Chiffrierung nach ELGAMAL

14. April 2013

1 Allgemeines

Das ELGAMAL-Verfahren ist ein asymmetrisches Chiffrierverfahren, d. h., zur Entschlüsselung ist ein anderer Schlüssel notwendig als zur Verschlüsselung. Derartige Chiffrierverfahren werden aber aufgrund ihrer Rechenintensität nur in hybriden Systemen eingesetzt, so daß damit nur ein z. B. symmetrischer Sitzungsschlüssel geschützt wird. Das ELGAMAL-Verfahren ist u. a. im „Handbook of Applied Cryptography“[1] beschrieben.

2 Erzeugung des Schlüsselpaares

1. Bestimme eine große, sichere Primzahl $p = 2q + 1$, so daß auch q eine Primzahl ist.
2. Bestimme ein primitives Element α, auch Generator[2] genannt, bzgl. $\mathbf{Z}_p^*$.
3. Wähle ein beliebiges a mit $1 \leq a \leq p-2$.
4. Berechne $\beta := \alpha^a \bmod p$.

Der öffentliche Schlüssel ist dann $\mathscr{K}_{\text{pub}} := (p, \alpha, \beta)$ und der private $\mathscr{K}_{\text{prv}} := (p, a)$.

3 Verschlüsselung mit öffentlichem Schlüssel

Sei nun $x \in \mathscr{P}$ mit $0 \leq x \leq p-1$ ein beliebiger Klartextblock. Dann wird wie folgt chiffriert:

$$\begin{aligned} \mathscr{E}_{\mathscr{K}_{\text{pub}}}(x) &:= (y_1, y_2) \\ y_1 &:= \alpha^\lambda \bmod p \\ y_2 &:= x \cdot \beta^\lambda \bmod p \end{aligned}$$

[1] http://cacr.uwaterloo.ca/hac/

[2] $\mathbf{Z}_p^* = \{\alpha^i \bmod p \mid 0 \leq i \leq p-2\}$

Wie man leicht sieht, ist der Geheimtext doppelt so lang wie der Klartext. Dies ist auch ein Grund dafür, weshalb dieses Verfahren ausschließlich in hybriden Systemen benutzt wird. Das λ ist für jeden Block zufällig zu wählen mit $1 \leq \lambda \leq p-2$. Die Menge der mögliche Geheimtexte ist größer als die Menge Klartexte, d.h., ein Klartext wird bei zufälligem λ auf unterschiedliche Geheimtexte abgebildet. Diese Eigenschaft nennt man probabilistische Verschlüsselung. Im Gegensatz dazu ist RSA eine deterministische Verschlüsselung.

4 Entschlüsselung mit privatem Schlüssel

Den Klartext erhält man wieder durch:

$$\begin{aligned} \mathscr{D}_{\mathscr{K}_{\text{prv}}}(y_1, y_2) &:= y_2 \cdot y_1^{-a} \bmod p \\ &:= x \cdot \beta^{\lambda} \cdot \alpha^{-a\lambda} \bmod p \\ &:= x \cdot \alpha^{a\lambda} \cdot \alpha^{-a\lambda} \bmod p \\ &= x \end{aligned}$$

Eine kleine Verringerung des Rechenaufwandes ergibt sich durch Anwendung des Kleinen Satzes von Fermat:

$$\mathscr{D}_{\mathscr{K}_{\text{prv}}}(y_1, y_2) := (y_2 \bmod p) \cdot (y_1^{p-1-a} \bmod p)$$

5 Zahlenbeispiel

Der interaktive Interpreter ARIBAS[3] bietet eine gute Unterstützung für Berechnungen mit großen Zahlen.

```
(*
** Sucht die naechste (wahrscheinliche)
** sichere Primzahl p=2*q+1, so dass auch
** q eine (wahrscheinliche) Primzahl ist.
*)
function nextsafeprime(q: integer): integer;
var
    p: integer;
begin
    if even(q) then inc(q); end;
    p := 2*q+1;
    while factor16(q) or factor16(p) do
        inc(q,2); inc(p,4);
    end;
    while not rab_primetest(q) or not rab_primetest(p) do
```

[3] http://www.mathematik.uni-muenchen.de/~forster/sw/aribas.html

```
        write('.');
        inc(q,2); inc(p,4);
        while factor16(q) or factor16(p) do
            inc(q,2); inc(p,4);
        end;
    end;
    return p;
end;

(*
** Sucht einen Generator g für Z*_p, wobei
** p eine sichere Primzahl ist.
*)
function generatorsafep(p: integer): integer;
var
    g: integer;
    q:=(p-1) div 2;
begin
    while true do
        g := random(p-1);
        write('.');
        if g = 0 then
            continue;
        end;
        if g**2 mod p = 1 then
            continue;
        end;
        if g**q mod p = 1 then
            continue;
        end;
        break;
    end;
    return g;
end;

p:=nextsafeprime(random(2**512));
alpha:=generatorsafep(p);
a:=random(p-2);
beta:=alpha**a mod p;
lambda:=random(p-2);
x:=random(p-1);
y1:=alpha**lambda mod p;
y2:=beta**lambda mod p;
y2:=y2*x mod p;
```

```
xdec:=y1**(p-1-a) mod p;
xdec:=y2*xdec mod p;
x-xdec.
```

6 Sicherheit

Das Brechen der ElGamal-Chiffrierung bei bekanntem öffentlichen Schlüssel und Geheimtext ist vergleichbar mit dem Lösen des sog. Diffie-Hellman-Problems. Bis zum heutigen Zeitpunkt ist bekannt, daß das Lösen des Diffie-Hellman-Problems nur über die Berechnung diskreter Logarithmen möglich ist. Z. B. könnte ein Angreifer versuchen, den privaten Schlüsselteil a aus dem öffentlichen Schlüssel (p, α, β) zu berechnen: $\beta \equiv \alpha^a \bmod p$. Das Berechnen von a (sog. diskreter Logarithums) ist schwer im Vergleich zur Berechnung von β. Deshalb wird diese Funktion auch Einwegfunktion genannt.

Weiterhin ist es wichtig, λ immer zufällig zu wählen und dafür einen gute Zufallsquelle zu verwenden.