مجله علمی-پژوهشی «علوم و فناوری‌های پدافند غیرعامل»
سال دوم، شماره ۲، تابستان ۱۳۹۰: ص ۷۰- ۵۷

# کشف حملات به جامعیت داده در یک سامانه مدیریت جریان داده

**مجید غیوری ثالث[۱*]، خسرو سلمانی[۲]، مصطفی حق‌جو[۳]**

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

(دریافت: ۱۳۸۹/۰۸/۳۰، پذیرش: ۱۳۹۰/۰۳/۲۳)

## چکیده

در سال‌های اخیر جنگ‌های سایبری تبدیل به یکی از اصلی‌ترین صحنه‌های نبرد سازمان‌های اطلاعاتی و نظامی شده است. همچنین با افزایش حجم داده‌ها، پردازش جریان‌های داده و حفظ امنیت آنها تبدیل به یکی از نیازهای اساسی سازمان‌های نظامی شده است. هرچند رمزنگاری داده‌ها از دسترسی کاربر به محتویات داده‌ها جلوگیری می‌کند، ولی دشمن می‌تواند با نفوذ به کانال‌های ارتباطی انتقال اطلاعات، بدون آنکه از محتویات آنها اطلاعی داشته باشد، آنها را حذف و یا اطلاعات جعلی وارد آنها نموده و فرماندهان عملیاتی را گمراه کند. از این‌رو، ارائه روش‌هایی در جهت جلوگیری از این نوع حملات و یا همان پدافند غیرعامل در سیستم‌های رایانه‌ای از اهمیت فراوانی برخوردار است. جلوگیری از دست‌کاری غیر مجاز در داده‌های جریان داده تحت عنوان کنترل جامعیت جریان داده معرفی شده است. در این مقاله، یک مدل احتمالاتی برای ممیزی جامعیت جریان داده دریافتی از طریق یک فضای ناامن ارائه شده است. در این معماری، کاربر و مالک‌های (تولیدکنندگان) جریان داده مورد اعتماد بوده و کانال-های ارتباطی بین آنها ناامن است. سرور به‌صورت جعبه سیاه در نظر گرفته شده و فرآیند ممیزی با همکاری مالک‌های جریان داده و کاربران انجام می‌شود. به این ترتیب، امکان بهره‌برداری از سیستم‌های موجود بدون نیاز به هیچ‌گونه دست‌کاری وجود دارد. در این مدل، کاربر با صرف هزینه بسیار کم، حملات را با سرعت و دقت مناسبی تشخیص می‌دهد. در این تحقیق درستی عملکرد مدل ارائه شده با استفاده از مدل احتمالاتی اثبات شده و نتایج ارزیابی به خوبی نشان‌گر کارایی معماری ارائه شده است.

**کلیدواژه‌ها: امنیت در جریان داده، جامعیت در جریان داده، سیستم مدیریت جریان داده، اطمینان از پرس و جو، کشف تغییرات**

# Detecting Integrity Attacks to a Data Stream Management System

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

Faculty of Computer, Iran University of Science and Technology (IUST)

*(Received: 11/21/2010, Accepted: 06/13/2011)*

## Abstract

*In recent years, Cyber Warfare has been one of the most essential war scenes for intelligence and military organizations. As the volume of data is increased, data stream processing and data security would be the basic requirements of military enterprises. Although, by data encryption no one can have access to data contents, but adversaries can delete some records from data stream or add some counterfeit records in it without any knowledge about records contents. Such attacks are called integrity attacks. Because integrity control in DSMS is a continuous process, it may be categorized in passive defense activities. In this paper, we present a probabilistic auditing model for integrity control of received stream results via an unsecured environment. In our architecture, users and data stream owners are trusted and channels are unsecured. The server is considered a black box and auditing process is fulfilled by contribution between data stream owner and users. We exploit existing Data Stream Management Systems (DSMS) without any modification. Our method has no significant cost on users side and detects integrity attacks accurately and rapidly. Correctness and convergence of our probabilistic algorithm is proved and our evaluation shows very good results.*

**Keywords: Data Stream Management Systems, Data Stream Security, Data Stream Inetgrity, Query Assurance, Change Detection**

* Corresponding author E-mail: ghayoori@ihu.ac.ir

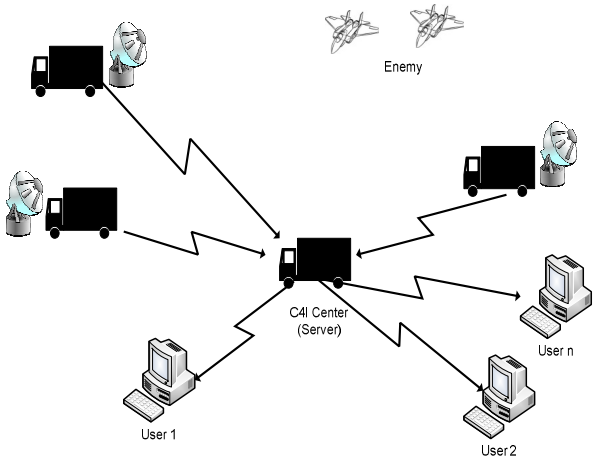
## ۱. مقدمه

توسعه روزافزون کاربردهای سامانه‌های مدیریت جریان داده[1] (DSMS) سبب تولید و عرضه سامانه‌های تجاری متنوعی در این زمینه شده است [۶-۱]. از طرف دیگر حجم داده‌های تولید شده در سازمان‌های نظامی مانند اطلاعات به دست آمده از رادارها و حس-گرهای مرزی با سرعت بسیار زیادی رو به افزایش است. بنابراین، امکان ذخیره‌سازی آنها وجود ندارد. به همین دلیل، این سازمان‌ها به شدت نیازمند بهره‌برداری از سیستم‌های مدیریت جریان داده هستند. اگر چه در شبکه‌های نظامی، داده‌ها به‌صورت رمز شده منتقل می‌گردند، ولی مکانیزم مشخصی برای تشخیص حذف، جعل و تکرار داده‌ها وجود ندارد. عملیات کنترل جامعیت باید به‌صورت مداوم توسط سامانه‌هایی که از خدمات پردازش جریان داده استفاده می‌کنند انجام شود. با توجه به اینکه پدافند غیرعامل[2] مجموعه اقدامات دائمی است که انجام می‌شود تا در صورت بروز جنگ، خسارت‌های احتمالی به حداقل میزان خود برسد، می‌توان گفت کنترل جامعیت یکی از فرآیندهایی است که در پدافند غیرعامل بایستی مد نظر قرار گیرند.

به‌عنوان مثال در یک سامانه جریان داده که بر روی جریان داده‌های دریافتی از رادارهای کشور عمل می‌کند، دشمن ممکن است پس از نفوذ به کانال‌های ارتباطی و بدون اطلاع از محتویات داده‌ها، برخی از رکوردهای جریان داده را حذف و از ارسال آنها به مرکز فرماندهی جلوگیری کند. در این محیط دشمن ممکن است تعدادی رکورد جعلی از فضای کشور را برای مرکز فرماندهی ارسال کند. همچنین برای بی‌خبر نگاه داشتن فرماندهی، ممکن است اطلاعات قبلی شبکه را به‌عنوان اطلاعات فعلی برای او ارسال کند.

شکل (۱) یک شبکه نمونه راداری را نمایش می‌دهد. در این ساختار جریان‌های داده برای سرور مرکز فرماندهی ارسال شده و کاربران پرس و جوهای خود را در سرور ثبت می‌نمایند. سرور پرس و جوهای ثبت شده را اجرا و پاسخ آنها در قالب یک مجموعه رکورد و یا یک جریان داده برای کاربران ارسال می‌کند. در این ساختار، کاربران به جریان داده اصلی دسترسی ندارد. به منظور کسب اطمینان از جامعیت پاسخ‌های دریافتی، با توجه به کانال‌های نامطمئن، کاربر باید روش‌هایی برای کنترل صحت، تمامیت و تازگی نتایج در اختیار داشته باشد [۸ و۷].

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.




**شکل ۱.** نمونه‌ای از یک شبکه پردازش جریان داده نظامی

تمامیت پاسخ‌ها به این معناست که مجموعه پاسخ بدون هیچ کم و کاستی به‌طور کامل توسط کاربر دریافت شده و رکوردی توسط دشمن از این مجموعه حذف نشده است. از طرف دیگر، سامانه‌های مدیریت جریان داده در مواقعی که با افزایش بار ورودی مواجه می‌شوند، برخی از رکوردهای ورودی را برای کاهش بار حذف می‌کنند [۱۰ و۹]. این مسئله، چالشی‌ترین مسئله موجود در کنترل تمامیت پاسخ‌هاست.

مسئله اصلی در این تحقیق ارائه راه‌حلی جهت کنترل جامعیت پاسخ‌های دریافتی از کانال‌های نامطمئن است. در معماری ما، کاربران به جریان داده اصلی دسترسی ندارد و پرس و جوهای انتخاب را برای سرور ارسال می‌کند. سرور در اختیار کاربر نیست بنابراین کنترل جامعیت باید بدون نیاز به تغییر در سرور انجام شود. در این روش، مالک جریان داده یک جریان داده ساختگی را با جریان داده واقعی تلفیق کرده برای سرور ارسال می‌کند. کاربر که از نحوه تولید جریان داده ساختگی مطلع است، پرس و جوهای خود را روی جریان داده ساختگی اجرا نموده و نتایج به‌دست آمده را با رکوردهای ساختگی دریافتی از سرور مقایسه می‌نماید. روش ارائه شده، ضمن برخورداری از دقت قابل قبول، سربار بسیار پایینی داشته و قابلیت تطبیق با دقت مورد نظر کاربر را دارد. از اصلی‌ترین چالش‌های این روش مسئله هم‌زمانی مالک جریان داده در موعد تولید رکوردهای ساختگی، دنباله رکوردهای ساختگی، مقایسه نتایج دریافتی با نتایج مورد انتظار با کمترین سربار ممکن و حداکثر دقت است.

مهمترین نوآوری‌های این تحقیق عبارتند از :

- ارائه یک معماری برای کنترل جامعیت در سامانه‌های مدیریت جریان داده
- مدل‌سازی روش ارائه شده بر مبنای مدل‌سازی احتمالاتی و تطبیق مناسب نتایج مدل‌سازی با نتایج ارزیابی عملی

[1] Data Stream Management System
[2] Passive Defense

- مدل‌سازی و فرموله کردن سطح (دقت) کنترل جامعیت بر اساس نتایج دریافتی

## ۲. بیان مسئله و فرضیات اولیه

در یک سامانه مدیریت جریان داده کاربران باید با استفاده از مکانیزم‌های قابل اعتماد از جامعیت پاسخ‌های دریافتی از سرور اطمینان حاصل کنند. برای تایید جامعیت داده‌ها، کاربر باید از موارد زیر مطمئن شود :

- تمامیت: از مجموعه پاسخ مورد انتظار برای پرس و جوی ارسالی 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.

فرض‌های اولیه این سامانه عبارتند از :

۱- اجرای پرس و جوها به‌وسیله سرور به صورت یک جعبه سیاه در نظر گرفته شده است.

۲- کانال ارتباطی ناامن است بنابراین می‌تواند منبع ایجاد تهدید باشد.

۳- در سامانه، تعدادی منبع جریان داده و تعدادی کاربر وجود دارند.

۴- رکوردها بین مالک‌های جریان داده، سرور و کاربران به‌صورت رمز شده منتقل می‌شوند.

۵- کاربران امکان ثبت پرس و جوهای پیوسته[1] به فرم زیر را دارند:

```
SELECT *
FROM Sp
WHERE Condition(A1, A2, A3,… ,An)
WINDOW SIZE n SLIDE EVERY t   (t>=n)
```

در این پرس و جوها

$S_p$ : جریان داده دریافتی از مالک داده‌ی p

$(A_1, A_2, A_3, \dots, A_n)$: خصیصه‌های جریان داده‌ی $S_p$

Condition $(A_1, A_2, A_3, \dots, A_n)$: هر ترکیب شرطی روی خواص $S_p$

بر اساس فرض‌های فوق، مسئله اصلی در این تحقیق ارائه روشی جدید جهت کنترل جامعیت پاسخ‌های دریافتی از یک سرور مدیریت جریان داده در یک محیط انتقال نامطمئن، بدون نیاز به تغییر در سرور و کانال‌های ارتباطی است.

## ۳. روش ارائه شده

در روش ما، مالک جریان داده به همراه رکوردهای واقعی، تعدادی رکورد ساختگی را نیز ارسال می‌کند. کاربران از نحوه تولید رکوردهای ساختگی مطلع هستند و می‌توانند این‌گونه رکوردها را به طور همزمان با مالک جریان داده ایجاد کنند. رکوردها به‌صورت رمز شده منتقل می‌شوند و بنابراین، دشمن امکان تفکیک و شناسایی رکوردهای واقعی و ساختگی را ندارد. سرور پرس و جوهای دریافتی را روی تمامی رکوردها (واقعی و ساختگی) اجرا کرده و پاسخ‌ها را برای کاربر ارسال می‌کند.

با توجه به اینکه کاربران از نحوه تولید رکوردهای ساختگی مطلع بوده و امکان تفکیک آنها از رکوردهای اصلی را دارند، بنابراین پس از تفکیک و مقایسه آنها با رکوردهای قابل انتظار، در رابطه با جامعیت پاسخ دریافتی قضاوت می‌کنند.

### ۳-۱. معماری سیستم

در شکل (۲) معماری سیستم مورد نظر نمایش داده شده است. برای دست‌یابی به اختفاء داده‌ها، تمامی رکوردها قبل از ارسال، رمزنگاری می‌شوند. بنابراین سرور پرس و جوها را روی داده‌های رمز شده اجرا می‌کند. تاکنون روش‌های مختلفی برای اجرای پرس و جوها روی داده‌های رمز شده ارائه شده‌اند[۱۳-۱۱]. بدیهی است شروط قابل اعمال در پرس و جوها، وابستگی کامل به‌روش رمزنگاری انتخاب شده داشته و از حیطه این تحقیق خارج است.

مطابق شکل، یک بخش مشترک در مالک جریان داده[2](DSO) و کاربر برای تولید رکوردهای ساختگی[3](FR) وجود دارد. این بخش بر اساس سطح جامعیت[4] (IL) مورد انتظار کاربر، اقدام به تولید رکوردهای ساختگی به‌صورت یک جریان داده نموده و این جریان داده ساختگی[5](FDS) پس از تلفیق با جریان داده واقعی[6](RDS) و رمزگذاری به سرور ارسال می‌شود. بر مبنای معماری فوق، فرضیات اولیه زیر را در نظر گرفته‌ایم :

۱- کاربران و مالک‌های جریان داده قبل از شروع عملیات، اطلاعاتی را به عنوان توافقات اولیه از طریق یک کانال امن تبادل می‌کنند.

۲- همه کاربران و منابع جریان داده از یک مکانیزم یکسان برای تولید جریان داده ساختگی استفاده می‌نمایند.

۳- به تمامی رکوردهای جریان داده یک سرآیند به صورت زیر اضافه می‌کنیم:

$$\begin{aligned} &\forall r \in FDS: H = Hash(a_1|a_2|\dots|a_n) \\ &\forall r \in (RDS): H = Hash(a_1|a_2|\dots|a_n) + 1 \end{aligned} \quad (۱)$$

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

[1] Continous

[2] Data Steam Owner
[3] Fake Record
[4] Integrity Level
[5] Fake Data Stream
[6] Real Data Stream

۴- سرور بر اساس نرخ جریان داده در زمان‌های کاری مختلف مجبور به کاهش بار ورودی است ولی حداکثر نرخ کاهش بار[1] (LSR) قبلاً توسط سرور و کاربر در قالب یک توافقنامه سطح سرویس[2](SLA) تعیین می‌شود. بر اساس همین توافقنامه، نرخ تولید جریان داده ساختگی[3] (FRR) نیز تعیین می‌گردد.

در این معماری کاربر پس از دریافت جریان داده پاسخ، رکوردهای ساختگی را از نتایج دریافتی حذف نموده و به عنوان نتایج نهایی به بهره بردار ارسال می‌کند. از طرف دیگر یک پردازش دائمی، رکوردهای ساختگی دریافتی را با رکوردهای مورد انتظار مقایسه نموده و در رابطه به تمامیت پاسخ‌ها اعلام نظر می‌کند.

بدیهی است چنانچه رکوردهای ساختگی در نتایج دریافتی وجود نداشته باشند یک وضعیت مشکوک به حمله رخ داده است. این وضعیت مشکوک به حمله است چون ممکن است سرور برای کاهش بار ورودی، برخی از داده‌های ورودی را حذف نموده باشد.

برای پیاده سازی معماری فوق با دو چالش اساسی زیر روبرو هستیم :

۱- تولید FDS با شرایط زیر :
- همزمانی DSO و کاربران در تولید رکوردهای ساختگی
- تعیین مقادیر فیلدهای رکوردهای ساختگی به طوری که در اکثر پرس و جوهای کاربران تعداد قابل قبولی رکورد ساختگی وجود داشته باشند.

۲- کنترل جامعیت (بررسی وجود رکوردهای ساختگی در پاسخ‌ها) توسط کاربر با حداقل سربار ممکن.

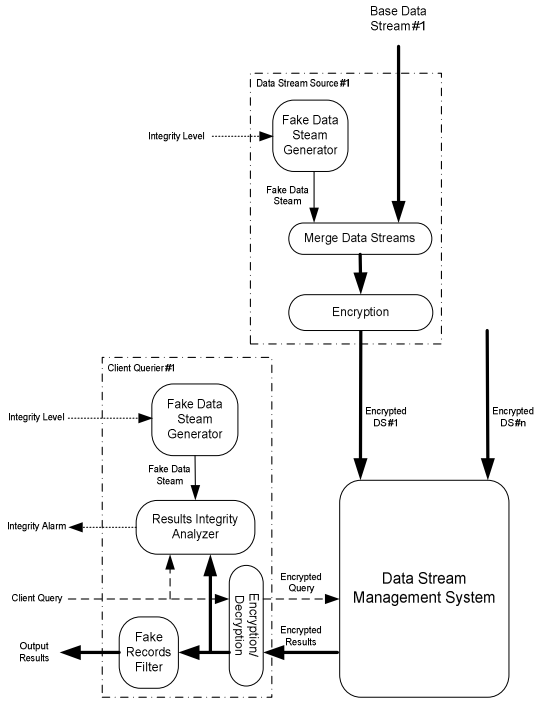
در ادامه این تحقیق راه حل‌هایی برای اجرای موارد فوق ارائه نموده‌ایم.

## ۳-۲. تولید جریان داده ساختگی

همان‌طور که قبلاً اشاره نمودیم، برای تولید جریان داده ساختگی باید دو مسئله اساسی زیر را مورد توجه قرار دهیم:

۱- موعد تولید رکوردهای ساختگی: در سامانه‌های جریان داده، رکوردها به‌صورت متوالی برای سرور ارسال می‌شوند، بنابراین محل قرارگیری رکوردهای ساختگی در جریان داده باید به‌طور دقیق توسط مالک جریان داده و کاربر تعیین شود.

۲- نحوه تولید رکوردهای ساختگی: با توجه به اینکه عملیات کنترل جامعیت پاسخ‌ها بر اساس کنترل وجود رکوردهای ساختگی در پاسخ‌های دریافتی از DSMS انجام می‌شود، این رکوردها باید به‌گونه‌ای ساخته شوند که تعداد هرچه بیشتری از آنها در محدوده پرس و جوهای کاربران قرار گیرند.




**شکل ۲.** معماری کلی سیستم

### ۳-۲-۱. موعد تولید رکوردهای ساختگی

برای دست‌یابی به همزمانی بین مالک جریان داده و کاربر، باید موعد تولید رکوردهای ساختگی بر اساس یک تابع معین انجام شود. در این تحقیق، یک الگوریتم مبتنی بر یک تابع تولید اعداد تصادفی برای تعیین موعد تولید رکوردهای ساختگی ارائه شده است. در این الگوریتم، مقدار seed اولیه و نوع تابع تولید اعداد تصادفی به‌عنوان کلید در نظر گرفته شده و قبلاً از طریق یک کانال امن بین مالک جریان داده و کاربر تبادل می‌شود.

فرآیند کلی تعیین موعد تولید رکوردهای ساختگی به‌صورت زیر است:

۱- DSO به تمامی رکوردهای تولید شده (اعم از واقعی و ساختگی) یک شماره سریال (#RS) اضافه می‌نماید. این فیلد از نوع عدد صحیح است و تأثیر چندانی در حجم رکوردها ندارد.

۲- DSO جریان داده نهایی را در پنجره‌های متوالی به نام پنجره عمومی[4] (GW) با اندازه‌ای نسبتاً بزرگ دسته‌بندی می‌کند. این پنجره‌ها برای کم کردن سربار محاسباتی کاربران در هنگام کنترل جامعیت و همزمانی مورد استفاده قرار می‌گیرند. DSO به اولین رکورد در هر GW مقدار #RS صفر اختصاص می‌دهد. طول این پنجره‌ها($L_{GW}$) یکی از پارامترهای اولیه سامانه است.

[1] Load Shedding Ratio
[2] Service Level Agreement
[3] Fake Records Ratio
[4] General Window

۳- مقدار $L_{GW}$ بسیار بزرگتر از پنجره پرس و جوهای کاربران (n) انتخاب می‌شود ($L_{GW}>>n$) تا هر GW تعدادی قابل توجهی از پنجره‌های جستجو را شامل شود. به این ترتیب احتمال ارسال تعدادی رکوردها پاسخ برای کاربر در هر پنجره عمومی بسیار بالاست.

۴- با توجه به اندازه بزرگ پنجره‌های عمومی، فرآیند تعیین اندیس رکوردهای ساختگی در این پنجره‌ها سربار محاسباتی و حافظه قابل توجهی دارد. برای کاهش این سربار، پنجره عمومی را به تعداد $C_p$ پنجره‌های کوچک‌تر با عنوان پنجره جزئی (PW) تقسیم نموده و محل رکوردهای را ساختگی بر اساس این پنجره‌ها تعیین می‌کنیم.

**الف- تعیین موعد رکوردهای ساختگی توسط DSO**

برای تعیین موعد رکوردهای ساختگی توسط DSO در هر پنجره عمومی به صورت زیر عمل می‌کنیم :

۱- با توجه به نرخ تولید رکوردهای ساختگی (FRR) و طول پنجره جزئی($L_{PW}$)تعداد رکوردهای ساختگی در هر پنجره جزئی ($C_f$) محاسبه می‌شود.

۲- با استفاده از تابع تولید اعداد تصادفی(RNG) انتخاب شده و Seed اولیه پنجره عمومی($S_{init}$) یک مجموعه اندیس با $C_f$ عضو غیرتکراری بین ۱و $L_{PW}$ تولید می‌شوند. این اندیس‌ها محل‌های قرار گرفتن رکوردهای ساختگی در پنجره جزئی اول هستند.

۳- برای پنجره‌های جزئی بعدی، آخرین اندیس ایجاد شده در پنجره قبلی به عنوان Seed در نظر گرفته شده و فرآیند قبلی تکرار می‌شود.

۴- برای اولین پنجره جزئی هر GW، الگوریتم فوق با مقدار اولیه $S_i=S_{init}$ اجرا می‌شود و برای پنجره‌های بعدی $S_i$ برابر مقدار Seed دریافتی از مرحله قبل خواهد بود.

جدول (۱) نمونه‌ای از توافقات اولیه بین مالک جریان داده و کاربران را نمایش می‌دهد. این جدول در شروع عملیات سامانه مورد توافق مالک جریان داده و کاربر قرار گرفته و از طریق یک کانال امن منتقل می‌شود.

**جدول۱.** توافقات اولیه مالک جریان داده و کاربران برای تولید جریان داده ساختگی

| RNG ($f_i$, $S_{init}$) | LSR (درصد) | FRR (درصد) | $L_{PW}$ (رکورد) | $L_{GW}$ (رکورد) | TR (تا , از) |
|---|---|---|---|---|---|
| ($f_1$, $s_1$) | 10 | 20 | 5000 | 100000 | (00:00:01, 08:00:00) |
| ($f_2$, $s_2$) | 15 | 25 | 2000 | 50000 | (08:00:01, 14:00:00) |
| ($f_2$, $s_3$) | 5 | 15 | 5000 | 60000 | (14:00:01, 00:00:00) |

**ب- تعیین موعد رکوردهای ساختگی توسط کاربران**

کاربر جدول (۱) را در اختیار دارد ولی برای رعایت همزمانی با مالک جریان داده، باید مقدار Seed اولیه پنجره جزئی جاری ($S_i$) را در اختیار داشته باشد. مقدار $S_i$ نیز با استفاده از $S_{init}$ قابل محاسبه است. برای تعیین Si پنجره جاری، کاربر با استفاده #RS اولین رکورد دریافتی شماره پنجره جزئی جاری را محاسبه می‌نماید ($k = \lfloor RI_r/L_{PW} \rfloor$). اکنون الگوریتم Estimate Server QW Position را به تعداد k-1 مرتبه (با شروع از $S_{init}$) اجرا می‌نماید. به این ترتیب، $S_i$ پنجره جاری که مالک جریان داده در آن قرار دارد، در اختیار کاربر است و کاربر می‌تواند به صورت همزمان با مالک جریان داده موعد رکوردهای ساختگی را تعیین کند.

### ۳-۲-۲. نحوه تولید رکوردهای ساختگی

مطابق معماری ارائه شده در شکل (۲)، رکوردهای ساختگی باید توسط مالک جریان داده در قالب یک جریان داده ساختگی (FDS) ایجاد شده و بر اساس موعدهای تعیین شده با جریان داده واقعی (RDS) ادغام شوند. از طرف دیگر کاربر نیز باید همین جریان داده ساختگی را به صورت هماهنگ و همزمان با مالک جریان داده ایجاد نماید. بنابراین برای تولید جریان داده ساختگی بایستی به دو مسئله زیر توجه نماییم:

۱- رکوردهای ساختگی باید در یک توالی یکسان توسط مالک جریان داده و کاربران ایجاد شوند. ما این مسئله را به عنوان "هماهنگی مالک جریان داده و کاربر در تولید جریان داده ساختگی" نامیده‌ایم و در زیر راه حل‌های مشخصی برای آن ارائه نموده‌ایم.

۲- رکوردهای ساختگی باید به‌نحوی تولید شوند که احتمال وجود آنها در نتایج پرس و جوهای کاربران حداکثر باشد. ما این مسئله را تحت عنوان "پوشش حداکثری جریان داده ساختگی در پرس و جوهای کاربران" مورد بررسی قرار داده و در ادامه راه حل‌های متنوعی برای آن پیشنهاد نموده‌ایم.

**الف- هماهنگی مالک جریان داده و کاربر در تولید FDS**

**روش ۱:** اولین روش برای تولید یکسان دنباله رکوردها توسط مالک جریان داده و کاربر، استفاده از یک مجموعه رکورد پیش ساخته و ذخیره شده است. با توجه به اینکه موعد ارسال رکوردهای ساختگی مستقل از محتویات رکوردها تعیین می‌شود، بنابراین از این مجموعه رکوردهای ذخیره شده تنها برای رعایت توالی در تولید رکوردهای ساختگی بهره‌برداری می‌شود.

این روش سربار پردازشی نسبتاً پایینی به مالک جریان داده و کاربر تحمیل می‌کند. چنانچه اندازه پنجره عمومی و نرخ تولید رکوردهای ساختگی بزرگ انتخاب شوند، تعداد رکوردهای ساختگی که باید

توسط دو طرف ذخیره شوند افزایش یافته و در نتیجه فضای مورد نیاز برای ذخیره‌سازی آنها افزایش می‌یابد. این مسئله علاوه بر سربار ذخیره‌سازی بالا، حجم اطلاعاتی انتقالی از طریق کانال امن بین مالک جریان داده و کاربر را به شدت افزایش می‌دهد.

**روش ۲:** برای رفع مشکلات فوق از یک تابع معین برای تولید جریان داده ساختگی استفاده نموده‌ایم. در این روش به‌جای ذخیره‌سازی مجموعه رکوردهای ساختگی، این رکوردها به‌طور هماهنگ بر مبنای یک تابع معین تولید می‌شوند. مقدار اولیه این تابع یکی از پارامترهایی است که توسط مالک جریان داده و کاربر مورد توافق قرار می‌گیرد. برای تولید یک رکورد ساختگی به ازای هر فیلد از رکوردهای اطلاعاتی، باید یک تابع تخمین و یک مقدار اولیه مورد توافق قرار گیرد. این روش سربار ذخیره‌سازی کمتری نسبت به روش قبلی دارد ولی مهمترین مسئله‌ای که در این روش با آن مواجه هستیم، مسئله سربار محاسباتی آن است. اگر نرخ تولید رکوردهای ساختگی بزرگ انتخاب شود در نتیجه مالک جریان داده و کاربر باید منابع پردازشی بیشتری برای تولید جریان داده ساختگی در نظر بگیرند.

### ب- پوشش حداکثری جریان داده ساختگی در پرس و جوهای کاربران

با توجه به اینکه تمامیت پاسخ‌ها با استفاده از کنترل رکوردهای ساختگی موجود در آنها انجام می‌شود، بنابراین هر چه تعداد بیشتری از رکوردهای ساختگی در محدوده پرس و جوی کاربران قرار گیرد، با دقت بالاتری می‌توان عملیات کنترل جامعیت را انجام داد. برای تولید رکوردهای ساختگی، می‌توان از روش‌های مورد استفاده در شبیه‌سازی و تولید مجموعه رکوردهای تستی که برای آزمایش نرم‌افزارها مورد استفاده قرار می‌گیرند بهره برد. این روش‌ها عموماً بر مبنای استفاده از یک تابع توزیع احتمال که بر اساس تحلیل‌های آماری از رکوردهای واقعی به دست آمده عمل می‌کنند. تاکنون توابع توزیع مختلفی جهت کاربردهای مختلف ارائه شده‌اند. به عنوان مثال برای شبیه‌سازی اندازه شهرها، طول کلمات و فرکانس تکرار کلمات در یک متن از تابع توزیع زیپفی[1] استفاده می‌شود. خطاهای اندازه‌گیری معمولاً با تابع گوسی تخمین زده می‌شوند و فاصله زمانی بین خرابی‌های سامانه‌ها با تابع توزیع پواسون یا نمایی (منفی) تخمین زده می‌شوند[۱۴]. یک روش ساده برای تولید رکوردها بر مبنای هیستوگرام توزیع مقادیر دامنه آنها ارائه شده است[۱۵].

به عنوان مثال، رکوردهای ایجاد شده توسط سامانه‌هایی مثل ترافیک شبکه‌ها یا دسترسی به صفحات وب که با حجم زیادی از داده و جریان‌های داده‌ای وابسته به زمان روبرو هستند، بر اساس مدل خودتشابهی[2] شبیه‌سازی می‌شوند. در این مدل که بر مبنای ایده فرکتال[3] طراحی شده است، تابع توزیع چندین نقطه مرکزی دارد که از لحاظ عملکرد شبیه به هم هستند ولی دارای احتمالات مختلف می‌باشند. به عنوان مثال، چنانچه ۸۰ درصد ترافیک از یک نقطه مشخص از شبکه عبور کند، در ۲۰ درصد باقیمانده، مجدداً ۸۰ درصد همان ویژگی را دارند و این روند قابل تکرار ادامه پیدا می‌کند.

تا کنون تولید جریان‌های داده که برای شبیه‌سازی دسترسی به صفحات وب و همچنین ارتباطات تلفنی تمرکز دارند، به خوبی مورد بررسی قرار گرفته و تابع توزیع مناسب برای هر دو مورد ارائه شده است[۱۷ و۱۶].

مثال دیگری از نحوه تولید رکوردهای ساختگی در[۱۴] ارائه شده است. در این تحقیق، یک روش تولید جریان داده چند بعدی معرفی شده است. در این روش، بر مبنای دامنه مقادیر فیلدها و توزیع احتمالات آنها، یک ساختار سلسله مراتبی(درختی) ایجاد شده و رکوردهای ساختگی بر اساس هرم مذکور ایجاد می‌شوند.

انتخاب تابع مناسب برای تولید رکوردهای ساختگی کاملاً وابسته به نوع جریان داده واقعی است. هر یک از توابعی که در این بخش معرفی شده‌اند در یک یا حداکثر چند حوزه کاربردی قابل استفاده می‌باشند. ما در این تحقیق سعی در ارائه یک روش جدید برای تولید رکوردهای ساختگی نداریم بلکه از روش‌های موجود که قبلاً توسط سایر محققین ارائه شده‌اند، استفاده نموده‌ایم.

## ۴. کنترل جامعیت پاسخ‌ها

کاربر از نحوه تولید رکوردهای ساختگی اطلاع دارد، بنابراین می‌تواند جریان داده ساختگی همانند مالک جریان داده ایجاد نماید. به این ترتیب، کاربر با اجرای پرس و جوی خود روی جریان داده ساختگی، رکوردهای ساختگی که انتظار می‌رود در پاسخ دریافتی از سرور وجود داشته باشند را تعیین می‌کند. اکنون می‌تواند با مقایسه نتایج به دست آمده با نتایج دریافتی از سرور، تمامیت پاسخ دریافتی را بررسی نماید.

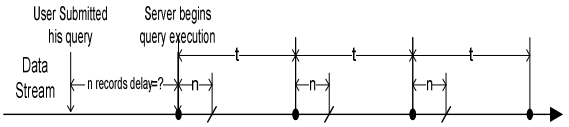
برای پیاده‌سازی فرآیند فوق با چالش‌های متفاوتی مواجه هستیم که مهمترین آنها عبارتند از :

۱- بر اساس توافقات اولیه سرور باید حداکثر پس از یک تأخیر زمانی مشخص، پرس و جوی درخواستی کاربر را اجرا کند. با این حال کاربر نمی‌تواند این تأخیر را به دقت تعیین کند. به این ترتیب کاربر نمی‌تواند موقعیت دقیق پنجره پرس و جو را در جریان داده تعیین کند (شکل(۳)). بنابراین نمی‌تواند پرس و جوی خود را روی جریان داده ساختگی به طور همزمان با سرور اجرا کند.

[1] Zipfian

[2] Self-Similar

[3] Fractal




**شکل ۳.** تأخیر سرور در اجرای پرس و جوی کاربر

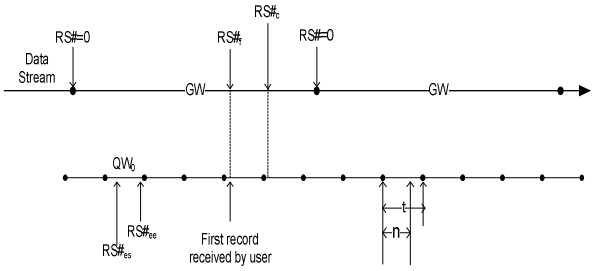
۲- سرور با توجه به نرخ جریان داده دریافتی ممکن است مجبور به کاهش بار ورودی شده و برخی از رکوردهای ورودی را حذف کند. هرچند حداکثر نرخ کاهش بار قبلاً توسط سرور و کاربر مورد توافق قرار گرفته است و سرور اجازه کاهش بار بیش از حد توافق شده را ندارد، ولی این مسئله در کنترل نتایج توسط کاربر موثر است و باید به آن توجه کرد.

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

### ۴-۱. تخمین موقعیت پنجره پرس و جو توسط کاربران

در این تحقیق ما برای تخمین موقعیت پنجره‌های پرس و جو یک الگوریتم تکرار شونده ارائه نموده‌ایم. در این الگوریتم موقعیت پنجره پرس و جوی اول تخمین زده شده و موقعیت بقیه پنجره‌ها با توجه به موقعیت این پنجره محاسبه می‌شوند.

منظور از پنجره پرس و جوی اول، اولین پنجره پرس و جویی است که در پنجره عمومی جاری آغاز گردیده و به پایان رسیده است ($QW_0$ در شکل(۴)). فرآیند اجرایی الگوریتم مورد نظر در شکل(۵) ارائه شده است.




**شکل ۴.** موقعیت پنجره‌های پرس و جو در پنجره‌های عمومی

چنانچه رکوردهای پاسخ به صورت یکسان در محدوده پنجره پرس و جو توزیع شده باشند، این الگوریتم پس از چند تکرار محدوده دقیق $QW_0$ را تخمین می‌زند.

**نکته ۱:** مطابق فرضیات اولیه، پنجره‌های عمومی را به اندازه‌ای بزرگ انتخاب می‌کنیم که در هر یک از آنها (هر $L_{GW}$ رکورد از جریان داده) حداقل یک پاسخ برای کاربر ارسال شود و فاصله دو رکورد پاسخ کمتر از اندازه یک پنجره عمومی است. بنابراین اگر $RS\#_c<RS\#_{c-1}$ باشد، به این معنی است پنجره عمومی جدیدی آغاز شده است. در این حالت، باید محل $QW_0$ را در پنجره عمومی جدید تعیین کنیم.

```
1  voidEstimateServerQWPosition(int RS#){
2 int RS#L=(RS# mod t) + t  // Map RS# to first QW
3      if (RS# < prevRS#) {  // GW is changed
4        RS#es = RS#es - (LGW mod t);
5        RS#ee = RS#ee - (LGW mod t);
6        if (RS#es< t) { RS#es += t;  RS#ee += t; }
7      } else
8       if ((RS#ee-RS#L>n | RS#L-RS#es>n)& !(RS#es==0 &
RS#ee==0)){// some records are deleted or freshness error
9         cer++;
10       if (cer > threshold) {
11          Alarm("Freshness Error");
12          RS#ee = RS#L; RS#es = RS#L;
13          cer = 0;
14       }
15       } else {
16          cer = 0;
17             if (RS#es==0 & RS#ee==0){ // First Received
Record
18             RS#es = RS#L; RS#ee = RS#L;
19          } else {
20             RS#es = Min(RS#L, RS#es);
21             RS#ee = Max(RS#L, RS#ee);
22             }
23      }
24      RS#cs = RS#es - ((n - (RS#ee - RS#es)) / 2);
25      RS#ce = RS#cs + n - 1;
26      prevRS# = RS#;
27} // End of EstimateServerQWPositions
```

**شکل ۵.** الگوریتم تخمین موقعیت پنجره پرس و جوی سرور توسط کاربران

**نکته ۲:** چنانچه به دلایلی مثل قطع موقتی شبکه، تعدادی GW بدون ارسال نتایج رد شوند، الگوریتم فوق بدون در نظر گرفتن شرایط شروع به اعلام هشدارهای پی در پی به کاربر می‌کند. در این حالت کاربر باید به صورت دستی الگوریتم را راه‌اندازی مجدد نماید.

برای آنکه نیاز به شروع مجدد الگوریتم به صورت دستی توسط کاربر مرتفع شود، یک مکانیزم شروع مجدد الگوریتم بر مبنای تعداد دفعات بروز خطا به الگوریتم اضافه نموده‌ایم. در این مکانیزم پس از هر مرحله برخورد با خطا، تعداد دفعات بروز خطا را با یک حد آستانه مقایسه نموده و در صورت عبور از حد آستانه محاسبه مقادیر تخمینی، پنجره پرس و جوی سرور را مجدداً آغاز می‌کنیم.

### ۴-۲. توجه به کاهش بار توسط سرور

مطابق فرضیات اولیه، مالک جریان داده، کاربر و سرور حداکثر کاهش بار مجاز توسط سرور را طبق یک زمان‌بندی مشخص توافق می‌نمایند. برای کنترل صحت نتایج، کاربر باید تعداد رکوردهای ساختگی دریافتی را با تعداد رکوردهای ساختگی قابل انتظار مقایسه نماید. در این صورت :

$C_{rf}$: تعداد رکوردهای ساختگی دریافتی

$C_{ef}$: تعداد رکوردهای ساختگی مورد انتظار

اگر $C_{ef} > C_{rf}$ *(1-LSR) باشد، یک حمله رخ داده است یعنی برخی از رکوردهای مورد انتظار در پاسخ برای کاربر ارسال نشده‌اند. برای آنکه کاربر امکان کنترل نتایج دریافتی را داشته باشد، باید نرخ جریان داده ساختگی، بزرگتر از نرخ کاهش بار سرور باشد. این مسئله در مثال جدول(۱) رعایت شده است. به عنوان مثال، در صورتی نرخ کاهش بار توافق شده بین کاربر و سرور برابر ۱۰٪ در یک ساعت معین باشد، در صورتی که الگوریتم کنترل جامعیت در سمت کاربر خطایی تا سطح ۱۰٪ را اعلام کند، به دلیل وجود کاهش بار در این ساعت، این اعلام منجر به اعلام هشدار حمله به کاربر نخواهد شد و در صورتی‌که حذف بیش از ۱۰٪ باشد، به کاربر هشدار داده خواهد شد.

### ۴-۳. الگوریتم نهایی کنترل جامعیت

شکل (۶) شمای کلی الگوریتم کنترل جامعیت را نمایش می‌دهد. همان‌طور که در شکل ملاحظه می‌شود، با استفاده از رکوردهای دریافتی، پنجره پرس و جوی سرور تخمین زده شده و جریان داده ساختگی تولید می‌شود. در قدم بعدی، پرس و جوی کاربر بر FDS اعمال شده و نتایج برای بخش "کنترل جامعیت" ارسال می‌شود.

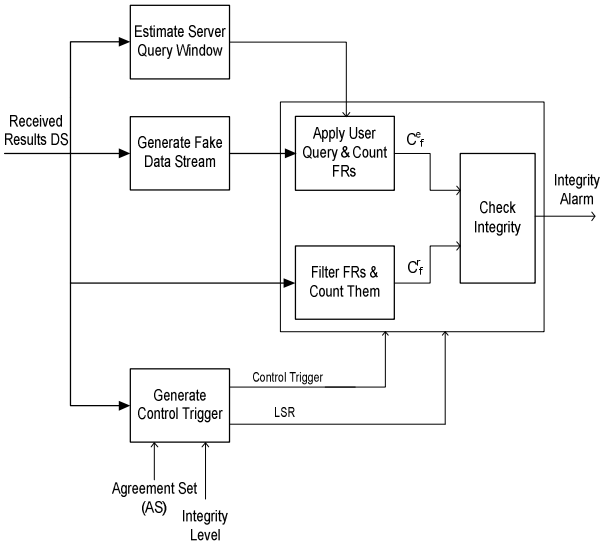
همزمان، رکوردهای ساختگی موجود در نتایج جدا شده و برای بخش "کنترل جامعیت" ارسال می‌شود. جامعیت نتایج بایستی بر اساس مجموعه توافقات کاربران و DSO کنترل شود. در بخش بعدی نشان داده‌ایم که با دریافت تعداد مشخصی از رکوردهای پاسخ، می‌توانیم با دقت قابل قبولی از جامعیت نتایج مطمئن شویم.

بنابراین، با توجه به سطح جامعیت (IL) مورد انتظار، موعد کنترل جامعیت توسط بخش "تولید ماشه کنترل" تعیین می‌شود. بخش "تولید ماشه کنترل" حداقل تعداد رکوردهای دریافتی (p) برای اطمینان از کشف خطا را تعیین نموده و پس از دریافت هر p رکورد، یک فرمان "ماشه کنترل" را شلیک می‌کند. در نهایت، بخش "کنترل جامعیت" به سادگی تعداد رکوردهای ساختگی دریافتی را با تعداد رکوردهای ساختگی مورد انتظار (با توجه به LSR) مقایسه کرده و خطاها را اعلام می‌کند.

در ضمن خطاهای مربوط به تازگی نتایج در بخش تخمین پنجره پرس و جوی سرور و خطاهای مربوط به صحت نتایج در بخش جداسازی، رکوردهای ساختگی کشف شده و از طریق بخش کنترل جامعیت به کاربر اعلام می‌شوند.

### ۴-۴. سطح جامعیت

در روش ما، سطح جامعیت مورد انتظار بر اساس میزان دقت سامانه در کشف حملات نقض جامعیت تعریف می‌شود. به این منظور، ابتدا یک مدل احتمالاتی جهت تخمین دقت سامانه ارائه نموده و سپس بر مبنای مدل مذکور، روش دست‌یابی به دقت مورد نظر را تعریف نموده‌ایم.




**شکل ۶.** شمای الگوریتم کنترل جامعیت توسط کاربر

#### ۴-۴-۱. مدل احتمالاتی ارزیابی سامانه

فرض کنید میانگین تعداد رکوردهای پاسخ به پرس و جوهای کاربر در یک بازه زمانی مشخص، برابر N باشد. رکوردهای ساختگی به‌طور یکنواخت در جریان داده توزیع شده‌اند بنابراین، میانگین تعداد رکوردهای واقعی در هر N رکورد پاسخ برابر (1 - FRR) * N است. اگر سرور فقط رکوردهای واقعی را حذف کند، کاربر نمی‌تواند حذف این رکوردها را تشخیص دهد. احتمال وقوع این وضعیت برابر است با :

$$p = \prod_{i=0}^{N(1-FRR)-1} \frac{N(1-FRR)-i}{N-i} \qquad (۲)$$

تمامی جملات حاصل‌ضرب فوق کوچکتر از یک هستند و بزرگترین آنها برابر با $\frac{N*(1-FRR)}{N}$ است، بنابراین :

$$p < (1-FRR)^{N*(1-FRR)} \qquad (۳)$$

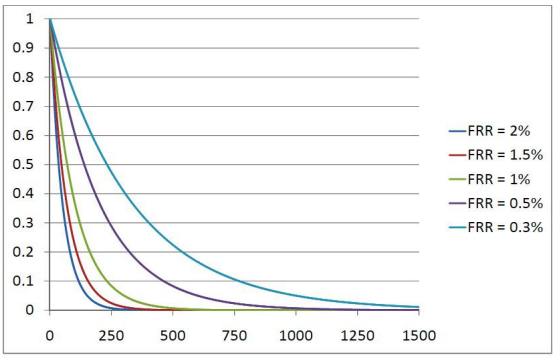
در DSMS، نرخ تولید رکوردهای جریان داده بالاست و به همین دلیل عموماً در بازه‌های زمانی کوتاه، تعداد قابل توجهی رکورد پاسخ برای کاربران ارسال می‌شود. بنابراین، بر اساس نامعادله فوق، می‌توان از FRRهای کوچک برای دست‌یابی به دقت‌های قابل قبول استفاده نمود. شکل (۶) تغییرات p بر مبنای N را برای چند FRR مختلف نمایش می‌دهد.

همانطور که در شکل ملاحظه می‌شود، چنانچه مقدار FRR را ۰/۵٪ در نظر بگیریم، با میانگین ۱۰۰۰ رکورد دریافتی، احتمال عدم کشف حمله تقریباً صفر است. برای در نظر گرفتن کاهش بار در این مدل، کافیست به‌جای FRR، FRR-LSR را در فرمول جایگزین نماییم.

به این ترتیب:

$$p < (1-(FRR-LSR))^{N*(1-(FRR-LSR))} \qquad (۴)$$

در ادامه این بخش، برای رعایت اختصار مقدار LSR صفر در نظر گرفته شده است.




**شکل ۷.** احتمال عدم کشف حمله حذف رکوردهای پاسخ توسط سرور

### ۵-۴. تعیین سطح جامعیت

در مدل ما، سطح جامعیت سامانه بر اساس دقت سامانه در کشف حملات تعریف می‌گردد. در بخش قبلی، یک مدل احتمالاتی برای تخمین احتمال عدم کشف حملات، بر مبنای میانگین رکوردهای دریافتی و FRR ارائه نمودیم. در این قسمت، یک روش برای تعیین دقت سامانه بر مبنای این دو پارامتر ارائه می‌نماییم.

اگر α دقت مورد انتظار کاربر باشد، سیستم باید با خطای کمتر از 1-α حملات را کشف کند. بنابراین:

$$1 - a < (1 - FRR)^{N(1-FRR)} \qquad (۵)$$

از آنجا که FRR قبلاً توسط DSO و کاربران مورد توافق قرار می‌گیرد، بنابراین برای دست‌یابی به دقت αکافیست مقدار N را طوری انتخاب کنیم که نامعادله بالا برقرار باشد. بنابراین داریم :

$$N > \frac{Ln(1-a)}{(1-FRR)*Ln(1-FRR)} \qquad (۶)$$

به عنوان مثال، برای دست‌یابی به دقت ۹۹ درصد با فرض FRR برابر ۲ درصد، کافیست کاربر به ازای دریافت حداقل ۲۳۳ رکورد، تمامیت کنترل کند(تعداد رکوردهای ساختگی دریافتی با تعداد قابل انتظار مقایسه شود) که یک نتیجه بسیار جالب است.

بخش "تولید ماشه کنترل" در شکل(۶)، با توجه به FRR و LSR و سطح جامعیت مورد انتظار (در اینجا α)، حداقل تعداد رکوردهای دریافتی(N) برای دست‌یابی به دقت مورد نظر را بر مبنای روش فوق محاسبه نموده و به ازای دریافت رکوردهای مورد نظر، یک "ماشه کنترل" را شلیک می‌کند.

در روش دیگر، با ثابت نگه داشتن N، نرخ رکوردهای ساختگی(FRR) را بر اساس سطح جامعیت مورد انتظار(α)، تعیین می‌کنیم. به این صورت که در فواصل زمانی مختلف بر اساس حداقل تعداد رکوردهای دریافتی(که به تعبیر دیگر تأخیر سامانه در کنترل جامعیت نتایج است)، نرخ رکوردهای تکراری تعیین می‌شود. در این مقاله از روش اول برای تولید ماشه استفاده نموده‌ایم.

## ۵. پیشینه تحقیق

روش‌های کنترل جامعیت در سامانه‌های مدیریت جریان داده را می‌توان در دو گروه اصلی مبتنی بر ساختارهای تصدیقی و احتمالاتی دسته‌بندی نمود. در این قسمت به بررسی مختصر فعالیت‌های محققین در دو گروه فوق می‌پردازیم.

### ۵-۱. روش‌های مبتنی بر ساختارهای تصدیقی

در این روش‌ها، رکوردها قبل از ارسال برای سرور در یک ساختار تصدیقی سازماندهی شده و پس از امضاء برای سرور ارسال می‌شود. سرور از این ساختار برای اجرای پرس و جوها استفاده می‌نماید و کاربر با استفاده از آن، جامعیت نتایج را ارزیابی می‌کند. مهمترین ساختارهای تصدیقی که تاکنون معرفی شده‌اند، ساختارهای زنجیره‌ای و درخت درهم‌سازی مرکل (MH-Tree)[1] [۱۹] هستند. ساختارهای زنجیره‌ای به علت سربار ذخیره‌سازی بالا، در سامانه‌های مدیریت جریان داده قابل استفاده نمی‌باشند. درخت مرکل برای کنترل جامعیت در معماری برون‌سپاری پایگاه داده[2] توسط محققین مختلفی مورد استفاده قرار گرفته است[۲۴-۲۰].

تاکنون دو پروژه تحقیقاتی در زمینه کنترل جامعیت در DSMS با استفاده از MH-Tree در دانشگاه علم و فناوری هنگ کنگ[3] [۲۵] و دانشگاه بوستون[4] با همکاری آزمایشگاه‌های تحقیقاتی‌ای تی اند تی[5] [۸] انجام شده است.

در [۲۵] مدلی توسعه یافته از MH-Tree سازماندهی شده بر مبنای B+-Treeها ارائه شده است. این مدل، تنها پرس و جوهای محدوده‌ای را پشتیبانی می‌کند و مبتنی بر اجرای پرس و جوهای طولانی مدت طراحی شده است، ولی امکان پشتیبانی از پرس و جوهای یک‌بار مصرف را نیز دارد. مهمترین مزیت این مدل، پشتیبانی همزمان آن از صحت و تمامیت پاسخ‌ها و مهمترین ایراد آن نیز نیاز به به‌روزرسانی درخت ارسالی قبلی برای کاربر به ازای دریافت هر رکورد جدید است. در این مدل، رکوردها بر مبنای دامنه آنها تقسیم‌بندی می‌شوند و ساختار درختی رکوردها بر اساس آن ایجاد می‌شود. انتخاب محدوده‌های دامنه برای ساختن درخت، چالش اساسی در این مدل است. اگر محدوده‌ها کوچک انتخاب شوند، درخت بزرگ می‌شود و اگر محدوده‌ها بزرگ انتخاب شوند، لازم است سرور تعداد قابل توجهی به‌روزرسانی پاسخ برای کاربران ارسال کند.

برای رفع مشکل ارسال مجدد درخت پاسخ‌ها، یک الگوریتم در [۸] ارائه شده است. در این الگوریتم، جریان داده ورودی به تعدادی درخت (بر مبنای بازه‌های زمانی) تقسیم می‌شود. مهمترین ایراد این

[1] Merkle Hash Tree
[2] Database Outsourcing
[3] Hong Kong University of Science and Technology
[4] Boston University
[5] AT & T Research Lab

روش، ایجاد یک تأخیر پردازشی در ساخت درخت‌ها و سپس ارسال پاسخ به کاربر و نیز سربار پردازشی نسبتاً بالای کاربر در کنترل نتایج است. از طرف دیگر، این روش تنها به پرس و جوهای یک بعدی پاسخ داده و تعداد زیادی رکورد اضافی برای کاربر ارسال می‌کند. همچنین در این روش هزینه اجرای اولین مرحله الگوریتم بالاست.

### ۵-۲. روش‌های احتمالاتی

در این روش‌ها، بخشی از پردازش سرور مجدداً توسط کاربر انجام شده و جهت کنترل جامعیت پاسخ‌ها مورد استفاده قرار می‌گیرد. با توجه به تحقیقات ما، تنها فعالیتی که تاکنون با استفاده از روش‌های احتمالاتی در سامانه مدیریت جریان داده برای کنترل جامعیت پاسخ‌ها انجام شده است، روشی است که در [۲۶] معرفی شده است. این روش که تحت عنوان خلاصه‌های تصادفی چندجمله‌ای‌های یکه[1] (PIRS) نام‌گذاری شده است، برای تحلیل داده‌های ترافیک شبکه مورد استفاده قرار گرفته است.

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، پرس و جوهای شمارشی و تجمیعی را پشتیبانی کرده و برای محاسبه خلاصه مورد انتظار از سربار محاسباتی و حافظه قابل قبولی برخوردار است.

کار در زمینه ارائه روش‌های احتمالاتی برای کنترل جامعیت نتایج دریافتی در یک محیط پردازش جریان داده ناامن، هنوز در ابتدای راه است و تنها روش‌های اولیه‌ای برای استفاده در محیط DAS ارائه شده‌اند که از آن جمله می‌توان به [۲۸ و۲۷ و۱۵ و۷] اشاره نمود.

## ۶. ارزیابی و تفسیر نتایج

برای شبیه‌سازی مالک داده، سرور و کاربر از سه کامپیوتر با پردازنده Pentium IV 2.0 GHz، حافظه اصلی ۲ گیگابایت و هارددیسک با ظرفیت ۲۰۰ گیگابایت استفاده کرده‌ایم. همچنین این سه ماشین از طریق یک شبکه محلی(100MBps) به یکدیگر متصل شدند.

از یک پردازنده پرس و جو در هسته سرور استفاده کردیم. تمامی الگوریتم‌ها در این آزمایش را به زبان جاوا و با استفاده از مجموعه (JDK/JRE 1.6) پیاده‌سازی نموده‌ایم.

برای ارزیابی، از جریان داده تولیدی بر اساس فراخوانی صفحات وب مربوط به ۷۶۲ کاربر متفاوت در دانشگاه بوستون آمریکا از تاریخ ۲۱ نوامبر ۱۹۹۴ تا تاریخ ۸ می ۱۹۹۵ که حدود ۲,۰۰۰,۰۰۰ رکورد را شامل می‌شود[۱۸]، استفاده نموده‌ایم.

از آنجایی که تا کنون هیچ سیستم مشابهی مورد آزمایش قرار نگرفته است، امکان مقایسه این سیستم با سیستم‌های مشابه وجود ندارد؛ از این رو، در این بخش به ارزیابی رفتار و کارآمدی سیستم خود پرداخته‌ایم.

### ۶-۱. مدل حمله

در حملات مربوط به تمامیت، ممکن است دشمن از کانال ارتباطی بین سرور و کاربران داده‌هایی را حذف کند. در این صورت، قسمتی از پاسخ برای کاربر ارسال نمی‌شود(حذف از پاسخ). همچنین ممکن است سرور قسمت‌هایی از داده‌های ارسالی از سنسورها به سرور را حذف نماید. در این صورت نیز بخشی از پاسخ‌ها برای کاربر ارسال نمی‌شود(حذف از جریان داده ورودی).

حذف از پاسخ: با توجه به اینکه حذف رکوردها توسط دشمن، مستقل از محل آنها در جریان داده است، بنابراین، رکوردهای حذف شده به صورت یکنواخت در سطح مجموعه جواب پخش شده‌اند. اگر دشمن $\beta$ درصد از رکوردهای پاسخ را حذف نماید، تابع $f(x)$ را به عنوان تابع تصمیم حذف، به صورت زیر تعریف می‌کنیم:

$$\forall r \in Results\ Stream,\ x = Random(r), 0 \le x \le 1,$$

$$f(x) = \begin{cases} True, & x < \frac{\beta}{100} \\ False, & x \ge \frac{\beta}{100} \end{cases} \quad (۷)$$

حذف از جریان داده ورودی: اگر $\gamma$ را به عنوان نرخ حذف و تابع $g(x)$ را به عنوان تابع تصمیم حذف آن در نظر بگیریم، خواهیم داشت:

$$\forall r' \in Input\ Stream,\ x = Random(r'), 0 \le x \le 1,$$

$$g(x) = \begin{cases} True, & x < \frac{\gamma}{100} \\ False, & x \ge \frac{\gamma}{100} \end{cases} \quad (۸)$$

#### ۶-۱-۱. تشخیص حذف

کاربر می‌تواند حذف رکوردهای ساختگی را تشخیص بدهد. بنابراین، تشخیص حمله تا زمان حذف رکوردهای ساختگی توسط دشمن دچار تأخیر می‌شود. نتایج ارزیابی و مدل‌سازی سیستم به‌خوبی نشان می‌دهد که این تأخیر بسیار کم و در حد قابل قبولی است. برای ارزیابی عملکرد سیستم، ما دو مقیاس "فاصله تشخیص" و "نرخ تشخیص حمله" را به صورت زیر تعریف می‌نماییم.

تعریف فاصله تشخیص: به میانگین تعداد رکوردهای موجود از جایگاه حذف رکورد تا جایگاه تشخیص آن فاصله تشخیص گفته می‌شود. این مقیاس نشان دهنده میانگین تأخیر در تشخیص حملات تمامیت می‌باشد. بنابراین:

$$DD = \frac{1}{n}\sum(l-k) \quad (۹)$$

واضح است که تأخیر کمتر نشان دهنده سرعت بیشتر در تشخیص حملات می‌باشد.

[1] Polynomial Identity Random Synopses

تعریف نرخ تشخیص حمله: نرخ "تعداد هشدارهای تولید شده توسط سیستم" بر روی "هشدارهای مورد انتظار" بر حسب درصد نرخ تشخیص حمله گفته می‌شود. این مقیاس نشان‌دهنده درصد موفقیت سیستم در تشخیص حملات می‌باشد. بنابراین:

$$ADR = \frac{Count(alarms)}{Count(expected\ alarm)} * 100 \qquad (۱۰)$$

واضح است که $ADR \geq 0$ به معنای تشخیص حمله می‌باشد و هر چه این مقدار بیشتر باشد، کارایی بهتری از تشخیص حمله را مشخص می‌کند.

### ۶-۲. موقعیت پنجره پرس و جو

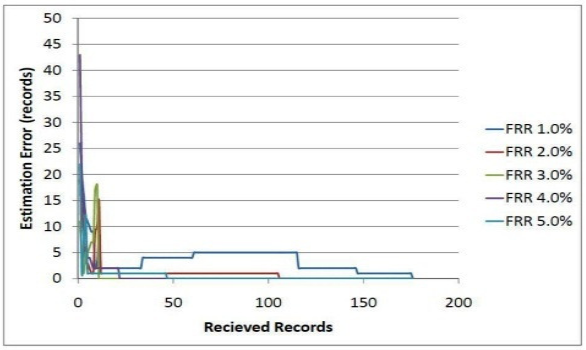
کاربران می‌باید از موقعیت پنجره پرس و جوی سرور مطلع باشند تا بتوانند آن را بر روی رکوردهای ساختگی که تولید نموده‌اند، اعمال نمایند. این تخمین یک گام کلیدی در مرحله تشخیص حملات تمامیت به شمار می‌آید. آزمایش‌های ما نشان می‌دهد که این الگوریتم بعد از دریافت تعداد متناهی از رکوردهای پاسخ، به صفر می‌رسد. شکل(۸) خطای الگوریتم تخمین موقعیت پنجره پرس و جو را بر مبنای تعداد رکوردهای دریافتی نشان می‌دهد.

### ۶-۳. تشخیص حملات تمامیت

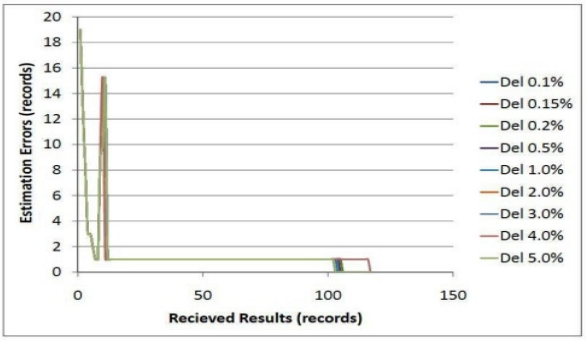
شکل(۹) فاصله تشخیص در حملات تمامیت را بر اساس نرخ تولید داده‌های ساختگی متفاوت نشان می‌دهد. همانطور که در شکل نشان داده شده است، روش ما به‌سرعت حملات تمامیت را تشخیص می‌دهد. بیشترین حد تأخیر زیاد نیست و با افزایش نرخ تولید داده‌های ساختگی، تأخیر در تشخیص حمله در نقاط ابتدایی کاهش می‌یابد. همچنین با افزایش نرخ حذف رکوردها، سرعت تشخیص حمله حذف بیشتر شده و حمله سریع‌تر تشخیص داده می‌شود.

شکل(۱۰– الف) نرخ تشخیص حملات را بر اساس حذف رکوردهای پاسخ با درصدهای متفاوت برای نرخ‌های مختلف تولید رکوردهای ساختگی نشان می‌دهد. همانطور که در شکل(۱۰–الف) مشاهده می‌شود مقدار منحنی به ازای تمامی حالات حذف در محدوده ۰٪-۲٪ یک مقدار مثبت است، بنابراین تمامی حملات حتی با نرخ‌های حذف کم و نرخ تولید داده‌های ساختگی اندک تشخیص داده می‌شوند. شکل(۱۰–ب) نیز رفتار سامانه را در هنگامی که کاهش بار بیش از حد مورد توافق انجام شده است، نشان می‌دهد. رفتار سیستم در این حالت کاملاً مشابه رفتار قبلی است.

شکل (۱۱) نرخ تشخیص حمله را با توجه به اندازه‌های متفاوت پنجره ارزیابی و بر اساس نرخ‌های متفاوت تولید داده‌های ساختگی را نشان می‌دهد. همانطور که در شکل (۱۱) نشان داده شده است، با افزایش اندازه پنجره ارزیابی نرخ تشخیص حمله با شیب نسبتاً زیاد افزایش می‌یابد.

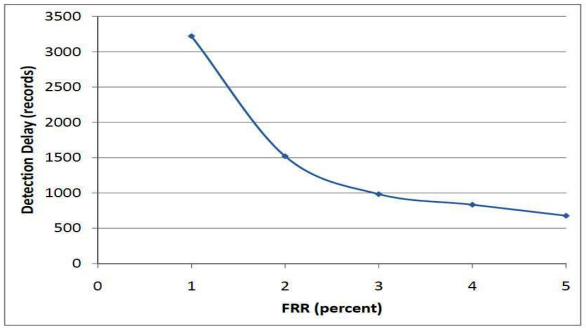



الف) بدون حذف رکوردها

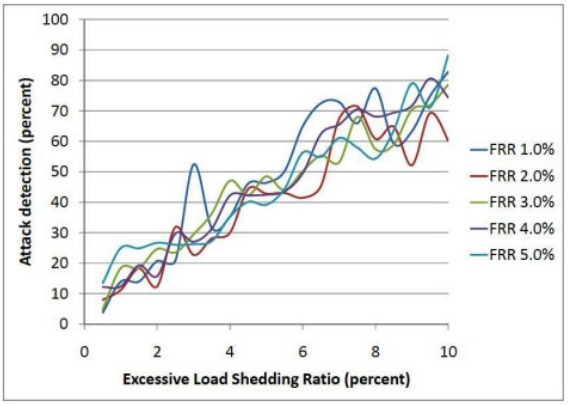



ب) با حذف رکوردها

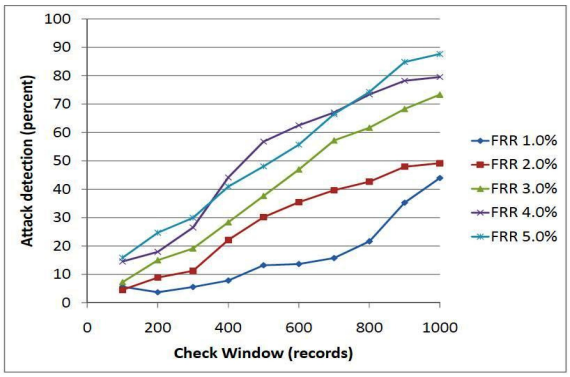
**شکل ۸.** خطای الگوریتم تخمین موقعیت پنجره پرس‌وجوی سرور توسط کاربران




**شکل ۹.** تأخیر تشخیص (بر حسب رکورد) با در نظر گرفتن نرخ تولید رکوردهای ساختگی متفاوت

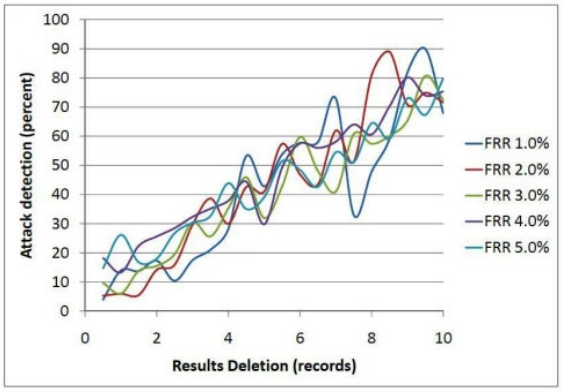



الف) نتایج حذف




ب) نتایج کاهش با بیش از حد توافق

**شکل۱۰.** تشخیص حملات حذف با در نظر گرفتن نرخ‌های متفاوت تولید داده‌های ساختگی




**شکل۱۱.** نرخ تشخیص حمله با توجه به اندازه‌های متفاوت پنجره ارزیابی و بر اساس نرخ‌های متفاوت تولید داده‌های ساختگی

### ۶-۴. بیانیه امنیت

در این بخش به اثبات درستی روش ارائه شده می‌پردازیم. صحت رکوردها با استفاده از سرآیند رکوردها که از درهم ریختگی محتویات آنها توسط DSO تولید می‌شود، ارزیابی می‌شود. رکوردها به صورت رمز شده بین مالک جریان داده، سرور و کاربران تبادل می‌شود. دشمن برای اضافه کردن یک رکورد جعلی، باید علاوه بر حدس زدن مقدار سرآیند رکورد، کلید رمزنگاری رکوردها را نیز حدس بزند. بنابراین، امنیت الگوریتم کنترل صحت وابسته به تابع درهم ریختگی و همچنین الگوریتم رمزنگاری رکوردها است.

برای اثبات درستی الگوریتم کنترل تمامیت دو قضیه و یک نکته به‌صورت زیر ارائه نموده‌ایم.

**قضیه ۱:** اگر تعداد رکوردهای پاسخ به سمت بی‌نهایت میل کند، حد احتمال خطای الگوریتم تخمین پنجره جستجوی سرور توسط کاربر به سمت صفر میل می‌کند.

**اثبات:** موقعیت رکوردهای پاسخ در جریان داده مستقل از شرط پرس و جوی کاربر و موقعیت پنجره پرس و جوی سرور در جریان داده است. اگر اندازه پنجره پرس و جو برابر n رکورد باشد، احتمال وجود رکوردهای پاسخ در هر یک از محل‌های یک پنجره پرس و جو 1/n است. پس از دریافت N رکورد توسط کاربر، احتمال وجود حداقل یک رکورد در موقعیت k پنجره پرس و جو عبارتست از:

$$p_k = \sum_{i=0}^{N}(\frac{n-1}{n})^i * \frac{1}{n} \qquad (۱۱)$$

با توجه به اینکه $\frac{n-1}{n} < 1$ است داریم :

$$\lim_{N\to\infty} p_k = \frac{\frac{1}{n}}{1-\frac{n-1}{n}} = 1 \qquad (۱۲)$$

به این ترتیب، با زیاد شدن تعداد رکوردهای پاسخ، احتمال دریافت حداقل یک رکورد پاسخ توسط کاربر در محل k پنجره پرس و جو (از جمله محل‌های ابتدایی و انتهایی آن) به سمت یک میل می‌کند. در الگوریتم تخمین پنجره پرس و جوی سرور، با دریافت حداقل یک رکورد در ابتدا و انتهای پنجره پرس و جو، موقعیت پنجره با دقت کامل تعیین می‌شود. بنابراین، با افزایش رکوردهای پاسخ احتمال خطای الگوریتم تخمین پنجره پرس و جو به سمت صفر میل می‌کند.

**قضیه ۲:** احتمال خطای الگوریتم کنترل تمامیت با افزایش تعداد رکوردهای دریافتی به سمت صفر میل می‌کند.

**اثبات:** بر اساس قضیه۱، با افزایش تعداد رکوردهای پاسخ، خطای الگوریتم تخمین پنجره پرس و جو به سمت صفر میل می‌کند. بنابراین خطای تخمین پنجره‌های پرس و جو تأثیری در دقت الگوریتم کنترل تمامیت ندارد.

فرض کنید تعداد رکوردهای پاسخ برابر N و رکوردهای ساختگی در مجموعه رکوردهای پاسخ به صورت یکنواخت توزیع شده باشد. در بدترین شرایط، دشمن باید فقط رکوردهای واقعی را حذف کند، در این صورت، کاربر به هیچ وجه از حذف این رکوردها مطلع نمی‌شود. احتمال وقوع این وضعیت عبارت است از :

$$p = \prod_{i=0}^{N(1-RRF)-1} \frac{N(1-RRF)-i}{N-i} \qquad (۱۳)$$

با توجه به اینکه بزرگترین جمله در عبارت فوق برابر با $\frac{N*(1-RRF)}{N}$ است، بنابراین :

$$p < (1\text{-RRF})^{N*(1\text{-RRF})} \quad (۱۴)$$

با توجه به اینکه 1>(1-FRR) است، بنابراین :

$$\lim_{N\to\infty} p = 0 \quad (۱۵)$$

به این ترتیب با افزایش رکوردهای پاسخ، احتمال موفقیت دشمن در حملات تمامیت به سمت صفر میل می‌کند.

## ۷. نتیجه‌گیری

در این مقاله، به ارائه یک روش جامع برای کنترل جامعیت در سیستم‌های جریان داده برون‌سپاری شده پرداخته‌ایم. در این روش، به همراه رکوردهای اصلی، رکوردهایی با عنوان ساختگی برای سرور ارسال می‌شود. جایگاه این رکوردها با استفاده از یک تابع تولید اعداد شبه تصادفی تعیین می‌گردد. در ابتدا مالک داده از طریق یک کانال امن، تابع مورد نظر به همراه مقدار اولیه آن را برای کاربر ارسال می‌نماید. در حین اجرای پرس و جوها، کاربر پس از دریافت پاسخ پرس و جوی خود از سرور، رکوردهای ساختگی را از طریق تابع مذکور تولید نموده و پس از اعمال پرس و جوی خود بر روی رکوردهای ساختگی، از طریق مقایسه پاسخ به دست آمده با رکوردهای دریافت شده از سرور، از جامعیت پاسخ‌های دریافتی اطمینان حاصل می‌کند. همزمانی کاربر و سرور از مهمترین چالش-های موجود در این مقاله بود که راه‌حل و الگوریتم آن ارائه شد. علاوه بر آن، معماری روش ارائه شده و همچنین الگوریتم‌های مورد نیاز برای کنترل جامعیت، در بخش‌های گذشته معرفی گردید. ارزیابی و نتایج به دست آمده نشان دهنده کارایی بسیار خوب روش ارائه شده می‌باشد.

در آینده قصد داریم به مواردی مثل کم کردن سربار کاربر در تولید جریان ساختگی و اعمال پرس و جوها بر آن و همچنین اثبات امنیت روش ارائه شده به یک روش دیگر پرداخته و نتایج تحقیق را به صورت یک یا چند مقاله علمی ارائه نماییم.

## ۸. مراجع


[1]. Arasu, A.; Babcock, B.; Babu, S.; Datar, M.; Ito, K.; Motwani, R.; Nishizawa, I.; Srivastava, U.; Thomas, D.; Varma, R.; Widom, J. "STREAM: The Stanford Stream Data Manager."; International Conference on Management of Data, San Diego, California 2003, 655 - 665.

[2]. Abadi, D. J.; Carney, D.; Etintemel, U. C.; Cherniack, M.; Convey, C.; Lee, S.; Stonebraker, M.; Tatbul, N.; Zdonik, S. "Aurora: A New Model and Architecture for Data Stream Management."; VLDB, 2003.

[3]. Chandrasekaran, S.; Cooper, O.; Deshpande, A.; Franklin, M.; Hellerstein, J.; Hong, W.; Krishnamurthy, S.; Madden, S.; Raman, V.; Reiss, F. M. "Telegraph CQ: Continuous Data Flow."; Processing for an Uncertain World, CIDR 2003.

[4]. Chen, J.; Dewitt, D. J.; Tian, F.; Wang, Y. "Niagara CQ: A Scalable Continuous Query System for Internet Databases."; ACM SIGMOD, 2000, 379-390.

[5]. Schmidt, S.; Legler, T.; Schär, S.; Lehner, W. "Robust Real-Time Query Processing with QStream."; VLDB, 2005, 1299-1301.

[6]. Abadi, D. J.; Ahmad, Y.; Balazinska, M.; Cherniack, M.; Hwang, J. H.; Lindner, W.; Maskey, A. S.; Rasin, E.; Ryvkina, E.; Tatbul, N.; Xing, Y.; Zdonik, S. "The Design of the Borealis Stream Processing Engine."; CIDR, 2005, 277-289.

[7]. Xie, M.; Wang, H.; Yin, J.; Meng, X. "Providing Freshness Guarantees for Outsourced Databases"; EDBT'08, 2008.

[8]. Li, F.; Yi, K.; Hadjieleftheriou, M.; Kollios, G. "Proof-Infused Streams: Enabling Authentication of Sliding Window Queries on Streams."; VLDB, 2007, 143-155.

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

[10]. Tatbul, N.; Detintemel, U.; Zdonik, S.; Cherniack, M.; Stonebraker, M. "Load Shedding in a Data Stream Manager."; VLDB , 2003.

[11]. Brinkman, R. "Searching in Encrypted Data."; Ph.D Thesis, Department of Information and Computer Science, University of Twente, 2007.

[12]. Dong, C.; Russello, G.; Dulay, N. "Shared and Searchable Encrypted Data for Untrusted Servers."; IFIP WG 11.3 working Conference on Data and Applications Security, 2008.

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

[14]. Wang, X.; Liu, H.; Er, D. "HIDS: a Multifunctional Generator of Hierarchical Data Streams."; SIGMIS Database, 2009.

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

[16]. Dill, S.; Kumar, R.; Mccurley, K. S.; Rajagopalan, S.; Sivakumar, D.; Tomkins, A. "Self-Similarity in the Web."; ACM Trans. Internet Technology, 2002, 2, 205-223.

[17]. Fiorini, P. M. "Modeling Telecommunication Systems with Self-Similar Data Traffic."; Ph.D. Thesis. The University of Connecticut, 1998.

[18]. Carlos A. B.; Cunha, A.; Mark E. Crovella Boston University Computer Science Department, spanning the Timeframe of 21 November 1994 through 8 May 1995.

[19]. Merkle, R. C. "A Certified Digital Signature."; Springer-Verlag, City, 1990, 218-238.

[20]. Pang, H. H.; Tan, K. L. "Authenticating Query Results in Edge Computing."; ICDE, 2004, 560.

[21]. Lazaridis, I.; Mehrotra, S. "Progressive Approximate Aggregate Queries with a Multi-Resolution Tree Structure."; International Conference on Management of Data 2001, 401-412.

[22]. Li, F.; Hadjieleftheriou, M.; Kollios, G.; Reyzin, L. "Authenticated Index Sturctures for Aggregation Queries."; ACM Trans. Information System Security 13, 2010, 13-32.

[23]. Li, F.; Hadjieleftheriou, M.; Kollios, G.; Reyzin, L. "Dynamic Authenticated Index Structures for Outsourced Databases."; International Conference on Management of Data 2006, 121–132.

[24]. Martel, C.; Nuckolls, G.; Devanbu, P.; Gertz, M.; Kwong, A.; Stubblebine, S. G. "A General Model for Authenticated Data Structures."; Algorithmica 2004, 39, 21-41.

[25]. Papadopoulos, S.; Yang, Y.; Papadias, D. "Continuous Authentication on Relational Streams."; VLDB Journal 2010, 19, 161-180.

[26]. Yi, K.; Li, F.; Hadjieleftheriou, M.; Kollios, G.; Srivastava, D. "Randomized Synopses for Query Assurance on Data Streams."; ICDE, IEEE International Conference on Data Engineering 2008, 416-425.

[27]. S Sion, R. "Query Execution Assurance for Outsourced Databases."; VLDB, 2005, 601-612.

[28]. Wang, H.; Yin, J.; Perng, C. S.; Yu, P. S. "Dual Encryption for Query Integrity Assurance."; ACM Conference on Information and Knowledge Management, 2008.