Angriffe auf NTRU Signature Scheme (NSS)

Seminar Gitter in der Kryptographie, Wintersemester 2002-2003

Jean-Pierre Schwickerath

`http://schwicky.net/projects/2003/nss/`

15. Januar 2003

Geschichte

Die moderne Form des »Veni, visus sum, victus sum« von Hoffstein, Pipher und Silverman.

- CRYPTO ’96 rump session: NTRU vorgestellt.
- Kurz darauf fanden Coppersmith und Shamir einen Angriff $\Rightarrow$ grössere Parameter, Neuformulierung des Problems als Gitterproblem.
- CRYPTO 2000: Ein Signaturverfahren, dass auch auf einem Gitterproblem basiert wird vorgeschlagen. Statistische Angriffe sind möglich weil Signaturen Informationen über den privaten Schlüssel preisgeben.
- Eurocrypt 2001: NSS vorgestellt. Es ist eine Verbesserung des vorherigen Verfahrens.

Die Schwachstellen

- Man behauptet NSS basiert auf dem gleichen schwierigen Problem wie NTRU.
- Das stimmt nicht: Das Problem auf dem NSS basiert ist viel einfacher.
- Man kann Signaturen für jede beliebige Nachricht erstellen (existentielle Fälschung).
- Signaturen verraten Informationen über den privaten Schlüssel: mit einigen 10’000 Signaturen kann man den privaten Schlüssel finden.

Und sie wollten es noch einmal retten. . .

Das Schema

Die mathematische Struktur hinter den Schlüsseln ist der Polynomring

$$R = \mathbb{Z}_q[X]/(X^N - 1)$$

$N \in \mathbb{P}$ und $q = 2^n,\ n \in \mathbb{N}$

Die Elemente in R sind Polynome vom Grad höchstens $N - 1$ mit Koeffizienten in $(-\frac{q}{2}, \frac{q}{2}]$.

Typischerweise $q = 128,\ N = 251$.

Schlüssel und Parameter

Öffentlicher Schlüssel: Polynom h vom Grad $N - 1$.
Privater Schlüssel: Polynome f und g mit »kleinen« Koeffizienten, so dass $f * h = g$.
$f, g, h \in R$.

Des weiteren brauchen wir noch ein paar ganze Zahlen:
$p = 3,\ d_f = 70,\ d_g = 40,\ d_m = 32$

Man benutzt ausserdem: $\mathcal{L}(d_1, d_2)$ um Polynome vom Grad höchstens $N - 1$ mit d_1 Koeffizienten $= 1$, d_2 Koeffizienten $= -1$ und alle anderen Koeffizienten $= 0$ zu beschreiben.

Schlüsselerzeugung

$$f = f_0 + p \cdot f_1 \qquad\qquad g = g_0 + p \cdot g_1$$

wo f_0 und g_0 kleine bekannte Polynome sind.

Typisch: $f_0 = 1$ und $g_0 = 1 - 2X$.
f_1 ist zufällig in $\mathcal{L}(d_f, d_f)$ gewählt. Analog auch g_1 in $\mathcal{L}(d_g, d_g)$.

f muss invertierbar sein: $\exists\, f^{-1} \,.\, f * f^{-1} = 1 \mod q$
$\Rightarrow h = f^{-1} * g$

Signaturerstellung

Hashwert ist ein Polynom in $\mathcal{L}(d_m, d_m)$.

$$w = m + w_1 + p \cdot w_2$$

w_1, w_2 sind zwei zufällig gewählte Polynome mit kleinen Koeffizienten.

$$s = f * w \mod q$$

Signatur: $(m,\ s)$

w_1, w_2 werden von m abhängig generiert mit mindestens 25 von 0 verschiedenen Koeffizienten und gegen »averaging attacks« geschützt.

Signaturverifikation

Zwei weitere Parameter: D_{min} und D_{max} benötigt.
Empfohlen: $D_{min} = 55$ und $D_{max} = 87$.

Die Abweichung $Dev(A, B)$ ist die Anzahl der Koeffizienten wo sich $(A \mod q) \mod p$ und $(B \mod q) \mod p$ unterscheiden.
Bei $p = 3$ sollte $Dev(A, B) = \frac{2}{3}N \approx 167$ sein.

$$s \neq 0$$

$$t = s * h \mod q$$

$$D_{min} \leq Dev(s, f_0 * m) \leq D_{max}$$

$$D_{min} \leq Dev(t, g_0 * m) \leq D_{max}$$

Fälschungsangriffe

- System soll so sicher sein wie RSA mit 1024 bit Moduli.
- Angreifer kann ohne Kenntnis des privaten Schlüssels Fälschungen mit etwas weniger als $D_{max} = 87$ Abweichungen beinahe so schnell erstellen wie der Signierer selbst.
- Mit Gitterreduktion kann der Angreifer Signaturen mit beträchtlich weniger als D_{max} Abweichungen erstellen.
- NTRU und NSS sollen auf dem gleichharten Gitterproblem basieren: NSS beruht aber eher auf einem Fehlerkorrekturproblem $\Rightarrow$ dafür braucht man viel mächtigere Dimensionen.

Fälschung: Das Prinzip

Ziel den Angreifers ist es ein Polynomtupel (s, t) zu finden, das $t = s * h \mod q$ genügt und die Abweichungen $55 \leq Dev(s, f_0 * m) \leq 87$ sowie $55 \leq Dev(t, g_0 * m) \leq 87$ erfüllt.

s und t haben $2N$ Koeffizienten und $t = s * h \mod q$ bedingt N lineare Abhängigkeiten.

Dem Angreifer bleiben N Freiheitsgrade bei der Bestimmung von s und t.

Er setzt $s_i \equiv (f_0 * m)_i \mod p$ und $t_j \equiv (g_0 * m)_j \mod p$ für $\lfloor N/2 \rfloor$ Koeffizienten von s und $\lceil N/2 \rceil$ Koeffizienten von t.

$\frac{1}{3}N \approx 84 \leq D_{max}$ für $N = 251, D_{max} = 87$.

Fälschung: Die Wahrheit

Der Angreifer hat mit einer Warscheinlichkeit von höchstens $\epsilon = \frac{1}{2^{N-2k}}$ kein Glück (N Koeffizienten, davon k bedingte).
Für $k = 121$, hat der Angreifer mit Warscheinlichkeit $1 - 2^{-9}$ Glück.
Abweichungen werden mit Warscheinlichkeit $\geq \frac{1}{4}$ erfüllt
$\Rightarrow$ Erfolg nach 4 Multiplikationen.

Man kann auch Lösungen suchen, die eher in der Mitte des Intervalls (D_{min}, D_{max}) liegen indem man die 128^9 möglichen Lösungen der linearen Gleichungen durchsucht.

Plan B: Gitterreduktion

Gitterreduktion ist eine Technik um »brauchbare« Gitterbasen in $\mathbb{Z}$ zu finden.
Es benutzt die Tatsache, dass man ein grosse Freiheit der Wahl von s und t haben.
Alle möglichen »einfachen« Fälschungen unterscheiden sich von einer gegebenen durch einen $2N$-Dimensionalen Vektor.

- (1) Man erstellt eine Fälschung mit dem Basisangriff;
- (2) Man verbessert die initiale Abweichung mit einer Gitterreduktion.
- Innerhalb weniger Minuten kann man Fälschungen mit durchschnittlich 56 Abweichungen erstellen.

Gitterumformung – 1

Sei (s'', t'') die initiale Fälschung. Da $t'' = s'' * h \mod q$, ist (s'', t'') in dem Gitter, dass durch die Zeilen der Matrix

$$L_{CS} = \begin{pmatrix} I_{(N)} & M_h \\ 0 & qI_{(N)} \end{pmatrix}$$

generiert wird.
L_{CS} ist die Matrix, die Coppersmith und Shamir mit ihrem Angriff auf NTRU eingeführt haben.

Mit dem Standardangriff haben wir eine invertierbare $k \times k$ Untermatrix U von M_h gefunden. Damit wird L_{CS} reorganisert und es entsteht:

Gitterumformung – 2

$$L_{CS,2} = \begin{pmatrix} I_{(N-k)} & 0 & R & S \\ 0 & I_{(k)} & T & U \\ 0 & 0 & qI_{(N-k)} & 0 \\ 0 & 0 & 0 & qI_{(k)} \end{pmatrix}$$

$$L_{CS,3} = \begin{pmatrix} I_{(N-k)} & -V & R-VT & 0 \\ 0 & I_{(k)} & T & U \\ 0 & qI_{(k)} & 0 & 0 \\ 0 & 0 & qI_{(N-k)} & 0 \\ 0 & 0 & 0 & qI_{(k)} \end{pmatrix}$$

$$V = SU^{-1} \mod q$$

Abweichungen verbessern

Wir bauen ein Gitter wo die kurzen Vektoren, Vektoren mit kleiner Abweichungen repräsentieren.
Dann suchen wir nach einen »harmlosen Vektor«, der mit (s''', t''') addiert einen sehr kurzen Vektor ergibt.

$$L_{pq} = \begin{pmatrix} pL_{harmless} \\ (s', t') \end{pmatrix}$$

(s', t') ist die Zeile mit Koeffizienten modulo pq , die $s' \equiv s''' \mod q$ und $t' \equiv t''' \mod q$ sowie $s' \equiv (f_0 * m) \mod p$ und $t' \equiv (g_0 * m) \mod p$ genügt.

Jeder Vektor (v_s, v_t) in diesem Gitter verifiziert $v_s * h = v_t \mod q$.

Bessere Koeffizienten

Ausserdem erfüllen v_s und v_t je nach Wert von (s', t') eine der drei Gleichungen:
$v_s \equiv v_t \equiv 0 \mod p$ oder
$v_s \equiv (f_0 * m) \mod p$ und $v_t \equiv (g_0 * m) \mod p$ oder
$-v_s \equiv (f_0 * m) \mod p$ und $-v_t \equiv (g_0 * m) \mod p$.

Statt L_{pq} zu reduzieren, wählen wir c Spalten aus L_{pq}, die unbestimmten Koeffizienten aus (s''', t''') entsprechen und erstellen daraus L_{final}.

Der Durchbruch

Die erhofften Abweichungen für s und t sind $(2N - 2k - c)/3 + \delta/2$, wo δ die Anzahl der Koeffizienten des c-Vektors ist, die ausserhalb von $(-\frac{q}{2}, \frac{q}{2}]$ liegen.

Wenn man die »praktischen Implementierungen« von NSS angreift, dann kann man $k = 95$ und $c = 150$ setzen um das Gitter mit einer Blockgrösse von 20 zu reduzieren.

Innerhalb nur weniger Minuten sind die Reduktionen vollzogen und die Abweichungen von s und t sind ungefähr 56.

»Transcript Attacks«

Es ist möglich die privaten Schlüssel f und g wiederherzustellen indem man Signaturen analysiert. Eine Abschrift ist eine Menge von Paaren (m, s) – Nachrichten und gültige Signaturen. Auch erhält man $t = s * h \mod q$ für jede Nachricht.

Die Basis des Angriffs ist die Verteilung der Koeffizienten in s und t für eine Untermenge von Nachrichten zu analysieren.

Indem man in m einen Koeffizienten fixiert, konvergiert die Verteilung der Koeffizienten von s und t in korrelation zu ausgewählen Koeffizienten in f und g.

Nichts ist geheim

Man vergleicht also die s und t Proben mit vorberechneten Abschätzungen der möglichen Verteilungen für alle möglichen Werte der Koeffizienten in f und g.

Wie wissen, dass $s = f * (m + w_1 + pw_2) \mod q$.
Benutzung der Standardparameter: $q = 128,\ p = 3,\ N = 251$.
Dann haben w_1 und w_2 jeweils ca. 25 bzw. 64 nicht-Null Koeffizienten und m hat 32 Koeffizienten 1 und -1.

Um den Koeffizienten f_0 zu erhalten setzen wir i_0 und j_0 mit $i_0 = j_0 + k \mod N$. Dann betrachten wir die Verteilung von s_{i_0} über eine Abschrift von Nachrichten mit $m_{j_0} = 1$.

Halb sicher ist immernoch sicher genug

$$s_{i_0} = \sum_{j+k=i_0} f_k(m_j + w_{1,j} + pw_{2,j})$$

Die Anzahl $W_j = m_j + w_{1,j} + pw_{2,j}$ ist fast gleichverteilt für jedes Index j für zufällige Werte von m.
s_{i_0} sollte die Summe der zufälligen Variablen W_j sein. Da f exakt 140 nicht-Null Einträge hat ist s_{i_0} die Summe von 140 gleichverteilten zufälligen Werten. Aber wir haben m_{j_0} festgesetzt: So trennt sich W_{j_0} vom Rest.

Beobachtungen zeigen, dass der Term $f_k W_{j_0}$ in der Summe s_{i_0} verschiedene Einflüsse hat, je nach Wert von f_k.

Und wie sicher ist es denn nun?

Versuch mit mehreren Millionen Nachrichten, alle mit einem anderen privaten Schlüssel signiert.

Signaturen	Versuche	Durchschnittlicher Fehler
100’000	31	7.3
300’000	16	2.6
400’000	5	1.2

Man könnte ein partielles Raten des Schlüssels benutzen um die Fälschungsangriffe zu verbessern.

Auch wird postuliert, dass die Verwendung von t und g statt s und f die Ergebnisse verbessern könnte, da g definiert ist um nur 80 nicht-Null Einträge zu haben (statt den 140 von f).

Eine NSS Variante

Für den EESS Standard (Efficient Embedded Security Standard) wurden Vorschläge gemacht um die Geschwindigkeit zu steigern.

Dazu hat man die Schlüssel neu generiert: $f = 1 + 3f_1 * f_2$ und $g = 1 + 2x + 3g_1 * g_2$ mit
$f_1 \in \mathfrak{L}(7,7), f_2 \in \mathfrak{L}(5,5), g_1 \in \mathfrak{L}(5,5), f_2 \in \mathfrak{L}(4,4)$

Auf den ersten Blick macht es die Vorberechnung von Wertverteilungen schwieriger. Jedoch: Bei einer Analyse von t und der Verwendung der L_2-Norm stellt man fest, dass die Ergebnisse noch schneller konvergieren.

Die traurige Wirklichkeit

Signaturen	Versuche	Durchschnittlicher Fehler
30’000	10	5.6
50’000	10	4.8
100’000	5	1.8
200’000	5	1.0

Gegenmassnahmen – 1

Nach der Veröffentlichung der Angriffe wurden Veränderungen am Verfahren vorgenommen.
Es scheint als würde das neue Schema den Angriffen standhalten.
Partielle Liste der Änderungen:
Generierung des öffenlichen Schlüssels: Statt $f = f_0 + pf_1$ und $g = g_0 + pg_1$ wo f_0 und g_0 öffentliche Parameter sind, werden nun $f = u + pf_1$ und $g = u + pg_1$ verwendet und u wird geheim gehalten.
Verifizierungskriterien: Es wird nicht mehr nur die Abweichung getestet.

- Normkondition: Es wird geprüft ob $|p^{-1}(s-m) \mod q| < B$ und $|p^{-1}(t-m) \mod q| < B$ mit B eine »centered norm«.

Gegenmassnahmen – 2

- Koeffizientenverteilungsüberprüfung: Eine Reihe von Tests über die Verteilung der Koeffizienten in s und t.
- »Moment Balancing«: Alternative Methoden zur Generierung von w_1 und w_2.

Das neue Verfahren macht das Fälschen schwieriger.

Auch sollte die Überprüfung der Norm wesentlich mächtiger sein als die ursprünglichen Abweichungstests.

Fazit

- Beweisbar sichere Kryptographie ist ’ne tolle Sache !
- NSS Designer sind ’ne Bastelgruppe.
- »Transcript« Attacken werden auf einem solchen Schema nie wirklich ausgeschlossen werden können.

Referenzen:

[1] Cryptanalysis of the NTRU Signature Scheme (NSS) from Eurocrypt 2001

[2] NTRU Dokumentation

[3] Google.com