دوفصل‌نامه علمی منادی امنیت فضای تولید و تبادل اطلاعات (افتا)
سال ۱۴۰۰، جلد ۱۹، شماره ۲ (۸۸–۹۸)
https://monadi.isc.org.ir

ارائه‌شده در هفدهمین کنفرانس بین المللی انجمن رمز ایران در دانشگاه علم و صنعت ایران، ۱۹ و ۲۰ شهریور ۱۳۹۹

الگوریتم‌های موازی برای ارزیابی جعبه‌های جانشانی بزرگ رمزنگاری★

رضا قاسمی*[۱،] و بهروز خادم[۲]

[۱] دانشکده مهندسی کامپیوتر، دانشگاه صنعتی شریف، تهران، ایران
[۲] گروه رمز و امنیت، دانشکده و پژوهشکده فناوری اطلاعات و ارتباطات، دانشگاه جامع امام حسین (ع)، تهران، ایران

اطلاعات مقاله

کلمات کلیدی:
جعبه‌های جانشانی
والش–هادامارد سریع
درجه غیرخطی
پیچیدگی محاسباتی
پیاده‌سازی
موازی‌سازی

dor: 10.0000/000000000

نوع مقاله: پژوهشی

چکیده

با موفقیت اخیر نخستین کامپیوتر کوانتومی گوگل در حل یک مساله سخت، امنیت رمزهای متقارن در برابر حمله جستجوی فراگیر فضای کلید (با کلید ۱۲۸ بیتی)، به طور جدی مورد سوال قرار گرفته است. به همین دلیل تولید و ارزیابی توابع بولی و جعبه‌های جانشانی بزرگ یکی از چالش‌های عمده در تحقیقات رمزنگاری مدرن است که نیازمند تسریع در محاسبه تبدیل والش–هادامارد و محاسبه درجه غیرخطی برای توابع بولی و جعبه‌های جانشانی بزرگ برای مقاوم‌سازی طرح‌های رمزنگاری در برابر حملات خطي و تفاضلي است. پیچیدگی زمان و حافظه الگوریتم‌های موجود برای محاسبه تبدیل والش و درجه غیرخطی جعبه‌های جانشانی بزرگ با بیت ورودی و بیت خروجی از از مرتبه $\mathcal{O}(2^{n+m})$ هستند. در این مقاله پیشنهاد شده است که با موازی‌سازی محاسبه الگوریتم والش سریع، تعریف و دسترسی به ترانهاده ماتریس والش به جای خود این ماتریس و در ادامه با ترکیب یک بخش مهم از عملیات محاسبه درجه غیرخطی در داخل الگوریتم محاسبه ماتریس والش، زمان محاسبه ماتریس والش تا ۶۸ برابر و زمان محاسبه درجه غیرخطی تا ۱۰۲۱ برابر بهبود یافته است. صحت الگوریتم‌های پیشنهادی با شبیه‌سازی و آزمایش بررسی‌شده است و تحلیل کلی منابع مصرفی الگوریتم‌های پیشنهادی در مقایسه با الگوریتم‌های اولیه انجام شده است.

© ۱۴۰۰ انجمن رمز ایران

۱ مقدمه

توابع بولی و جعبه‌های جانشانی وظیفه ایجاد آشفتگی مورد نظر شانون [۱] در سامانه‌های رمزنگاری دنباله‌ای[۱] و قالبی[۲] را بر عهده دارند. تبدیل والش–هادامارد[۳] روشی برای ساده‌سازی نمایش و محاسبه‌ی برخی از خواص رمزنگاری توابع بولی و جعبه‌های جانشانی[۴] (نظیر درجه غیرخطی[۵] و مرتبه مصونیت از همبستگی) است. اما پیچیدگی زمانی و پیچیدگی حافظه در محاسبه آن برای یک جعبه جانشانی با n بیت ورودی و m بیت خروجی به ترتیب از مرتبه نمایی $\mathcal{O}(n2^{n+m})$ و $\mathcal{O}(2^{n+m})$ است. به علاوه با توجه به تأثیر مقاومت جعبه‌های جانشانی در مقابل حملات خطی [۲]، محاسبه درجه غیرخطی جعبه‌های جانشانی امری لازم و اجتناب‌ناپذیر است. اما پیچیدگی زمانی و پیچیدگی حافظه در محاسبه درجه غیرخطی یک جعبه جانشانی نیز از مرتبه $\mathcal{O}(2^{n+m})$ هستند. بنابراین تسریع و موازی‌سازی[۶] محاسبات مربوط به تبدیل والش–هادامارد، می‌تواند تأثیر زیادی در کاهش زمان محاسبه این ویژگی

★از کمیته علمی هفدهمین کنفرانس بین‌المللی انجمن رمز ایران برای داوری این مقاله تشکر می‌شود.
*نویسنده مسئول
آدرس‌های رایانامه: ghasemir@ce.sharif.edu (رضا قاسمی)، bkhadem@ihu.ac.ir (بهروز خادم)
© ۱۴۰۰ تمامی حقوق متعلق به انجمن رمز ایران است.

[1] stream ciphers [2] block Ciphers [3] Walsh-Hadamard transform
[4] substitution-boxes [5] nonlinearity [6] parallel

داشته باشد.

با توجه به نمایی بودن پیچیدگی محاسبات و حافظه در محاسبه درجه غیرخطی و تبدیل والش–هادامارد یک جعبه جانشانی بزرگ، چالش محاسبات انجام ارزیابی جعبه‌های جانشانی در ابعاد بزرگ (بزرگ‌تر از ۸ بیت ورودی و ۸ بیت خروجی) به مراتب بیشتر است به طوری که این چالش خصوصا بعد از تحقیقات اخیر تیم گوگل که منجر به به دست آوردن یک ماشین کوانتومی فوق سریع شده [۳] یکی از مسائل کلیدی و باز در رمزنگاری در مقاوم سازی رمزهای متقارن با کلید ۱۲۸ بیتی و بزرگتر است [۴] و طبق بررسی‌های انجام‌شده ارزیابی جعبه‌های جانشانی بزرگ، نیاز به زمان و حافظه بسیار زیادی دارد، به طوری که در پیاده‌سازی با متلب [۱] به دلیل کمبود حافظه، عملیات ارزیابی متوقف شده است [۵].

از جمله دلایل استفاده از جعبه‌های جانشانی بزرگ می‌توان به جلوگیری از حملات پیشرفته کانال جانبی و چالش‌های ایجادشده توسط ماشین‌های کوانتومی و رمزنگاری پساکوانتوم اشاره کرد. اگر چه این چالش‌ها تهدیدات عملی برای جعبه‌های جانشانی کوچک هستند ولی در مقابل جعبه‌های بزرگ فاقد کارایی عملی هستند. به همین دلیل اخیراً استفاده از جعبه‌های بزرگ در رمزهای قالبی مورد استفاده در پروتکل‌های ارتباطی متداول (مانند کاسومی در GSM) موردتوجه محققین قرارگرفته است [۶]. همچنین نشان داده شده است که بزرگ بودن جعبه‌های جانشانی می‌تواند باعث بهبود برخی از خواص رمزنگاری این جعبه‌ها مانند افزایش درجه جبری، بالا بردن مقاومت آن‌ها در مقابل حملات تفاضلی و خطی، بهبود خاصیت بهمنی، بهمنی اکید [۲] و استقلال بیتی [۳] شود [۷–۹].

برای جعبه‌های جانشانی بزرگ، محاسبه ماتریس والش و دسترسی به حافظه در محاسبه درجه غیرخطی، پیچیدگی محاسباتی و حافظه بسیار زیادی دارد. به همین دلیل هدف ما در این مقاله استفاده از روش‌های موازی‌سازی جدید و تغییر نحوه تخصیص حافظه و نحوه دسترسی به حافظه و نیز قرار دادن بخشی از الگوریتم محاسبه درجه غیرخطی در داخل الگوریتم محاسبه ماتریس والش، برای کاهش مؤثر زمان و حافظه مصرفی می‌باشد که هر یک از این روش‌ها به تنهایی و همچنین در کنار هم به عنوان یک نوآوری می‌باشند.

در ادامه سازمان‌دهی این مقاله به شرح زیر است. در بخش ۲ به معرفی پیشینه تحقیقاتی موضوع پرداخته می‌شود. در بخش ۳ مفاهیم و الگوریتم‌های مورد نیاز و اولیه تعریف می‌شود. در بخش ۴ مسئله اصلی تحقیق و اهداف فرعی پیرامون آن بیان می‌شوند. راه حل تسریع پیشنهادی مقاله در بخش ۵ معرفی می‌شود. بخش ۶ شامل مقایسه نتایج تحلیلی و تجربی به دست آمده از انجام آزمایشات قبل و بعد از تسریع است. در انتها در بخش ۷ مروری بر نتایج کلی به دست آمده انجام می‌شود و برخی پیشنهادها برای ادامه کار ارائه می‌شود.

[1]Matlab [2]Strict Avalanche Criterion (SAC) [3]bit independence

۲ پیشینه تحقیقات مرتبط

برای نخستین بار در زمینه موازی‌سازی تبدیل والش–هادامارد اوسوسانیا و همکاران (۱۹۹۴) [۱۰] تبدیل والش را برای توابع بولی با موازی‌سازی و از طریق کامپیوترهای معمولی و یک ابرکامپیوتر محاسبه کرده‌اند، اما هیچ الگوریتمی را معرفی نکرده‌اند و فقط نتایج به دست آمده را ثبت کردند. آن‌ها در تحقیق خود در بهترین حالت موفق به تسریع ۶ برابری نسبت به قبل شدند و دلیل بهتر نشدن این تسریع را عدم توانایی سیستم‌عامل برای به حداکثر رساندن هم‌زمانی نخ‌ها بیان کردند.

نخستین بار برای محاسبه درجه غیرخطی جعبه‌های جانشانی، یوسف و همکاران (۱۹۹۵) [۱۱] با استفاده از تبدیل والش سریع، تخمینی از درجه غیرخطی را به دست آوردند. در این مقاله یک کران بالا از درجه غیرخطی با استفاده از جدول تقریبات خطی [۴]، برای جعبه‌های جانشانی به دست آوردند، اما در مقاله خود بحثی در مورد پیچیدگی محاسباتی و حافظه‌ای روش پیشنهادی نکردند.

گاندوبادی [۵] و همکاران (۲۰۱۸) [۱۲] نیز تقریبی از درجه غیرخطی جعبه‌های جانشانی برای به دست آوردن یک جعبه جانشانی ایمن به دست آوردند ولی آن‌ها هم وارد پیاده‌سازی و تحلیل پیچیدگی محاسباتی و حافظه نشده است و یک کران را محاسبه کرده‌اند.

سوسا [۶] و همکاران (۲۰۱۶) [۱۳] برای محاسبه درجه غیرخطی توابع بولی با استفاده جعبه‌های جانشانی بیان کرده‌اند که مرتبه $\mathcal{O}(n2^n)$ برای توابع بولی f_n می‌باشد اما در مورد تعمیم این روش برای جعبه‌های جانشانی، پیاده‌سازی و بهینه‌سازی کردن روش خود بیان نکرده‌اند.

آندراد [۷] و همکاران (۲۰۱۴) [۱۴] روشی برای موازی‌سازی نحوه‌ی محاسبه‌ی تبدیل والش ارایه دادند که از خط لوله [۸] و حافظه سلسله مراتبی [۹] در پردازنده‌های گرافیکی بهره می‌برد و به این ترتیب محدودیت‌های بانک حافظه در پردازنده‌های گرافیک را به حداقل رساندند. نویسندگان در مقاله خود اشاره کردند که برای یک تابع بولی با ۱۲۸ بیت ورودی، حداکثر به تعداد ۶۹۵۲۱ تبدیل والش در هر واحد زمان (میلی ثانیه) دست یافتند. آن‌ها هیچ اشاره‌ای به تعمیم روش خود به جعبه‌های جانشانی و حتی توابع بولی بزرگ‌تر از ۲۵۶ بیت ورودی، نداشتند و روش موازی‌سازی خود را نیز شرح ندادند.

پیکک [۱۰] و همکاران (۲۰۱۴) [۱۵] برای ارزیابی توابع بولی و جعبه‌های جانشانی ابزار نرم‌افزاری سِت [۱۱] را ارائه کردند. دو هدف اصلی در طراحی این نرم‌افزار لحاظ شد، هدف اول ارزیابی یک مجموعه گسترده‌تر از خواص نسبت به ابزارهای مشابه بود و هدف دوم به دست آوردن یک جعبه جانشانی با خواص مورد نظر بود. نویسندگان مقاله اذعان کردند که در تحقیق خود معیار استقلال بیتی را در نظر نگرفتند و روش پیشنهادی ایشان برای جعبه‌های جانشانی بزرگ ۱۶ × ۱۶ زمان

[4]Linear approximation Table(LAT) [5]Gangopadhyay [6]Sosa [7]Andrade
[8]pipeline [9]hierarchy [10]Picek [11]SET

و حافظه بسیار زیادی را صرف می‌کند و بررسی روش‌های بهبود آن را برای جعبه‌های جانشانی بزرگ پیشنهاد کردند.

آمارال[1] و همکاران (۲۰۱۶) [۵] یک ابزار نرم‌افزاری برای ارزیابی خواص رمزنگاری جعبه‌های جانشانی از جمله درجه غیرخطی و تبدیل والش، با استفاده از فنون محاسباتی مختلف پیاده‌سازی کردند. آن‌ها در مقاله خود پیشنهاد کردند برای جعبه‌های جانشانی با اندازه کوچک، پیاده‌سازی ابزار با متلب ساده‌تر است اما برای جعبه‌های بزرگ‌تر بهتر است پیاده‌سازی با زبان C++ انجام شود به دلیل آنکه پیاده‌سازی با زبان C++ استفاده بهتری از حافظه می‌کند. همچنین آن‌ها در مقاله خود به این نتیجه رسیدند که پیاده‌سازی انجام‌شده برای بررسی جعبه‌های جانشانی بزرگ ازنظر زمان و حافظه مناسب نیست و پیشنهاد کردند که برای حل مسئله از موازی‌سازی استفاده شود.

بیکوف[2] و همکاران (۲۰۱۸) [۱۶] با استفاده از الگوریتم والش سریع محاسبه طیف والش توابع بولی به شکل‌های دیگری موازی کردند و بر روی واحد پردازنده گرافیکی[3] پیاده‌سازی کردند و در آخر نتایج را با هم مقایسه کرده‌اند. موازی‌سازی‌های مختلف الگوریتم والش سریع در مقاله بیکوف ازنظر پیچیدگی پیاده‌سازی، منابع مورد استفاده و استراتژی بهینه‌سازی، فنون متفاوتی دارند. در این مقاله طیف والش فقط برای یک تابع بولی محاسبه شد و به جعبه‌های جانشانی اشاره نشد.

همان طور که از بررسی پیشینه تحقیقات مرتبط روشن می‌شود، در هیچ یک از آن‌ها موازی‌سازی تبدیل والش–هادامارد برای جعبه‌های جانشانی انجام نشده است و نیز در هیچ یک از تحقیقات قبلی، ارزیابی درجه غیرخطی برای جعبه‌های جانشانی ۱۶ × ۱۶ انجام نشده است. برخلاف مقالات [۱۴] و [۱۶] ما بروی CPU روش موازی‌سازی جدید خود را پیاده کردیم، چون در زمانی که اندازه جعبه بزرگ می‌شود مقدار حافظه مورد نیاز برای ماتریس ها از مرتبه چند ده گیگابایت می‌شود و پردازنده‌های گرافیکی برای پردازش باید این مقدار داده‌ها را بین حافظه خود و حافظه اصلی جابه‌جا کنند که سبب زمان زیاد آن می‌شود.

نوآوری این مقاله در پیاده‌سازی یک روش موازی‌سازی جدید (با استفاده از CPU) برای محاسبه تبدیل والش–هادامارد و کاربرد آن در محاسبه درجه غیرخطی جعبه‌های جانشانی بزرگ است. برای این پیاده‌سازی‌ها برای اولین بار ما از ترانهاده ماتریس والش به جای خود ماتریس والش استفاده کردیم و بخش زمان‌بر الگوریتم محاسبه درجه غیرخطی در داخل الگوریتم والش قرار دادیم. روش پیشنهادی خود به همراه تحلیل کامل منابع مصرفی آن روش، در پاسخ به هر دو چالش فوق را بیان می‌کنیم.

[1]Amaral [2]Bikov [3]Graphics processing Unit (GPU)

۳ مفاهیم و الگوریتم‌های اولیه

در الگوریتم‌های موازی در اکثر موارد بین دستورات بعد با دستورات قبلی وابستگی وجود ندارد و می‌توان دستورهای مختلف الگوریتم موازی را به صورت هم‌زمان (در یک واحد زمانی) اجرا کرد. موازی‌سازی قابلیتی است که در آن می‌توان بخشی از دستورات الگوریتم (که قابلیت موازی‌سازی دارند) را به صورت هم‌زمان و موازی به وسیله چند پردازنده و یا چند هسته پردازشی اجرا کرد.

در الگوریتم‌های موازی‌شده، پیاده‌سازی هر یک از بخش‌های قابل موازی‌سازی را به نخی[4] نگاشت می‌کنیم و نخ‌ها در واحد پردازشگر مرکزی[5] رایانه، به صورت همروند اجرا می‌شوند. سیستم‌عامل، می‌تواند نخ‌ها را با درخواست خودشان یا به تشخیص خودش به حالت خواب[6] برده و یا حتی از حافظه تصادفی[7] خارج کند و بعد با توجه به درخواست آن‌ها را بیدار کند. به عملیات جابه جایی بین حافظه تصادفی و حافظه ذخیره‌سازی، مبادله می‌گویند.

در ادامه به جای واژه‌های حافظه تصادفی، جعبه جانشانی، تبدیل والش–هادامارد به ترتیب از واژه‌های رم، جعبه و تبدیل استفاده می‌کنیم.

یک جعبه جانشانی تعمیم مشخصی از یک تابع بولی با خروجی تک‌بیتی، به یک تابع بولی برداری[8] $s = (f_1, f_2, \ldots, f_m)$ با خروجی چندبیتی است (f_i تابع مؤلفه i ام نامیده می‌شود). به طور دقیق‌تر، یک جعبه‌ی جانشانی $n \times m$ از یک ورودی n بیتی به یک خروجی m بیتی است [۱۷]. با توجه به این تعریف ماتریس والش–هادامارد $s : f_2^n \to f_2^m$ یک جعبه جانشانی S به صورت رابطه (۱) است [۱۸].

$$w_s(u,v) = \sum_{x=0}^{2^n-1} (-1)^{\langle v, S(x)\rangle + \langle u,v\rangle} \qquad (1)$$
$$(0 \le u \le 2^n - 1), \quad (0 \le v \le 2^m - 1)$$

الگوریتم ۱، نشان‌دهنده الگوریتم والش سریع برای محاسبه ماتریس والش جعبه‌های جانشانی با n ورودی و m خروجی است که از الگوریتم والش سریع ارائه شده در [۱۹] بهره می‌بریم.

درجه غیرخطی یک جعبه‌های جانشانی S کمینه درجه غیرخطی همه ترکیبات خطی توابع مؤلفه g_i (که در رابطه (۲) آمده است) می‌باشد [۲۰].

$$g_i(x) = \sum_{i=0}^{m-1} a_i^{(j)} f_i(x) \qquad (2)$$
$$(0 \le j \le 2^m - 1, \ a_i^{(j)} \in f_2)$$

با توجه به تعریف بالا، کمینه فاصله همینگ همه توابع $v \cdot S(x)$ با توابع آفینی $w \cdot x$ را درجه غیرخطی جعبه‌های جانشانی S می‌نامند، به صورت رابطه (۳) تعریف می‌شود [۲۱].

[4]thread [5]Central Processing Unit (CPU) [6]sleep mode [7]random access memory [8]vectorial boolean function

Algorithm 1 Pseudcode of Fast Walsh-Hadamard(FWT)

Require: The Polarity Truth Table PTT of the Sbox S, with 2^n Rows and $2^m - 1$ Columns

Ensure: The Walsh Transform WT of the Sbox S, with 2^n Rows and $2^m - 1$ Columns

```
z ← 0; j ← 1; WT ← PTT;
while z < 2^m − 1 do
    while (j < 2^n do
        for i = 0 to 2^n − 1 do
            if (i&j) = 0 then
                temp ← WT[i][z];
                WT[i][z] ← WT[i][z] + WT[i + j][z];
                WT[i + j][z] ← tempWT[i + j][z];
            end if
        end for
        j ← 2 ∗ j;
    end while
    z ← z + 1;
end while
```

$$g_i(x) = ۲^{n-۱} - \frac{۱}{۲} \max_{v \in f_۲^m, w \in f_۲^n} \left| \sum_{x \in f_۲^n} (-۱)^{\langle v, S(x) \rangle \oplus \langle u, v \rangle} \right|$$
$$(۰ \leq u \leq ۲^n - ۱),\ (۰ \leq v \leq ۲^m - ۱) \qquad (۳)$$

الگوریتم ۲ شبه کد محاسبه درجه غیرخطی برای جعبه‌های جانشانی است که آقای پیکک آن را در [۱۷] آورده است.

۴ تعریف مسئله و چالش‌های موجود

برای ارزیابی بعضی از ویژگی‌های مهم جعبه‌های جانشانی از قبیل درجه غیرخطی، نیاز به محاسبه ماتریس والش جعبه‌ها است. اما پیچیدگی محاسباتی و حافظه بسیار زیادی دارند. هدف اصلی این مقاله تسریع اجرای محاسبه ماتریس والش و محاسبه درجه غیرخطی برای جعبه‌های جانشانی بزرگ است.

ماتریس والش جعبه‌های بزرگ حافظه‌ی زیادی را اشغال می‌کنند و با افزایش ورودی و خروجی جعبه‌های جانشانی، تعداد دفعات دسترسی به درایه‌های آن در داخل الگوریتم‌ها، به صورت نمایی افزایش می‌یابد و به دنبال آن محاسبه درجه غیرخطی جعبه‌های بزرگ (مانند ۱۶ × ۱۶) زمان زیادی به طول می‌انجامد. برای تحقق هدف اصلی به تحقق چند هدف فرعی نیازمند هستیم. انجام اقداماتی از جمله بهبود موازی‌سازی تبدیل، استفاده از ترانهاده ماتریس والش-هادامارد و ترکیب هوشمندانه زیرمسئله‌ها (انجام عملیات مختلف با فقط یک بار خواندن ماتریس والش)، اهداف فرعی این مقاله هستند.

Algorithm 2 Pseudcode of Computing Nonlinearity of Sbox

Require: The Walsh Transform WT of the Sbox S, with 2^n Rows and $2^m - 1$ Columns

Ensure: The Nonlinearity degree

```
temp ← 0; Nonlinearity ← 10000; var[2^m] ← 0;
while z < 2^m − 1 do
    while (j < 2^n do
        /*start Time-consuming part */
        if abs(WT[j][z]) > temp then
            temp ← abs(WT[j]z]);
        end if
        var[z] = (2^n − temp) ≫ 1;
        /*end Time-consuming part */
    end while
    if var[z] < Nonlinearity then
        Nonlinearity = var[z];
    end if
    j ← j + 1;
    z ← z + 1;
end while
```

همان طور که در تحقیقات مرتبط [۱۰، ۱۴، ۱۶] تشریح شد، راه‌حل‌های ارائه‌شده قبلی برای موازی‌سازی تبدیل والش فقط منحصر به توابع بولی بوده‌اند و کاری در زمینه موازی‌سازی برای جعبه‌های جانشانی انجام نشده است. همچنین با اینکه در زمینه ارزیابی درجه غیرخطی جعبه‌های جانشانی تحقیقات زیادی انجام شده است، اما همه آن‌ها در زمینه ارزیابی درجه غیرخطی جعبه‌های بزرگ (۱۶ × ۱۶) دچار چالش بوده‌اند [۵، ۱۹]. در بخش بعدی راه‌حل‌های پیشنهادی خود را شرح می‌دهیم.

۵ راه‌حل‌های پیشنهادی

ابتدا موضوع تسریع در نحوه دسترسی به ماتریس والش با استفاده از ترانهاده ماتریس والش را مطرح می‌کنیم. به موضوع کاهش پیچیدگی محاسباتی تبدیل والش برای جعبه‌های جانشانی می‌پردازیم و راه حل خود را که یک الگوریتم موازی‌شده متناسب با جعبه‌های جانشانی است معرفی می‌کنیم. سومین راه حل پیشنهادی ما مربوط به موضوع کاهش پیچیدگی محاسباتی و حافظه در محاسبه درجه غیرخطی یک جعبه جانشانی است که برای این کار از یک ترکیب خاص بین زیرمسئله‌های مربوطه استفاده می‌کنیم، به طوری که در هنگام محاسبه ماتریس والش، محاسبات مربوط به (بخش زمان‌بر[1]) درجه غیرخطی جعبه جانشانی را نیز انجام می‌دهیم. در ادامه این راه‌حل‌های پیشنهادی را به ترتیب شماره روش، تشریح می‌کنیم.

[1] time-consuming part

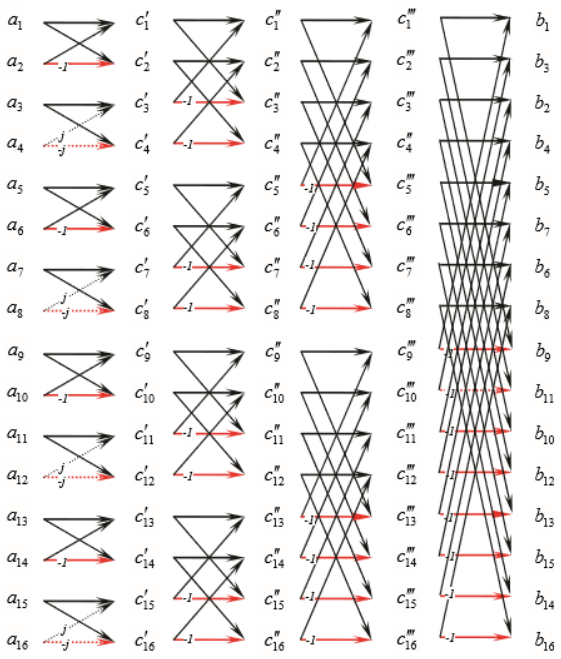
۱.۵ تسریع در دسترسی به ماتریس والش با استفاده از ترانهاده ماتریس والش

برای تغییر نحوه تعریف و دسترسی به درایه‌های ماتریس والش، راه حل پیشنهادی ما استفاده از ترانهاده ماتریس والش به جای خود ماتریس والش در الگوریتم‌های ۱ و ۲ است. برای ماتریس‌های بزرگ به دلیل آن که کمپایلر قادر به تعریف ماتریس با حجم بالا نیست، باید تعریف هر درایه را با تخصیص به صورت دستی انجام دهیم. به طور پیش‌فرض با توجه به تعریف ماتریس والش باید این ماتریس به صورت ۲^n ردیف از آرایه‌های با $۲^m - ۱$ عضو (اعداد صحیح) تعریف شود. بنابراین چون در داخل حلقه اول الگوریتم والش کامپایلر باید به تمام درایه‌های هر ستون از ماتریس دسترسی داشته باشد، لازم است در هر دور حلقه اول الگوریتم ۱ به همه آرایه‌ها دسترسی داشته باشد تا درایه‌ها را بخواند و تغییر دهد. در هر دور حلقه اول الگوریتم والش اشارگر به آرایه‌ها به تعداد ۲^n (تعداد سطر ماتریس والش)، تغییر می‌کند و سیستم‌عامل کل آرایه‌ها یا بخش‌هایی از آن‌ها را به داخل رم می‌آورد. پس زمان اجرای این عملیات، به طور نمایی به تعداد مبادله این آرایه‌ها بین رم و حافظه و تغییر اشارگرها بستگی دارد. به عبارت دیگر با افزایش تعداد بیت‌های ورودی و خروجی جعبه جانشانی، تعداد دور حلقه اول و تعداد آرایه‌ها به صورت نمایی افزایش یافته و درنتیجه عملیات تغییر اشارگرها و مبادله آرایه‌ها زمان بسیار بیشتری به طول می‌انجامد.

پیشنهاد ما برای رفع این مشکل، استفاده از ترانهاده ماتریس والش به جای خود ماتریس والش است. ماتریس ترانهاده را به صورت $۲^m - ۱$ سطر از آرایه‌های با ۲^n عضو (اعداد صحیح) تعریف می‌کنیم. به این ترتیب در الگوریتم والش سریع در زمان فراخوانی جای سطر و ستون را جابجا می‌کنیم؛ به‌طور مثال به جای فراخوانی $WT[i][z]$، درایه $WT[z][i]$ را فراخوانی می‌کنیم. درنتیجه با تغییرات انجام‌شده، در هر دور حلقه فقط یک آرایه فراخوانی می‌شود و با کاهش تغییر اشارگر و فراخوانی آرایه به یک عدد، در هر دور حلقه اول الگوریتم ۱، عملیات مبادله آرایه‌ها به صورت مؤثری کاهش می‌یابد. به طور مثال برای $m = ۳$ و $n = ۴$ با استفاده از خود ماتریس والش در هر دور حلقه اول الگوریتم ۱ با توجه به ساختار سیستم‌عامل، حداقل یک بار کل آرایه‌ها را داخل رم می‌آورد و بعد از تغییرات داخل حافظه می‌نویسد. بنابراین با توجه به تعداد دور خود حلقه اول حداقل ۱۲۲ آرایه بین حافظه و رم جابه‌جا می‌شوند. در صورتی که با استفاده از ترانهاده ماتریس والش این عدد به ۷ می‌رسد چون در این صورت در هر دور حلقه اول فقط یک آرایه به داخل رم می‌آید.

۲.۵ کاهش پیچیدگی محاسباتی تبدیل والش با استفاده از بهبود موازی‌سازی تبدیل والش سریع

پیشنهاد دوم این مقاله برای کاهش پیچیدگی محاسباتی الگوریتم ۱ استفاده از یک روش موازی‌سازی و متفاوت از روش‌های قبلی [۹، ۱۰] است. روش‌های قبلی برای موازی‌سازی تبدیل والش برای یک تابع بولی طراحی شدند که برای توسیع آن‌ها به محاسبه تبدیل والش یک جعبه جانشانی لازم



شکل ۱. فرایند انجام الگوریتم والش سریع

است یک حلقه دیگر (به تعداد تکرار بیت‌های خروجی جعبه جانشانی) به الگوریتم محاسبه والش اضافه شود. در این صورت چون همه نخ‌ها در داخل یک حلقه اصلی قرار می‌گیرند، موازی‌سازی به طور بهینه اجرا نمی‌شود و سیستم‌عامل ناچار است بعد از هر دور حلقه اصلی، نخ‌ها را به حالت‌های خواب برده و دوباره آن‌ها را فراخوانی کند که فرایند و مبادله‌ای طولانی دارد.

شکل ۱ نشان‌دهنده روند انجام الگوریتم ۱ برای فقط یک ستون از ماتریس والش برای جعبه‌ای جانشانی با ۴ ورودی و ۴ خروجی است. با توجه به الگوریتم ۱، محاسبات هیچ یک از ستون‌های ماتریس والش به دیگری وابسته نیست، یعنی هر یک از ستون‌های عملیات شکل ۱ را به صورت کاملاً مستقل انجام می‌دهند. پس برای موازی‌سازی می‌توان محاسبات همه ستون‌ها را با هم به صورت همروند و موازی انجام داد.

ما برای موازی‌سازی تعداد ستون‌ها را به تعداد نخ‌هایی که داریم تقسیم می‌کنیم، یعنی تعداد دور حلقه اول الگوریتم ۱ را به تعداد نخی داریم تقسیم می‌کنیم و عدد به دست آمده نشان‌دهنده آن تعدادی از ستون‌های ماتریس والش است که محاسبه آن‌ها به هر نخ واگذار می‌شود. به طور مثال برای محاسبه ماتریس والش با استفاده از ۸ نخ برای $m = ۴$ و $n = ۳$، حلقه اول الگوریتم ۱ شامل ۱۵ دور می‌باشد که هر دور برای محاسبه یک ستون از ماتریس والش می‌باشد. بنابراین محاسبات به هر نخ محاسبات مربوط به دو ستون ماتریس والش را می‌دهیم، بجز نخ آخری که فقط محاسبات یک ستون به آن می‌رسد. بنابراین بطور نظری به حداقل تسریع ۷ برابری در همین الگوریتم می‌رسیم. نویسندگان این مقاله شبه کد الگوریتم موازی‌شده والش را (بعد از انجام تغییرات برای اجرای راه‌کارهای پیشنهادی بعدی) در الگوریتم ۴ گسترش داده‌اند. در

بخش ۶ نتایج آزمایشات عملی آمده است.

۳.۵ کاهش پیچیدگی محاسباتی و حافظه در محاسبه درجه غیرخطی با استفاده از ترکیب زیرمسئله‌ها

پیشنهاد سوم این مقاله برای تسریع اجرای محاسبه درجه غیرخطی جعبه جانشانی، قرار دادن بخش زمان‌بر عملیات محاسبه درجه غیرخطی در داخل الگوریتم محاسبه ماتریس والش هست، به طوری که کامپایلر بتواند با یک بار فراخوانی ماتریس والش در الگوریتم والش سریع، محاسبات مربوط به محاسبه درجه غیرخطی را هم انجام دهد. دلیل اصلی استفاده از این روش جلوگیری از فراخوانی دوباره ماتریس والش داخل الگوریتم محاسبه درجه غیرخطی می‌باشد. با توجه به این روش پیشنهادی در هنگام محاسبه ماتریس والش بخش اصلی الگوریتم محاسبه درجه غیرخطی (بخش زمان‌بر در الگوریتم ۲) هم محاسبه می‌شود و به کمک یک آرایه نتایج این بخش به الگوریتم محاسبه درجه غیرخطی ارسال می‌شود. بخش اصلی الگوریتم محاسبه درجه غیرخطی (بخش زمان‌بر در الگوریتم ۲) به صورت سریالی اجرا می‌شود به علاوه با استفاده از قابلیت موازی‌سازی این بخش را در داخل الگوریتم محاسبه والش موازی اجرا می‌کنیم تا نیاز به تکرار این عمل برای الگوریتم محاسبه درجه غیرخطی نباشد.

الگوریتم محاسبه درجه غیرخطی جعبه‌های جانشانی در بخش ۳ با عنوان الگوریتم ۲ آمده است. ما با قرار دادن بخش اصلی الگوریتم محاسبه درجه غیرخطی (بخش زمان‌بر در الگوریتم ۲) در داخل الگوریتم ۱ و با استفاده از ماتریس ترانهاده والش به جای ماتریس والش، الگوریتم والش سریع را به صورت الگوریتم ۳ تغییر داده‌ایم تا یک نوآوری دیگر این مقاله محسوب می‌شود.

نسخه موازی‌شده الگوریتم ۳، (الگوریتم والش سریع بعد از اعمال همه روش‌های تسریع) در الگوریتم ۴ آمده است.

الگوریتم ۴ اقدام به ایجاد و فراخوانی نخ‌هایی که طراحی‌شده می‌کند که تابع نخ‌ها در الگوریتم ۵ آمده است (الگوریتم ۵ ابداع ما می‌باشد)، در ادامه منتظر اتمام همه آن‌ها می‌ماند و ورودی و خروجی نخ‌ها همان ورودی و خروجی الگوریتم والش سریع است با این تفاوت که نخ‌ها هرکدام مسئول انجام محاسبه ستون‌های از ماتریس والش هستند.

تعداد ستون‌ها و بازه‌ی آن به شماره و تعداد کل نخ‌ها بستگی دارد، که در الگوریتم ۵ به طور واضح مشخص شده است. این الگوریتم در ابتدا منتظر صدا زدن نخ‌ها از طریق تابع اصلی والش است و هر کدام از نخ‌ها در آخر، تابع اصلی را از تمام‌شدن عملیات خود مطلع می‌کند.

نویسندگان همین مقاله الگوریتم محاسبه درجه غیرخطی(الگوریتم ۲) را به صورت الگوریتم ۶ تغییر داده‌اند. محاسبات اولیه الگوریتم ۶، که مربوط به بخش زمان‌بر الگوریتم ۲ است، از طریق الگوریتم ۳ یا با موازی‌شده آن یعنی الگوریتم ۴ به دست می‌آید. این الگوریتم یک آرایه شامل نتیجه محاسبات برای هرکدام از ستون‌های ماتریس والش (که از الگوریتم والش سریع پیشنهادی این مقاله به دست آمده) را می‌گیرد و با

Algorithm 3 Pseudcode of FWT with Part of the calculations Nonlinearity

Require: The Polarity Truth Table PTT of the Sbox S, with $2^m - 1$ Rows and 2^n Columns

Ensure: The Walsh Transform WT of the Sbox S, with $2^m - 1$ Rows and 2^n Columns, The var array with 2^m entries

```
z ← 1; j ← 1; WT ← PTT; temp ← 0;
while z < 2^m − 1 do
    while (j < 2^n do
        for i = 0 to 2^n − 1 do
            if (i&j) = 0 then
                temp ← WT[z][i];
                WT[z][i] ← WT[z][i] + WT[z][i + j];
                WT[z][i + j] ← tempWT[z][i + j];
                if i == 2^n − 1 then
                    if abs(WT[z][j]) > temp then
                        temp = abs(WT[z][j]);
                    end if
                end if
            end if
        end for
        j ← 2 ∗ j;
    end while
    var[z] = (2^n − temp) ≫ 1;
    z ← z + 1;
end while
```

Algorithm 4 Pseudcode of Parallel Implementation of FWT

Require: The Polarity Truth Table PTT of the Sbox S, with $2^m - 1$ Rows and 2^n Columns

Ensure: The Walsh Transform WT of the Sbox S, with $2^m - 1$ Rows and 2^n Columns, The var array with 2^m entries

```
Create FWT threads;
Calling FWT threadss;
Wait for FWT threads to finish;
Joining the FWT threads;
```

پیدا کردن کوچک‌ترین عدد این آرایه، مقدار غیرخطی را به‌عنوان خروجی برمی‌گرداند.

برای مقایسه میزان منابع مصرفی بین الگوریتم‌های قبلی با الگوریتم‌های پیشنهادی، پیچیدگی زمانی و حافظه هر کدام از الگوریتم‌های والش و درجه غیرخطی در جدول ۱ آورده شده است. در محاسبه پیچیدگی‌ها فرض‌شده‌است که تعداد بیت‌های ورودی و بیت‌های خروجی جعبه جانشانی به ترتیب n و m است. در الگوریتم محاسبه درجه

Algorithm 5 Pseudcode of threads FWT

Require: The Polarity Truth Table PTT of the Sbox S, with $((2^m - 1$ / Total number of threads) $*$ number thread) Rows and 2^n Columns

Ensure: The Walsh Transform WT of the Sbox S, with $((2^m - 1$ / Total number of threads) $*$ number thread) Rows and 2^n Columns, The var array with (2^m / Total number of threads) entries

```
z ← 1; j ← 1; WT ← PTT; temp ← 0;
waiting for call by main function;
for index from ((2^m − 1 / Total number of threads) ∗ (number thread-1)) ← (( 2^m −1 / Total number of threads) ∗ number thread) do
    while j < 2^n do
        for i = 0 to 2^n − 1 do
            if (i&j) = 0 then
                temp ← WT[z][i];
                WT[z][i] ← WT[z][i] + WT[z][i + j];
                WT[z][i + j] ← tempWT[z][i + j];
                if i == 2^n − 1 then
                    if abs(WT[z][t1]) > temp then
                        temp = abs(WT[z][j]);
                    end if
                end if
            end if
        end for
        j ← 2 ∗ j;
    end while
    var[z] = (2^n − temp) ≫ 1;
    z ← z + 1;
end for
the main function of completion notifications.
```

Algorithm 6 Pseudcode of Computing Nonlinearity if Sbox after doing a large part of the operation in algorithm 4

Require: The var array with 2^m entries

Ensure: The Nonlinearity degree

```
temp ← 0; Nonlinearity ← 10000; var[2^m] ← 0;
while z < 2^m − 1 do
    if var[z] < Nonlinearity then
        Nonlinearity = var[z];
    end if
    z ← z + 1;
end while
```

غیرخطی، در هر دور حلقه، به تبدیل والش جعبه‌های جانشانی که قبلا محاسبه شده است و در هر حافظه موجود است، دسترسی داریم و دیگر این تبدیل را محاسبه نمی‌کنیم؛ به همین دلیل تبدیل والش در مرتبه زمانی محاسبه درجه غیرخطی تاثیری ندارد.

الگوریتم محاسبه والش (الگوریتم ۱) سریع شامل سه حلقه تودرتو است که دو حلقه با تعداد دور ۲^n و ۲^m می‌باشد و حلقه سوم n بار اجرا می شود. بنابراین پیچیدگی زمانی از مرتبه $\mathcal{O}(n۲^n۲^m)$ می‌باشد. پیچیدگی حافظه این الگوریتم نیز مربوط به مقدار حافظه مصرفی ماتریس والش است که برای جعبه‌های جانشانی از مرتبه $\mathcal{O}(۲^{n+m})$ است و در جدول ۱ آمده است .

در داخل والش سریع پیشنهادی (الگوریتم ۴) ما یک ماتریس والش و یک آرایه از مرتبه $\mathcal{O}(۲^n)$ برای محاسبات اولیه درجه غیرخطی (مربوط به راه حل پیشنهادی سوم) داریم، پس پیچیدگی حافظه برابر مجموع حافظه‌ای است که ماتریس والش و این آرایه اشغال می‌کنند. پیچیدگی زمانی والش سریع پیشنهادی به این صورت می‌باشد که بیرون‌ترین حلقه (حلقه اصلی) بین نخ‌ها تقسیم می‌شود و به صورت موازی اجرا می‌شوند (مربوط به راه حل پیشنهادی دوم) بنابراین همانطور که جدول ۱ نشان می‌دهد پیچیدگی زمانی این الگورتیم برابر پیچیدگی الگوریتم ۱ تقسیم بر تعداد نخ می‌باشد.

در الگوریتم درجه غیرخطی (الگوریتم ۲) ماتریس والش فراخوانی می‌شود و این الگوریتم شامل دو حلقه تو در تو با تعداد دور برابر با سطر و ستون ماتریس والش (۲^n و ۲^m) می‌باشد. بنابراین پیچیدگی زمانی و حافظه آن از مرتبه $\mathcal{O}(۲^n۲^m)$ می‌باشد. در ادامه با استفاده از روش سوم پیشنهادی محاسبات درجه غیرخطی را در قسمت محاسبه ماتریس والش قرار دادیم و فقط یک آرایه به اندازه ۲^n فراخوانی می‌شود. بنابراین پیچیدگی زمانی و حافظه الگوریتم محاسبه درجه غیرخطی پیشنهادی از مرتبه $\mathcal{O}(۲^n۲^m)$ می‌باشد.

در جدول ۱ الگوریتم‌ها و روش‌های پیشنهادی نمایی هستند، اما باید این را لحاظ کرد که همانطور که در بخش ۱.۵ گفتیم با استفاده از ترانهاده ماتریس تعداد دسترسی از مرتبه نمایی کاهش یافته ولی این کاهش به صورت کاهش پیچیدگی خود را نشان نداده است، ولی با توجه به جدول ۴ که در بخش ۶ بیان می کنیم در زمان اجرا حتی برای n و m بزرگ کاملا خود را نشان می‌دهد.

۶ نتایج آزمایش‌های تجربی

برای اطمینان از صحت الگوریتم‌های پیشنهادی، درجه غیرخطی جعبه جانشانی AES و دو جعبه جانشانی دیگر را با الگوریتم اولیه محاسبه درجه غیرخطی (الگوریتم ۲) و الگوریتم پیشنهادی (الگوریتم ۶ شامل ترکیب کل روش‌ها) سنجیده‌ایم.

لازم به ذکر است در هنگام اجرای الگوریتم درجه غیرخطی نیاز به

جدول ۱. مقایسه پیچیدگی محاسباتی و حافظه‌ی الگوریتم‌ها

مرتبه پیچیدگی الگوریتم	والش سریع الگوریتم ۱	والش سریع پیشنهادی الگوریتم ۴	محاسبه درجه غیرخطی الگوریتم ۲	درجه غیرخطی پیشنهادی الگوریتم ۶
محاسباتی (زمانی)	$n2^n2^m$	$n2^n(2^m/TNT)$	2^n2^m	2^n
حافظه	2^n2^m	$2^n2^m+2^n$	2^n2^m	2^n

Help: Total_number_threads:TNT, The Integers are 8 bits for storage.

جدول ۲. مقایسه نتایج محاسبه درجه غیرخطی

اندازه جعبه جانشانی	۱۶ × ۱۶	۱۰ × ۱۰	۸ × ۸
الگوریتم اولیه (الگوریتم ۱)	۳۲۰۰۰	۳۸۴	۱۱۲
الگوریتم‌های پیشنهادی (الگوریتم ۶)	۳۲۰۰۰	۳۸۴	۱۱۲

جدول ۳. مشخصات سیستم پردازشی

Enviroment	platform
CPU	Inter Core i7 8400K-12 thread
Memory(RAM)	16Gb DDR4
OS	Win10 x64
Compiler	MSVC 2017

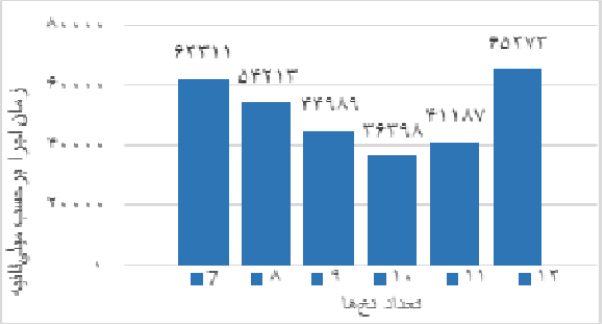
محاسبه والش (الگوریتم ۱) می‌باشد. از آنجا که الگوریتم‌ها ۱ و ۲ از مقالات مهمی آمده‌اند صحت آن‌ها مورد تایید می‌باشد. بنابراین با مقایسه نتایج الگوریتم ۲ با الگوریتم ۶ (که ترکیب کل روش‌های پیشنهادی می‌باشد) صحت کل الگوریتم‌های تولیدی خود (یعنی الگوریتم‌های ۳، ۴، ۵ و ۶) بررسی شد.

همان طور که در جدول ۲ آمده است، نتایج هر دو الگوریتم کاملاً با هم منطبق بوده‌اند.

برای مقایسه پیچیدگی زمانی از سه جعبه جانشانی تولید شده با اندازه های مختلف (دو جعبه بزرگ‌تر با نرم‌افزار سیج [1] تولید شده‌اند) استفاده شده است. هر کدام از نتایج عددی به دست آمده (شکل‌های ۲ و ۳)، میانگینی از چند بار اجرای الگوریتم‌ها است. در جدول ۳ مشخصات سامانه محاسباتی که ارزیابی‌ها روی آن انجام شده، معرفی شده است.

در ابتدا الگوریتم والش سریع به صورت موازی بر روی تعداد نخ‌های مختلف و به دفعات پیاده شده است و میانگین زمان اجرا برای تعداد نخ مختلف، در نمودار شکل ۲ آمده است. در این نمودار دیده می‌شود که پیاده‌سازی با ۱۰ نخ بیشترین کاهش زمان اجرا را در بین تعداد نخ‌های دیگر دارد. دلیل بهتر شدن زمان اجرا با ۱۰ نخ این است که با تعداد نخ‌های کمتر سطح موازی‌سازی ما کمتر می‌شود و با بیشتر از ۱۰ نخ، سیستم‌عامل برای اجرای نخ‌های دیگر اجباراً برخی از نخ ها را به خواب می‌برد و دوباره باز می‌گرداند، که این کار باعث افزایش زمان اجرا می‌شود. بنابراین بهترین تعداد نخ برای کاهش زمان اجرا از دو پارامتر به دست می‌آید که اولی مربوط به منابع سیستمی در دسترس و پارمتر بعدی میزان منابع مصرفی الگوریتم‌ها می‌باشد.

[1] Sage



شکل ۲. زمان اجرای الگوریتم ۴ بر روی تعداد نخ‌های مختلف

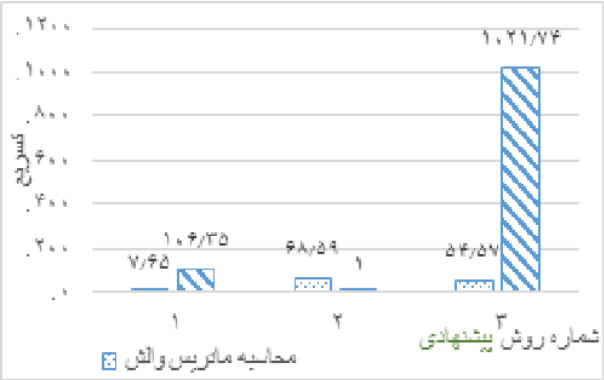
در جدول ۴ زمان اجرای الگوریتم‌های والش سریع و محاسبه درجه غیرخطی جعبه‌های جانشانی ۱۰ × ۱۰، ۱۲ × ۱۲ و ۱۶ × ۱۶ برحسب میلی‌ثانیه، برای روش های اولیه و روش‌های پیشنهادی آمده است. شکل ۳، نشان‌دهنده میزان کاهش زمان اجرا برای هر یک از روش‌های جدول ۴ است که برابر با نسبت زمان اجرای الگوریتم (روش) اولیه به زمان اجرای الگوریتم (روش‌های) پیشنهادی برای جعبه ۱۶ × ۱۶ است. این عدد همیشه بزرگ‌تر از یک است که نشان‌دهنده سرعت بیشتر روش‌های پیشنهادی ۱، ۲ و ۳ ما نسبت به روش اولیه است.

با توجه به شکل ۳ و جدول ۴ دیده می‌شود با استفاده از روش اول پیشنهادی (استفاده از ترانهاده ماتریس والش به جای ماتریس والش)، تعداد جابه‌جایی‌های بین حافظه و تغییر اشاره‌گرها به طور نمایی کاهش یافته است و همانطور که بخش ۱.۵ شرح داده‌ایم این کاهش به این علت می‌باشد که مانع از فراخوانی چندین باره یک آرایه می‌شود چون سیستم عامل در هنگام خواندن یک درایه از ماتریس کل آرایه سطر آن را به داخل حافظه اصلی می‌آورد. در این صورت اگر در الگوریتم از ماتریس والش استفاده کنیم در دور بعدی حلقه، درایه بعدی در سطر بعدی است. بنابراین باید برای فقط یک درایه کل آرایه سطر را داخل حافظه آورد و این زمان بسیار زیادی را تلف می‌کند؛ در صورتی که در الگوریتم‌های ۱ و ۲ اگر از ترانهاده ماتریس والش استفاده کنیم برای هر درایه، کل آرایه سطر به داخل جافظه می‌آید و درایه بعدی مربوط به دور بعدی حلقه در همان آرایه سطر درایه قبلی می‌باشد. بنابراین به اندازه تعداد سطرها در زمان صرفه‌جویی می‌کنیم. به عبارتی برای این دو الگوریتم به اندازه $\mathcal{O}(2^m)$ تسریع در زمان اجرا با استفاده از روش پیشنهادی داریم.

سوالی که در ذهن خواننده ممکن است به وجود بیاید مقاسیه زمان اجرا محاسبه درجه غیرخطی با استفاده از ترکیب روش ۱ و ۲ و با استفاده از ترکیب روش‌های ۱، ۲ و ۳ می‌باشد به این صورت که برای محاسبه

جدول ۴. زمان اجرای روش‌های پیشنهادی برحسب میلی‌ثانیه

روش		الگوریتم اولیه (روش اولیه)	(روش ۱)	ترکیب روش ۱ و ۲	الگوریتم (۶) (ترکیب روش ۱ ، ۲ و ۳)
زمان اجرا تبدیل والش (میلی‌ثانیه)	۱۰ × ۱۰	۱۶۰	۴۸	۱۰	۱۳
	۱۲ × ۱۲	۱۶۴۱	۴۵۲	۵۳	۸۶
	۱۶ × ۱۶	۲۴۹۶۷۲۶	۳۲۶۳۱۱	۳۶۳۹۸	۴۵۷۴۶
زمان اجرا محاسبه درجه غیرخطی (میلی‌ثانیه)	۱۰ × ۱۰	۳۷۲	۶	۱	۱۳
	۱۲ × ۱۲	۲۷۳۶	۳۱	۶	۸۶
	۱۶ × ۱۶	۴۶۷۴۱۶۲۷	۲۰۱۹۰۷	۲۱۳۲۸	۴۵۷۴۷



شکل ۳. میزان بهبود زمان اجرای هرکدام از روش‌ها نسبت به زمان اجرای الگوریتم اولیه برای جعبه ۱۶ × ۱۶

جعبه‌های ۱۰ × ۱۰ و ۱۲ × ۱۲ این زمان بیشتر شده است. باید گفت دلیل اصلی این است که در ستون مربوط به ترکیب روش ۱ و ۲ فقط درجه غیرخطی محاسبه شد و محاسبه ماتریس والش به عنوان پیش پردازش در نظر گرفته شده است، ولی در نتایج ترکیب ۱، ۲ و ۳ ماتریس والش و درجه غیرخطی با هم محاسبه می‌شود. البته برای ۱۶ × ۱۶ این تغییر کمتر است، چون هزینه زمانی ساخت و اجرا و اتمام نخ در مقابل کل زمان اجرا ناچیز است و نشان می‌دهد که با افزیش n و m روش‌های پیشنهادی بهتر نمایان می‌شوند.

زمان اجرای الگوریتم والش برای جعبه جانشانی ۱۶×۱۶ از ۲۴۹۶۷۲۶ میلی‌ثانیه به ۳۲۶۳۱۱ میلی‌ثانیه رسیده است (بهبود زمان اجرا ۷/۶۵ برابری). در ادامه موازی‌سازی آن با ۱۰ نخ این زمان اجرا به ۳۶۳۹۸ میلی‌ثانیه کاهش‌یافته است (نسبت به روش اول پیشنهادی به بهبود ۸/۹۶ برابری رسیده است). با وجود اینکه انتظار داشتیم بعد از موازی‌سازی با ۱۰ نخ به بهبود ۱۰ برابری برسیم، اما به دلیل اینکه اولاً سیستم‌عامل نخ‌ها را به طور کاملاً همروند اجرا نمی‌کند و ثانیا جابه‌جایی اشاره‌گرهای آرایه‌ها و مبادله خود آرایه‌ها به صورت همروند سربار زمانی دارد، به این نتیجه نرسیدیم. اما می‌توان دید که برای محاسبه ماتریس تبدیل والش با موازی‌سازی و استفاده از ترانهاده ماتریس والش به بهبود زمان اجرای ۶۸/۵۹ برابری رسیدیم.

همچنین در محاسبه درجه غیرخطی، زمان اجرای الگوریتم ۱ برای جعبه جانشانی ۱۶ × ۱۶ از ۴۶۷۴۱۶۲۷ میلی‌ثانیه به ۲۰۱۹۰۷ میلی‌ثانیه با اجرای الگوریتم اصلاح‌شده (الگوریتم ۴ ترکیب روش تسریع پیشنهادی ۱ و ۲)، رسانده شده است که نشان‌دهنده بهبود ۲۳۱/۵۰ برابری است. دلیل این بهبود، کاهش مبادله درایه‌های ماتریس والش سریع و تغییر اشاره‌گر است. در ادامه با قرار دادن بخشی از این محاسبه در داخل الگوریتم موازی‌شده والش سریع (الگوریتم ۶ ترکیب کل روش‌های تسریع پیشنهادی)، زمان اجرای آن ۴۵۷۴۷ میلی‌ثانیه شد، که در اینجا نسبت به الگوریتم ۴ به بهبود ۹/۶۰ برابری رسیدیم. به طور کلی با توجه به نتایج، پیاده‌سازی روش‌های پیشنهادی در محیط‌های کم‌حافظه نسبت به الگوریتم‌های اولیه تبدیل والش و درجه غیرخطی بسیار بهتر عمل می‌کند، چون زمانی که از ترانهاده ماتریس والش و روش ترکیب مسائل استفاده می‌کنیم دیگر نیاز به نگهداری چندین ستون و چند ماتریس (برای محاسبه درجه غیرخطی) همزمان در حافظه نمی‌باشد. همچنین برای پردازنده‌های با تعداد پردازنده بیشتر از یکی چه با توان کم چه با توان زیاد با استفاده از روش پردازش موازی، راه حل پیشنهادی مقاله موثرتر می‌باشد.

۷ نتیجه‌گیری

هدف اصلی این مقاله کاهش زمان اجرای الگوریتم‌های تبدیل والش سریع و محاسبه درجه غیرخطی برای جعبه‌های جانشانی بزرگ است. برای تحقق این هدف سه روش پیشنهادی ارائه شد که در بهترین انتخاب برای محاسبه تبدیل والش سریع، با توجه به جدول ۴ به بهبود زمان اجرای ۶۸/۵۹ برابری رسیدیم و برای محاسبه درجه غیرخطی با استفاده از ترانهاده ماتریس والش به بهبود ۱۰۲۱/۷۴ برابری رسیدیم .همان طور که مشاهده شد با استفاده از روش‌های پیشنهادی این مقاله ارزیابی درجه غیرخطی جعبه‌های جانشانی در زمان بسیار کمتری انجام می شود و در نتیجه می‌توان تعداد بسیاری زیادی از جعبه‌ها را در مدت زمان کمتری بررسی کرد.

در ادامه دو پیشنهاد برای ادامه این تحقیقات ارائه می‌شود. اولین پیشنهاد اجرای روش‌های مشابه برای محاسبه ویژگی‌های دیگر جعبه‌های جانشانی از قبیل متوازن بودن[1]، معیار انتشار[2]، استقلال بیتی و معیار

[1]balanced [2]propagation criterion

ایمنی همبستگی[1] است که از تبدیل والش استفاده می‌کنند. با توجه به امکانات بسیار زیاد برای موازی‌سازی روی پردازنده‌های گرافیکی، دومین پیشنهاد تلاش برای پیاده‌سازی راه‌حل‌های پیشنهادی مقاله بر روی GPU به منظور ارزیابی جعبه‌های جانشانی است.

مراجع

[1] Claude E Shannon. Communication theory of secrecy systems. *The Bell system technical journal*, 28(4):656–715, 1949.

[2] Emmanuel Prouff. Dpa attacks and s-boxes. In *International Workshop on Fast Software Encryption*, pages 424–441. Springer, 2005.

[3] F Arute et al. Quantum supremacy using a programmable superconducting processor. nature https://doi.org/10.1038/s41586-019-1666-5 (2019).

[4] Kamsiah Mohamed, Fakariah Hani Hj Mohd Ali, Suriyani Ariffin, Nur Hafiza Zakaria, and Mohd Nazran Mohammed Pauzi. An improved aes s-box based on fibonacci numbers and prime factor. *Int. J. Netw. Secur.*, 20(6):1206–1214, 2018.

[5] Bernardo Amaral, Stanislav Darachev, Maximilian Ludvigsson, Ronak Molazem, and Alejandro Molero Casanova. S-box analysis. *ECMI modelling week, July*, pages 17–24, 1724.

[6] Raja Muthalagu and Subeen Jain. Modifying the structure kasumi to improve its resistance towards attacks by inserting fsm and s-box. *Journal of Cyber Security Technology*, 2(1):37–50, 2018.

[7] Christopher A Wood. *Large substitution boxes with efficient combinational implementations*. Rochester Institute of Technology, 2013.

[8] Paulo Barreto, Ventzislav Nikov, Svetla Nikova, Vincent Rijmen, and Elmar Tischhauser. Whirlwind: a new cryptographic hash function. *Designs, codes and cryptography*, 56(2):141–162, 2010.

[9] Musheer Ahmad, Farooq Seeru, Ahmed Masihuddin Siddiqi, and Sarfaraz Masood. Dynamic 9× 9 substitution-boxes using chaos-based heuristic search. In *Soft Computing: Theories and Applications*, pages 839–851. Springer, 2018.

[10] ET Ososanya, Jung-Chi Chen, and Reyes Poo. Performance evaluation of parallel fast walsh transform algorithms on a shared-memory multiprocessor computer. In *Proceedings of 26th Southeastern Symposium on System Theory*, pages 562–566. IEEE, 1994.

[11] A Youssef, S Tavares, S Mister, and C Adams. Linear approximation of injective s-boxes. *Electronics letters*, 31(25):2165–2165, 1995.

[12] Sugata Gangopadhyay, Goutam Paul, Nishant Sinha, and Pantelimon Stănică. Generalized nonlinearity of s-boxes, 2018.

[13] Pedro Miguel Sosa. Calculating nonlinearity of boolean functions with walsh-hadamard transform. *UCSB, Santa Barbara*, pages 1–4, 2016.

[14] Joao Andrade, Gabriel Falcao, and Vitor Silva. Optimized fast walsh–hadamard transform on gpus for non-binary ldpc decoding. *Parallel Computing*, 40(9):449–453, 2014.

[15] Stjepan Picek, Lejla Batina, Domagoj Jakobović, Barış Ege, and Marin Golub. S-box, set, match: a toolbox for s-box analysis. In *IFIP International Workshop on Information Security Theory and Practice*, pages 140–149. Springer, 2014.

[16] Dusan Bikov and Ilija Bouyukliev. Parallel fast walsh transform algorithm and its implementation with cuda on gpus. *Cybernetics and Information Technologies*, 18(5):21–43, 2018.

[17] Stjepan Picek. *Applications of evolutionary computation to cryptology*. PhD thesis, University of Zagreb. Faculty of Electrical Engineering and Computing …, 2015.

[18] Claude Carlet, Yves Crama, and Peter L Hammer. Vectorial boolean functions for cryptography. In *Boolean Models and Methods in Mathematics*. Computer Science and Engineering, 2013.

[19] Ilija Bouyukliev and Dusan Bikov. Applications of the binary representation of integers in algorithms for boolean functions. In *Proceedings of the Forty Fourth Spring Conference of the Union of Bulgarian Mathematicians*, number 44, pages 161–166, 2015.

[20] Linda Dee Burnett. *Heuristic optimization of Boolean functions and substitution boxes for cryptography*. PhD thesis, Queensland University of Technology, 2005.

[21] Claude Carlet and Cunsheng Ding. Nonlinearities of s-boxes. *Finite fields and their applications*, 13(1):121–135, 2007.

[1] correlation immunity