سیستم های عدد مانده ای در رمزنگاری: مروری بر کاربردهای امنیتی مدرن

چکیده :

در سال های اخیر، سیستم عدد مانده توجه جامعه علمی را به خود جلب کرده و به یک گزینه جالب توجه در حوزه پیاده سازی سخت افزارهای امنیتی تبدیل شده است. در این مطالعه مروری، به بعضی کاربردهای مدرن و غیرمتداول RNS در حوزه رمزنگاری پساکوانتومی ، زیرساخت های ابری و رمزنگاری هومورفیک پرداخته می شود. ما به بررسی روش هایی که از عدد مانده در این رویکردها استفاده می کنند همچنین راهکارهای مکانیزه کردن راه حل های ابری RNS ایمن و مقاوم می پردازیم. این مطالعه می تواند مقدمه ای بر اعداد مانده تلقی شده و به جهت گیری مطالعات آتی و مسائل بازی که می توانند توسط RNS به طور موثر حل شوند کمک کند.

تعریف مساله و هدف اصلی مقاله:

- **سوال اصلی مطرح شده در مقاله چیست؟**

این مقاله به صورت Survey است. کاربردهای RNS برای رمزنگاری و بالا بردن امنیت در بحث Cloud و IoT مطرح میشود.

- **چه مشکلاتی باید بر طرف شود؟**

- همه محاسبات در نمایش RNS کارامد نیستند. بعنوان مثال، مقایسه اعداد RNS نیازمند تبدیل به نمایش وزن دار هستند که همانطور که در CRT می بینیم بسیار پرهزینه هستند. بنابراین، هنگام طراحی یک

RNS کامل باید از انشعاب های شرطی که نیازمند مقایسه هستند اجتناب شود. مشکلات مشابهی در عملیات تشخیص علامت و تقسیم بوجود می آید[16،74]. تنها استثنا، تقسیم عدد صحیحی است که در آن بتوان نوشت $a/b = a.b^{-1}$ و بنابراین اگر بتوان b^{-1} را در RNS بصورت معکوس های ضربی $b^{-1} \bmod m_i$ نوشت آنگاه تقسیم دقیق به ضرب RNS می رسد. این ویژگی مفید در رمزنگاری کلید عمومی RNS بکار می رود(بخش 3).

بعلاوه، تبدیل های ورودی و خروجی برای حفظ سازگاری در سیستم های غیر RNS و کاربردهای RNS ، اجتناب ناپذیر هستند بعلت ماهیت ترتیبی شان، توصیه می شود از تبدیل های پیوسته به / از قالب RNS اجتناب شود. از دیدگاه مهندسی، این نیازمند طراحی سیستم RNS است که در آن تنها می توان یک تبدیل ورودی در شروع پردازش انجام داد، سپس کاربرد را در نمایش RNS کامل نشان داده و در نهایت تنها هنگام مواجه به کاربردهای غیر RNS به نمایش وزن دار تبدیل کرد. در مواردی که استفاده از تبدیل ها اجتناب ناپذیر است.طراح باید بررسی کند آیا استفاده از RNS به صرفه است یا خیر و درباره بهترین شکل ممکن، تعداد و طول کلمه ماژول تصمیم گیری درستی داشته باشد تا عملکرد بهینه شود[61]. نکته کلیدی این است که اگر کاربرد اصلی را نتوان به شکل موازی تبدیل کرد و از بخش های ترتیبی زیادی استفاده شود احتمالا کاندید مناسبی برای کاربرد RNS نخواهد بود.

- چالش اصلی در رمزنگاری مبتنی بر RNS اینست که آیا می توان عملیات AB (mod N) را با تمام عملوندها در قالب RNS انجام داده و تضمین کرد که خروجی مدول N کاهش یافته باشد اما هنوز در نمایش RNS باشد یا خیر.

- چه ضرورتی برای مطرح شدن مسله است؟

پیشرفت روز افزون و مطرح شدن مباحث جدید مثل IoT، SDN، Cloud و ... نیاز به راه کارهایی برای افزایش امنیت سرویس های جدید میباشد. استفاده از RNS در بحث رمزنگاری خلا های امنیتی این سرویس های جدید را پوشش میدهد.

- چه روش هایی قبلا برای این کار انجام شده است؟

از اواسط دهه 1970، تکنولوژی و تئوری به آهسنگی شروع به همگرایی کردند. بیش از 150 مقاله از اواسط دهه 1970 تا اواسط دهه 1980 منتشر شد در حالی که اولین ثبت ها و کتاب ها درباره RNS نیز در این دوره ارائه شده اند. اساسا، کاربرد اصلی RNS پردازش سیگنال دیجیتال (DSP) بوده است. هوانگ یک فیلتر RNS دوبعدی که قادر به عملیات M 20 بر ثانیه بود را ساخت و تست کرد[31]، اسمیت در مارتین ماریتا یک FFT پرسرعت ارائه داد[69]. جولین یک فیلتر شانه ای RNS در [35] و تیلور سیستم های RNS با سخت افزار VLSI را پیشنهاد داده اند.

تا امروز، مطالعات متعددی روی کاربردهای RNS DSP انجام شده است. در دهه 1990، تیلور و همکاران یک پردازنده DSP با محتوای RNS به نام "ماشین گاوس" ارائه دادند[72]. همچنین کلائودو و همکاران ماژول های ریاضی RNS ترکیبی سریعی برای DSP پیشنهاد دادند[20]. در 2001 ، رامیرز و همکاران یک پیاده سازی RNS برای تبدیل موجک گسسته (DWT) ارائه کردند[55]. بهبودهای انجام شده روی RNS در [39،42،50،51،67] مرور شده اند.

رمزنگاری در دهه های 1970 و 1980 از این روند پیروی نکرد چرا که در این دوره نیاز به ارتباطات ایمن جدی نبود و تکنولوژی معطوف عملیات روی داده هایی با طول کلمه بسیار طولانی که در رمزنگاری کلید عمومی مورد نیاز بودند، شده بود[58]. این موضوع باعث دلسردی محققان نسبت به استفاده از RNS برای

رمزنگاری شد تا اینکه در اواسط دهه 1990 مطالعه پاش[1] و پاش[54، 53] یک زمینه تحقیقاتی جدید را بوجود آورد. مطالعه ایشان، هرچند صراحتا به رمزنگاری اشاره ای نداشت، روشی برای استفاده از RNS در ضرب ماژولار فراهم ساخت و همانطور که بعدا خواهیم دید این ضرب پرمصرف‌ترین عملیات برحسب زمان و حافظه است و تا حد زیادی تعیین کننده عملکرد کلی سیستم رمزنگاری است. از اواسط دهه 2000 ، تحقیقات زیادی در زمینه ضرب ماژولار RNS و طراحی سیستم رمزنگاری کارامد انجام شده؛ اولین استفاده از RNS در رمزنگاری منحنی بیضوی(ECC) در [62] ارائه شده است و همچنین RNS کاربرد گسترده ای در رمزنگاری RSA داشته است[10،22،23،36،48،61].

در نهایت، مطالعات به مضرب هاب ماژولی مونتکومری RNS که در توسعه سیستم های رمزنگاری مبتنی بر RNS و استفاده از آن ها بعنوان یک گزینه طراحی قابل قبول نقش داشت پرداختند[6،60]. اخیرا، کاربردهای جدید RNS در زمینه های شبکه، SDN و محاسبات ابری ارائه شده اند[14،15،40].

این سیستم حسابی باستانی دنیایی از ایده های جدید برای محققان به ارمغان آورده است. در ادامه، به تجزیه و تحلیل چالش های طراحی RNS در رمزنگاری کلیدی عمومی و کاربردهای جدید RNS در حوزه های معماری ابر، هومومورفیک(هم ریختی) و رمزنگاری پسا کوانتومی(رمزنگاری مبتنی لاتیس) می پردازیم.

[1] posch

- **روش های پیشنهادی ارایه شده:**
- استفاده از RNS در رمزنگاری ECC و RSA
- رمزنگاری مبتنی بر لاتیس
- حالت رمزنگاری هوموروفیک کامل
- کاربردهای RNS در EHE
- استفاده از RNS در امنیت Cloud و IOT

توضیح راه حل پیشنهادی:

(با توجه به خلاصه سازی و حذف بخشی از متن [بخش 1 و 2]، **شماره گذاری ها** جهت ارجاع راحت به مقاله اصلی، تغییر نکرده است)

1- RNS در رمزنگاری کلید عمومی

1-3- ضرب ماژولی منت کومری

عملیات اصلی در رمزنگاری کلید عمومی، ضرب ماژولی $A.B \bmod N$ است. بعنوان مثال، توان ماژولار RSA اساسا یک سری مدول ضرب ماژولی است در حالی که در ECC جمع ها و ضرب های ماژولار متوالی برای عملیات جمع دوتا دوتا و جمع نقطه ای بکار می روند[12]. بنابراین چالش اصلی در رمزنگاری مبتنی بر RNS اینست که آیا می توان عملیات $AB \pmod N$ را با تمام عملوندها در قالب RNS انجام داده و تضمین کرد که خروجی مدول N کاهش یافته باشد اما هنوز در نمایش RNS باشد یا خیر.

مسئله پیچیده تر از آن چیزی است که به نظر می رسد و نیازمند پیشرفت در حوزه های مختلف ریاضیات و حسابان است. در سال 1985، ریاضیدان پی . ال . منت کومری یک روش غیرمعمول برای ضرب ماژولار کارامد بدون تقسیم های متوالی ارائه داد[45]. این دقیقا نقطه ای بود که RNS را قادر به استفاده از ضرب منت کومری کرد چرا که تقسیم در RNS ناکارامد است. اولین تغییر الگوریتم منت کومری برای سازگاری داده ها در قالب RNS 10 سال بعد توسط کارل سی پاش و رینالد پاش [54] ارائه شد. سپس، پیاده سازی ها و الگوریتم های مقاوم تری عمدتا برای استفاده در زمینه توان ماژولار برای RSA پیشنهاد شدند[10،22،36]. ضرب مدول منت کومری اصلی(MMM) در الگوریتم 3-1 نشان داده شده است.

Algorithm 3.1: Montgomery Modular Multiplication

Input: $A, B, N, C, (-N)^{-1}$ such that $A, B < N$
Output: $W \equiv ABC^{-1} \mod N$, such that $W < 2N$

```
1 begin
2     S ← A · B
3     T ← S · (−N^{-1}) mod C
4     U ← T · N
5     V ← S + U
6     W ← V/C
7     return W
8 end
```

اول از همه، توجه شود که الگوریتم ضرب AB mod N را دقیقا محاسبه نمی کند درعوض ضرب $W \equiv ABC^{-1}$ را بدست می آورد. C یک ثابت به نام مبنای منت کومری است و معمولا بصورت 2^n انتخاب می شود بنابراین عملیات C مدول به شیفت های ساده کاهش می یابد[45]. همچنین توجه شود که نتیجه W کاملا کاهش نیافته است یعنی کافی است W<2N و نه W<N. در حقیقت، الگوریتم نیازمند یک تفریق W-N است که آن هم وابسته به اینست که آیا $W \geq N$ یا خیر. در کاربردهایی با ضرب های منت کومری متوالی (مانند رمزنگاری) این گام می تواند حذف شود و تنها یکبار در پایان پردازش انجام شود. این ویژگی برای نسخه RNS

الگوریتم بسیار متداول است چرا که توضیح دادیم باید از مقایسه ها و انشعاب های شرطی هنگام کار با RNS اجتناب شود.

Algorithm 3.2: RNS Montgomery Modular Multiplication

Input: $\vec{A}, \vec{A'}, \vec{B}, \vec{B'}$ such that $A, B < 2N$
Output: $\vec{W}, \vec{W'}$, such that $W < 2N$ and $W \equiv ABM^{-1} \mod N$
Pre-compute: $\left(-\vec{N}^{-1}\right), (\vec{M'^{-1}})', \vec{N}$

begin
  $(\vec{S}, \vec{S'}) \leftarrow (\vec{A} \cdot \vec{B}, \vec{A'} \cdot \vec{B'})$ // Computations in both bases
  $\vec{T} \leftarrow \vec{S} \cdot \left(-\vec{N}^{-1}\right)$ // Already computed mod M
  $\vec{T'} \leftarrow \vec{T}$ // 1st conversion from base $\mathcal{B}$ to $\mathcal{B'}$
  $\vec{U'} \leftarrow \vec{T'} \cdot \vec{N'}$
  $\vec{V'} \leftarrow \vec{S'} + \vec{U'}$
  $\vec{W'} \leftarrow \vec{V'} \cdot (\vec{M^{-1}})'$ // $(\vec{M^{-1}})'$ exists in base M'
  $\vec{W} \leftarrow \vec{W'}$ // 2nd conversion from base $\mathcal{B'}$ to $\mathcal{B}$
  return $\vec{W}$ and $\vec{W'}$
end

بعلاوه، الگوریتم ابتدا باید عملوندهای ورودی را به نمایش های منت کومری مربوطه تبدیل کند[45]. با فرض عدد صحیح A ، نمایش منت کومری آن بصورت $\overline{\mathrm{A}} = \mathrm{AC} \bmod \mathrm{N}$ تعریف می شود. این نسخه می تواند به وسیله یک ضرب منت کومری اضافی C^2 یعنی $\overline{\mathrm{A}} = \mathrm{A} \times (\mathrm{C}^2 \bmod \mathrm{N}) \times \mathrm{C}^{-1} \bmod \mathrm{N} = \mathrm{AC} \bmod \mathrm{N}$ انجام شود. با این ورودی ها، الگوریتم مانده منت کومری نتیجه $\overline{\mathrm{W}} = \mathrm{WC} \bmod \mathrm{N} = \mathrm{ABC} \bmod \mathrm{N}$ را برمی گرداند. با یک ضرب منت کومری اضافی با ورودی ها نتیجه $\overline{\mathrm{W}} = \mathrm{WC} \bmod \mathrm{N}$ و $1 \bmod \mathrm{N}$ ، مبنای منت کومری حذف شده

و نتیجه دقیق $WC \times 1 \times C^{-1}$ بدست می آید. خوشبختانه، این تبدیل ها تنها یکبار در شروع و در انتهای الگوریتم اجرا می شوند حتی اگر الگوریتم مکررا برای مکانیزه کردن ضرب ماژولی به کار رود.

MMM نسبتا ساده است: گام های 2، 4 و 5 ضرب ها و جمع های ساده هستند بنابراین می توانند مستقیما به محاسبات RNS نگاشت شوند. اما، گام های 3 و 6 مشکلاتی ایجاد می کنند؛ گام 3 یک عملات C مدول است بنابراین باید روشی برای محاسبه کاهش مدول در نمایش RNS در نظر گرفته شود. مشکلات مشابهی برای گام 6 بروز می کند که در آن تقسیم با مبنای منت کومری باید در RNS انجام شود.

ترفند غلبه بر این مشکلات انتخاب یک مبنای منت کومری مناسب و به ویژه تنظیم C=M است که در آن M دامنه RNS پایه $B = \{m_1, m_2, \dots, m_L\}$ است. بدین ترتیب، گام 3 الگوریتم 3-1 به گام محاسبه مدول M تبدیل می شود. متاسفانه، محاسبات در پایه مشابه ادامه نمی یابد چرا که در گام 6 کمیتی به شکل $M^{-1} \bmod M$ باید محاسبه شود که به لحاظ ریاضی وجود ندراد. جواب انتخاب یک پایه دوم $B' = \{m'_1, m'_2, \dots, m'_L\}$ است بطوریکه یک تبدیل پایه از پایه B به پایه B' بعد از گام 3 الگوریتم انجام شود. سپس گام های 4، 5 و 6 در پایه جدید B' محاسبه می شوند چرا که حالا دیگر محاسبه $M^{-1} \bmod M$ در گام 6 امکان پذیر است. الگوریتم ضرب ماژولار منت کومری RNS کامل (RNSMMM) در الگوریتم 3-2 ارائه شده است. یک BC اضافی در پایان الگوریتم بکار می رود تا ورودی ها و خروجی ها با یکدیگر سازگار شوند و الگوریتم بتواند مکررا در هر الگوریتم توان(نما) ماژولار اجرا شود[22،23،36،48].

واضح است، تمام گام های RNSMMM را می توان بصورت موازی در RNS جدای از 2 BC ها انجام داد؛ بنابراین این گام ها عملکرد کلی و پیچیدگی الگوریتم را تعیین می کنند. اعتبار الگوریتم تضمین می شود اگر $\gcd(N, M) = 1$ و $\gcd(M, M^{-1}) = 1$، بطوریکه $-N^{-1} \bmod M$ و $M^{-1} \bmod M'$ (step 6) وجود داشته باشند.

Algorithm 3.3: Base Conversion algorithm

Input: $\vec{T} = \{t_1, t_2, \ldots, t_L\}, \mathcal{B}, \mathcal{B}', \alpha$
Output: $\vec{T}' = \{t'_1, t'_2, \ldots, t'_L\}$
Pre-compute: $\langle M_i^{-1} \rangle_{m_i}, \langle M_i \rangle_{m'_i}, \langle -M \rangle_{m'_i}, \forall i$

begin
  $\xi_i = \left\langle t_i \cdot M_i^{-1} \right\rangle_{m_i}$ // $\forall i$
  $\sigma_0 = \alpha, y_{i,0} = 0$ // $\forall i$
  for $j = 1, \ldots, n$ **do**
    $\sigma_j = \sigma_{j-1} + \mathtt{trunc}(\xi_j)/2^r$
    $k_j = \lfloor \sigma_j \rfloor$ // $k_j \in \{0, 1\}$
    $\sigma_j = \sigma_j - k_j$
    $y_{i,j} = y_{i,(j-1)} + \xi_j \cdot \langle M_j \rangle_{m'_i} + k_j \cdot \langle -M \rangle_{m'_i}$ // $\forall i$
  end
  return $t'_i = \langle y_{i,n} \rangle_{m'_i}$ // $\forall i$
end

3-2-استفاده از RNS در رمزنگاری ECC و RNS

ضرب مونت کومری RNS امکان استفاده از RNS در رمزنگاری کلید عمومی را فراهم می سازد. اما حقیقت اینست که تنها تعداد انگشت شماری گزینه در مطالعات وجود دارد که عمدتا با هدف ساده سازی عملیات BC طاقت فرسا انجام شده اند. در این مطالعه دو دسته یعنی BC مبتنی بر CRT و BC مبتنی MRC در نظر گرفته می شوند.

تبدیل پایه مطرح شده در [36] در الگوریتم 3-3 ارائه شده است. اساسا، الگوریتم نسخه تغییریافته ای از CRT در رابطه 6 را اجرا می کند. گام 2، بعنوان مثال، ضرب های داخلی ξ_i در پایه اول را محاسبه کرده و حلقه for مابقی محاسبات رابطه 6 را انجام می دهد تا نتیجه را به پایه جدید کاهش دهد(محاسبات m'_i مدول). توجه شود عامل تصحیح k مربوط به CRT به روشی بیت به بیت محاسبه می شود[36]. هر دو دسته مشخصات مشترکی

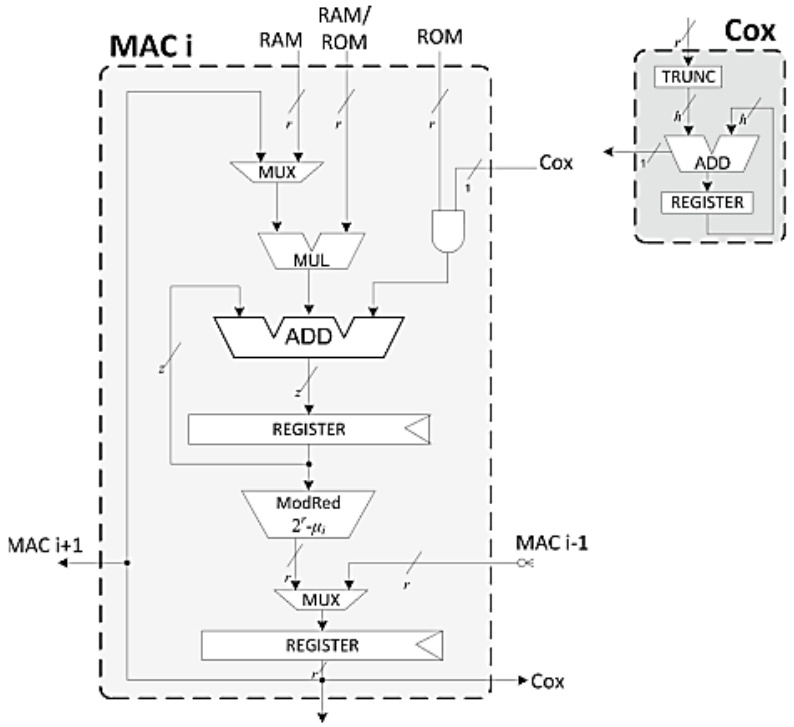
دارند؛ بعنوان مثال، تمام محاسبات به عملیات ضرب-جمع ماژولی (MAC) کاهش یافته اند(یعنی ضرب ارقام RNS کوچک و سپس یک جمع و یک ضرب با نماش RNS مربوطه کاهش می یابد؛ خروجی مکررا به ورودی داده می شود تا فرآیند تکرار شود).مثالی از معماری سخت افزاری MAC مناسب معماری است که از الگوریتم BC (الگوریتم 3-3) پشتیبانی کند(شکل 3)[36]. هر واحد MAC متشکل از یک ضرب کننده، یک جکع کننده و یک واحد کاهش ماژولی در هر ماژول پایه RNS (فرم خاصی از $2^r - \mu_i$) است. سایر واحدهای MAC زمانی فعال می شوند که جمع تمام نتایج RNS مورد نیاز باشد(بعنوان مثال در حالت تبدیل پایه) [36]. واحد Cox روشی بسیار ساده برای تخمین عامل تصحیح k در مراحل 5 تا 7 الگوریتم 3-3 فراهم می سازد و مربوط به موارد خاصی در [36] است. ساختار موازی که واحدهای MAC را ترکیب می کند، که هر یک به یک ماژول RNS تخصیص داده می شود، یک معماری معمول در کاربردهای رمزنگاری است[5،10،22،23،36،48،54،61،63]. مثالی در شکل 4 ارائه شده است.

در [36،54]، استفاده از RNS در RSA اولین بار بعد از پیشرفت هایی که در معماری رخ داد در [36،48] مطرح شد. در این مطالعات، CRT برای BC و پردازنده RNS کامل برای محاسبات RSA بکار رفته اند. پردازنده متشکل از 11 واحد MAC است که می تواند 22، 33 یا 66 ماژول را به ترتیب برای RSA 678 بیتی، 1024 بیتی یا 1024 بیتی بکار برد. بعد از آن، کاندینو و همکاران، طراحی های قبلی را با استفاده از پیش محاسبات بهینه سازی کردند[22،23]. طراحی ایشان احتمالا بهترین پردازنده رمزنگاری RNS تاکنون است. در همان زمان، در [61] ایده پردازنده های RNS دو فیلدی معرفی شدند یعنی معماری هایی که از محاسبات $GF(2^n)$ و $GF(p)$ ،هردو، پشتیبانی می کنند. این مشخصه باعث انعطاف پذیری بالا می شود چرا که الگوریتم های رمزنگاری مختلف می توانند در یک طراحی پشتیبانی شوند. در [61] از دو طراحی های قبلی در پیاده سازی BC فاصله گرفته و به جای CRT از MRC استفاده کرده است همچنین عملیات در آن موازی سازی شده اند. در نهایت، طراحی پردازنده های RNS برای رمزنگاری منحنی بیضوی هنوز در ابتدای راه قرار دارند[21،62].

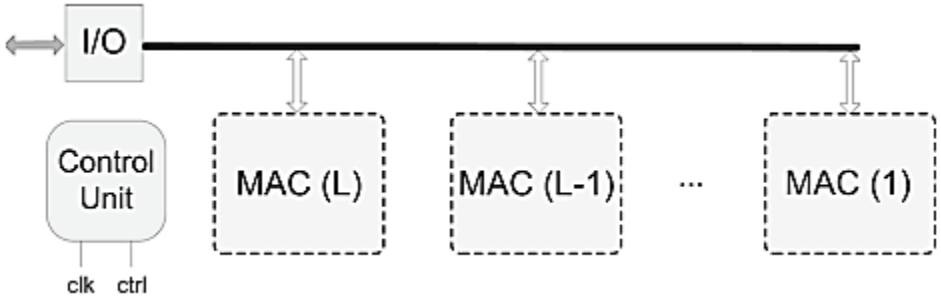
3-2-1- نکاتی درباره امنیت

شاید مهم ترین پیامد این مطالعات مزیت های امنیتی پردازنده های رمزنگاری مبتنی بر RNS باشد که ناشی از استفاده از محاسبات RNS است[6،60]. در زمینه RSA-CRT اثبات شده است که عملیات BC مکانیزم کلیدی است که در مقابل حملات سخت افزاری مقاومت می کند. مطالعات قبلی برای جعل کردن چنین حملاتی نیازمند تغییر سطح پروتکل بودند لذا باعث افزایش پیچیدگی رمزنگاری می شدند.

حملات سمت کانال که منجر به مصرف زمان یا توان پردازنده می شوند نیز می توانند با استفاده از RNS کاهش یابند. بعنوان مثال، می توان در مسیر داده محاسبات RNS کامل با تصادفی سازی مسیرهای ماژولی، مبهم سازی کد انجام داد[52] بنابراین تاثیر الکترومغناطیسی محو می شود. ASE نیز از نمایش RNS بی بهره نمانده است. در [17]، سه هسته AES مبتنی بر RSA موازی داده های ورودی را در قالب RNS پردازش می کنند. بعد از تکمیل پردازش، نتیجه به نمایش دودویی تبدیل شده و مکانیزم حذف سربار مشابه با آنچه در بخش 2-2 گفته شد تا بیش از 4 بیت خطا را تشخیص می دهد.



شکل 3 : سلول RNS ضرب-جمع (MAC) . برگرفته از [36]



شکل 4 : معماری کلی رمزنگاری مبتنی بر MAC. برگرفته از [62]

4-RNS در رمزنگاری پساکوانتومی

4-1- پایه ها در لاتیس

درحالی که آینده محاسبات کوانتومی هنوز مبهم است و هنوز مشخص نیست ما می توانیم مسئله برتری کوانتومی را حل کنیم یا خیر، تلاش برای ارائه طرح های رمزنگاری پساکوانتومی واقع گرایانه نیست. دلیلش کاملا واضح است چرا که یک کامپیوتر کوانتومی کامل دنیای رمزنگاری کلید عمومی را متحول می کند[65]. این یک مسئله بدیهی است چرا که سال ها طول می کشد تا سیستم های رمزنگاری کلید عمومی بالغ و کامل شوند.

سیستم های رمزنگاری مبتنی بر لاتیس(LBC) گزینه های بسیار امیدوارکننده ای برای دوره رمزنگاری پساکوانتومی هستند چرا که امنیت قوی آن ها در سختی بدترین حالت اثبات شده است و بعلاوه پیاده سازی های نسبتا کارامدی داشته و بسیار ساده هستند. محاسبات LBC بسیار پیچیده هستند.

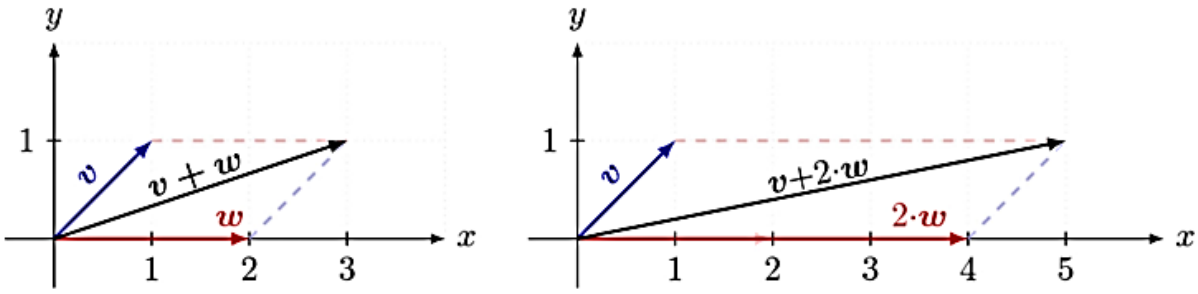
قبل از توضیح اینکه لاتیس چیست اجازه دهید بعضی مفاهیم پایه ای جبر خطی را بررسی کنیم. قلب یک لاتیس یک بردار است؛ برای سادگی ما فضای دوبعدی را تصور می کنیم(جبر خطی در فضاهایی با ابعاد بالاتر نیز میسر

است). در سمت چپ شکل 5، دو بردار $v = (1,1)$ و $w = (2,0)$ و جمع آنها یعنی $v + w = (3,1)$ با قانون شناخته شده جمع برداری را رسم می کنیم. در سمت راست، v و $2w = (2,0)$ جمع می شوند تا نقطه برآیند $v + 2.w = (5,1)$ بدست آید. بطور کلی، اگر ضرب بردارهای v و w به ترتیب در اعداد c و d را ترکیب کنیم و جمع آن ها را در نظر بگیریم، ترکیب خطی $cv + dw$ را بدست می آوریم. بعنوان مثال، عملیات روی سمت راست شکل 5 مربوط به ترکیب خطی v و w برای c=1 و d=2 است.

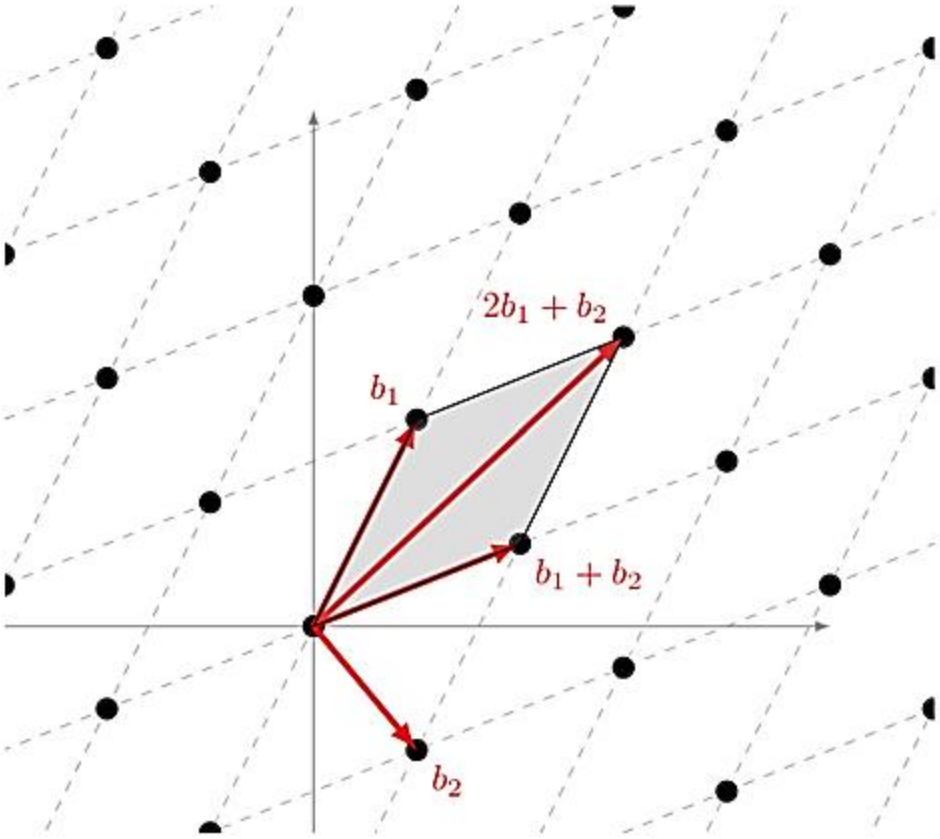
حال اگر تمام ترکیب های ممکن برای اعداد c و d را در نظر بگیریم می توانیم دیاگرامی مانند شکل 6 رسم کنیم. با دو بردار بعنوان پایه (b_1, b_2) شروع می کنیم و تمام مقادیر ممکن برای c و d را در نظر می گیریم تا نمودار نقطه ای بدست آید که در آن هر نقطه مربوط به نقطه پایانی نتیجه $cb_1 + db_2$ است. این مجموعه نقاط را یک لاتیس می نامند. از آنجا که می توان ترکیب خطی از بردارهای n بعدی تعریف کرد، بدیهی است می توان لاتیس های n بعدی نیز داشت. در نهایت، برای n بردار مستقل خطی $b_1, b_2, \dots, b_n \in R^n$ لاتیس بدست آمده مجموعه ای از بردارهاست:

$$\mathcal{L}(\mathbf{b_1}, \mathbf{b_2}, \dots, \mathbf{b_n}) = \left\{ \sum_{i=1}^{n} x_i \mathbf{b_i} : x_i \in \mathbb{Z} \right\}. \qquad (7)$$

بردارهای $b_1, b_2, \dots, b_n$ را پایه لاتیس می نامند. برای سادگی، می توان پایه را بصورت یک ماتریس $B = [b_1, b_2, \dots, b_n] \in R^{n \times n}$ با بردارهای پایه b_i بعنوان ستون ها نشان داد. واضح است، لاتیس های مشابه دارای پایه های بسیاری هستند(در شکل 6 بعنوان مثال، پایه دیگر می تواند $(v, w) = (2b_1 + b_2, b_1 + b_2)$ باشد. همین مشخصه است که ویژگی های امنیتی مفیدی را در LBC فراهم می سازد. در پاراگراف بعدی به این مسئله می پردازیم.)



شکل 5 : مثالی از جمع برداری



شکل 6 : مثالی از لاتیس

4-2- رمزنگاری مبتنی بر لاتیس

روش استفاده از لاتیس ها در رمزنگاری اندکی مبهم است و تا انتشار مقاله آجتای [1] اطلاعات زیادی درباره آن وجود نداشت. این مطالعه یک چارچوب مستحکم برای ساختن سیستم های رمزنگاری مبتنی بر لاتیس فراهم ساخت. در پاراگراف های 4-2 تا 4-5 ، بعضی جنبه های مهم سیستم های رمزنگاری مبتنی بر لاتیس بررسی شده و سپس یکی از اولین سیستم های رمزنگاری مبتنی بر لاتیس بعنوان نمونه ارائه می شود. همچنین، بعضی از جدیدترین کاربردهای محاسبات RNS در این زمینه مطرح می شوند.

درست مانند ECC مبتنی بر حل پذیری مسئله لگاریتم گسسته منحنی بیضوی (ECDLP) یا RSA مبتنی بر دشواری فاکتورگیری از اعداد اول است، بنابراین امنیت سیستم های LBC به حل پذیری مسائل ریاضی مختلف وابسته است. شناخته شده ترین مسائل مرتبط با لاتیس عبارتند از:

- SVP با داشتن یک پایه B ، کوتاه ترین بردار غیرصفر را در لاتیس $L(B)$ می یابد.
- CVP با داشتن پایه B و بردار مرجع r (لزوما یک در لاتیس نیست)، نزدیک ترین نقطه لاتیس $w \in L(B)$ به r را می یابد.
- SIVP با داشتن پایه $B \in Z^{n\times n}$ ، n بردار لاتیس مستقل خطی $U = [u_1, u_2, \dots, u_n]$ را می یابد بطوریکه برای تمام بردارهای b_i داشته باشیم: $\max||u_i|| < \max||b_i||$. $||.||$ نشان دهنده نرم اقلیدسی $||v|| = \sqrt{\sum_i v^2}$ است.).

همچنین دو نسخه تخمینی از این مسائل وجود دارد که باعث انعطاف پذیری بیشتر و متعاقبا کارایی بهتر می شود. بعنوان مثال، مسئله SVP به مسئله ای تبدیل می شود که در عوض یافتن کوتاه ترین بردار ، کوتاه ترین برداری را می یابد که از کران از پیش تعیین شده γ بزرگتر باشد. این روش ها خیلی کاربردی هستند اما برای سادگی تنها مسائل اصلی در اینجا بررسی می شوند[44،56].

بسیاری از روش های مبتنی بر لاتیس کاملا کارامد هستند؛ آن ها از محاسبات ساده تر از RSA یا ECC استفاده می کنند و پیاده سازی نسبتا ساده ای دارند و مهم تر از همه اینکه در مقابل کامپیوترهای کوانتومی منعطف هستند (یا حداقل هیچ الگوریتم کوانتومی برای حل مسئله SVP کارامدتر از الگوریتم های غیرکوانتومی وجود ندارد) .

وقتی بحث امنیت است، LBC را می توان به دو دسته تقسیم کرد: دسته اول شامل روش های کاربردی است که بسیار کارامد هستند اما اغلب فاقد اثبات قوی می باشند. نوع دوم امنیت قابل اثباتی بر اساس سختی بدترین حالت مسائل لاتیس دارند اما تنها تعداد انگشت شماری از آن ها در عمل قابل استفاده هستند[44].

سختی بدترین حالت به معنای آنست که اگر سیستم رمزنگاری بتواند به چند بخش تقسیم شود، هر یک از این بخش ها یک مسئله مبتنی بر لاتیس قابل حل(قابل کنترل) باشند. این یک تضمین قوی برای امنیت است و همین باعث تمایز LBC در مقایسه با سایر سیستم های رمزنگاری کلید عمومی می شود. تقریبا تمام روش های رمزنگاری دیگر مبتنی بر سختی حالت متوسط هستند. بعنوان مثال، شکستن یک سیستم رمزنگاری بر اساس فاکتورگیری(مانند RSA) ممکن است حاکی از توانایی اش برای فاکتورگیری از بعضی اعداد انتخابی در یک توزیع معین است و نه توانایی فاکتورگیری از تمام اعداد. بعنوان مثال، باید اعدادی انتخاب شوند که فاکتورهای کوچکی ندارند اما ممکن است اعداد دیگری نیز وجود داشته باشد. چنین مشکلاتی در رمزنگاری مبتنی بر لاتیس وجود ندارد.

4-3- NTRU : یک سیستم رمزنگاری مبتنی بر لاتیس

حال از مفاهیم اصلی بخش های قبلی برای توصیف سیستم رمزنگاری NTRU استفاده می کنیم. NTRU توسط هافمن، پیفر و سیلورمن در جسله Crypo'96 پیشنهاد شده است[31]. NTRU اساسا با در نظر گرفتن حسابان روی چندجمله ای ها در $Z_q[x]/(x^n - 1)$ برای n عدد اول و q عدد صحیح کوچک پیشنهاد شده است(یعنی داده ها بصورت چندجمله ای هایی از درجه حداکثر n و ضرایبی بصورت مدول اعداد صحیح بیان می شوند). بعدا

ثابت شد که شکستن سیستم رمزنگاری مسئله ای است که می تواند روی لاتیس های اقلیدسی بیان شود[31،70].

یک ماتریس تصادفی $H \in Z^{n\times m}$ فرض می شود که می توان آن را به صورت یک ماتریس بلاک نوشت:

$$\mathbf{H} = [\mathbf{H}^{(1)}|\mathbf{H}^{(2)}|\dots|\mathbf{H}^{(m/n)}], \tag{8}$$

که در آن هر بلاک $H^{(i)} \in Z^{n\times m}$ یک ماتریس دوری است:

$$\mathbf{H}^{(i)} = \begin{bmatrix} h_1^i & h_n^i & \cdots & h_3^i & h_2^i \\ h_2^i & h_1^i & \cdots & h_4^i & h_3^i \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ h_{n-1}^i & h_{n-2}^i & \cdots & h_1^i & h_n^i \\ h_n^i & h_{n-1}^i & \cdots & h_2^i & h_1^i \end{bmatrix} \tag{9}$$

یعنی تمام ستون ها از ستون اول $h^{(i)} = (h_1^i, h_2^i, \dots, h_3^i)$ با چرخش های دایره ای بدست آمده اند. در چرخش ماتریسی، می توان نوشت:

$$\mathbf{H} = \mathbf{P}^*\mathbf{H}^{(i)} = [\mathbf{h}^{(i)}, \mathbf{P}\mathbf{h}^{(i)}, \dots, \mathbf{P}^{n-1}\mathbf{h}^{(i)}], \tag{10}$$

که P یک ماتریس جایگشت است:

$$\mathbf{P} = \left[\begin{array}{c|c} \mathbf{0}^T & \mathbf{1} \\ \hline \ddots & \\ \mathbf{I} & \mathbf{0} \\ & \ddots \end{array}\right] \tag{11}$$

که مختصات $h^{(i)}$ را بطور دایره ای می چرخاند. از نامگذاری $P * H^{(i)}$ استفاده شده که نشان می دهد جایگشت P روی ستون های $H^{(i)}$ اعمال می شود. سیستم NTRU بصورت زیر ایجاد می شود[32]:

-پارامترها : یک بعد اول n ، یک ماژول صحیح q ، یک عدد صحیح کوچک p و یک کران وزنی کوچک b_f .

-**کلید خصوصی**: کلید خصوصی متشکل از دو چندجمله ای است که بصورت بردارهای دودویی (f, g) نشان داده می شوند بطوریکه $[P * F]$ ماژول معکوس پذیر q است، $[P * F] = I(\text{mod } p)$ و $[P * g] = O(\text{mod } p)$.

- **کلید عمومی:** بدون وارد شدن به جزئیات ریاضی، کلید عمومی یک ماتریس ساخت یافته بصورت زیر است:

$$\mathbf{E} = \begin{bmatrix} \mathbf{I} & \mathbf{O} \\ \mathbf{P^*e} & q \cdot \mathbf{I}, \end{bmatrix} \tag{12}$$

که $e = [P * F]^{-1} g(\text{mod } q)$. ساختار ماتریس ساده است چون کلید عمومی می تواند تنها با بردار e نمایش داده شود.

-**رمزنگاری**: پیام ورودی بصورت یک بردار $m \in \{-1,0,1\}^n$ (یعنی سه مقدار برای ضرایب چندجمله ای در یک بردار n بیتی) با دقیقا $b_f + 1$ ورودی مثبت و b_f ورودی منفی رمزنگاری می شود. بردار m با یک بردار تصادفی $r \in \{-1,0,1\}^n$ به دقیقا $b_f + 1$ ورودی مثبت و b_f ورودی منفی الحاق می شود تا یک بردار کوتاه (r,m) بدست آید. سپس متن رمز بصورت زیر محاسبه می شود:

$$\mathbf{c} = \mathbf{m} + [\mathbf{P^*e}]\mathbf{r} \pmod{q}. \tag{13}$$

محدودیت هایی برای تعداد ورودی های غیرصفر در نظر گرفته می شود تا کران بالای احتمال خطاهای رمزگشایی تعیین شود[18،31]. اساسا، رمزنگاری یک عملیات با فرم کلی $c = m + er(\text{mod } q)$ است که در آن m پیام، e کلید عمومی و r یک مقدار تصادفی است. این الگو مجددا در مورد رمزنگاری هومومورفیک در نظر گرفته می شود که در آن عناصر در بعضی از الگوها می توانند عدد صحیح باشند. واضح است تمام عملیات به جمع ها و ضرب های ماژولار چندجمله ای میان عناصر ماتریس منتهی می شوند.

-**رمزگشایی**: رمزگشایی با ضرب متن رمز در ماتریس رمز $[P * F](\text{mod } q)$ بصورت زیر انجام می شود:

$$[\mathbf{P}^*\mathbf{f}]\mathbf{c} \pmod{q} = \\ = [\mathbf{P}^*\mathbf{f}]\mathbf{m} + [\mathbf{P}^*\mathbf{f}][\mathbf{P}^*\mathbf{e}]\mathbf{r} \pmod{q} = \\ = [\mathbf{P}^*\mathbf{f}]\mathbf{m} + [\mathbf{P}^*\mathbf{g}]\mathbf{r} \pmod{q}. \tag{14}$$

می توان اثبات کرد که کمیت $[P * f]m + [P * g]r$ محدود به q/2 است بنابراین سمت رمزگشایی می توان پیام را بدون مدول کاهش اضافی q بازیابی کند[31]. با کاهش مدول نهایی داریم:

$$[\mathbf{P}^*\mathbf{f}]\mathbf{m} + [\mathbf{P}^*\mathbf{g}]\mathbf{r} \pmod{p} = \mathbf{I} \cdot \mathbf{m} + \mathbf{O} \cdot \mathbf{r} \pmod{p} = \\ = \mathbf{m}. \tag{15}$$

بعلت محدودیت های موجود، تولید کلید خصوصی برای کمیت های $[P * F] = I(\text{mod } p)$ و $[P * F] = O(\text{mod } p)$ تعیین می شوند.

مجددا رمزگشایی به ضرب ها و کاهش های ماژولار در عناصر ماتریس منتهی می شود. این نشان می دهد با این کار توصیف اصلی NTRU برای پارامترهای اصلی به دست می آید. گزینه ها بیشتر و تحلیل های ریاضی مفصل تر را می تواند در [18،31،32،43،56] مطالعه کرد.

4-4- ملاحظات امنیتی LBC

برای درک بهتر، یکی از اولین سیستم های رمزنگاری LBC یعنی GGH را بررسی می کنیم که در سال 1997 توسط گلدریچ، گلدواسر و هالوی (GGH حروف اول این سه نفر است) ارائه شده و مسئله CVP استفاده شده است[27]. نویسندگان یک تابع یک طرفه دریچه ای معرفی کردند که مبتنی بر مسئله کاهش لاتیس است.

ایده پشت این تابع دریچه ای این است که با داشتن یک پایه اختیاری برای لاتیس، تولید یک بردار نزدیک به یک نقطه لاتیس موثر است. در حقیقت، استراتژی محاسبه یک نقطه لاتیس و سپس اضافه کردن یک بردار کوچک

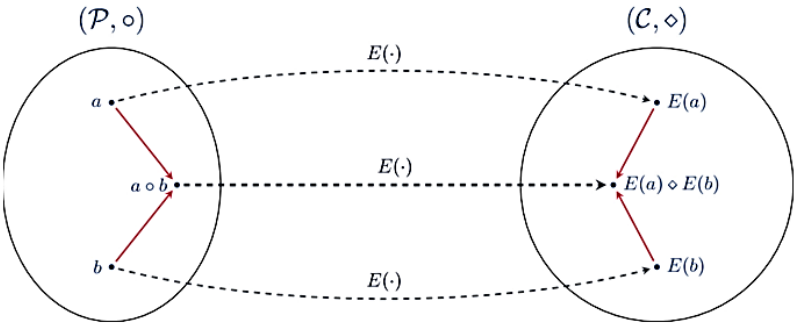
بعنوان بردار خطاست. نکته اینجاست که به منظور برگشتن به نقطه لاتیس اصلی باید یک بردار پایه خاص (دریچه) وجود داشته باشد.

صحت و امنیت هر دو وابسته به انتخاب پایه خصوصی و بردار خطا هستند. هرچند یک حمله خوب متناوب به GGH مفید نیست، حملات شناخته شده ای وجود دارند که سیستم رمزنگاری را با استفاده از مقادیر بزرگ پارامتر امنیت دچار مشکل می کند. البته، می توان از این حملات با بزرگ تر کردن پارامتر امنیت اجتناب کرد اما دیگر این سیستم غیرعملی است. بطور کلی، ذخیره سازی یک پایه لاتیس n برداری در R^n نیازمند فضای ذخیره سازی $\Omega(n^2)$ است و بنابراین زمان اجرای رمزنگاری/رمزگشایی بصورت یک تابع درجه دوم با توجه به پارامتر امنیت رشد می کند. یک تحلیل رمز کامل GGH توسط Phong Q. Nguyen در سال 1999 منتشر شد[47].

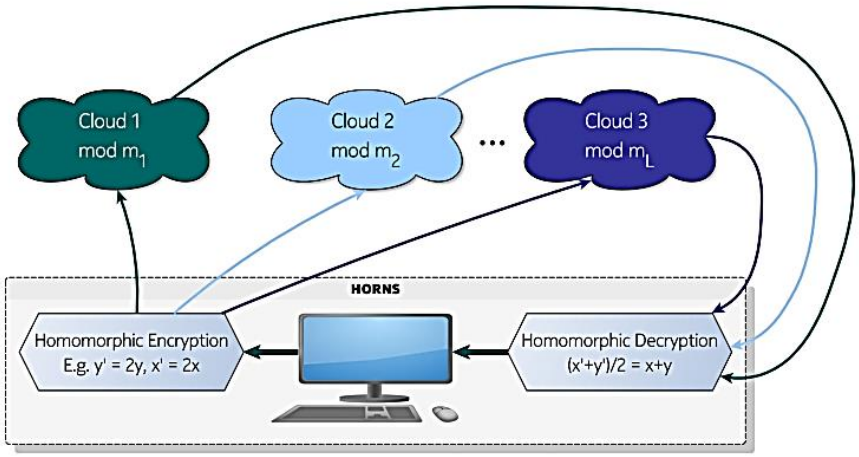
همانند NTRU، دو الگوریتم اصلی یعنی `NTRUEncrypt` و `NTRUSign` وجود دارد که عملیات آن ها نسبتا واضح است. `NTRUEncrypt` هنوز بعنوان یکی از کاربردی ترین سیستم های رمزنگاری مبتنی بر لاتیس شناخته می شود که با RSA یا ECC رقابت می کند هرچند اثبات رسمی امنیت در بدترین حالت برای آن وجود ندارد. اعتماد به این روش ریشه در بهترین حملات موجود دارد. برای NTRUEncrypt، مهم ترین حمله ناشی از Howgrave-Graham است[32]. بعد از انتشار این حمله، `NTRU CRYPTOSYSTEMS` مجموعه ای از مجموعه پارامترهای پیشنهادی را منتشر کرد[30]. (یک انتخاب در جدول 1 ارائه شده است). حالت NTRUSign متداول تر است چرا که یک سری لز حملات وجود داشته و قبل از پیکربندی روش کامل شود بررسی و راهکاری برای آن ها در نظر گرفته شده است[34،70]. اما هنوز هر دو روش در استاندارد IEEEP1363 وجود دارند[34].

Table 1 Recommended parameters for NTRU (see [30] for more options)

Security	n	q	b_f	Key-length (bits)
80	257	2^{10}	77	2.570
80	449	2^8	24	3.592
256	797	2^{10}	84	7.970
256	14.303	2^8	26	114.424



شکل 7 : هوموورفیسم گروهی



شکل 8 : رمزنگاری هوموورفیک با RNS

به موازات تلاش برای بهبود امنیت و کارایی سیستم های رمزنگاری مبتنی بر لاتیس کاربردی، پیشرفت هایی در سطح نظری و تئوری برای ارائه سیستم های رمزنگاری مبتنی بر لاتیس ایمن و قابل اثبات نیز مشاهده می شود. یکی از پیشرفت های مهم مدیون مطالعه آجتای است که منجر به یک تابع هش مقاوم در مقابل برخورد شده است که معادل با شکستن یک مسئله سخت به چندین مسئله لاتیس بدترین حالت روی لاتیس های اقلیدسی است[1]. مسئله آجتای را مسئله SIS می نامند. رویکرد مهم دیگر معرفی مسئله یادگیری با خطا(LWE) است که توسط رجیو [57] مطرح شده است. LWE روی حالت میانگین سخت است(معمولا مسائل لاتیس بدترین حالت بطور کوانتومی به آن کاهش می یابند)، اما به اندازه کافی برای ساختن گزینه های رمزنگاری انعطاف پذیر است. در سال های گذشته، چندین سیستم رمزنگاری مبتنی بر LWE و SIS ارائه شده است که امنیت آن ها تا زمانی که LWE و SIS سخت هستند قابل اثبات است. از میان روش های رمزنگاری که در مقابل حملات CCA و CPA مقاوم هستند می توان به رمزنگاری مبتنی بر هویت، امضاهای دیجیتال و ... نام برد[44]. اما مشکل اصلی سیستم های رمزنگاری مبتنی بر LWE و SIS ، کارایی است. کلیدها معمولا حاوی یک ماتریس تصادفی هستند که ابعاد آن حداقل بطور خطی با پارامترهای امنیتی رشد می کند و ازینرو فضا و زمان مورد نیاز نیز با توجه به پارامترهای امنیتی درجه دوم خواهد بود[70].

زمینه تئوری لاتیس و ساختارهای ریاضی و رمزنگاری بسیار گسترده است و بررسی کامل آن ها خارج از موضوع این مطالعه قرار دارد. آنچه مهم است، درک تحلیل بخش 4-3 برای تجزیه عملیات ریاضی اصلی موجود در LBC است. این می تواند فرصت هایی برای بهینه سازی بواسطه RNS فراهم سازد؛ بخش 4-5 به این موضوع می پردازد.

4-5- کاربرد RNS در رمزنگاری مبتنی بر لاتیس

در بخش 4-3، به بهینه سازی توسط محاسبات RNS اشاره شد. مشابه با RSA، ایده اصلی در NTRU اینست که عملیات اصلی عمدتا ضرب های ماژولار چندجمله ای و کاهش های ماژولار روی عناصر ماتریس هستند.

عناصر در NTRU چندجمله ای هستند و محاسبات مدول یک چندجمله ای $x^n - 1$ را انجام می دهند اما این تفاوت چندانی با محاسبات صحیح در الگوریتن 3-2 ندارد. در حقیقت، ساختاری به نام RNS چندجمله ای

(PRNS) وجود دارد که به جای اعداد صحیح با چندجمله ای ها کار می کند. نسخه های PRNS ضرب ماژولار منت کومری برای چندجمله ای ها نیز ارائه شده اند[11،37،59] و در نتیجه یک PRNS NTRU گزینه ای جالب برای سرعت بخشیدن به محاسبات است. چالش این است که الگوریتم های اصلی برای ضرب ماژولار چندجمله ای با PRNS شامل چندجمله ای ها و محاسبات در $GF(2^n)$ هستند(یعنی چندجمله ای های درجه n با ضرایبی در GF(2)) درحالی که در این حالت محاسبات روی ضرایب با توجه به q انجام می شوند.

بعدها، پیشرفت هایی در محاسبات RNS در روش های GGH به ویژه برای عملیات رمزگشایی انجام شد[9،41] که در ادامه برای درک بهترارائه می شوند چرا که منابع مفیدی برای پیشرفت بیشتر سایر روش های LBC هستند. پیام m بصورت یک بردار m با ضرایب کوچک یا تنها صفر و یک رمزنگاری می شود و رمز متن به فرم کلی $c = B.r + m$ بیان می شود که در آن B یک مبنای لاتیس "بد" است که بعنوان کلید عمومی بکار می رود و r برداری کوچک با ضرایب کوچک است(پایه "بد" عمومی با یک پایه / کلید خصوصی "خوب" G از طریق G=B.U در ارتباط است که در آن U یک ماتریس تک ماژولار است). بردار B.r متعلق به لاتیس و نزدیک ترین بردار به c است.

یکی از متداول ترین روش ها برای بازیابی یک پیام که تحت GGH رمزنگاری شده استفاده از الگوریتم گرد کردن بابایی [4] است. ایده کلیدی این است که برای رمزگشایی باید از متن رمز c و کلید خصوصی G استفاده و رابطه زیر حل شود:

$$\mathbf{m = c - Br}. \qquad (16)$$

توجه شود که باید r بگونه ای پیدا شود که Br نزدیک ترین بردار به m باشد. برای این منظور $r \leftarrow U\lfloor G^{-1}c \rceil$ تخمین زده می شود که در آن $\lfloor . \rceil$ نشان دهنده تابع صحیح گرد کردن به نزدیک ترین است. دلیل این که تخمین هیچ خطایی ندارد این است که ضرایب برای بردارهای r و m بطور ویژه انتخاب می شوند[27]. باید محاسبات کامل بررسی شوند چرا که آن ها به ما در حل مشکلات RNS کمک می کنند. به ویژه، می توان نوشت:

$$\mathbf{m} = \mathbf{c} - \mathbf{Br} = \mathbf{c} - \mathbf{BU}\lfloor \mathbf{G}^{-1}\mathbf{c} \rceil = \mathbf{c} - \mathbf{G}\lfloor \mathbf{G}^{-1}\mathbf{c} \rceil. \qquad (17)$$

بنابراین، در واقع، باید محاسبات رمزگشایی برحسب پایه خصوصی G و متن رمز c بیان می شوند. اگر رابطه 17 تنها شامل ضرب و جمع صحیح باشد، آنگاه کاربرد RNS ساده خواهد بود. متاسفانه، هرچند ماتریس G دارای عناصر صحیح است، معکوس اش ، G^{-1} ، گویا است. بعلاوه، ما باید روشی برای انجام عملیات گرد کردن بطور مستقیم در RNS داشته باشیم. در مجموع، محاسبات به فرم $\lfloor G^{-1}c \rceil$ در RNS مطلوب هستند.

درک چگونگی پیشرفت از سیستم اعداد موقعیتی به سیستم اعداد غیرموقعیتی بسیار مفید است. اول از همه، از آنجا که G^{-1} یک ماتریس مربعی است، از جبر خطی می دانیم که:

$$\mathbf{G^{-1}} = \frac{\mathbf{G'}}{d}, \qquad (18)$$

که d دترمینان G و $G' \in Z^{n\times n}$ ترانهاده ماتریس هم عامل G هستند. بنابراین مشکل اول اعداد گویا این است که بطور جزئی حل می شوند چرا که می توانیم G^{-1} را با تقسیم های صحیح عناصر G' بر دترمینان جایگزین کنیم. با توجه به تابع گرد کردن داریم:

$$\left\lfloor \frac{a}{b} \right\rceil = \left\lfloor \frac{a}{b} + \frac{1}{2} \right\rfloor = \left\lfloor \frac{2a+b}{2b} \right\rfloor = \frac{2a + b - \langle 2a+b \rangle_{2b}}{2b}. \qquad (19)$$

کسر نهایی ضریب تقسیم صحیح $2a + b/2b$ است بنابراین در حقیقت رابطه 19 یک تقسیم دقیق است. به یاد می آوریم که تقسیم دقیق در واقع یک ضرب در RNS است بنابراین بر مشکل گرد کردن برحسب ضرب های RNS غلبه می شود. با جایگزین کردن مقادیر G' و d در رابطه 19، محاسبات نهایی برای رمزگشایی GGH بصورت زیر است:

$$\left\lfloor \mathbf{G^{-1}c} \right\rceil = \left\lfloor \frac{\mathbf{G'c}}{d} \right\rfloor = \frac{2\mathbf{G'c} + \mathbf{d} - \langle 2\mathbf{G'c} + \mathbf{d} \rangle_{2d}}{2d}, \qquad (20)$$

که d برداری با تمام ورودی هایی است که دترمینان d هستند.

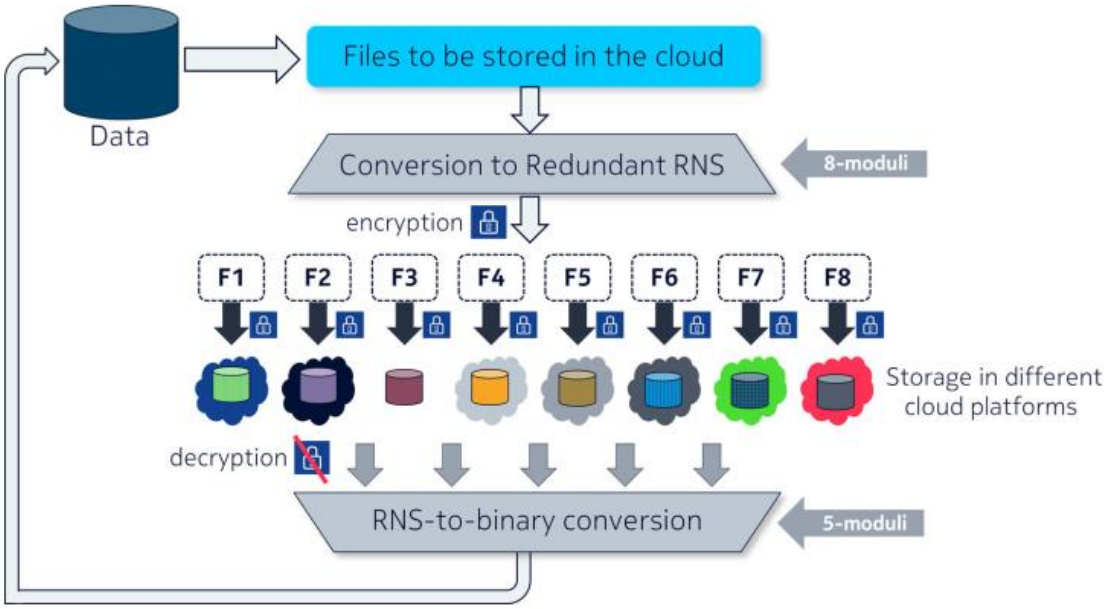
با عملیات رمزگشایی GGH در رابطه 17 شروع کرده و می بینیم که شرط اصلی برای بیان رمزگشایی در نمایش RNS اعداد گویا در ماتریس G^{-1} و عملیات گرد کردن هستند. سپس از جبر خطی ساده برای بیان G^{-1} برحسب تقسیم های صحیح در رابطه 18 استفاده می شود. در نهایت، از این تقسیم صحیح در عملیات گرد کردن در رابطه 20 استفاده شده و ارتباط آن با تقسیم های صحیح دقیق که برای انجام محاسبات RNS موثر است، اثبات می شود.

هنوز یک مشکل حل نشده وجود دارد که مانع کاربرد RNS در رابطه 20 می شود. ابتدا باید کاهش ماژولار کامل $\langle 2G'c + d \rangle$ در RNS در رابطه 20 محاسبه شود. بعبارت دیگر، آیا می توان کاهش های ماژولار با فرم کلی $< A >_{2d}$ را در RNS انجام داد؟ (فرض می شود : $A = 2G'c + d$ و N=2d). ما روشی غیرمستقیم برای انجام این کار در الگوریتم 3-2 برای ضرب ماژولار در RNS داریم.

با تنظیم $\vec{B} = \overrightarrow{B'} = \vec{1}$ ، الگوریتم $\overrightarrow{W}$ را به گونه ای محاسبه می کند که $W \equiv AM^{-1} (\text{mod } N) = A$ که نتیجه مورد نیاز است.

در این نقطه، اثبات شد تمام عبارت های رابطه 20 بطور کامل در RNS قابل حل هستند و بنابراین یک روش مشابه با RNS GGH امکان پذیر است[9،41]. در حقیقت، معماری ماژولار در شکل 4 می تواند RNS GGH کامل را با اندکی تغییرات پیاده سازی کند. اخیرا، معماری های GPU موازی نیز برای سرعت بخشیدن به محاسبات پیشنهاد شده اند[41]. این حوزه کاملا جدید است و هنوز هیچ نتیجه قاطعی منوط به مفید بودن استفاده از RNS در LBC وجود ندارد اما می تواند یک موضوع تحقیقاتی نویدبخش در آینده باشد.

در اینجا می توانیم نتیجه گیری کنیم که نمایش RNS در رمزنگاری کلید عمومی نسبتا سطحی است. نکته برجسته این است که در تمام الگوریتم هایی که بررسی کردیم(RSA ، ECC و LBC) یک الگوی مشترک برحسب محاسبات RNS وجود دارد یعنی تمام محاسبات در دسته ضرب های ماژولار منت کومری یا کاهش های ماژولار منت کومری در RNS قرار می گیرند. این نتیجه بسیار حائز اهمیت است چرا که هرگونه بهبودی روی این محاسبات به بهینه سازی مجموعه بزرگ تری از الگوریتم های رمزنگاری کلید عمومی منجر می شود.



شکل 9 : ذخیره فایل توزیع شده در ابر با RNS

5- حالت رمزنگاری هوموورفیک کامل

رمزنگاری هوموورفیک کامل (FHF) بی شک مبنایی برای پیشرفت رمزنگاری مدرن است[2،24] که امید می رود مشکل نهایی اعتماد به هویت های شخص ثالث را با محاسبات مستقیم بدون نیاز به افشای کلید، حل کند. این ویژگی بسیار پر اهمیت است به ویژه امروزه که ابرهای توزیع شده و شبکه های ناهمگن چارجوبی متصل را تعریف می کنند.

برای درک کاربردهایی که بواسطه مفهوم FHE میسر شده اند، فرض کنید یک کاربر داده های مربوط به سلامتی اش مانند فشار خون ، سطح کلسترول ، قند خون و غیره را به یک مرکز خدمات پزشکی ارسال می کند. مرکز می تواند با محاسبات آماری ، توصیه های بهداشتی و پزشکی متناسب با بیمار را تعیین و ارائه دهد. بدیهی است که این به معنای دسترسی به اطلاعات حساس شخصی نیست بلکه مرکز تنها می تواند تحلیل خود را روی داده های رمزنگاری شده انجام دهد. مرکز پرس و جو (کوئری) را بطور مستقیم روی فایل های رمزنگاری شده و بدون

نیاز به رمزگشایی آن ها انجام داده و سپس نتیجه را برمی گرداند و تنها کاربر است که می تواند آن را رمزگشایی کند. برای مطالعه کاربردهای مفید دیگر به [3] مراجعه کنید.

در یونان باستان، هوموورفیسم به معنای " چیزهایی با شکل یکسان" است. برای اینکه ربط آن با رمزنگاری هوموورفیسم را در کنیم، فرض کنید P فضایی برای امام متن های ساده و C فضایی برای تمام متن رمزهاست. بیایید تابع رمزنگاری E(.) را تصور کنیم که متن ساده در P را به یک متن رمز در C نگاشت می کند. این در شکل 7 برای عناصر $a, b \in P$ نشان داده شده است. واضح است که می توان تابع معکوس رمزگشایی D(.) را برای نگاشت عناصر از C به عناصر مربوطه در P تعریف کرد. حال فرض کنید که هر یک از مجموعه های P و C با عملیات اختیاری $(\circ,)$ تجهیز شده اند. این بدان معناست که می توان عملیات $a \circ b$ را برای عناصر a و b در مجموعه P انجام داد و نتیجه نیز متعلق به P است. عملیات مشابهی برای مجموعه C و عملیات انجام می شود. این فرآیند در شکل 7 با استفاده از کمان های قرمز نشان داده شده است. یک هوموورفیسم بین این مجموعه ها به معنای آنست که می توانیم نتیجه عملیات $a \circ b$ در P را به عنصر $E(a) \triangle E(b)$ نگاشت کنیم(خط چین نازک در شکل 7) یا بعبارت دیگر :

$$E(a \circ b) = E(a) \diamond E(b). \qquad (21)$$

آنچه می توان از این رابطه فهمید این است که به جای محاسبه یک عملیات رمزگشایی $a \circ b$ و سپس رمزنگاری آن می توان یک عملیات هوموورفیک $E(a) \triangle E(b)$ را مستقیم روی متن رمزهای E(a) ، E(b) و نتیجه انجام داد، سپس آن را رمزگشایی کرد و عملیات مطلوب $a \circ b$ یعنی $D(E(a)\ E(b)) = D(E(a \circ b)) = a \circ b$ را بدست آورد. این اصل رمزنگاری است چرا که شخص ثالث می تواند محاسبات معنادار را روی پیام های رمزنگاری شده بطور مستقیم اجرا کند و تضمین شود که نتایج به درستی بدون افشای هیچ پیام رمزگشایی نشده ای به شخص ثالث، رمزگشایی شود.

ایده اصلی پشت FHF ارائه یک روش رمزنگاری است که از تمام عملیات پشتیبانی کند بطوریکه بتوان کاربردهای معناداری ایجاد کرد. خوشبختانه، ما واقعا نیازمند عملیات بسیار زیادی نیستیم. در واقع، تمام توابع بولین می

توانند برحسب دروازه های AND (ضرب) و XOR (جمع) بیان شوند بنابراین رمزنگاری هممورفیک تنها نیازمند پشتیبانی از این دو عملیات است.

سادگی این گزاره بسیار زیرکانه است. پس از اینکه ریوست در 1978 برای اولین بار این سوال را مطرح کرد که آیا عملکرد رمزگذاری هممورفیک می تواند موثر است یا خیر ، تقریباً 30 سال می گذرد تا اینکه در نهایت جنتری در پایان نامه دکترای خود ثابت کرد که FHE امکان پذیر است[24].

بررسی رویکرد رمزگذاری هممورفیک یا انواع مختلف آن از حوصله این مقاله خارج است. مانند هر سیستم رمزنگاری شده ، FHE نیز به حل ناپذیری برخی از مسائل اساسی ریاضی متکی است. در حقیقت ، مسائل بسیاری وجود دارد که در حال حاضر پشتیبانی می شوند و بسیاری از آنها از حوزه تئوری شبکه ها [3] ناشی می شوند. مروری کوتاه در جدول 2 ارائه شده است و لیست کامل تر را می توان در [3] مطالعه کرد. به طور کلی ، اکثر مسائل مرتبط با FHE به SVP یا CVP کاهش می یابد ، بنابراین بسیاری از ایده ها برای سیستم های مبتنی بر شبکه که در بخش 4-3 ارائه شد در FHE نیز اعمال می شوند. به نظر می رسد روش های مبتنی بر اعداد صحیح یا چند جمله ای ها ، کاندیداهای مناسبی برای کاربردهای RNS هستند که در بخش 5-1 مورد بررسی قرار می گیرند. هنوز مشخص نیست آیا چنین کاربردپذیری برای سایر روش ها محسوس است یا خیر.

Table 2 Selected FHE schemes and their complexities

Scheme	Underlying problem	Complexity	Runtime
Gentry C.: A Fully Homomorphic Encryption Scheme [24]	BDDP & SSSP	$O(\lambda^{3.5})$ per gate for ciphertext refreshing	–
van Dijk et al.: FHE over the integers [75]	AGCD & SSSP	$O(\lambda^{10})$ no gate cost given	–
Coron et al.: Public key compression and modulus switching for FHE over the integers [19]	AGCD & SSSP	$O(\lambda^5 \log(\lambda))$ no gate cost given	Recryption takes about 11 minutes
Brakerski et al.: Leveled FHE without bootstrapping [13]	RLWE	$O(\log\lambda \cdot \lambda \cdot d^3)$ without bootstrapping (d is the depth of the circuits) and $O(\log\lambda \cdot \lambda^2)$ with bootstrapping	In [26] AES-128 encryption with 2 seconds/block (3GB RAM). With bootstrapping 6 seconds/block (3.7GB RAM). In [29]: Vectors of 1024 elements from $GF(2^{16})$ were recrypted in 5.5 minutes at security level $\approx$ 76 with a single CPU core
Gentry et al.: Implementing Gentry's fully homomorphic encryption scheme [25]	SVP & BDDP	Key generation is $O(\log n \cdot n^{1.5})$, where n is the dimension of the lattice	Bootstrapping from 30s for small setting to 30 min for large setting

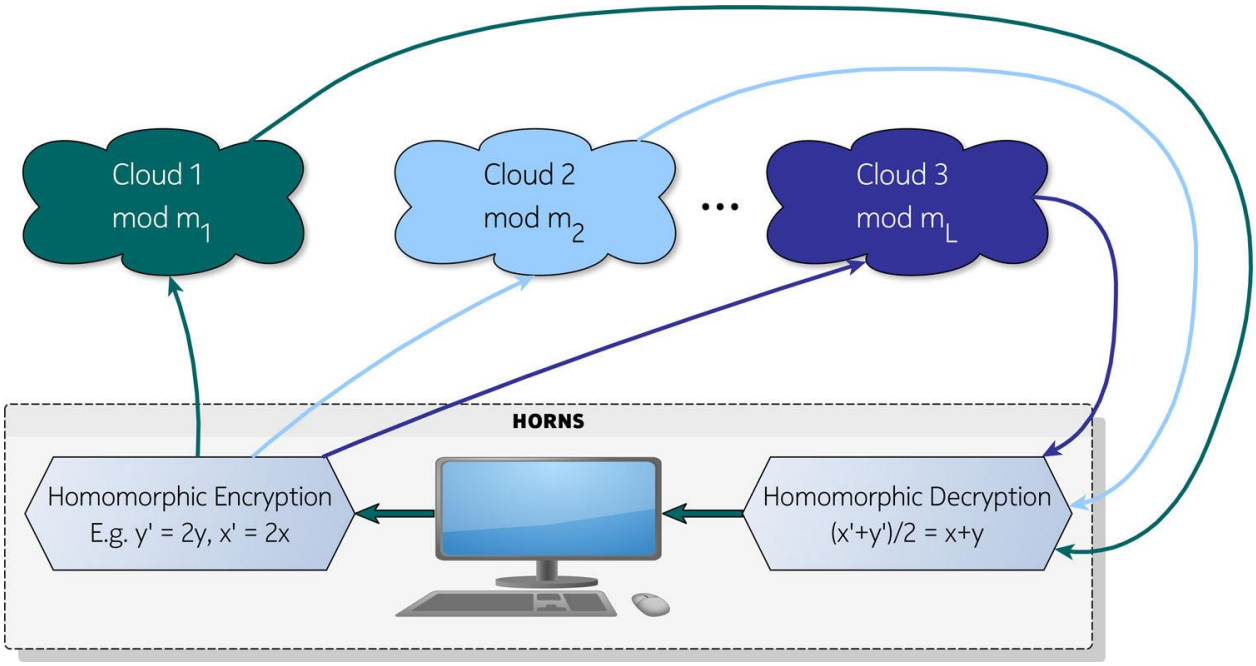


Fig. 8 Homomorphic encryption with RNS [28]

5-1- کاربردهای RNS در EHE

علیرغم اینکه FHF گل سرسبد رمزنگاری است اما هنوز راه زیادی برای کاربرد آن در عمل وجود دارد. اولین پیاده سازی FHF مبتنی بر لاتیس بیش از 1 ثانیه صرف رمزنگاری یک بیت و 17 مگابایت کلید عمومی در سرورهای مجهز به پردازنده های Intel Xeon می کند[25].

سپس، چند بهینه سازی نرم افزاری و سخت افزاری پیشنهاد شده است[8،49،76] در حالی که روش های FHF کارامدتر تحت عنوان leveled-FHF اخیرا در [13] پیشنهاد شده اند. در این تنظیمات، رمزنگاری اساس یک ضرب مبتنی بر ماتریس اعداد بزرگ است. RNS می تواند نقطه شروعی برای بهبود های چشمگیر باشد؛ نشان داده شده است که هر عدد صحیح بزرگ می تواند به مانده های کوچک تر تجزیه شده و سپس ضرب ماتریسی روی این

مقادیر کوچک تر انجام شود[76]. نتیجه این است که در مقایسه با کتابخانه تئوری اعداد مرتب CPU، عملیات با استفاده از تسریع GPU 273 مرتبه سریع تر و بدون تسریع GPU ، 7.8 مرتبه سریع تر خواهند بود[13].

کاربرد جالب توجه دیگر RNS عملیات روی رمزنگاری ساده به شکل $c = \varepsilon(m) = pq + m$ است که در آن m پیام، p کلید و q یک عدد تصادفی است. در حال حاضر الگوهای مشابهی در سیستم های رمزنگاری NTRU و GGH در بخش 4 مشاهده کردیم. الگو برای جمع، تفریق و ضرب هممورفیک است. بعنوان مثال، برای متن رمزهای $\varepsilon(m_1) = c_1$ و $\varepsilon(m_2) = c_2$ داریم :

$$c_1 + c_2 = (pq_1 + m_1) + (pq_2 + m_2) = p(q_1 + q_2) + (m_1 + m_2) = pq + (m_1 + m_2) = E(m_1 + m_2)$$

بعبارت دیگر، می توان متن رمزهای دو پیام را به یکدیگر جمع کرد و نتیجه رمزنگاری جمع پیام هاست. این برای دو عملیات دیگر صادق است.

این یک روش جالب است چرا که رمزگشایی، عملیات مانده است یعنی $m = c \bmod p$ و رمزگشایی دقیقا فرآیند معکوس آن است. این ایده اصلی پشت امنیت HORNS (رمزنگاری هممورفیک با RNS) است(شکل 8)[28]. این روش داده های رمزنگاری شده با رمزنگاری هممورفیسم مشابه $c = E(m) = pq + m$ تولید کرده و آن ها را به قالب RNS تبدیل می کند. سپس پارتیشن های RNS روی سرورهای ابری مختلف توزیع می شوند و ابر می تواند عملیات RNS را روی داده های مانده رمزنگاری شده بطور مستقیم انجام دهد. واضح است، RNS برای جمع، تفریق و ضرب هممورفیسک است(رابطه 2) بنابراین منحصربفرد بودن نتیجه تضمین می شوئد. برای بازیابی نتیجه، کاربر چندین عملیات انجام می دهد یعنی ابتدا داده های RNS را به قالب دودویی تبدیل می کند و سپس آن ها را از طریق هممورفیک ساده $m=c \bmod p$ رمزگشایی می کند.

مسئله ای که مشئول امنیت این روش است بیان می کند با داشتن دو عدد $C_1 = pq_1 + m_1$ و $C_2 = pq_2 + m_2$ که مضرب های p هستند و فرض اینکه p تنها تخمینی gcd است آیا امکان محاسبه p بطور کارامد وجود دارد؟ تا آنجا که ما می دانیم، خیر. در واقع، با دانستن اینکه هریک از اعداد m_1

، p و q_1 طول کلمه خاصی (به ترتیب λ، λ^2 و λ^5) دارند آنگاه یافتن p حتی وقتی متن رمزهای متعددی موجود باشد کار سختی است.

6- ذخیره سازی ابر و IoT

بحث را در زمینه ای ارائه می دهیم که حداقل اخیرا به آن پرداخته نشده است و آن کاربرد محاسبات RNS است. در حقیقت، RNS می تواند گزینه های منحصربفردی در یکی از مهم ترین مسائل در سیستم های ابر یعنی مسئله اعتماد فراهم کند. خواهیم دید که چگونه RPNS می تواند بطور موثر بعضی از نگرانی های مهم در معماری های ابری مانند یکپارچگی داده ها، دسترس پذیری و اعتمادپذیری را حل کند[15].

در معماری شکل 9 [15] استفاده از PRNS با هشت ماژول فرض می شود که سه تای آن ها بعنوان ماژول تکراری بکار می روند. کاربر ابتدا داده هایی که قرار است در ابر ذخیره شوند را در قالب RNS ذخیره می کند یعنی در چانک داده های مختلف () که هر یک مربوط به یک ماژول RNS هستند. هر چانک با یک تراشه متقارن رمزنگاری شده و در یک ابر متفاوت ذخیره می شود درحالی که کاربر یک فایل متا داده را همراه با اطلاعاتی درباره موقعیت فایل ها و روش بازیابی نگهداری می کند. کاربر مسئول ذخیره سازی فایل متاداده ها به شکلی ایمن است. به محض درخواست، کاربر می تواند فایل را با استفاده از پنج ماژول برای روش تبدیل بازسازی کرده و فایل را با استفاده از کلید خصوصی اش رمزگشایی کند.

این روش چندین مزیت دارد؛ اول از همه، هنگام خرابی سیستم کاربر می تواند هنوز داده ها را بازسازی کند با این فرض که 5 ماژول کافی است. بعلاوه، مراکز سرویس دهی ابر از وجود فایل ها مطلع نیستند چرا که فقط کاربر می داند فایل ها کجا هستند و چگونه بازیابی می شوند.در نهایت، دانلود موازی فایل ها از سرویس دهندگان مختلف باعث افزایش بهره وری پهنای باند می شود. واضح است، اندازه ذخیره سازی یک فایل تابعی از ماژول تکراری و مینیمم تعداد ماژول های مورد نیاز برای بازسازی فایل است[16].

همچنین می توان مفاهیم مشابهی را مطرح کرد. بعنوان مثال، مطالعه ایزوله سازی امنیتی در سیستم های ابر مبتنی بر RNS با استفاده از کلیدهای مختلف برای هر چانک نیز می تواند جالب باشد. در مثال شکل 9، نقض

یک کلید در یک یا چند کانال نباید باعث افشای اطلاعات برای کلیدها در سایر کانال ها شود. در کل، اسپارس و پراکنده بودن ذاتی که توسط RNS ایجاد می شود با مفاهیم مربوط به سیستم های تکه تکه (فرگمنت) و محاسبات موازی سازگار است. امنیت در چنین مواردی می تواند رایگان باشد چرا که RNS امکان مبهم سازی کد و تغییر ماژول را ممکن می سازد اما اثبات رسمی مورد نیاز است.

از کاربردهای دیگر RPNS می توان به شبکه های حسگر بیسیم (WSN) که در سناریوهای IoT برای شهرهای هوشمند مورد نیاز هستند، اتوماسیون صنعتی، کاربردهای کشاورزی و غیره اشاره کرد[38]. اثبات شده که با استفاده از RPNS ، روش های تصحیح خطای کارامد می توانند پیچیدگی کم، تحمل پذیری خطا و ارسال داده به صرفه را فراهم سازند[38]. سوال این است که آیا روش های رمزنگاری RNS می توانند با روش های تصحیح خطای RPNS برای ایجاد یک کانال ارتباطی مقاوم ترکیب شوند یا خیر.

جمع بندی:

در این مطالعه، گزینه های بهینه سازی مختلف ارائه شده توسط محاسبات مانده با توجه به کاربردهای امنیتی بررسی شده اند. چگونگی تبدیل RNS به یک گزینه پیاده سازی مناسب برای شناخته شده ترین سیستم های کلید عمومی یعنی RSA و ECC نیز توضیح داده شده است. این کاربردها به پیشرفته ترین سیستم های رمزنگاری مبتنی بر لاتیس بسط داده شده اند که گزینه هایی نویدبخش برای محاسبات پساکوانتومی هستند. شباهت هایی میان این سیستم ها وجود دارد که منجر به مشاهده مهم می شود اینکه هر گونه بهبود در محاسبات RNS به معنای بهبود در مجموعه بزرگ تری از الگوریتم های کلید عمومی است.

همچنین در این مطالعه مقدمه ای بر یکی از جالب توجه ترین حوزه های رمزنگاری مدرن یعنی رمزنگاری هومومورفیک ارائه شده است. ما هنوز با پیاده سازی های کاربردی و عملی فاصله زیادی داریم اما RNS می تواند

این مسیر را هموار سازد. ما روش های جدیدی برای محافظت از داده ها در ابر و مکانیزم هایی ارائه دادیم که RNS برای مبهم سازی داده های اضافی و ایزوله سازی در سیستم های توزیع شده بکار می برد.

هنوز سوالات تحقیقاتی باز متعددی وجود دارد که باید به آن ها پاسخ داد. اول از همه، بررسی راه حل های پساکوانتومی غیرممکن است چرا که راه حل های رمزنگاری هنوز در اول راه قرار دارند و سال ها طول می کشد که به بلوغ برسند. سیستم های رمزنگاری مبتنی بر لاتیس و منحنی بیضوی کاندیدهای خوبی برای بهینه سازی RNS و در عین حال ویژگی هایی هستند که باعث کاربردی شدن آن ها می شود. ایده هایی برای محاسبات مانده چندجمله ای در NTRU و روش هایی برای تبدیل محاسبات در سیستم های GGH برای نمایش RNS نیز بررسی شدند. شکاف های تحقیقاتی زیادی در تمامی جنبه های محاسبات کارامد ، امنیت، طراحی سخت افزار و اینکه RNS چگونه می تواند در مقابل انواع حملات مقاومت کند، وجود دارد[6،60]. RNS قطعا گزینه ای است که ، به ویژه برای معماری های محاسبات موازی، باید در نظر گرفته شود.

ما بر این باوریم که بررسی محاسبات مانده در رمزنگاری هممورفیک ارزش بالایی دارد. بعلت پیچیدگی بسیار زیاد، کاربرد آن در ابر محدود است. بعبارت دیگر، موازی سازی ذاتی RNS با معماری های مبتنی بر ابر موازی توزیع شده نطابقت دارد و بنابراین ارتباط میان این دو مفهوم بدیهی است. بررسی افزایش سرعت ناشی از RNS نیز بسیار حائز اهمیت است.

در نهایت، روش استفاده محاسبات RNS برای امنیت داده ها در سیستم های ابری نیز نسبتا جدید است. ما بر این باوریم که دست یابی به نتایج قطعی درباره اینکه آیا RNS می تواند جنبه های مختلف مسئله اعتماد را در پلتفرم های ابری حل کند و باعث امنیت بیشتر RNS شود.

منظور ما این نیست که RNS راه حل تمام مشکلات محاسباتی در چنین محیط هایی است و بر این باوریم که هنوز از محاسبات سریالی مربوط به تبدیل خروجی و تبدیل های پایه رنج می برند. با این حال، وقتی در محاسبات سنگین وزن و تکرارشونده مانند رمزنگاری استفاده می شود این محدودیت ها در مقایسه با افزایش سرعت محاسبات RNS رنگ می بازد. در نتیجه، بر این باوریم که چارچوب محاسبات توزیع شده مبتنی بر ابر که کاربردهای

متعددی در مقیاس 5G و IoT دارد. این سیستم اعداد باستانی یک گزینه مفید است و راه حل هایی ارائه می دهد که در آن ها محاسبات کلاسیک موثر نیستند.