[آسان داک](http://www.asandoc.com/) (www.Asandoc.com)

**الگوریتم TSFS بهبودیافته برای سرّی‌سازی امن دیتابیس(پایگاه داده)**

چکیده: امنیت دیتابیس تبدیل به طور فزاینده در حوزه اپلیکیشن ها به امری جدی تبدیل شده است. سرّی‌سازی دیتابیس مکانیزمی مهمی برای آن سازی دیتابیس از حملات و دسترسی‌های غیرمجاز است. الگوریتم سرّی‌سازی جابجایی-تعویض-چین خوردگی-انتقال (TSFS) الگوریتم سرّی‌سازی متقارنی است که از سه کلید همراه با تکنیکی مبسوط برای ایجاد امنیت بالا استفاده می‌نماید: این مقوله به بهبود کارایی زمان اجرای جستار به وسیله سرّی‌سازی تنها اطلاعات حساس؛ منجر می‌گردد. لکن تنها صرفاً برای کارکترهای الفبایی کاربرد دارد. این مقاله به توسعه مجموعه اطلاعات الگوریتم سرّی‌سازی TSFS برای کاربردهای ویژه نیز می‌پردازد و به اصلاح تغییر و انتقال با ایجاد بیش از یک عامل مودولار و 16 آرایه ای، به ترتیب جدید به منظور دوری از خطاهای حادث شونده در روند سرّی‌سازی می‌پردازد. نتایج تجربی نشان می‌دهد که الگوریتم سرّی‌سازی TSFS بهبود یافته به قابلیت اجرای الگوریتم استاندارد سرّی‌سازی اطلاعات (DES) و الگوریتم استاندار سرّی‌سازی پیشرفته (AES) در رابطه با زمان اجرای جستارها و اندازه اطلاعات الحاقی؛ کمک می‌نماید.

واژگان کلیدی: سرّی‌سازی، امنیت دیتابیس، جابجایی (ترانهش)، تغییر شکل، چین‌خوردگی و انتقال.

**مقدمه**

توسعه شگرف فناوری و ذخیره‌سازی اطلاعات منجر می‌شود که سازمان‌هایی وابسته به سیستم های دیتابیس به وجود آیند. این سازمان‌ها حجم عظیمی از اطلاعات دیتابیس را در شرایطی امن ذخیره می‌کنند برای اینکه آن‌ها را به گونه‌ای سریع و امن بازیابی نمایند. آسیب اطلاعات می‌تواند حادث شود اگر از سوی حملات و دسترسی‌های غیرمجاز تهدید گردد. در حضور تهدیدات امنیتی، امنیت دیتابیس به یکی از چالش‌های جدی بدل شده است، پیچیدگی الگوریتم سرّی‌سازی باید بیشتر شود و در همان زمان کارایی دیتابیس هم مدنظر قرار گیرد؛ درحالی‌که میزان کارایی هم متأثر نگردد.

مطالعات جستجوگرانه بسیاری در زمینه امنیت دیتابیس یافت می‌شود، برخی از این تحقیقات به کاربردهای سودمندی منجر شده‌اند. همچنین بسیاری از الگوریتم‌های سرّی‌سازی مد نظر بوده‌اند. برخی از آن‌ها رخ نمودند اما توسعه نیافتند؛ یکی از این‌ها الگوریتم سرّی‌سازی جابجایی-تعویض-چین خوردگی-انتقال (TSFS) است. الگوریتم TSFS ارائه گر درجه بالایی از امنیت با استفاده از شماری از ویژگی‌هاست لکن تنها ارقام و گونه‌های الفبایی را پشتیبانی می‌کند که برای حفاظت از گونه‌های مختلف اطلاعات حساس کافی نیست.

این مقاله به ارائه روش سرّی‌سازی امنی می‌پردازد که تنها به سرّی‌سازی اطلاعات بدون استفاده از سخت‌افزاری ویژه می‌انجامد. این امر به تقویت الگوریتم TSFS از طریق گسترش مجموعه اطلاعات الگوریتم سرّی‌سازی TSFS برای کارکترهای ویژه نیز می‌انجامد، و به اصلاح تغییر و انتقال با ایجاد بیش از یک عامل مودولار و 16 آرایه ای به ترتیب جدید به منظور دوری از خطاهای حادثه در روند سرّی‌سازی می‌پردازد. بعلاوه این مقاله به ارائه قیاسی بین الگوریتم تقویت‌شده TSFS (ETSFS) و دو الگوریتم سرّی‌سازی مشهور بهنام الگوریتم استاندارد سرّی‌سازی اطلاعات (DES) و استاندار سرّی‌سازی پیشرفته (AES) پرداخته و به ارزیابی عملکرد آن‌ها در رابطه با زمان انجام جستارها و اطلاعات الحاقی دیتابیس مبادرت می‌نماید.

بخش دیگر این مقاله بدین گونه سازمان می یابد: بخش 2 به مرور کارهای موجود در مورد فنون سرّی‌سازی دیتابیس؛ بخش 3 به معرفی الگوریتم ETSFS و تشریح فرایند آن می‌پردازد درحالی‌که بخش 4 به معرفی نحوه اعمال الگوریتم ETSFS و ساختار پیشنهادی می‌پردازد؛ بخش 5 ارائه مطالعه‌ای قیاسی میان الگوریتم‌ها، ارزیابی عملکردها و گزارش و بحث در مورد نتایج است. در نهایت بخش 6 به جمع‌بندی پیشنهادها برای تحقیقات بیشتر می‌پردازد.

**2- کارهای مرتبط**

به سبب نقش مهمی که تکنیک سرّی‌سازی در ایمن سازی اطلاعات دیتابیس ایفا می‌نماید، بسیاری الگوریتم‌ها با فنون و قابلیت اجرای متفاوت رخ نمودند. بوگانیم و پوچرال یک کارت هوشمند کوچک را پیشنهاد گردند تا از امنیت اطلاعات محافظت کنند در جایی که صاحبان دیتابیس می‌توانند با استفاده از یک ترمینال کلاینت که توسط کارت‌های هوشمند پشتیبانی می‌شود، دسترسی داشته باشند. این راه حل پیشنهادی دربردارنده راه‌حلی امن و کاراست اما پیچیده و گران نیز هست. سرّی‌سازی اطلاعات عملکرد دیتابیس را بسیار متأثر می‌کند چرا که هر باری که یک جستار اجرا می‌شود حجم زیادی از اطلاعات باید سرّی‌سازی گردد. بنابراین پیشنهاد می‌گردد که سرّی‌سازی اطلاعات حساس تنها زمانی ارائه شود که امنیت مورد نیاز است؛ بدون متأثر شدن از اجرا. در الگوریتم‌ها 1، 4 و 5؛ پیشنهاد شده است که تنها وابسته به سرّی‌سازی اطلاعات حساس باشند. کاور و همکاران تکنیکی پیشنهاد نمودند که تنها به سرّی‌سازی داده‌های عددی تنها با یک گونه و درازای معین پرداخته شود. لکن این الگوریتم به پشتیبانی سرّی‌سازی داده‌های کارکتری نمی‌انجامد. آگراوال و همکاران نیز طرح‌واره‌ای در مورد سرّی‌سازی برای اطلاعات عددی دارای ویژگی‌های مهم پیشنهاد کرد که به جستارها و دیگر عملکردهای تطبیقی اجازه می‌داد تا مستقیماً در یک مجموعه سرّی‌سازی شده اطلاعات بدون سرّی‌سازی خودشان به کار گرفته شوند. به علاوه به بررسی مدیریت کلیدی نپرداخته بود.

الگوریتم DES نیز یکی از الگوریتم‌های سرّی‌سازی مشهور است که از کلید متقارن برای متن ساده 64 بیت به متن رمزی 64 بیتی با استفاده از کلید 56 بیتی و 16 چرخه؛ بهره‌برداری نموده است. افزایش شگرفی در بسیاری اپلیکیشن ها ملاحظه می‌گردد؛ گه عمدتاً مربوط به اندازه کلید است که بسیار کوچک است. کارهای جدید نشان می‌دهد که الگوریتم AES به عنوان جایگزینی برای الگوریتم DES به مثابه استانداردی برای سرّی‌سازی اطلاعات. این الگوریتم کلیدی متقارن که برای متن ساده 128 بیت و برای کلید 128، 192 یا 256 بیت را نیاز دارد. درازای کلید تعداد و چرخه‌های الگوریتم را مشخص می‌نماید.

در نهایت مانیوانان و سوجارانی تکنیک سرّی‌سازی دیتابیس کارایی را با استفاده از الگوریتم TSFS پیشنهاد کردند که الگوریتمی کلیدی = متقارن است. ویژگی‌های اصلی آن از فنون ترانهشی و رمزداری استفاده می‌کند که در الگوریتم‌های متقارن مدرن دارای اهمیت است که دارای پخشایش و پریشانی است. همچنین تنها به سرّی‌سازی اطلاعات حساس می‌پردازد که منجر به محدود شدن زمان اضافی برای کاربران سرّی‌سازی و کشف رمز می‌گردد. الگوریتم به تسهیل سه کلید و توسعه آن‌ها در قالب 12 کلیدی جزیی با استفاده از فنون گسترش کلید می‌پردازد تا امنیتی کارا برای دیتابیس فراهم سازد. لکن الگوریتم TSFS تنها برای کارکترهای الفبایی کاربرد داشته و کارکترهای ویژه و نمادین را نمی‌پذیرد. اطلاعات بیشتر در مورد الگوریتم TSFS در مورد 9 ارائه شده است که ارقام متفاوتی را برای کلیدهای سری مبتنی بر الگوریتم TSFS در خلال دیگر الگوریتم‌ها ایجاد می‌نماید تا تأمین امنیت بالا در مورد اطلاعات سری شده را محقق نماید.

**3- الگوریتم TSFS توسعه‌یافته (ETSFS)**

موضوع اصلی این مقاله تقویت الگوریتم TSFS و مطابق آن امنیت بالا برای اطلاعات است درحالی‌که هزینه زمانی اضافی را محدود می‌کند و تنها اطلاعات سری شده مهم را رمزگشایی می‌نماید. الگوریتم ETSFS می‌تواند به سرّی‌سازی اطلاعاتی می‌پردازد که شامل کارکترهای الفبایی از A تا Z گرفته، همه اعداد و این نمادها: (\*, -,., /,:, @ and \_). ETSFS یک الگوریتم متقارن سرّی‌سازی است بدین معنی که هر گونه ترانهش یا فرایندی باید وارونه شدنی بوده و بتواند عملیاتی برعکس را انجام دهد که می‌تواند به لغو اثر منجر شود. کلید همچنین باید دارای ترتیبی وارون باشد.

الگوریتم ETSFS زا 4 تکنیک تغییرش کل شامل ترانهش-تعویض-چین خوردگی-انتقال استفاده می‌نماید. نگاره 1 به نمایش الگوریتم سرّی‌سازی ای می‌پردازد که الگوریتم رمزگشایی آن وارونه کننده الگوریتم سرّی‌سازی است. این بخش به تشریح 4 تکنیک می‌پردازد:

الف) ترانهش

تغییر ترانهش مکان‌یابی عناصر ماتریکسی را با استفاده از ترانهش قطری عوض می‌نماید که ماتریس داده را به صورت زیگزاکی قطری می‌خواند که از گوشه بالایی چپ شروع‌شده و بعد از لایی گذاری آن با \*s آن کمتر از 16 رقم خواهد بود. شکل 2 نمایان گر فرایند ترانهش ای است که زمانی که اطلاعات ورودی 6923@domain.Sa بوده است. شکل 3 نشان‌دهنده الگوریتم ترانهش از جانب سرّی‌سازی است.

ب) تعویض

تکنیک دوم تکنیک انتقال تعویض است. این تکنیک به جابجایی یک عنصر ماتریس داده با با دیگری از طریق اعمال عملکردی معین می‌پردازد. اگر آن عنصر بازتابنده کارکتری الفبایی باشد آن نیز با کارکتری دیگر جایگزین می‌گردد. اگر شامل عدد باشد با عددی دیگر جایگزین می‌شود و اگر سمبل باشد با سمبلی دیگر جابه‌جا می‌گرد.

عملیات سرّی‌سازی برای E برای هر X کوچک داده‌شده بدین گونه است:

*E(x) = (((k1+p) mod M +k2) mod M*

اگر p عنصر ماتریسی ساده باشد، k1 و k2 عناصری کلیدی هستند که موقعیتی برابر با p را دار هستند و M نشانگر اندازه عملکرد مدولار است. الگوریتم ETSFS به اندازه مدول ها سه مقدار می‌دهد بجای اینکه یک مقدار را مانند الگوریتم TSFS بدهد. فرایند تغییر در مورد 1 تشریح شده دارای پریشانی است. پریشانی زمانی رخ می‌دهد که داده از ارقام الفبایی و عددی تشکیل‌شده اندازه مدول برای هر رقم 26 باشد همان طور که در مثال آتی نشان داده می‌شود. اگر یک عامل از داده 4 باشد پس k1=5, k2=5, M = 26, و نتیجه فرایند تغییر 14 است همان طور که مقاله نشان می‌دهد. این نتیجه دارای دو مشکل است: اول اینکه درازای داده تغییر خواهد یافت و اضافه خواهد شد مانند زمانی که اندازه متن ساده 16 رقمی است به عنوان مثال اندازه متن رمزی‌شده 17 رقم خواهد بود. مشکل دوم این است ه از زمانی که عملیات رمزگشایی داده به صورت رقم به رقم انجام شود د راین حالت با هر یک از عناصر منفردا به صورت یک متن رمزدار رفتار می‌کند. در نتیجه اطلاعات رمزگشایی شده متفاوت است با آنچه سرّی‌سازی شده است. بنابراین الگوریتم ETSFS، M را بدن گونه می‌دهد: 26 اگر p الفبایی باشد، 10 اگر رقم باشد و 7 اگر سمبل باشد.

عملیات رمزگشایی D بدین گونه است:

D(E(x)) = (((E(x) – k2) mod M) – k1) mod M

زمانی که بسیاری از زبان‌های برنامه‌نویسی مانند جاوا و C++ با مدولها مانند باقیمانده یک عدد صحیح برخورد می‌کنند برخی نتایج ممکن است دارای علامت منفی بوده و موجب مشکل شوند به طوری که داده‌ای وجود تداشته باشد که دارای علامت منفی باشد. براین اساس یک مرحله بیشتر به الگوریتم ETSFS اضافه می‌شود به منظور تعیین حالتی که نتیجه دربردارنده علامت منفی باشد.

*D(E(x)) = M - |D(E(x))|*

ج) چین‌خوردگی

فعل‌وانفعال پین خوردگی یکی از عناصر ماتریس را با دیگری در داده مشابه جایگزین می‌کند، مثل تا زدن کاغذ. ماتریس داده به طور افقی، عمودی و قطری تا می‌شود. چین‌خوردگی افقی بدین گونه است که ردیف او ل با ردیف آخر جایگزین می‌شود. مورد عمودی بدین شکل است که ستون اول با ستون آخر جابجا می‌شود و حالت قطری بدین نحو است که سلول‌های داخلی مثل سلول بالایی از چپ با سلول پایینی از راست جایگزین می‌شود و بر عکس.

د) انتقال

قسمت آخر الگوریتم مربوط به انتقال است که ارائه گر راهی آسان برای سرّی‌سازی عناصر 16-آرایه‌ای مربوط به ارقام عددی برای تغییر به حروف با دیگری است. هر عامل آرایه شامل بازنمایی رقومی از داده است. هر عدد ظاهر می‌شود زمانی که هر عنصر متناظر آرایه ظاهر گردد. اعداد می‌توانند با هر ترتیبی ظاهر شوند. در فرایند انتقال الگوریتم به جابجایی هر عامل ماتریس داده به واسطه موقعیتش در میان عوامل آرایه می‌پردازد. الگوریتم ETSFS از 4 16-آرایه به جای یک آرایه مانند آنچه TSFS انجام می‌داد؛ استفاده می‌کند. به طور نمونه اگر متن ساده 4 و موقعیتش در آرایه 15 است فرایند انتقال در (1) سپس 15 را بر می‌گرداند؛ به سبب دو مشکلی که در عملیات ترانهش تشریح شد. بنابراین هر گونه را از دیگری تمییز می‌دهد. الگوریتم ETSFS چهار 1-آرایه را استفاده‌کند یکی برای نشانه‌ها؛ اما از آنجا که بازشماری همه نشانه‌ها سخت است الگوریتم ETSFS پیشنهادی تنها دو گونه از نشانه‌ها را در نر می‌گیرد. سمبل های مورد استفاده در ایمیل (-,., @, \_) و سمبل های مورد استفاده در نشانی‌های آی پی (/,:).

فرایند رمزگشایی پیشین به عنوان نتیجه دور نخست الگوریتم ETSFS مورد ملاحظه است. خروجی دور اول به عنوان ورودی دور دوم است و این فرایند تا 12 دور ادامه می‌یابد، به طوری که خروجی این دور متنی رمزی است در دیتابیس ذخیره می‌گردد. برای کلیدهای هر دور دو کید برای رمزگشایی انتخاب می‌شوند. در سری یازی هر دور i یک کلید i را انتخاب می‌کند و کلید i+1 در دور 12 کلیدهای 12 و یک را انتخاب می‌کند. در رمزگشایی کلیدها به ترتیبی معکوس انتخاب می‌شوند.

5- پیاده‌سازی

یک پروژه بر پایه جاوا بر پایه ETSFS الگوریتم اصلاحی و پیاده‌سازی ساخته شده است.پیاده ساری از معماری ای 3 لایه استفاده می‌کند همچنان که در نگاره 10بازنمایی شده است. این سه لایه عملکردها را در سه گونه سیمای داخلی، پردازش و کاربردهای مدیریت اطلاعات جداسازی می‌نماید. معماری چند لایه توسعه‌دهندگان را قادر می‌سازد تا اپلیکیشن هایی معقول و انعطاف‌پذیر ایجاد نمایند. در این مقاله لایه درونی برای ورود و بازیابی اطلاعات از دیتابیس مورد استفاده است. لایه پردازش برای انباشتن داده یا جستار زا لایه سیمای درونی و نیز تکمیل فرایندهای سرّی‌سازی و رمزگشایی برای اعمال جستار بر روی اطلاعات پایه است. این امر منجر به ذخیره‌سازی کلیدها در فایلی جداگانه می‌شود بجای اینکه آن‌ها را در دیتابیس ذخیره کند؛ به جهت افزایش امنیت. در نهایت لایه مدیرت داده اطلاعات را ذخیره می سازد.

بر پایه معماری پیشنهادی ساختار پیاده‌سازی گسترش یافت آن طور که در نگاره 11 نشان داده شده است. این نگاره به تصویرسازی همه طبقات و ارتباطات آن‌ها با یکدیگر می‌پردازد. این طبقات شامل:

1-طبقه اصلی: به طور کلی در آغاز کاربر می تواند به ورود اطلاعاتی که سرّی‌سازی شده‌اند اقدام کند. طبقه اصلی داده را هنگام اجرا از فایل برای حصول نتایج هم ارز می‌خواند. بخش سیمای درونی مسئول اتخاذ داده از فایل و ارسال به بخش ترجمان است که سپس در دیتابیس ذخیره می‌شود. دیگر عملکرد مربوط به سیمای درونی بازیابی اطلاعات از دیتابیس با استفاده از بخش ترجمان است.

2- طبقه ترجمه گر: این مقاله تنها بر روی الگوریتم‌های سرّی‌سازی – رمزگشایی متمرکز است بیش از چگونگی اینکه یک جستار باید ترجمه یا برگردان شود به جستاری که باید بکار گرفته شود جهت دیتابیس سری شده بکار گرفته شود. بنابراین نخش مترجم اطلاعات را از بخش سیمای درونی دریافت می‌کند از طریق روش SaveInDatabase و به ایجاد ارتباط با دیتابیس به منظور بکار گماری جستار درونی در دیتابیس پس از سرّی‌سازی اطلاعات دریافتی؛ مبادرت می‌ورزد. روش دیگر از دو جستار انتخابی خاص برای انتخاب داده استفاده می‌کند : یکی برای بازیابی جدول کامل و دیگری برای انتخاب جستار بسته به موقعیت.

طبقه سرّی‌سازی رمزگشایی: این طبقه مهم‌ترین طبقه برای این کار است. این طبقه از چندین روش برای آماده سازی اندازه و شکل داده که سرّی‌سازی/ رمزگشایی خواهد شد؛ استفاده می‌کند و سپس زا فرایند TSFS را برای سرّی‌سازی/ رمزگشایی داده‌هایی که از کلیدهای ذخیره‌شده استفاده می‌کنند؛ بهره می‌برد؛ سپس نتایج را به طبقه مترجم برای اعمال در جستار برمی‌گرداند.

3- طبقه کلید: طبقه کلید سه کلیدی اولیه را از فایل می‌خواند و بر اساس آن کلیدها را برای ایجاد 12 کلید جزیی گسترش می‌دهد به وسیله انتقال ردیف‌های که در (1) شرح داده شد.

**5-مطالعه تطبیقی**

این تحقیق الگوریتم ETSFS را با DESو AES مقایسه کرده که همگی اپن سورس بوده‌اند . بنابر اعمال و سنجش الگوریتم‌ها این موارد مورد استفاده بوده است

* زبان برنامه‌نویسی: جاوا
* پایه اپلیکیشن: NetBeans IDE 6.9.1
* توسعه: Java Development Kit (JDK) 1.6
* سیستم مدیریت دیتابیس: MySQL Server 5.6
* برنامه دان خارجی جاوا: Connector-java 5.1.23-bin.
* ابزار طراحی دیتابیس بصری: MySQL Workbench 5.2 CE مورد استفاده برای طراحی دیتابیس، مدلینگ و توسعه SQL
* سیستم‌عامل: Windows Vista Home Premium, 32-bit
* سخت‌افزار کامپیوتری: Dell XPS M1330 laptop, Intel(R) Core(TM) 2 Duo CPU T7300 2.00GHz and 3.00 GB for RAM

در این تحقیق برای حصول قیاسی معین و منصفانه بین الگوریتم‌ها، اطلاعاتی یکسان واردشده و عملکردی همسان مسئول دسترسی به دیتابیس مورد استفاده در همه الگوریتم‌ها بوده است. بعلاوه هر الگوریتم توسط این اندازه از داده‌ها آزمون شد:100، 500، 1000، 1500 و 2000 ردیف. برای هر اندازه این تجربه 3 باز تکرار شد، باارزش متوسط برای هر زمانسنج؛ سپس تجمیع شده برای زدودن اثر پردازش کامپیوتری و اطمینان دهی از تقریب ارزش واقعی.

ب)متریک ارزیابی: آزمایش‌های بسیاری بر روی الگوریتم‌ها اعمال شد . دو متریک ارزیابی مورد ملاحظه است: زمان اجرا (دوم) که فرایند سرّی‌سازی/رمزگشایی اتخاذشده و اندازه (کیلوبایت) دیتابیس پس از ذخیره‌سازی داده‌های رمزگشایی شده برای آگاهی از تأثیر فرایند سرّی‌سازی اندازه عمومی دیتابیس. برای سرعت اجرا، سه زمان‌سنج برای هر الگوریتم و سه گونه از جستار استفاده شد .

جانمایی: ما زمان جانمایی داده را محاسبه کردیم تا بفهمیم چقدر زمان لازم است تا عملیات جانمایی در طی فرایند سرّی‌سازی زمان می‌برد. نمونه جستار جانمایی: *INSERT INTO person VALUES*

انتخاب همه: برای تعیین زمان مورد نیاز جستار برای انتخاب همه ردیف‌ها و رمزگشایی فیلدهای سری شده. نمونه‌ای برای جستار انتخاب همه: *SELECT name, phone, mail, job FROM person*.

انتخاب مشروط: برای تثبیت اینکه چه میزان زمام لازم است برای یک جستار دربردارنده سرّی‌سازی داده به طور مشروط ، سپس مقایسه این داده با داده‌های سری شده داخلی دیتابیس برای بازیابی اطلاعات مورد نیاز و بر طبق آن بازیابی فیلد های سری شده از طریق فرایند بازیابی. نمونه‌ای برای جستار انتخاب مشروط: *SELECT name, phone, mail, job FROM person WHERE name = encrypted name*.

ج)نتایج و مباحث

زمان اجرا: نتایج آزمایش در مورد زمان اجرا نشان داده شده است در سه شکل اول. شکل 12 نشان می‌دهد که ارتباط میان زمان اجرای عملیات جانمایی با فرایند سری ساری چقدر است و نیز تعداد چندتایی ها برای هر الگوریتم. شکل 13 نشانگر ارتباط میان زمان سرّی‌سازی در حین جستجوی دیتابیس که از جستار انتخاب همه استفاده می‌کند برای بازیابی جدول کامل و تعداد چندتایی ها. شکل 14 نیز نشان می‌دهد که ارتباط میان رمان سرّی‌سازی و تعداد چندتایی ها برای هر الگوریتم چقدر است. بر پایه نتایج مشخص است که الگوریتم AES بیش‌ترین زمان را برای سرّی‌سازی و الگوریتم ETSFS کمترین زمان را در این زمینه مصرف می‌کند . این بدین معنی است که الگوریتم ETSFS دارای بهترین کارکرد در میان دیگر الگوریتم‌ها در این زمینه است. به عکس الگوریتم DES کمترین زمان را مصرف می‌کند هنگامی که جستارهای انتخاب مورد استفاده باشند. ناپایداری ارتباط ممکن است موجب قرض گرفتن زمان در حین ایجاد ارتباط یا قطع ارتباط با دیتابیس یا عملکردهای فایل باشد . بعلاوه موضوعات مربوط به سخت‌افزار ممکن است سبب ارتباطات ناپایدار باشد. عموماً فرایند سرّی‌سازی زمان بیشتری را از کاربران می‌گیرد به نسبت فرایند رمزگشایی. همچنین اجرای جستار که موجب بازیابی جدول کامل می‌شود زمان حتی کمتری می‌شود به نسبت آن چه وابسته به شرایط است چرا که پیش‌تر چندتایی های بیشتری بازیابی شده است، این چندتایی ها کاربردهای سرّی‌سازی بیشتری را می‌طلبد.

حجم دیتابیس: شکل 15 نشان می‌دهد که ارتباط میان تعداد چندتایی ها و حجم دیتابیس بر پایه کیلوبایت پس از ذخیره‌سازی اطلاعات سری شده با استفاده از سه الگوریتم؛ چیست. نتایج نشان می‌دهد که ETSFS فضای کتری را مصرف می‌کند به نسبت دیگر الگوریتم‌ها، این امر وقوع می‌یابد به خاطر اندازه داده‌های سری شده در ETSFS که بیشتر افزایش نمی‌یابد و حجم اصلی بیشتر دست‌نخورده باقی می‌ماند.

آزمایش نشان داد که الگوریتم ETSFS می‌تواند به طور موفقی به ایمن‌سازی داده بپردازد زمانی که مجموعه داده‌های ETSFS افزایش می‌یابد و همچنین فنون تغییر و انتقال اصلاح می‌شود بدون اینکه بر اجرا تأثیرگذار باشد.

**جمع‌بندی و کارهای بیشتر**

ذخیره‌سازی داده و تبادل بین رایانه‌ها به سرعت در جهان توسعه می‌یابد. امنیت این داده‌ها موضوعی مهم در جهان است. بهترین راه حل اما بر ایمن‌سازی داده‌ها از طریق رمز دهی در کنار سایر روش‌ها است.

این مقاله پیشنهاد می‌کند که به تقویت الگوریتم TSFS برای پشتیبانی سرّی‌سازی کاراکترهای ویژه، اصلاح فرایند تغییر به وسیله ایجاد بیش از یک مدول عامل برای متفاوت سازی گونه‌های داده و دوری از افزایش حجم داده و نیز برای تصحیح فرایند انتقال به همین دلایل به وسیله چهار 16-آرایه؛ مبادرت گردد. نتایج تجربی نشان داده‌اند که الگوریتم ETSFS به طور موفق به سرّی‌سازی سمبل همانند داده‌های الفبایی اقدام نمود. عملکرد بهتر، بدون سازش بر زمان پردازش جستار یا حجم دیتابیس به دست می‌آید. با استفاده از الگوریتم‌های سرّی‌سازی جاافتاده به عنوان سرمشق مانند DES و AES,، الگوریتم ETSFS پیشنهادی نشان داد که کمترین حجم و کمترین زمان سرّی‌سازی را در مقایسه با دیگر الگوریتم‌ها مصرف می‌کند.

به سبب محدودیت‌های زمانی سخت است که همه سمبل های ویژه در این مقاله پوشش داده شود؛ لکن الگوریتم ETSFS می‌تواند توسعه یابد تا دیگر سمبل ها را نیز پوشش دهد. برای کار بیشتر مطلوب است که این الگوریتم برای تطابق یابی با اندازه‌ای از داده –بیشتر از 16 رقمی- بهبود داده شود. بعلاوه مطلوب است که ارزیابی بیشتری بر روی ایمنی الگوریتم ETSFS به وسیله اعمال پاره‌ای عملکردها انجام شود و حمله گران زمانی بیشتر نیاز داشته باشند تا به بازیابی کلیدها و هک اطلاعات سری شده دست یابند.