht möglich, sie sind nach dem Abhören verbraucht und ür die Weiterleitung müssen neue erzeugt und polarisiert werden. Wie zuvor verständigen sich auch in diesem Fall die beiden berechtigten Partner über die Positionen scheinbar sicherer Schlüsselbits, diese sind aber nun nicht mehr gelichverteilt, siehe Bild 5.9 auf Seite 321. Der Kanal wird darauf als unsicher verworfen.

In Verbindung mit dem one time pad steht damit auch zukünftig ein perfektes Verfahren bereit, unabhängig davon, ob die schnelle Zerlegung von Zahlen in Primfaktoren gelingen wird.