UNIVERSITÀ DEGLI STUDI ROMA TRE
FACOLTÀ DI SCIENZE M.F.N.

Aspetti Matematici del Crittosistema NTRU

Sintesi della Tesi di Laurea in Matematica

di

Chiara Cocorullo

Relatore: Prof. Francesco Pappalardi

La parola "crittografia", che deriva dal greco $\kappa\rho\upsilon\pi\tau\acute{o}\varsigma$ = nascosto e $\gamma\varrho\acute{\alpha}\varphi\eta$ = scrittura, è usata, ai giorni nostri, per indicare una vera e propria disciplina che si occupa dello scambio di informazioni confidenziali rese difficilmente accessibili ad altri utenti. L'"arte" di scrivere messaggi segreti, che possano essere letti e compresi solo dal destinatario, era in realtà nota sin dall'antichità se si pensa che già la Bibbia parla di un codice segreto per scrivere il nome di Babele (**Codice Atbash**). Si hanno tracce di applicazioni di crittografia risalenti anche all'Antico Egitto: uno dei più antichi cifrari che si conosca è il **Codice di Cesare**, utilizzato dagli Imperatori Romani. Era piuttosto semplice, ma per i tempi efficace, e si basava sul comporre le parole traslando le lettere del messaggio originale di un numero di posizioni pari a n (chiave di cifratura): per $n = 3$, ad esempio, la parola "CESARE" diventa "FHVDUH". In questo modo, nonostante un "intruso" possa conoscere l'algoritmo su cui si basa il codice, per decifrare il messaggio deve conoscere la chiave giusta (nel nostro caso $n = 3$).

Per secoli la crittografia è stata di uso esclusivo dei militari e dei diplomatici e i metodi crittografici erano specifici per l'invio di messaggi affidati a corrieri.

Col passare del tempo, le tecniche sono diventate sempre più sofisticate e, nel XX secolo, con l'invenzione prima della radio e poi del computer, la necessità di comunicare informazioni in modo riservato si è ampliata notevolmente. Sono così nate nuove tecniche di crittografia: prima macchine cifranti come la celebre macchina "**Enigma**" usata dai tedeschi nella II Guerra Mondiale, poi, a partire dagli anni '70, con l'avvento del computer, sono stati sviluppati nuovi metodi specifici per l'uso informatico.

Al giorno d'oggi, l'utente del Bancomat, di una pay-tv e chi effettua acquisti su Internet con la Carta di Credito, fa uso, spesso senza rendersene conto, di tecniche crittografiche.

In questa tesi, dopo aver dato una panoramica sui più importanti algoritmi di crittografia attualmente conosciuti, abbiamo soffermato la nostra attenzione sull'analisi e sugli aspetti matematici che sono alla base dell'algoritmo NTRU.

Nel Capitolo 1, abbiamo analizzato i due tipi di algoritmi crittografici, quelli a chiave privata e quelli a chiave pubblica. Gli *algoritmi a chiave privata* usano una sola chiave per la cifratura e la decifratura, si basano su calcoli relativamente semplici e risultano veloci e, quindi, adatti per cifrare grandi moli di dati, ma hanno lo svantaggio di richiedere la distribuzione della chiave privata attraverso un canale sicuro. Ad esempio, i due utenti dovrebbero essersi scambiati la chiave in un incontro precedente. Il più noto algoritmo di cifratura a chiave pubblica è sicuramente il **DES** ([11]), che, per molti anni, è stato adottato come standard federale dal governo degli Stati Uniti ed è stato recentemente sostituito dal **Rijndael**, noto anche come **AES** ([3]). Gli *algoritmi a chiave pubblica* utilizzano invece due chiavi, una pubblica e una privata, create in modo tale che ciò che viene cifrato con una chiave possa essere decifrato solo con l'altra chiave della coppia e viceversa. Sono algoritmi molto astuti, facilmente calcolabili in un senso, ma difficili da risolvere in senso inverso, e sono in grado, in genere, di risolvere i due principali problemi legati all'invio di messaggi segreti: garantirne la segretezza e l'autenticità. L'idea alla base della crittografia a chiave pubblica fu introdotta da W. Diffie e M. Hellman nel 1976 in [4]. In seguito tre scienziati, R. Rivest, A. Shamir e L. Adleman, hanno ideato l'algoritmo **RSA** ([13]), che rappresenta il primo crittosistema che utilizza un metodo a chiave pubblica.

Questo algoritmo è utilizzato anche per il processo di *firma digitale*, che, analogamente alla firma autografa, garantisce l'autenticità del messaggio.

Per garantire l'identità dell'interlocutore, si ricorre invece ad una *Autorità di Certificazione*, un organismo ufficiale il cui compito è stabilire l'associazione tra firma digitale e soggetto che l'ha apposta.

Nel corso del capitolo abbiamo dato poi alcuni cenni sulla *crittoanalisi*, la scienza che si occupa di decifrare i codici segreti e, in generale, di "rompere" la sicurezza dei sistemi crittografici tramite tecniche di vario tipo.

Nel Capitolo 2 abbiamo analizzato il crittosistema a chiave pubblica NTRU, sviluppato da Jeffrey Hoffstein, Jill Pipher e Joseph H. Silverman e presentato per la prima volta alla conferenza Crypto '96 (vedi [19]).

Gli elementi di base usati dal crittosistema NTRU sono i polinomi appartenenti all'anello

$$\mathbf{R} = \frac{\mathbb{Z}[X]}{(X^N - 1)},$$

ovvero i *polinomi troncati* di grado $N-1$ con coefficienti interi. Inoltre i coefficienti dei polinomi sono ridotti modulo un intero q, per cui si considera l'anello

$$\mathbf{R}_{q,N} = \frac{(\mathbb{Z}/q\mathbb{Z})[X]}{(X^N - 1)}.$$

Un elemento $\boldsymbol{F} \in \mathbf{R}$ può essere scritto come un polinomio o un vettore:

$$\boldsymbol{F} = \sum_{i=0}^{N-1} F_i x^i = [F_0, F_1, ..., F_{N-1}].$$

Il simbolo * denoterà la moltiplicazione in $\mathbf{R}$, ovvero la convoluzione:

$$\boldsymbol{F} * \boldsymbol{G} = \boldsymbol{H}, \quad \text{con} \quad H_k = \sum_{i=0}^{k} F_i G_{k-i} + \sum_{i=k+1}^{N-1} F_i G_{N+k-i} = \sum_{i+j \equiv k \mod N} F_i G_j.$$

Oltre al numero N che specifica il grado dei polinomi troncati, un'implementazione completa del crittosistema NTRU richiede che l'utente stabilisca i moduli p e q in modo tale che $\gcd(p, q) = 1$ e q sia più grande di p. Inoltre deve considerare quattro sottoinsiemi di $\mathbf{R}$: l'insieme dei messaggi in chiaro $\mathcal{L}_m$, gli insiemi delle chiavi private $\mathcal{L}_f$ e $\mathcal{L}_g$ e l'insieme ausiliario $\mathcal{L}_r$.

In particolare, si prendono i polinomi in $\mathcal{L}_m$ in modo che abbiano coefficienti compresi tra $-p/2$ e $p/2$ e si considera l'insieme $\mathcal{L}(d_1, d_2)$ dei polinomi in $\mathbf{R}$ che hanno d_1 coefficienti uguali a 1, d_2 coefficienti uguali a -1 e gli altri uguali a 0. Si scelgono quindi tre parametri interi d_f, d_g e d_r e si pone:

$$\mathcal{L}_f = \mathcal{L}(d_f, d_f - 1) \quad \mathcal{L}_g = \mathcal{L}(d_g, d_g) \quad \mathcal{L}_r = \mathcal{L}(d_r, d_r).$$

Quindi, (N, q, p, d_f, d_g, d_r) rappresentano i vari parametri di NTRU. I valori tipici di questi parametri, utilizzati nelle applicazioni commerciali a vari livelli di sicurezza, sono esposti nella seguente tabella.

Livello di sicurezza	N	q	p	d_f	d_g	d_r
Medio	107	64	3	15	12	5
Alto	167	128	3	61	20	18
Massimo	503	256	3	216	72	55

Per meglio illustrare il procedimento di cifratura e decifratura di un messaggio tramite l'algoritmo NTRU, abbiamo immaginato l'interazione tra due utenti, Alice e Bernardo, che eseguono i seguenti passi:

- **Creazione della chiave pubblica**: Per creare la chiave pubblica, Bernardo sceglie due chiavi private $\boldsymbol{f} \in \mathcal{L}_f$ e $\boldsymbol{g} \in \mathcal{L}_g$. $\boldsymbol{f}$ deve inoltre essere scelto in modo che abbia gli inversi modulo q e modulo p. Egli calcola quindi l'inverso $\boldsymbol{f}_q$ di $\boldsymbol{f}$ in $\mathbf{R}_{q,N}$ e l'inverso $\boldsymbol{f}_p$ di $\boldsymbol{f}$ in $\mathbf{R}_{p,N}$. Infine calcola la chiave pubblica

  $$\boldsymbol{h} = p\boldsymbol{f}_q * \boldsymbol{g} \in \mathbf{R}_{q,N}.$$

- **Cifratura**: Alice rappresenta il suo messaggio in forma di un polinomio $\boldsymbol{m} \in \mathcal{L}_m$, sceglie un polinomio ausiliario $\boldsymbol{r} \in \mathcal{L}_r$ e utilizza la chiave pubblica $\boldsymbol{h}$ per calcolare il polinomio:

  $$\boldsymbol{e} = \boldsymbol{r} * \boldsymbol{h} + \boldsymbol{m} \in \mathbf{R}_{q,N},$$

  che rappresenta il messaggio cifrato che Alice spedisce a Bernardo.

- **Decifratura**: Bernardo riceve il messaggio cifrato $\boldsymbol{e}$ da Alice e calcola

  $$\boldsymbol{a} = \boldsymbol{f} * \boldsymbol{e} \in \mathbf{R}_{q,N},$$

dove i coefficienti di $\boldsymbol{a}$ sono rappresentati da interi compresi tra $-q/2$ e $q/2$. Poi calcola il polinomio $\boldsymbol{b}$, ottenuto riducendo ciascuno dei coefficienti di $\boldsymbol{a}$ modulo p, e usa l'altra sua chiave privata $\boldsymbol{f}_p$ per calcolare

$$\boldsymbol{c} = \boldsymbol{f}_p * \boldsymbol{b} \in \mathbf{R}_{p,N},$$

dove i coefficienti di $\boldsymbol{c}$ sono rappresentati da interi compresi tra $-p/2$ e $p/2$. $\boldsymbol{c}$ corrisponderà allora al messaggio originale $\boldsymbol{m}$ di Alice.

Il crittosistema funziona perché il polinomio $\boldsymbol{a}$ è ottenuto nel modo seguente:

$$\begin{aligned}
\boldsymbol{a} &= \boldsymbol{f} * \boldsymbol{e} \mod q \\
&= \boldsymbol{f} * (\boldsymbol{r} * \boldsymbol{h} + \boldsymbol{m}) \mod q && [\text{perché } \boldsymbol{e} = \boldsymbol{r} * \boldsymbol{h} + \boldsymbol{m} \mod q] \\
&= \boldsymbol{f} * (\boldsymbol{r} * p\boldsymbol{f}_q * \boldsymbol{g} + \boldsymbol{m}) \mod q && [\text{perché } \boldsymbol{h} = p\boldsymbol{f}_q * \boldsymbol{g} \mod q] \\
&= p\boldsymbol{r} * \boldsymbol{g} + \boldsymbol{f} * \boldsymbol{m} \mod q && [\text{perché } \boldsymbol{f} * \boldsymbol{f}_q = 1 \mod q]
\end{aligned}$$

Con una scelta appropriata dei parametri, come quella da noi eseguita, si garantisce che i coefficienti del polinomio $p\boldsymbol{r} * \boldsymbol{g} + \boldsymbol{f} * \boldsymbol{m}$ sono compresi tra $-q/2$ e $q/2$, cosicché ridurre i coefficienti modulo q non avrà alcun effetto. Quindi, quando Bernardo calcola $\boldsymbol{a}$ prima moltiplicando $\boldsymbol{f} * \boldsymbol{e}$ e poi riducendo i coefficienti modulo q nell'intervallo da $-q/2$ a $q/2$, il polinomio che ottiene è esattamente uguale al polinomio $p\boldsymbol{r} * \boldsymbol{g} + \boldsymbol{f} * \boldsymbol{m}$. Perciò, quando riduce $\boldsymbol{a}$ mod p, trova il polinomio $\boldsymbol{f} * \boldsymbol{m}$, che, moltiplicato per $\boldsymbol{f}_p$, restituisce proprio il messaggio originale $\boldsymbol{m} \mod p$.

Nell'ultima parte del capitolo, abbiamo considerato possibili variazioni all'implementazione di NTRU. Ad esempio, si può prendere $\boldsymbol{p} = 2 + X$ e $\boldsymbol{f} = 1 + \boldsymbol{p} * \boldsymbol{F}$, dove $\boldsymbol{F}$ è un polinomio con d_F coefficienti non nulli oppure è il prodotto di più polinomi con un numero piccolo di coefficienti diversi da zero (*polinomi a basso peso di Hamming*).

Il crittosistema NTRU risulta quindi veloce ed efficiente nella cifratura e decifratura ed è realizzabile anche su processori di scarsa potenza e dotati di poca memoria. Inoltre la generazione delle coppie di chiavi pubblica/privata è un'operazione molto veloce e semplice. Per questi motivi si ritiene adatto in molte situazioni, tra le quali: smart card, telefonia cellulare, pay-tv, commercio elettronico, distribuzione di musica e video digitali, autenticazione.

Nel Capitolo 3, oltre all'analisi di un algoritmo di inversione dei polinomi troncati ideato da R. Schroeppel, S. O'Malley, H. Orman, O. Spatscheck in [26], abbiamo scritto un codice in PARI ([43]) dell'algoritmo NTRU ed abbiamo esposto i risultati, ai vari livelli di sicurezza, della nostra implementazione su un Pentium II 350 MHz, che opera con sistema operativo Windows.

Nel Capitolo 4, abbiamo introdotto la teoria dei reticoli, che è alla base di un attacco contro NTRU. Abbiamo perciò considerato la seguente definizione:

Definizione 4.2.1. *Dati n vettori linearmente indipendenti* $\boldsymbol{b}_1, \boldsymbol{b}_2, ..., \boldsymbol{b}_n \in \mathbb{R}^m$*, il* reticolo $L = L(\boldsymbol{b}_1, \boldsymbol{b}_2, ..., \boldsymbol{b}_n)$ *di dimensione n è l'insieme delle combinazioni lineari* $a_1\boldsymbol{b}_1 + ... + a_n\boldsymbol{b}_n$ *degli elementi* $\boldsymbol{b}_1, \boldsymbol{b}_2, ..., \boldsymbol{b}_n$*, con* $a_1, a_2, ..., a_n \in \mathbb{Z}$*, ovvero:*

$$L(\boldsymbol{b}_1, \boldsymbol{b}_2, ..., \boldsymbol{b}_n) := \sum_{i=1}^{n} \boldsymbol{b}_i\mathbb{Z} = \left\{ \sum_{i=1}^{n} a_i\boldsymbol{b}_i \mid a_i \in \mathbb{Z} \right\}.$$

L'insieme $B = \{\boldsymbol{b}_1, \boldsymbol{b}_2, ..., \boldsymbol{b}_n\}$ è chiamato *base* del reticolo e può essere rappresentato in maniera compatta con una matrice $m \times n$ le cui colonne sono date dai vettori $\boldsymbol{b}_1, \boldsymbol{b}_2, ..., \boldsymbol{b}_n$.

Una volta definito il *determinante* $\det(L) = \sqrt{det\left(\langle \boldsymbol{b}_i, \boldsymbol{b}_j \rangle\right)_{i,j=1}^{n}}$ e l'invariante primo *minimo successivo* nella norma euclidea

$$\lambda_1(L) = min\left\{\|\boldsymbol{v}\| \mid \boldsymbol{v} \in L \setminus \{0\}\right\},$$

siamo arrivati alla seguente disuguaglianza, detta anche *euristica gaussiana*, soddisfatta dal vettore minimo di un reticolo L di dimensione n:

$$\sqrt{\frac{n}{2\pi e}}(det(L))^{\frac{1}{n}} \leq \lambda_1 \leq \sqrt{\frac{n}{\pi e}}(det(L))^{\frac{1}{n}}.$$

Abbiamo poi considerato uno dei più importanti problemi sui reticoli, SVP (*Shortest Vector Problem*), che consiste nel trovare un vettore *minimo* in L, ovvero un vettore $\boldsymbol{v} \in L \setminus \{0\}$ tale che $\|\boldsymbol{v}\| \leq \|\boldsymbol{w}\|, \forall \boldsymbol{w} \in L \setminus \{0\}$.

Per trovare vettori minimi si utilizzano i cosiddetti metodi di *riduzione del reticolo*. Nel caso bidimensionale abbiamo analizzato un metodo, descritto da Gauss per la norma l_2 e generalizzato da Kaib e Schnorr per qualsiasi norma ([33]), con il quale è possibile trovare un vettore minimo calcolando una base "ridotta" (Cap. 4.3).

Abbiamo inoltre analizzato il metodo basato sul famoso algoritmo LLL, ideato da A. Lenstra, H. Lenstra e L. Lovasz in [34] e ritenuto uno dei migliori attualmente conosciuti.

In particolare, abbiamo considerato la base ortogonalizzata di Gram-Schmidt $B^* = \{\boldsymbol{b}_1^*, \boldsymbol{b}_2^*, ..., \boldsymbol{b}_n^*\}$, che generalmente non è una base per il reticolo $L(B)$ (Cap. 4.2), e definito le proiezioni ortogonali π_i da $\mathbb{R}^m$ in $\sum_{j \geq i} \mathbb{R}\boldsymbol{b}_j^*$:

$$\pi_i(\boldsymbol{x}) = \sum_{j=i}^{n} \frac{\langle \boldsymbol{x}, \boldsymbol{b}_j^* \rangle}{\langle \boldsymbol{b}_j^*, \boldsymbol{b}_j^* \rangle} \boldsymbol{b}_j^*.$$

Abbiamo poi considerato la seguente definizione:

Definizione 4.3.4. *Una base $B = \{\boldsymbol{b}_1, \boldsymbol{b}_2, ..., \boldsymbol{b}_n\} \in \mathbb{R}^{m \times n}$ è una base LLL-ridotta rispetto al parametro δ se:*

- $|\mu_{i,j}| \leq \frac{1}{2}$ *per ogni* $i > j$ (lunghezza ridotta)
- *Per ogni coppia di vettori consecutivi $\boldsymbol{b}_i, \boldsymbol{b}_{i+1}$, abbiamo*
$$\delta \left\| \pi_i(\boldsymbol{b}_i) \right\|^2 \leq \left\| \pi_i(\boldsymbol{b}_{i+1}) \right\|^2.$$

Abbiamo dimostrato il seguente risultato:

Corollario 4.3.1 *Sia $B = \{\boldsymbol{b}_1, \boldsymbol{b}_2, ..., \boldsymbol{b}_n\}$ una base LLL-ridotta con parametro δ. Allora, per $\alpha = \frac{1}{\delta - \frac{1}{4}}$, si ha:*

$$\| \boldsymbol{b}_1 \| \leq \alpha^{\frac{n-1}{2}} \lambda_1(L).$$

Abbiamo visto quindi come l'algoritmo LLL trasformi, per un dato δ, una base intera di un reticolo in una base per lo stesso reticolo che è LLL-ridotta con parametro δ. L'algoritmo procede in questo modo:

1. `Esegui il passo di riduzione (`*`reduction step`*`).`
2. `Se per qualche coppia di vettori consecutivi non vale la seconda proprietà di una base` LLL`-ridotta, allora scambiali (`*`swap step`*`).`
3. `Se è stato eseguito uno scambio, ritorna al passo 1.`

L'algoritmo di riduzione LLL è quindi un algoritmo di approssimazione che non trova necessariamente il vettore minimo nel reticolo, ma calcola un vettore che è al più γ volte la lunghezza del vettore minimo, dove γ è il fattore di approssimazione (nel nostro caso $\gamma = \alpha^{\frac{n-1}{2}}$). Abbiamo inoltre dimostrato che l'agoritmo LLL ha tempo di esecuzione polinomiale in $D = \prod_{k=1}^{n} (\det L(\boldsymbol{b}_1, ..., \boldsymbol{b}_k))^2$.

Nel Capitolo 5 abbiamo accennato ai vari attacchi e tentativi di "rottura" del crittosistema NTRU, soffermandoci sull'attacco a reticolo e definendo il *reticolo NTRU*, ovvero:

Definizione 5.2.1 *Il* reticolo standard NTRU L^{NT} *è il reticolo di dimensione* $2N$ *generato dalle righe della matrice* $2N \times 2N$*:*

$$L^{NT} = \left(\begin{array}{cccc|cccc} \lambda & 0 & \dots & 0 & h_0 & h_1 & \dots & h_{N-1} \\ 0 & \lambda & \dots & 0 & h_{N-1} & h_0 & \dots & h_{N-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda & h_1 & h_2 & \dots & h_0 \\ \hline 0 & 0 & \dots & 0 & q & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & 0 & q & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & 0 & \dots & \dots & q \end{array}\right)$$

dove $(h_0, ..., h_{N-1})$ *sono i coefficienti della chiave pubblica* $\boldsymbol{h} = \boldsymbol{f}^{-1} * \boldsymbol{g} \mod q$ *e la costante* λ *è una costante di bilanciamento scelta per massimizzare l'efficienza della ricerca dei vettori piccoli nel reticolo.*

Il crittoanalista conosce il vettore pubblico $\boldsymbol{h}$, e quindi conosce il reticolo L^{NT}, e il suo scopo è ricavare da L^{NT} il vettore non noto $\tau = (\lambda \boldsymbol{f}, \boldsymbol{g}) \in L^{NT}$, la cui lunghezza è data da $|\tau|_2 = \sqrt{\lambda^2 |\boldsymbol{f}|_2^2 + |\boldsymbol{g}|_2^2}$.

Abbiamo considerato quindi un altro parametro c_h, dato dal rapporto tra la lunghezza del vettore τ in L e la lunghezza prevista del vettore minimo (non nullo). Un'implementazione di un algoritmo di riduzione a reticolo avrà le migliori probabilità di individuare τ, o un altro vettore la cui lunghezza è vicina a quella di τ, se il crittoanalista sceglie la costante di bilanciamento λ tale che c_h sia più piccolo possibile.

Abbiamo dimostrato quindi che la scelta ottimale per la costante di bilanciamento è $\lambda = \frac{|\boldsymbol{g}|_2}{|\boldsymbol{f}|_2}$, dalla quale si ricava:

$$c_h = \sqrt{\frac{2\pi e \, |\boldsymbol{f}|_2 \cdot |\boldsymbol{g}|_2}{Nq}}.$$

Abbiamo poi considerato l'analisi della sicurezza riportata in [19], dalla quale sono state ricavate le seguenti ipotesi per il tempo necessario per rompere una coppia di chiavi ai differenti livelli di sicurezza di NTRU:

Sicurezza	q	c_h	N	Tempo (secondi)
Media	64	0.26	107	780.230 (9 giorni)
Alta	128	0.23	167	$1.198 \cdot 10^{10}$ (380 anni)
Massima	256	0.18	503	$1.969 \cdot 10^{35}$ ($6.2 \cdot 10^{27}$ anni)

Inoltre abbiamo preso in esame il confronto tra NTRU e alcuni dei più noti algoritmi a chiave pubblica, come RSA, ElGamal e ECC (curve ellittiche), effettuato dagli stessi autori del crittosistema. In particolare sono state paragonate le velocità di NTRU e RSA, prendendo in considerazione dimensioni della chiave che offrano livelli di sicurezza simili. La seguente tabella mostra il risultato del confronto tra NTRU e RSA nella velocità di generazione delle chiavi, criptazione e decriptazione di messaggi brevi.

	NTRU 167 / RSA 512	**NTRU 263 / RSA 1024**
Cifratura	2.4	3.9
Decifratura	16.8	74.4
Creazione chiave	55.3	146.0

Si ricava quindi che NTRU 167 è 2.4 volte più veloce di RSA 512 nella criptazione, 16.8 volte nella decriptazione e 55.3 volte nella creazione della chiave.

La seguente tabella, infine, mette a confronto il tempo (in millisecondi) richiesto per creare una singola chiave pubblica/privata con NTRU e RSA a vari livelli di sicurezza. Ricordiamo che NTRU 167 e RSA 512 hanno livelli di sicurezza confrontabili, così come NTRU 263 e RSA 1024.

NTRU (300MHz Pentium III)	Creaz. chiave (ms)
NTRU 107	1.5
NTRU 167	4.0
NTRU 263	7.5
NTRU 503	17.5
RSA (255 MHz Digital AlphaStation)	Creaz. chiave (ms)
RSA 512	260.0
RSA 768	590.0
RSA 1024	1280.0

Si ha quindi un incredibile vantaggio di velocità nella creazione della chiave utilizzando NTRU rispetto a RSA a livelli di sicurezza confrontabili. Per questo e altri motivi, secondo gli autori del crittosistema, "NTRU occupa una posizione unica nel mercato crittografico attuale e gioca un ruolo fondamentale nel mondo commerciale mondiale per la capacità di fornire comunicazioni elettroniche sicure".

Bibliografia

[Estratto della bibliografia]

[3] J. Daemen, V. Rijmen, *Rijndael, the Advanced Encryption Standard*, Dr. Dobb's Journal , vol. 26, No. 3, pp. 137-139, March 2001.

[4] W. Diffie e M. Hellman, *New Directions in Cryptography*, IEEE Trans. on Inf. Theory, vol. 22, pp. 644-654, Nov. 1976.

[11] National Bureau of Standards, *Data Encryption Standard (DES)*, FIPS 46, Washington, DC, Apr. 1977.

[13] R. L. Rivest, A. Shamir e L. Adleman, *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*, Comm. ACM, vol. 21, pp. 120-126, Feb. 1978.

[19] J. Hoffstein, J. Pipher, J.H. Silverman, *NTRU: A new high speed public key cryptosystem*, in Algorithmic Number Theory (ANTS III), Portland, OR, June 1998, Lecture Notes in Computer Science 1423 (J.P. Buhler, ed.), Springer-Verlag, Berlin, 1998, pp. 267-288.

[26] R. Schroeppel, S. O'Malley, H. Orman, O. Spatscheck, *Fast Key exchange with elliptic curve systems*, Advances in Cryptology-CRYPTO 95, Lecture Notes in Computer Science 973, D. Coppersmith, ed., Springer-Verlag, New York, 1995, pp. 43-56.

[33] M. Kaib, C.P. Schnorr, *The generalized Gauss reduction algorithm*, Journal of Algorithms, Band 21, Nr. 3, pp. 565-578, Nov. 1996.

[34] A.K. Lenstra, H.W. Lenstra, L. Lovasz *Factoring polynomials with polynomial coefficients*, Math. Annalen 261, pp. 512-534, 1982.

[43] C. Batut, K. Belabas, D. Bernardi, H. Cohen, M. Olivier, 'Pari-GP Versione 2.0.17 (beta)', **http://www.parigp-home.de**, 2001.