مجله علمی-پژوهشی «علوم و فناوری‌های پدافند غیرعامل»
سال دوم، شماره ۳، پاییز ۱۳۹۰: ص ۲۳۰ - ۲۱۷

ممیزی تمامیت در نتایج پرس و جوهای پیوسته

مجید غیوری ثالث[1*]، مصطفی حق‌جو سانیجی[2]، خسرو سلمانی[3]

۱- دانشجوی دکترا، ۲- استادیار، ۳- کارشناس ارشد، دانشکده کامپیوتر، دانشگاه علم و صنعت ایران

(دریافت: ۱۳۹۰/۰۲/۱۰، پذیرش: ۱۳۹۰/۱۰/۰٤)

چکیده

علی‌رغم آنکه استفاده از خدمات سرورهای برون‌سپاری شده جریان داده مورد اقبال زیادی قرار گرفته ولی هنوز مساله کسب اطمینان از جامعیت نتایج دریافتی از این سرورها یکی از چالش‌های اساسی سازمان‌ها است. برای برون‌سپاری این خدمات، کاربر باید از عملکرد درست و امانت‌دارانه سرور و کانال ارتباطی مطمئن شود، چرا که سرور ممکن است به دلایل اقتصادی و یا بدخواهانه جامعیت نتایج را مورد حمله قرار دهد. در این حملات بخشی از پاسخ‌ها برای کاربر ارسال نشده و یا پس از دست‌کاری یا با تاخیر برای آنها ارسال می‌شود. در این مقاله یک روش کارا برای کشف حملات جامعیت در سیستم‌های جریان داده برون‌سپاری شده بر مبنای ممیزی محاسبات تقاطعی ارائه می‌شود. در این روش جریان داده اصلی با یک کلید رمزنگاری شده و بخش کوچکی از داده‌ها به صورت یک جریان داده مستقل با کلید دیگری رمزنگاری شده و برای سرور ارسال می‌شوند. پرس و جوی درخواستی کاربر برروی دو جریان داده اجرا می‌شود و کاربر با مقایسه نتایج در رابطه با جامعیت نتایج قضاوت می‌کند. این روش سربار بسیار کمی بر کاربر تحمیل نموده و برای اجرای آن نیازی به تغییر ساختار سرور نیست. مدل‌سازی احتمالاتی روش نشان می‌دهد که این روش کارآیی بسیار بالایی دارد و نتایج ارزیابی عملی این نتیجه را به خوبی تایید می‌کنند.

کلیدواژه‌ها: امنیت در جریان داده، جامعیت در جریان داده، برون‌سپاری سیستم مدیریت جریان داده، محاسبات تقاطعی.

Completeness Auditing of Continuous Query Results

M. Ghayoori Sales[1*], M. Haghjoo[2], K. Salmani[3]

Faculty of Computer, Iran University of Science and Technology

(Received: 04/30/2011 , Accepted: 12/25/2011)

Abstract:

Despite the fact that using the services of outsourced data stream servers has extremely been welcomed, but still the problem of obtaining certainty about received results from these servers is one of the basic challenges in enterprises. For outsourcing these services, the user should be assured by a mechanism about the security of communication channels as well as the correct and honest function of the server, because the server may attack the integrity of the results due to economic and malicious reasons. In such attacks, some parts of results are not sent to the user or sent after being modified or delayed. In this article, we have come up with an efficient method for detecting integrity attacks in outsourced data steam systems based on auditing the results of cross computation. In this method, the main data stream has been enciphered by a key and a small part of data has been enciphered by a different key, as a dependant data stream, and sent to the server. The requested query is applied on both streams and the user judges the integrity of results by comparing the results. Our method imposes a little overhead on the user and needs no change in the structure of the server. The probabilistic modeling of the method shows that this method has a high efficiency and the results of the exprimental analysis confirm this very well.

Keywords: Service Oriented Architecture, Service Elicitation, Identify Service, Candidate Service, Tree Organizational Goals, Compliance Services

[*] Corresponding author E-mail ghayoor@iust.ac.ir

۱. مقدمه

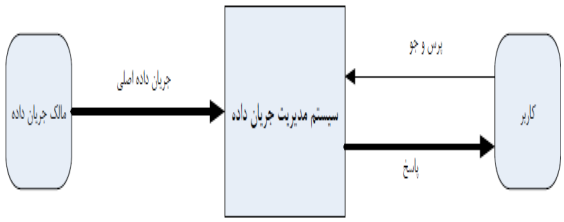
افزایش حجم داده‌های تولیدی در بسیاری از سیستم‌ها سبب توجه هرچه بیشتر سازمان‌ها به استفاده از سیستم‌های مدیریت جریان داده شده است[۱]. از طرف دیگر پیاده‌سازی و نگهداری سیستم‌های مدیریت جریان داده نیازمند صرف هزینه‌های زیادی می‌باشد، به‌طوریکه بسیاری از سازمان‌ها و ادارات توانایی راه‌اندازی و پرداخت هزینه‌های مربوط به آن را ندارند. به همین دلیل، این سازمان‌ها علاقه‌مندند سرویس پردازش جریان داده را از یک ارائه کننده سرویس خارج از سازمان خود دریافت کنند. ایده استفاده از سرویس‌های پردازشی برون‌سپاری شده قبلا در سیستم‌های مدیریت پایگاه داده[۱] (DBMS) تحت عنوان پایگاه داده به عنوان سرویس[۲] (DAS) مورد استفاده قرار گرفته است[۲].

مهمترین چالشی که در برون‌سپاری پردازش داده‌ها وجود دارد اطمینان از عملکرد صادقانه سرور است. سرور ممکن است برای به دست آوردن سود بیشتر قسمتی از پاسخ را برای کاربر ارسال نکند و یا همان پاسخ‌های قدیمی را برای کاربر دوباره ارسال کند هم‌چنین سرور ممکن است به دلایل بدخواهانه حتی با صرف هزینه بیشتر پاسخ‌های نادرست را برای کاربر ارسال کند[۳].

برای اطمینان از حفظ جامعیت داده‌ها می‌باید به بررسی صحت، تازگی و تمامیت داده‌ها پرداخته شود. صحت رکوردهای پاسخ به معنی حفظ محتویات داده‌ها بدون هیچ گونه تغییر در آن است. برای حفظ تازگی می‌باید اطمینان حاصل شود که پاسخ‌های دریافت شده بر اساس آخرین رکوردهای جریان داده، و نه بر اساس رکوردهای قبلی تهیه شده باشد. تمامیت رکوردهای پاسخ به معنی آن است که سرور در پاسخ به پرس‌وجوهای دریافت شده می‌باید تمام رکوردهای پاسخ و نه فقط بخشی از آن را به کاربر ارسال کند.

شکل (۱) معماری یک سیستم مدیریت جریان داده برون‌سپاری شده را نمایش می‌دهد. مالک جریان داده، داده‌های خودرا از طریق یک کانال ارتباطی به صورت رمز شده برای سرور ارسال می‌نماید. کاربر نیز پس از رمز نمودن پرس‌وجوهای خود، آن‌ها را برای سرور ارسال می‌کند. سرور در نتیجه اجرای پرس و جوهای درخواستی، پاسخ را در قالب جریان داده‌ای از رکوردها برای کاربر ارسال می‌کند. همان‌طور که در شکل نشان داده شده است در این معماری، با وجود آنکه، کاربر به جریان داده اصلی دسترسی ندارد می‌باید با استفاده از مکانیزم‌هایی از جامعیت (صحت،تازگی،تمامیت) پاسخ‌ها اطمینان حاصل کند[۴].

هم‌اکنون برای ممیزی صحت معمولا از امضای رکوردها و برای ممیزی تازگی از روش مهر زمانی استفاده می‌شود، اما ممیزی تمامیت به این سادگی نیست.



شکل ۱. جریان داده برون‌سپاری شده

در هنگام اعمال مکانیزم‌های تمامیت باید توجه داشت که سیستم‌های مدیریت جریان داده در مواقعی که با افزایش بار ورودی روبه‌رو می‌شوند بخشی از رکوردهای ورودی را به منظور کاهش بار حذف می‌کنند[۵ و۶]. این مساله چالشی‌ترین مساله موجود در کنترل تمامیت سیستم‌های مدیریت جریان داده برون‌سپاری شده است.

به طور کلی روش‌های کنترل جامعیت به دو دسته روش‌های مبتنی بر ساختار تصدیقی و روش‌های احتمالاتی تقسیم می‌شوند[۷]. در روش‌های مبتنی بر ساختارهای تصدیقی، مالک جریان داده، رکوردهای جریان داده را در یک ساختار تصدیقی (مثلا یک زنجیره یا یک درخت) سازمان‌دهی نموده و برای سرور ارسال می‌کند. در ضمن مالک جریان داده ساختار تصدیقی را امضاء کرده و مقدار امضاء را از طریق یک کانال امن برای کاربران ارسال می‌کند. کاربران از چگونگی تولید امضاء بر مبنای ساختار تصدیقی رکوردها مطلع هستند. سرور در پاسخ پرس و جوهای کاربر، علاوه بر داده‌های نتیجه، بر مبنای ساختار تصدیقی، اطلاعات تکمیلی برای تولید مجدد امضاء را برای کاربر ارسال می‌کند. کاربر با استفاده از رکوردهای دریافتی و اطلاعات تکمیلی، مجددا امضاء را تولید نموده و با امضای دریافتی از مالک جریان داده مقایسه می‌کند.

به این ترتیب کاربران می‌توانند از تمامیت نتایج دریافتی مطمئن شوند. روش‌های مبتنی بر ساختارهای تصدیقی امکان کنترل دقیق نتایج دریافتی برای کاربران را فراهم می‌کنند، ولی برای پیاده‌سازی آنها می‌باید تغییراتی را در ساختار سرور اعمال کرد. در معماری سیستم‌های جریان داده برون‌سپاری شده به دلیل آن که یک سیستم توسط چند کاربر مورد استفاده قرار می‌گیرد، نمی‌توان ساختار سرور را برای یک کاربر خاص تغییر داد. بنابراین، این روش‌ها برای سیستم‌های جریان داده برون‌سپاری شده مناسب نمی‌باشند. از طرف دیگر این روش‌ها کاهش بار ورودی توسط سرور را به عنوان یک حمله تشخیص می‌دهند.

در روش‌های احتمالاتی، کاربران بخش کوچکی از پردازش‌های سرور را انجام داده و با مقایسه نتایج به‌دست آمده با نتایج دریافتی، تمامیت پاسخ‌ها را ارزیابی می‌کنند. مهم‌ترین اشکال روش‌های احتمالاتی، عدم کسب اطمینان کامل از تمامیت پاسخ‌ها است. این

[1] Database Management System
[2] Database as a Service

روش‌ها طیف بیشتری از شروط و پرس‌وجوها را تحت پوشش قرار می‌دهند. اما مهم‌ترین مزیت آنها عدم نیاز به تغییر در سیستم‌های مدیریت جریان داده موجود برای بهره‌برداری از آنها و عدم شکنندگی در وضعیت تحمل کاهش بار توسط سرور است. به همین دلیل از این روش‌ها به خوبی می‌توان در برون‌سپاری خدمات DSMS [1] استفاده نمود.

در این تحقیق، یک راه حل مناسب برای ممیزی احتمالاتی جامعیت سیستم‌های جریان داده برون‌سپاری شده ارائه شده است. مهم‌ترین خصوصیت این روش استقلال کامل آن از سرور و نحوه اجرای پرس و جوها توسط آن و نیز عدم نیاز کاربر به دسترسی به جریان داده اصلی است. روش در ساده‌ترین حالت مشابه روش‌های بازرسی تقاطعی می‌باشد. در این روش‌ها مجموعه‌ای از رکوردهای جریان داده برای دو سرور A و B ارسال می‌گردند.

این سرورها هیچ‌گونه ارتباط داخلی با یکدیگر ندارند. پرس‌وجوهای کاربر برای هر دو سرور ارسال می‌گردد. بدیهی است یکسان نبودن پاسخ‌ها نشانگر آن است که حداقل یکی از سرورها به درستی کار نمی‌کند. مهمترین تفاوت این روش با روش‌های مبتنی بر ارزیابی تقاطعی، این است که در این روش تنها از یک سرور استفاده می‌شود که این موضوع باعث کاهش هزینه‌های سیستم می‌گردد.

در این روش مالک جریان داده بخش بسیار کوچکی از رکوردهای جریان داده را انتخاب کرده و پس از رمزنگاری با یک کلید مجزا، به صورت یک جریان داده مستقل به سرور ارسال می‌کند. سرور پرس و جوی کاربر را روی جریان داده اصلی و تکراری اجرا نموده و پاسخ‌ها را برای کاربر ارسال می‌کند. برای ممیزی تمامیت پاسخ‌ها، کاربر مجموعه پاسخ‌های دریافتی از دو جریان داده را به‌صورت تقاطعی با یکدیگر مقایسه نموده و در خصوص تمامیت آنها قضاوت می‌کند. این روش علاوه بر سربار بسیار کم برای سرور و کاربر، نتایج بسیار قابل قبولی دارد که در بخش‌های مدلسازی احتمالاتی و ارزیابی به آن پرداخته می‌شود.

از اصلی‌ترین چالش‌های این روش می‌توان به مسئله نحوه ایجاد هم‌زمانی بین مالک جریان داده و کاربر، چگونگی انتخاب رکوردهای تکرار و چگونگی مقایسه رکوردهای اصلی با رکوردهای تکرار با بیشترین دقت و کمترین سربار ممکن اشاره کرد.

مهم‌ترین نوآوری‌های این تحقیق عبارتند از:

- ارائه یک معماری مبتنی بر محاسبات تقاطعی برای ممیزی جامعیت در سیستم‌های مدیریت جریان داده برون‌سپاری شده
- مدل‌سازی روش ارائه شده بر مبنای مدل‌سازی احتمالاتی و تطبیق مناسب نتایج مدل‌سازی با نتایج ارزیابی عملی
- پیاده‌سازی کامل روش و دست‌یابی به نتایج بسیار مناسب و قابل قبول

[1] Data Stream Management System

۲. کارهای مرتبط

به طور کلی برای حفظ جامعیت در سیستم‌های مدیریت جریان داده می‌باید صحت، تمامیت و تازگی داده‌ها مورد بررسی قرارگیرد. صحت رکوردهای پاسخ به معنی تعیین اصالت رکوردهای دریافتی از سرور توسط کاربر است. برای اطمینان از تازگی، کاربر باید مطمئن شود که نتایج دریافتی بر اساس اعمال شرایط پرس و جو به آخرین رکوردهای جریان داده و نه از رکوردهای قبلی به‌دست آمده‌اند. تمامیت پاسخ‌ها به این معناست که مجموعه پاسخ بدون هیچ کم و کاستی به‌طور کامل توسط کاربر دریافت شده و رکوردی توسط سرور و یا دشمن از این مجموعه حذف نشده است[۸]. از یک دیدگاه روش‌های کنترل جامعیت در سیستم‌های مدیریت جریان داده را می‌توان به دو دسته مبتنی بر ساختارهای تصدیقی و احتمالاتی دسته بندی نمود.

از مهم‌ترین روش‌های مبتنی بر ساختار تصدیقی می‌توان به روش‌هایی اشاره نمود که از درخت درهم‌سازی مرکل به عنوان ساختار تصدیقی استفاده می‌کنند[۹]. در این روش‌ها جریان داده بر اساس یک ساختار تصدیقی (بر پایه B+-Tree) سازماندهی شده بر اساس درخت مرکل توسط مالک جریان داده امضاء شده و برای سرور ارسال می‌شوند. سرور پس از اجرای پرس‌وجوی مورد نظر، نتایج را به همراه اطلاعاتی که توسط آنها کاربر امکان بازسازی امضاء ریشه داشته باشد در یک شیی تصدیقی[2] قرار داده و برای کاربر ارسال می‌کند. کاربر با استفاده از اطلاعات دریافتی ریشه درخت را مجددا ساخته و امضاء آن‌را بررسی می‌کند[۳ و۴]. روش‌های مبتنی بر ساختار تصدیقی ضمن دارا بودن دقت زیاد، سربار نسبتا زیادی را به سیستم تحمیل می‌کنند و در مقابل کاهش بار ورودی، که یکی از ملزومات اساسی DSMSها هستند، شکننده می‌باشند.

در روش‌های احتمالاتی بخشی از پردازش سرور، دوباره توسط کاربر انجام می‌شود و نتایج به‌دست آمده با یکدیگر مقایسه می‌گردد. از کارهای انجام شده در این زمینه می‌توان به روش قبلی خودمان که در سال ۲۰۱۱ ارائه شده است، اشاره نمود[۱۰]. در این روش برای اطمینان از جامعیت داده‌ها، مالک جریان داده اقدام به تولید رکوردهای ساختگی می‌نماید و سپس آنها را بر اساس یک الگوی مشخص در بین رکوردهای اصلی برای سرور ارسال می‌کند. کاربر که در ابتدا از طریق یک کانال امن از الگوی تولید رکوردهای ساختگی مطلع شده است به تولید مجدد رکوردهای ساختگی و اعمال پرس وجو بر روی آنها می‌پردازد و در انتها رکوردهای دریافتی را با رکوردهای مورد انتظار مقایسه می‌کند. مهم‌ترین ضعف این روش سربار کاربر در تولید و پرس و جوی جریان داده ساختگی است که در این مقاله یک راه حل مناسب برای رفع آن ارائه شده است.

[2] Verification Object - VO

روش دیگری در سال ۲۰۰۸ تحت عنوان خلاصه‌های تصادفی چند جمله‌ای‌های یکه[1] (PIRS) ارائه شد[۱۱]. این روش برای تحلیل داده‌های ترافیک شبکه توسط سرورهای DSMS مورد استفاده قرار گرفت. در مدل PIRS، الگوریتمی برای تولید یک خلاصه احتمالاتی بر مبنای مدلسازی جریان داده ارائه شده است. این الگوریتم با محاسبه این خلاصه و مقایسه آن با مقدار مورد انتظار، در رابطه با صحت نتایج دریافتی از سرور تصمیم‌گیری می‌کند. این الگوریتم پرس‌وجوهای شمارشی و تجمیعی را پشتیبانی کرده و برای محاسبه خلاصه مورد انتظار از سربار محاسباتی و حافظه قابل قبولی برخوردار است.

۳. بیان مسئله و فرض‌های اولیه

در یک سیستم مدیریت جریان داده، کاربر می‌باید با استفاده از مکانیزم‌هایی از جامعیت پاسخ‌های دریافتی از سرور اطمینان حاصل کند برای تایید جامعیت داده‌ها کاربر باید از موارد زیر مطمئن شود:

- تمامیت: از مجموعه پاسخ مورد انتظار هیچ رکوردی حذف نشده باشد.
- صحت: همه رکوردهای دریافتی از سرور از منبع جریان داده دریافت شده و رکورد جعلی در آنها وجود نداشته باشد.
- تازگی: پاسخ‌های دریافتی از سرور بر اساس آخرین رکوردهای دریافتی از جریان‌های داده ورودی تهیه شده باشند.

در این تحقیق به ارائه یک روش کارا برای کنترل تمامیت در یک محیط متشکل از منبع جریان داده، سرور و کاربر پرداخته می‌شود. در این روش نیازی به هیچ‌گونه تغییر در ساختار سرور نمی‌باشد.

فرضیات اولیه این تحقیق عبارتند از:

- کانال‌های ارتباطی بین مالک داده، سرور و کاربر ناامن بوده و احتمال وجود حملاتی مانند:
  INSIDER و یا MAN-IN THE-MIDDLE وجود دارد.
- سرور امن و مورد اعتماد نمی‌باشد. مشکلات مربوط به سرور به دو صورت می‌باشد:
  - سرور تنبل و یا کم کار است بنابراین پاسخ‌ها ناکامل یا با تاخیر ارسال می‌شوند.
  - سرور بدخواه است. در چنین مواردی سرور بخشی از اطلاعات را به عمد حذف کرده و یا تغییر می‌دهد.
- کاربر ارتباط مستقیمی با مالک جریان داده ندارد.
- کاربر و مالک جریان داده امکان تبادل اطلاعات مختصر از روی کانال امن را در ابتدای برقراری ارتباط دارند.
- توان پردازشی کاربران محدود می باشد. کاربر ممکن است با استفاده از یک دستگاه سیار مانند تلفن همراه به سیستم متصل گردد. از این رو می باید از تحمیل هر گونه سربار اضافی تا حد امکان خودداری نمود زیرا چنین وسیله هایی دارای پهنای باند محدود و محدودیت در نیروی باتری می‌باشند.
- رکوردها بین مالک جریان داده، سرور و کاربران به‌صورت رمز شده رد و بدل می‌شوند.
- یک مالک جریان داده، تعدادی متناهی جریان داده تولید می‌کند. جریان داده p دریافتی از مالک جریان داده k، دنباله‌ای نامتناهی از رکوردهای اندیس‌دار به صورت زیر است:

$$S_p^k = \{r_0, r_1, \dots\}$$

- کاربر امکان ثبت پرس‌وجوی Q روی جریان داده p از مالک جریان داده k (Q_p^k) به صورت زیر را دارد.

Q_p^k: SELECT *
FROM S_p^k
WHERE $Condition(A_1, A_2, \dots, A_m)$
WINDOW SIZE n SLIDE EVERY t (t>=n)

در این پرس‌وجوها Condition (A_1, A_2, A_3, … ,A_n)، هر ترکیب شرطی روی خصیصه‌های S_p^k است که با عملگرهای منطقی با هم ترکیب شده‌اند.

- حاصل اجرای پرس‌وجوی Q_p^k روی S_p^k دنباله نامتناهی از رکوردها به‌صورت زیر است:

$$R_p^k = Apply(Q_p^k, S_p^k) = \{r_i | r_i \in S_p^k \wedge r_i \; satisfies \; Q_p^k\}$$

توجه: در این مقاله فقط برای ساده‌سازی نگارش فرض شده است که تنها یک مالک جریان داده وجود دارد، بنابراین از نوشتن اندیس k صرف‌نظر شده است.

بر اساس فرض‌های فوق، مساله اصلی در این تحقیق عبارتست از ارائه روشی جدید جهت ممیزی جامعیت پاسخ‌های دریافتی از یک سرور مدیریت جریان داده نامطمئن، بدون نیاز به تغییر در سرور.

۴. روش ارائه شده

همان‌طور که در مقدمه اشاره شد در روش پیشنهادی مالک جریان داده تعدادی از رکوردهای جریان داده را انتخاب نموده و به‌صورت یک جریان داده مستقل برای سرور ارسال می‌کند. مالک جریان داده، از دو کلید برای رمزنگاری داده‌ها استفاده می‌کند. کلید اول برای رمزکردن جریان داده اصلی و از کلید دوم برای رمزنگاری جریان داده دوم استفاده می‌کند. نرخ انتخاب رکوردهای جریان داده تکراری با توجه به سطح دقت مورد انتظار در تشخیص حملات انتخاب می‌شود. با توجه به آنکه همه رکوردها به صورت رمز شده تبادل می‌گردند، سرور امکان تشخیص رکوردهای تکراری را ندارد. پس از اعمال پرس‌جوهای ثبت شدهٔ کاربر بر روی هر دو جریان داده، رکوردهای

[1] Polynomial Identity Random Synopses

پاسخ برای کاربر ارسال می‌گردند. کاربر نیز پس از رمزگشایی رکوردهای پاسخ، با توجه به سرآیندی که مالک جریان داده بر روی رکوردها قرار داده است می‌تواند رکوردهای تکرار را تشخیص دهد. بنابراین کاربر پس از تشخیص رکوردهای تکرار و مقایسه آنها با رکوردهای پاسخ از جریان داده دوم می‌تواند در رابطه با جامعیت پاسخ دریافتی از سرور قضاوت می‌کنند.

۴-۱. معماری سیستم

در شکل (۲) معماری سیستم مورد نظر نمایش داده شده است. برای دست‌یابی به محرمانگی داده‌ها، تمامی رکوردها قبل از ارسال به سرور، رمزنگاری می‌شوند. بنابراین سرور پرس‌وجوها را روی داده‌های رمز شده اجرا می‌کند. تاکنون روش‌های مختلفی برای اجرای پرس و جوها روی داده‌های رمز شده ارائه شده‌اند[۱۴- ۱۲]. بدیهی است شروط قابل اعمال در پرس و جوها وابستگی کامل به روش رمزنگاری انتخاب شده دارند و از حیطه این تحقیق خارج هستند. سرور از نحوه انتخاب رکوردهای تکرار مطلع نیست هم‌چنین در روش‌های رمزنگاری جدید امکان نگاشت[1] مقادیر به یکدیگر وجود ندارد. بنابراین در روش پیشنهادی، حتی اگر سرور بداند که جریان داده با حجم کمتر نمونه‌هایی از جریان داده اصلی می‌باشند که با یک کلید دیگر رمزنگاری شده‌اند، باز هم امکان نگاشت دو جریان بر روی یکدیگر وجود ندارد. بسیار واضح است که توفیق حمله کنندگان در مقایسه دو جریان داده و یافتن رکوردهای یکسان در دو جریان داده کاملا وابسته به امنیت الگوریتم رمزنگاری انتخاب شده است.

مطابق شکل (۲)، یک بخش در قسمت مالک جریان داده[2] (DSO) برای انتخاب رکوردهای تکرار[3] (RR) وجود دارد. این بخش بر اساس سطح جامعیت[4] (IL) مورد انتظار، اقدام به انتخاب رکوردهای تکرار می‌نماید. جریان داده اصلی (S_m) با کلید رمزنگاری اول (K_m) و جریان داده تکرار (S_r) با کلید رمزنگاری دوم (K_r) رمزنگاری شده و به DSMS ارسال می‌شود. یکی از مزایای روش این است که کاربر نیازی به دانستن الگوریتم انتخاب رکوردهای تکرار ندارد از این رو مالک جریان داده می‌تواند در ساعات مختلف از مکانیزم‌ها و نرخ‌های متفاوتی برای انتخاب رکوردهای تکرار بدون نیاز به مطلع ساختن کاربر، استفاده کند. این مطلب استقلال کامل مالک جریان داده و کاربر را نشان می‌دهد. هم‌چنین انتخاب روش‌های متفاوت انتخاب رکوردهای تکرار در ساعات مختلف، ضریب امنیتی سیستم را بالا خواهد برد.

با توجه به معماری ارائه شده، فرضیات اولیه زیر در نظر گرفته شده است:

- به تمامی رکوردهای جریان داده (اصلی و تکراری) یک سرآیند به‌صورت زیر اضافه می‌گردد:

$$\forall\, r \notin S_m\colon H = Hash(a_1|a_2|\dots|a_n)$$

$$\forall r \in S_r\colon H = Hash(a_1|a_2|\dots|a_n) + 1 \qquad (۱)$$

کاربران با استفاده از مقدار سرآیند فوق صحت رکوردهای دریافتی از DSMS را کنترل می‌کنند. هم‌چنین کاربر از این سرآیند برای تشخیص رکوردهای تکرارشده از رکوردهای غیر تکرارشده استفاده می‌کند.

- سرور بر اساس نرخ جریان داده ورودی زمان‌های مختلف مجبور به کاهش بار ورودی است ولی حداکثر نرخ کاهش بار[5] (LSR) از قبل در قالب یک توافقنامه سطح سرویس[6] (SLA) تعیین می‌شود. جدول (۱) نمونه‌ای از این توافقنامه را نمایش می‌دهد. براساس همین توافقنامه، مالک جریان داده نرخ تولید جریان داده تکرار (RRR) را مشخص می‌کند.

جدول ۱. نمونه‌ای از جدول توافقنامه سطح سرویس سرور و کاربر

از ساعت	تا ساعت	LSR (درصد)
۰۰:۰۰	۰۸:۰۰	۸
۰۸:۰۱	۱۶:۰۰	۱۴
۱۶:۰۱	۲۳:۵۹	۵

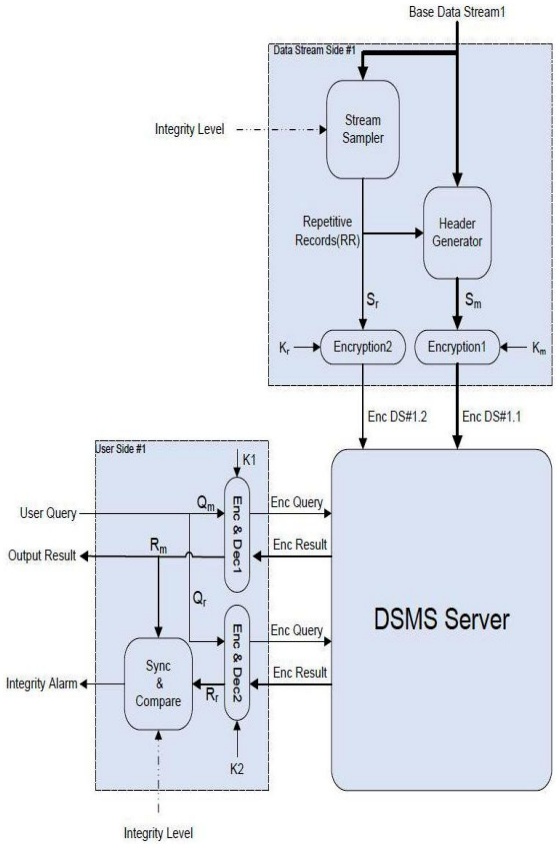
در این معماری به ازاء هر پرس و جوی درخواستی کاربر(Q_m) یک پرس‌وجوی دیگر با همان شرایط برروی جریان داده تکراری (Q_r) ایجاد شده و برای سرور ارسال می‌گردد. کاربر برای ممیزی تمامیت، نتایج این پرس‌وجو (R_r) را با نتایج پرس و جوی اصلی (R_m) به‌صورت تقاطعی مقایسه می‌کند. با توجه به اینکه دو جریان داده اصلی و تکرار با کلید مختلف رمزنگاری شده‌اند، بنابراین سرور امکان تطبیق دو جریان داده و کشف رکوردهای متناظر بین دو جریان داده را ندارد. به این صورت عملا شرط عدم ارتباط بین دو سرور در کنترل‌های تقاطعی محقق شده و با استفاده از یک سرور امکان کنترل تقاطعی عملکرد سرور به‌دست می‌آید.

برای پیاده‌سازی معماری فوق سه چالش اساسی زیر روبه‌رو هست:

- تولید جریان داده تکرار
- همزمانی اعمال پرس‌وجوی کاربر بر روی هر دو جریان داده اصلی و تکرار
- ممیزی جامعیت (بررسی رکوردهای تکرار در پاسخ‌ها) توسط کاربر با حداقل سربار ممکن.

در ادامه این مقاله راه‌حل‌هایی برای اجرای موارد فوق ارائه شده است.

[1] Mapping
[2] Data Stream Owner
[3] Repetitive Records
[4] Integrity Level
[5] Load Shedding Ratio
[6] Service Level Agreement



شکل ۲. معماری کلی سیستم

۲.۴. تولید جریان دادهٔ تکرار

همان‌طور که قبلا اشاره شد، جریان داده تکرار (S_r) با نمونه‌برداری از جریان داده اصلی (S_m) ساخته می‌شود. برای نمونه‌برداری از یک جریان داده روش‌های متفاوتی از یک نمونه برداری ساده تصادفی تا روش‌های پیچیده‌تر ارائه شده است که از حیطه این تحقیق خارج هستند[۱۸- ۱۵]. روش‌های ساده‌تر عمدتا سربار کمتری دارند و روش‌های پیچیده‌تر سربار بیشتری را به مالک جریان داده تحمیل می‌کنند. از طرف دیگر در روش ما، هرچه نمونه‌های انتخاب شده بیشتر در محدوده پرس و جوی کاربران قرار گیرند احتمال کشف حملات افزایش می‌یابد. بنابراین نحوه انتخاب رکوردها تاثیر مستقیمی در کیفیت کشف حملات دارد.

توجه به این نکته ضروری است که در این روش تولید جریان داده توسط DSO انجام می‌شود و کاربر هیچ‌گونه سرباری در تولید جریان داده تکرار متحمل نمی‌شود. نکته قابل توجه دیگر اینکه در روش پیشنهادی نیازی نیست کاربر از چگونگی تولید جریان داده تکرار مطلع شود و به همین دلیل DSO می‌تواند در شرایط زمانی مختلف الگوریتم‌های متفاوتی را برای نمونه‌برداری انتخاب نماید.

برای تولید جریان داده تکرار عملیات زیر توسط DSO انجام می‌شود:

- DSO به تمامی رکوردهای جریان داده اصلی (S_m) یک شماره رکورد[1] (#RS) اضافه می‌نماید. این فیلد از نوع عددی (عدد صحیح بدون علامت ۴ بایتی) است و تاثیر چندانی در حجم رکوردها ندارد. DSO جریان داده اصلی را در پنجره‌های متوالی به نام پنجره عمومی[2] (GW) با اندازه‌ای نسبتا بزرگ دسته‌بندی می‌کند. مقدار #RS اولین رکورد در هر GW برابر صفر است. طول این پنجره‌ها[3] (L_{GW}) یکی از پارامترهایی است که قبلا توسط DSO و کاربران مورد توافق قرار گرفته است.
- مقدار L_{GW} بسیار بزرگتر از پنجره پرس و جوهای کاربران(t_m) انتخاب می‌شود ($L_{GW}>>n$) تا هر پنجره عمومی تعدادی قابل توجهی از پنجره‌های جستجو را شامل شده و به ازای هر پنجره عمومی، حتما تعدادی رکوردهای پاسخ برای کاربران ارسال می‌شود.
- DSO با استفاده از یکی از الگوریتم‌های نمونه‌برداری از جریان داده و بر مبنای نسبت تکرار رکوردها (RRR) تعدادی از رکوردهای جریان داده اصلی را انتخاب نموده و با همان #RS قبلی در قالب یک جریان داده جدید (پس از رمز نگاری با یک کلید ثانویه) برای سرور ارسال می‌کند.
- در جریان داده اصلی، رکوردهایی که برای ارسال در جریان تکراری انتخاب شده‌اند از طریق سرآیند آنها قابل تفکیک از سایر رکوردها هستند.

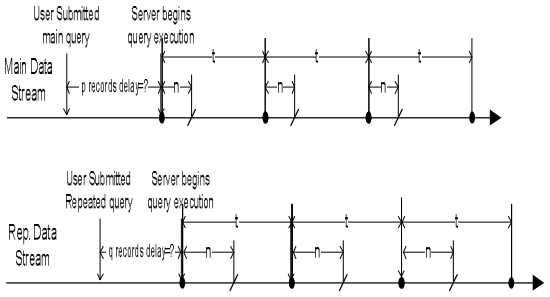
۵. کنترل جامعیت پاسخ‌ها

برای ممیزی جامعیت پاسخ‌ها بر مبنای مدل محاسبات تقاطعی بایستی نتایج به‌دست آمده از دو پرس و جوی اصلی و تکراری با یکدیگر مقایسه شوند. اما برای مقایسه نتایج دو پرس‌وجو چالش‌های زیر روبه‌رو هست:

- سرور پس از یک تاخیر زمانی، شروع به اجرای پرس‌وجوی‌های درخواستی کاربر می‌کند ولی کاربر از مدت دقیق این تاخیرها مطلع نیست. در نتیجه کاربر نمی‌تواند موقعیت دقیق پنجره پرس و جوها را در جریان‌های داده تعیین کرده و هم‌پوشانی این پنجره‌ها را تعیین کند(شکل (۳)).
- مطابق شکل (۳)، مقایسه نتایج تنها در محدوده‌هایی که دو پرس و جو با هم هم‌پوشانی دارند قابل انجام است. چنانچه مقادیر t و n دو پرس‌وجو به درستی انتخاب نشوند ممکن است پرس و جوی اصلی (Q_m) و تکراری (Q_r) با یکدیگر هم‌پوشانی نداشته باشند و عملا امکان کنترل نتایج از بین برود.

[1] Record Sequence Number
[2] General Window
[3] Length of GW

- مقایسه پاسخ‌ها باید با دقت مورد انتظار کاربر و با حداقل سربار و حداکثر سرعت انجام شود. هم‌چنین مسئله کاهش بار ورودی که سرور در زمان های پرباری انجام می‌دهد باید در نظر گرفته شود.



شکل ۳. تاخیر سرور در اجرای پرس و جوها و هم‌پوشانی آنها

در ادامه این بخش ابتدا راه حل‌های لازم برای هر یک از چالش‌های فوق را مورد بررسی قرار داده و سپس الگوریتم جامع ممیزی جامعیت پاسخ‌ها را ارائه می‌نماییم.

۱.۵. تخمین موقعیت پنجره پرس‌وجو توسط کاربران

پس از آنکه کاربر پرس و جوهای Q_m و Q_r را در سرور ثبت نمود سرور با یک تاخیر زمانی شروع به اجرای این دو پرس و جو ارسال نتایج برای کاربر می‌نماید ولی سرور لزوما اجرای این دو پرس و جو را به‌طور هم‌زمان آغاز نمی‌کند (شکل (۳)). کاربر از اندازه تاخیر شروع اجرای پرس و جوها مطلع نیست. یک الگوریتم برای تخمین موقعیت پنجره پرس و جوی کاربر بر مبنای نتایج به‌دست آمده ارائه شده است[۱۰]. با توجه به نتایج مناسب این الگوریتم، در این تحقیق نیز از همان الگوریتم برای تخمین موقعیت پنجره پرس و جوها استفاده شده است. تخمین موقعیت پنجره پرس و جوهای Q_m و Q_r به‌صورت مستقل با استفاده از دو instance متفاوت از کلاس Estimate QW انجام می‌شود.

۲.۵. هم‌پوشانی پرس‌وجوهای Q_M و Q_R

مطابق شکل (۳)، سرور پرس و جوی Q_m با پارامترهای t_m و n_m و پرس و جوی Q_r با پارامترهای t_r و n_r اجرا می‌کند. یکی از چالش‌های اساسی در روش ارائه شده تضمین هم‌پوشانی Q_m و Q_r است. به دست آوردن رکوردهای هم‌پوشان از اهمیت خاصی برخوردار است زیرا با ممیزی رکورهای به‌دست آمده از هر دو جریان می‌توانیم از صحت عملکرد سرور اطمینان حاصل نماییم. باید توجه داشت که مقادیر t_m و n_m توسط کاربر تعیین می‌گردد و تنها می‌توان مقادیر t_r و n_r را تعیین کرد. هر چه این مقادیر به نحوی تعیین شود که تعداد رکوردهای هم‌پوشان بیشتر شوند دقت و سرعت در تشخیص حملات جامعیت بیشتر خواهد شد. برای آنکه در هر بار هم‌پوشانی حداکثر رکوردهای تکرار در ناحیه هم‌پوشان قرار داده شود، مقدار n_r به صورت زیر تعیین می‌شود:

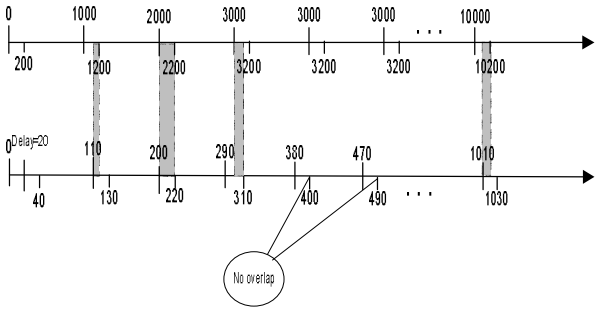
$$n_r = n_m \times RRR \qquad (۲)$$

با در نظر گرفتن وضعیت فوق تعیین t_r از اهمیت ویژه‌ای برخوردار می‌شود. واضح است که هر چه مقدار t_r به مقدار n_r نزدیک‌تر باشد تعداد رکوردهای در محدوده هم‌پوشان بیشتر می‌شود ولی سربار سیستم نیز افزایش می‌یابد، چرا که Q_r در محدوده‌هایی از S_r اجرا می‌شود که در محدوده پرش Q_m روی S_m قرار دارد. به هر حال بهترین حالت زمانی رخ می‌دهد که مقدار t_r برابر n_r تعیین گردد.

برای توضیح بهتر هم‌پوشانی پرس و جوها، مثال زیر را در نظر بگیرید:

RRR=10%,
Q_m: t_m=1000, n_m=200
Q_r: $n_r = n_r \times RRR$=20, t_r=90
Delay=20

شکل (۴) وضعیت هم‌پوشانی Q_m و Q_r را نمایش می‌دهد. باید توجه داشت که با توجه به نرخ رکوردهای تکراری انتخاب شده، به طور متوسط به ازای هر ۱۰ رکورد دریافتی از Q_m یک رکورد از Q_r دریافت می‌شود.



شکل ۴. هم‌پوشانی پرس و جوهای اصلی و تکراری

سرور Q_m و Q_r را با تاخیرهای متفاوتی اجرا می‌کند، بنابراین در شکل فرض شده Q_r نسبت به Q_m با مقداری تاخیر اعمال شده است. با توجه به بازه در نظر گرفته شده دو جریان داده در بخش‌هایی دارای هم‌پوشانی هستند. باید توجه داشت این مسئله هر هم‌پوشانی در سه پنجره متوالی رخ داده است و هم‌پوشانی‌های بعدی نیز به همین ترتیب در سه پنجره متوالی رخ می‌دهد اما ممکن است بین هر همپوشانی تا همپوشانی بعدی تعدادی پنجره بدون هم‌پوشانی وجود داشته باشند. در حالت کلی می‌توان فاصله هر هم‌پوشانی تا هم‌پوشانی بعدی را به صورت زیر به‌دست آورد.

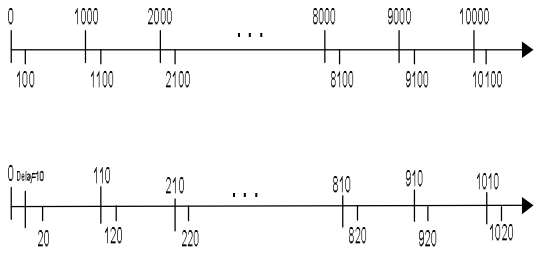
(۳) فاصله محدوده‌های هم‌پوشان = ک.م.م $(t_m , (t_r / RRR))$

با توجه به رابطه بالا فاصله هم‌پوشانی‌ها از یکدیگر در مثال قبل برابر است با :

۹۰۰۰ =((۹۰/۰.۱) , ۱۰۰۰) ک.م.م

اما در موارد بسیار خاص (شکل (۵)) ممکن است با انتخاب یک مقدار t_r نامناسب هیچ هم‌پوشانی بین دو جریان داده رخ ندهد. به مثال زیر توجه کنید:

RRR=10%,
Q_m: t_m=1000, n_m=100
Q_r: $n_r = n_r \times RRR$=10, t_r=100
Delay=10



شکل ۵. عدم هم‌پوشانی پرس‌وجوهای اصلی و تکراری

در صورت رخداد این موارد بسیار نادر، با توجه به فرض‌های اولیه چون اندازه پنجره عمومی به قدری بزرگ تعیین می‌شود که در هر یک از آنها حداقل یک رکورد پاسخ قرار می‌گیرد بنابراین در صورتی که پنجره عوض شود و هیچ رکوردی از جریان داده دوم در محدوده هم‌پوشان دو پرس و جو قرار نگیرد این وضعیت قابل تشخیص است. در چنین مواردی با تغییر مقدار t_r، اعمال درخواست بر جریان داده دوم دوباره آغاز می‌شود. هم‌چنین می‌توان با قرار دادن یک تایمر از وقوع چنین اتفاقی جلوگیری کرد، در صورتی که این تایمر به مقدار صفر رسید و هیچ رکوردی هم‌پوشانی از جریان داده دوم دریافت نشد، می‌توان مقدار t_r را تغییر داد. عدم هم‌پوشانی دو جریان زمانی پیش می‌آید که رابطه زیر برقرار باشد:

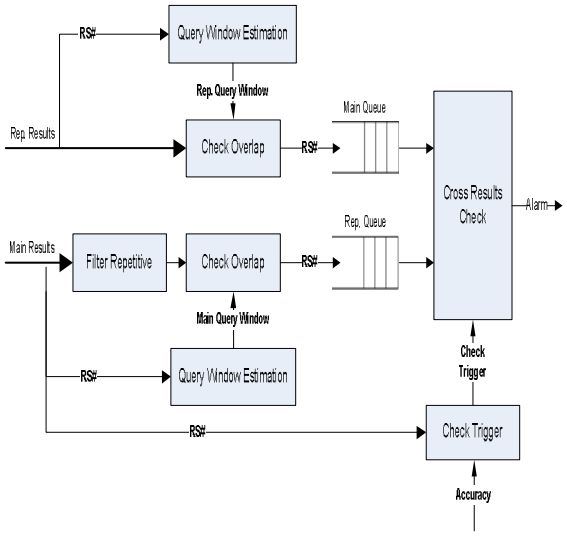
$$\text{ک.م.م}\ (t_m , (t_r / RRR)) = t_m \qquad (۳)$$

برای عدم هم‌پوشانی دو جریان داده برقراری این شرط لازم اما کافی نیست. اما در صورتی که این شرط برقرار باشد با احتمال 1-RRR عدم هم‌پوشانی رخ می‌دهد. پس تنها کافی است که مقدار t_r به نحوی انتخاب شود که رابطه بالا برقرار نباشد.

۳.۵. ساختار الگوریتم کنترل جامعیت

شکل (۶) ساختار کلی الگوریتم ممیزی جامعیت را نمایش می‌دهد. همان‌طور که در شکل ملاحظه می‌شود، کاربر با استفاده از شماره رکوردهای دریافتی، پنجره پرس و جوهای اصلی و تکراری را تخمین می‌زند (این پنجره‌ها برای تعیین محدوده هم‌پوشان پرس و جوها مورد استفاده قرار می‌گیرند). در ضمن کاربر رکوردهای تکراری موجود در R_m جدا نموده و رکوردهایی که در محدوده هم‌پوشان دو پرس و جو قرار دارند را در صف اصلی (Main Queue) وارد می‌کند. هم‌چنین رکوردهای موجود در R_r که در محدوده هم‌پوشان قرار دارند را نیز در صف تکراری (Rep. Queue) وارد می‌کند. اکنون برای ممیزی جامعیت کافی است شماره رکوردهای موجود در دو صف را با یکدیگر مقایسه نماییم. بدیهی است چنانچه یک شماره رکورد موجود در یک صف در صف دیگر وجود نداشته باشد این رکورد از نتایج حذف شده است. برای قضاوت در خصوص جامعیت نتایج باید به موارد زیر توجه نماییم:

۱. حذف رکوردها ممکن است به علت کاهش بار ورودی توسط سرور انجام شده باشد. به همین دلیل نمی‌توان در خصوص حذف یک رکورد از R_m یا R_r به صورت مستقل قضاوت نمود.
۲. با توجه به حجم بالای رکوردهای جریان داده، بررسی صف‌ها به ازای دریافت هر رکورد موجب تحمیل سربار بالایی به کاربر می‌شود.
۳. عملیات ممیزی جامعیت باید براساس دقت مورد نظر کاربر انجام شود. چون ممکن است کاربر در مقابل کاهش سربار محاسباتی خود، حذف برخی از رکوردها توسط سرور را نادیده بگیرد.



شکل ۶. شمای الگوریتم ممیزی جامعیت توسط کاربر

با توجه به موارد فوق عملیات ممیزی جامعیت (یعنی مقایسه محتویات صف‌ها) را باید بر اساس دریافت یک مجموعه از رکوردهای پاسخ انجام داد و نه دریافت تک‌تک رکوردها. به این صورت جریان[1] داده پاسخ، در قالب تعدادی پنجره متوالی تحت عنوان پنجره ارزیابی (CW) مورد بررسی قرار گرفته و هشدارهای لازم بر اساس این پنجره‌ها اعلام می‌شود. در ادامه این بخش، شرح الگوریتم‌های "کنترل هم‌پوشانی"، "ارزیابی جامعیت" و "الگوریتم نهایی" ارئه شده است.

۱.۳.۵. کنترل هم‌پوشانی

برای بررسی وجود یک شماره رکورد از R_m در محدوده هم‌پوشان Q_r باید شماره رکورد مورد نظر را به محل قرار گرفتن این رکورد در Q_r نگاشت نماییم. برای این منظور کافیست عبارت زیر را محاسبه کنیم:

$$Map_r(RS\#_m) = (RS\#_m - StartOf(QW_r)) \bmod t_r \quad (۴)$$

به این صورت چنانچه $Map_r(RS\#_m)$ در بازه $[0, n_r]$ باشد رکورد دریافتی در محدوده هم‌پوشان Q_m و Q_r است. چنانچه در زمان کنترل هم‌پوشانی Q_r و Q_m هر دو در یک GW نباشند باید $StartOf(QW_r)$ بر اساس پنجره متناظر Q_m محاسبه گردد. برای کنترل هم‌پوشانی رکوردهای R_r با Q_m فرآیند فوق با جایگزینی m و r اجرا می‌شود.

۲.۳.۵. ارزیابی تقاطعی نتایج

فرض کنید R_m و R_r به ترتیب مجموعه رکوردهای حاصل از اجرای Q_m و Q_r روی S_m و S_r به صورت زیر باشند:

$$R_m = Apply(Q_m, S_m)$$

$$R_r = Apply(Q_r, S_r) \quad (۵)$$

و R'_m و R'_r به ترتیب مجموعه رکوردهای دریافتی توسط کاربر باشند.

مطابق آنچه در بخش قبل اشاره شد مقایسه تقاطعی نتایج تنها در محدوده هم‌پوشان دو پرس و جو امکان پذیر است. چنانچه D_m و D_r به ترتیب مجموعه رکوردهای حذف شده از R_m و R_r در محدوده هم‌پوشان Q_m و Q_r توسط سرور یا دشمن به صورت زیر باشند:

$$D_m = \{r \mid r \in OverLap(Q_m, Q_r) \wedge r \in R_m \wedge r \notin R'_m\}$$

$$D_r = \{r \mid r \in OverLap(Q_m, Q_r) \wedge r \in R_r \wedge r \notin R'_r\} \quad (۶)$$

چهار وضعیت قابل انتظار است که عبارتند از:

- وضعیت۱: رکورد r_k از R_r حذف ولی از R_m حذف نشده است، یعنی:

$$r_k \in D_m \wedge r_k \notin D_r \quad (۷)$$

- وضعیت۲: رکورد r_k از R_m حذف شده است ولی از R_r حذف نشده است یعنی:

$$r_k \notin D_m \wedge r_k \in D_r \quad (۸)$$

- وضعیت ۳: رکورد r_k از R_r و R_m حذف شده است یعنی:

$$r_k \in D_m \wedge r_k \in \quad (۹)$$

- وضعیت ۴: رکورد r_k از R_r و R_m حذف نشده است یعنی:

$$r_k \notin D_m \wedge r_k \notin D_r \quad (۱۰)$$

واضح است که حالت ۴ حالت کارکرد صحیح سرور بدون وقوع حمله است. با استفاده از روش تقاطعی، حالت‌های ۱ و ۲ قابل کشف توسط کاربر هستند و حالت ۳ حالتی است که کاربر امکان کشف حمله را ندارد (حالت فرار). در بخش ۶ حالت فرار را به صورت دقیق با یک مدل احتمالاتی مورد ارزیابی قرار گرفته و نشان داده شده که احتمال وقوع چنین حالتی بسیار کم است.

برای کشف حالت‌های ۱ و ۲ کاربر باید رکوردهای در محدوده هم‌پوشانی پرس‌وجوها را با یکدیگر مقایسه کند. بدیهی است در این مقایسه لازم نیست محتویات رکوردها به‌طور کامل با هم مقایسه شوند، بلکه تنها کافی است #RS این رکوردها با هم مقایسه شوند. با توجه به حجم زیاد رکوردها در سیستم‌های جریان داده، انجام عملیات مقایسه به ازای دریافت هر رکورد پاسخ، بار زیادی را به کاربر تحمیل می‌کند.

از طرف دیگر کاربر بلافاصله پس از دریافت یک رکورد از R_m رکورد متناظر آن از R_r را دریافت نمی‌کند. به همین جهت در روش پیشنهادی، #RS رکوردهای دریافتی که در محدوده هم‌پوشانی دو پرس‌وجو قرار دارند در دو صف مجزا قرار داده می‌شوند. کاربر پس از دریافت تعداد معینی از رکوردهای پاسخ به بررسی و مقایسه محتویات صف‌ها پرداخته و در رابطه با جامعیت نتایج قضاوت می‌کند. الگوریتم مقایسه تقاطعی نتایج در زیر ارائه شده است.

در این الگوریتم در یک حلقه تکراری تا زمانی که هر یک از صف‌ها خالی نشود، شماره رکوردهای موجود در ابتدای دو صف با یکدیگر مقایسه می‌شوند. چنانچه #RS موجود در ابتدای هر دو صف یکسان باشد، این رکورد از هیچ‌یک از نتایج حذف نشده است (وضعیت ۴). چنانچه #RS موجود در ابتدای صف رکوردهای تکراری از #RS ابتدای صف رکوردهای اصلی کوچک‌تر باشد به این معناست که یک رکورد در R_r وجود دارد که در R_m دریافت نشده است (وضعیت ۲). بدیهی این یک وضعیت حمله است. وضعیت عکس حالت فوق (وضعیت ۱) نیز در ادامه الگوریتم بررسی شده است.

[1] Check Window

```
Algorithm Check Integrity
    Ok Count←0
    Not Ok Count←0
    While (Rep Queue is not empty AND Main Queue is not
empty)
        Rep Record RS#←First Item of Rep Queue
        Main Record RS#← First Item of Main Queue

        If (Rep Record RS# = Main Record RS#)
            OkCount++
            Dequeue First Item of Rep Queue
            Dequeue First Item of Main Queue
        Else
            If (Rep Record RS# < Main Record RS#)
                Not Ok Count++
                Dequeue First Item of Rep Queue
            Else
                Not Ok Count++
                Dequeue First Item of Main Queue
            End of If
        End of If
    End of While

    Delete Ratio = Not Ok Count/(Ok Count+ Not Ok Count)
                If (Delete Ration>LSR)
        Alarm Error
End of Algorithm
```

در الگوریتم فوق فرآیند مقایسه محتویات صف‌ها تا خالی شدن یک صف ادامه پیدا می‌کند. این مساله به این علت است که ممکن است نتایج یک پرس جو دیرتر از دیگری به کاربر برسند و کاربر باید برای مقایسه باقیمانده رکوردهای موجود در یک صف تا رسیدن نتایج پرس و جوی دیگر صبر کند.

چنانچه سرور مجاز به کاهش بار نباشد، بایستی مقدار متغیر NotOkCount پس از هر اجرای الگوریتم صفر باشد و هر مقدار غیر صفر این متغیر به معنی کشف یک حمله است. اما در وضعیت کاهش بار سرور، برای قضاوت در خصوص جامعیت پاسخ‌ها باید نسبت تعداد رکوردهای حذف شده بر تعداد کل رکوردها به‌صورت زیر محاسبه گردد:

$$DeleteRation = \frac{NotOkCount}{OkCount+NotOkCount} \qquad (۱۱)$$

چنانچه DeleteRatio بیش از LSR مجاز باشد به معنی حذف غیر مجاز رکوردها توسط سرور و بنابراین وقوع یک حمله است. بنابراین داریم:

$$Alarm = f(x) = \begin{cases} True, & DeleteRatio \geq LSR \\ False, & DeleteRatio < LSR \end{cases} \qquad (۱۲)$$

۴.۵. الگوریتم ممیزی جامعیت

مطابق معماری الگوریتم ارائه شده در شکل (۶) الگوریتم ممیزی جامعیت نتایج به صورت زیر است:

```
Algorithm DoCheck
  While (True) {
    InputRecord←Recive One Record From Server
    If (InputRecord is From Qr) {
      Estimate Qr Query Window Using InputRecord(RS#)
      If (InputRecord is in Overlap Area)
        Enqueue InputRecord(RS#) to Rep. Queue
    Else
      Estimate Qm Query Window Using InputRecord(RS#)
      If (InputRecord is Repetitive Record)
          If (InputRecord is in Overlap Area)
        Enqueue InputRecord(RS#) to Main Queue
    End of If

    If (A Check Trigger is Recieved)
      Do CheckIntegrity
  End of While
End of Algorithm
```

با توجه به اینکه فرآیند این الگوریتم در قسمت‌های قبلی به طور کامل شرح داده شده است، بنابراین از شرح بیشتر آن در این قسمت صرف‌نظر شده می‌شود.

۶. مدل‌سازی احتمالاتی سیستم

برای تحلیل رفتار سیستم در این بخش یک مدل احتمالاتی از رفتار سیستم ارائه نموده‌ایم. در این مدل، احتمال فرار سرور یا دشمن در حملات به تمامیت مدل شده است. در این حملات تعدادی از رکوردهای پاسخ از نتایج حذف شده و برای کاربر ارسال نمی‌شوند. بر مبنای الگوریتم ممیزی ما، کاربر نتایج دریافتی از دو پرس و جوی اصلی و تکرار را در محدوده هم‌پوشانی آنها به‌صورت تقاطعی مقایسه نموده و در خصوص تمامیت آنها قضاوت می‌کند. به این ترتیب، حملات تمامیت در شرایط زیر قابل کشف توسط کاربر نیستند:

۱. رکوردها از محدوده غیرهم‌پوشان پرس‌وجوها حذف شوند.
۲. اگر رکوردی از نتایج واقع در محدوده هم‌پوشان یک پرس و جو حذف شود همان رکورد از نتایج پرس و جوی دیگر حذف شود.

در این مدل احتمالاتی، فرض‌های زیر در نظر گرفته شده است:

۱. تعداد رکوردهای پاسخ حاصل از اجرای Qm در یک محدوده زمانی مشخص برابر N است.
۲. تعداد رکوردهای پاسخ (از N رکورد) که در محدوده هم‌پوشانی Qm و Qr قرار دارند برابر k است.

۳. سرور یا دشمن تعداد m رکورد از رکورد پاسخ را حذف می‌کند.
۴. رکوردهای جریان داده تکرار به‌صورت یکنواخت و با احتمال یکسان از جریان داده اصلی انتخاب شده‌اند.
۵. احتمال حذف هریک از رکوردهای پاسخ توسط سرور یکسان است.

با توجه به فرض‌های فوق مساله به‌دست آوردن احتمال فرار سرور یا دشمن در حذف m رکورد از N رکورد پاسخ پرس و جوی اصلی به شرطی که پرس‌وجوی اصلی و تکرار مجموعا k رکورد پاسخ در محدوده هم‌پوشانی داشته باشند.

فرض شود که متغیر تصادفی X تعداد رکوردهای حذف شده توسط سرور یا دشمن باشد، در نتیجه احتمال حذف X رکورد پاسخ از محدوده غیر هم‌پوشان Q_m و Q_r به صورت $P_1^N(X)$ تعریف می‌شود. با توجه به اینکه احتمال حذف هر رکورد از مجموعه پاسخ یکسان است، بنابراین:

$$P_1^N(X) = \prod_{j=0}^{X} \frac{N-k-j}{N-j} \quad (۱۳)$$

احتمال حذف X رکورد پاسخ از محدوده هم‌پوشان Q_m و Q_r و حذف همان رکورد از نتایج Q_r (حذف یک رکورد از مجموع N*RRR رکورد) به صورت $P_2^N(X)$ تعریف می‌شود. در این صورت:

$$P_2^N(X) = \prod_{j=0}^{X} \frac{k-j}{N-j} * \frac{1}{N*RRR-j} \quad (۱۴)$$

اکنون برای محاسبه احتمال فرار در حذف m رکورد، مجموعه فضای حالت به صورت زیر است:

- حذف تمامی m رکورد از محدوده غیرهم‌پوشان Qm و Qr
- حذف m-1 رکورد از محدوده غیر هم‌پوشان Qm و Qr و حذف یک رکورد از محدوده هم‌پوشان Qm و حذف همان رکورد از Qr
- ...
- حذف m رکورد از محدوده هم‌پوشان Qm و حذف همان رکوردها از Qr

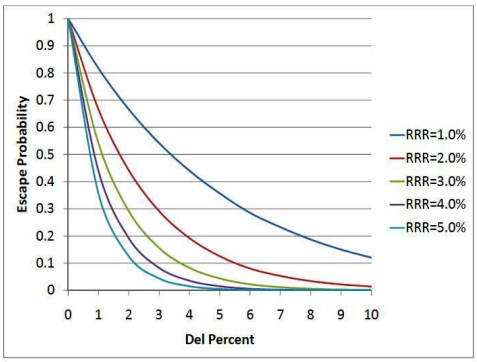
بنابراین $P^N(m)$ یعنی احتمال فرار در حذف m رکورد از مجموع N رکورد دریافتی به‌صورت زیر است:

$$P^N(m) = P_1^N(m) * P_2^{N-i}(0) + P_1^N(m-1) * P_2^{N-m+1}(1) + \dots + P_1^N(0) * P_2^{N-m}(m) \quad (۱۵)$$

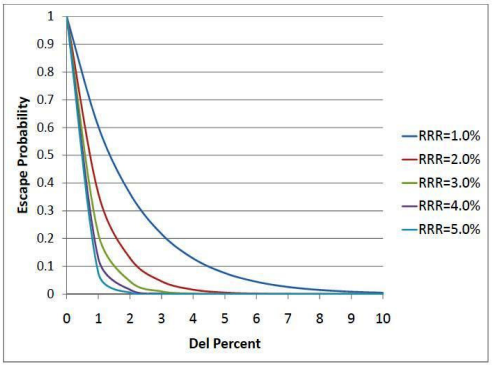
$$P^N(m) = \sum_{i=0}^{\min(m,k)} P_1^N(m-i) * P_2^{N-i}(i) \quad (۱۶)$$

$$P^N(m) = \sum_{i=0}^{m} \prod_{j=0}^{m-i} \frac{N-k-j}{N-j} * \prod_{j=0}^{i} \frac{k-j}{N-i-j} * \frac{1}{(N-i)*RRR-j} \quad (۱۷)$$

در عبارت فوق چنانچه k<m باشد مقدار k جایگزین m خواهد شد. بر مبنای این مدل احتمالاتی، چنانچه تعداد رکوردهای دریافتی را عدد ثابت ۲۰۰۰ در نظر بگیریم و فرض کنیم ۲٪ رکوردهای جریان داده تکراری در محدوده هم‌پوشان Q_m و Q_r قرار گیرند. شکل (۷)، احتمال فرار سرور یا دشمن را در حملات تمامیت را بر اساس درصد حذف(β) نتایج نشان می‌دهد. در این شکل، رفتار سیستم بر اساس دو پنجره ارزیابی مختلف و چند RRR متفاوت نمایش داده شده است. همان‌طور که در شکل مشاهده می‌شود، هر چه بر میزان درصد حذف از رکوردها یا بر مقدار RRR افزوده می‌شود احتمال فرار با شیب نسبتا زیادی به سمت صفر میل می‌کند. به عنوال مثال با RRR=4% و β=5% احتمال فرار تقریبا برابر صفر است. هر چند بزرگ‌تر شدن CW باعث افزایش تاخیر در کشف حملات می‌شود، ولی به شدت سبب بهبود عملکرد سیستم در کشف حملات می‌گردد. مشابه شکل (۷)، شکل (۸) نیز احتمال فرار در حملات حذف براساس RRR برای درصد حذف‌های متفاوت را نشان می‌دهد. همان‌طور که مشاهده می‌شود، با افزایش مقدار RRR احتمال فرار به سرعت به سمت صفر میل می‌کند به عنوال مثال در حالتی که ۵٪ از رکوردها حذف می‌شوند، حتی هنگامی که مقدار RRR، ۳٪ می‌باشد احتمال فرار کمتر از ۵٪ می‌باشد. پرواضح است که در هر حال با افزایش رکوردهای دریافتی احتمال فرار سرور یا دشمن در حملات جامعیت به سمت صفر میل می‌کند، بنابراین روش پیشنهادی حملات جامعیت را حتما کشف می‌کند.

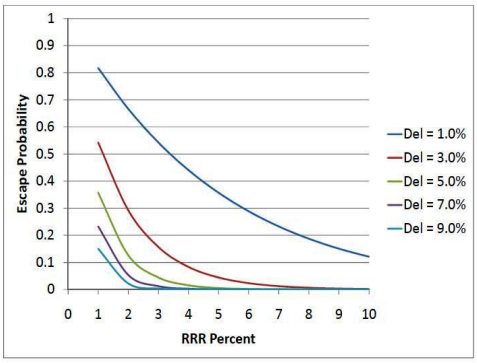


a) Check Window =2000

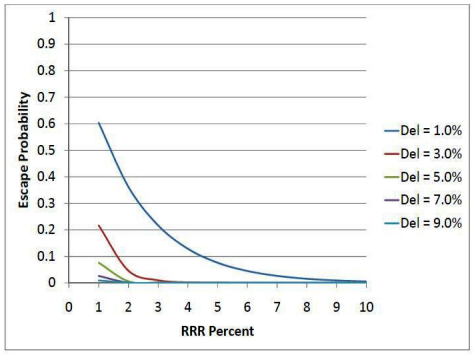


b) Check Window = 5000

شکل ۷. احتمال فرار سرور یا دشمن در حملات تمامیت



a) Check Window = 2000



b) Check Window = 5000

شکل ۸. احتمال فرار در حملات حذف بر اساس RRRهای مختلف

۷. ارزیابی و تفسیر نتایج

برای شبیه‌سازی مالک داده، سرور و کاربر، از سه کامپیوتر با پردازنده Pentium IV 2.0GHz، حافظه اصلی ۲ گیگابایت و هارد دیسک با ظرفیت ۲۰۰ گیگا بایت استفاده شده است که این سه دستگاه از طریق یک شبکه محلی(100MBps) به یکدیگر متصل شدند. از یک پردازنده پرس‌وجو در هسته سرور استفاده کردیم. تمامی الگوریتم‌ها در این آزمایش به زبان جاوا و با استفاده از مجموعه (JDK/JRE 1.6) پیاده‌سازی شد.

برای ارزیابی، از جریان داده تولیدی بر اساس فراخوانی صفحات وب مربوط به جام جهانی ۱۹۹۸ در روزهای ۴۰ و ۴۱ که حدود ۲۰,۰۰۰,۰۰۰ رکورد را شامل می‌شود، استفاده شده است[۱۹].

۱.۷. مدل حمله

در حملات مربوط به تمامیت سرور، قسمتی از پاسخ را برای کاربر نمی‌فرستد. این نوع از حملات را می‌توان به دو دسته تقسیم‌بندی نمود، حذف قسمت‌هایی از پاسخ و همچنین کاهش بار بیش از حد مورد توافق بین کاربر و سرور. با توجه به اینکه نتایج در دو حمله مذکور تقریبا رفتار یکسانی دارند، بنابراین در این بخش تنها به بررسی حملات حذف رکوردها از نتایج پرداخته می‌شود.

در این آزمایش سرور یا دشمن β درصد از رکوردهای پاسخ را حذف می‌نماید. هم‌چنین فرض شده است رکوردهای حذف شده به صورت یکنواخت در سطح مجموعه جواب پخش شده‌اند. به این صورت تابع $f_m(x)$ به‌عنوان تابع تصمیم حذف رکوردها از جریان داده اصلی به صورت زیر تعریف می‌شود:

$$\forall r \in R_m,\ x = Random(r), 0 \le x \le 1,\ f_m(x) = \begin{cases} True, & x < \frac{\beta}{100} \\ False, & x \ge \frac{\beta}{100} \end{cases} \quad (۱۸)$$

به طریق مشابه، تابع $f_r(x)$ به عنوان تابع تصمیم حذف رکوردها از جریان داده تکراری تعریف می‌شود.

۲.۷. تشخیص حذف

کاربر می‌تواند حذف رکوردهای جریان داده اصلی و تکرار را در محدوده هم‌پوشان پرس‌وجوها تشخیص دهد. برای ارزیابی سیستم احتمال فرار سرور یا دشمن باید تعریف شود. با توجه به اینکه در الگوریتم ممیزی جامعیت، ارزیابی نتایج بر اساس پنجره‌های پرس و جوی متوالی انجام می‌شود و اخطارهای حملات بر اساس این پنجره‌ها اعلام می‌شوند، بنابراین احتمال فرار به صورت زیر تعریف شده است:

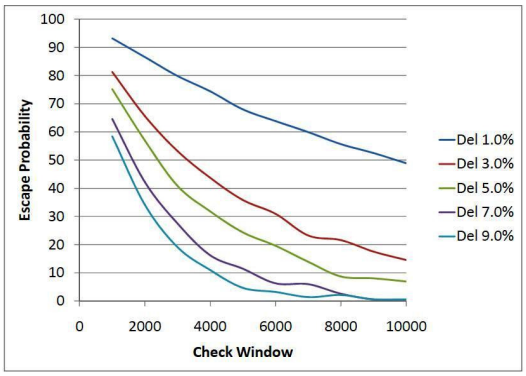
" احتمال فرار عبارت است از تعداد حملات کشف شده به کل تعداد پنجره‌های حمله شده"

$$ADR = 1 - \frac{Count(alarms)}{Count(expected\ alarm)} \quad (۱۹)$$

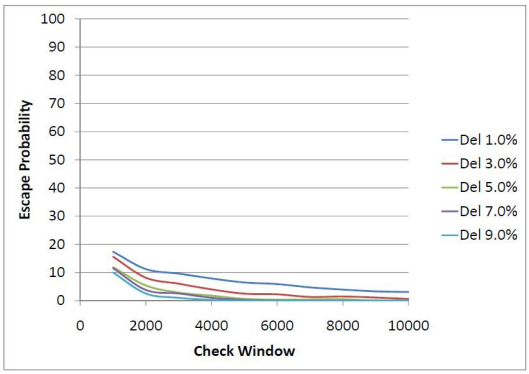
۳.۷. تشخیص حملات

شکل (۹) تاثیر اندازه پنجره ارزیابی بر احتمال فرار را نشان می‌دهد. همان‌طور که در شکل (۹) مشاهده می‌شود، با بزرگتر شدن پنجره ارزیابی، احتمال فرار به سرعت به سمت صفر میل می‌کند. این شکل هم‌چنین به‌خوبی نشان‌دهنده کارایی روش است؛ چرا که حتی هنگامی‌که انداز پنجره چک برابر ۱۰۰۰ و میزان حذف ۱٪ می‌باشد سیستم در ۱۰۰٪ موارد حملات را تشخیص می‌دهد و در هیچکدام از خطوط احتمال فرار هیچ‌گاه به ۱۰۰ نمی‌رسد.

در قسمت b شکل (۹) به وضوح دیده می‌شود حتی در حالتی که مقدار حذف ۱٪ می‌باشد و اندازه پنجره کنترل نیز برابر ۱۰۰۰ است(بدترین وضعیت در نمودار) احتمال فرار کمتر ۲۰٪ است هم‌چنین تقریبا تمام خطوط نمودار با از اندازه پنجره کنترل بیش از ۴۰۰۰ بسیار به صفر نزدیک می‌شوند.



a) RRR = 3.0%



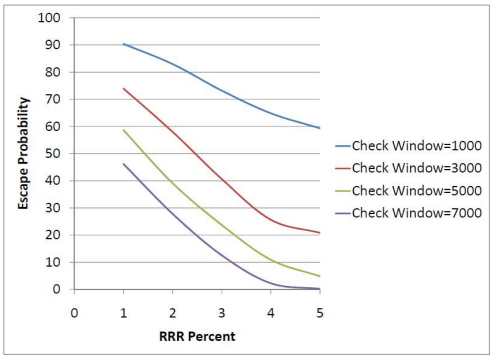
b) RRR = 5%

شکل ۹. تاثیر اندازه پنجره ارزیابی بر احتمال فرار

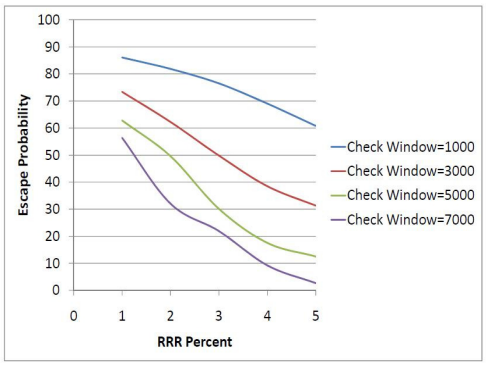
شکل (۱۰) احتمال فرار درRRRهای مختلف و با توجه به اندازه پنجره‌های ارزیابی متفاوت را برای حذف ۳٪ و ۵٪ از رکوردهای پاسخ نشان می‌دهد. همان‌طور که در شکل ۱۰) مشاهده می‌شود، با افزایش اندازه پنجره ارزیابی میزان دقت در کشف حملات به سرعت افزایش یافته و احتمال فرار کاهش می‌یابد.

۴.۷. مقایسه با روش‌های دیگر

در این قسمت به مقایسه روش ارائه شده با روشی که در مرحله قبلی این تحقیق انجام شده است، پرداخته می‌شود[۱۰]. همان‌طور که در شکل (۱۱) مشاهده می‌شود در این روش هزینه سربار بخش کاربر به شدت کاهش می‌یابد. این مساله دور از انتظار نیست چرا که در روش قبلی کاربر می‌باید در ابتدا باید اقدام به تولید رکوردهای ساختگی نموده و سپس پرس‌وجوی خود را بر روی آن اعمال نماید. این مراحل سربار زیادی را به کاربر تحمیل می‌نماید. در این روش با توجه به اینکه کاربر، ممیزی تمامیت را از طریق مقایسه شماره رکوردهای به‌دست آمده از دو جریان داده اصلی و تکرار به ممیزی می‌کند بنابراین هزینه‌های روش قبل را متحمل نمی‌شود.

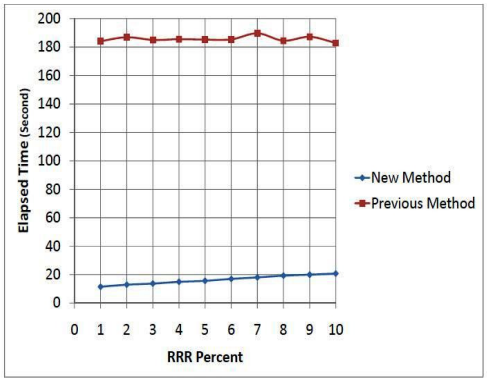


a) Del Percent = 5%



b) Del Percent = 3%

شکل ۱۰. تاثیر RRR بر احتمال فرار با توجه به پنجره‌های ارزیابی متفاوت



شکل ۱۱. مقایسه زمان اجرایی روش قدیم و جدید

تفاوت عمده دیگر بین دو روش در میزان داده‌های انتقالی از طریق کانال امن است. روش ارائه شده در تحقیق، داده‌های بسیار کمتری را از روش قبلی از طریق کانال امن رد و بدل می‌کند، چرا که در روش قبلی در ابتدا کاربر می‌باید از طریق یک کانال امن اطلاعات مربوط به نحوه تولید رکوردهای ساختگی و... را از مالک جریان داده دریافت کند، اما در روش جدید نیازی به ارتباط اولیه بین کاربر و مالک جریان داده نمی‌باشد که خود باعث کاهش سربار سیستم می‌شود.

این روش برای وضعیت‌هایی که دسترسی به کانال امن امکان‌پذیر نیست یا کاربران توان پردازشی بسیار پایینی دارند بسیار مناسب است. از طرف دیگر چنانچه کانال ارتباطی با محدودیت‌های پهنای باند مواجه باشد روش قبلی با سربار مشابه نتایج بهتری دارد.

۸. نتیجه‌گیری

در این تحقیق، به ارائه روشی، برای ممیزی تمامیت در سیستم‌های مدیریت جریان داده پرداخته شده است. در این روش، بخش کوچکی از جریان داده دوباره توسط کلید دیگری رمزنگاری شده و برای سرور ارسال می‌شود. پرس‌وجوی کاربر بر روی هر دو جریان داده اصلی و تکراری اعمال می‌شود و کاربر با استفاده از مقایسه نتایج دریافتی در خصوص جامعیت نتایج قضاوت می‌کند. مهم‌ترین خصوصیت این روش سربار بسیار کم آن برای کاربر و نتایج مناسب آن در مدلسازی احتمالاتی است. نتایج مدلسازی احتمالاتی به خوبی توسط ارزیابی عملی تایید می‌شوند. مهم‌ترین چالش باقیمانده در این مقاله نحوه انتخاب رکوردهای برای تولید جریان داده تکرار است. هرچه این انتخاب به‌صورت تصادفی‌تر و از بین رکوردهایی که با احتمال بالاتر در نتایج پرس‌وجوها قرار می‌گیرند، باشد، کارایی روش بهتر خواهد بود. در این تحقیق تمرکز عمدتا بر روی پرس‌وجوهای انتخاب است.

۹. مراجع

[1]. Chandramouli, B.; Ali, M.; Goldstein, J.; Sezgin, B.; Raman, B. S. “Data Stream Management Systems for Computational Finance.”; Computer 2010, 43, 45-52.

[2]. Hacigümüs, H.; Mehrotra, S.; Iyer, B. “Providing Database as a Service.”; 18th International Conference on Data Engineering, 2002.

[3]. Papadopoulos, S.; Yang, Y.; Papadias, D. “Continuous Authentication on Relational Streams.”; The VLDB Journal 2010, 19, 161-180.

[4]. Li, F.; Yi, K.; Hadjieleftheriou, M.; Kollios, G. “Proof-Infused Streams: Enabling Authentication of Sliding Window Queries On Streams.”; VLDB Conference, 2007.

[5]. Babcock, B.; Babu, S.; Datar, M.; Motwani, R.; Widom, J. “Models and Issues in Data Stream Systems.”; The Twenty-First ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, 2002.

[6]. Tatbul, N.; Betintemel, U.; Zdonik, S.; Cherniack, M.; Stonebraker, M. “Load Shedding in a Data Stream Manager.”; VLDB Conference, 2003.

[7]. Xie, M.; Wang, H.; Yin, J.; Meng, X. "Integrity Auditing of Outsourced Data"; VLDB Conference, 2007.

[8]. Xie, M.; Wang, H.; Yin, J.; Meng, X. "Providing Freshness Guarantees for Outsourced Databases."; The 11th International Conference on Extending Database Technology: Advances in Database Technology, 2008.

[9]. Merkle, R. C. "A Certified Digital Signature."; Springer-Verlag, 1990.

[10]. Ghayoori, M.; Salmani, K.; Haghjoo, M. S. "Detecting Changes in Stream Query Results."; The 3rd Asian Conference on Intelligent Information and Database Systems, 2011.

[11]. Yi, K.; Li, F.; Hadjieleftheriou, M.; Kollios, G.; Srivastava, D. "Randomized Synopses for Query Assurance on Data Streams."; The IEEE 24th International Conference on Data Engineering, 2008.

[12]. Brinkman, R. "Searching in Encrypted Data."; Ph.D Thesis, University of Twente, 2007.

[13]. Dong, C.; Russello, G.; Dulay, N. "Shared and Searchable Encrypted Data for Untrusted Servers."; The 22nd Annual IFIP WG 11.3 Working Conference on Data and Applications Security, 2008.

[14]. Song, D. X.; Wagner, D.; Perrig, A. "Practical Techniques for Searches on Encrypted Data."; The IEEE Symposium on Security and Privacy, 2000.

[15]. Aquino, A. L. L.; Oliveira, R. A. R.; Wanner, E. F. "A Wavelet-Based Sampling Algorithm for Wireless Sensor Networks Applications."; The ACM Symposium on Applied Computing, 2010.

[16]. Al-Kateb, M.; Lee, B. S. "Stratified Reservoir Sampling over Heterogeneous Data Streams."; The International Conference on Scientific and Statistical Database Management, 2010.

[17]. Johnson, T.; Muthukrishnan, S.; Shkapenyuk, V.; Spatscheck, O. "Query-Aware Sampling for Data Streams."; The IEEE 23rd International Conference on Data Engineering Workshop, 2007.

[18]. Bagchi, A.; Chaudhary, A.; Eppstein, D.; Goodrich, M. T. "Deterministic Sampling and Range Counting in Geometric Data Streams."; ACM Trans. Algorithms, 2007, 3, 6.

[19]. Bellare, M.; Desai, A.; Jokipii, E.; Rogaway, P. "A Concrete Security Treatment of Symmetric Encryption."; The 38th Annual Symposium on Foundations of Computer Science, 1997.