تاریخ دریافت:۱۴۰۰/۱۲/۰۲ پژوهشنامه علوم دفاعی، شماره ٦، تابستان ۱۴۰۱
تاریخ پذیرش:۱۴۰۱/۰۴/۲۲ ص ص ۱۰۱ تا ۱۱۸

پروتکل رمزنگاری مبتنی بر مشبکه و امنیت شبکه‌های نظامی

آرش جوان[۱] و جواد شرفی[۲]

چکیده

امنیت و حفاظت از اطلاعات همواره مد نظر دولت‌ها بوده و دانشمندان روش‌ها و الگوریتم‌های متنوعی را به مرور زمان ارائه نموده‌اند. امروزه سیستم‌های رمزنگاری با استفاده از حلقه‌های چندجمله‌ای از محبوبیت بسیاری برخوردار است. زیرا این سیستم‌ها در مقایسه با بسیاری از سیستم‌های دیگر برای داشتن یک امنیت بالا به کلید کوچک‌تری نیاز دارند. برای استفاده از یک سیستم رمزنگاری مبتنی بر مشبکه ابتدا باید پیام به صورت دنباله‌ای از نقاط بر روی حلقه‌ای جندجمله‌ای درآید تا سپس سیستم رمز مورد نظر بر روی آن پیاده شود. یکی از روش‌های مقابله با جملات مبتنی بر فرکانس، استفاده از رمزهای متشابه الصوت هموفون است. در این مقاله یک رمز مبتنی بر مشبکه و امنیت این رمزنگاری در شبکه‌های نظامی را ارائه می‌کنیم. برای رمزگشایی پیام، باید به مولدهای خاص استفاده شده توسط طرفین دسترسی داشت که الگوریتمی کارا برای این‌کار وجود ندارد.

واژه‌های کلیدی: حلقه چندجمله‌ای‌ها، رمزنگاری مشبکه مبنا ، تبدیل نظری عددی، امنیت شبکه‌های نظامی، کلید عمومی و خصوصی.

۱. دکتری ریاضی، دانشگاه علوم و فنون دریایی امام خمینی (ره)، دانشکده علوم پایه، نوشهر، ایران (نویسندهٔ مسئول)، ✉ arash.javan91@gmail.com
۲ دانشجوی دکتری رمزنگاری، دانشگاه افسری امام علی(ع)، تهران، ایران

مقدمه

در سامانه‌های دفاعی، امنیتی و نظامی اطمینان از عدم افشای اطلاعات توسط فرآیندها با عوامل غیر مجاز از مصادیق بارز محرمانگی است، که در سامانه‌های مخابراتی با روش‌های رمزنگاری به‌دست می‌آید.

در این طرح راه‌کارهای امنیتی مبتنی بر پروتکل‌های رمزنگاری و سیستم‌های امنیت اطلاعات و ارتباطات با رویکرد بومی سازی در صنایع و علوم دفاعی خواهد بود. این پروتکل و امنیت آن در ارتباطات نظامی و دفاعی بسیار حساس است.

این طرح یک پروتکل رمزنگاری کلید عمومی مبتنی بر مشبکه است که دارای امنیت CCA و CPA بر اساس سختی کوانتومی مساله RLWE است. مساله RLWE با در نظر گرفتن مساله LWE روی حلقه چندجمله‌ای‌های پیمانه‌ای و به‌منظور افزایش کارایی معرفی گردید. در این الگوریتم برای سه پارامتر (q,n,k) که در آن q عدد اول ثابت ۱۲۲۸۹ و k=8 برای نویز دو جمله‌ای و برای n=1024 سختی حمله کوانتومی شناخته شده ۲۳۳ کیلوبیت است. در مورد حمله‌ها باید یادآور شویم که تمام حمله‌هایی که به رمزنگاری‌های کلید عمومی انجام می‌شود نیز به این طرح نیز وارد است. در واقع اگر روش حمله‌ای به این طرح وارد بشود در واقع معادل با روشی برای شکستن مساله RLWE است، لذا اگر حمله‌ای با احتمال خوبی برای این طرح پیدا شود از آن روش حمله می‌توان برای شکستن مساله RLWE استفاده کرد. چون هیچ روشی برای شکستن مساله RLWE وجود ندارد، بنابراین می‌توان نتیجه گرفت که هیچ روش حمله موثری برای شکستن این طرح وجود ندارد. تنها مساله‌ای که وجود دارد این است که کلید عمومی را بتوان درست حدس زد که این موضوع هم با انتخاب پارامترهای به اندازه کافی بزرگ این احتمال را به صفر می‌رساند.

در نهایت الگوی جدیدی با رویکرد ترکیبی و امکان کاربرد همزمان رمزنگاری برای سامانه‌های نظامی و دفاعی ارائه می‌شود. بیان این مطلب کاربرد رمزنگاری و احراز اصالت

پیام در عرصه سنجش و فرمان از راه دور در سازمان‌های نظامی و دفاعی را تبیین خواهد نمود.

مبانی نظریه و پیشینه

سامانه رمز ما دارای سه الگوریتم تولید کلید، رمزگذاری و رمزگشایی خواهد بود :

اول، یک الگوریتم برای تولید کلید عمومی (CPA-PKE Key Generation) که آلیس با اختیار کردن یک مقدار تصادفی و محرمانه (کلید خصوصی) اقدام به ساخت کلید عمومی می‌کند به طوری‌که محاسبه کلید خصوصی از روی کلید عمومی برای دشمنی که دارای توان محاسباتی محدود است، غیر ممکن باشد. کلید خصوصی به‌صورت امن و محرمانه نگهداری می‌شود و کلید عمومی به‌صورت عمومی اعلام می‌شود تا در صورت نیاز هر شخصی که خواست بتواند به سازندگان کلید عمومی پیام رمزشده بفرستد.

دوم، یک الگوریتم برای رمزگذاری (CPA-PKE Encryption) که هر کسی با داشتن کلید عمومی آلیس می‌تواند رمز شده پیام مورد نظر را محاسبه کند.

سوم، یک الگوریتم برای رمزگشایی (CPA-PKE Decryption) که با داشتن کلید خصوصی و کلید عمومی آلیس می‌توان هر پیامی که با کلید عمومی آلیس رمز شده را رمز گشایی کرد.

نکته : چون در این تحقیق تمامی اعداد درایه‌ها کمتر از ۲۵۵ هستند لذا نمایش هر عدد صحیح ۸ بیتی خواهد بود.

ایده اصلی رمزنگاری از الگوریتم رمزنگاری رگیو[۱] استفاده شده است که بر مبنای سختی RLWE است. در نسخه رمزنگاری‌ای که رگیو ارائه کرده بود، پیام یک بیت بوده است. در واقع از یک چندجمله‌ای نیز به‌عنوان یک پیام استفاده کرد ولی در این طرح یک آرایه (رشته) بایتی داریم، لذا تمام چندجمله‌ای‌هایی که داریم باید به‌صورت یک رشته بایتی نمایش داده

[۱] Regev

شود که هم سرعت بالاتر رود و هم عملگرهایی که از آن‌ها استفاده می‌کنیم با سرعت بیشتری کار خود را انجام می‌دهند. بنابراین کلید عمومی که ساخته می‌شود، نمایش بایتی آن به‌عنوان خروجی فرستاده می‌شود، چون بدین شکل، ذخیره سازی آن در کامپیوتر راحت انجام می‌شود. همچنین محاسباتی که در رمزگذاری این الگوریتم انجام می‌شود، خروجی آن‌ها به‌جای این‌که به شکل حاصل‌ضرب یا حاصل‌جمع دو تا چندجمله‌ای باشد، به‌صورت آرایه بایتی است.

در جداول زیر به‌ترتیب به ارائه پارامترها و اندازه کلیدهای به کار رفته در این طرح پرداخته‌ایم و اعداد مورد نظر را به‌طور دقیق ذکر می‌کنیم.

پارامترها	۵۱۲	۱۰۲٤
بعد n	۵۱۲	۱۰۲٤
پیمانه q	۱۲۲۸۹	۱۲۲۸۹
پارامترk در نویز دوجمله‌ای	۸	۸
پارامترγ در NTT	٤۹	۷
احتمال خطای رمزگشایی	$۲^{-۲۱۳}$	$۲^{-۲۱٦}$
امنیت پساکوانتومی	۱۰۱	۲۳۳

پارامترها	اندازه کلید عمومی	اندازه کلید خصوصی	اندازه متن رمزشده
512-CPA-KEM	۹۲۸	۸٦۹	۱۰۸۸
1024-CPA-KEM	۱۸۲٤	۱۷۹۲	۲۱۷٦
512-CCA-KEM	۹۲۸	۱۸۸۸	۱۱۲۰
1024-CCA-KEM	۱۸۲٤	۳٦۸۰	۲۲۰۸

روش شناسی تحقیق

در این تحقیق ما ابتدا به معرفی الگوریتم تولید کلید می‌پردازیم. سپس الگوریتم رمزگذاری این طرح را به‌طور مفصل بیان می‌کنیم. در انتها الگوریتم رمزگشایی را ارائه می‌دهیم. که این الگوریتم درستی این طرح را برای ما اثبات می‌کند. لذا درستی اثبات رمزگشایی را نشان می‌دهیم.

معرفی الگوریتم تولید کلید: این الگوریتم برای تولید کلید عمومی و خصوصی است. در این الگوریتم با استفاده از یک آرایه ۳۲ بایتی تصادفی، کلید خصوصی و کلید عمومی که از نوع چندجمله‌ای در R_q هستند، تولید شده و سپس به‌صورت آرایه‌هایی بایتی (کلید خصوصی $\frac{7n}{4}$ و کلید عمومی $32 + \frac{7n}{4}$) برگردانده می‌شوند. در این الگوریتم برای سه پارامتر (q,n,k) بیان شده است که در آن q عدد اول ثابت ۱۲۲۸۹ و k=8 برای نویز دو جمله‌ای و برای n=512 سختی حمله کوانتومی شناخته شده ۱۰۱ کیوبیت است و برای n=1024 سختی حمله کوانتومی شناخته شده ۲۳۳ کیلوبیت است. امنیت این مرحله به سختی محاسبه $\hat{s}$ با داشتن $\hat{a}$ و $\hat{b}$ است.یعنی حل مساله محاسباتی RLWE.

توضیح : این الگوریتم توسط آلیس اجرا می‌شود :

$$\text{seed} \leftarrow \{0, \dots, 255\}^{32}$$

یک آرایه ۳۲ بایتی از اعداد صحیح یکنواخت که یک رشته 32×8 بیتی است را بین ۰ تا ۲۵۵ انتخاب می‌کنیم که برای تولید هر بیت تصادفی هم از روش سکه انداختن استفاده می‌کنیم یا از روش موثرتر توابع تولید اعداد شبه تصادفی هم می‌توان استفاده کرد که غیرقابل پیش بینی است.

$$z \leftarrow \text{SHAKE256}(64, \text{seed})$$

تابع چکیده‌ساز shake256 (توابع چکیده‌ساز یا هش توابعی هستند که محاسبه وارون در آن‌ها برای دشمن با توان محدود غیر ممکن است) را روی seed اثر می‌دهد. خروجی این تابع یک آرایه بایتی ٦٤ تایی (با توجه با پارامتر داده شده تابع) است و آن را به عنوان z در نظر می‌گیریم. از آنجاییکه seed یک رشته ۳۲ بایتی است و به کمک تابع Shake به دو آرایه

شبه تصادفی با امنیت قابل قبول (غیر قابل معکوس سازی) دست خواهیم یافت که یک آرایه از آن برای ساخت کلید خصوصی و دیگری برای ساخت عنصر $\hat{a}$ که می‌بایست دارای توزیع خاص و شبه تصادفی باشد، مورد استفاده قرار می‌گیرد.

publicseed ← z[0: 31]

۳۲ درایه اول z (منظور همان ۳۲ بایت اول z است چون z دارای ٦٤ بایت بود) را به‌عنوان publicseed در نظر می‌گیریم.

noiseseed ← z[32: 63]

۳۲ بایت آخر z را به‌عنوان noiseseed در نظر می‌گیریم.

â ← GenA(publicseed)

با استفاده از الگوریتم GenA و آرایه publicseed چند جمله‌ای â را می‌سازیم.

الگوریتم GenA الگوریتمی است که ورودی ۳۲ بایتی می‌گیرد و یک چندجمله‌ای تصادفی در حلقه R_q به‌عنوان خروجی تحویل می‌دهد. خاصیت الگوریتم GenA این است که چندجمله‌ای که ساخته می‌شود در دامنه NTT قرار می‌گیرد که در الگوریتم 2 در ذیل به آن اشاره می‌کنیم. از آن جایی‌که â توسط GenA تولید می‌شود. یک چندجمله‌ای است که کد می‌شود و به‌صورت یک رشته بایتی در خروجی نمایش داده می‌شود. چون این خروجی مقداری عمومی است نیازی به توزیع احتمال خاصی ندارد. لذا در الگوریتم GenA نیاز به خواص خاصی ندارد بلکه باید به‌صورت یکنواخت اختیار گردد که این الگوریتم این کار را انجام می‌دهد. در مورد این الگوریتم بایتی می‌توان فهمید که قابل جایگزینی با الگوریتم‌های دیگر است، البته به شرط اینکه با ورودی یک رشته ۳۲ بایتی، خروجی آن باید عنصری تصادفی باشد و در دامنه NTT قرار بگیرد.

s ← PolyBitRev(Sample(noiseseed, 0))

با استفاده از الگوریتم sample و مقدار noiseseed و الگوریتم PolyBitRev چند جمله‌ای s را محاسبه می‌کنیم. برخلاف الگوریتم GenA، الگوریتم sample چندجمله‌ای‌هایی را که دارای توزیع احتمال ψ_8^n (که در فصل اول بخش مفاهیم آماری توضیح داده شد) است، تولید

می‌کند و در مسئله RLWE مقدار مخفی s (چندجمله‌ای مخفی s) و چندجمله‌ای نویز e باید با توزیع احتمال ψ_8^n باشند که حل کردن مسئله معکوس سخت باشد وگرنه اگراین چندجمله‌ای‌ها دارای توزیع دلخواهی باشند ممکن است مسئله RLWE دیگر سخت نباشد. الگوریتم Sample در صورتی قابل جایگزین با الگوریتم‌های دیگر است که با ورودی رشته ۳۲ بایتی، خروجی‌اش یک عنصر تصادفی از $\frac{\mathbb{Z}_q[x]}{<x^n+1>}$ با توزیع ψ_8^n باشد.

$\hat{s} \leftarrow \mathrm{NTT}(s)$

با استفاده از تبدیل NTT و چند جمله‌ای s ، چند جمله‌ای $\hat{s}$ را محاسبه می‌کنیم. NTT اگرچه صرفاً یک تبدیل است اما با توجه به ضرب تعریف شده و تابع‌های massage Encode و massage Decode سازگاری داشته و تغییر در هر یک، تغییر در بعد را به دنبال خواهد داشت. برای هر پیام و برای هر e از توزیع ψ_8^nداریم:

$\mathrm{massage\ Decode}(\mathrm{massage\ Encode}\ (\mu) + e) = \mu$

$e \leftarrow \mathrm{PolyBitRev}(\mathrm{Sample}(\mathrm{noiseseed}, 1))$

با استفاده از الگوریتم sample و مقدار noiseseed و الگوریتم PolyBitRev چند جمله‌ای e را محاسبه می‌کنیم.

$\hat{e} \leftarrow \mathrm{NTT}(e)$

با استفاده از تبدیل NTT و چند جمله‌ای e چند جمله‌ای $\hat{e}$ را محاسبه می‌کنیم.

$\hat{b} \leftarrow \hat{a} \circ \hat{s} + \hat{e}$

سپس چند جمله‌ای $\hat{b}$ را محاسبه می‌کند به طوریکه $\hat{b_i} = \hat{a_i} \circ \hat{s_i} + \hat{e_i}$.

$\mathrm{return}\ (pk = \mathrm{EncodePK}(\hat{b}, \mathrm{publicseed}), sk = \mathrm{EncodePolynomial}(\hat{s}))$

با استفاده از الگوریتم EncodePolynomial و $\hat{s}$ کلید خصوصی sk را محاسبه کرده و به‌صورت امن نگه‌داری می‌کند و همچنین با استفاده از الگوریتم EncodePK و مقادیر $\hat{b}$ و publicseed کلید عمومی pk را محاسبه کرده و اعلام عمومی می‌کند. در مورد جایگزینی الگوریتم‌های EncodePolynomial و DecodePolynomial فقط باید اندازه ورودی و خروجی آن‌ها هماهنگ با الگوریتم‌های EncodePK، DecodePK و همچنین الگوریتم‌های

Compress، Encode C، Decode C وDecompress باشد. جایگزینی موارد ذکر شده در صورت رعایت سازگاری ورودی و خروجی‌شان با هم مشکلی به‌وجود نمی‌آورد.

توضیح الگوریتم تولید کلید : به‌طور خلاصه الگوریتم تولید کلید یک آرایه ۳۲ بایتی را به‌عنوان ورودی دریافت می‌کند که بعد از عمل تابع چکیده ساز Shake256 این ورودی ۳۲ بایتی را به یک خروجی ٦٤ بایتی تبدیل می‌کند. سپس ۳۲ بایت اول آن را به عنوان publicseed و ۳۲ بایت آخر آن‌را به عنوان noiseseed در نظر می‌گیرد. از publicseed در الگوریتم GenA استفاده می‌کند و چندجمله‌ای $\hat{a}$را تولید می‌کند که نیاز به توزیع احتمال خاصی ندارد. اما $\hat{s},\hat{e}$ را با الگوریتم sample و آرایه noiseseed تولید می‌کند، چون که دارای توزیع احتمال ψ_8^n باشند. ولی قبل از اینکه $\hat{s},\hat{e}$ را تولید کند تبدیل NTT را در این چندجمله‌ای‌ها اجرا می‌کند، زیرا با توجه به فرآیندی که $\hat{a}$ ساخته شده است در دامنه NTT قرار می‌گیرد. سپس $\hat{s},\hat{e}$ را به نیز در دامنه NTT قرار می‌دهد به خاطر اینکه $\hat{b} = \hat{a} \circ \hat{s} + \hat{e}$ نیز در دامنه NTT قرار بگیرد و عمل ضربی که بین چندجمله‌ای $\hat{a}, \hat{s}$ است خیلی سریع انجام بشود. در نهایت در الگوریتم تولید کلید، خروجی به‌صورت یک رشته از بایت‌ها است. چرا به‌صورت یک رشته از بایت‌ها است؟ چون اگر به‌صورت یک چندجمله‌ای می‌خواست کلیدهای عمومی و خصوصی را ذخیره و اعلام عمومی کند، سخت بود. لذا با عمل کدینگ که صرفاً برای ساده سازی ذخیره در کامپیوتراست، این‌کار را انجام می‌دهد.

نکته: یک آرایه تصادفی بایتی به طول ۳۲ بایت را وقتی بخواهیم به شکل بیتی در نظر بگیریم به فرم 32×8 بیت است و برای تولید یک رشته بیتی می‌توان از روش سکه انداختن استفاده کرد که اگر رو آمد بیت مورد نظر برابر یک و اگر خط آمد بیت را برابر با صفر در نظر بگیرید. روش دیگر استفاده از روش شبه تصادفی[۱] است که یک ورودی دریافت می‌کنند و حاصل

[۱] Pseudo random

آن‌ها یک رشته بیتی تصادفی است سپس یک قطعه از این رشته را که طول آن ۳۲ بایت باشد را به‌عنوان آرایه تصادفی که نیاز داریم، بهره ببریم.

معرفی الگوریتم رمزگذاری : این الگوریتم با استفاده از کلید عمومی که یک آرایه به طول $\frac{7n}{4} +$ 32 بایت است، متن اصلی که یک آرایه به طول ۳۲ بایت است و coin که یک آرایه تصادفی بایتی به طول ۳۲ بایت است. این الگوریتم ابتدا کلید عمومی را به یک چند جمله‌ای در R_q و یک آرایه ۳۲ بایتی تبدیل می‌کند، سپس به کمک آرایه تصادفی ۲ چندجمله‌ای شبه تصادفی e', e'' تولید می‌کند. سپس متن اصلی را با کد گذاری تبدیل به یک چند جمله‌ای در R_q می‌کند. پس از رمز کردن حاصل را با یک تابع کد گذاری تبدیل به یک آرایه $\frac{3n}{8} + \frac{7n}{4}$ بایتی تبدیل می‌کند.

توضیح : این الگوریتم توسط هر شخصی که بخواهد به آلیس پیامی بفرستد اجرا می‌شود. این الگوریتم به ۳ ورودی $(pk, \mu, coin)$ نیاز دارد که به‌ترتیب به‌صورت زیر معرفی می‌شوند :

ورودی pk کلید عمومی است، ورودی μ متنی است که می‌خواهیم رمز کنیم و به‌صورت یک آرایه ۳۲ تایی است و ورودی coin که یک آرایه ۳۲ تایی تصادفی است.

$(\hat{b}, publicseed) \leftarrow DecodePk(pk)$

با استفاده از DecodePk مقادیر $\hat{b}$ و publicseed محاسبه می‌شود.

$\hat{a} \leftarrow GenA(publicseed)$

با استفاده از الگوریتم GenA و آرایه publicseed چند جمله‌ای $\hat{a}$ را می‌سازد.

$s' \leftarrow PolyBitRev(Sample(coin, 0))$

با استفاده از الگوریتم sample و مقدار coin و الگوریتم PolyBitRev چند جمله‌ای s' را محاسبه می‌کند.

$e' \leftarrow PolyBitRev(Sample(coin, 1))$

با استفاده از الگوریتم sample و مقدار coin و الگوریتم PolyBitRev چند جمله‌ای e' را محاسبه می‌کند.

$e'' \leftarrow \mathrm{Sample}(coin, 2)$

با استفاده از الگوریتم sample و مقدار coin آرایه e'' را محاسبه می‌کند.

$\hat{t} \leftarrow \mathrm{NTT}(s')$

با استفاده از تبدیل NTT و چند جمله‌ای s' چند جمله‌ای $\hat{t}$ را محاسبه می‌کند.

$\hat{u} \leftarrow \hat{a} \circ \hat{t} + \mathrm{NTT}(e')$

چندجمله‌ای $\hat{u}$ را محاسبه می‌کند (در این قسمت حاصل $\mathrm{NTT}(e')$ یک چندجمله‌ای است. همچنین حاصل $\hat{a} \circ \hat{t}$ یک چندجمله‌ای است که ضرایب آن به‌صورت $\hat{a_i}, \hat{t_i}$ محاسبه می‌شوند).

$v \leftarrow \mathrm{Encode}(\mu)$

با استفاده از الگوریتم Encode و آرایه μ ، چندجمله‌ای v محاسبه می‌شود.

$v' \leftarrow \mathrm{NTT}^{-1}(\hat{b} \circ \hat{t}) + e'' + v$

در این قسمت چند جمله‌ای v' محاسبه می‌شود (دقت داشته باشید که $\mathrm{NTT}^{-1}(\hat{a} \circ \hat{t})$ و v همگی چندجمله‌ای هستند و با هم جمع می‌شوند).

$h \leftarrow \mathrm{Compress}(v')$

با استفاده از الگوریتم Compress و چندجمله‌ای v'، مقدار h محاسبه می‌شود.

$\mathrm{return}\ c = \mathrm{EncodeC}(\hat{u}.h)$

متن رمز شده نهایی با استفاده از الگوریتم EncodeC و مقادیر h و $\hat{u}$ محاسبه شده و با نام c بازگردانده می‌شود.

توضیح الگوریتم رمز گذاری : این الگوریتم دارای سه ورودی است. اولین ورودی pk کلید عمومی است که حالت Encode شده چندجمله‌ای‌هایی که در قسمت اول تولید شده است و طول ورودی آن $32 + \frac{7n}{4}$ است. دومین ورودیμ ، متنی است که می‌خواهیم رمز کنیم که یک آرایه بایتی به طول ۳۲ بایت است. سومین ورودی coin یک آرایه تصادفی ۳۲ بایتی است. ابتدا کلید عمومی را Decode می‌کند تا شکل چندجمله‌ای که pk داشت را به ما بدهد.

سپس در قسمت Decode کردن کلید عمومی، publicseed نیز تولید می‌شود. لذا از الگوریتم GenA که یک الگوریتم قطعی[۱] بوده، استفاده می‌کنیم و چندجمله‌ای $\hat{a}$ را برای ما تولید می‌کند. حال دو چندجمله‌ای e', s' نیاز داریم که دارای توزیع احتمال ψ_8^n باشد. برای این منظور از الگوریتم sample استفاده می‌کنیم. همچنین چندجمله‌ای e'' را از الگوریتم sample تولید می‌کنیم. از آن جایی‌که s'به همین صورت برای ما کاربرد ندارد و باید در دامنه NTT باشد. لذا تبدیل NTT را روی چندجمله‌ای $\boldsymbol{s'}$ اجرا می‌کنیم و $\hat{t}$ را تولید می‌کنیم. حال $\hat{t}$ می‌تواند با $\hat{a}$ و $\hat{b}$ ضرب شود. همچنین روی چندجمله‌ای e' نیز تبدیل NTT را اجرا می‌کنیم که حاصل جمع $\hat{\mathrm{a}} \circ \hat{\mathrm{t}}$ با $\mathrm{NTT}(\mathrm{e}')$ در دامنه NTT قرار بگیرد. لذا $\hat{u}$ تولید شده در دامنه NTT قرار می‌گیرد. حال چون می‌خواهیم μ را با روش Regev رمز کنیم . پس اولین چیزی که نیاز داریم تبدیل رشته بایتی μ به یک چندجمله‌ای است که این‌کار را با استفاده از تابع Encode به یک چندجمله‌ای v تبدیل می‌کند. برای رمز کردن v هم عبارت $\mathrm{NTT}^{-1}(\hat{\mathrm{b}} \circ \hat{\mathrm{t}}) + \mathrm{e}'' +$ v را حساب می‌کند و در نهایت از الگوریتم compress استفاده می‌کند. به‌خاطر این‌که فشرده سازی کند و طول آن را کاهش دهد از الگوریتم Compress بهره می‌برد. در نهایت چیزی را که Encode کرده و می‌فرستد یعنیh و v' که تولید شده به‌صورت چندجمله‌ای هستند. چون به‌صورت چندجمله‌ای سخت است که استفاده شود. از این‌رو این چندجمله‌ای‌ها را Encode می‌کنند و c که تولید می‌شود و به‌عنوان خروجی می‌فرستند به‌صورت رشته بایتی است که طول آن $\frac{7n}{4} + \frac{3n}{8}$ است.

معرفی الگوریتم رمزگشایی : این الگوریتم برای رمزگشایی استفاده می‌شود که ورودی آن یک متن رمزشده (آرایه $\frac{7\mathrm{n}}{4} + \frac{3\mathrm{n}}{8}$ بایتی) و کلید خصوصی (آرایه $\frac{7\mathrm{n}}{4}$ بایتی) است. این الگوریتم ابتدا متن رمز شده را از حالت کد خارج کرده و سپس کلید خصوصی را از حالت کد خارج

[۱] Deterministic

کرده و سپس عمل رمز گشایی انجام گرفته و شکل کد شده متن اصلی به‌دست می‌آید که با دی‌کد کردن آن متن اصلی به‌دست می‌آید.

توضیح : این الگوریتم توسط آلیس اجرا می‌شود.

این الگوریتم دارای دو ورودی است : متن رمز شده با کلید عمومی و کلید خصوصی متناظر با آن کلید عمومی.

$(\hat{u}.h) \leftarrow \text{DecodeC}(c)$

با استفاده از تابع DecodeC و متن رمزشده c، مقادیر h و $\hat{u}$ محاسبه می‌شوند.

$\hat{s} \leftarrow \text{DecodePolynomial}(sk)$

با استفاده از الگوریتم DecodePolynomial و کلید خصوصی sk، چند جمله‌ای $\hat{s}$ محاسبه می‌شود.

$v' \leftarrow \text{Decompress}(h)$

با استفاده از الگوریتم Decompress و مقدار h، چند جمله‌ای v' محاسبه می‌شود.

$\mu \leftarrow \text{Decode}\left(v' - \text{NTT}^{-1}(\hat{u} \circ \hat{s})\right)$

return μ

ابتدا مقدار $v' - \text{NNT}^{-1}(\hat{u}\ o\ \hat{s})$ محاسبه شده و سپس با استفاده از الگوریتم Decode، متن اصلی μ محاسبه می‌شود و به‌عنوان خروجی بازگردانده می‌شود.

توضیح الگوریتم رمزگشایی : الگوریتم رمزگشایی به دو ورودی نیاز دارد. اولین ورودی متن رمزشده c است که زمانی‌که کدگشایی شده بود، یک آرایه بایتی به طول $\frac{7n}{4} + \frac{3n}{8}$ بود و یک کلید خصوصی sk که در الگوریتم تولید کلید ساخته شده بود که یک آرایه بایتی به طول $\frac{7n}{4}$ است. اولین کار این الگوریتم رمزگشایی، معکوس آخرین کاری است که در الگوریتم رمزگذاری انجام شده است. آخرین کاری که در الگوریتم رمزگذاری انجام شده بود، Encode کردن $\hat{u}$ و $\hat{h}$ بوده است. لذا اولین کاری که الان انجام می‌دهیم، دی کد کردن c است برای این‌که $\hat{u}$ و $\hat{h}$ را به‌دست بیاوریم. می‌دانیم تمام محاسباتی را که در این الگوریتم و عملگرها انجام می‌دهیم به‌صورت چندجمله‌ای‌ها است. سپس DecodePolynomial(sk) را انجام

می‌دهیم که sk یک ورودی بایتی بود. اما در این‌جا چون دارای طول مشخص است و می‌خواهیم به‌صورت قطعی باشد، به‌جای این‌که از الگوریتم Sample یا الگوریتم GenA استفاده کنیم از الگوریتم DecodePolynomial(sk) استفاده می‌کنیم. ما اینجا فقط یک تبدیل داریم و تولید نداریم، چون صرف این‌که تبدیل است از الگوریتم Decode و Encode استفاده می‌کنیم. از این‌رو از الگوریتم DecodePolynomial استفاده می‌کند و $\hat{s}$ را تولید می‌کند. چون h را هم با فشرده سازی v' تولید کرده بودیم. اکنون با عملیات تابع عدم فشردگی را روی h انجام می‌دهیم و v' را تولید می‌کنیم. سرانجام چندجمله‌ای μ را با محاسبه $(v' - \mathrm{NTT}^{-1}(\hat{u} \circ \hat{s}))$ به‌دست می‌آوریم و خود متن μ بعد از فرآیند Decode کردن $v' - \mathrm{NTT}^{-1}(\hat{u} \circ \hat{s})$ تولید می‌شود و μ برگردانده می‌شود.

نکته : در تمام چندجمله‌ای‌های به‌کار رفته در این طرح منظور از $\mathbb{Z}$ حلقه اعداد صحیح و منظور از نماد $\mathbb{Z}_q$ حلقه خارج قسمتی $\frac{\mathbb{Z}}{q\mathbb{Z}}$ است. تمامی چند جمله‌ای‌های بالا همگی از حلقه چندجمله‌ای‌های $R_q = \frac{Z_q[X]}{<X^n+1>}$ با ضرایب صحیح به پیمانه $X^n + 1$ هستند یا به‌عبارت ساده‌تر چند جمله‌ای‌های با درجه حداکثر n-1 بوده و ضرایب آنها بین ۰ تا q-1 است. یک چندجمله‌ای در R_q به شکل $g = \sum_{i=0}^{n-1} g_i X^i$ است که در آن g_i ، i–امین ضریب در چندجمله‌ای g که $i \in \{0,1, \dots, n\}$.

در جمع (و تفریق) این چندجمله‌ای‌ها ضرایب نظیر با هم جمع (تفریق) می‌شوند و در محاسبات بالا نیز چنین است با این تفاوت که جمع‌های (تفریق‌های) انجام شده ضرایب به پیمانه q محاسبه می‌شوند.

برای ضرب دو چندجمله‌ای در این حلقه از عملگر o استفاده شده است که نوعی ضرب چند جمله‌ای است به گونه‌ای که در عملگر تنها ضرایب نظیر با هم ضرب شده و به پیمانه q محاسبه می‌شوند.

یافته‌های تحقیق

در این تحقیق در حین ارائه الگوریتم از یک‌سری توابع استفاده کردیم. در این‌جا این‌گونه توابع و کاربرد آن‌ها را بیان می‌کنیم. لذا یافته‌های این توابع برای ادام مباحث بسیار حائز اهمیت است. لذا به توضیح و ارائه این یافته‌ها می‌پردازیم. الگوریتم Sample برای نمونه برداری تصادفی از R_q استفاده می‌شود که خود نیازمند یک آرایه تصادفی است. الگوریتم GenA نیز برای نمونه برداری از R_q استفاده می‌شود که در آن از فرآیندهای بازگشتی و تابع درهم ساز SHAKE128 استفاده شده است.

نکته : الگوریتم‌های EncodePolynomial (یک چندجمله‌ای را کد می‌کند و به‌صورت آرایه بایتی تبدیل می‌کند) و DecodePolynomial (که آرایه بایتی را به‌صورت چندجمله‌ای تبدیل می‌کند) قابل تعویض با هر الگوریتم کدگذاری و کدگشایی هستند (البته به شرط اینکه آن طول‌هایی را که می‌خواهیم برای ما رعایت کنند). ما در محاسباتی که داریم به صورت چندجمله‌ای نیاز داریم و در خروجی‌ها به فرم آرایه بایتی نیاز داریم. لذا نیازمند توجه و توضیح قابل تاملی (مگر در مرحله پیاده سازی) نیستند و ارتباطی با امنیت ندارند.

نکته : به محاسبات NTT، PolyBitRev، BitRev و NTT^{-1} در مرحله پیاده سازی دقت داشته باشیم که ضرایبی که محاسبه می‌شوند را درست تولید کنیم. یکی از مهم‌ترین مباحث این تحقیق اثبات درستی این رمزنگاری است. در این‌جا راجع به این درستی نتیجه می‌گیریم که این طرح کاملاً درست است و می‌توان برای طرح‌های امنیتی مورد استفاده قرار داد.

اثبات درستی رمزگشایی : با توجه به فرمول‌های گفته شده در توضیحات فوق و این نکته که چون تابع NTT خطی است، درمی‌یابیم که

$$\mathrm{NTT}(a+b)=\mathrm{NTT}(a)+\mathrm{NTT}(b)\ ,\ \mathrm{NTT}^{-1}(a+b)=\mathrm{NTT}^{-1}(a)+\mathrm{NTT}^{-1}(b)$$

لذا داریم :

$$\begin{aligned} v' - NTT^{-1}(\hat{u} \circ \hat{s}) &= NTT^{-1}(\hat{b} \circ \hat{t}) + e'' + v - NTT^{-1}(\hat{u} \circ \hat{s}) \\ &= NTT^{-1}[((\hat{a} \circ \hat{s}) + \hat{e}) \circ \hat{t}] + e'' + v \\ &- NTT^{-1}[((\hat{a} \circ \hat{t}) + NTT(e')) \circ \hat{s}] \\ &= NTT^{-1}(\hat{a} \circ \hat{s} \circ \hat{t}) + NTT^{-1}(\hat{e} \circ \hat{t}) + e'' + v - NTT^{-1}(\hat{a} \circ \hat{t} \\ &\circ \hat{s}) - NTT^{-1}(NTT(e') \circ \hat{s}) \\ &= v + NTT^{-1}(\hat{e} \circ \hat{t}) - NTT^{-1}(NTT(e') \circ \hat{s}) \end{aligned}$$

$e''' = NTT^{-1}(\hat{e} \circ \hat{t}) - NTT^{-1}(NTT(e') \circ \hat{s}) \in \psi_8^n$. می‌دانیم

برای الگوریتم‌های Encode و Decode متن اصلی برای هر μ داریم:

$Decode(Encode(\mu) + e) = \mu.$

در نتیجه

$Decode(v + e''') = Decode(Encode(\mu) + e''') = \mu$

نتیجه گیری

در سازمان‌های نظامی و دفاعی با توجه به نوع مأموریت سامانه و نیازمندی‌های عملیاتی نظیر سرعت، دقت در محاسبات و تشخیص فرامین و همچنین وجود سامانه‌های هوشمند بلادرنگ و برخط و نیز انواع سرویس‌های ذخیره‌سازی و نیاز به امنیت مربوط به هر کدام از این سامانه‌ها، می‌توان از روش‌های ترکیبی مطرح شده بهره گرفت.

در این‌صورت می‌توان با تحلیل سربار ناشی از انجام الگوریتم‌های رمزنگاری و حساست خاص عملیات نظامی و دفاعی بین سرعت انجام پردازش‌ها، محرمانگی و میزان امنیت مورد نیاز موازنه برقرار نمود.

در مواردی حتی تحلیل ترافیک سامانه نیز به نگرانی‌های مربوطه می‌افزاید و نیاز به رمز نمودن اطلاعاتی چون فرایندهای هر ریزبسته نیز احساس می‌شود. این امر سبب کاهش احتمال نفوذ و خرابکاری در نقاط حساس و گلوگاه‌های سامانه می‌شود. در احراز اصالت و سندیت فرامین و سنجه‌ها نیز می‌توان از روش‌های مطرح شده بهره گرفت و حتی علاوه بر لایه کاربرد نیز می‌توان از تکنولوزی مربوطه کمک گرفت تا مواردی چون جعل، انکار و تکرار فرامین و سنجه‌ها از لایه‌ای به لایه دیگر در تقویت کننده‌ها، مسیریاب‌ها و ایستگاه‌های فضایی یا زمینی اتفاق نیافتد. البته می‌توان با استفاده از الگوریتم‌های نظیر رمزهای اشتراکی و تشکیل گروه‌های

متعدد با سطوح دسترسی متفاوت و تولید ساختارهای دسترسی به کلیدهای رمزنگاری باعث افزایش ضریب امنیت می‌شود.

پیشنهادات

حال از محاسبات امنیتی نیز صحبت می‌کنیم که دشمن حتی با قدرت محاسبات بالا باز هم قدرت شکستن رمز را نداشته باشد، هر چند امکان شکستن رمز ممکن است ولی ما حالت عملی آن را دشوار می‌کنیم که با استفاده از محاسبات سنگین ریاضی می‌توانیم به این موضوع دست یابیم. همچنین اگر تهدیداتی که برای این طرح‌ها ممکن است به‌وجود آید را مطالعه کنیم، می‌بینیم که این طرح‌ها ممکن است در مقابل چه حملاتی شکسته شوند را بیان می‌کنیم. امنیت KEMها در برابر حملات تمایزناپذیر IND-CCA و IND-CPA را بررسی می‌کنیم. الگوریتم برقراری کلید یک پروتکل تعاملی بین دو نهاد که هدف آن تولید یک کلید نشست غیرقابل تمایز از حالت تصادفی است که بر دو نوع کلید احراز اصالت شده و اصالت نشده است. IND-CPA-KEMها نوعی طرح برقراری کلید احراز اصالت نشده هستند که در آن مهاجم منفعل است و تنها رونوشتی از پیام‌های ردوبدل شده دریافت می‌کند. لذا پیشناد می‌شود برای بالا بردن امنیت امنیت CCA بر اساس نسخه پساکوانتومی Fujisaki Okamoto ایجاد شود.

منابع

Ajtai. M, Generating hard instances of lattice problems (extended abstract), in Proceedings of the twenty-eighth annual ACM symposium on Theory of computing Philadelphia, Pennsylvania, United States: ACM (1996).

Bos. J, Ducas. L, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, and D. Stehlé. CRYSTALS – Kyberc: a CCA-secure module-lattice-based KEM. In 2018 IEEE European Symposium on Security and Privacy, EuroS&P (2018).

Brakerski. Z, A .Langlois, C .Peikert, O .Regev and D .Stehl´e .Classical hardness of learning with errors. In STOC, pages 575–584 (2013).