دوفصل‌نامه علمی منادی امنیت فضای تولید و تبادل اطلاعات (افتا)
سال ۱۴۰۲، جلد ۲۱، شماره ۱ (۳۰-۴۸)
https://monadi.isc.org.ir

تأمین محرمانگی داده‌ها و توابع بطور همزمان در محیط‌های ابری با استفاده از رمزنگاری همریخت

امین حسینقلی زاده، فرهاد رحمتی و محمد علی*

دانشکده ریاضی و علوم کامپیوتر دانشگاه صنعتی امیرکبیر، تهران، ایران.

اطلاعات مقاله

تاریخچه مقاله:
تاریخ دریافت: ۲۸ آبان ۱۴۰۲
تاریخ پذیرش: ۱۸ بهمن ۱۴۰۲
انتشار آنلاین: ۲۲ اسفند ۱۴۰۲

کلمات کلیدی:
سیستم رمزنگاری همریخت
رایانش ابری
محرمانگی اطلاعات
امنیت داده و الگوریتم

نوع مقاله: پژوهشی

چکیده

با ظهور پدیده‌های جدید در حوزه‌ی مخابرات و فناوری اطلاعات مانند رایانش ابری و شبکه‌های هوشمند، شاهد چالش‌های جدیدی در این حوزه‌ها هستیم. یکی از مهمترین این چالش‌ها محرمانگی داده‌های برون‌سپاری‌شده است. در واقع، به دلیل توان پردازشی محدود افزاره‌های هوشمند جدید، مانند تبلت و موبایل، برون‌سپاری محاسبات در این بسترها بیشتر مورد توجه کاربران قرار گرفته است. علاوه بر محرمانگی داده‌ها، امنیت الگوریتم‌های موجود در نرم‌افزارهای برخط نیز از اهمیت بسیاری برخوردار است. در نتیجه، صاحبان نرم‌افزار ممکن است نگران فاش شدن الگوریتم‌های خود پس از برون‌سپاری در محیط‌های ابری باشند. سیستم‌های رمزنگاری همریخت موجود می‌توانند محرمانگی داده‌هایی که باید به صورت برخط پردازش شوند را فراهم کنند. با این حال، محرمانگی همزمان الگوریتم‌ها در این سیستم‌ها مورد توجه قرار نگرفته است. برای حل این مسئله، ما یک سیستم رمزنگاری همریخت بنام SHDF را معرفی می‌کنیم. این سیستم قادر است تمامی الگوریتم‌های یک نرم‌افزار و داده‌هایی که قرار است روی آنها پردازش شوند را به طور همریخت رمز کند و امکان انجام محاسبات لازم را در یک سرور ناامن فراهم کند. به علاوه، نشان می‌دهیم که سیستم ارائه شده دارای امنیت اثبات‌پذیر است و نتایج پیاده‌سازی ما نیز نشان می‌دهد که قابل استفاده در محیط‌های ابری با کارایی مناسب می‌باشد.

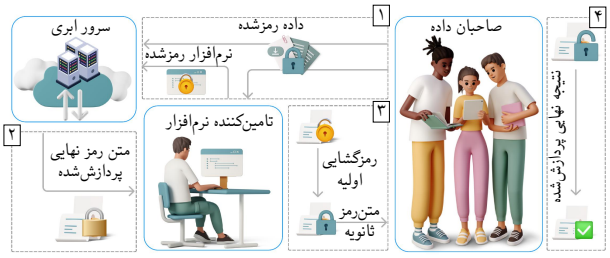
۱ مقدمه

امروزه با گسترش روزافزون نرم‌افزارهای برخط، تمایل کاربران برای استفاده از چنین ابزارهایی برای پردازش و ذخیره‌سازی اطلاعات خود گسترش یافته است. اما، به دلیل حجم بسیار بالای داده‌ها و هزینه‌های سنگین پردازش اطلاعات، استفاده از این برنامه‌ها در عمل غیرممکن است. به همین دلیل، استفاده از رایانش ابری در سال‌های اخیر بسیار مورد توجه صاحبان داده قرار گرفته است. به وسیله‌ی این فناوری، کاربران می‌توانند به ظرفیت قابلیت توجهی از توان پردازشی و ذخیره‌سازی اطلاعات خود در ابر با هزینه‌ی کمتر و بدون نیاز به داشتن امکانات ویژه اختصاصی برسند [۱]. به علاوه، این فناوری سازمان‌ها را از نصب و اجرای نرم‌افزارهای سنگین روی رایانه‌های شخصی خود بی‌نیاز کرده و مانع از اشغال شدن حجم بسیار زیادی از حافظه‌ی آنها می‌شود [۲]. چنین مزیت‌هایی موجب شده است که پدیده رایانش ابری از محبوبیت قابل توجهی بین سازمان‌ها و کاربران برخوردار باشد. بر اساس آمار منتشر شده از یک گزارش [1] نشان می‌دهد که اندازه‌ی بازار نرم‌افزار ابری در سرتاسر جهان در حدود ۱۲۱ میلیارد دلار در سال ۲۰۲۰ می‌باشد. انتظار می‌رود این میزان به حدود تقریبی ۷۰۰ میلیارد دلار در سال ۲۰۳۰ افزایش یابد.

همان طور که در شکل ۱ ملاحظه می‌کنید، یک سیستم نرم‌افزاری ابری

*نویسنده مسئول
آدرس‌های رایانامه: amin_hgholizadeh@aut.ac.ir (امین حسینقلی زاده)، frahmati@aut.ac.ir (فرهاد رحمتی)، mali71@aut.ac.ir (محمد علی)

[1] https://www.alliedmarketresearch.com/software-as-a-service-saas-market-A14951



شکل ۱. طرح سیستم رمزنگاشتی SHDF

از سه قسمت اصلی به نام‌های صاحبان داده، تأمین کنندگان نرم‌افزار و همچنین از یک سرور ابری تشکیل شده است. در این سیستم‌ها، محرمانگی داده بسیار برای صاحبان داده حائز اهمیت است. اما، همان طور که گفته شد، به دلیل حجم بسیار بالای داده‌های محرمانه ممکن است تأمین‌کنندگان نرم‌افزار فاقد توان محاسباتی لازم برای پردازش داده‌های ارسالی از جانب صاحبان داده باشند. در این شرایط، نیاز به یک سرور خارجی که بتواند با در اختیار داشتن نرم‌افزار و همچنین داده‌های محرمانه، پردازش مورد نیاز را انجام داده و نتیجه نهایی را در اختیار صاحبان داده قرار دهد، بسیار مورد توجه قرار می‌گیرد. اما، برون‌سپاری داده‌ها و نرم‌افزار به یک سرور خارجی برای انجام محاسبات دلخواه کاربران چالش‌های امنیتی زیر را به وجود می‌آورد.

چالش اول (محرمانگی داده‌ها) همان‌طور که در مراجع [۳، ۴] اشاره شده است، داده‌های برون‌سپاری شده به یک نهاد خارجی ممکن است در معرض شنود یا فاش شدن آن توسط دشمن قرار بگیرند و لذا فاقد امنیت کافی می‌باشند. به همین دلیل، صاحبان داده به دلیل محرمانگی داده‌هایشان تمایلی برای استفاده از خدمات ابری ندارند. واضح است که رمز کردن داده‌ها قبل از برون‌سپاری آنها به یک سرور ناامن این مسئله را برطرف می‌کند ولی با توجه به اینکه رمزکردن داده‌ها ساختار کلی آنها را تغییر می‌دهد، این عمل چالش جدید پردازش روی داده‌های رمز شده را به وجود می‌آورد.

چالش دوم (پردازش داده‌های رمز شده) همانطور که قبلا گفته شد، سیستم‌های رمزنگاری کلاسیک امکان پردازش روی داده‌های رمز شده را فراهم نمی‌کنند. اما، خوشبختانه سیستم‌های رمزنگاری همریخت و تماماً همریخت مشکل پردازش روی داده‌های رمز شده را برطرف می‌سازند. چنین سیستم‌هایی، داده‌های رمز شده را مبتنی بر یک الگوریتم یا یک تابع خاص پردازش کرده و نتیجه را به صورت رمز شده به صاحبان داده برمی‌گردانند. چنین خاصیت ارزشمندی، به صاحبان داده اجازه می‌دهد تا داده‌های خصوصی خودشان را به صورت امن در یک سرور خارجی پردازش کنند و در نهایت اطلاعات رمز شده را با کلید مخفی خود رمزگشایی کرده و نتیجه نهایی را به دست آورند [۵]. علی‌رغم اینکه این سیستم‌ها امنیت داده‌ها را برای پردازش در یک بستر ناامن تأمین می‌کنند، نیاز تأمین کنندگان نرم‌افزار را برای حفظ امنیت الگوریتم‌هایشان برآورده نمی‌سازند. در نتیجه، تأمین محرمانگی نرم‌افزار همچنان به عنوان یک چالش جدی پابرجا می‌ماند.

چالش سوم (محرمانگی نرم‌افزار) با توجه به گسترش روز افزون الگوریتم‌ها و اهمیت ویژه‌ی آنها در یک نرم‌افزار، صاحبان این الگوریتم‌ها نیز همانند صاحبان داده تمایلی برای به اشتراک‌گذاری الگوریتم‌هایشان در یک محیط ناامن را ندارند. به علاوه، اشتراک‌گذاری یا لو رفتن چنین الگوریتم‌هایی موجب می‌شود درآمد کسانی که چنین نرم‌افزارهایی را تولید کرده‌اند با خطر جدی روبرو شود. بنابراین، ارائه‌دهندگان خدمات نرم‌افزاری نیز تمایل دارند تا نرم‌افزار خود را به صورت رمز شده در یک سرور به اشتراک بگذارند. درنتیجه، تأمین محرمانگی تمامی محصولات نرم‌افزاری که الگوریتم‌های استفاده‌شده در آنها دارای ارزش معنوی، نظری و مبتنی بر دستاوردها و یافته‌های علمی هستند را می‌توان به عنوان یک سناریوی کاربردی در نظر گرفت. به عنوان مثال، یک شرکت نرم‌افزاری را در نظر بگیرید که در حوزه‌ی تولید محصولات مبتنی بر هوش مصنوعی فعالیت می‌کند. این شرکت به تازگی الگوریتم جدیدی را در زمینه یادگیری هوش مصنوعی ارائه کرده و چالش آن تأمین محرمانگی این الگوریتم‌ها در یک محیط ناامن می‌باشد. در این صورت شرکت فوق می‌تواند الگوریتم تولیدی خود را با هر نهاد دیگری به صورت امن به اشتراک بگذارد و درآمد قابل توجهی از بابت حق استفاده آن کسب نماید. از طرفی، با توجه به اینکه یک نرم‌افزار از کنار هم قرار گرفتن الگوریتم‌ها و توابع مختلفی تشکیل شده است، وجود یک سیستم رمزنگاری همریخت که بتواند توابع را نیز رمزگذاری کند می‌تواند این چالش را برطرف کند. روش‌هایی مانند مبهم‌سازی [1] می‌تواند ساختار یک نرم‌افزار را طوری تغییر دهد که نتیجه حاصل از پردازش داده‌ها روی نرم‌افزار تغییریافته دقیقا همان نتیجه‌ی مطلوب باشد. اما، باراک [۶] نشان داد که اعمال این روش برای تمامی توابع امکان‌پذیر نیست. یکی دیگر از روش‌های مهمی که می‌تواند امنیت نرم‌افزار را تأمین کند استفاده از مدارهای درهم قابل استفاده‌ی مجدد [2] است. در این روش، امنیت مدار و ورودی مدار تأمین می‌گردد. اما، استفاده از این روش چالش جدید اشتراک‌گذاری کلید را به وجود می‌آورد که در ادامه به توصیف آن خواهیم پرداخت.

چالش چهارم (مسئله به اشتراک گذاشتن کلید رمزگشایی) با وجود در دسترس بودن روش‌هایی مانند رمزنگاری تابعی یا مدارهای درهم قابل استفاده‌ی مجدد، این روش‌ها برای رمزگشایی نتیجه نهایی محاسبه شده توسط سرور نیازمند اشتراک‌گذاری یک کلید مخفی بین صاحبان داده و تأمین کنندگان نرم‌افزار می‌باشد. با توجه به اینکه در یک محیط ابری این دو نهاد غالباً جدا از هم و غیر قابل اعتماد هستند، اشتراک‌گذاری کلید نیازمند یک کانال امن می‌باشد که این کار ممکن است بسیار هزینه‌بر یا حتی غیرممکن باشد.

[1] Obfuscation [2] Reusable garbled circuits

۱.۱ نوآوری‌های اصلی

به منظور برطرف نمودن چالش‌های مطرح شده، در این مقاله ما یک سیستم رمزنگاشتی همریختی به نام SHDF [۱] ارائه کرده‌ایم که می‌تواند به طور همزمان داده‌ها و الگوریتم‌ها را به طور همریخت رمز کرده و آنها را برای انجام محاسبات امن در یک سرور برون‌سپاری نماید. در ادامه نوآوری‌های اصلی این مقاله را ارائه می‌کنیم.

ایجاد محرمانگی داده و نرم‌افزار: طرح پیشنهادی ما محرمانگی را برای داده‌ها و نرم‌افزار فراهم می‌کند. در واقع، برخلاف سیستم‌های رمزنگاری همریخت موجود، نرم‌افزاری که قرار است داده‌ها روی آن پردازش شوند نیز به صورت همریخت رمز شده و قابلیت پردازش در سرور را هم برای داده‌های حساس و هم برای الگوریتم‌های موجود در یک نرم‌افزار فراهم می‌کند. این ویژگی راه‌حل کاملی برای چالش‌های ۱ و ۳ می‌باشد.

برون‌سپاری محاسبات: سیستم ارائه شده ما به صاحبان داده و همچنین تأمین‌کنندگان نرم‌افزار به طور کاملا مستقل این اجازه را می‌دهد تا داده‌های محرمانه‌ی خود را برای انجام محاسبات سنگین به یک سرور با توان محاسباتی بالا برون‌سپاری کنند. با توجه به اینکه داده‌ها و الگوریتم‌ها به طور مستقل و همریخت رمز شده‌اند، سرور خارجی بدون کسب هیچگونه اطلاعاتی در مورد داده و نرم‌افزار، پردازش مورد نیاز کاربران را انجام داده و نتیجه نهایی را به صورت رمز شده به کاربران ارسال می‌کند. این ویژگی راه‌حلی برای چالش ۲ می‌باشد.

مدیریت کلید: برخلاف روش‌های موجود گفته شده در چالش چهارم که برای امنیت یک نرم‌افزار مورد استفاده قرار می‌گیرند، در روش پیشنهادی ما نیازی به اشتراک‌گذاری کلید مخفی برای رمزگشایی نتیجه‌ی بازگردانده شده از سرور وجود ندارد. رمزگشایی اولیه ابتدا توسط تأمین‌کنندگان نرم‌افزار صورت می‌گیرد و سپس صاحبان داده می‌توانند نتیجه‌ی نهایی پردازش‌شده را با کلید مخفی خود رمزگشایی کنند. لازم به ذکر است که نتیجه نهایی پس از رمزگشایی صاحبان داده حاصل می‌شود و رمزگشایی اولیه توسط تأمین‌کنندگان نرم‌افزار هیچگونه اطلاعاتی از محتوای اصلی آن فاش نمی‌سازد. این خاصیت راه‌حل مناسبی برای چالش ۴ می‌باشد.

امنیت اثبات‌پذیر: در این مقاله مدل [۲] و تعاریف امنیتی برای طرح SHDF ارائه خواهد شد. سپس، نشان می‌دهیم که SHDF دارای امنیت اثبات‌پذیر با فرض سخت بودن مسائل یادگیری باخطای چندجمله‌ای [۳]، مسئله تصمیم یادگیری با خطا [۴] و مساله تصمیم ضریب چندجمله‌ای کوچک [۵] است.

۲.۱ سازماندهی مقاله

قسمت‌های بعدی این مقاله به صورت زیر سازماندهی شده است. در بخش ۲ مرور ادبیاتی از کارهای پیشین را ارائه می‌دهیم. پیش‌نیازهای لازم برای دنبال کردن این کار در بخش ۳ ارائه شده است. در بخش ۴ معماری کلی طرح رمزنگاشتی خود را ارائه می‌کنیم و همچنین در این بخش تعاریف امنیتی و مدل مهاجم را نیز بیان می‌کنیم. در بخش ۵ ساختار کلی سیستم رمزنگاشتی خود را ارائه می‌دهیم. در بخش ۶ نشان می‌دهیم که سیستم رمزنگاشتی ما به درستی کار می‌کند و تمامی ویژگی‌های همریختی را دارد. امنیت اثبات‌پذیر بودن سیستم ارائه شده را در بخش ۷ بیان می‌کنیم. در بخش ۸ نتایج پیاده‌سازی و مقایسه سیستم رمزنگاشتی خودمان را با کارهای مشابه ارائه می‌دهیم. در نهایت، قسمت نتیجه‌گیری و کارهای آتی در بخش ۹ ارائه خواهد شد.

۲ مرور ادبیات و کارهای پیشین

این بخش یک مرور جامع ادبیات درباره استفاده از روش‌های رمزنگاری همریخت برای ارائه محرمانگی داده در خدمات نرم‌افزاری مبتنی بر ابر ارائه می‌دهد. به علاوه، تحقیقات مرتبط برای حفظ محرمانگی نرم‌افزار در سرورهای ابری ارائه می‌شوند.

۱.۲ محرمانگی داده با استفاده از رمزنگاری همریخت

با توجه به فناوری محاسبات ابری، حوزه فناوری اطلاعات کاملاً تغییر کرده است. جذابیت‌های این فناوری، مانند کاهش هزینه‌های مالی و زمانی، کاربران را علاقه‌مند به استفاده از خدمات ابری در فعالیت‌های روزمره و کسب و کار خود می‌کند [۷]. تحقیقات نشان می‌دهند که بیش از نیمی از تمامی سازمان‌ها، محاسبات ابری را به عنوان استراتژی اصلی کسب و کار خود در نظر می‌گیرند و سعی می‌کنند در خدمات مختلف آن سرمایه‌گذاری کنند [۸، ۹]. با این حال، مسئله حریم خصوصی و محرمانگی داده، به چالش بزرگی در این حوزه تبدیل شده است زیرا سرورهای ابری قابل اعتماد تلقی نمی‌شوند. طبق آمارهای گاردنر، ۷۰ درصد کاربران به دلیل نگرانی‌های امنیتی مردد به اشتراک‌گذاری داده‌های خود از طریق سرورهای ابری هستند [۱۰]. برای حل این مشکل، زانگ و همکاران [۱۱]، زنگ و همکاران [۱۲] و گوپتا [۱۳] طرح‌های رمزنگاری مبتنی بر ابر جدیدی ارائه کردند. اگرچه این رویکردها مشکل محرمانگی داده را حل می‌کنند، اما پردازش امن داده‌ها همچنان یک چالش بحرانی در این طرح‌هاست. در این‌حالت، سرور ابری باید دسترسی به کلیدهای رمزگشایی داشته باشد تا محاسبات مورد نیاز را برروی داده‌ها انجام دهد [۱۴].

طرح‌های رمزنگاری همریخت به برقراری خواص همریختی جبری بین عملیات‌های متناظر با متن رمز و متن اصلی پیام اشاره دارد. به عنوان مثال، سیستم رمزنگاری پایلیر [۶] [۱۵] خاصیت جمعی همریختی را دارد و روش RSA [۷] خاصیت ضربی همریختی را دارد. در حالی که چنین طرح‌های رمزنگاری همریخت جزئی (PHE [۸]) وجود داشته‌اند، طراحی یک طرح رمزنگاری همریخت کامل (FHE [۹]) در سال‌های گذشته یک مسئله باز بوده است. اگر چه مکانیزم سیستم رمزنگاری همریخت

[1] Simultaneous Homomorphic Encryption of Data & Function [2] Adversary model [3] Polynomial learning with errors [4] Decision-learning with errors [5] Decisional small polynomial ratio

[6] Pailier [7] Rivest-Shamir-Adleman [8] Partial Homomorphic Encryption [9] Fully Homomorphic Encryption

(HE) در دهه ۱۹۷۰ مطرح شد [۱۶]، اما تعداد محدودی از عملیات را حفظ می‌کند، مانند طرحی که توسط Boneh-Goh-Nissim (BGN) ارائه شده است [۱۷]. یک FHE محاسبات دلخواه داده‌های رمز شده را روی توابع دلخواه امکان‌پذیر می‌نماید [۱۶]. اگر چه در این بین پیشرفتی صورت گرفته است [۱۸-۲۱]، اما کار جنتری [۵] به عنوان اولین رویکرد قابل‌توجه، سی سال بعد صورت گرفت. جنتری یک طرح رمزنگاری نسبتاً همریخت (SWHE [1]) ایجاد کرد و با فشرده‌سازی مدار رمزگشایی، یک طرح FHE را به دست آورد [۵، ۲۲]. از سال ۲۰۰۹ به‌بعد، چندین طرح FHE ارائه شده است. جنتری و همکاران [۲۳، ۲۴] تعدادی از روش‌های FHE با ویژگی‌های مهم طراحی کردند. با اینکه طرح‌های اولیه سربارهای ثابت محاسباتی بالایی داشتند، اما روش‌های بعدی طرح را بهبود بخشیدند و امکان پیاده‌سازی‌های اولیه را فراهم کردند [۲۴]. با این حال، نسل اولیه طرح‌های FHE به عنوان یک پیشرفت قابل توجه مطرح شد که کارایی مناسبی نداشتند، زیرا هزینه‌های محاسباتی بالایی به همراه داشتند. برای دستیابی به طرح‌های عملی چندین روش از جمله Brakerski-Gentry-Vaikuntanathan (BGV) [۲۵] و Brakerski/Fan-Vercauteren (BFV) [۲۶، ۲۷] طراحی شدند. آنها ایده رمزنگاری همریخت سطح‌بندی شده [2] را ابداع کردند تا از هزینه‌های اجرایی ناشی از روش‌های خوداره‌اندازی [3] ارائه شده در [۲۵] پرهیز کنند.

طرح CKKS [4] [۲۸] بهینه‌سازی بیشتری را برای رمزنگاری همریخت اعداد تقریبی معرفی کرد. اگرچه ساختار آن قابل مقایسه با طرح BGV است، اما به صراحت یک طرح FHE نیست؛ زیرا عملیاتی که انجام می‌دهد فقط روی نمونه‌های تقریبی از متون رمزگذاری شده است. سپس، جنتری و همکاران [۲۹] یک طرح FHE جدید به نام GSW [5] طراحی کردند که از روش بردار ویژه تقریبی برای سرعت بخشیدن به اجرای عملیات همریختی استفاده می‌کند. علاوه بر موارد ذکر شده، روش CGGI [6] [۳۰، ۳۱] قادر است در کمتر از ۰٫۱ ثانیه فرآیند خوداره‌اندازی را انجام دهد، در حالی که خوداره‌اندازی BFV یا BGV اغلب به مدت چند دقیقه زمان نیاز دارد. برای بهبود عملکرد طرح‌های FHE، کاهش اندازه کلیدهای عمومی و مخفی یک استراتژی قابل قبول است. در این رابطه، آلبریت و همکاران [۳۲] نوعی از مسئله یادگیری با خطا (LWE) را پیشنهاد دادند که استفاده از کلیدهای با اندازه‌ی کوچکتر را ممکن می‌کند و منجر به کاهش هزینه محاسباتی طرح‌های FHE می‌شود.

در حوزه ارسال امن داده به سرورهای ابری ناامن، رمزنگاری همریخت در سال‌های اخیر توجه زیادی به خود جلب کرده است [۳۳]. رمزنگاری همریخت امکان محاسبات روی داده‌های رمزگذاری شده را فراهم می‌کند و در نتیجه، حریم خصوصی بین سرورهای ابری و کاربران داده را بهبود می‌بخشد. به همین دلیل، کاربردهای گسترده‌ای در محاسبات حفظ حریم خصوصی دارد. به عنوان مثال، لو و همکاران [۳۴] یک چارچوب مبتنی بر رمزنگاری همریخت برای حفاظت از داده‌های بزرگ و حساس پیشنهاد کرده‌اند، در حالی که لو [۳۵] یک راه‌حل حفظ حریم خصوصی با کارایی مناسب برای برنامه‌های اینترنت اشیا با استفاده از تکنیک‌های رمزنگاری همریخت BGN ارائه داده است. به علاوه، دینگ و همکاران [۳۶] یک زیرساخت محاسبات ابری امن را معرفی کردند که از رمزنگاری همریخت برای حفاظت از حریم خصوصی داده کاربر استفاده می‌کند.

۲.۲ محرمانگی نرم‌افزار

همانطور که در زیربخش قبلی اشاره کردیم، رمزنگاری همریخت امکان محاسبات روی داده‌های رمزنگاری شده را بدون نیاز به رمزگشایی فراهم می‌کند، اما در خصوص محرمانگی نرم‌افزار دچار نقص است. برای حل این مشکل، ابتدا ایده‌ی مبهم‌سازی برنامه به عنوان یک راه‌حل مطرح شد. مبهم‌سازی برنامه روشی است که در آن یک برنامه به عنوان ورودی دریافت می‌شود و یک برنامه‌ی دیگری تولید می‌شود که عملکرد آن با برنامه‌ی اصلی معادل است. هدف از مبهم‌سازی این است که تضمین شود هیچ مهاجمی در زمان چندجمله‌ای قادر به به دست آوردن هیچ اطلاعاتی درباره برنامه اصلی نباشد. با این حال، باراک و همکاران [۶] و گلدواسر و همکاران [۳۷] نشان دادند که عملیات مبهم‌سازی برنامه برای تمامی توابع به تنهایی غیرممکن است. به همین علت، رویکردهای مختلفی پیشنهاد شده‌اند تا این محدودیت‌ها را برطرف کنند [۳۸-۴۰]. مسئله مبهم‌سازی به نظر می‌رسد به طور نزدیکی با مسئله مدارهای درهم مرتبط است. مفهوم این مدارها که توسط یائو [۴۱] معرفی شده است، امکان ارزیابی یک مدار C روی متن اصلی پیام x را بدون فاش کردن هیچ اطلاعاتی به جز $C(x)$ فراهم می‌کند. با این حال، مدارهای درهم در صورت استفاده از ورودی‌های چندگانه، امنیت کافی را ندارند. برای حل این مشکل، گلدواسر و همکاران مدارهای درهم قابل استفاده‌ی مجدد (RGC) را ارائه دادند که بر اساس رمزنگاری تابعی (FE) می‌باشند [۴۲]. FE امکان دسترسی کنترل‌شده‌تری به داده‌ها را فراهم می‌کند. برای تحقق چنین دسترسی‌ای، تابع f با یک کلید مخفی مرتبط می‌شود. سپس، کاربری که کلید مخفی و یک متن رمزنگاری شده مرتبط با متن پیام m را داشته باشد، می‌تواند نتیجه $f(m)$ را بیابد [۴۲]. بونه و همکاران [۴۳] مفهوم FE را عمومی‌سازی کرده‌اند تا رمزنگاری مبتنی بر هویت (IBE) [۴۴]، رمزنگاری مبتنی بر ویژگی‌ها (ABE) [۴۵، ۴۶] و رمزنگاری مبتنی بر پیش‌بینی (PE) [۴۷] را توسعه دهند. آگراول یک ساختار RGC ارائه کرد که امنیت قوی‌تری را بر اساس فرضیه یادگیری با خطا به دست می‌دهد [۴۸]. با این حال، اگرچه FE و RGC کاربردهای فراوانی در مخفی کردن مدارها یا توابع دارند، اما به اشتراک گذاشتن کلید خصوصی بین صاحبان تابع و داده‌ها ضروری است [۴۲، ۴۳]. بنابراین، اگر صاحبان داده و صاحبان تابع دو نهاد جداگانه باشند، آن‌ها باید این کلید را با یکدیگر به اشتراک بگذارند. سیستم رمزنگاشتی ما یک راه‌حل بر اساس مسئله یادگیری با خطا و سیستم رمزنگاری NTRU [7] ارائه می‌دهد که به صاحبان داده و صاحبان تابع امکان رمزگشایی نتیجه نهایی پردازش شده

[1] Somewhat Homomorphic Encryption [2] Leveled homomorphic encryption [3] Bootstrapping [4] Cheon-Kim-Kim-Song [5] Gentry-Sahai-Waters [6] Chillotti-Gama-Georgieva-Izabachene

[7] N-th degree truncated polynomial ring units

جدول ۱. مقایسه امنیت و کارایی طرح‌های مختلف با یکدیگر

طرح	$\mathbb{F}_1$	$\mathbb{F}_2$	$\mathbb{F}_3$	$\mathbb{F}_4$	$\mathbb{F}_5$	$\mathbb{F}_6$	$\mathbb{F}_7$
[۷]	✗	✓	✗	✓	✗	✗	✗
[۸]	✗	✓	✗	✓	✗	✗	✗
[۹]	✗	✓	✗	✓	✗	✗	✗
[۵۳]	✗	✓	✓	✓	✗	✗	✗
[۵۱]	✗	✗	✓	✗	✗	✗	✗
[۲۵]	✗	✓	✓	✗	✗	✗	✗
[۴۹]	✗	✓	✓	✗	✗	✗	✗
[۴۲]	✓	✓	✓	✓	✓	✗	✗
SHDF	✓	✓	✓	✓	✓	✓	✓

$\mathbb{F}_1$: برون‌سپاری تابع رمزشده، $\mathbb{F}_2$: برون‌سپاری داده رمزشده، رمزگذاری همریخت، $\mathbb{F}_4$: رایانش ابری، $\mathbb{F}_5$: امنیت اثبات‌پذیر متناظر با سیستم رمزگذاری تأمین‌کنندگان نرم‌افزار، $\mathbb{F}_6$: رمزگذاری همریخت و همزمان داده و تابع، $\mathbb{F}_7$: مدیریت کلید.

را به طور جداگانه با استفاده از روش HE خود فراهم می‌کند، بدون اینکه کلید خصوصی را به اشتراک بگذارند.

استفاده از رمزنگاری تماماً همریخت در موارد چندکلیدی نیز امکان‌پذیر است. در یک رمزنگاری تماماً همریخت چندکلیدی (MKFHE [1])، هر موجودیت می‌تواند کلیدهای مربوط به خود را تولید کند [۴۹]. لوپز–آلت و همکاران [۵۰] یک نوع تغییر داده شده از طرح اصلی NTRU را به عنوان یک طرح MKFHE برمبنای NTRU معرفی کردند. رمزگذاری و رمزگشایی سریع ارائه شده توسط الگوریتم‌های برمبنای NTRU آنها را به عنوان یک گزینه حیاتی برای رمزنگاری پساکوانتومی تبدیل می‌کند [۵۱]. دوروز و همکاران [۵۲] ساختار حلقه‌ی پایه را تغییر دادند تا طرح MKFHE بر مبنای NTRU را بهبود بخشند. در طرح پیشنهادی ما، الگوریتم رمزنگاری همریخت مورد استفاده برای رمزگذاری تابع یک تغییر جدید از طرح رمزنگاری کلید عمومی مبتنی بر NTRU است [۵۰]. لازم به ذکر است که SHDF را می‌توان یک طرح چندکلیدی و همریخت برای سیستمی که امنیت نرم‌افزار را تأمین می‌کند درنظر گرفت. در SHDF می‌توان در صورت نیاز توابع استفاده شده در هر الگوریتم را با کلیدی جداگانه رمزگذاری کرد. در این صورت، برای رمزگشایی نتیجه پردازش‌شده کافی است حاصلضرب کلیدهای مخفی را به عنوان کلید رمزگشایی در نظر بگیریم.

مسئله LWE که توسط رِگو معرفی شد [۵۴] تأثیر قابل توجهی بر رمزنگاری تئوری و کاربردی داشته است. به همین دلیل، نتایج و کاربردهای آن در حوزه‌ی دانشگاهی مورد استفاده قرار گرفته است [۵۳]. توجه زیاد به مسئله LWE به دلیل سختی آن نسبت به مسئله تصمیم بردار کوتاه تقریبی (SVP) و مسئله بردار مستقل کوتاه (SIVP) در بدترین حالت است [۵۱]. یکی از مزایای اصلی مسئله LWE سخت بودن آن در مقابل اطلاعات اضافی مخرب و خطاها است. گلدوازر و همکاران [۵۵] اثبات کردند که وقتی یک مهاجم اطلاعات کمی درباره‌ی یک مسئله LWE دارد، سختی آن معادل زمانی است که مهاجم هیچ اطلاعاتی از آن ندارد. تعداد زیادی تحقیق قابل توجه درباره‌ی طرح رمزنگاری همریخت مبتنی بر LWE انجام شده است [۵۶–۶۰]. با این حال، همه‌ی آنها به حفظ محرمانگی تابع توجه نکرده‌اند. در SHDF، ما یک طرح رمزنگاری همریخت مبتنی بر LWE ارائه می‌دهیم که محرمانگی تابع را نیز فراهم می‌کند.

درنهایت، جدول ۱ طرح SHDF پیشنهادی را با چند طرح از طرح‌های بیان شده در این قسمت مقایسه می‌کند. همانطورکه از این جدول مشخص است SHDF تنها طرحی است که تمام ویژگی‌های امنیتی و کارایی بیان شده در این جدول را برآورده می‌سازد.

۳ پیش‌نیازها

فرض کنید $\mathcal{A}(I) \to O$ نشان‌دهنده اجرای الگوریتم $\mathcal{A}$ با ورودی I و خروجی O باشد. به علاوه، از نماد $e \leftarrow \chi$ برای نشان دادن نمونه‌برداری یک خطا e از یک توزیع خطا χ استفاده می‌شود. در این بخش، مقدماتی درباره‌ی رمزنگاری همریخت و تعاریفی مرتبط با توزیع یادگیری با خطا ارائه می‌شود.

۱.۳ رمزنگاری همریخت

فرض کنید $\mathcal{C}$ یک کلاس از مدارها و κ یک پارامتر امنیتی باشد. طرح $\Pi = (\mathbf{Gen}, \mathbf{Enc}, \mathbf{Eval}, \mathbf{Dec})$ به صورت زیر تعریف می‌شود:

- $\mathbf{Gen}(1^\kappa) \to (pk, sk, ek)$: این تابع با دریافت یک پارامتر امنیتی κ، کلید عمومی pk، کلید مخفی sk و کلید ارزیابی ek را تولید می‌کند.
- $\mathbf{Enc}(pk, (m_1, \ldots, m_\ell)) \to (c_1, \ldots, c_\ell)$: این تابع با دریافت کلید عمومی pk و متن‌های $m_1, \ldots, m_\ell$، متن رمزهای $c_1, \ldots, c_\ell$ را تولید می‌کند.
- $\mathbf{Eval}(ek, C, (c_1, \ldots, c_\ell)) \to CT$: این تابع با دریافت کلید ارزیابی ek، مدار $C \in \mathcal{C}$ و متن رمزهای $c_1, \ldots, c_\ell$، نتیجه پردازش شده CT را تولید می‌کند.
- $\mathbf{Dec}(sk, CT) \to C(m_1, \ldots, m_\ell)$: این تابع با دریافت نتیجه پردازش شده CT و کلید مخفی sk، $C(m_1, \ldots, m_\ell)$ را تولید می‌کند.

تعریف ۱ (طرح رمزنگاری همریخت). با فرض $C \in \mathcal{C}$، طرح Π یک طرح رمزنگاری همریخت خوانده می‌شود اگر برای همه‌ی متن‌های پیام $(m_1, \ldots, m_\ell)$ و متن رمزهای متناظر $(c_1, \ldots, c_\ell)$ به طوری که $c_i = \mathbf{Enc}(pk, m_i)$، رابطهٔ زیر برقرار باشد:

$$\mathbf{Dec}(sk, \mathbf{Eval}(ek, C, (c_1, \ldots, c_\ell))) = C(m_1, \ldots, m_\ell)$$

به علاوه، اگر این رابطه برای همهٔ $C \in \mathcal{C}$ برقرار باشد، طرح Π یک طرح رمزنگاری تماماً همریخت خوانده می‌شود [۵۰].

[1] Multi-Key Fully Homomorphic Encryption

۲.۳ توزیع‌های LWE و RLWE

حلقه‌ی چندجمله‌ای $R = \mathbb{Z}[x]/\langle x^n + 1\rangle$ را که در آن $n = 2^k$ برای یک $k \in \mathbb{Z}$ هست را در نظر بگیرید. پارامترهای RLWE شامل حلقه‌ی R با درجه n بر روی $\mathbb{Z}$، یک مقدار صحیح مثبت q که حلقه‌ی خارج‌قسمتی $R_q = R/qR = \mathbb{Z}_q[x]/\langle x^n + 1\rangle$ را تعریف می‌کند، و یک توزیع خطا χ بر روی R (معمولاً توزیع گوسی) می‌باشد. به عبارت دیگر، R_q شامل تمام چندجمله‌های از درجه‌ی حداکثر $(n-1)$ با ضرایب عددی است که جمع و ضرب آنها به پیمانه $(x^n + 1, q)$ تعریف شده است.

تعریف ۲ (توزیع LWE). فرض کنید $s \in \mathbb{Z}_q^n$ یک مقدار مخفی باشد، توزیع LWE یک نمونه $a \in \mathbb{Z}_q^n$ را به صورت یکنواخت و تصادفی انتخاب کرده و خطای $e \leftarrow \chi$ را استخراج می‌کند. نتیجه‌ی این توزیع $(a, b = \langle a, s\rangle + e \mod q) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ است، جایی که $\langle\cdot\rangle$ عملگر ضرب داخلی را نشان می‌دهد [۵۳].

تعریف ۳ (توزیع RLWE). فرض کنید $s \in R_q$ یک مقدار مخفی است، توزیع RLWE یک نمونه $a \in R_q$ را به صورت یکنواخت و تصادفی انتخاب کرده و $e \leftarrow \chi$ را در نظر می‌گیرد. نتیجه‌ی این توزیع $(a, b = a \cdot s + e \mod q) \in R_q \times R_q$ است [۵۳].

مسئله‌ی تصمیم یادگیری با خطای حلقه‌ای (RLWE) بیان می‌کند که بین نمونه‌های $(a_i, a_i \cdot s + e_i)$ و (a_i, b_i) تمایز قائل شویم، جایی که s، a_i و b_i از توزیع یکنواخت بر روی R_q انتخاب می‌شوند و e_i یک چندجمله‌ای نویز است که ضرایب آن به طور مستقل از χ انتخاب می‌شوند.

تعریف ۴ (یادگیری با خطای چندجمله‌ای (PLWE[1])). فرضیه‌ی PLWE می‌گوید که برای هر چندجمله‌ای $\ell = \ell(\kappa)$، تشخیص بین $\{(a_i, r_i)\}_{i=1}^{\ell}$ و $\{(a_i, a_i \cdot s + e_i)\}_{i=1}^{\ell}$ برای تمام تمایزگرهای زمان چندجمله‌ای قابل انجام نیست، جایی که برای هر $i \in \{1, \ldots, \ell\}$، $r_i \in R_q$ به صورت یکنواخت و تصادفی انتخاب می‌شود [۵۰].

تعریف ۵ (تصمیم ضریب چندجمله‌ای کوچک (DSPR[2])). فرضیه‌ی DSPR می‌گوید که موارد زیر قابل تمایز نیستند [۵۰]:

- یک چندجمله‌ای $gf^{-1} \pmod q$، جایی که f^{-1} معکوس f در R_q است.
- یک چندجمله‌ای r که به صورت یکنواخت و تصادفی از R_q انتخاب شده است.

۴ مدل‌سازی سیستم و تعاریف امنیتی

در این بخش، ابتدا مدل و ساختار کلی سیستم SHDF را معرفی می‌کنیم و سپس مدل مهاجم و تعاریف امنیتی را ارائه می‌دهیم. لازم به ذکر است نمادهایی که در این بخش استفاده شده است در جدول ۲ آمده است.

[1]Polynomial Learning With Errors [2]Decisional Small Polynomial Ratio

جدول ۲. نمادها

نماد	توضیحات
sk_d	کلید خصوصی صاحبان داده
sk_f	کلید خصوصی صاحبان تابع
ek_d	کلید ارزیابی صاحبان داده
ek_f	کلید ارزیابی صاحبان تابع
pk_d	کلید عمومی صاحبان داده
pk_f	کلید عمومی صاحبان تابع
p	مؤلفه‌ی اول pk_f استفاده شده در رمزگذاری تابع
q	عدد صحیح مثبت در حلقه خارج قسمتی
f	یک تابع
g	رمزشده تابع
m	متن اصلی پیام
c	متن رمز شده پیام
c^*	متن رمز پردازش شده
ct	متن رمز ثانویه
κ	پارامتر امنیتی

۱.۴ مدل‌سازی سیستم

همانطور که در شکل ۱ مشاهده می‌شود، طرح رمزنگاشتی ما از سه بخش اصلی تشکیل شده است. این بخش‌ها عبارت‌اند از: صاحبان داده (DOs)، تأمین کنندگان نرم‌افزار یا صاحبان تابع (FOs)، و یک سرور ابری (CSP). در ادامه هر کدام از آنها را به تفصیل توضیح می‌دهیم.

- صاحبان داده: آنها باید داده‌های محرمانه‌ی خود را بر اساس برخی توابع یا الگوریتم‌ها پردازش کنند. در این راستا داده‌های خود را رمزگذاری کرده و به CSP ارسال می کنند.
- صاحبان تابع: این نهادها نرم‌افزار و الگوریتم‌های مورد نیاز را در مرحله پردازش داده‌ها ارائه می کنند. برای فراهم کردن محرمانگی توابع و الگوریتم‌های خود، آنها توابع خود را به شکل رمزگذاری شده با CSP به اشتراک می گذارند.
- سرور ابری: فرض بر این است که این موجودیت دارای توان محاسباتی نامحدودی است. توابع و داده‌های رمزگذاری شده را از صاحبان داده و تابع دریافت می کند. مسئولیت اصلی آن پردازش داده‌ها بر اساس توابع و داده‌های دریافتی است.

به علاوه، سیستم رمزنگاشتی ما از ۵ فاز اصلی به نام‌های تولید کلید، برون‌سپاری داده‌ها، پردازش داده‌ها، رمزگشایی تابع و رمزگشایی داده تشکیل شده است. در ادامه به بیان هرکدام از آنها می‌پردازیم.

(۱) تولید کلید: در این مرحله، صاحبان داده و صاحبان تابع کلیدهای عمومی و کلیدهای مخفی خود را تولید می‌کنند. فرض بر این است که هر کلید عمومی برای عموم در دسترس است و هر کلید مخفی توسط مالک

آن محرمانه نگه داشته می شود.

(۲) برون‌سپاری: در این مرحله، در ابتدا، صاحبان داده و صاحبان تابع به ترتیب داده‌ها و توابع خود را با استفاده از طرح‌های رمزنگاشتی همریخت خود رمز می‌کنند. سپس، داده‌ها و توابع رمزگذاری شده به CSP برون‌سپاری می‌شوند. به علاوه، داده‌های رمزگذاری شده با صاحبان تابع به اشتراک گذاشته می‌شود تا یک کلید مخفی بر اساس داده‌های رمز شده را بیابند.

(۳) پردازش داده‌ها: در این مرحله، CSP ابتدا داده‌ها و توابع رمزگذاری شده دریافتی را پردازش می‌کند. سپس، نتیجه پردازش شده را به شکل رمزگذاری شده به صاحبان تابع برمی‌گرداند. توجه داشته باشید که رمزگشایی متن رمز تولید شده در این مرحله باید در دو مرحله انجام شود. ابتدا صاحبان تابع باید آن را با استفاده از کلید مخفی تولید شده در فاز (۲) رمزگشایی کنند و سپس صاحبان داده باید الگوریتم رمزگشایی خود را با کلید مخفی خودشان برای به دست آوردن داده‌های پردازش شده اجرا کنند.

(۴) رمزگشایی تابع: در این مرحله، صاحبان تابع متن رمز دریافتی از سرور را توسط کلید مخفی مربوطه‌ی تولید شده در فاز (۲) رمزگشایی می‌کنند. سپس، نتیجه را برای رمزگشایی نهایی به صاحبان داده ارسال می‌کنند.

(۵) رمزگشایی داده: در این مرحله، صاحبان داده نتیجه دریافتی از صاحبان تابع را رمزگشایی کرده و نتیجه‌ی پردازش‌شده‌ی نهایی را بازیابی می‌کنند.

۲.۴ مدل مهاجم

در سیستم رمزنگاشتی ما، فرض بر این است که CSP قابل اعتماد نیست. این نهاد ممکن است برای دسترسی غیرمجاز به محتوای داده‌ها و توابع تلاش کند. به علاوه، این نهاد در سیستم ما یک نهاد صادق اما کنجکاو در نظر گرفته می‌شود. این بدان معناست که تمام محاسبات را به درستی انجام می دهد، اما کنجکاو است که اطلاعات غیرمجاز در مورد داده‌ها و توابع بدست آورد. به علاوه، هر دوی صاحبان داده و تابع نسبت به دیگری مخرب و غیرمجاز فرض می‌شوند. آنها ممکن است سعی کنند اطلاعاتی در مورد اسرار یکدیگر کسب کنند. کانال‌های ارتباطی بین CSP و سایر نهادها امن نیستند. اطلاعات ارسال شده از طریق کانال‌ها ممکن است توسط یک استراق سمع شنود شوند. به علاوه، هیچ کانال امنی بین صاحبان داده و تابع وجود ندارد.

۳.۴ معرفی سیستم رمزنگاشتی SHDF

طرح SHDF ما شامل الگوریتم‌های زیر است:

$\mathbf{DOGen}(1^\kappa) \to (pk_d, ek_d, sk_d)$: این الگوریتم توسط صاحبان داده اجرا می‌شود. این الگوریتم با دریافت یک پارامتر امنیتی κ، کلید عمومی pk_d، کلید مخفی sk_d و کلید ارزیابی ek_d را به عنوان خروجی تولید می‌کند.

$\mathbf{FOEnc}(1^\kappa, f) \to (pk_f, ek_f, sk_f, g)$: با گرفتن پارامتر امنیتی κ و یک تابع f به عنوان ورودی، این الگوریتم کلید عمومی pk_f، کلید مخفی sk_f و کلید ارزیابی ek_f را تولید می‌کند. سپس از pk_f و f برای تولید تابع رمزنگاری شده g استفاده می‌کند. این الگوریتم خروجی pk_f، sk_f، ek_f و g را تولید می‌کند.

$\mathbf{DOEnc}(pk_d, m) \to c$: صاحبان داده این الگوریتم را اجرا می‌کنند. با دریافت کلید عمومی pk_d و متن اصلی پیام m، متن رمز c را تولید می‌کند.

$\mathbf{Eval}(c, g, ek_d, ek_f) \to c^*$: سرویس‌دهنده CSP این الگوریتم را اجرا می‌کند. با دریافت متن رمزشده‌ی c، تابع رمزگذاری‌شده g و کلیدهای ارزیابی مربوطه، ek_d و ek_f، یک متن رمز c^* را تولید می‌کند که مربوط به نتایج پردازش است.

$\mathbf{FODec}(c, c^*, sk_f) \to ct = f(c)$: صاحبان تابع این الگوریتم را اجرا می‌کنند. با دریافت متن رمزهای c و c^* و یک کلید رمزگشایی sk_f، یک متن رمز ثانویه ct را تولید می‌کنند.

$\mathbf{DODec}(ct, sk_d) \to f(m)$: صاحبان داده این الگوریتم را اجرا می‌کنند. با دریافت متن رمز ثانویه ct و کلید رمزگشایی sk_d، مقدار $f(m)$ را تولید می‌کنند.

۵ ساختار سیستم رمزنگاشتی SHDF

در این بخش، طرح رمزنگاشتی SHDF خود را به تفصیل ارائه می‌دهیم. نمادهای استفاده شده در این بخش در جدول ۲ ارائه شده است. به علاوه، شکل ۲ نحوه‌ی تعامل هر یک از نهادها با یکدیگر را پس از اجرای الگوریتم‌های متناظرشان نشان می‌دهد. همان‌طور که در زیربخش ۱.۴ ذکر شد، سیستم رمزنگاری همریخت ما در مراحل مختلف اجرا می شود: تولید کلید، برون‌سپاری، پردازش داده‌ها، رمزگشایی تابع، و رمزگشایی داده‌ها. در ادامه هر یک از مراحل را به طور جداگانه شرح می دهیم. قبل از شرح سیستم خود، لازم است چند نماد و تعاریف را معرفی کنیم.

۱.۵ مقدمات اولیه سیستم SHDF

دو فرضیه‌ی بنیادی درباره‌ی توابع موجود در سیستم SHDF وجود دارد:

- همه‌ی توابع f در حلقه‌ی $R = \mathbb{Z}[[x]]$ قرار دارند، جایی که
$$\mathbb{Z}[[x]] = \lim_{n\to\infty} R_n = \lim_{n\to\infty} \mathbb{Z}[x]/\langle x^n\rangle.$$
- هر تابع f تحلیلی است و دارای یک سری توانی با ضرایب کمتر از B است. به عبارت دیگر، تمام توابع f می‌توانند به صورت $f(x) = \sum_{i=1}^{\infty} a_i x^i$ نمایش داده شوند، جایی که $a_i \in \mathbb{Z}$ و برای یک عدد صحیح B داریم $a_i \le B$.

می‌توان به‌راحتی دید که کران B نتیجه‌ی همگرایی دنباله‌ی ضرایب $(a_i)_{i\in\mathbb{N}}$ است. در واقع، اگر $(a_i)_{i\in\mathbb{N}}$ یک دنباله‌ی همگرا به عدد صحیح a باشد، آنگاه برای هر $\varepsilon > 0$، یک عدد صحیح N وجود دارد به طوری که اگر $i > N$ باشد آنگاه $|a_i - a| < \varepsilon$ است. بنابراین، $a_i \in \mathcal{N}_\varepsilon(a)$ برای همه‌ی $i \in \mathbb{N}$، جایی که $\mathcal{N}_\varepsilon(a)$ همسایگی a با شعاع ε است. بنابراین، تمام جملات سری توانی فوق در $\mathcal{N}_r(x)$ قرار دارند، جایی که $r = \max\{\varepsilon, a_1, \ldots, a_N\}$.

فرض کنید f یک چندجمله‌ای از درجه t باشد که شرایط فوق را برآورده می‌کند، سه مقدار α، α' و β که با f مرتبط هستند را به صورت زیر در نظر بگیرید. α' بزرگترین ضریب تابع f است، $\alpha > \max\{q^t, \alpha'\}$ و $\beta > \alpha$ که بسیار بزرگتر از B است. فرض کنید $\mathcal{G}$ و $\mathcal{G}'$ دو الگوریتم احتمالی باشند به طوری که برای یک پارامتر امنیتی κ داشته باشیم $\mathcal{G}(1^\kappa, f) \to (\alpha, \beta, R_\beta, \chi_\beta)$ و $\mathcal{G}'(1^\kappa) \to (n, q, R_q, \chi_q)$، جایی که α و β همان مقادیر فوق هستند، $R_\beta = R/\beta R$ یک حلقه‌ی خارج‌قسمتی است، $k = k(\kappa)$ و $q = q(\kappa)$ دو مقدار صحیح هستند، $n = 2^k$، $R_q = R'/qR'$ یک حلقه‌ی خارج‌قسمتی دیگر است که $R' = \mathbb{Z}[x]/\langle x^n + 1\rangle$ و χ_β و χ_q توزیع‌های نرمال گاوسی با واریانس‌های کوچک به ترتیب بر روی R_β و R_q هستند. به علاوه، فرض کنید $[\cdot]_\beta$ یک تابع پیمانه‌ای بوده که ضرایب حلقه R را به مجموعه‌ای از اعداد $\{-\lfloor\beta/2\rfloor, \ldots, \lfloor\beta/2\rfloor\}$ می‌نگارد.

۲.۵ تولید کلید

در این مرحله، صاحبان داده و تابع ابتدا در مورد یک پارامتر امنیتی κ توافق می‌کنند. سپس، تابع $\mathbf{DOGen}(1^\kappa) \to (pk_d, ek_d, sk_d)$ توسط صاحبان داده اجرا می‌شود تا کلیدهای متناظر تولید شوند. کلید عمومی pk_d در دسترس عمومی قرار می‌گیرد، کلید مخفی sk_d محرمانه نگهداری می‌شود، و کلید ارزیابی ek_d با سرور به اشتراک گذاشته می‌شود.

- $\mathbf{DOGen}(1^\kappa)$: صاحبان داده با دریافت یک پارامتر امنیتی κ، ابتدا $\mathcal{G}'(1^\kappa) \to (n, q, R_q, \chi_q)$ را اجرا می‌کنند و $s \leftarrow R_q$، $a \leftarrow R_q$، $\tilde{a} \leftarrow R_q$، $e \leftarrow \chi_q$ و $\tilde{e} \leftarrow \chi_q$ را برمی‌گزینند. سپس یک کلید مخفی $sk_d = s$، یک کلید عمومی $pk_d = (pk_\ell, R_q)$ $(a, b = as + 2e, R_q)$ و یک کلید ارزیابی $ek_d = (\tilde{a}, \tilde{a}s + 2\tilde{e})$ را تولید می‌کنند.

۳.۵ برون‌سپاری

ابتدا، الگوریتم‌های $\mathbf{DOEnc}(pk_d, m) \to c$ و

$$\mathbf{FOEnc}(1^\kappa, f) \to (pk_f, sk_f, ek_f, g)$$

به ترتیب توسط صاحبان داده و تابع اجرا می‌شوند تا داده‌ها و توابع خود را رمزگذاری کنند. سپس، داده رمزشده‌ی c و تابع رمزشده‌ی g به منظور پردازش بیشتر به سرور برون‌سپاری می‌شوند. الگوریتم‌های مذکور در زیر توضیح داده شده‌اند.

- $\mathbf{DOEnc}(pk_d, m)$: این الگوریتم دو ورودی پیام m و کلید عمومی pk_d را دریافت می‌کند و در خروجی، متن رمزگذاری شده

$$c = (c_0 = -pk_{d0}, c_1 = pk_{d1} + m)$$

را تولید می‌کند، که در آن pk_{d0} و pk_{d1} به ترتیب مؤلفه‌های اول و دوم pk_d هستند.

- $\mathbf{FOEnc}(1^\kappa, f)$: با دریافت پارامتر امنیتی κ به عنوان ورودی، این الگوریتم ابتدا $\mathcal{G}(1^\kappa, f) \to (\alpha, \beta, R_\beta, \chi_\beta)$ را اجرا می‌کند و چند جمله‌ای‌هایی با ضرایب کوچک به نام h و t انتخاب می‌کند، به نحوی که h دارای وارونی به پیمانه‌ی β باشد و مقدار h به پیمانه‌ی α برابر ۱ باشد. وارون h به پیمانه β را با h_β نشان می‌دهیم. این الگوریتم در ابتدا $pk_f = (p, p') = (h_\beta \times t (\mathrm{mod}\, \beta), R_\beta)$، $sk_f = h$ و $ek_f = [\alpha\tilde{\varphi}p + \alpha\tilde{e} + h]_\beta$ تولید می‌کند، جایی که $\tilde{\varphi}, \tilde{e} \leftarrow \chi$ هستند. سپس $\varphi, e \leftarrow \chi_\beta$ انتخاب می‌شوند و $g = [\alpha\varphi \times p + \alpha e + f]_\beta$ محاسبه می‌شود.

تذکر ۱. توجه کنید که فرآیند تولید کلید و رمزگذاری تابع در الگوریتم **FOEnc** صورت می‌پذیرد. این الگوریتم با دریافت پارامتر امنیتی و تابع f به عنوان ورودی، ابتدا کلیدهای مربوطه را تولید کرده و سپس براساس کلیدهای تولید شده تابع f را رمزگذاری می‌نماید.

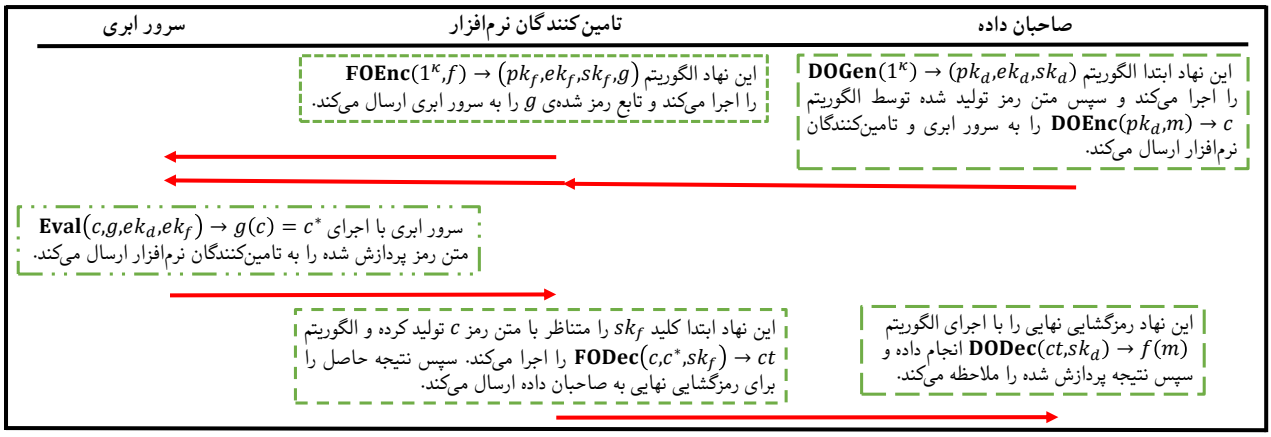
۴.۵ پردازش داده‌ها

در این مرحله، با استفاده از کلیدهای ارزیابی، ek_d و ek_f، سرور الگوریتم $\mathbf{Eval}(c, g, ek_d, ek_f) \to c^*$ را به منظور محاسبه‌ی $g(c)$ اجرا می‌کند. با توجه به اینکه نویز موجود در متون رمز تولید شده پس از انجام عملیات روی آنها افزایش می‌یابد، این الگوریتم با استفاده از کلیدهای ارزیابی متناظر با صاحبان داده و تابع، ek_d و ek_f، و با بهره‌گیری از روش‌های کنترل نویز پردازش لازم را انجام می‌دهد. در نتیجه مقدار تابع رمزگذاری‌شده g روی متن رمزگذاری شده c در سرور به درستی محاسبه گردیده و مقدار $g(c)$ به دست می‌آید. مقدار پردازش شده $c^* = g(c)$ به صاحبان تابع ارسال می‌شود تا متن رمزگذاری شده ثانویه ct را محاسبه کنند. توجه داشته باشید که برای انجام عملیات پردازش در سرور، از کلیدهای ارزیابی ek_d و ek_f مطابق با [۵۰] استفاده می‌شود.

۵.۵ رمزگشایی تابع

ابتدا صاحبان تابع پس از دریافت متن رمزشده‌ی c^* از سرور الگوریتم $\mathbf{FODec}(c, c^*, sk_f) \to ct$ را با کلید مخفی خود به منظور به دست آوردن متن رمز ثانویه ct اجرا می‌کنند. سپس، متن رمز ثانویه‌ی تولید شده را برای رمزگشایی نهایی به صاحبان داده می‌فرستند.

- $\mathbf{FODec}(c, c^* = g(c), sk_f)$: این الگوریتم با دریافت متن رمزگذاری شده c، مقدار c^* و یک کلید مخفی $sk_f = h$ به عنوان ورودی، $ct = \Big[[h(c) \times g(c)]_\beta\Big]_\alpha$ را به عنوان خروجی برمی‌گرداند.



شکل ۲. فلوچارت مرتبط با سیستم رمزنگاشتی SHDF.

۶.۵ رمزگشایی داده

در این مرحله صاحبان داده پس از دریافت متن رمز ثانویه‌ی ct از صاحبان تابع با اجرای الگوریتم $\mathbf{DODec}(ct, sk_d) \to f(m)$ نتیجه پردازش دلخواه $f(m)$ را بازیابی می‌کند.

- $\mathbf{DODec}(ct, sk_d)$: این الگوریتم متن رمز ثانویه $ct = (c'_0, c'_1, \ldots, c'_t)$ و یک کلید مخفی s را به عنوان ورودی دریافت می‌کند. خروجی آن مقدار $\sum_{i=0}^{t} c_i s^{t-i} \mod 2$ است که برابر با $f(m)$ می‌باشد، که درآن t بیشینه طول متن رمز ثانویه است.

تذکر ۲. توجه کنید که طول متن رمزگذاری شده c پس از هر عملیات ضرب طبق معادله‌ی زیر افزایش می‌یابد. با در نظر گرفتن دو متن رمز $c = (c_0, \ldots, c_t)$ و $c' = (c'_0, \ldots, c'_t)$، ما

$$c_{\text{add}} = c + c' = (c_0 + c'_0, \ldots, c_t + c'_t) \in R_q^{t+1}$$

را تعریف می‌کنیم. به علاوه، فرض کنید $c = (c_0, \ldots, c_t)$ و $c' = (c'_0, \ldots, c'_{t'})$ دو متن رمز متمایز باشند و s متغیر نمادین (کلید خصوصی در ساختار الگوریتم $\mathbf{DOEnc}$) باشد. متون رمز فوق روی R_q را می‌توان به صورت $c = \sum_{i=0}^{t} c_i s^{t-i}$ و $c' = \sum_{i=0}^{t'} c'_i s^{t'-i}$ در نظر گرفت. در این صورت c_{mult} به شکل زیر محاسبه می‌شود:

$$cc' = \left(\sum_{i=0}^{t} c_i s^{t-i}\right)\left(\sum_{i=0}^{t'} c'_i s^{t'-i}\right) = \sum_{i=0}^{t+t'} \hat{c}_i s^{(t+t')-i}$$

رمزنگاری مطلوب پس از عملیات ضرب $c_{\text{mult}} = (\hat{c}_0, \ldots, \hat{c}_{t+t'})$ است [۵۳]. بنابراین، ما در الگوریتم $\mathbf{DODec}$ متن رمز ثانویه را $ct = (c'_0, c'_1, \ldots, c'_t)$ درنظر می‌گیریم.

۶ اثبات درستی الگوریتم ها و بررسی ویژگی‌های همریختی آن‌ها

در این بخش، درستی سیستم SHDF را ثابت می‌کنیم. سپس، نشان می‌دهیم که الگوریتم‌های مرتبط با صاحبان داده و تابع دارای خاصیت همریختی نیز هستند.

۱.۶ الگوریتم‌های متناظر با صاحبان داده

در این بخش، نشان می‌دهیم که متن‌های رمزگذاری شده‌ای که از الگوریتم $\mathbf{DOEnc}$ به دست می‌آیند، نسبت به جمع و ضرب حلقه‌ی چندجمله‌ای R_q بسته هستند. فرض کنید m و m' پیام‌های متمایزی باشند،

$$\mathbf{DOEnc}(pk_d, m) \to c = (c_0, c_1) = (-a, as + 2e + m)$$

و

$$\mathbf{DOEnc}(pk_d, m') \to c' = (c'_0, c'_1) = (-a', a's + 2e' + m'),$$

که در آن $a, a', s \leftarrow R_q$ و $e, e' \leftarrow \chi_q$ هستند. در ادامه، اثبات می‌کنیم که $\mathbf{DODec}(c_{\text{add}} = c + c', s) = m + m'$ و $\mathbf{DODec}(c_{\text{mult}} = cc', s) = mm'$. در این صورت داریم:

$$\begin{aligned} c_{\text{add}} &= (c_0 + c'_0, c_1 + c'_1) \\ &= (-(a + a'), (a + a')s + 2(e + e') + (m + m')). \end{aligned}$$

بنابراین،

$$\begin{aligned} \mathbf{DODec}(c_{\text{add}}, s) &= -(a + a')s + ((a + a')s \\ &\quad + 2(e + e') + (m + m') \\ &= m + m' \bmod 2. \end{aligned}$$

در نتیجه، الگوریتم رمزنگاری دارای خاصیت همریختی جمعی می‌باشد.

برای اثبات ویژگی همریخت ضربی، نشان می‌دهیم که

$$c_{\text{mult}} = (c_0, c_1) \cdot (c'_0, c'_1) = (\tilde{c}_0, \tilde{c}_1, \tilde{c}_2)$$

به mm' رمزگشایی می‌شود. توجه کنید که پس از هر عملیات ضرب، طول متن رمزگذاری شده بر اساس رابطه (۲) افزایش می‌یابد. برای راحتی، برای اثبات ادعای بالا، متن‌های رمز شده با طول دو را در نظر می‌گیریم. در نتیجه، با در نظر گرفتن رابطه زیر، ویژگی همریختی ضربی برای متن‌های رمزشده با طول دلخواه را اثبات می‌کنیم. درنتیجه:

$$\begin{aligned}\tilde{c}_0 &= aa'\\ \tilde{c}_1 &= c_1a' + c_1'a\\ \tilde{c}_2 &= aa's^2 + 2ase' + 2a'se + asm' + a'sm\\ &\quad + 2(2ee' + e'm + em') + mm'.\end{aligned}$$

لذا رابطه زیر ادعای ما را ثابت می‌کند.

$$\begin{aligned}\mathbf{DODec}(c_{\text{mult}}, s) &= \tilde{c}_0 s^2 + \tilde{c}_1 s + \tilde{c}_2 \mod 2\\ &= aa's^2 + (2aa's + a'm + am')s\\ &\quad + (aa's^2 + asm' + a'sm + mm')\\ &= mm' \mod 2.\end{aligned}$$

توجه کنید که c_{mult} و c_{add} به ترتیب سه و دو مؤلفه دارند. این ممکن است برای انجام پردازش‌های ترکیبی مانند $cc' + c''$ ابهام ایجاد کند. برای رفع ابهام تأکید می‌کنیم که شکل چندجمله‌ای هر متن رمزگذاری شده باید در فرایند رمزگشایی مورد توجه قرار گیرد. بنابراین، با فرض اینکه

$$\begin{aligned}c &= (c_0, c_1) = (a, as + 2e + m),\\ c' &= (c_0', c_1') = (a', a's + 2e' + m'),\\ c'' &= (c_0'', c_1'') = (a'', a''s + 2e'' + m'')\end{aligned}$$

داریم:

$$\begin{aligned}&\mathbf{DODec}(cc' + c'', s) \qquad (1)\\ &= \mathbf{DODec}((c_0c_0', c_0c_1' + c_0'c_1, c_1c_1') + (c_0'', c_1''), s)\\ &= aa's^2 + (2aa's + a'm + am')s\\ &+ (aa's^2 + asm' + a'sm + mm') + c_0''s + c_1'' \mod 2\\ &= mm' + a''s + a''s + 2e'' + m'' \mod 2\\ &= mm' + m''.\end{aligned}$$

۲.۶ الگوریتم‌های متناظر با صاحبان تابع

در این زیربخش، ابتدا نشان می‌دهیم که الگوریتم‌های رمزنگاری و رمزگشایی متناظر با صاحبان تابع به درستی کار می‌کنند. بنابراین داریم:

$$\begin{aligned}&\mathbf{FODec}(c, c^* = g(c), sk_f) = \Big[[h(c) \times g(c)]_\beta\Big]_\alpha\\ &= [(h(c) \times \alpha\varphi(c) \times p(c)) + (h(c) \times \alpha e)\\ &\quad + (h(c) \times f(c))]_\beta\\ &= [(h(c) \times \alpha\varphi(c) \times h_\beta(c) \times t(c))\\ &\quad + (h(c) \times \alpha e) + (h(c) \times f(c))]_\beta.\end{aligned}$$

حال، با توجه به اینکه h_β و h وارون یکدیگر به‌پیمانه‌ی β هستند و β به طور قابل توجهی بزرگتر از ضرایب هر چندجمله‌ای در R_β و χ_β است که بین $-\beta/2$ تا $\beta/2$ واقع می‌شود، کاهش به پیمانه‌ی β ضرایب را تغییر نمی‌دهد. بنابراین:

$$\begin{aligned}&\mathbf{FODec}(c, c^* = g(c), sk_f)\\ &= [\alpha\varphi(c) \times t(c) + h(c) \times \alpha e + h(c)f(c)]_\alpha.\end{aligned}$$

در نهایت، به دلیل اینکه $h \equiv 1 \pmod{\alpha}$ است، تابع رمزشده g به $f(c)$ رمزگشایی می‌شود. این اثباتی برای صحت فرآیندهای رمزنگاری و رمزگشایی متناظر با صاحبان تابع می‌باشد.

حال نشان می‌دهیم که الگوریتم رمزنگاری دارای خاصیت همریختی جمعی است. فرض کنید g و g' دو متن رمزگذاری شده متمایز باشند که توسط الگوریتم $\mathbf{FOEnc}$ تولید شده‌اند. بنابراین،

$$\begin{aligned}\mathbf{FOEnc}(1^\kappa, f) &\to (pk_f, sk_f, ek_f, g),\\ \mathbf{FOEnc}(1^\kappa, f') &\to (pk_f, sk_f, ek_f, g').\end{aligned}$$

ما نشان می‌دهیم که $\mathbf{FODec}(g_{\text{add}} = g + g', sk_f = h) = f + f'$. فرض کنید α بزرگترین ضرایب f و f' باشد و $\beta > \alpha$ باشد. بنابراین، $g = [\alpha\varphi p + \alpha e + f]_\beta$ و $g' = [\alpha\varphi' p + \alpha e' + f']_\beta$. حال،

$$\begin{aligned}g_{\text{add}} &= g + g'\\ &= [\alpha\varphi p + \alpha e + f]_\beta + [\alpha\varphi' p + \alpha e' + f']_\beta\\ &= [\alpha\varphi p + \alpha e + f + \alpha\varphi' p + \alpha e' + f']_\beta.\end{aligned}$$

بنابراین، باید $\Big[[hg_{\text{add}}]_\beta\Big]_\alpha$ را برای رمزگشایی g_{add} محاسبه کنیم. چون $p = h_\beta \times t$ و β بزرگتر از تمام ضرایب است، داریم:

$$\begin{aligned}&[(h \times \alpha\varphi p) + (h \times \alpha e) + (hf) + (h \times \alpha\varphi' p) \qquad (2)\\ &\quad + (h \times \alpha e') + (hf')]_\alpha\\ &= [h(f + f')]_\alpha = f + f.'\end{aligned}$$

ثابت کردیم که الگوریتم رمزنگاری دارای خاصیت همریختی جمعی است. حال نشان می‌دهیم که خاصیت همریختی ضربی را نیز دارا می‌باشد. برای این منظور فرض کنید g و g' دو تابع رمزگذاری شده متناظر با f و f' هستند. در این صورت داریم:

$$\begin{aligned}g_{\text{mult}} &= g \times g' = [\alpha\varphi p + \alpha e + f]_\beta \times [\alpha\varphi' p + \alpha e' + f']_\beta\\ &= [(\alpha\varphi p + \alpha e + f)(\alpha\varphi' p + \alpha e' + f')]_\beta.\end{aligned}$$

نشان می‌دهیم $\mathbf{FODec}(g_{\text{mult}} = g \times g', h) = ff'$. بنابراین داریم:

$$\begin{aligned}&\Big[[hg_{\text{mult}}]_\beta\Big]_\alpha \qquad (3)\\ &= \left[h\alpha^2\varphi\varphi' p^2 + h\alpha^2\varphi pe' + h\alpha\varphi pf' + h\alpha^2\varphi' pe\right]_\alpha\\ &+ \left[h\alpha^2 ee' + h\alpha ef' + h\alpha\varphi' pf + h\alpha e'f + hff'\right]_\alpha\\ &\overset{h\overset{\alpha}{\equiv}1}{=} ff'.\end{aligned}$$

حال برای اینکه نشان دهیم که اثر تابع رمزشده روی داده‌های رمزشده نیز به درستی رمزگشایی می‌شوند رابطه $(gg'+g'')(cc'+c'')$ را با الگوریتم‌های مربوطه رمزگشایی می‌نماییم.

$$\begin{aligned}&\mathbf{FODec}\left(\mathbf{DODec}\left((gg'+g'')(cc'+c'')\right)\right)\\&\overset{(۱)}{=}\mathbf{FODec}\left((gg'+g'')(mm'+m'')\right)\\&\overset{(۲),(۳)}{=}(ff'+f'')(mm'+m'')\end{aligned}$$

در نتیجه اثر تابع رمزشده‌ی فوق روی متن رمز به دست آمده از عملیات همریختی به درستی رمزگشایی می‌شود.

تاکنون، ما می‌توانیم هر تابع بی‌نهایت بار مشتق‌پذیر را رمز کنیم. اکنون می‌خواهیم هر تابع پیوسته‌ای را رمزگذاری کنیم. در این رابطه، فرض کنید f یک تابع پیوسته باشد. بنابراین، یک دنباله‌ی چندجمله‌ای $(f_n)_{n\in\mathbb{N}}$ وجود دارد به طوری که $\lim_{n\to\infty} f_n = f$. با ثابت فرض کردن κ، φ، e و c، الگوریتم‌های $\mathbf{FOEnc}$ و $\mathbf{FODec}$ را می‌توان به عنوان توابع پیوسته در نظر گرفت. بنابراین، ما داریم:

$$\begin{aligned}&\mathbf{FODec}\left(c,\mathbf{FOEnc}(pk_f,f)(c),sk_f\right)\\&=\mathbf{FODec}(c,\lim_{n\to\infty}\mathbf{FOEnc}(pk_{f_n},f_n)(c),sk_{\lim_{n\to\infty}f_n})\\&=\lim_{n\to\infty}\mathbf{FODec}\left(c,\mathbf{FOEnc}(pk_{f_n},f_n)(c),sk_{f_n}\right)\end{aligned}$$

در نتیجه، با این روش توانستیم هر تابع پیوسته‌ای را به صورت همریخت رمز کنیم.

۷ تحلیل امنیت

در این بخش، ابتدا تعاریف امنیتی مربوطه را بیان می‌کنیم و سپس نشان می‌دهیم که طرح SHDF با توجه به تعاریف امنیت بیان شده دارای امنیت اثبات‌پذیر است.

۱.۷ تعاریف امنیتی

فرض کنید Π و $\mathcal{A}$ نمایش دهنده‌ی طرح SHDF پیشنهادی ما و یک مهاجم زمان چندجمله‌ای و احتمالاتی (PPT) باشند. به آزمایشات و تعریف‌های زیر توجه کنید:

آزمایش $\mathbf{SHDF}^{\mathbf{DO}}_{\Pi,\mathcal{A}}(\kappa)$

(۱) **فاز ۱:** یک چالشگر $\mathbf{DOGen}(1^\kappa)\to(pk_d,ek_d,sk_d)$ را اجرا می‌کند و کلید عمومی pk_d را به $\mathcal{A}$ می‌دهد.

(۲) **چالش:** مهاجم $\mathcal{A}$ دو پیام با طول برابر m_0 و m_1 را به چالشگر می‌دهد.

(۳) **فاز ۲:** چالشگر یک بیت تصادفی $b\in\{0,1\}$ را انتخاب کرده و $\mathbf{DOEnc}(pk_d,m_b)\to c$ را اجرا می‌کند. سپس متن رمز چالش c را به مهاجم $\mathcal{A}$ ارسال می‌کند.

(۴) **حدس:** مهاجم $\mathcal{A}$ بیت $b'\in\{0,1\}$ را بازمی‌گرداند.

می‌گوییم که مهاجم $\mathcal{A}$ بازی را برنده می‌شود اگر $b=b'$ باشد. خروجی آزمایش فوق را ۱ تعریف می‌کنیم اگر مهاجم بازی را ببرد و در غیر این صورت ۰ تعریف می‌نماییم. ما $\mathbf{SHDF}^{\mathbf{DO}}_{\Pi,\mathcal{A}}(\kappa)=1$ می‌نویسیم و می‌گوییم که $\mathcal{A}$ موفق می‌شود اگر $b=b'$ باشد.

تعریف ۶. می‌گوییم که طرح Π امنیت محرمانگی داده را فراهم می‌کند اگر برای همه‌ی مهاجم‌های چندجمله‌ای $\mathcal{A}$ داشته باشیم:

$$\Pr\left(\mathbf{SHDF}^{\mathbf{DO}}_{\Pi,\mathcal{A}}(\kappa)=1\right)\le 1/2+\mathrm{negl}(\kappa),$$

که negl یک تابع ناچیز است و احتمال بر روی تصادفی‌گری استفاده شده توسط $\mathcal{A}$ گرفته می‌شود [۶۱].

فرض کنید Π و $\mathcal{A}$ به ترتیب بیانگر طرح SHDF ما و یک مهاجم زمان چندجمله‌ای و احتمالاتی باشند. به مراحل و تعریف‌های زیر توجه کنید:

آزمایش $\mathbf{SHDF}^{\mathbf{FO}}_{\Pi,\mathcal{A}}(\kappa)$

(۱) **فاز ۱:** چالشگر $\mathbf{FOEnc}(1^\kappa,f)\to(pk_f,ek_f,sk_f,g)$ را اجرا می‌کند و کلید عمومی pk_f را به $\mathcal{A}$ می‌دهد.

(۲) **چالش:** مهاجم $\mathcal{A}$ دو تابع با طول برابر f_0 و f_1 را به چالشگر برمی‌گرداند.

(۳) **فاز ۲:** چالشگر یک بیت تصادفی $b\in\{0,1\}$ را انتخاب کرده و $\mathbf{FOEnc}(1^\kappa,f_b)\to(pk_f,ek_f,sk_f,g)$ را اجرا می‌کند. سپس تابع رمزشده‌ی چالش g را به مهاجم $\mathcal{A}$ ارسال می‌کند.

(۴) **حدس:** مهاجم $\mathcal{A}$ بیت $b'\in\{0,1\}$ را بازمی‌گرداند.

می‌گوییم که مهاجم $\mathcal{A}$ بازی را برنده می‌شود اگر $b=b'$ باشد. خروجی آزمایش فوق را ۱ تعریف می‌کنیم اگر مهاجم بازی را ببرد و در غیر این صورت ۰ تعریف می‌نماییم. ما $\mathbf{SHDF}^{\mathbf{FO}}_{\Pi,\mathcal{A}}(\kappa)=1$ می‌نویسیم و می‌گوییم که $\mathcal{A}$ موفق می‌شود اگر $b=b'$ باشد.

تعریف ۷. می‌گوییم که طرح Π امنیت تابع را فراهم می‌کند اگر برای همه‌ی مهاجم‌های زمان چندجمله‌ای $\mathcal{A}$ داشته باشیم:

$$\Pr\left(\mathbf{SHDF}^{\mathbf{FO}}_{\Pi,\mathcal{A}}(\kappa)=1\right)\le 1/2+\mathrm{negl}(\kappa),$$

که negl یک تابع ناچیز است و احتمال بر روی تصادفی‌گری استفاده شده توسط $\mathcal{A}$ گرفته می‌شود [۵۰].

۲.۷ اثبات امنیت

در این زیر بخش با اثبات دو قضیه نشان می‌دهیم که سیستم رمزنگاشتی متناظر با صاحبان داده و صاحبان تابع داری امنیت اثبات‌پذیر است.

قضیه ۱. طرح رمز نگاشتی SHDF محرمانگی را داده را براساس تعریف ۶ فراهم می‌کند.

اثبات. فرض کنید Π طرح رمزنگاری همریخت SHDF باشد و $\mathcal{A}$ یک مهاجم چندجمله‌ای در آزمایش $\mathbf{SHDF}^{\mathbf{DO}}_{\Pi,\mathcal{A}}(\kappa)=1$ باشد که در

زیربخش ۱.۷ تعریف شده است. اثبات می‌کنیم که تابع ناچیز $negl$ وجود دارد به طوری که:

$$\Pr\left(\mathrm{SHDF}^{\mathrm{DO}}_{\Pi,\mathcal{A}}(\kappa)=1\right)\leq 1/2+negl(\kappa)$$

که در آن κ یک پارامتر امنیتی دلخواه است.

فرض کنید $\mathcal{B}$ یک مهاجم مسئله DRLWE باشد. یک چالشگر در تعامل با $\mathcal{B}$ در نظر بگیرید که ابتدا $\mathcal{G}'(1^\kappa)\rightarrow(n,q,R_q,\chi_q)$ را اجرا می‌کند و سپس سکه‌ی $\omega\in\{0,1\}$ را پرتاب می‌کند. اگر $\omega=1$، چالشگر $p_s=(a,as+2e)$ را محاسبه می‌کند. در غیر این صورت، دو عنصر تصادفی r و r' را تولید کرده و $p_s=(r,r')$ را قرار می‌دهد. سپس، چندگانه‌ی (n,q,R_q,χ_q,p_s) به $\mathcal{B}$ داده می‌شود. هدف مهاجم $\mathcal{B}$ تشخیص حالت p_s است. در این رابطه، او $\mathcal{A}$ را به صورت یک زیرروال اجرا می‌کند:

(۱) **فاز ۱:** $\mathcal{B}$ پارامترهای عمومی $pk_d=(p_s,R_q)$ را به $\mathcal{A}$ می‌دهد.

(۲) **چالش:** $\mathcal{A}$ پیام‌های m_0 و m_1 را از R_q انتخاب کرده و به $\mathcal{B}$ برمی‌گرداند.

(۳) **فاز ۲:** $\mathcal{B}$ سکه‌ی $b\in\{0,1\}$ را پرتاب می‌کند و $c=(c_0,c_1)$ را با استفاده از کلید عمومی دریافتی از چالشگر محاسبه می‌کند، که درآن $c_0=-pk_{d_0}$ و $c_1=pk_{d_1}+m_b$ است و pk_{d_0} و pk_{d_1} به ترتیب اولین و دومین مؤلفه‌ی p_s هستند. سپس، c را به $\mathcal{A}$ می‌دهد.

(۴) **حدس:** $\mathcal{A}$ متن رمز c را دریافت کرده و حدس b' را به $\mathcal{B}$ برمی‌گرداند. سپس، $\mathcal{B}$ بررسی می‌کند که آیا $b=b'$ یا خیر. اگر بله، $\omega'=1$ را خروجی می‌دهد. در غیر این صورت، $\omega'=0$ را در خروجی چاپ می‌کند.

در این صورت، احتمال اینکه $\mathcal{B}$ مقدار $\omega'=1$ را چاپ کند زمانی که $\omega=1$ است برابر با احتمال موفقیت $\mathcal{A}$ است. به عبارت دیگر:

$$\Pr\left(\omega'=1|\omega=1\right)=\Pr\left(\mathrm{SHDF}^{\mathrm{DO}}_{\Pi,\mathcal{A}}(\kappa)=1\right). \quad (۴)$$

به علاوه، احتمال اینکه $\mathcal{B}$ مقدار $\omega'=1$ را چاپ کند زمانی که $\omega=0$ است برابر با ۱/۲ می‌باشد، زیرا از یک رشته تصادفی برای رمزگذاری m_b استفاده می‌کند. بنابراین، داریم:

$$\Pr\left(\omega'=1|\omega=0\right)=1/2. \quad (۵)$$

ازطرف‌دیگر، فرضیه سختی مسئله DRLWE نتیجه می‌دهد:

$$\Pr\left(\omega'=1|\omega=1\right)-\Pr\left(\omega'=1|\omega=0\right)\leq negl(\kappa), \quad (۶)$$

که در آن $negl$ یک تابع ناچیز است. در نهایت، با درنظرگرفتن رابطه‌های (۴)، (۵) و (۶)، می‌بینیم که:

$$\Pr\left(\mathrm{SHDF}^{\mathrm{DO}}_{\Pi,\mathcal{A}}(\kappa)=1\right)\leq 1/2+negl(\kappa). \quad (۷)$$

بنابراین، قضیه اثبات می‌شود. □

قضیه ۲. طرح رمز نگاشتی *SHDF* امنیت تابع را براساس تعریف ۷ فراهم می‌کند.

اثبات. فرض کنید Π طرح رمزنگاری همریخت SHDF باشد و $\mathcal{A}$ یک مهاجم چندجمله‌ای در آزمایش $\mathrm{SHDF}^{\mathrm{FO}}_{\Pi,\mathcal{A}}(\kappa)=1$ باشد که در زیربخش ۱.۷ تعریف شده است. ثابت می‌کنیم که یک تابع ناچیز $negl$ وجود دارد به طوری که:

$$\Pr\left(\mathrm{SHDF}^{\mathrm{FO}}_{\Pi,\mathcal{A}}(\kappa)=1\right)\leq 1/2+negl(\kappa)$$

که در آن κ یک پارامتر امنیتی است.

امنیت Π توسط مسئله DSPR اثبات می‌شود. طبق DSPR، تمایز بین $p=h_\beta\times t$ و $p=r$ که در آن h، β، h_β و t مانند بخش ۳.۵ هستند و r به صورت یکنواخت و تصادفی از R_β انتخاب می‌شود، سخت است.

فرض کنید $\mathcal{B}$ یک مهاجم مسئله DSPR باشد. یک چالشگر در تعامل با $\mathcal{B}$ را در نظر بگیرید که ابتدا $\mathcal{G}(1^\kappa,f)\rightarrow(\alpha,\beta,R_\beta,\chi_\beta)$ را اجرا کرده و سپس سکه‌ی $\omega\in\{0,1\}$ را پرتاب می‌کند. اگر $\omega=1$، چالشگر $p_s=h_\beta\times t$ را محاسبه می‌کند. در غیر این صورت، یک چندجمله‌ای تصادفی r تولید کرده و $p_s=r$ را قرار می‌دهد. چندتایی $(\alpha,\beta,R_\beta,\chi_\beta,p_s)$ به $\mathcal{B}$ داده می‌شود.

هدف مهاجم $\mathcal{B}$ تشخیص حالت p_s است. به این منظور، او $\mathcal{A}$ را به صورت یک زیرروال اجرا می‌کند.

(۱) **فاز ۱:** $\mathcal{B}$ پارامترهای عمومی $pk_f=(p_s,R_\beta)$ را به $\mathcal{A}$ ارائه می‌دهد.

(۲) **چالش:** $\mathcal{A}$ چندجمله‌ای‌های f_0 و f_1 را از R_β انتخاب کرده و به $\mathcal{B}$ برمی‌گرداند.

(۳) **فاز ۲:** $\mathcal{B}$ سکه‌ی $b\in\{0,1\}$ را پرتاب می‌کند و $g=[\alpha\varphi\times p_s+\alpha e+f_b]_\beta$ را محاسبه می‌کند، که در آن α، φ، e و p مانند بخش ۳.۵ هستند. سپس g را به $\mathcal{A}$ می‌دهد.

(۴) **حدس:** $\mathcal{A}$ تابع رمزشده‌ی g را دریافت کرده و حدس b' را به $\mathcal{B}$ برمی‌گرداند. سپس $\mathcal{B}$ بررسی می‌کند که آیا $b=b'$ یا خیر. اگر بله، $\omega'=1$ را خروجی می‌دهد. در غیر این صورت، $\omega'=0$ را چاپ می‌کند.

در این صورت، احتمال اینکه $\mathcal{B}$ مقدار $\omega'=1$ را چاپ کند زمانی که $\omega=1$ است برابر با احتمال موفقیت $\mathcal{A}$ است. به عبارت دیگر:

$$\Pr\left(\omega'=1|\omega=1\right)=\Pr\left(\mathrm{SHDF}^{\mathrm{FO}}_{\Pi,\mathcal{A}}(\kappa)=1\right). \quad (۸)$$

به علاوه، احتمال اینکه $\mathcal{B}$ مقدار $\omega'=1$ را چاپ کند زمانی که $\omega=0$ است برابر با ۱/۲ می‌باشد. زیرا، از یک رشته تصادفی برای رمزگذاری f_b استفاده می‌کند. بنابراین، داریم:

$$\Pr\left(\omega'=1|\omega=0\right)=1/2. \quad (۹)$$

از طرف دیگر، فرضیه سختی مسئله DSPR نتیجه می‌دهد:

$$\Pr\left(\omega'=1|\omega=1\right)-\Pr\left(\omega'=1|\omega=0\right)\leq negl(\kappa), \quad (۱۰)$$

جدول ۳. نمادهای استفاده شده در تجزیه و تحلیل مجانبی

نماد	توضیحات
T_{mult}	زمان اجرای عملیات ضرب
T_{add}	زمان اجرای عملیات جمع
T_{mod}	زمان اجرای عملیات محاسبه پیمانه
q	عدد صحیح مثبت
$T_{s\leftarrow R_q}$	انتخاب تصادفی از R_q
T_{rand}	انتخاب تصادفی از توزیع گاوسی نرمال
ℓ	اندازه پیام
ℓ_{sk}	اندازه کلید مخفی
k	جز صحیح لگاریتم q در مبنای ۲
max_f	میزان افزایش طول پیام پس از پردازش
$\deg_{i_*}$	کمترین درجه‌ی تابع رمزشده
$\deg_{i_m}$	درجه‌ی تابع رمز شده

که در آن $negl$ یک تابع ناچیز است. در نهایت، با درنظر گرفتن رابطه‌های (۸)، (۹) و (۱۰) می‌بینیم:

$$\Pr\left(\mathrm{SHDF}^{\mathrm{FO}}_{\Pi,\mathcal{A}}(\kappa)=1\right)\le 1/2+negl(\kappa). \qquad (11)$$

در نتیجه، رابطه فوق اثبات قضیه را کامل می‌کند. □

۸ تجزیه و تحلیل کارایی

در این بخش، طرح SHDF و روش‌های [۵۰، ۵۶–۶۰] پیاده‌سازی شده‌اند تا زمان اجرا و سربار ارتباطی آن‌ها با یکدیگر مقایسه گردد. تحلیل مقایسه‌ای در دو نوع نتایج مجانبی و واقعی ارائه شده است. نمادهای استفاده شده در این بخش در جدول ۳ آمده است.

جدول ۴ تحلیل مجانبی را نشان می‌دهد. برای ارائه‌ی یک مقایسه‌ی منطقی، طرح SHDF به‌همراه طرح‌های ارائه شده در [۵۰، ۵۶–۶۰] پیاده‌سازی شده‌اند. این طرح‌ها به این دلیل انتخاب شده‌اند که روش ارائه شده در [۵۰] یک طرح رمزنگاری برمبنای سیستم رمزنگاری NTRU است، در حالی که [۵۶–۶۰] طرح‌های رمزنگاری مبتنی بر مسئله LWE هستند. به علاوه، سربار ارتباطی هر کدام از سیستم‌ها در جدول ۴ ارائه شده است. همان‌طور که در جدول ۴ نشان داده شده است ($\mathcal{F}_1-\mathcal{F}_3$)، با توجه به اینکه عمل ضرب نسبت به سایر عملگرها سنگین‌تر و در نتیجه زمانبر می‌باشد، زمان اجرای الگوریتم تولید کلید صاحبان داده در SHDF کوچکتر از طرح‌های [۵۰، ۵۷] و تقریباً برابر با طرح‌های [۵۶، ۵۸–۶۰] است. مقایسه مذکور در بخش (الف) شکل ۳ مشاهده می‌شود. به علاوه، زمان اجرای الگوریتم رمزنگاری صاحبان داده کمتر از روش توصیف شده در طرح [۵۷] است و بسیار نزدیک به روش‌های توصیف شده در طرح‌های [۵۰، ۵۶، ۵۸–۶۰] است. بخش (ب) شکل ۳ مقایسه مذکور را نشان می‌دهد. در نهایت، الگوریتم رمزگشایی صاحبان داده زمان کمتری برای اجرا نسبت به روش‌های ارائه شده در [۵۶، ۵۷] می‌برد و تقریباً دارای زمان اجرای نزدیک با طرح‌های توصیف شده در [۵۰، ۵۸–۶۰] دارد. مقایسه مورد اشاره در بخش (ج) شکل ۳ نشان داده شده است. جدول ۴ اطلاعاتی درباره سربار ارتباطی بین سرور و کاربران را نیز ارائه می‌دهد ($\mathcal{F}_4$ و $\mathcal{F}_5$). بررسی این جدول نشان می‌دهد که سربار ارتباطی از کاربران به سرور و بالعکس در SHDF بیشتر از طرح‌های ارائه شده در [۵۰، ۵۶] و کمتر از طرح [۵۷] و برابر با طرح‌های [۵۸–۶۰] است. بخش‌های (الف) و (ج) شکل ۵ تحلیل سربار ارتباطی فوق را به درستی بین سرور ابری و کاربران نشان می‌دهند.

برای محاسبه سربار ارتباطی از سرور ابری به کاربران (صاحبان داده یا صاحبان تابع)، فرض کنید چندجمله‌ای رمزگذاری شده شامل j جمله‌ی متمایز با درجه $\deg_{i_j}$ به‌ازای $j\in\{0,1,\ldots,m\}$ باشد که در آن $\deg_{i_*}$ و $\deg_{i_m}$ به ترتیب حداقل و حداکثر درجه تابع رمزگذاری شده را نشان می‌دهند. برای محاسبه اندازه نتیجه‌ی بازگردانده‌شده از سرور، ابتدا تابع رمزگذاری شده را از درجه بیشینه به درجه کمینه به ترتیب نزولی مرتب می‌کنیم. سپس $\max_f=\max_m=\max\{\max_{m-1},\deg_{i_m}\}$ را قرار می‌دهیم، جایی که $max_j=\max_j\{\max_{j-1},\deg_{i_j}\}+1$ و $\max_0=\deg_{i_*}$ برای $j\in\{1,\ldots,m\}$ است.

پیاده‌سازی مذکور به‌کمک Ubuntu 20.04، پردازنده Intel Core i5-5200U 2 x 2.2 - 2.7 گیگاهرتز و ۸ گیگابایت رم و با استفاده از کتابخانه پایتون Numpy [1] انجام گرفته است. به علاوه، در این بخش فرض شده است که طول پیام، تابع و کلید در بازه 2^{12} تا 2^{17} قرار دارند. همچنین، در فرایند پیاده‌سازی $q=\beta=2^{20}$ در نظر گرفته شده است. شکل ۳ زمان تولید کلید، رمزگذاری و رمزگشایی الگوریتم‌های متناظر با صاحبان داده را با روش‌های موجود در مراجع [۵۰، ۵۶–۶۰] مقایسه می‌کند. مشاهده می‌شود که زمان اجرای الگوریتم‌های استفاده شده توسط صاحبان داده کمتر یا مساوی روش‌های مذکور است. شکل ۴ زمان تولید کلید، رمزگذاری و رمزگشایی الگوریتم‌های استفاده شده توسط صاحبان تابع در SHDF را نشان می‌دهد. همانطور که در شکل ۴ نشان داده شده است، زمان رمزگذاری و رمزگشایی توابع تقریباً مشابه رمزگذاری و رمزگشایی داده‌ها با همان اندازه است.

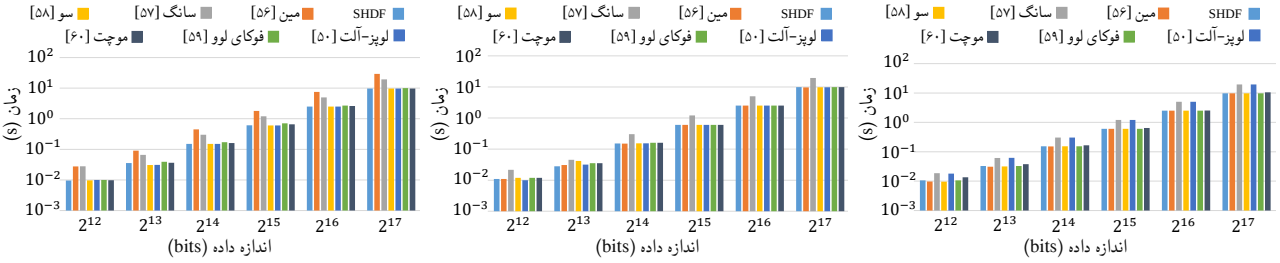
شکل ۵ سربار ارتباطی بین کاربران و سرور ابری را نمایش می‌دهد. بخش (الف) سربار ارتباطی از صاحبان داده به سرور ابری در SHDF را با روش‌های [۵۰، ۵۶–۶۰] مقایسه می‌کند. در این بخش $\ell_{sk}=1/2\times10^5$ همانند [۵۷] در نظر گرفته شده است. در بخش (ب) میزان سربار ارتباطی از صاحبان تابع به سرور ابری را نشان می‌دهد و در نهایت در بخش (ج) اندازه نتیجه‌ی به دست آمده از سرور محاسبه شده است. در بخش (ج) درجه‌ی تابع رمزگذاری شده ($\deg_{i_m}$) در هر حالت ۵۰۰، ۱۰۰۰، ۱۵۰۰، ۲۰۰۰، ۲۵۰۰ و ۳۰۰۰ می‌باشد. مقایسه‌ی ارائه شده در نمودار بخش (ج) نشان می‌دهد که اگرچه تابع در SHDF رمزگذاری شده و به سرور ارسال می‌شود، اما سربار ارتباطی از سرور به کاربران با سایر

[1] https://numpy.org/doc/stable/reference/

جدول ۴. مقایسه زمان اجرا و سربار ارتباطی SHDF با طرح‌های مختلف

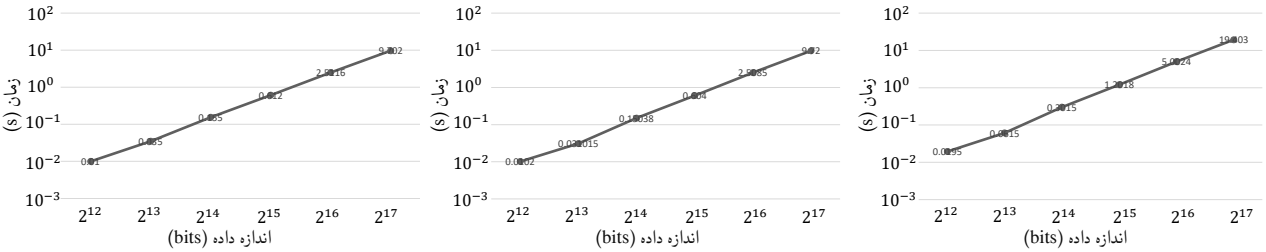
زمان / سربار	SHDF		مین [۵۶]	سانگ [۵۷]	سو [۵۸]	لوپز-آلت [۵۰]	فوکای لوو [۵۹]	موچت [۶۰]
	صاحبان داده	صاحبان تابع						
$\mathcal{F}_1$	$T_{s\leftarrow R_q}+3T_{rand}+T_{mult}$	$2T_{mult}+3T_{mod}$	$T_{rand}+T_{mult}+T_{mod}$	$2T_{mult}+3T_{rand}$	$3T_{rand}+T_{mult}$	$2T_{mult}+2T_{mod}+2T_{rand}$	$T_{s\leftarrow R_q}+2T_{rand}+T_{mult}$	$T_{s\leftarrow R_q}+2T_{rand}+T_{mult}+2T_{add}$
$\mathcal{F}_2$	$2T_{add}+2T_{rand}+T_{mult}$	$T_{mult}+3T_{rand}+2T_{add}+2T_{mod}$	$2T_{add}+T_{mod}+T_{mult}$	$4T_{add}+3T_{rand}+2T_{mult}$	$3T_{add}+3T_{rand}+T_{mult}$	$2T_{add}+2T_{rand}+T_{mult}+T_{mod}$	$3T_{add}+3T_{rand}+T_{mult}$	$3T_{add}+3T_{rand}+T_{mult}$
$\mathcal{F}_3$	$T_{add}+T_{mult}+T_{mod}$	$T_{mult}+2T_{mod}$	$3T_{mult}+T_{mod}$	$2T_{mult}+2T_{mod}$	$2T_{add}+T_{mult}+T_{mod}$	$T_{mult}+2T_{mod}$	$2T_{add}+T_{mult}+2T_{mod}+T_{rand}$	$3T_{add}+T_{mult}+T_{rand}$
$\mathcal{F}_4$	$2k\ell$	$k\ell$	$k\ell$	$k(\ell+\ell_{sk})$	$2k\ell$	$k\ell$	$2k\ell$	$2k\ell$
$\mathcal{F}_5$	$2max_f k\ell$	$max_f k\ell$	$max_f k\ell$	$max_f k(\ell+\ell_{sk})$	$2max_f k\ell$	$max_f k\ell$	$2max_f k\ell$	$2max_f k\ell$

توضیحات: $\mathcal{F}_1$: زمان اجرای الگوریتم تولید کلید، $\mathcal{F}_2$: زمان اجرای الگوریتم رمزنگاری، $\mathcal{F}_3$: زمان اجرای الگوریتم رمزگشایی، $\mathcal{F}_4$: سربار ارتباطی از کاربران به سرور، $\mathcal{F}_5$: سربار محابراتی از سرور به کاربران.



(ج) (ب) (الف)

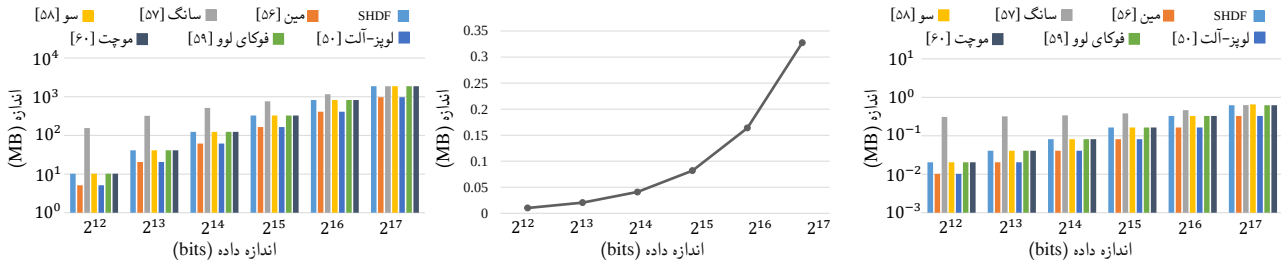
شکل ۳. (الف) زمان اجرای الگوریتم تولید کلید صاحبان داده، (ب) زمان اجرای الگوریتم رمزگذاری صاحبان داده، (ج) زمان اجرای الگوریتم رمزگشایی صاحبان داده.



(ج) (ب) (الف)

شکل ۴. (الف) زمان اجرای الگوریتم تولید کلید صاحبان تابع، (ب) زمان اجرای الگوریتم رمزگذاری صاحبان تابع، (ج) زمان اجرای الگوریتم رمزگشایی صاحبان تابع.

سیستم‌های ارائه شده در [۵۰، ۵۶-۶۰] تفاوت چندانی ندارد.



شکل ۵. (الف) سربار ارتباطی انتقال داده از صاحبان داده به سرور ابری، (ب) سربار ارتباطی انتقال داده از صاحبان تابع به سرور ابری، (ج) سربار ارتباطی انتقال داده از سرور ابری به صاحبان داده.

۹ نتیجه‌گیری و کارهای آتی

در این مقاله، ما یک سیستم رمزنگاری همریخت برای خدمات نرم‌افزار ابری طراحی کردیم که می‌تواند داده‌ها و الگوریتم‌هایی که قرار است داده‌ها روی آنها پردازش شوند را بطور همزمان رمز کند. طرح SHDF ما توانست موارد زیر را فراهم کند:

(۱) SHDF می‌تواند بطور همزمان محرمانگی داده و الگوریتم‌ها را برخلاف سیستم‌های رمزنگاری همریخت موجود فراهم کند.
(۲) تمامی محاسبات سنگین می‌تواند بطور امن به یک سرور ابری برون‌سپاری شوند.
(۳) SHDF می‌تواند بطور همریخت داده‌ها و توابع را با سیستم‌های رمزگذاری همریخت متناظرشان رمز کرده و آنها را برای برون‌سپاری به محیط‌های ناامن برای پردازش یا ذخیره‌سازی آماده کند.
(۴) در این مقاله، ما نشان دادیم که سیستم پیشنهادی SHDF ما برای پیاده‌سازی در محیط‌های ابری سازگار است.
(۵) درآخر، با ارائه مدل مهاجم و تعاریف امنیتی نشان دادیم که سیستم رمزگذاری همریخت ما دارای امنیت اثبات‌پذیر برای محرمانگی داده و تابع با فرض سخت بودن مسائل DRLWE و DSPR می‌باشد. همچنین تجزیه و تحلیل کارایی ما نشان داد که سیستم ما در مقایسه با سایر سیستم‌های قابل‌استفاده موجود، کارایی مناسبی را نیز دارد.

با توجه به اینکه فرآیند رمزگشایی طرح ارائه شده دو مرحله‌ای بوده و همین امر باعث می‌شود تا صاحبان تابع در فرآیند رمزگشایی شرکت داشته باشند، درآینده سعی می‌شود برای بهبود عملکرد طرح پیشنهادی و همچنین کاهش سربار مخابراتی یک طرح غیرتعاملی نیز ارائه گردد. در طرح ارائه شده، سعی می‌شود تا صاحبان تابع در فرآیند رمزگشایی شرکت نکرده و تنها صاحبان داده بتوانند نتیجه پردازش شده از سرور را رمزگشایی کنند بدون اینکه نشت اطلاعاتی از ساختار تابع صورت پذیرد.

مراجع

[1] Christos Stergiou, Kostas E Psannis, Byung-Gyu Kim, and Brij Gupta. Secure integration of iot and cloud computing. *Future Generation Computer Systems*, 78:964–975, 2018.

[2] Aaqib Rashid and Amit Chaturvedi. Cloud computing characteristics and services: a brief review. *International Journal of Computer Sciences and Engineering*, 7(2):421–426, 2019.

[3] Yang Yang, Yanjiao Chen, Fei Chen, and Jing Chen. An efficient identity-based provable data possession protocol with compressed cloud storage. *IEEE Transactions on Information Forensics and Security*, 17:1359–1371, 2022.

[4] Morey J Haber, Brian Chappell, and Christopher Hills. Cloud computing. In *Cloud Attack Vectors*, pages 9–25. Springer, 2022.

[5] Craig Gentry et al. *A fully homomorphic encryption scheme*, volume 20. Stanford university Stanford, 2009.

[6] Boaz Barak, Oded Goldreich, Rusell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (im) possibility of obfuscating programs. In *Advances in Cryptology—CRYPTO 2001: 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19–23, 2001 Proceedings*, pages 1–18. Springer, 2001.

[7] Mohammad Ali, Javad Mohajeri, Mohammad-Reza Sadeghi, and Ximeng Liu. A fully distributed hierarchical attribute-based encryption scheme. *Theoretical Computer Science*, 815:25–46, 2020.

[8] Mohammad Ali, Mohammad-Reza Sadeghi, and Ximeng Liu. Lightweight revocable hierarchical attribute-based encryption for internet of things. *IEEE Access*, 8:23951–23964, 2020.

[9] Mohammad Ali, Mohammad-Reza Sadeghi, Ximeng Liu,

posite degree residuosity classes. In *International conference on the theory and applications of cryptographic techniques*, pages 223–238. Springer, 1999.

[20] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-dnf formulas on ciphertexts. In *Theory of cryptography conference*, pages 325–341. Springer, 2005.

[21] Yuval Ishai and Anat Paskin. Evaluating branching programs on encrypted data. In *Theory of Cryptography Conference*, pages 575–594. Springer, 2007.

[22] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 169–178, 2009.

[23] Craig Gentry and Shai Halevi. Fully homomorphic encryption without squashing using depth-3 arithmetic circuits. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 107–109. IEEE, 2011.

[24] Craig Gentry and Shai Halevi. Implementing gentry's fully-homomorphic encryption scheme. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 129–148. Springer, 2011.

[25] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT)*, 6(3):1–36, 2014.

[26] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. *Cryptology ePrint Archive*, 2012.

[27] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In *Annual Cryptology Conference*, pages 868–886. Springer, 2012.

[28] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 409–437. Springer, 2017.

[29] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Annual Cryptology Conference*, pages 75–92. Springer, 2013.

[30] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachene. Faster fully homomorphic encryp-

Yinbin Miao, and Athanasios V Vasilakos. Verifiable online/offline multi-keyword search for cloud-assisted industrial internet of things. *Journal of Information Security and Applications*, 65:103101, 2022.

[10] KVN Islam. Review on benefits and security challenges of cloud computing. *International Journal of Computer Science and Information Technologies*, 8(2):224–228, 2017.

[11] Zhiyong Zhang, Cheng Li, Brij B Gupta, and Danmei Niu. Efficient compressed ciphertext length scheme using multi-authority cp-abe for hierarchical attributes. *IEEE Access*, 6:38273–38284, 2018.

[12] Qiming Zheng, Xiaomin Wang, Muhammad Khurram Khan, Wenfang Zhang, Brij B Gupta, and Wei Guo. A lightweight authenticated encryption scheme based on chaotic scml for railway cloud service. *IEEE Access*, 6:711–722, 2017.

[13] BB Gupta. An efficient kp design framework of attribute-based searchable encryption for user level revocation in cloud. *Concurrency and Computation: Practice and Experience*, 32(18):e5291, 2020.

[14] Bineet Joshi, Bansidhar Joshi, Anupama Mishra, Varsha Arya, Avadhesh Kumar Gupta, and Dragan Peraković. A comparative study of privacy-preserving homomorphic encryption techniques in cloud computing. *International Journal of Cloud Applications and Computing (IJCAC)*, 12(1):1–11, 2022.

[15] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *International conference on the theory and applications of cryptographic techniques*, pages 223–238. Springer, 1999.

[16] Ronald L Rivest, Len Adleman, Michael L Dertouzos, et al. On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11):169–180, 1978.

[17] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-dnf formulas on ciphertexts. In *Theory of cryptography conference*, pages 325–341. Springer, 2005.

[18] Shafi Goldwasser and Silvio Micali. Probabilistic encryption & how to play mental poker keeping secret all partial information. In *Providing sound foundations for cryptography: on the work of Shafi Goldwasser and Silvio Micali*, pages 173–201. 2019.

[19] Pascal Paillier. Public-key cryptosystems based on com-

Venkatesan, and Akshay Wadia. Founding cryptography on tamper-proof hardware tokens. In *Theory of Cryptography: 7th Theory of Cryptography Conference, TCC 2010, Zurich, Switzerland, February 9-11, 2010. Proceedings 7*, pages 308–326. Springer, 2010.

[41] Andrew Chi-Chih Yao. How to generate and exchange secrets. In *27th annual symposium on foundations of computer science (Sfcs 1986)*, pages 162–167. IEEE, 1986.

[42] Shafi Goldwasser, Yael Kalai, Raluca Ada Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 555–564, 2013.

[43] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *Theory of Cryptography: 8th Theory of Cryptography Conference, TCC 2011, Providence, RI, USA, March 28-30, 2011. Proceedings 8*, pages 253–273. Springer, 2011.

[44] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 197–206, 2008.

[45] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *2007 IEEE symposium on security and privacy (SP'07)*, pages 321–334. IEEE, 2007.

[46] Sanjam Garg, Craig Gentry, Shai Halevi, Amit Sahai, and Brent Waters. Attribute-based encryption for circuits from multilinear maps. In *Advances in Cryptology–CRYPTO 2013: 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, pages 479–499. Springer, 2013.

[47] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Predicate encryption for circuits from lwe. In *Advances in Cryptology–CRYPTO 2015: 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, pages 503–523. Springer, 2015.

[48] Shweta Agrawal. Stronger security for reusable garbled circuits, general definitions and attacks. In *Advances in Cryptology–CRYPTO 2017: 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, Au-*

tion: Bootstrapping in less than 0.1 seconds. In *international conference on the theory and application of cryptology and information security*, pages 3–33. Springer, 2016.

[31] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Faster packed homomorphic operations and efficient circuit bootstrapping for tfhe. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 377–408. Springer, 2017.

[32] Martin R Albrecht, Carlos Cid, Jean-Charles Faugere, Robert Fitzpatrick, and Ludovic Perret. On the complexity of the bkw algorithm on lwe. *Designs, Codes and Cryptography*, 74:325–354, 2015.

[33] Jung Hee Cheon and Jinsu Kim. A hybrid scheme of public-key encryption and somewhat homomorphic encryption. *IEEE transactions on information forensics and security*, 10(5):1052–1063, 2015.

[34] Rongxing Lu, Hui Zhu, Ximeng Liu, Joseph K Liu, and Jun Shao. Toward efficient and privacy-preserving computing in big data era. *IEEE Network*, 28(4):46–50, 2014.

[35] Rongxing Lu. A new communication-efficient privacy-preserving range query scheme in fog-enhanced iot. *IEEE Internet of Things Journal*, 6(2):2497–2505, 2018.

[36] DING Wenxiu, Zheng Yan, and Robert H Deng. Privacy-preserving data processing with flexible access control. *IEEE Transactions on Dependable and Secure Computing*, 17(2):363–376, 2017.

[37] Shafi Goldwasser and Yael Tauman Kalai. On the impossibility of obfuscation with auxiliary input. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*, pages 553–562. IEEE, 2005.

[38] Nir Bitansky, Ran Canetti, Shafi Goldwasser, Shai Halevi, Yael Tauman Kalai, and Guy N Rothblum. Program obfuscation with leaky hardware. In *Advances in Cryptology–ASIACRYPT 2011: 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings 17*, pages 722–739. Springer, 2011.

[39] Shafi Goldwasser, Yael Tauman Kalai, and Guy N Rothblum. One-time programs. In *Crypto*, volume 5157, pages 39–56. Springer, 2008.

[40] Vipul Goyal, Yuval Ishai, Amit Sahai, Ramarathnam

Han. Multi-key fully homomorphic encryption without crs from rlwe. *Computer Standards & Interfaces*, 86:103742, 2023.

[60] Christian Mouchet, Elliott Bertrand, and Jean-Pierre Hubaux. An efficient threshold access-structure for rlwe-based multiparty homomorphic encryption. *Journal of Cryptology*, 36(2):10, 2023.

[61] Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography*. CRC press, 2020.

gust 20–24, 2017, Proceedings, Part I 37, pages 3–35. Springer, 2017.

[49] Hao Chen, Wei Dai, Miran Kim, and Yongsoo Song. Efficient multi-key homomorphic encryption with packed ciphertexts with application to oblivious neural network inference. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 395–412, 2019.

[50] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. Multikey fully homomorphic encryption and applications. *SIAM Journal on Computing*, 46(6):1827–1892, 2017.

[51] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 1–23. Springer, 2010.

[52] Yarkın Doröz, Yin Hu, and Berk Sunar. Homomorphic aes evaluation using the modified ltv scheme. *Designs, Codes and Cryptography*, 80(2):333–358, 2016.

[53] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-lwe and security for key dependent messages. In *Annual cryptology conference*, pages 505–524. Springer, 2011.

[54] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):1–40, 2009.

[55] Shafi Goldwasser, Yael Tauman Kalai, Chris Peikert, and Vinod Vaikuntanathan. Robustness of the learning with errors assumption. *Innovations in Computer Science*, pages 230–240, 2010.

[56] Zhaoe Min, Geng Yang, Jin Wang, and Gwang-jun Kim. A privacy-preserving bgn-type parallel homomorphic encryption algorithm based on lwe. *Journal of Internet Technology*, 20(7):2189–2200, 2019.

[57] Xinxia Song, Zhigang Chen, and Liang Chen. A multi-bit fully homomorphic encryption with shorter public key from lwe. *IEEE Access*, 7:50588–50594, 2019.

[58] Yang Su, Bailong Yang, Chen Yang, and Luogeng Tian. Fpga-based hardware accelerator for leveled ring-lwe fully homomorphic encryption. *IEEE Access*, 8:168008–168025, 2020.

[59] Fucai Luo, Haiyan Wang, Al-Kuwari Saif, and Weihong

MonadiAfta-Bianual Scientific Journal for Cyberspace Security
2024, Volume 21, Number 1 (pp. 30–48)
https://monadi.isc.org.ir

Privacy-preserving data and function in cloud environments using homomorphic encryption

Amin Hosseingholizadeh, Farhad Rahmati and Mohammad Ali *

[1] *Department of Mathematics and Computer Science, Amirkabir University of Technology, Tehran, Iran.*

ARTICLE INFO.

Article history:
Received: November 19, 2023
Accepted: February 7, 2024
Published Online: March 12, 2024

Keywords:
Homomorphic encryption cryptosystems
Cloud computing
Data confidentiality
Data and software security.

Type: Research paper

ABSTRACT

With the emergence of new phenomena in the telecommunications and information technology fields, such as cloud computing and smart networks, we are witnessing new challenges in these areas. One of the most significant challenges is the privacy of outsourced data. Due to the limited processing power of new intelligent devices such as tablets and mobile phones, outsourcing computations to these platforms has gained more attention from users. In addition to data privacy, the security of algorithms used in online software is also of great importance. Therefore, software providers may be concerned about the disclosure of their algorithms after outsourcing them to cloud environments. Existing homomorphic encryption systems can provide privacy for data that needs to be processed online. However, the concurrent privacy of algorithms in these systems has not been addressed. To address this, we introduce a simultaneous homomorphic encryption of data and function called SHDF. This system can homomorphically encrypt all algorithms used in the software and the data to be processed on them, enabling necessary computations to be performed on an insecure server. Furthermore, we show that the proposed system is provably secure. Our implementation results indicate that it is usable in cloud environments with the desired efficiency.

* Corresponding author
Email addresses: amin_hgholizadeh@aut.ac.ir (Amin Hosseingholizadeh), frahmati@aut.ac.ir (Farhad Rahmati), mali71@aut.ac.ir (Mohammad Ali)