Versleuteling van een reeks bytes

Hans Lunsing

Bijvoorbeeld van een tekst of een heel bestand.

Inleiding

Om berichten tussen twee personen uit te wisselen zonder dat derden mee kunnen lezen, en om informatie in het algemeen privé te houden, worden die berichten en informatie versleuteld tot geheimschrift. Alleen iemand die over de juiste sleutel beschikt kan het geheimschrift ontcijferen (ontsleutelen) tot leesbare tekst dan wel bruikbare informatie. Zo'n sleutel kan bestaan uit een aantal cijfers en letters.

Versleuteling ofwel encryptie en ontsleuteling ofwel decryptie worden uitgevoerd aan de hand van een recept in de vorm van een algoritme. Vroeger gebeurde dat veelal met de hand (handcijfer), en kon het algoritme dus niet te complex zijn. Tegenwoordig wordt voor cryptografie[1], het proces van encryptie en decryptie, de computer gebruikt, die ook complexere algoritmes zeer snel kan uitvoeren. Voor ontsleuteling is natuurlijk naast de sleutel ook het juiste algoritme, als het ware de juiste deur, nodig.

Er kunnen twee vormen van cryptografie worden onderscheiden:
- *symmetrische*, waarbij voor versleuteling en ontsleuteling dezelfde sleutel wordt gebruikt.
- *asymmetrische*, waarbij een sleutel wordt gebruikt om de klare tekst te versleutelen of te ondertekenen, en een andere sleutel om de versleutelde tekst te ontsleutelen of de identiteit van de afzender te verifiëren.

Er zijn twee typen symmetrische cryptografie:
- *continue versleuteling*, waarbij de bytes van een bericht een voor een worden versleuteld.
- *blokversleuteling*, waarbij een bepaald aantal bits als een blok wordt versleuteld, en waarvoor het bericht wordt aangevuld tot een veelvoud van de blokgrootte. Voorbeeld is AES.

Vrijwel alle symmetrische versleutelingen zijn *reciproque versleutelingen*, waarbij voor versleuteling en ontsleuteling hetzelfde algoritme wordt toegepast.

Voorbeelden van symmetrische cryptografie zijn de vroegere handcijfers, en bekende computeralgoritmes als DES (Data Encryption Standard), IDEA (International Data Encryption Algorithm) en AES (Advanced Encryption Standard).

Nadeel van symmetrische cryptografie is dat in tegenstelling tot asymmetrische cryptografie beide partijen, verzender en ontvanger, toegang hebben tot de geheime sleutel. Voordeel ervan is dat het geschikter is voor versleuteling van grote hoeveelheden data omdat het minder rekenwerk vergt en sneller is dan asymmetrische cryptografie.

Bij asymmetrische cryptografie is sprake van twee sleutels, een publieke en een private. Een bericht dat met een publieke sleutel wordt versleuteld, kan alleen met de geheime private sleutel worden ontsleuteld. Alleen degenen in het bezit van die geheime sleutel kunnen het bericht dus lezen.
Andersom geldt dit ook: informatie die is versleuteld met iemands private sleutel kan alleen met de bijbehorende publieke sleutel worden ontsleuteld. Dit wordt gebruikt bij het digitaal ondertekenen van berichten. De ontvanger weet dan zeker dat het bericht afkomstig is van degene die zegt de afzender te zijn.

Een voorbeeld van asymmetrische cryptografie is het RSA (Rivest-Shamir-Adleman) algoritme. Protocollen die van asymmetrische cryptografie gebruik maken zijn bijvoorbeeld PGP (Pretty Good Privacy) en SSH (Secure Shell Protocol).

Belangrijk voordeel van asymmetrische cryptografie is dat de benodigde sleutels via een onveilig kanaal kunnen worden uitgewisseld. Wel moeten afzender en ontvanger elkaars identiteit vaststellen en hun publieke sleutels bevestigen via een veilig kanaal. Nadeel van asymmetrische cryptografie is dat lange sleutels nodig zijn, bijvoorbeeld 4096 bytes, waardoor het veel rekenwerk vergt.

Vaak wordt een combinatie van asymmetrische en symmetrische cryptografie gebruikt: de sleutel voor de snellere symmetrische cryptografie van grote blokken data wordt door middel van asymmetrische cryptografie tussen afzender en ontvanger uitgewisseld.

Een eigen algoritme

Een aantal jaren geleden heb ik eens een encryptie-algoritme bedacht voor een e-mailprogramma dat door een kennis van me was ontwikkeld. Het is een symmetrisch algoritme met continue versleuteling. Het is ongetwijfeld niet goed genoeg voor staatsgeheimen, maar voldoende om privé te houden wat privé moet blijven. In deze bijdrage leg ik uit hoe het algoritme werkt. Ik heb het algoritme uitgewerkt in een beknopte module voor FreeBASIC. Die komt een volgende keer aan de orde.

Bytes en coderingstabellen

Alle computerbestanden bestaan uit bits - enen en nullen - en zijn zoals gebruikelijk ingedeeld in bytes, dat zijn eenheden van 8 bits. Als decimaal getal lopen bytes van 0 (alle 8 bits 0) t/m 255 (alle 8 bits 1). In veel gevallen, onder meer in teksten, kunnen ze tekens zoals letters en cijfers coderen. In de gebruikelijke tekencodering is de betekenis van de 128 bytes 0 t/m 127, waarvan het hoogste bit 0 is, is vastgelegd in de standaard ASCII-tabel[2]. De standaard ASCII-tabel maakt ook deel uit van de tekenset ‘Latijn ISO 8859-1’, die veelal in Windows wordt gebruikt, en van de moderne tekenset UTF=8.

- Bytes 0 t/m 31 zijn zgn. besturingscodes (*control characters*), waaronder de *backspace*, de *tab*, de *carriage return*, *linefeed* en *escape*. Byte 127 is de DEL (*delete*), en hoort eigenlijk ook tot de besturingscodes. Deze bytes kunnen niet worden afgedrukt.

- De 95 bytes 32 t/m 126 zijn de cijfers, hoofdletters, kleine letters en leestekens. Deze kunnen allemaal worden afgedrukt.

De bytes 128 t/m 255 zijn opgenomen in de uitgebreide ASCII-tabel. Hoe dat deel van de ASCII-tabel is ingevuld hangt af van de gebruikte coderingstabel (*code page*).

- In MSDOS werd coderingstabel 437 (CP437)[3] gebruikt. Ook de moderne FreeBASIC compiler gebruikt CP437. De bytes 128 t/m 255 coderen letters met diakritische tekens en verscheidene bijzondere symbolen zoals de breuk ½, allerlei grafische tekens en wiskundige symbolen zoals ∫.

- In modernere systemen wordt voor de westerse tekenverzameling normaliter ISO-8859-1 of de verbeterde ISO-8859-15 met euroteken, en in Windows wel Windows 1252 toegepast[4].

In ISO-8859-1 zijn de eerste 32 bytes 128 t/m 159 besturingscodes, en zijn de overige bytes letters met diakritische tekens, zoals ë, en een aantal bijzonder symbolen, zoals het copyright teken ©. ISO-8859-15 is grotendeels gelijk aan ISO-8859-1, maar een klein aantal codes is vervangen, zodat onder meer de euro (€) een plaats kreeg. Ook Windows 1252 is grotendeels gelijk aan ISO-8859-1.

- Geleidelijk aan wordt steeds meer UTF-8 (8-bit *Unicode Transformation Format*)[5] toegepast. Hierin worden 1 tot 4 bytes voor een teken gebruikt. De standaard ASCII codes 0 t/m 127 worden met één byte weergegeven, terwijl voor de tekens van de uitgebreide ASCII-tabel van ISO-8859-1 twee bytes worden gebruikt. Hierbij is de eerste byte 110 gevolgd door 000 en de hoogste twee bits van het teken, en de tweede 10 gevolgd door de overige zes bits. Dit komt erop neer dat de uitgebreide ASCII codes 128 t/m 191 als zodanig worden weergegeven voorafgegaan door byte 194 (1100010, Â), en de ASCII codes 192 t/m 255 ook als 128 t/m 191, voorafgegaan door 195 (1100011, Ã).

Of de tekens die in de uitgebreide ASCII-tabel zijn gecodeerd als 128 t/m 255 kunnen worden afgedrukt, hangt dus af van de codetabel die door het afdrukmedium wordt gebruikt.

Caesar-versleuteling

De encryptieprocedure is gebaseerd op de eenvoudige gedachte om de bytes van een bytereeks te versleutelen, door een of ander getal, de sleutel, bij hun bytecode op te tellen, en bij ontsleuteling er weer af te trekken. Bij de optelling kan het gebeuren dat het resultaat buiten het bereik van de bytereeks komt. Stel dat we alle bytes versleutelen door bij hun bytecode, lopend van 0 t/m 255, 43 op te tellen. Dan zouden de hoogste 43 bytes 213 t/m 255 de waarden 256 t/m 298 krijgen, maar die passen niet meer in een byte. We nemen ze modulo 256 (dat is de rest bij deling door 256), waardoor ze naar het begin van de bytereeks verhuizen en de waarden 0 t/m 42 krijgen. Dat komt erop neer dat we na het bereiken van 255 opnieuw beginnen met 0.

Stel je een cirkel voor met op de buitenrand de te versleutelen bytereeks, en daarom heen een draaibare rand met eveneens dezelfde bytereeks. Deze wijze van versleuteling komt dan neer op het draaien van de rand over een aantal plaatsen, waarna voor elke byte op de cirkel de versleutelde byte is af te lezen op de rand. Dit wordt ook wel een Caesar-versleuteling genoemd omdat Caesar die als eerste zou hebben toegepast op zijn correspondentie met Cicero.

Over modulair rekenen: M modulo N (M en N zijn getallen) is niet anders dan het bepalen van de rest van M bij deling door N. Denk bijvoorbeeld aan de 12-uurs klok, terwijl een dag 24 uur heeft. 17 uur, bijvoorbeeld, is op de klok dan 17 modulo 12 = 5 uur[6]. Zo wordt de byteversleuteling een verschuiving (Engels: shift) binnen de gekozen reeks bytes.

We kunnen ook besluiten om niet alle bytes te versleutelen, maar een deelreeks ervan. We stellen dan als voorwaarde dat het resultaat ook binnen deze bytereeks valt. Dat houdt in dat we na het bijtellen van de sleutel testen of het resultaat buiten de bytereeks ligt, m.a.w. of het resultaat groter is dan de grootste byte van de bytereeks.

Als dat zo is komen we weer binnen de bytereeks door de lengte van de bytereeks eraf te trekken. Met modulair rekenen met de lengte van de bytereeks als modulus kunnen we het resultaat in één formule bereiken zonder dat het nodig is te testen of het resultaat buiten de bytereeks ligt. Hiertoe verminderen we de uitkomst van de bijtelling van de sleutel eerst met de kleinste byte van de reeks, en tellen die na het nemen van de modulus er weer bij.

De sleutel kiezen we zo dat hij niet meer is dan de lengte van de bytereeks, verminderd met 1. Meer heeft immers geen zin omdat we dezelfde uitkomst krijgen als we een of meerdere keren de lengte van de bytereeks van de sleutel eraf trekken. Dat betekent dat we de sleutel modulo de lengte van de bytereeks nemen.

Stel nu bij wijze van voorbeeld dat we alleen de 26 kleine letters (bytecodes 97 t/m 122) versleutelen met de sleutel 23. Als we boven de 122 uitkomen trekken we gewoon de lengte van de bytereeks eraf, zodat we weer binnen de bytereeks komen. Versleuteling van 120 geeft dan 143, wat na aftrek van 26 het resultaat 117 geeft.

Geen enkele andere byte in de reeks kan dat resultaat geven. Trek er maar eens 23 vanaf, dan is de uitkomst 94. Om binnen de bytereeks te vallen moet daar weer 26 bij, zodat we uitkomen op de 120 waarmee we begonnen.

Met modulair rekenen krijgen we in dit voorbeeld:

(120 + 23 - 97) modulo 26 + 97 = 46 modulo 26 + 97 = 20 + 97 = 117.

Om te ontsleutelen gebruiken we dezelfde formule maar trekken we er 23 af. Om een negatief getal te vermijden tellen we eerst het aantal te versleutelen bytes, 26, erbij. Dat mag omdat 26 de modulus is: 26 modulo 26 is nul en heeft dus geen invloed op het resultaat.

(117 + 26 – 23 – 97) modulo 26 + 97 = 23 modulo 26 + 97 = 23 + 97 = 120

De modulus mogen we bij de versleuteling ook gerust bijtellen, zodat we in beide gevallen, afgezien van het teken van de sleutel 23, dezelfde formule gebruiken:

(120 + 26 + 23 - 97) modulo 26 + 97 = 72 modulo 26 + 97 = 20 + 97 = 117.

Goed, nu een compleet voorbeeld van een stukje tekst. Alleen de bytecodes van de afdrukbare tekens komen voor versleuteling in aanmerking, dat zijn de bytecodes 32 t/m 126. Met dit uitgangspunt versleutelen we de tekst:

Hans en Grietje

De ASCII bytecodes zijn, weergegeven tussen < >:

H a n s e n G r i e t j e
<72> <97><110><115> <32><101><110> <32>
<71><114><105><101><116><108><101>

1. We trekken eerst van alle bytecodes de kleinste bytecode, dat is 32, van de te versleutelen bytereeks af:

<40> <65> <78> <83> <0> <69> <78> <0> <39> <82> <73>
<69> <84> <76> <69>

2. Vervolgens tellen we bij alle ASCII-codes, bij wijze van voorbeeld, 72 op. Dat is de sleutel. Het resultaat is:

<112><137><150><155> <72><141><150>
<72><111><154><145><141><156><148><141>

3. Omdat we binnen de reeks afdrukbare ASCII-tekens, die lopen van 32 t/m 126, na aftrek van 32, dus van 0 t/m 94, willen blijven, nemen we al deze getallen modulo 95, dat is de lengte van de reeks:

<17> <42> <55> <60> <72> <46> <55> <72> <16> <59> <50>
<46> <61> <53> <46>

4. Tot slot tellen we er weer 32 bij:

<49> <74> <87> <92><104> <78> <87><104> <48> <91> <82>
<78> <93> <85> <78>
1 J W \ h N W h 0 [R N] U N

De versleutelde tekst is dus (oorspronkelijke tekst eronder):

1JW\hNWh0[RN]UN
Hans en Grietje

Ontsleuteling gaat in omgekeerde richting, met dien verstande dat er, bij aftrek van 72, negatieve getallen kunnen ontstaan. Om dat te voorkomen tellen we er 95 bij op. Dat mag omdat we toch modulo 95 werken.

1. Eerst trekken we er weer 32 af, en tellen er 95 bij. Per saldo tellen we er dus 63 bij:

<112><137><150><155><167><141><150><167><111><154><1
45><141><156><148><141>

2. Dan trekken we er 72 af:

<40> <65> <78> <83> <95> <69> <78> <95> <39> <82> <73>
<69> <84> <76> <69>

3. Vervolgens nemen we modulo 95

<40> <65> <78> <83> <0> <69> <78> <0> <39> <82> <73>
<69> <84> <76> <69>

4. En tot slot tellen we er weer 32 bij:

<72> <97><110><115> <32><101><110> <32>
<71><114><105><101><116><108><101>

Dit zijn, zoals bedoeld, weer de oorspronkelijke byte-codes. De ontsleutelde tekst is dus:

Hans en Grietje

Pseudo-toevalsgetallen als variërende bijtelling

Probleem met deze methode is dat hij, zeker bij wat langere teksten, vrij gemakkelijk te kraken is door te kijken naar het relatieve voorkomen van de verschillende letters, gegeven een bepaalde taal. Die frequentie verandert namelijk niet als je alle bytecodes met een zelfde getal ophoogt. Wat we dus eigenlijk zouden willen is de achtereenvolgende bytecodes met verschillende getallen ophogen. We kunnen daarvoor alle 256 bytes gebruiken, maar minder mag natuurlijk ook. Probleem is dat je voor ontsleuteling de hele reeks bijtellingen zou moeten kennen. Dat is niet echt handig. Is daar wat aan te doen? Ja, dankzij een pseudo-toevalsgenerator, die bij dezelfde startwaarde (‘seed’) steeds dezelfde serie pseudo-toevalsgetallen produceert . Echt toevallig zijn die getallen niet, maar het is een goede benadering. Het is immers vrijwel onmogelijk te voorspellen wat, gegeven een al bepaald toevalsgetal, het volgende toevalsgetal zal zijn. Dankzij het feit dat bij een zelfde startwaarde steeds dezelfde reeks toevalsgetallen wordt geproduceerd, kunnen we een versleutelde reeks bytes met dezelfde startwaarde ook weer ontsleutelen. Anders zou dat onmogelijk zijn. Die startwaarde is dus de sleutel.

De belangrijkste algemene programmeertalen hebben wel zo'n toevalsgenerator. Zo ook FreeBASIC, dat zelfs vijf verschillende algoritmes voor de toevalsgenerator aanbiedt. De standaard is de Mersenne-twister, die een hoge graad van willekeurigheid (‘randomness’) heeft. Dat is degene die we zullen gaan gebruiken.

De lengte van de reeks toevalsgetallen is niet groter dan het aantal mogelijke bytes, dat is 256. Als de te versleutelen serie bytes (bijvoorbeeld een tekst) langer is dan de lengte van de reeks toevalsgetallen beginnen we weer van voren af aan met de reeks toevalsgetallen. De toevalsgetallen zelf zijn maximaal het aantal voor versleuteling in aanmerking komende bytes. Als we ons beperken tot de afdrukbare tekens is het maximum dus 95.

Hier volgt een voorbeeld met zeven toevalsgetallen, die maximaal het aantal afdrukbare tekens zijn: [7 25 8 50 47 91 32], toegepast op de tekst ‘Hans en Grietje’. We beginnen weer net als hierboven:

De ASCII bytecodes zijn, weergegeven tussen < >:

H a n s e n G r i e t j e
<72> <97><110><115> <32><101><110> <32>
<71><114><105><101><116><108><101>

1. We trekken eerst van alle bytecodes 32 af:

<40> <65> <78> <83> <0> <69> <78> <0> <39> <82> <73>
<69> <84> <76> <69>

2. Vervolgens tellen we bij alle ASCII-codes achtereenvolgens de reeks toevalsgetallen op. Eenmaal bij de laatste aangekomen beginnen we weer met de eerste:

<7> <25> <8> <50> <47> <91> <32> <7> <25> <8> <50> <47>
<91> <32> <7>

met als resultaat:

<47> <90> <86><133> <47><160><110> <7> <64>
<90><123><116><175><108> <76>

3. Dan gaan we op dezelfde manier als boven verder. Omdat we binnen de reeks afdrukbare ASCII-tekens, die lopen van 32 t/m 126, na aftrek van 32, dus van 0 t/m 94, willen blijven, nemen we al deze getallen modulo 95, dat is het aantal afdrukbare ASCII-tekens:

<47> <90> <86> <38> <47> <65> <15> <7> <64> <90> <28>
<21> <80> <13> <76>

4. Tot slot tellen we er weer 32 bij:

<79><122><118> <70> <79> <97> <47> <39> <96><122> <60>
<53><112> <45><108>
O z v F O a / ' ` z < 5 p - l

De versleutelde tekst is dus (oorspronkelijke tekst eronder):

OzvFOa/'`z<5p-l
Hans en Grietje

We zien dat dezelfde oorspronkelijke letter, bijvoorbeeld de ‘e’, steeds een andere versleuteling krijgt, in dit geval ‘a’, ‘5’ en ‘l’. Dat is precies wat we wilden.

Ontsleuteling gaat in omgekeerde richting, met dien verstande dat bij aftrek van de reeks toevalsgetallen negatieve getallen kunnen ontstaan. Om dat te voorkomen tellen we er 95 bij op. Dat mag omdat we toch modulo 95 werken.

1. Eerst trekken we er weer 32 af, en tellen er 95 bij. Per saldo tellen we er dus 63 bij:

<142><185><181><133><142><160><110><102><159><185><1
23><116><175><108><171>

2. Dan trekken we de achtereenvolgende reeks toevalsgetallen eraf:

<135><160><173> <83> <95> <69> <78> <95><134><177> <73>
<69> <84> <76><164>

3. Vervolgens nemen we de bytecodes modulo 95

<40> <65> <78> <83> <0> <69> <78> <0> <39> <82> <73>
<69> <84> <76> <69>

4. En tot slot tellen we er weer 32 bij:

<72> <97><110><115> <32><101><110> <32>
<71><114><105><101><116><108><101>

Dit zijn, zoals bedoeld, weer de oorspronkelijke bytecodes. De ontsleutelde tekst is dus:

Hans en Grietje

Nog een extra moeilijkheidsniveau

Om het voor eventuele hackers nog wat moeilijker te maken, gebruiken we de reeks toevalsgetallen niet achtereenvolgens, maar laten we het voor elke byte te gebruiken toevalsgetal bepalen door een tweede, even lange reeks, toevalsgetallen. Deze tweede reeks bevat de nummers van de te gebruiken toevalsgetallen en loopt dus van 0 t/m de lengte van de eerste reeks toevalsgetallen -1. Let wel dat een bepaald nummer dan best meerdere keren kan worden gebruikt.

Voorbeeld: stel dat de eerste reeks toevalsgetallen is als in het voorbeeld hierboven: [7 25 8 50 47 91 32], dan zou de tweede reeks met de nummers van de te gebruiken toevalsgetallen er als volgt uit kunnen zien: [3 1 1 6 1 2 5]. De waarden van de te gebruiken toevalsgetallen zijn dus achtereenvolgens [50 25 25 32 25 8 91].

We kunnen deze tweede reeks toevalsgetallen bepalen zonder een nieuwe startwaarde voor de toevalsgenerator te geven. We kunnen ook dezelfde startwaarde opnieuw geven. We krijgen dan dezelfde reeks toevalsgetallen, die nu naar nummers van bijtellingen in plaats van de bijtellingen zelf worden vertaald. Extra moeilijk maken we het voor hackers als we voor deze tweede reeks een andere startwaarde kiezen. Om een versleutelde bytereeks te ontsleutelen zijn dan dus twee sleutels nodig.

Slotopmerkingen

Zoals we hebben gezien is de uitkomst van dit versleutelingsalgoritme afhankelijk van enkele uitgangswaarden:
- De reeks voor versleuteling in aanmerking komende bytes, weer te geven door de bytecodes van de kleinste en de grootste byte in de reeks. Stel dat we alleen de afdrukbare tekens versleutelen, dan zijn de kleinste en grootste byte resp. 32 en 126.
- Het aantal te gebruiken toevalsgetallen in het byte-formaat, maximaal 255.

Met uitgangswaarden die afwijken van de uitgangswaarden die bij versleuteling zijn gebruikt is een versleutelde reeks bytes niet te ontsleutelen. Een programma voor ontsleutelen met deze methode zal dus dezelfde uitgangswaarden moeten hebben als het programma dat voor versleutelen is gebruikt.

Het moge duidelijk zijn dat de rangorde van een byte in de te versleutelen reeks bytes mede bepaalt hoe die byte zal worden versleuteld. Dat heeft tot gevolg dat, wanneer een byte uit de versleutelde reeks wordt verwijderd, alles wat daarna komt niet zonder meer nog te ontsleutelen is.

Het hier beschreven versleutelingsalgoritme heb ik uitgewerkt in een FreeBASIC module, die in FreeBASIC programma’s kan worden gebruikt. Dat komt een volgende keer aan de orde.

Links

1. De hier gegeven Informatie over cryptografie is ontleend aan Wikipedia.
2. zie: **https://nl.wikipedia.org/wiki/ASCII_(tekenset)**
3. zie: **https://en.wikipedia.org/wiki/Code_page_437**
4. zie: **https://nl.wikipedia.org/wiki/ISO_8859-1**
5. zie: **https://nl.wikipedia.org/wiki/UTF-8**
6. zie: **https://nl.wikipedia.org/wiki/Modulair_rekenen**
7. zie: **https://nl.wikipedia.org/wiki/Pseudotoevalsgenerator**