مقاله پژوهشی

ارائه رویکردی جدید مبتنی بر نگاشت آشوب برای پیاده‌سازی فازهای امنیتی بر روی شبکه ابر

آزیتا رضائی، علی برومندنیا و سید جواد میرعابدینی

چکیده: شرکت‌ها و سازمان‌های بزرگ در دنیای امروز، تمایل بسیاری به استفاده از سرویس‌های ETI جهت ذخیره‌سازی و انتقال داده‌های خود دارند. افزایش نرخ استفاده از سرویس‌های تحت شبکه، افزایش مخاطرات و حملات به داده‌ها در زمان ارسال و دریافت و ایجاد محیطی امن برای انجام خدمات در بستر شبکه ابر، چالشی بزرگ را در حوزه سیاست امنیتی مطرح می‌کند. بخش مدیریت امنیت بروکر در شبکه ابر، تأثیر زیادی در مدیریت و متعادل‌سازی متغیرهای امنیت، کارایی و دسترس‌پذیری دارد. این مقاله سعی دارد با ارائه چهار فاز امنیتی بر روی داده در زمان ارسال و دریافت بین مشتری و CSP، ضمن پیشگیری از خرابی داده در زمان سرقت، میزان بازدهی متغیرهای کارایی، متغیر دسترس‌پذیری و شاخص رضایتمندی مشتریان را بررسی نماید. نتایج ارزیابی بر روی سه مدل آزمایشی نشان می‌دهند میزان جرائم تخصیصی سیستم به میزان ۶۱/۴۱٪ کاهش و شاخص رضایت مشتریان تا ۶۰/۶۷٪ افزایش می‌یابد.

کلیدواژه: **الگوریتم امنیتی چندلایه (MLS)، رمزنگاری، قطعه‌بندی داده‌ها، مدیریت کارگزار امنیتی.**

۱- مقدمه

با افزایش دامنه خدمت سرویس‌دهنده‌های با زیرساخت ابر، میزان شرکت‌های علاقه‌مند به استفاده از آن نیز رو به افزایش است. انتقال داده، سرویس ETI[1] و ارائه خدمت به‌عنوان یک چالش دارای اهمیت در حوزه امنیت مطرح است. از طرفی، الگوریتم‌ها و روش‌های سارقین جهت سرقت و صدمه به داده هر روز تغییر می‌کنند که برای جلوگیری از این مخاطره، باید الگوریتم‌های ابتکاری رمزگذاری داده‌ها نیز متنوع گردند. با این حال، آسیب‌پذیری در برابر شنود[2] [۱]، دستکاری[3]، انکار[4]، افشای اطلاعات[5]، انکار سرویس یا تزریق سرویس غیرمجاز[6]، مجازی‌سازی و حمله از طریق ابر جعلی[7] [۲] در چهار گونه معماری ابری [۳] رو به افزایش است. اگرچه حملاتی نظیر هک‌کردن و هرزنامه باعث افزایش آسیب‌پذیری داده‌ها می‌شود، استفاده از روش‌ها و الگوریتم‌های پیشگیرانه می‌تواند از نشت اطلاعات جلوگیری کند.

CSP[8] به‌عنوان مسئول حفاظت از داده و برنامه‌ها [۴] می‌تواند از کارگزار[9] به‌عنوان حافظ امنیتی مناسب در این خصوص استفاده نماید. برای این منظور از رمزگذاری داده به‌عنوان امری تأثیرگذار بر روی ابر استفاده می‌شود. داده‌ها باید بر روی بیت‌های داده‌ای قرار گیرند که این عمل می‌تواند به سه حالت استاندارد[10]، غیراستاندارد[11] یا ترکیبی[12] انجام گیرد. در واقع این عمل با استفاده از فرمول و تقسیم‌بندی داده‌ها به‌صورت بلوکی و استفاده از روش کلید رمز انجام می‌شود [۵].

در دهه اخیر، سیاست‌های امنیتی در حوزه شبکه‌های بزرگ با افزایش نرخ سرقت و نشت اطلاعات در شبکه‌های بی‌سیم بسیار مطرح و افزایش نرخ کارایی و دسترس‌پذیری نیز در این حوزه حائز اهمیت است [۶]. روش‌های متنوعی در این حوزه مانند تقسیم‌بندی شبکه [۷] و [۸] و نگاشت آشوب جهت افزایش امنیت [۹] و جلوگیری از حملات وجود دارد [۱] و [۱۰]. ضمن این که استفاده از روش‌های امنیتی به‌طور مستقیم بر روی کارایی سیستم [۸] تأثیر می‌گذارد، استفاده از این روش‌ها همواره مورد نیاز کاربران است.

استفاده از قابلیت ماتریس و نگاشت آشوب [۹] در الگوریتم‌های رمزگذاری جهت افزایش امنیت داده به‌عنوان روشی نوین مطرح است. در این مقاله به دنبال آن هستیم تا از این ویژگی به‌منظور افزایش امنیت داده در زمان انتقال به‌واسطه کارگزارها با درنظرگرفتن میزان کارایی و تعادل‌سازی بین متغیرهای امنیت، کارایی و دسترس‌پذیری استفاده کنیم [۱۱]. نقاط قوت روش پیشنهادی عبارتند از

۱) استفاده از امنیت چندفازی (MLS)[13] به‌منظور جلوگیری از حملات در زمانی که داده بین مشتری و CSP مبادله می‌شود.
۲) استفاده از سرویس قطعه‌بندی‌شده جهت ارسال داده در قالب چهار فاز امنیتی
۳) بررسی مجموع جریمه تخصیصی در متغیرهای امنیت، کارایی و دسترس‌پذیری به‌ویژه در متغیر امنیت و ارائه روش وزن‌گذاری

این مقاله در تاریخ ۲۱ خرداد ماه ۱۴۰۲ دریافت و در تاریخ ۱۸ آبان ماه ۱۴۰۲ بازنگری شد.

آزیتا رضائی، گروه کامپیوتر، واحد تهران جنوب، دانشگاه آزاد اسلامی، تهران، ایران، (email: st_az_rezaei@azad.ac.ir).

علی برومندنیا (نویسنده مسئول)، گروه کامپیوتر، واحد تهران جنوب، دانشگاه آزاد اسلامی، تهران، ایران، (email: Broumandnia@azad.ac.ir).

سید جواد میرعابدینی، گروه کامپیوتر، واحد تهران مرکز، تانشگاه آزاد اسلامی تهران، ایران، (email: j_mirabedini@iauctb.ac.ir).

1. Thing by Internet
2. Spoofing
3. Tampering
4. Repudiation
5. Information Disclosure
6. Denial of Service or Service Injection
7. Virtualization and Metadata Spoofing Attacks
8. Cloud Service Provider
9. Broker
10. Standard Method
11. Non-Standard Method
12. Hybrid Method
13. Multi-Layered Security

جدول ۱: مقایسه بین روش پیشنهادی مقاله و سایر روش‌ها.

بررسی مقاومت داده در برابر تهدید	رویکرد کارایی	رویکرد امنیت	روش پیشنهادی	منبع
□	×	□	پشتیبانی از امنیت ابر در سکوهای توزیع‌شده	[۱۲]
×	□	×	افزایش کارایی با الگوریتم بلوک‌بندی ویدمن	[۸]
×	×	□	مبهم‌سازی و رمزگذاری RSA	[۱۴]
×	×	□	اصلاح ویژگی امنیت به‌منظور جلوگیری از منابع بلااستفاده در ایستگاه‌های موبایل	[۱۵]
×	×	□	استفاده از روش رمزگذاری برای امنیت داده‌ها و شخصی‌سازی	[۱۶]
□	×	□	استفاده هم‌زمان از مزایای رمزگذاری آشوب و درخت هش مرکل	[۱۷]
□	×	□	ارائه پروتکل کم‌وزن و حفظ درجه امنیت در شبکه MQTT	[۱۸]
□	□	□	لایه‌گذاری امنیتی (MLS) (روش پیشنهادی)	MLS

پارامترها به‌منظور محاسبه جرائم

۴) بررسی تأثیر روش پیشنهادی بر روی بازدهی متغیرهای کارایی، دسترس‌پذیری و شاخص رضایت مشتری

یکی از روش‌های مؤثر در جلوگیری از سوء استفاده داده‌ها، بهره‌گیری از روش‌های پیشگیری از خرابی داده‌هاست. مقاله حاضر می‌خواهد با استفاده از روش فازبندی و به‌صورت مرحله‌بندی‌شده، داده را به‌صورت ایمن ارسال و دریافت نماید. در این حالت از مزایای متدهای به‌روز مانند ویدمن، نگاشت آشوب، تولید کلید خصوصی با بهره‌گیری از خاصیت اعداد اول و بهره‌گیری از قابلیت‌های مدیریت بروکر استفاده می‌گردد. هر یک از موارد ذکرشده موجب قراردهی قابلیت‌هایی برای افزایش مقاومت داده در برابر حمله می‌شود. در واقع هدف از این مقاله، بهره‌گیری از قابلیت‌های روش‌های جدید و پویا به‌منظور تولید داده امن در زمان ارسال و دریافت می‌باشد. تمرکز اصلی این مقاله بر عملیات رمزنگاری و تولید داده رمزگذاری‌شده است. از مزایای روش پیشنهادی، بررسی از جنبه‌های مختلف مثل بررسی تغییرات در بازدهی متغیرهای کارایی و دسترس‌پذیری در مقابل متغیر امنیت و نیز تأثیر این روش بر میزان جرائم تخصیصی به سیستم با استفاده از روش وزن‌گذاری، محاسبه رضایت مشتری و میزان بازدهی آن با سایر کارهای مشابه، تأثیر کار بر زمان انجام عملیات مورد ارزیابی می‌باشد تا بتوان وجاهت کار پیشنهادی را به نمایش گذاشت. همچنین رصد و پشتیبانی از سرویس و حفظ یکپارچگی داده‌ها در زمان انتقال بین ابرها دارای اهمیت است.

سایر بخش‌های این مقاله بدین صورت است: بخش دوم به بررسی رویکردهای گذشته می‌پردازد. در بخش سوم، معماری زیرساخت و سیاست‌های امنیتی بررسی می‌شود. بخش چهارم به طرح کلی روش پیشنهادی و بخش پنجم به ارائه روش پیشنهادی، انگیزه و نوآوری موجود در طرح می‌پردازد. بخش ششم ارزیابی کارایی، آزمایش‌ها و نمودارهای مقایسه‌ای را نشان می‌دهد و بخش آخر نیز نتایج حاصل را بیان می‌کند.

۲- کارهای مرتبط

با وجود این که بسیاری از کاربران در دهه اخیر نیازمند اتصال و انتقال داده از طریق شبکه هستند، بسیاری از تحقیق‌های انجام‌یافته بر حوزه سیاست‌های امنیتی در شبکه تمرکز دارد.

مونوگرام[1] و همکاران [۱۲] از چارچوب کاهش نقشه ابر داده [۱۳] برای پشتیبانی از امنیت داده استفاده کردند. این کار برای سکوهای[2] توزیع‌شده برای حل مشکل داده‌های بزرگ مناسب است؛ با این حال نویسنده به مسئله گم‌شدن یا سرقت داده‌ها نپرداخته است.

یانگ[3] و همکاران [۸] به دنبال پیداکردن راهی برای افزایش‌دادن کارایی به‌وسیله قطعه‌بندی و تمرکز بر روی اثرگذاری بر سیستم به‌واسطه روش بلوک‌بندی ویندمن[4] بودند. در مقابل، مقاله حاضر به دنبال اصلاح سیاست امنیتی با الگوریتم فازبندی به‌گونه‌ای است که بتواند کارایی و دسترس‌پذیری را نیز بهبود دهد.

گاتام[5] و همکاران [۱۴] به‌دنبال روشی جهت مبهم‌سازی[6] منابع و کد RSA[7] در رکوردهای سلامت الکتریکی در شبکه ابری بودند.

راتا[8] [۱۵] خاصیت امنیت را در ایستگاه‌های موبایل برای جلوگیری از منابع غیرلازم استفاده نمود. او از ایستگاه‌های مجازی با پهنای باند قابل انتخاب و سایر ویژگی‌های CSP استفاده کرد.

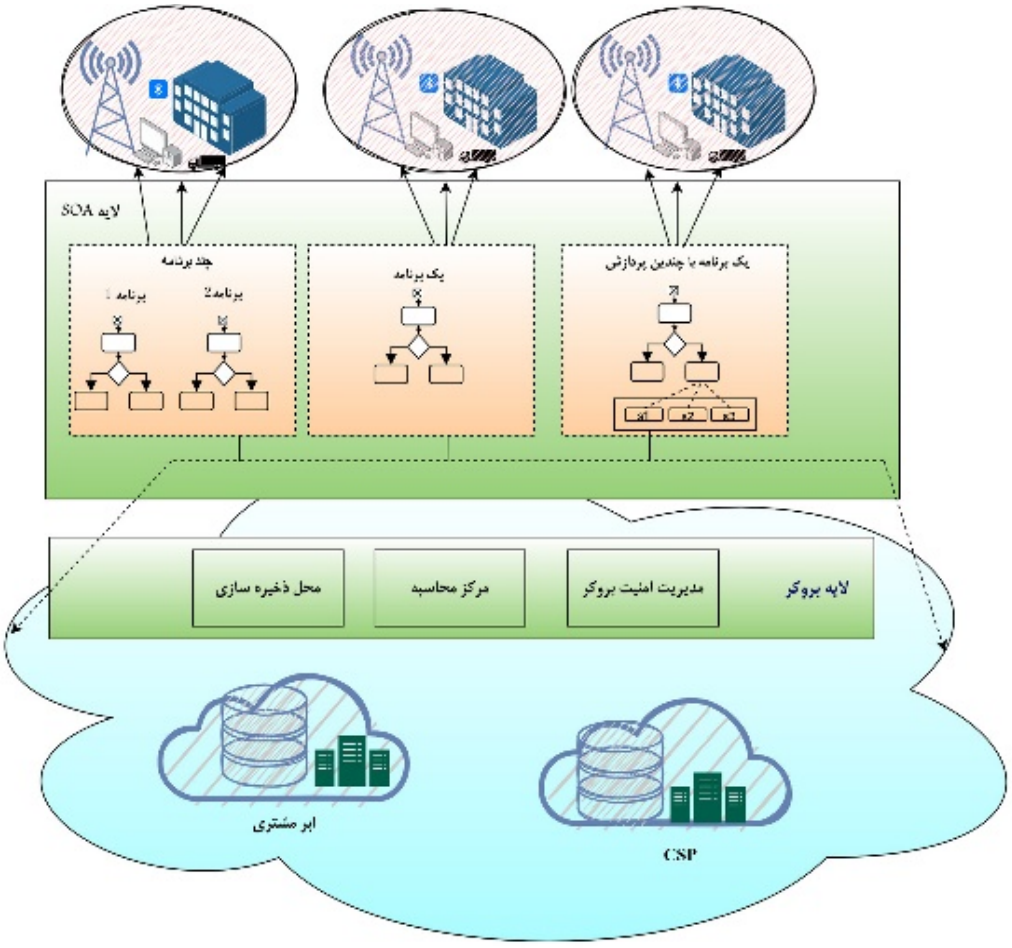
شارما[9] و همکاران [۱۶] از روش‌های رمزگذاری متعدد برای محافظت از امنیت داده و شخصی‌سازی استفاده نمودند. هرچند آنها به نتایج حاصل از حمله به داده‌ها و یا افزایش درخواست‌ها در شبکه نپرداخته‌اند، با این حال، استفاده از روش‌های رمزگذاری مؤثر برای محافظت از داده‌های حساس ضروری است.

نسا[10] و همکاران [۱۷] از رمزگذاری آشوب و درخت هش مرکل[11] برای ارسال و دریافت داده در شبکه استفاده نمودند؛ ولی نویسندگان به اثر این کار در متغیرهای کارایی و دسترس‌پذیری نپرداخته‌اند.

هینتا[12] و همکاران [۱۸] به دنبال ارائه یک پروتکل کم‌وزن و هم‌زمان حفظ درجه امنیت در شبکه MQTT[13] بودند. آنها سعی کردند با بیان آخرین سناریوهای حمله در محیط MQTT و مطالعه روی انواع حملات و کاراکترهای آنها، مکانیسم دفاعی خوبی در برابر حملات ارائه کنند.

با توجه به رویکردهای عنوان‌شده، فرضیه MLS می‌خواهد با ارائه رویکردی جدید، علاوه بر معرفی روش رمزگذاری در چهار فاز با استفاده از ویژگی‌های الگوریتم‌های ویدمن و آشوب و استفاده از خواص اعداد اول، میزان امنیت را در زمان ارسال و دریافت داده به‌واسطه CSP افزایش دهد. جدول ۱ به مقایسه اجمالی بین روش‌های موجود و روش پیشنهادی

1. G. Manogaran
2. Platform
3. Yang
4. Block Wiedemann Method
5. P. Gautam
6. Obfuscation
7. Rivest-Shamir-Adleman Algorithm
8. M. Rathan
9. Y. Sharma
10. Nesa
11. Merkle Hash Tree
12. A. J. Hintaw
13. Message Queuing Telemetry Transport



شکل ۱: معماری کار.

در این مقاله می‌پردازد. علاوه بر این، مقاله حاضر یک پژوهش جامع در حوزه امنیت است تا در نتیجه آن بتوان میزان جرائم تخصیصی بر سه متغیر را مشاهده نمود و تأثیر آن را بر میزان بازدهی متغیرهای کارایی، دسترس‌پذیری و امنیت و سنجش میزان رضایت مشتری به‌دست آورد.

۳- معماری زیرساخت و مدل امنیتی

۳-۱ معماری زیرساخت

در دنیای امروز، بسیاری از پیام‌ها به‌واسطه خطوط بی‌سیم انتقال می‌یابند. سناریوهای مختلفی جهت برقراری امنیت عنوان شده و یکی از این معماری‌ها استفاده از SOA[1] است که روشی جدید برای فراهم‌سازی سرویس در شبکه توزیع می‌باشد. سرویس‌ها در این حالت از شبکه‌ها و دامین‌های متفاوتی ارسال می‌شوند که باعث پیچیده‌سازی برنامه‌ها و لایه‌های پردازش می‌شوند و باید به‌صورت مدیریت‌شده و لایه‌بندی‌شده به کار خود ادامه دهند [۱۹]. SOA علاوه بر قابلیت پردازش لایه‌ها و برنامه‌های پیچیده می‌تواند خصوصیت‌های سیستم را نیز اصلاح نماید.

در کنار این معماری، یک سرویس‌دهنده CSP می‌تواند امور جزئی همانند دریافت و ذخیره‌سازی داده یا کارهای پیچیده‌تر همانند یافتن و پرینت عکس‌ها و دسته‌بندی داده‌ها را انجام دهد. انعطاف‌پذیری، پشتیبانی و معتبربودن در تمامی این کارها برای رسیدن به یک سیستم توزیع‌شده با کمترین میزان وابستگی مهم است. در این حالت ابر از کارگزار به‌منظور مدیریت و کنترل ارتباطات و همچنین اجازه به مشتریان در جهت هماهنگی بیشتر با CSP کمک می‌گیرد. معماری شبکه ابری بر اساس سرویس (SOCCA)[2] به‌عنوان یک مفهوم که هم‌زمان شبکه ابر و SOA را هماهنگ و یک معماری سلسله‌مراتبی را ارائه می‌نماید، معرفی می‌شود. در این معماری در بالاترین لایه، SOA و کارگزارها قرار دارند. شکل ۱ معماری کار را نشان می‌دهد. این گزارش بر لایه مدیریت سرور و انتقال داده‌ها از طریق ابر تمرکز دارد.

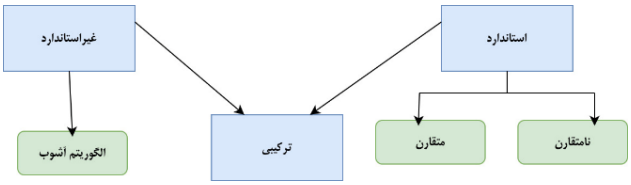
۳-۲ مدل امنیتی

تمرکز این مقاله روی پیشگیری از سرقت و آسیب به داده و سرویس است. روش‌های متنوعی برای انعطاف‌پذیری سیستم در زمان حمله وجود دارند [۲۰]؛ ولی در این مقاله به دنبال استفاده از روش پیشگیری هستیم.

روش‌های رمزگذاری از ایجاد خطا در داده‌ها جلوگیری می‌کنند. در روش‌های استاندارد، داده به بلوک‌های مساوی تقسیم و به روش‌های متقارن[3] و نامتقارن[4] کلید تولید می‌شود. روش‌های غیراستاندارد نیز از الگوریتم آشوب استفاده می‌کنند. در روش ترکیبی از هر دو روش استاندارد و غیراستاندارد توأماً استفاده می‌شود [۵]. شکل ۲ نمودار سلسله‌مراتب مربوط به این روش‌ها را نمایش می‌دهد. هر ابر می‌تواند در برابر موقعیت‌های داخلی و خارجی آسیب‌پذیر باشد [۲۱]. امنیت خارجی به امنیت داده، دخالت در مقدار داده، انبار داده و میزان دسترسی کاربران به داده بستگی دارد [۲۲]. دامنه کاری این مقاله نیز بر روی امنیت خارجی بر

1. Service-Oriented Architecture

2. Service-Oriented Cloud Computing Architecture
3. Symmetric
4. Asymmetric



شکل ۲: سلسله‌مراتب متقارن و نامتقارن.

اساس میزان درخواست داده و سرویس متمرکز است. علاوه بر آن، یک الگوریتم بهینه به نوع درخواست مشتری و بودجه آن بستگی دارد [۲۳]. کاربران همواره در انتخاب سیاست امنیتی بر اساس نیازمندی و زیرساخت نرم‌افزاری و سخت‌افزاری مورد نیاز خود دچار چالش هستند. یک انتخاب درست می‌تواند از افزایش هزینه جلوگیری کرده و انعطاف‌پذیری سرویس را نیز تأمین نماید.

به‌منظور سنجیدن متغیر امنیت، پارامترهایی مانند فایروال، رمزها، مسیریاب‌ها، ترافیک شبکه، IP و خرابی یا گم‌شدن داده‌ها در نظر گرفته می‌شود. خطا و نشت اطلاعات در هر یک از این عوامل باعث بروز خسارت و در نتیجه محاسبه جریمه تخصیصی به سیستم می‌شود. بدین ترتیب هر پارامتر یک ضریب تأثیر خواهد داشت که مجموع این ضرایب، جریمه قابل تحمیل به متغیر امنیت را نمایش می‌دهد. به همین ترتیب متغیرهای کارایی و دسترس‌پذیری نیز دارای پارامتر/ پارامترهایی هستند که مجموع این پارامترها به همراه ضریب تأثیر خود، وزن (۱) را محاسبه می‌نماید. فرمول (۲) میانگین مجموع وزن‌هایی را که در یک متغیر می‌تواند به سیستم تحمیل شود نشان می‌دهد

$$W = aw_1 + bw_2 + \ldots + zw_n \qquad (۱)$$

در این حالت $a,b,\ldots,z$ نشان‌دهنده ضریب تأثیر و مقدار آنها همواره بین صفر تا یک است. $W_1,\ldots,W_n$ وزن‌های متغیر هستند که مقادیرشان بین صفر تا نه است؛ بنابراین وزن نهایی به‌صورت زیر محاسبه می‌گردد

$$W_{total} = \frac{W}{n} \qquad (۲)$$

در این قسمت مقدار n نشان‌دهنده تعداد کل پارامترهای مؤثر در یک متغیر است. بیشترین مقدار هر یک از این متغیرها عدد نه است که در محاسبه جریمه تخصیصی در یک متغیر در نظر گرفته می‌شود.

هدف اصلی این مقاله، ارسال داده به‌صورت رمزگذاری‌شده به‌گونه‌ای است که قابلیت دزدیده‌شدن یا خوانده‌شدن توسط سارقین را از دست داده باشد؛ بنابراین روش پیشنهادی می‌تواند به‌عنوان روشی مناسب در زمان انتقال داده‌ها به‌صورت دسته‌ای مورد استفاده قرار بگیرد.

در کنار این مبحث به‌دنبال بررسی میزان دسترس‌پذیری و کارایی در زمان استفاده از الگوریتم فازبندی‌شده امنیت هستیم. استفاده از این ترفند می‌تواند بر روی سنجش رضایت مشتری تأثیر گذارد و ما می‌خواهیم با بررسی این موضوع، اندیس رضایتمندی مشتریان را بررسی کنیم.

٤- طرح کلی روش پیشنهادی

کارگزار در زمان کار CSP، مسئول جابه‌جایی پیام‌ها بین مشترکین است. داده‌ها جهت انتقال به کارگزار داده می‌شوند و CSP جهت انجام سرویس درخواستی، داده‌ها را از کارگزار دریافت می‌کند. استفاده از سرآیند برای افزایش امنیت پک‌ها با استفاده از مدیریت امنیت بروکر (SB)[1] انجام می‌شود. داده‌ها با استفاده از کلید عمومی یا کلید خصوصی انتقال داده می‌شوند. اگرچه سینگ بِیل[2] [۱] پیشنهاد سه مؤلفه برای افزایش امنیت را داده است، این روش وقت‌گیر بوده و ایجاد افزونگی در زمان ساخت کلید را به دنبال خواهد داشت. امورتی[3] [۲۴] نیز یک لایه جدید برای هر خط و سایت به‌وسیله کارگزار را پیشنهاد داد که به‌عنوان راه حلی برای پروژه‌های تکنولوژی IT عمل می‌نماید. این روش از یک حافظه ردیاب برای حل مشکل سربار استفاده می‌کند؛ با این حال نویسنده هنوز با چالش پشتیبانی از ابر مواجه است.

یکی از روش‌های رمزگذاری انتقال داده‌ها، استفاده از نگاشت آشوب است. نگاشت آشوب با استفاده از کارایی مناسب، یک تکنیک استاندارد را در این زمینه پیشنهاد می‌دهد [۵]. فرمول (۳)، نحوه نگاشت و بازگشت از نگاشت آشوب است. طبق (۳)، یک نقطه با موقعیت x و y با نقشه a,b,c,d به نقطه x_1 و y_1 می‌رسد

$$\begin{bmatrix} x_{n+1} \\ y_{n+1} \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \times \begin{bmatrix} x_n \\ y_n \end{bmatrix} \bmod n$$
$$x_{n+1} = (a \times x_n) + (b \times y_n) \qquad (۳)$$
$$y_{n+1} = (c \times x_n) + (d \times y_n)$$

در صورتی که $gcd(A,n) = 1$ باشد، روش بازگشت به‌صورت ماتریس (۴) به‌صورت زیر است

$$\begin{bmatrix} x_n \\ y_n \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} \times \begin{bmatrix} x_{n+1} \\ y_{n+1} \end{bmatrix} \qquad (۴)$$

با استفاده از این روش می‌توان داده را به‌صورت فازبندی‌شده از طریق بخش مدیریت امنیت کارگزار ارسال نمود. این مقاله از روش‌های قطعه‌بندی، مبهم‌سازی، نگاشت آشوب و کدهای ناهمگام برای افزایش سطح امنیت استفاده می‌کند.

شکل ۳ روند حرکت داده از ابتدا تا پردازش را نشان می‌دهد. R_1 تا R_n میزان درخواست‌های کلاینت از SB و US_1 تا US_i نیز میزان داده‌های غیرواقع هستند که به‌صورت تصادفی برای مبهم‌سازی تولید می‌شوند. هر درخواست می‌تواند به‌صورت داده، عکس یا فیلم باشد که در ماتریس M_1 تا M_n جایگذاری می‌شود.

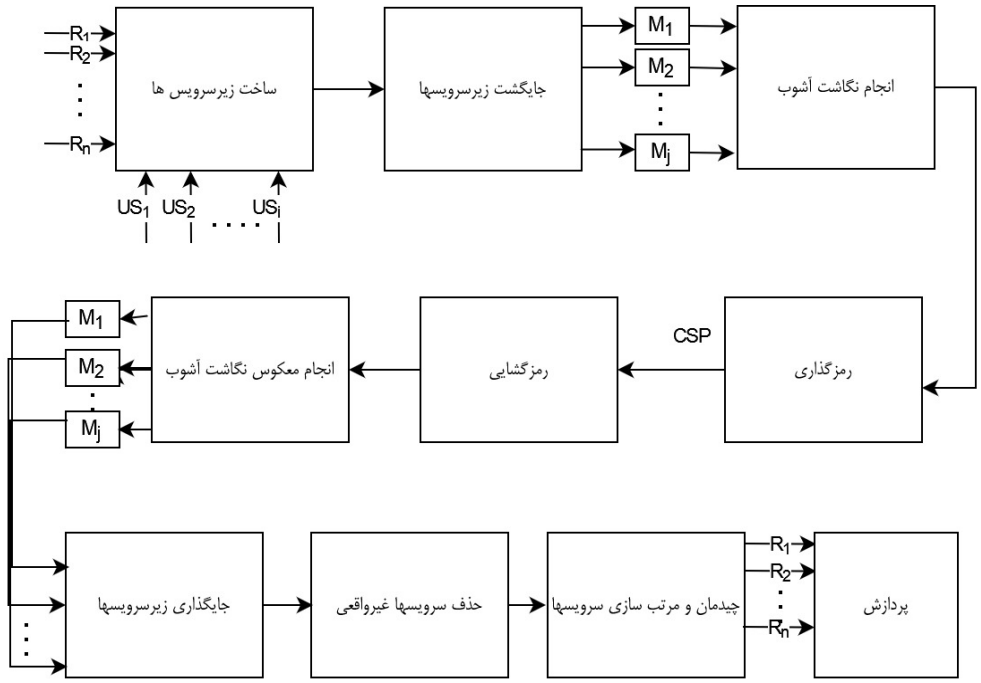
٥- روش پیشنهادی

در این بخش به معرفی فازهای روش پیشنهادی (MLS) می‌پردازیم. شکل ۴ نشان‌دهنده روابط بین مشتریان، کارگزار و سرویس‌دهنده است. همان طور که مشخص است بعد از آن که مشتری درخواست خود را به کارگزار ارسال کرد، فرایند رمزنگاری در بخش مدیریت امنیت بروکر (SB) انجام و پیام به کارگزار سمت CSP ارسال می‌شود. کارگزار پس از دریافت درخواست بر اساس نوع درخواست، سرویس‌دهنده مناسب را پیدا می‌کند. سپس با فرض این که کارگزار نسبت به نحوه یافتن و ارسال پیام به CSP مطلع است، نسبت به ارسال آن از طریق رویکرد سیاست امنیتی اقدام می‌کند. بر این اساس، چهار فاز انجام فرایند به‌صورت زیر است:

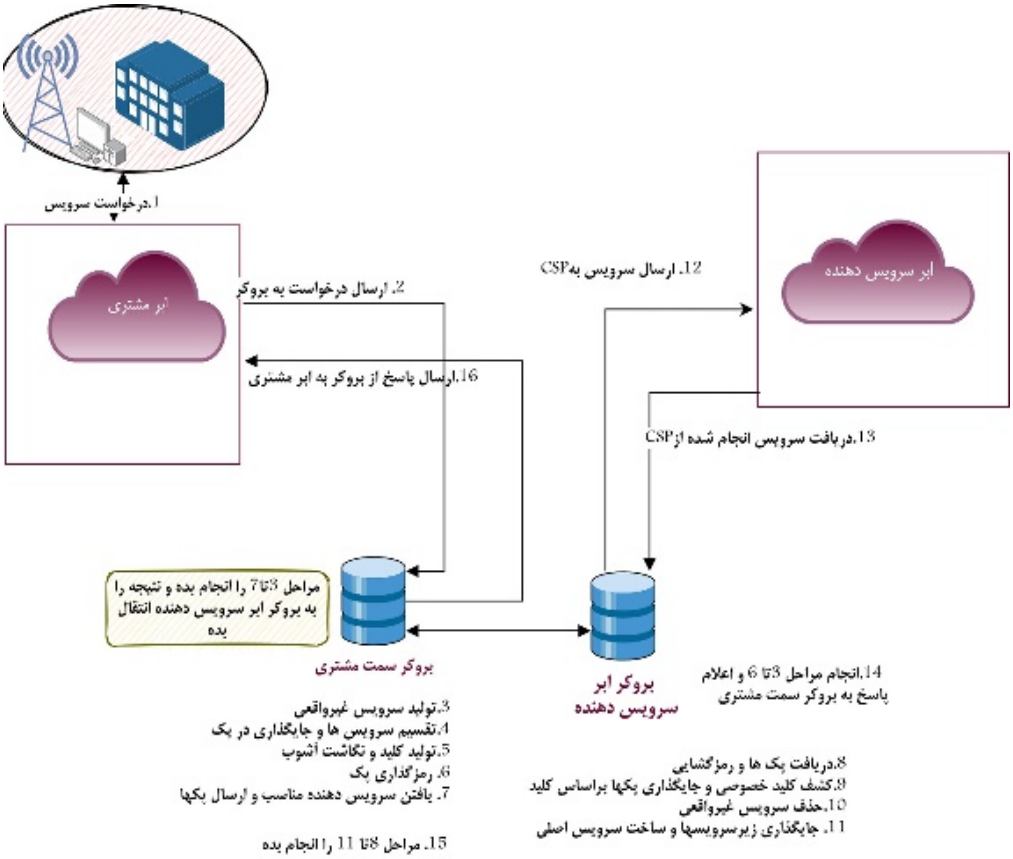
A) فاز بلوک‌بندی داده‌ها: این مرحله به‌منظور دسته‌بندی و تقسیم‌بندی سرویس‌ها به زیرسرویس انجام می‌شود که شامل مراحل زیر است:

۱) هر سرویس به n زیرسرویس تقسیم می‌شود. حداقل تعداد تقسیم باید بزرگ‌تر از سه باشد.

1. Security Broker
2. Singh Bail
3. Amoretti



شکل ۳: نمای کلی از رویکرد مقاله.



شکل ۴: معماری ارسال و دریافت سرویس بر اساس رویکرد MLS.

۲) تمام زیرسرویس‌های یک سرویس به‌صورت تقسیم‌شده در یک ستون قرار می‌گیرند تا در نهایت از جایگذاری n سرویس، یک ماتریس $n \times n$ ساخته شود.
۳) اولین بخش از سرویس اول در اولین ستون و اولین ردیف ماتریس جایگذاری می‌شود.
۴) دومین بخش از سرویس اول در اولین ستون و دومین ردیف ماتریس جایگذاری می‌شود.
۵) این کار به همین ترتیب ادامه می‌یابد تا اولین ستون ماتریس تکمیل گردد و تمام قطعات سرویس اول در ماتریس قرار گیرند.
۶) دومین سرویس به همین ترتیب در تمام خانه‌های ستون دوم ماتریس جایگذاری می‌شود.
۷) این کار ادامه می‌یابد تا زمانی که n خانه ماتریس تکمیل شود.

الگوریتم ۱ (شکل ۵) نحوه انجام آن را نمایش می‌دهد.

ورودی: داده
خروجی: جایگذاری داده در ماتریس مربعی

```
initialization
catch n data
while n is not equal 0 do
    calculating maximum size of n's service
    data is divided to n sub services(Eq.5)
    while matrix is not full do
        for i = 1, i = n, i++ do
            for j = 1, j = n, j++ do
                data_i set in cell_ij of matrix
                if data_ij = null then
                    set cell_ij = null
                endif
            endfor
        endfor
    endwhile
endwhile
```

شکل ۵: جایگذاری داده.

توجه شود که مقدار n نشان‌دهنده تعداد زیرسرویس‌هایی است که در یک ماتریس $n\times n$ جایگذاری می‌شوند. حداقل تعداد n، سه است و تمامی داده‌ها در یک ماتریس $n\times n$ جایگذاری می‌شوند؛ لذا هنگامی که میزان تقسیم‌بندی زیرسرویس‌ها به تعداد n نرسد، مابقی سلول‌های باقیمانده ماتریس با مقدار null پر می‌شوند.

فرمول (۵) نشان‌دهنده نحوه محاسبه مقدار n در یک ماتریس است

$$\text{if } x \text{ then } n=3 \text{ else } n=\sqrt{\frac{\sum x}{k}} \qquad (۵)$$

که در آن، n طول ماتریس، x طول هر سرویس و k تعداد کل سرویس‌ها است.

B) فاز مبهم‌سازی: استفاده از روش مبهم‌سازی در مخفی‌کردن داده‌ها به‌عنوان روشی مناسب مطرح است. مبهم‌سازی به معنای خوانانبودن داده بوده و به‌منظور محافظت از داده به‌کار می‌رود. البته داده بعد از بازسازی، مجدداً قابل استفاده است. از ابزارهای تغییر ظاهر داده‌ها، الگوریتم‌هایی مانند ابزار ابهام‌سازی دن جاوا اسکریپت[1] [۱۴] است؛ البته در این مقاله با تولید داده‌های غیرواقعی (کدهای غیرواقعی) سعی در ایجاد ابهام داریم تا در صورت سرقت پک‌ها و حتی بازخوانی کل پک‌ها، سارقین قادر به تشخیص داده‌های صحیح از داده‌های غیرواقعی نباشند.

تولید مداوم این نوع داده‌ها باعث سنگین‌شدن حجم ارسال و دریافت می‌شود. در این مقاله پیشنهاد شده تا داده‌های غیرواقعی به‌صورت اتفاقی[2] تولید شوند که شامل اعداد، حروف و کاراکتر هستند و ظاهری مشابه داده واقعی دارند. ضمن اینکه این نوع داده‌ها همانند سایر داده‌ها طبق فازهای تعریف‌شده قطعه‌بندی، رمزگذاری و ارسال می‌گردند و تنها سرویس‌دهنده و سرویس‌گیرنده قادر به تفکیک و حذف آنها هستند. در مجموع این مقاله به دنبال پیشنهاد راهی است که ضمن ایجاد ایمنی در داده‌ها، سربار ایجادشده در این روش را در حد پایین نگه دارد.

الگوریتم مبهم‌سازی به‌صورت زیر است:

۱) سیستم، عدد تصادفی صفر یا یک را تولید کرده و اگر یک بود، یک داده غیرواقعی تولید کرده و طبق الگوریتم بخش A در داخل ماتریس قرار می‌دهد.

1. JavaScript Dan's Obfuscator Tool
2. Random

جدول ۲: نحوه جابه‌جایی تگ‌ها.

Tag	Tag B۱			Tag C۱			Tag A۱			Tag D۱		
Tag address	۴	۲	۱	۴	۲	۱	۴	۲	۱	۴	۲	۱
P_i	۲	۳	۵	۷	۱۱	۱۳	۱۷	۱۹	۲۳	۲۹	۳۱	۳۷

۲) بخش هِدر[3] یک پرچم تگ را به نشانه داده غیرواقعی مثبت می‌کند تا در زمان رسیدن توسط CSP قابل تشخیص و حذف باشد.

C) فاز ساخت نگاشت هوشمند آشوب و تولید کلید خصوصی: با توجه به این که هر بخش دارای تگ و مکان خاص است می‌توان با استفاده از مکان‌نمای هر سطر ماتریس و جابه‌جانمودن سلول‌ها با استفاده از نگاشت آشوب، مکان جدید برای داده‌های هر سطر تولید کرد.

B۱ (sub)	C۱ (sub)	A۱ (sub)	D۱ (sub)

هر سرویس به‌ترتیب دارای مکان مشخص و ترتیبی است. مثلاً سرویس الف دارای شماره یک و سرویس ب دارای شماره دو است. در صورتی که شماره‌های آنها به کدهای باینری تغییر کنند می‌توان کدهای باینری آنها را در یک ردیف به‌ترتیب قرار داده و اندیس‌های مربوط به مکان‌ها را در یک نگاشت به‌صورت (۶) ضرب نمود

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1y} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2y} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{x1} & a_{x2} & a_{x3} & \cdots & a_{xy} \end{bmatrix} \qquad (۶)$$

هر اندیس x و y به‌ترتیب در $\begin{bmatrix}1\\k\end{bmatrix}$ ضرب می‌شود (k فرد است) تا یک مکان جدید تبدیل گردد. در این وضعیت مقدار جدید برابر با (۷) است

$$\begin{bmatrix} x_1+ky_1 & x_1+ky_2 & \cdots & x_1+ky_j \\ x_2+ky_1 & x_2+ky_2 & \cdots & x_2+ky_j \\ \vdots & \vdots & \ddots & \vdots \\ x_i+ky_1 & x_i+ky_2 & \cdots & x_i+ky_j \end{bmatrix} \qquad (۷)$$

سایز ماتریس $n\times n$ است. در این وضعیت مقدار x و y به‌صورت زیر محاسبه می‌شوند

$$a_{xy}x_n=(x_i+y_j) \bmod n \qquad (۸)$$

که n اندازه ماتریس است و اندیس‌های x و y نشان‌دهنده مکان‌های جدید تگ‌ها هستند. جدول ۲ نحوه جابه‌جایی تگ‌ها را نشان می‌دهد

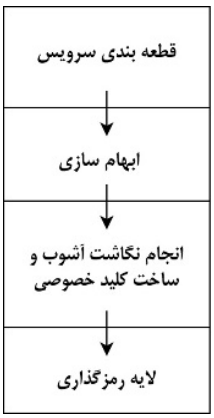
$$Private\ key(PK)=\sum Bit(Tag\ a_i)\times P_i \qquad (۹)$$

در این وضعیت تگ‌ها در یک عدد اول به‌ترتیب ضرب می‌شوند. نحوه ضرب‌نمودن در عدد اول و به‌دست آوردن کلید اولیه به‌صورت (۱۰) است

$$\begin{bmatrix} pk_1 \\ pk_2 \\ \vdots \\ pk_n \end{bmatrix} = \begin{bmatrix} tag_{11} & tag_{12} & \cdots & tag_{1n} \\ tag_{21} & tag_{22} & \cdots & tag_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ tag_{n1} & tag_{n2} & \cdots & tag_{nn} \end{bmatrix} \times \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{bmatrix} \qquad (۱۰)$$

هر تگ با استفاده از یک عدد اول، یک کلید اولیه دریافت می‌کند که داده را به مکان جدید منتقل می‌نماید. کلید اولیه برای سرویس‌دهنده مقصد ارسال شده تا CSP قادر به بازیابی و دریافت تگ‌ها به حالت اولیه باشد.

3. Header



شکل ۶: لایه‌های فازبندی‌شده امنیت (MLS).

در جدول ۲، p عددی اول است که با الگوریتم دِفی هلمن[1] در سلول خاص جایگذاری می‌شود. مطابق با الگوریتم ویدمن[2] [۸] به‌عنوان یکی از روش‌های مؤثر در ارسال داده، هر ماتریس در سه مرحله به یک بردار تبدیل می‌گردد. این کار به‌منظور ارسال‌کردن یک پردازش به یک CSP انجام می‌شود.

الگوریتم ویدمن تولیدکننده سیستم خطی برای فضای محدود است. در صورتی که ماتریس M دارای یک ماتریس $n\times n$ باشد و x یک بردار تصادفی باشد، مقدار $S=[x,Mx,M^{۲}x,\dots]$ نشان‌دهنده بردار ترتیبی است که یک بردار را برای ماتریس M تولید می‌کند. در صورتی که z بردار دیگری با طول n باشد، بردار $S_z=[z.x,z.Mx,z.M^{۲}x,\dots]$ تولید می‌شود؛ به طوری که $w=\sum_{k=۰}^{n} zM^{k}x$ است که چندجمله‌ای w با مقدار حداقل است و مقدار $n_{۰}$ از مقدار n بزرگ‌تر نیست.

پس از آنکه داده در جایگشت جدید جایگذاری شد با استفاده از الگوریتم ویدمن، تبدیل به بردار می‌شود و به‌صورت ردیف به ردیف برای CSP ارسال می‌گردد. تفاوت رویکرد ما با الگوریتم ویدمن در این است که مقدار بردار به‌دست آمده تبدیل به یک رشته نمی‌شود و به‌صورت یک ردیف برای مقصد ارسال می‌گردد.

D) فاز رمزگذاری: در این مرحله، هر ماتریس که مشتمل بر سرویس رمزگذاری‌شده است با استفاده از الگوریتم رمزگذاری متقارن یا نامتقارن برای ارسال آماده می‌شود؛ به‌گونه‌ای که SB قادر به رمزگشایی و تفکیک سرویس‌های رمزگذاری از یکدیگر است.

شکل ۶ فازهای انجام‌دادن هر مرحله را نشان می‌دهد. مقدار کلید به‌دست آمده که برای CSP ارسال می‌شود، یکتاست. در صورت مفقودی هر یک از پک‌ها، گیرنده می‌تواند دوباره درخواست خود را به کارگزار ارسال نماید.

۵-۱ مثال

شرح رویکرد پیشنهادی با یک مثال در ادامه آمده است. با فرض اینکه تمامی سرویس‌ها دارای طول واحد ۸ باشند، طول هر زیرسرویس ۴ است؛ بنابراین سرویس‌ها و سرویس‌های غیرواقعی به ۴ زیرسرویس تقسیم و جایگذاری می‌شوند. در صورتی که هر زیرسرویس در یک سلول به‌صورت جدول ۳ تقسیم‌بندی شود، بر اساس قسمت A، زیرسرویس‌های یک سرویس در یک ستون ماتریس جایگذاری گردیده و نحوه جایگذاری در ذیل آمده است

جدول ۳: نحوه تقسیم‌بندی سرویس.

segmentation	sub ۱	sub ۲	sub ۳	sub ۴
Service one	A۱	A۲	A۳	A۴
Service two	B۱	B۲	B۳	B۴
Service three	C۱	C۲	C۳	C۴
Service four	D۱	D۲	D۳	D۴

A۱	B۱	C۱	D۱
A۲	B۲	C۲	D۲
A۳	B۳	C۳	D۳
A۴	B۴	C۴	D۴

نگاشت تغییر مکان به فرم ماتریس (۱×۲) است و با فرض نگاشت $\begin{bmatrix}۱\\۱\end{bmatrix}$، اندیس‌ها به‌صورت زیر تغییر می‌کنند

$$\begin{bmatrix} x_{۱}+y_{۱} & x_{۱}+y_{۲} & \dots & x_{۱}+y_{n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n}+y_{۱} & x_{n}+y_{۲} & \dots & x_{n}+y_{n} \end{bmatrix} \quad (۱۱)$$

مقدار x و y به‌صورت فرمول زیر محاسبه می‌گردند

$$X_{n}=x_{i}+y_{i} \bmod ۴$$
$$\text{if } x_{n}=۰\rightarrow x_{n}=n \quad (۱۲)$$

جایگذاری مقادیر در مکان‌های جدید به‌صورت زیر است

B۱	C۱	D۱	A۱
C۲	D۲	A۲	B۲
D۳	A۳	B۳	C۳
A۴	B۴	C۴	D۴

هر زیرسرویس یک مکان جدید و اندیس‌هایی مشخص به‌صورت زیر دارد

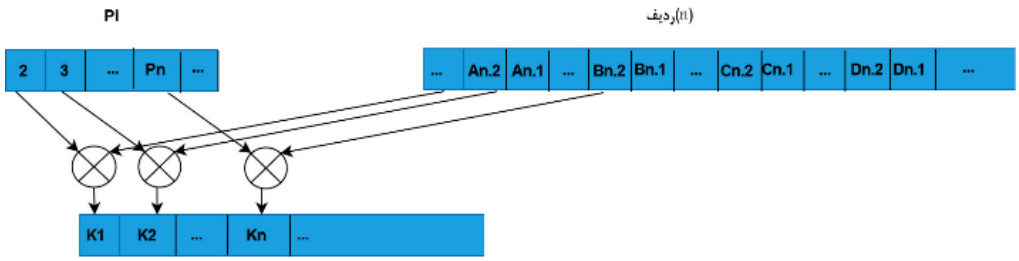
$$A_{۱}=۱\ ,\ B_{۱}=۲\ ,\ C_{۱}=۳\ ,\ D_{۱}=۴\ ,\ \dots$$
$$A_{۲}=۱\ ,\ B_{۲}=۲\ ,\ C_{۲}=۳\ ,\ D_{۲}=۴\ ,\ \dots$$

تمام اندیس‌ها به کد باینری تبدیل و در یک عدد اول ضرب می‌شوند تا جدول ۴ به‌دست آید

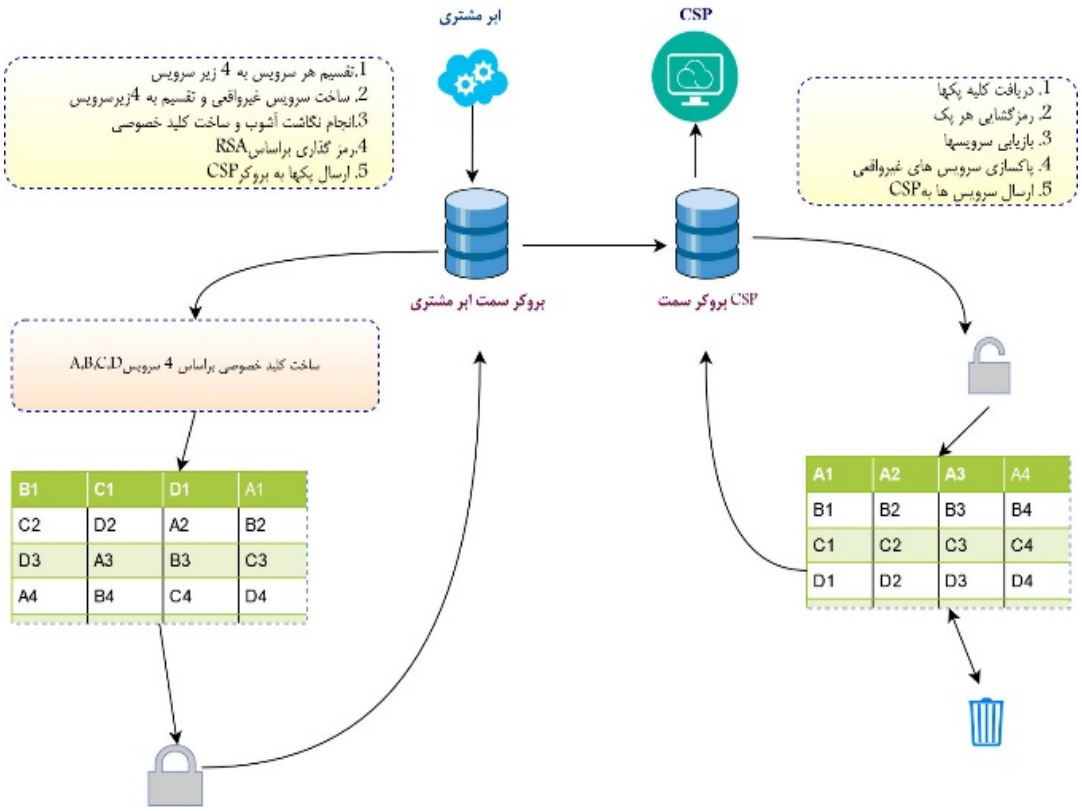
$$\begin{bmatrix}۲۶۹۸۴۱\\۷۴۸۶۵\\۴۷۴۱۴۵\\۵۶۶۶۱۸\end{bmatrix}=\begin{bmatrix}۰&۱&۰&۰&۱&۱&۱&۰&۰&۰&۰&۱\\۰&۱&۱&۱&۰&۰&۰&۰&۱&۰&۱&۰\\۱&۰&۰&۰&۰&۱&۰&۱&۰&۰&۱&۱\\۰&۰&۱&۰&۱&۰&۰&۱&۱&۱&۰&۰\end{bmatrix}\times\begin{bmatrix}۲\\۳\\۵\\۷\\۱۱\\۱۳\\۱۷\\۱۹\\۲۳\\۲۹\\۳۱\\۳۷\end{bmatrix} \quad (۱۳)$$

طبق (۱۴)، اگر طول کدهای باینری ۲۳ باشد در یک ماتریس ۴×۴، ۱۲ بار ضرب می‌شود. شکل ۷ نشان‌دهنده نحوه تولید یک کلید خصوصی توسط ضرب یک ردیف در یک عدد اول است

1. DiffieHellman Algorithm
2. Wiedemann's Algorithm



شکل ۷: تولید پیش کلید در یک ردیف.



شکل ۸: مثال معماری ارسال و دریافت داده در رویکرد MLS.

جدول ۴: ضریب کد باینری با عدد اول.

Tag B_1			Tag C_1			Tag D_1			Tag A_1		
۰	۱	۰	۰	۱	۱	۱	۰	۰	۰	۰	۱
Tag C_2			Tag D_2			Tag A_2			Tag B_2		
۰	۱	۱	۱	۰	۰	۰	۰	۱	۰	۱	۰
Tag D_3			Tag A_3			Tag B_3			Tag C_3		
۱	۰	۰	۰	۰	۱	۰	۱	۰	۰	۱	۱
Tag A_4			Tag B_4			Tag C_4			Tag D_4		
۰	۰	۱	۰	۱	۰	۰	۱	۱	۱	۰	۰
۲	۳	۵	۷	۱۱	۱۳	۱۷	۱۹	۲۳	۲۹	۳۱	۳۷

$$\text{if} \begin{cases} ۱ & \text{then} \quad tag_i \times P_i = ۰ \\ tag_i \times P_i & \text{otherwise} \end{cases} \qquad (۱۴)$$

۲-۵ فضای کلید

در مثال بیان‌شده، مقدار فضای کلید به‌دست آمده $۲^{۲۰} \cong ۱۰۷۲۴۴۵$ و حداقل مقدار کلید $۲^{۱۶} \geq ۷۳۸۱۵$ است. اگر طول هر لیبل را ۸ کاراکتر در نظر بگیریم، مقدار $P_i \geq ۲^{۲۳}$ و مقدار کلید برای قسمت اول بزرگ‌تر از $۲^{۸} \times ۲^{۲۰} = ۲^{۲۸}$ می‌شود؛ بنابراین برای چهار قسمت محاسبه‌شده، مقدار کلید $۲^{۵۴} \times ۲^{۵۲} \times ۲^{۴۴} \times ۲^{۲۸} \cong ۳٫۸ \times ۱۰^{۵۳} = ۲^{۱۷۸}$ است؛ بنابراین فضای کلید برای هر قسمت $۲^{۱۷۸}$ می‌باشد. در صورتی که فضای کلید استاندارد را به اندازه $۲^{۱۲۸} \cong ۳٫۴ \times ۱۰^{۳۸}$ [۵] در نظر بگیریم، اندازه کلید مطلوب است. هر رایانه در طول یک سال می‌تواند $۳٫۱ \times ۱۰^{۷}$ پردازش در هر ثانیه انجام دهد و سریع‌ترین رایانه می‌تواند $۲^{۸۰}$ یا $۱٫۲ \times ۱۰^{۲۴}$ پردازش در ثانیه را انجام دهد ($۳٫۸ \times ۱۰^{۳۱}$ پردازش در یک سال) [۵]؛ بنابراین کلید ما $(۳٫۸ \times ۱۰^{۵۳}) / (۱٫۲ \times ۱۰^{۲۴}) = ۳٫۱۶ \times ۱۰^{۲۹}$ ثانیه نیاز به رمزگشایی دارد. یک سارق نیازمند $۲^{۱۷۸} / ۲^{۸۰} = ۲^{۹۸}$ ثانیه یا $۱۰^{۲۲}$ سال جهت رمزگشایی است؛ بنابراین فضای کلید پیشنهادی به اندازه کافی قوی است. جدول ۵ مقایسه بین فضای کلید تولیدشده در سایر روش‌ها و روش ما را نشان می‌دهد. در این وضعیت در صورتی که فضای کلید بالاتر از استاندارد باشد، حد مطلوب را کسب نموده است.

همان گونه که بیان گردید، هر سرویس پس از آن که به زیرسرویس تقسیم و مبهم‌سازی شد، وارد فاز تولید کلید و آماده‌سازی برای ارسال به‌صورت بردار می‌شود و نهایتاً الگوریتم RSA بر روی آن اجرا می‌گردد. کلیه مراحل عنوان‌شده را می‌توان در شکل ۸ مشاهده نمود.

ما برای نمایش ساده‌تر الگوریتم‌ها از نمادهایی جهت معرفی متغیرها استفاده می‌کنیم. جدول ۶ نشان‌دهنده نحوه نام‌گذاری متغیرهاست.

ورودی: n تولیدشده از (۵)
خروجی: ارسال پک‌ها و کلید خصوصی به کارگزار

```
1 : initialization
2 : Set header for each pack
3 : while element of list is greater than 0 do
4 :   EP_i is divided to n sub services (Eq.5)
5 :   Select each sub and arrange cells in the array(p) sequentially
6 :   if y_i = null then
7 :     set all remain column_i by null
8 :   endif
9 :   if service is traced then
10 :    make random number (0or1)
11 :  endif
12 :  if A = 1 Make FS and arrange in the array then
13 :    Divide FS to n subs that has calculated
14 :    Select each sub and arrange in the array's cells (p) sequentially
15 :  endif
16 :  go to the next service
17 :  pack list
18 :  Set T and permute them in cells and make PK(Algorithm3) and arrange in IP
19 : endwhile
20 : while pack is greater than 0 do
21 :   encrypt pack by RSA
22 :   Send Packs and PK to the Broker
23 : endWhile
```

شکل ۹: رمزگذاری سرویس در بخش امنیت کارگزار مشتری.

ورودی: ماتریس $n \times n$
خروجی: ساخت کلید خصوصی سطر به سطر

```
1 : get T of each matrix
2 : select k
3 : Pi ← 0
4 : for i = 1 to n do
5 :   for j = 1 to n do
6 :     xi_j = i + k * j
7 :     x_ij ← x_ij mod n
8 :     if x_ij = 0 then
9 :       x_ij ← n
10 :    endif
11 :    transition each part to new T;
12 :  endfor
13 :  forz = 1 to n * amountbitword do
14 :        (PK) = (Bitwordtag) * Pi_z * (PK)
15 :        if Bitwordtag == 0 then go next
16 :    endif
17 :  endfor
18 :  save PK and send by each row;
19 :  PK ← 0
20 : endfor
```

شکل ۱۰: جایگذاری قسمت‌ها و ساخت کلید خصوصی (PK).

EP_i نشانه تقسیم‌بندی سرویس با شماره پردازش i است که بر اساس میزان سرویس‌ها و اندازه آنها بر اساس (۵) محاسبه می‌شود. نکته اینکه محاسبه مقادیر بزرگ برای n و در نتیجه ایجاد ماتریس با ابعاد بزرگ ممکن است باعث ایجاد افزونگی و کندی در عملکرد سیستم شود. P_k آرایه موقت است که هر سرویس قبل از جایگذاری در ماتریس بر اساس شماره آرایه (k) در آن قرار می‌گیرد. IP_i آرایه دوبعدی است که هر قسمت را به‌صورت موقت در یک لیست نگهداری می‌کند. تگ‌پک‌ها باید در داخل IP_i نگهداری شوند. در زمان اجرای سیستم، سیستم نیازمند تولید FS_x به‌صورت تصادفی بوده و X نمایش‌دهنده شماره پردازش غیرواقعی است. بر اساس (۷)، کلیدهای خصوصی و نگاشت‌ها جهت رمزگذاری تولید و رمزگذاری RSA بر روی لیست‌های آماده ارسال، اجرا می‌شود تا کارگزار درخواست‌ها را به CSP ارسال نماید.

کارگزار مقصد پس از دریافت پک‌ها، رمزگشایی کلید و برگرداندن سرویس‌ها به حالت اولیه را بر اساس تگ‌های مندرج انجام می‌دهد. کارگزار با استفاده از تگ‌ها، سرویس‌های غیرواقعی را شناسایی و حذف می‌کند. پس از جایگذاری زیرسرویس‌ها در کنار هم، سرویس‌های سالم شناسایی و با استفاده از هِدر به‌صورت یک سرویس کامل برای اجرا ارسال می‌کند [۶].

ورودی: پک‌ها
خروجی: بازساخت سرویس در کارگزار CSP

```
1 : while pack is greater than 0 do
2 :   Get PK and pack and decrypt
3 :   Copy this pack in to a list
4 :   Relocate each cell by PK
5 : end while
6 : while list is greater than 0 do
7 :   Find FS and remove from list
8 :   Get each part of element and arrange together
9 :   Set each element in service part
10 :   if list is empty then
11 :     select next list
12 :   else send fault massage to sender
13 :   endif
14 : endwhile
```

شکل ۱۱: رمزگشایی و ساخت سرویس در کارگزار CSP.

جدول ۵: مقایسه فضای کلید در رویکردهای مختلف.

رویکرد	مجموع فضای کلید
MLS	$۲^{۱۷۸}$
[۲۵]	$۲^{۱۷۰}$
[۲۶]	$۲^{۱۷۰}$
[۲۷]	$۲^{۱۸۶}$

جدول ۶: نمادها و توصیف متغیرهای استفاده‌شده در الگوریتم.

نماد	توضیحات
i	تعداد پردازش‌ها
EP_i	تقسیم هر سرویس به n قسمت
T_i	تگ نشانه‌گذاری سرآیند هر پک
IP_i	آرایه کمکی جهت جایگذاری تکه‌ها در سلول مشخص‌شده
x	شماره پردازنده تصادفی
FS_x	سرویس غیرواقعی
PK	کلید خصوصی تولیدشده با توجه به (۹)
k	شماره آرایه
P_k	آرایه موقت که هر بخش را قبل از ارسال در آن ذخیره می‌کند.
P	عدد اول

الگوریتم ۲ (شکل ۹) نشان‌دهنده نحوه ساخت MLS در سمت مشتری است و الگوریتم ۳ (شکل ۱۰) طریقه ساخت کلید خصوصی در کارگزار را نشان می‌دهد. در واقع الگوریتم ۳، زیربخش الگوریتم ۲ است. الگوریتم ۴ (شکل ۱۱) نحوه بازخوانی و رمزگشایی CSP را نشان می‌دهد.

۶- ارزیابی کارایی و آزمایش‌ها

این قسمت با درنظرگرفتن انجام رخدادها و حملاتی که می‌تواند صورت گیرد، اندازه‌گیری میزان خطای سیستمی و جرائمی را که به سیستم تخصیص می‌یابد اندازه‌گیری می‌کند.

۶-۱ آنالیز کارایی

برای ارزیابی میزان کارایی در الگوریتم MLS با استفاده از زمان اجرا و جرائم تخصیصی، نیازمند آن هستیم که رابطه بین امنیت و میزان رضایت مشتری را با کارایی ارزیابی کنیم. امنیت و کارایی در هر سیستم، نقشی اساسی دارند و افزایش زمان برای رمزگذاری سرویس می‌تواند منجر به کاهش میزان کارایی و دسترس‌پذیری برای پاسخ به مشتری شود. برای ارزیابی کارایی، سه سناریو را در نظر می‌گیریم. اولین سناریو بر اساس

جدول ۷: مقایسه بین خروجی سایبرها.

آزمایش	متوسط درصد جریمه تخصیصی	جریمه تخصیصی	کل جریمه قابل تحمیل	زمان انجام رمزگذاری‌ها (میلی‌ثانیه)*	متوسط زمان انجام سرویس‌ها (میلی‌ثانیه)*
FCFS	۷۶/۴۱	۱۲۰۵	۱۵۷۷	۰	۲۵۴
FCFS (ECC)	۲۰	۲۶۲	۱۳۲۶	۱۷	۳۰۲
MLSRCT	۸۴	۱۱۹۱	۱۴۱۳	۲۶	۲۵۳
MLS	۱۵	۱۹۵	۱۲۹۰	۳۰	۲۵۱

* متوسط زمان انجام رمزنگاری و متوسط زمان انجام سرویس به‌ازای هر ۱۰۰ کاربر محاسبه شده است. زمان رمزنگاری در زمان انجام سرویس محاسبه نشده است.

الگوریتم FCFS[1] و FCFS (ECC)[2] است که در حالت دوم، رمزگذاری ECC بر روی سرویس‌ها اجرا می‌شود تا سیستم ارزیابی، میزان جرائم تخصیصی را مشخص کند. سناریوی دوم بر اساس پیاده‌سازی MLS بدون اختصاص‌دادن زمان برای اجرای عملیات رمزگذاری است. نهایتاً سناریوی سوم MLS با احتساب زمان انجام رمزگذاری پیاده‌سازی می‌شود.

این آزمایش، ضریب تأثیر امنیت، کارایی و دسترس‌پذیری را مورد ارزیابی قرار می‌دهد. بیشترین ضریب تأثیر برای متغیر امنیت و ضرایب پایین‌تر به‌ترتیب برای متغیرهای کارایی و دسترس‌پذیری در سیستم در نظر گرفته می‌شود. سیستم بنا به نیاز خود و اولویت‌بندی به‌صورت اتوماتیک و در صورت عدم تحقق هر یک از متغیرها، جریمه‌ای را که برای هر سرویس (عدد بین ۰ تا ۹) در نظر گرفته است به‌عنوان جریمه، تخصیص می‌دهد که مجموع این جرائم بعد از وزن‌گذاری (فرمول (۲))، میزان جریمه تخصیصی به متغیر را نمایش می‌دهد.

برای متغیر امنیت، پارامترهای دیوار آتشین، رمز، مسیریاب، ترافیک شبکه، امنیت IP، خرابی و گم‌شدن داده در نظر گرفته شده است. متغیر دسترس‌پذیری به معنای زمان در دسترس بودن سیستم و ارائه خدمات نسبت به مجموع زمانی است که سیستم خراب، خاموش، بیکار یا در حال انجام خدمت است. متغیر کارایی نیز به پارامتر برآیند زمان انجام سرویس به زمان کارکرد CPU بستگی دارد.

در زمان پیاده‌سازی روش پیشنهادی در شبیه‌ساز، اولویت‌دهی و شماره‌گذاری بر روی آنها انجام پذیرفته و شبیه‌ساز بر همین اساس، میزان جرائم را محاسبه می‌نماید. ضمن اینکه در قسمت قبل، تعریف کلید خصوصی و معتبربودن فضای کلید نسبت به سایر رویکردها بررسی گردیده و در ادامه، بازدهی سیستم در مقایسه با سایر رویکردها نمایش داده می‌شود. ضمن اینکه بررسی تأثیر روش پیشنهادی بر روی انواع حملات رایج، میزان مقاومت روش پیشنهادی و سناریوهای مرتبط با آن در کارهای آینده بررسی خواهد شد.

۶-۲ شبیه‌سازی

برای انجام‌دادن آزمایش‌ها از شبیه‌ساز کلودسیم[3] برای محاسبه و پیاده‌سازی سیاست‌های اجرایی استفاده شده است. یکی از شبیه‌سازهای پرکاربرد برای پیاده‌سازی، پردازش و ارسال درخواست پیام‌ها در شبکه، کلودسیم است [۲۸] که قادر است تا منابع ابری و وظیفه‌مندی آنها را شبیه‌سازی نماید. این شبیه‌ساز شامل داده‌های کلاس سرور است که برای مدیریت منابع، وظیفه‌مندی و انتقال سرویس‌ها مناسب است [۲۹] و [۳۰]. تمام آزمایش‌ها بر اساس یک ماشین x۶۸، سیستم‌عامل ویندوز و ماشین مجازی‌ساز Xen انجام گردیده و روش تولید داده‌ها به‌صورت تصادفی است.

سیستم با توجه به (۲)، جریمه قابل تحمیل را محاسبه می‌کند و در صورتی که سرویس در اجرای هر متغیر شکست بخورد، عدد جریمه قابل تحمیل در هر متغیر را به‌عنوان جریمه تخصیصی اعلام می‌کند. در واقع جریمه قابل تحمیل، حداکثر میزان جریمه‌ای است که سیستم با توجه به پارامترها و ضرایب آنها برای هر متغیر محاسبه می‌کند و جریمه تخصیصی، میزان جریمه‌ای است که پس از انجام سرویس در صورت شکست در اجرای هر متغیر به‌عنوان جریمه تخصیصی برای هر متغیر به سیستم اعلام می‌کند. مقدار کل جریمه تخصیصی نیز از مجموع جرائم تخصیصی در سه متغیر امنیت، کارایی و دسترس‌پذیری در یک سیستم به‌دست می‌آید. بر این اساس، آزمایش اول با شرایط گفته‌شده و با سیاست FCFS [۳۰] انجام می‌شود. در این حالت نتایج به‌صورت زیر است:

- انجام آزمایش بر اساس سیاست FCFS است.
- زمان انجام عملیات رمزگذاری صفر است؛ زیرا سیاست رمزگذاری در سیستم وجود ندارد.
- متوسط زمان انجام سرویس برای هر درخواست ۲۵۴ میلی‌ثانیه به‌ازای هر ۱۰۰ کاربر است.
- جریمه تخصیصی سیستم ۱۲۰۵ از مجموع ۱۵۷۷ جریمه قابل تحمیل است؛ بنابراین ٪۷۶/۴۱ سرویس‌های انجام‌یافته شامل جریمه تخصیصی شده‌اند.

در صورتی که همین سیاست را با اضافه‌کردن الگوریتم رمزگذاری ECC اجرا نماییم نتایج همانند جدول ۷ می‌شود.

آزمایش دوم بر اساس سیاست لایه‌گذاری MLS بدون اختصاص زمان برای عملیات رمزگذاری انجام شد که آن را MLSRCT[4] نام‌گذاری کردیم. برای انجام آزمایش، سیستم نیازمند زمان جهت انجام عملیات رمزگذاری است و به دلیل اینکه این امکان به آن داده نمی‌شود، سیستم شامل جریمه تخصیصی سنگینی می‌شود. این امر نشان‌دهنده اهمیت تخصیص زمان برای انجام عملیات رمزگذاری در سیستم است.

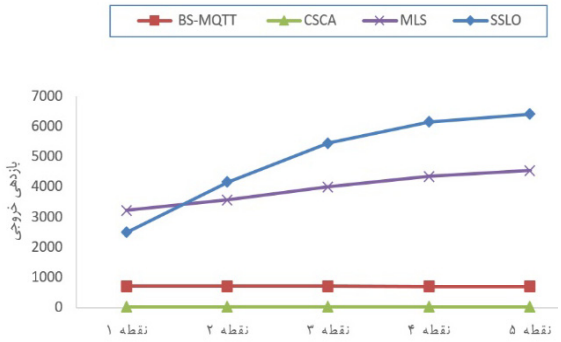
- سیستم برای انجام عملیات رمزگذاری به‌طور متوسط نیازمند ۲۶ میلی‌ثانیه به ازای هر ۱۰۰ کاربر است.
- متوسط زمان انجام هر سرویس ۲۵۳ میلی‌ثانیه است که به لحاظ ایجاد یک محیط استاندارد، تقریباً با متوسط زمان اجرای سایر آزمایش‌ها مشابه است.
- جریمه تخصیصی سیستم ۱۱۹۱ از مجموع ۱۴۱۳ جریمه قابل تحمیل است؛ بنابراین ٪۸۴ سرویس‌های انجام‌یافته شامل جریمه تخصیصی شده‌اند.

این آزمایش بر اساس انجام عملیات رمزگذاری لایه‌ای انجام شده است؛ ولی سیستم زمان اضافه‌ای برای انجام این کار ندارد.

آزمایش سوم بر اساس انجام سیاست لایه‌ای MLS و با افزایش زمان اجرا برای انجام سرویس و عملیات رمزگذاری انجام می‌شود.

- سیستم برای انجام عملیات رمزگذاری به‌طور متوسط نیازمند ۳۰ میلی‌ثانیه به‌ازای هر ۱۰۰ کاربر است.

1. First Come First Service
2. Elliptic Curve Cryptography
3. CloudSim
4. Multi-Layered Security Regardless of Crypto runTime



شکل ۱۲: مقایسه بازدهی خروجی در چهار رویکرد.

- متوسط زمان انجام هر سرویس ۲۵۱ میلی‌ثانیه است که به لحاظ ایجاد یک محیط استاندارد، تقریباً با متوسط زمان اجرای سایر آزمایش‌ها مشابه است.
- جریمه تخصیصی سیستم ۱۹۵ از مجموع ۱۲۹۰ جریمه قابل تحمیل است؛ بنابراین ۱۵٪ سرویس‌های انجام‌یافته شامل جریمه شده‌اند.

در نتیجه سیاست MLS می‌تواند داده‌ها و سرویس‌ها را منتقل نماید و جریمه تخصیصی کمتری بپردازد.

۶-۳ آنالیز خروجی

همان گونه که گفته شد، داده‌ها به‌صورت تصادفی در شبیه‌ساز تولید می‌شوند و میزان زمان انجام سرویس و رمزگذاری نزدیک به یکدیگر است. برای اثبات میزان تأثیرگذاری متغیر امنیت بر روی سیستم، نیازمند اثبات میزان جرائم تخصیصی و بازدهی سیستم هستیم. طبق (۱۵)، کل زمان از ابتدا تا انتها T_t است که شامل T_i (میزان زمان اولیه جهت ارسال درخواست به کارگزار)، T_c (زمان مورد نیاز جهت انجام عملیات رمزگذاری) و T_e (میزان زمان اجرای پردازش) است [۱]

$$T_t = T_e + T_c - T_i \qquad (۱۵)$$

جدول ۷ نتایج به‌دست آمده را نشان می‌دهد. سیستم قبل از شروع انجام سرویس، مقادیر جریمه را برای هر متغیر در نظر می‌گیرد (کل جریمه قابل تحمیل) و در صورت عدم تحقق متغیرها، جریمه در نظر گرفته شده را به آن تخصیص می‌دهد (جریمه تخصیصی). مثلاً در آزمایش MLS حداکثر میزان جریمه‌ای که توسط سیستم پیش‌بینی شده (جریمه قابل تحمیل)، ۱۲۹۰ بوده است. بعد از تکرار ۱۰ مرتبه که در هر مرتبه ۵ سرور و هر سرور ۱۰۰ کاربر را پشتیبانی می‌نماید، متوسط جریمه تخصیصی، ۱۹۵ به سیستم تخصیص داده می‌شود؛ بنابراین نسبت متوسط جریمه تخصیصی به متوسط جریمه قابل تحمیل ۱۵٪ خواهد شد.

برای آن که بتوان سایر متغیرها را با یکدیگر مقایسه نمود، زمان انجام سرویس یکسان‌سازی شده و بر این اساس MLS و MLSRCT نیازمند زمان انجام رمزگذاری هستند. آزمایش دو در واقع همان روش پیشنهادی ماست؛ با این تفاوت که سیستم، دیگر زمانی جهت انجام عملیات رمزگذاری به زمان کل اضافه نمی‌نماید و باعث بروز خسارت و جریمه سنگین به کل سرویس‌ها می‌گردد. در واقع هدف از عنوان آزمایش دوم (MLSRCT) بیان این است که حتی اگر عملیات رمزگذاری به‌درستی صورت پذیرد و از نشت داده در شبکه جلوگیری کند، ولی سیستم زمانی جهت انجام عملیات رمزگذاری اختصاص ندهد، میزان جرائم تخصیصی تا چه اندازه می‌تواند افزایش یابد و حتی از میزان جرائم مربوط به آزمایش FCFS نیز بیشتر گردد و کل سیستم را با مخاطره مواجه سازد.

جدول ۸: متوسط زمان اجرا و رمزگذاری در روش‌های مختلف.

آزمایش	متوسط زمان رمزگذاری (میلی‌ثانیه)	متوسط زمان انجام سرویس (میلی‌ثانیه)	نسبت متوسط زمان رمزگذاری به متوسط زمان انجام سرویس
FCFS	۰	۲٫۵۴	۰
MLSRCT	۰٫۲۶	۲٫۵۳	۰٫۱
MLS	۰٫۳	۲٫۵۱	۰٫۱۲
CSCA [۱۰]	۱٫۶	۲	۰٫۸
BS-MQTT [۱]	۱٫۵	۲٫۴	۰٫۶۳
SSLO [۳۱]	۳٫۷	۷٫۲	۰٫۵۱

جدول ۸ متوسط زمان انجام سرویس روش پیشنهادی را در مقایسه با سایر روش‌های موجود نشان می‌دهد. یکی از مزایای عملیات رمزگذاری، کاهش زمان به‌منظور افزایش قابلیت سرویس‌دهی است. در واقع روشی می‌تواند مؤثر و بهینه باشد که علاوه بر محفوظ نگه‌داشتن داده‌ها، زمان کمتری جهت انجام عملیات رمزگذاری صرف کند که این امر در جدول ۸ در مقایسه با سایر رویکردها آمده است. در این جدول متوسط زمان اجرای عملیات رمزگذاری در دو آزمایش MLS و MLSRCT نسبت به سایرین کمترین است. همچنین نسبت زمان رمزگذاری به زمان اجرا در MLS و MLSRCT در مقایسه با سایر روش‌های موجود کمترین است؛ اما MLS کمترین میزان جریمه تخصیصی نسبت به سایر رویکردها را دارد.

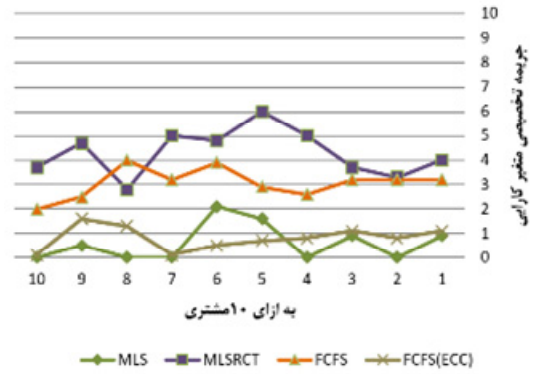
محاسبه بازدهی خروجی (Th) به‌صورت (۱۶) است. M میزان پردازش‌های انجام‌یافته و T زمان انجام پردازش‌هاست

$$Th = \frac{M}{T} \qquad (۱۶)$$

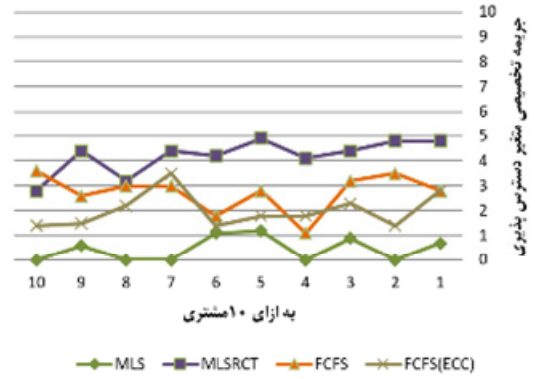
شکل ۱۲ نمایش میزان بازدهی خروجی را بر اساس (۱۶) در رویکردهای متفاوت نشان می‌دهد. به‌منظور نرمال‌سازی میزان خروجی‌ها هر برآیند در هر نقطه محاسبه و نرمال‌سازی شده است. SSLO در هر نقطه میزان CSPهای خود را ۱۰۰ واحد افزایش می‌دهد که این امر باعث افزایش بازدهی در خروجی می‌شود. در نتیجه، بازدهی دو رویکرد MLS و SSLO با اختلاف زیاد نسبت به دو رویکرد دیگر بهتر هستند.

شکل ۱۳ مقایسه‌ای جامع را بین سه آزمایش نشان می‌دهد. شکل ۱۳- الف، مقایسه جریمه تخصیصی در متغیر کارایی، شکل ۱۳- ب، مقایسه میزان جریمه تخصیصی بر اثر نقص متغیر دسترس‌پذیری و شکل ۱۳- ج، جریمه تخصیصی در متغیر امنیت است. هر بخش بر اثر پارامترهای مؤثر، مقدار صفر تا ۹ را می‌پذیرد و بر اثر وزن‌هایی که دریافت می‌کند میزان جریمه تخصیصی محاسبه می‌شود.

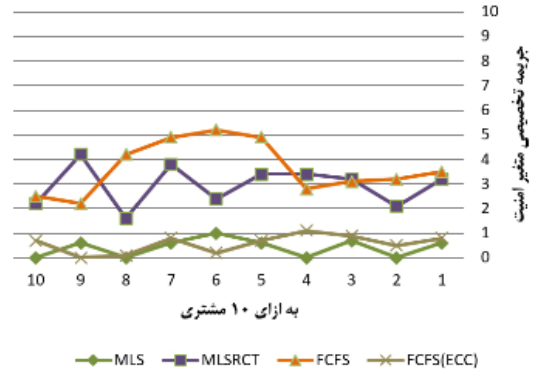
این مقاله از عددگذاری پویا در زمان وزن‌دهی بر اساس پارامترهای هر متغیر استفاده می‌نماید؛ به دلیل آن که ممکن است ضریب دسترس‌پذیری در یک سرویس، بالاتر از کارایی باشد و در سرویس دیگر بالعکس. در حالی که تخصیص اعداد ثابت به سیستم میزان وزن ثابت را تولید می‌نماید و نهایتاً خروجی سیستم همواره یک عدد ثابت خواهد بود. در واقع سیاست تخصیص عدد بر عهده سیستم است که به هر سرویس بنا به نیازهای تعریف‌شده، عددی را به‌عنوان جریمه قابل تحمیل، تخصیص دهد؛ ولی محدوده هر عدد نباید خارج از دامنه ۰ تا ۹ باشد.



(الف)



(ب)

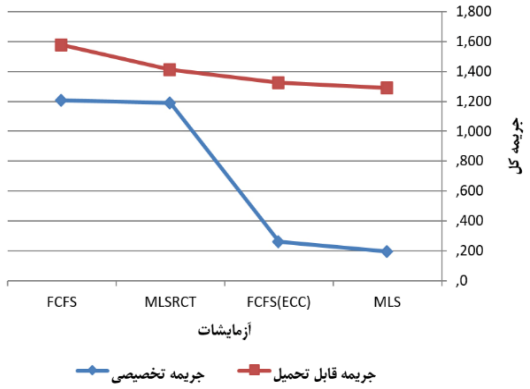


(ج)

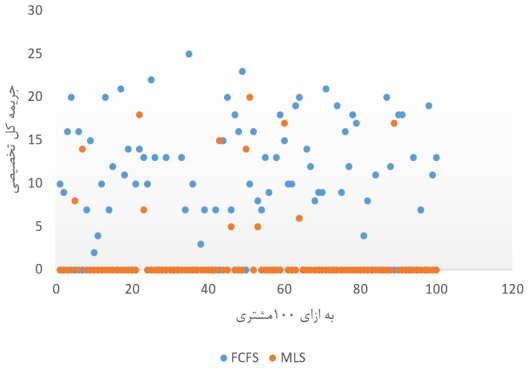
شکل ۱۳: نمودار مقایسه‌ای جرایم تخصیصی در سه متغیر (الف) کارایی، (ب) دسترس‌پذیری و (ج) امنیت.

در شکل ۱۳ محور x نشان‌دهنده متوسط جریمه تخصیصی به‌ازای ۱۰ مشتری می‌باشد. جریمه کارایی به این دلیل رخ می‌دهد که سرویس‌دهنده نمی‌تواند کار سپرده‌شده را در زمان تعیین‌شده به سرانجام برساند. در آزمایش MLSRCT جریمه تخصیصی در متغیر کارایی به این دلیل رخ می‌دهد که سیستم فقط زمان انجام عملیات نقل و انتقال و انجام سرویس را در نظر گرفته و زمانی جهت انجام عملیات رمزگذاری در نظر نگرفته است. در نتیجه پس از انجام عملیات رمزگذاری، زمان در نظر گرفته شده جهت انجام خدمت ممکن است به اتمام برسد و این امر سبب تخصیص جریمه در متغیر کارایی گردد. در مقابل در آزمایش مربوط به MLS، سیستم زمان لازم جهت انجام عملیات رمزگذاری را به زمان انجام سرویس اضافه می‌نماید و باعث می‌گردد کلیه عملیات رمزگذاری، نقل و انتقال و انجام سرویس در زمان تعیین‌شده انجام پذیرد و بازدهی متغیر کارایی، افزایش و میزان جریمه تخصیصی به متغیر کارایی، کاهش یابد.

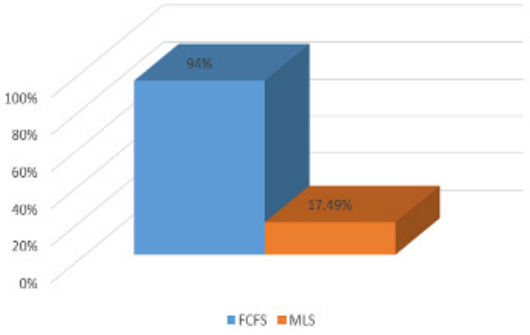
در متغیر امنیت، MLS دارای جریمه تخصیصی کمتری نسبت به FCFS (ECC) است؛ ضمن آن که روش پیشنهادی ما بازدهی متغیرهای کارایی و دسترس‌پذیری را اصلاح می‌کند.



شکل ۱۴: مقایسه جرایم قابل تحمیل و تخصیصی در رویکردهای مختلف.



(الف)

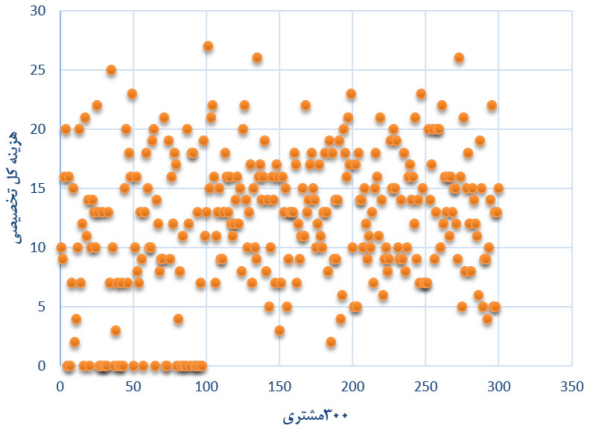


(ب)

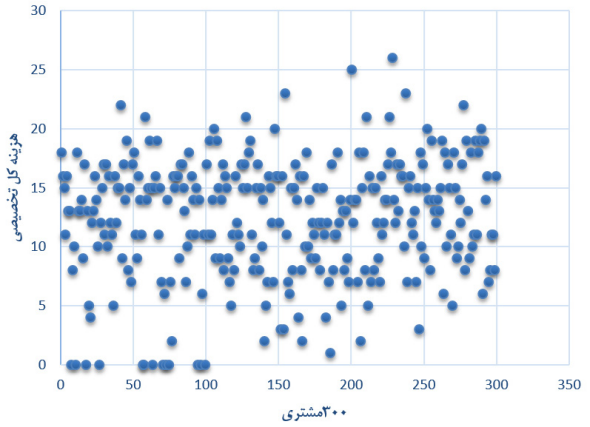
شکل ۱۵: (الف) مقایسه هزینه جریمه در MLS و FCFS و (ب) نمودار ستونی درصد هزینه جریمه در MLS و FCFS.

شکل ۱۴ مقایسه کلی بین حداکثر آستانه جریمه قابل تحمیل و میزان جریمه تخصیصی را نشان می‌دهد. رنگ قرمز نشان‌دهنده میزان جریمه قابل تحمیل در هر آزمایش و رنگ آبی نیز میزان جریمه تخصیصی را نشان می‌دهد. در این وضعیت MLSRCT و FCFS به‌ترتیب کمترین میزان اختلاف و رویکرد پیشنهادی MLS بیشترین میزان اختلاف را دارد؛ بنابراین MLS می‌تواند میزان جرائم را تا ۶۱/۴۱٪ کاهش دهد.

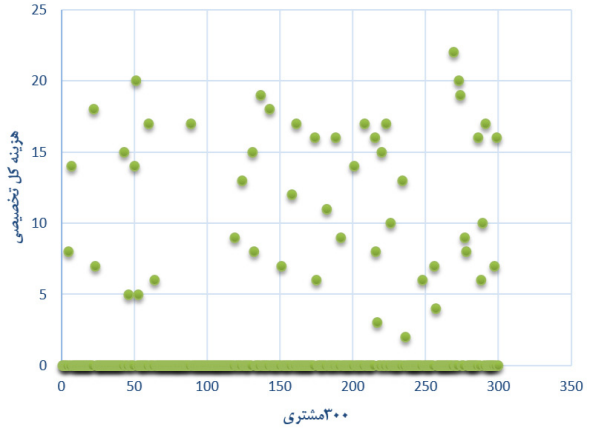
شکل ۱۵- الف مقایسه بین هزینه جریمه آزمایش اول (FCFS) و سوم (MLS) را نشان می‌دهد. هر نقطه نشان‌دهنده ۱۰۰ مشتری است که پس از انجام سرویس، سیستم شامل جریمه تخصیصی شده است. طبق شکل، میزان جریمه در آزمایش سوم (MLS) به طرز چشم‌گیری کاهش دارد. بخش دوم شکل ۱۵- ب، نمودار میله‌ای مقایسه بین دو آزمایش و میزان کاهش جریمه تخصیصی را نشان می‌دهد. مطابق با شکل ۱۵-ب، درصد جریمه تخصیصی در آزمایش اول (FCFS) به اندازه ۹۴٪ است؛ این در حالی است که درصد جریمه تخصیصی در آزمایش سوم (MLS) ۱۷/۴۹٪ است؛ بنابراین رویکرد پیشنهادی در برابر حملات و پیشگیری از حمله مؤثر است.



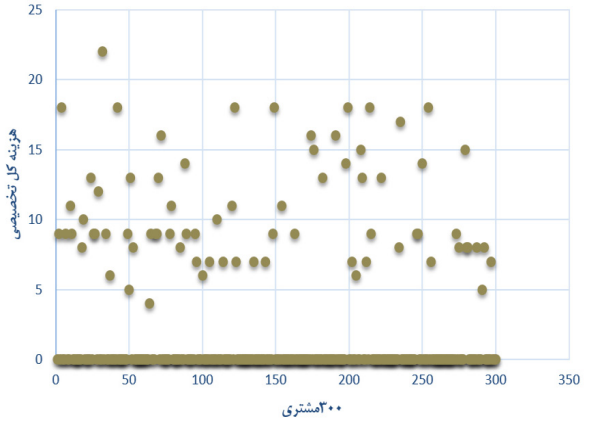
(الف)



(ب)



(ج)



(د)

شکل ۱۶: نمودار تراکم جریمه تخصیصی در (الف) FCFS، (ب) MLSRCT، (ج) MLS و (د) FCFS (ECC).

پس از تکرار آزمایش‌ها به تعدادی که قبلاً ذکر شد به‌منظور محاسبه میزان جریمه تخصیصی در هر یک از متغیرها و هزینه کل، نتایج به‌دست آمده نشان می‌دهند که هزینه جریمه تخصیصی در متغیر امنیت به میزان ۷۷٪ کاهش یافته است؛ در نتیجه هزینه کل جرائم تخصیصی نیز به میزان ۶۱/۴۱٪ کاهش دارد.

شاخص رضایت مشتری به‌صورت (۱۷) در نظر گرفته شده و به‌صورت (۱۸) بیان می‌شود

$$CS = ۱۰۰\% - \frac{\text{متوسط مجموع جریمه تخصیصی}}{\text{متوسط جریمه قابل تحمیل}} \quad (۱۷)$$

$$TCS = CS_i - CS_j \quad (۱۸)$$

که در این حالت CS_i و CS_j به‌ترتیب نشان‌دهنده شاخص رضایت مشتریان در MLS و FCFS هستند و در نتیجه، شاخص رضایت مشتریان ۶۰/۶۷٪ به‌دست می‌آید.

شکل ۱۶ نشان‌دهنده مقایسه نمودارهای میزان تراکم[1] جریمه‌های تخصیصی در هر یک از آزمایش‌هاست. هرچه میزان نقاط مشخص‌شده بر روی بردار y بالاتر و با تعداد بیشتری باشد، میزان جرائم تخصیصی سنگین‌تر است. با توجه به چهار شکل نشان‌داده‌شده، هزینه جریمه در رویکرد MLS کمترین تراکم و بعداز آن رویکرد FCFS (ECC) است. به‌منظور بهینه‌سازی MLSRCT برای کاهش جریمه تخصیصی، آزمایش MLS جایگزین شد.

با توجه به جداول و نمودارهای نشان‌داده‌شده می‌توان نتیجه‌گیری کرد رویکرد MLS بهینگی و شرایط مناسب‌تری نسبت به سایر رویکردها دارد.

۷- نتیجه‌گیری

برای پیاده‌سازی سیستم امنیتی و جلوگیری از نفوذ سارقین و با استفاده از ویژگی‌های الگوریتم‌های آشوب و ویدمن و فازهای تقسیم‌بندی، تکه‌سازی و مبهم‌سازی به تولید کلید خصوصی و رمزنگاری سرویس‌ها در شبکه ابری پرداختیم. استفاده از چهار فاز برای رمزگذاری سرویس‌ها علاوه بر تأمین امنیت روی دیگر متغیرها (امنیت، کارایی و دسترس‌پذیری) تأثیر می‌گذارد که تحلیل آن در این پژوهش بررسی گردید.

استفاده از روش ابتکاری در تولید کلید خصوصی، فازبندی سرویس که موجب کاهش سربار و ارسال سرویس‌ها به‌صورت دسته‌بندی‌شده می‌شود از رویکردهای جدید این پژوهش است. فضای کلید در بخش دوم حداقل $۲^{۱۲۸}$ تعریف شد که مناسب‌بودن آن به‌عنوان کلید استاندارد اثبات گردید. رویکرد جلوگیری از سرقت و دسترسی غیرمجاز به داده‌ها در روش پیشنهادی و همچنین کشف داده توسط CSP توضیح داده شد.

ارائه الگوریتم مبهم‌سازی، نحوه قطعه‌بندی سرویس‌ها، استفاده از سرآیند در تولید کلید خصوصی، کدهای ناهمگام و همچنین توجه به زمان اجرای پردازش و اندازه سرویس از قابلیت‌های این پژوهش است که با سایر رویکردها مقایسه شد. ما با ارائه روش وزن‌گذاری برای محاسبه پارامترها در سه متغیر امنیت، کارایی و دسترس‌پذیری و انجام سه روش آزمایشی نشان دادیم میزان جرائم تخصیصی ۶۱/۴۱٪ کاهش و شاخص رضایت مشتری به میزان ۶۰/۶۷٪ افزایش یافته و نتایج را به‌وسیله جداول و نمودارها با رویکردهای FCFS، FCFS (ECC) و MLSRCT مقایسه کردیم. عدم وابستگی روش حاضر به یک نوع الگوریتم رمزگذاری (RSA) از قابلیت‌های این رویکرد است. بدیهی است این رویکرد برای

1. Spot

رمزگذاری داده‌های برخط[1] مناسب نیست.

در آخر، مدیریت حافظه در زمان رمزگذاری و بررسی تأثیر روش پیشنهادی بر روی انواع حملات رایج، میزان مقاومت روش پیشنهادی و سناریوهای مرتبط با آن از چالش‌های مطرح است که در مباحث پژوهشی آتی در نظر گرفته خواهند شد.

مراجع

[1] P. Zavarasky, R. S. Bali, and F. Jaafar, "Lightweight authentication for MQTT to improve the security of iot communication," in *Proc. of the 3rd Int. Conf. on Cryptography, Security and Privacy*, *ICCSP'19,* pp. 6-12, Kuala Lumpur Malaysia, 9-21 Jan. 2019.

[2] M. V. Pawar and J. Anuradha, "Network security and types of attacks in network," *Procedia Computer Science*, vol. 48, pp. 503-506, 2015.

[3] H. Singh, Z. Amin, and N. Sethi, "Review on fault tolerance techniques in cloud computing," *International J. of Computer Applications*, vol. 116, no. 18, pp. 11-17, 2015.

[4] S. Venugopala, J. Broberg, I. Brandic, R. Buyya, and C. S. Yeoa, "Cloud computing and emerging it platforms: vision, hype, and reality for delivering computing as the 5th utility," *Future Generation Computer Systems*, vol. 25, no. 6, pp. 599-616, Jun. 2008.

[5] H. Ghazanfaripour and A. Broumandnia, "Designing a digital image encryption scheme using chaotic maps with prime modular," *Optics and Laser Technology*, vol. 131, Article ID: 106339, 2020.

[6] A. Coen-Porisini, A. Rizzardi, and S. Sicari, "Analysis on functionalities and security features of internet of things related protocols," *Wireless Networks*, vol. 28, no. 7, pp. 2857-2887, 2022.

[7] C. Liao, *et al.*, "MODECP: a multi-objective based approach for solving distributed controller placement problem in software defined network," *Sensors*, vol. 22, no. 15, Article ID: 22155475, 2022.

[8] L. T. Yang, G. Huang, J. Feng, and L. Xu, "Parallel GNFS algorithm integrated with parallel block wiedemann algorithm for RSA security in cloud computing," *Information Sciences*, vol. 387, pp. 254-265, May 2017.

[9] A. Broumandnia, "Image encryption algorithm based on the finite fields in chaotic maps," *J. of Information Security and Applications*, vol. 54, Article ID: 102553, Oct. 2020.

[10] E. B. Sanjuan, I. A. Cardiel, J. A. Cerrada, and C. Cerrada, "Message queuing telemetry transport (MQTT) security: a cryptographic smart card approach," *IEEE Access*, vol. 8, pp. 115051-115062, 2020.

[11] Z. Shen and Q. Tong, "The security of cloud computing system enabled by trusted computing technology," in *Proc. 2nd Int. Conf. on Signal Processing Systems*, vol. 2, pp. 11-15, Dalian, China, 5-7 Jul. 2010.

[12] M. V. Kumar, G. Manogaran, and C. Thota, "Metaclouddatastorage architecture for big data security in cloud computing," *Procedia Computer Science*, vol. 87, pp. 128-133, 2016.

[13] Q. Zheng, "Improving MapReduce fault tolerance in the cloud," in *Proc. IEEE Int. Symp. on Parallel Distributed Processing, Workshops and PhD Forum, IPDPSW'10*, 6 pp., Atlanta, GA, USA, 19-23 Apr. 2010.

[14] S. K. Sharma, P. Gautam, and M. D. Ansari, "Enhanced security for electronic health care information using obfuscation and RSA algorithm in cloud computing," *International J. of Information Security and Privacy*, vol. 13, no. 1, pp. 59-69, Jan./Mar. 2019.

[15] M. Ratha, "Resource provision and QoS support with added security for client side applications in cloud computing," *International J. of Information Technology*, vol. 11, pp. 357–364, 2019.

[16] S. Khatri, Y. Sharma, and H. Gupta, "A security model for the enhancement of data privacy in cloud computing," in *Proc. Amity Int. Conf. on Artificial Intelligence, AICAI'19*, pp. 898-902, Dubai, United Arab Emirates, 4-6 Feb. 2019.

[17] I. Banerjee and N. Nesa, "Combining merkle hash tree and chaotic cryptography for secure data fusion in IoT," *Trans. on Computational Science,* vol. 35, pp. 85-105, 2020.

[18] M. F. Aboalmaaly, A. J. Hintaw, S. Manickam, and S. Karuppayah, "MQTT vulnerabilities, attack vectors and solutions in the internet of things (IoT)," *IETE J. of Research*, vol. 69, no. 6, pp. 3368-3397, 2023.

[19] G. Raines, *Cloud Computing and SOA*, Technical Report, the MITRE Corporation, 2009.

[20] M. Santambrogio, R. Jhawar, and V. Piur, "A comprehensive conceptual system-level approach to fault tolerance in cloud computing," in *Proc. IEEE Int. Systems Conf., SysCon'12*, 5 pp., Vancouver, Canada, 19-22 Mar. 2012.

[21] K. Sood, "A combined approach to ensure data security in cloud computing," *J. of Network and Computer Applications*, vol. 35, no. 6, pp. 1831-1838, Nov. 2012.

[22] P. Wiedera, "Fault-tolerant service level agreement lifecycle management in clouds using actor system," *Future Generation Computer Systems, Elsevier B.V*, vol. 54, pp. 247-259, Jan. 2016.

[23] H. Liang, D. Huang, L. X. Cai, X. Shen, and D. Peng, "Resource allocation for security services in mobile cloud computing," in *Proc. IEEE Conf. on Computer Communications Workshops, INFOCOM WKSHPS'11*, pp. 191-195, Shanghai, China, 10-15 Apr. 2011.

[24] Y. Protskaya, L. Veltri, F. Zanichelli, M. Amoretti, and R. Pecori, "A scalable and secure publish/subscribebased framework for industrial IoT," *IEEE Trans. on Industrial Informatics*, vol. 17, no. 6, pp. 3815-3825, Jun. 2021.

[25] L. Huang, S. Cai, X. Xiong, and M. Xiao, "On symmetric color image encryption system with permutation-diffusion simultaneous operation," *Optics and Lasers in Engineering*, vol. 115, pp. 7-20, Apr. 2019.

[26] P. Li, *et al.,* "A novel color image encryption scheme using DNA permutation based on the lorenz system," *Multimedia Tools and Applications*, vol. 77, no. 5, pp. 6243-6265, Mar. 2018.

[27] Y. Q. Zhang, X. Y. Wang, J. Liu, and Z. L. Chi, "An image encryption scheme based on the MLNCML system using DNA sequences," *Optics and Lasers in Engineering*, vol. 82, pp. 95-103, Jul. 2016.

[28] N. Chauhan, H. Banka, and R. Agrawal, "Delay-aware application offloading in fog environment using multi-class Brownian model," *Wireless Networks*, vol. 27, no. 7, pp. 4479-4495, 2021.

[29] C. A. F. De Rose, R. Buyya, R. N. Calheiros, and M. A. S. Netto, "EMUSIM: an integrated emulation and simulation environment for modeling, evaluation, and validation of performance of cloud computing applications," *Software: Practice and Exprience*, vol. 43, no. 2, pp. 595-612, 2013.

[30] A. Beloglazov, C. A. F. De Rose, R. Buyya, R. N. Calheiros, and R. Ranjan, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software-Practice and Experience*, vol. 41, no. 1, pp. 23-50, Jan. 2010.

[31] T. Halabi and M. Bellaiche, "A broker-based framework for standardization and management of cloud security-SLAs," *Computers and Security*, vol. 75, pp. 59-71, Jun. 2018.

آزیتا رضایی در سال ۱۳۸۵ مدرک کارشناسی مهندسی کامپیوتر- نرم افزار خود را از دانشگاه پیام نور مرکز تهران و در سال ۱۳۹۲ مدرک کارشناسی ارشد مهندسی کامپیوتر- نرم افزار خود را از دانشگاه آزاد واحد قزوین دریافت نمود. هم‌اکنون بعنوان دانشجوی دکتری دانشگاه آزاد واحد تهران جنوب در رشته مهندسی کامپیوتر- سیستم های نرم افزاری مشغول به تحصیل می باشد. از سال ۱۳۸۰ تاکنون به‌عنوان کارمند وزارت تعاون، کارو رفاه اجتماعی مشغول به کار می‌باشد. وی در سال‌های ۱۳۹۳ الی ۱۳۹۵ در دانشگاه آزاد واحد سماء اسلامشهر و دانشگاه آزاد شهرقدس به صورت پاره وقت تدریس نموده است. زمینه‌های علمی مورد علاقه نام‌برده متنوع بوده و شامل موضوعاتی مانند نحوه ارسال داده در شبکه‌های ابری، برقراری امنیت در شبکه‌های ابری و MQTT ، ارائه الگوریتم جهت افزایش امنیت داده‌ها در زمان ارسال در شبکه می‌باشد.

علی برومندنیا در سال ۱۳۷۱ مدرک کارشناسی مهندسی کامپیوتر خود را از دانشگاه صنعتی اصفهان و در سال ۱۳۷۴ مدرک کارشناسی ارشد مهندسی کامپیوتر خود را از دانشگاه علم و صنعت ایران دریافت نمود. از سال ۱۳۷۴ الی ۱۳۷۸ نام‌برده به عنوان عضو هیأت علمی مربی کامپیوتر در دانشگاه آزاد واحد تهران سما مشغول به کار بود و پس از آن دوره دکترای مهندسی کامپیوتر را در دانشگاه آزاد اسلامی واحد علوم تحقیقات به‌عنوان بورسیه شروع نمود و در سال ۱۳۸۵ موفق به اخذ درجه دکترا در مهندسی کامپیوتر از دانشگاه مذکور گردید. دکتر برومندنیا از سال ۱۳۷۸ در دانشکده فنی کامپیوتر دانشگاه آزاد اسلامی واحد تهران جنوب شروع به فعالیت نمود و اینک نیز عضو هیأت علمی این دانشکده می‌باشد. زمینه‌های علمی مورد علاقه نام‌برده متنوع بوده و شامل موضوعاتی مانند ایده‌های نو در هوش مصنوعی، پنهان‌سازی اطلاعات، امنیت سیستم‌های چند رسانه‌ای می‌باشد.

1. Real-Time

سید جواد میرعابدینی در سال ۱۳۷۰ مدرک دیپلم خود را در رشته ریاضی و فیزیک از دبیرستان باهنر دریافت نمود. در سال ۱۳۷۵ مدرک کارشناسی مهندسی کامپیوتر خود را از دانشگاه شهید بهشتی دریافت نمود. ایشان دوره‌های DOS پیشرفته و DBMS، فاکس پرو، پارادوکس، MS Access و برنامه‌نویسی C، پاسکال، C++، C# و اسمبیلی را در دانشکده‌های شهر تهران تدریس نمود. بین سال‌های ۱۳۷۷ الی ۱۳۸۶ در وزارت ارتباطات به‌عنوان سرپرست منابع انسانی و بخش IT مشغول به کار بود. هم‌زمان در سال ۱۳۸۰ مدرک کارشناسی ارشد رشته مهندسی کامپیوتر- هوش مصنوعی و رباتیک خود را از دانشگاه آزاد واحد علوم و تحقیقات دریافت نمود. دکتر میرعابدینی مدرک دکترای مهندسی کامپیوتر- سیستم‌های نرم افزاری را در سال ۱۳۸۱ شروع نمود و در سال ۱۳۸۷ موفق به اخذ درجه دکترا گردید. ایشان هم‌اکنون به‌عنوان عضو هیأت علمی دانشگاه آزاد واحد تهران مرکز مشغول به کار می‌باشد. زمینه های علمی مورد علاقه نام‌برده متنوع بوده و شامل موضوعاتی مانند سیستم‌های هوشمند، هوش تجاری، سیستم‌های توسعه‌گر، و یادگیری ماشین کاربردی می‌باشد.