TEEGES

Technovations in Electrical Engineering & Green Energy System

Research Article (2022) 1(3):27-39

Optimal Hardware Accelerator Design for Implementation of BLAKE2b Hash Function Algorithm

Mohsen Dadkhah [1], *MSc Student*, Atefeh Salimi[1], *Assistant Professor*,
Nadia Hajikhiadani [2], *Assistant Professor*

[1] *Department of Electrical Engineering, Isfahan (Khorasgan) Branch, Islamic Azad University, Khorasgan, Isfahan, Iran*
[2] *Electrical Engineering Department, Arak University of Technology, Arak, Iran*

Abstract:

Recently, there has been a surge in the popularity of cryptocurrencies, which are digital currencies that enable transactions through a decentralized consensus mechanism. In this paper, one of the most effective Equihash algorithms subcategories, known as BLAKE2, is presented, and then effort has been made to optimize the compression function as one of the main and most challenging blocks of the BLAKE2 algorithm. In addition, by cognitive partitioning the algorithm between the software/hardware parts of the device, efforts have been made to improve the speed and the number of resource usage. For comparison, implementation was carried out with high-level vs HDL design methods for full and semi-parallel structures. All three methods were implemented using Vivado tools exploiting ZC706 evaluation board. The implementation results indicated that the number of resource usage (LUT/FF) and power consumption of the proposed structure is equal to (6575/4726) and 0.316(W) respectively Which has created a significant reduction compared to other methods. Moreover, the hash rate and the energy efficiency of the proposed structure are equal to 50 MHash/s and 6.3 ($nJ/Hash$) respectively.

Keywords: Blockchain, Equihash, Cryptocurrency, BLAKE2b, SoC.

Received: 06 March 2022
Revised: 28 June 2022
Accepted: 26 July 2022
Corresponding Author: Dr. Atefeh Salimi, Atefeh.salimi@gmail.com
DOI: http://dx.doi.org/10.30486/TEEGES.2022.1960348.1019

نشریه فناوری‌های نوین در مهندسی برق و سیستم انرژی سبز

TEEGES

..مقاله پژوهشی

طراحی شتاب دهنده سخت افزاری بهینه برای پیاده سازی الگوریتم تابع درهم ساز BLAKE2b

محسن دادخواه[1]، *دانشجوی کارشناسی ارشد*، عاطفه سلیمی[1]، *استادیار*، نادیا حاجی خیادانی[2]، *استادیار*

۱- دانشکده مهندسی برق، واحد اصفهان (خوراسگان)، دانشگاه آزاد اسلامی، خوراسگان، اصفهان، ایران

۲- دانشکده مهندسی برق، دانشگاه صنعتی اراک، اراک، ایران

چکیده: در سال های اخیر رمز ارزها به عنوان ارزهای دیجیتال که از مکانیسم اجماع غیرمتمرکز برای تراکنش ها استفاده می کنند، بسیار مورد توجه قرار گرفته اند. بهینه سازی پیاده سازی الگوریتم های در هم ساز مورد استفاده در کاربردهای بلاکچین به منظور بهبود سرعت و توان مصرفی بسیار حائز اهمیت است. در این مقاله الگوریتم در هم ساز BLAKE2b از بین الگوریتم های موجود انتخاب و بهینه سازی سخت افزاری آن مورد بررسی قرار گرفت. بهینه سازی تابع فشرده- ساز (F) این الگوریتم به عنوان اصلی ترین و چالش برانگیزترین بلوک الگوریتم در این مقاله انجام شده است. علاوه بر این با تقسیم هوشمندانه الگوریتم بین نرم افزار/سخت افزار تلاش برای افزایش سرعت و نیز کاهش تعداد منابع مصرفی شده است. برای مقایسه، پیاده سازی با روش های طراحی سطح بالا و همینطور روش ساختار موازی نیز انجام شد. هر سه روش طراحی شده با استفاده از نرم افزار Vivado با برد توسعه ZC706 پیاده سازی شدند. نتایج نشان می دهد که تعداد منابع مصرفی (FF/LUT) و توان مصرفی روش پیشنهاد شده به ترتیب برابر (۴۷۲۶/۶۵۷۵) و(W) ۰/۳۱۶ است که کاهش قابل توجهی در مقایسه با سایر روش ها ایجاد کرده است. همینطور سرعت انجام در هم سازی و انرژی بر واحد تعداد در هم سازی انجام شده برای ساختار پیشنهادی به ترتیب برابر ۵۰ MHash/s و (nJ/Hash) ۶/۳ است.

واژه های کلیدی: زنجیره بلوکی، عصاره پیام، رمزنگاری، الگوریتم BLAKE2b، سیستم بر تراشه.

تاریخ ارسال مقاله: ۱۴۰۰/۱۲/۱۵

تاریخ بازنگری مقاله: ۱۴۰۱/۰۴/۰۷

تاریخ پذیرش مقاله: ۱۴۰۱/۰۵/۰۴

نویسنده‌ی مسئول: دکتر عاطفه سلیمی، Atefeh.salimi@gmail.com

DOI: http://dx.doi.org/10.30486/TEEGES.2022.1960348.1019

۱- مقدمه

ارتباطات کامپیوتری امروزه از ضروریات هر سازمانی به حساب می آید. با گسترش روز افزون استفاده مردم و سازمان ها از کامپیوترها و نیاز به اشتراک گذاری منابع اطلاعاتی و همچنین نیاز مدیران ارشد دولتها و سازمانها برای تبادل اطلاعات و جدیدا تراکنشهای بانکی و مالی مانند رمز ارزها بحث اشتراک گذاری اطلاعات را بصورت روز افزون مطرح می نماید. ولی درکنار این مزایا، تبادل اطلاعات خطرات زیادی را در پی دارد که می تواند باعث ایجاد مشکلاتی مانند از بین رفتن یا وارد شدن زیان اقتصادی به کسب و کارها، فاش شدن اطلاعات مهم سازمان ها و دولت ها یا به سرقت رفتن اطلاعات پولی و مالی شود. بنابراین یکی از عناصر تبادل اطلاعات بحث امنیت است که امری مهم و غیر قابل اجتناب می باشد.

تکنیک رمزنگاری و توابع درهم ساز در اواخر دهه ۱۹۸۰ بعنوان یکی از راهکارهای تامین امنیت و جلوگیری از نفوذ به سیستم های اطلاعاتی به منظور کاربرد در طرح های امضای دیجیتال معرفی شدند. از آن زمان تا کنون توابع درهم ساز زیادی پیشنهاد شده و برخی از آنها مورد استفاده بسیاری در پلتفرم های مطرح تبادل و ذخیره اطلاعات در زمینه های تشخیص هویت، اطمینان از صحت و تمامیت، حفظ محرمانگی و صیانت از اطلاعات قرار گرفته اند. اما مانند بسیاری از الگوریتم ها و مسائل دیگر درحیطه رمزنگاری، این توابع نیز با چالش های مختلفی روبه رو بوده اند و مسائل و ابهاماتی در ارتباط با انواع آن ها وجود دارد. توابع در هم ساز یکی از توابع رایج در زمینه رمزنگاری می باشند. درهم سازی داده ها یک فرآیند یک طرفه است که در آن هر طول داده ی ورودی در نهایت تبدیل به یک رشته داده خروجی با یک اندازه ی ثابت می شود که به آن مقدار تابع درهم سازی یا عصاره[۱] پیام گفته می شود. این قابلیت این امکان را به ما می دهد که یکپارچگی داده های ورودی را بررسی کنیم. الگوریتم درهم ساز یک عمل خلاصه سازی را روی جریان ورودی انجام می-دهد. با توجه به استفاده از توابع درهم ساز در کاربردهای مختلف، نیاز به انعطاف پذیری و ارائه بازدهی های متفاوت از توابع درهم سازی احساس می شود.

یکی ازمهمترین نکات در زمینه امنیت اطلاعات به طور عام و توابع درهم ساز به طور خاص بحث ایجاد تعادل بین سطح امنیت و امکان پیاده سازی بهینه الگوریتم های رمزنگاری می باشد. از این رو شاخصهایی چون احتمال تولید عصاره یکسان برای ورودی های متفاوت و میزان مقاومت در برابر حملات مختلف هکرها در بحث امنیت و همچنین سرعت تولید عصاره و امکان انطباق با منابع سخت افزاری، توان، بازدهی و میزان منابع سخت افزاری استفاده شده همواره مورد توجه استفاده کنندگان الگوریتم های مختلف بوده است. علاوه بر موارد فوق که عمدتا مسائل را از دیدگاههای ریاضی و آماری و بعضا تجربی مورد تجزیه و تحلیل قرار می دهد، در بحث پیاده سازی الگوریتم مسائل تکنیکی دیگری که مستقیما با علم الکترونیک مرتبط است مطرح می شود که عدم توجه به آنها می تواند موجب تضعیف قدرت الگوریتم یا حتی کسب نتیجه ای مخالف آنچه در طراحی الگوریتم مورد نظر بوده است شود. بعنوان مثال استفاده از سخت افزار نامناسب یا عدم وجود یک کد نویسی سازگار با الگوریتم می تواند در نهایت موجب تاخیر در ایجاد عصاره پیام شده و این تاخیر می تواند زمان مناسبی در اختیار هکرها برای اجرای مقاصد خود بگذارد. همچنین عدم توجه به ملاحظات الکتریکی مانند توان و فرکانس می تواند موجب تولید گرما و کاهش عمر مفید سیستم شده یا بحث اتلاف انرژی در مقیاس بالا را مطرح نماید.

استخراج رمزارزها در مراحل اولیه توسط ریزپردازنده[۲] (CPU) انجام می شد. بعدها برای افزایش سرعت استخراج عصاره واحدهای پردازنده گرافیکی[۳] (GPU) ها مورد استفاده قرار گرفتند[۱]. به مرور استخراج با استفاده از آرایه های دریچه ای برنامه پذیر[۴] (FPGA) و همینطور چیپ های خاص منظوره[۵] (ASIC) بر استفاده از GPU فزونی گرفتند. امروزه مزارع ASIC برای استخراج رمز ارزها بسیار مورد استفاده قرار می گیرند. از آنجایی که دسترسی به شرایط استخراج غیر متمرکز اهمیت زیادی دارد [۲]، الگوریتم های جدید پیشنهاد شده برای توابع استخراج عصاره که توابع در هم ساز نیز گفته می شوند بیشتر به صورت سریال و پارامتری و حافظه دار هستند. با تغییر پارامترهای مربوط به این الگوریتم ها تراشه های ASIC ساخته شده نیز منسوخ و غیر قابل استفاده می شوند. با توجه به قیمت بالای ساخت تراشه استفاده از پلتفرم های قابل جایگزین نظیر GPU ها یا FPGA ها برای پیاده سازی الگوریتم های در هم ساز اهمیت پیدا می کند [۳]. از آنجایی که FPGA ها نسبت به GPU ها توان مصرفی پایین تری دارند و همینطور به دلیل قابلیت های انعطاف پذیری، قابلیت پیکربندی و در دسترس بودن آن ها برای عموم انتخاب مناسبی برای پیاده سازی الگوریتم های در هم ساز مورد استفاده در کاربردهای بلاکچین می باشند.

کلاس های مختلفی از توابع در هم ساز در مراجع پیشنهاد شده است که مرسوم ترین الگوریتم های در هم ساز استفاده شده الگوریتم در هم ساز امن[۶] (SHA) [۴, ۵]، الگوریتم چکیده پیام[۷] (MD5) [۶]، الگوریتم Keccak [۷, ۸] و الگوریتم BLAKE2 [۹, ۱۰] است. در این مقاله از بین توابع در هم ساز موجود، تمرکز بر روی پیاده سازی الگوریتم BLAKE2 و بطور خاص ورژن BLAKE2b [۱۱] است. الگوریتم BLAKE2 یکی از انواع توابع در هم ساز است که برای پیاده سازی نرم افزاری/ سخت افزاری سریع طراحی شده است. ورژن BLAKE2b دارای ساختار پردازشی ۶۴ بیتی است و بدلیل استفاده در الگوریتم های چند تابعی سریال و پارامتری نظیر X15 [۱۲]، X17 [۱۳] و Lyra2 [۳, ۱۴, ۱۵] پیاده سازی بهینه آن همواره مورد توجه طراحان بوده است. در این مقاله با بهینه سازی تابع F الگوریتم BLAKE2b و کاهش تعداد توابع مخلوط مربوط به الگوریتم و طراحی یک ماشین حالت محدود[۸] (FSM) برای کنترل و همزمانی مقادیر عصاره خروجی سعی در بهینه سازی الگوریتم در هم سازی BLAKE2b از نقطه نظر سرعت و توان مصرفی شده است. علاوه بر این با تقسیم بین نرم افزار/ سخت افزار نیز سعی در افزایش سرعت و نیز کاهش تعداد منابع مصرفی قسمت منطق قابل برنامه ریزی قطعه مورد استفاده شده است که ساختار آن در ادامه این مقاله توضیح داده می شود.

در بخش دوم این مقاله در مورد جزئیات الگوریتم در هم ساز BLAKE2b و نحوه عملکرد آن توضیحاتی داده می شود. در این بخش با ارائه فلوچارت طراحی و پیاده سازی الگوریتم در مورد نحوه طراحی و پیاده سازی سخت افزاری/ نرم افزاری بخش های مختلف الگوریتم نیز توضیح داده می شود. در بخش ۴ نتایج پیاده سازی الگوریتم با دو روش طراحی سخت افزاری و نیز طراحی سطح بالا ارائه می شود و روش های مختلف مورد مقایسه قرار می گیرند. و در بخش آخر نیز نتیجه گیری کلی از انجام مقاله و نیز نتایج به دست آمده از پیاده سازی الگوریتم با کمک گرفتن از ابزار Vivado و با در نظر گرفتن برد توسعه ZC706 بیان می شود.

۲- ساختار الگوریتم BLAKE2

خانواده BLAKE2 در سال ۲۰۱۲ معرفی شدند و هدف از طراحی این خانواده جایگزینی الگوریتمهای MD5 و SHA-1 در کاربردهایی است که به کارایی بالایی نیاز می باشد. علی رغم اینکه پیاده سازی این الگوریتم نیاز به منابع کمی دارد، این الگوریتم در حوزه پیاده سازی سخت افزاری و نرم افزاری سرعت بالایی دارد. الگوریتم BLAKE2 شامل یک تابع F با امکان استفاده از یک کلید مشترک جهت بالا بردن سطح امنیت است. این الگوریتم ورژن های مختلفی دارد که از آن بین ویرایش BLAKE2b که دارای پلتفرم ۶۴ بیتی است در این مقاله برای پیاده سازی مورد توجه قرار گرفته است. ویرایش های دیگر این الگوریتم مانند BLAKE2s که مناسب کاربردهای ۳۲ بیتی و همینطور ویرایش های BLAKE2bp که مختص پردازش های موازی و BLAKE2sp که برای بالابردن کارایی طرح روی پردازنده های چند هسته ای طراحی شده اند نیز ارایه شده که مورد بحث این مقاله نمی باشند. [۱۶, ۱۷]

جدول (۱): جدول تعریف پارامترهای اصلی تابع BLAKE2b

ردیف	پارامتر	نام کامل	تعداد	تعریف عملکرد
۱	*w*	Bits in words	۶۴ بیت	تعداد بیت در هر کلمه پردازش شده
۲	*r*	Rounds in F	۱۲ دور	تعداد تکرار عمل چرخش در تابع F
۳	*bb*	Block bytes	۱۲۸	تعداد بایت مورد پردازش در هر بلاک از پیام
۴	*nn*	Hash bytes	۱<=nn<=۶۴ بایت	سایز خروجی عصاره پیام بر اساس بایت
۵	*kk*	Key bytes	۰<=kk<=۶۴ بایت	سایز کلید امنیتی اعمال شده بر اساس بایت
۶	*ll*	Input bytes	۰<=ll<=$۲^{۱۲۸}$	سایز طول پیام بر اساس بایت
۷	*tt*	Message byte offset	۱<=tt<=$۲^{۱۲۸}$	شمارنده تعداد بایت پیام از ابتدا تا آخر بلوک جاری
۸	*ff*	Last flag indicator	۱۲۸	پرچم نشان دهنده آخرین پیام
۹		G Rotation Constants	32,24,16,63	ثابت تعداد چرخش تابع G برای R1/R2/R3/R4

۱-۲- ساختار الگوریتم BLAKE2b

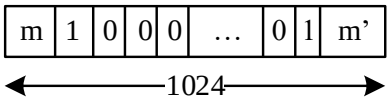
همانطور که در قسمت قبل توضیح داده شد، الگوریتم BLAKE2b یکی از ویرایش های الگوریتم BLAKE2 است که مناسب پردازشگرهای ۶۴ بیتی و تک هسته ای می باشد. در این قسمت به منظور پیاده سازی بهینه این الگوریتم ضمن ارائه یک دید کلی از ساختار الگوریتم، قسمت های مختلف آن را بررسی می نماییم. در ابتدا مراحل مختلف الگوریتم در هم سازی BLAKE2b برای پیغام m توضیح داده می شود. برای این منظور قبل از ادامه معرفی الگوریتم لازم است تا پارامترهای مورد استفاده در الگوریتم BLAKE2b را طبق جدول (۱) تعریف و تعداد بیتهای آنرا مشخص نماییم:

همچنین ثابتها و پارامترهای واسطه زیر در محاسبات توابع و خروجی این الگوریتم طبق تعریف از آنها استفاده می شود که در جدول (۲) به آنها اشاره شده است:.

جدول (۲): جدول تعریف پارامترهای واسطه تابع BLAKE2b

ردیف	پارامتر	نام کامل	عملکرد
۱	*IV [0....7]*	Initializing Vector constant	بردار مقادیر ثابت برای مقداردهی اولیه
۲	*SIGMA [0....9]*	Message word permutation constant	ثابتهای جایگشتی کلمه های پیام
۳	*p [0....7]*	Parameter block (Bytes)	بلوک پارامتر شامل طول عصاره و کلید (بایت)
۴	*m [0....15]*	16 words of each message	تعداد ۱۶ کلمه ۶۴ بیتی از هر پیام ۱۰۲۴ بیتی
۵	*h [0....7]*	Internal state of the hash	حالتهای میانی عصاره
۶	*d [0....dd-1]*	Padded input blocks	بلوکهای پیام لایه گذاری شده ۱۲۸بیتی

همانطوری که در جدول (۱) نشان داده شده است الگوریتم BLAKE2b توانایی پردازش پیام هایی با حداکثر طول 2^{128} بایت (ll) را دارد که در هنگام پردازش آن، پیام مورد نظر را به بلوکهای ۱۰۲۴ بیتی (۱۲۸ بایتی) (bb) که هر بلوک را به ۱۶ کلمه ۶۴ بیتی (w) تبدیل می نماید تا با پردازنده مورد نظر سازگاری داشته باشد و در نهایت عصاره های با طول ۱ تا ۶۴ بایت (nn) تولید می نماید. اولین مرحله مشابه سایر الگوریتم های در هم سازی، لایه گذاری[۹] پیغام ورودی است. لایه گذاری به منظور سازگار کردن تعداد بیت ورودی با تعداد بیت مورد نیاز برای پردازش است. برای این منظور مشابه شکل (۱) تعدادی صفر و یک به پیغام ورودی m اضافه می شود بطوریکه طول کلی پیغام برابر ۱۰۲۴ بیت و ۱۲۸ بیت آخر آن که با m' نمایش داده شده است با عدد طول پیغام جایگذاری می شود. لازم به ذکر است که از این پس در این الگوریتم برای نمایش و انجام محاسبات اعداد باینری از سیستم اعداد مبنای شانزده که قسمت پر ارزش آن در ابتدا و قسمت کم ارزش آن در انتها قرار گرفته[۱۰] استفاده شده است.

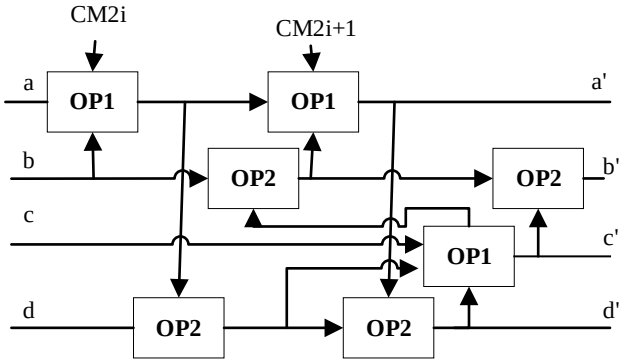


شکل (۱): لایه گذاری سیگنال پیغام

در مرحله بعد باید عملیات فشرده سازی[۱۱] پیغام لایه گذاری شده انجام شود. تابع فشرده سازی الگوریتم BLAKE مشابه تابع در هم ساز خانواده LAKE [۱۸] است. فشرده سازی شامل سه مرحله مقدار دهی اولیه[۱۲]، اعمال توابع مخلوط[۱۳] (G) و نهایی سازی[۱۴] است. الگوریتم BLAKE برخلاف الگوریتم ChaCha [۱۹] با اعمال جایگشت پیغام ورودی بر مبنای مقادیر رند شده از مشابهت بین نتایج خروجی جلوگیری می نماید. این مساله موجب می شود تا از حملاتی نظیر حملات لغزشی[۱۵] [۲۰, ۲۱] جلوگیری شود.

در مرحله مقدار دهی اولیه، بردارهای حالت های داخلی[۱۶] الگوریتم (*V [0....15]*) مقدار دهی اولیه می شوند و در مرحله بعد وارد توابع G می شوند.

۲-۲- شرح عملکرد تابع G :



شکل (۲): ساختار تابع مخلوط G

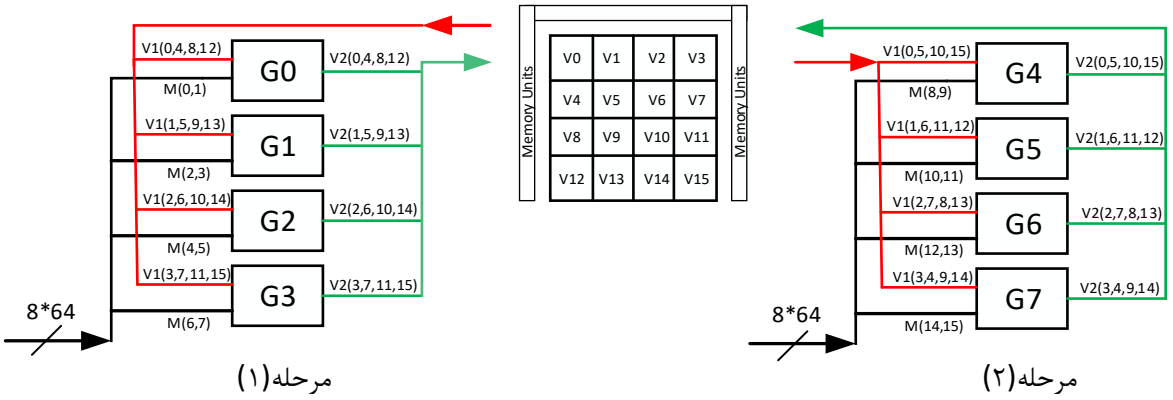
تابع G شامل تعدادی از توابع است که ساختار و الگوریتم آن به ترتیب در شکلهای (۲) و (۳) نشان داده شده است. در ساختار نشان داده شده در شکل (۲) بلوک های OP1 شامل دو جمع کننده و بلوک OP2 شامل توابع XOR و شیفت است. تعداد شیفت های مورد نیاز برای هر بلوک OP2 استفاده شده در ساختار متغیر است که در هنگام استفاده در طرح برای آن تعیین می شود. تعداد شیفت های هر مرحله در الگوریتم شکل (۳) مشخص شده است و در جدول (۱) نیز با پارامترهای *R1* تا *R4* از پیش تعریف شده اند. پارامترهای a، b، c و d ورودی های تابع G هستند. *CM* نیز مقادیر ثابت هستند و σ_i بردارهای جایگشت مربوط به تابع الگوریتم را مشخص می کنند. تعداد ۱۰ بردار ۱۶ تایی جایگشت σ_i که در جدول (۲) با *SIGMA [0....9]* تعریف شده است برای الگوریتم در نظر گرفته می شود که به ازای مقادیر رند r در هر بار اجرای الگوریتم مورد استفاده قرار می گیرند. با توجه به اینکه ۱۲ مقدار رند برای الگوریتم BLAKE2b مورد استفاده قرار می گیرد و در جدول ارائه شده ۱۰ مقدار بیشتر وجود ندارد، برای مقادیر r بزرگتر از ۹ باقیمانده r بر عدد ۱۰ به عنوان پارامتر رند برای انتخاب بردار جایگشت σ_i مورد استفاده قرار می گیرد.

$$
\begin{aligned}
&Function\ G(V[0\ldots15], a, b, c, d, x, y)\\
&\quad a \leftarrow a + b + \left(m_{\sigma r(2i)} \oplus CM_{\sigma r(2i+1)}\right)\\
&\quad d \leftarrow (d \oplus a) \gg 32\\
&\quad c \leftarrow c + d\\
&\quad b \leftarrow (b \oplus c) \gg 25\\
&\quad a \leftarrow a + b + \left(m_{\sigma r(2i+1)} \oplus CM_{\sigma r(2i)}\right)\\
&\quad d \leftarrow (d \oplus a) \gg 16\\
&\quad c \leftarrow c + d\\
&\quad b \leftarrow (b \oplus c) \gg 11
\end{aligned}
$$

شکل (۳): الگوریتم تابع G

در هنگام پیاده سازی الگوریتم به دلیل پیچیدگی این ساختار و وابسته بودن شروع اجرای عملیات در هر بلوک به خروجی های بلوک های قبل، مسیر بحرانی این طرح پیچیده و محدود کننده حداکثر فرکانس کلاک[۱۷] طرح می باشد. همانطوری که در شکل (۲) مشخص است مسیر بحرانی شامل چهار بلوک OP1 و شش بلوک OP2 است که با استفاده از تکنیک پایپلاینینگ می توان فرکانس کلاک طرح را تا حد زیادی بهبود بخشید.

طبق شکل (۴) در مجموع تعداد ۸ تابع G برای به روز کردن مقادیر حالتهای داخلی *V [0....15]* نیاز است و در این الگوریتم به روز کردن مقادیر مربوط به حالتهای داخلی به صورت یک ماتریس ۴*۴ انجام می شود که توابع مخلوط G0-G3 برای پر کردن ستون های ماتریس و توابع مخلوط G4-G7 برای پر کردن قطر های ماتریس مورد استفاده قرار می گیرند [۲۲]. به این منظور راهکار ارائه شده در این مقاله این است که برای پیاده سازی می توان توابع G0-G3 را با هم و توابع G4-G7 را نیز با هم در سیکل های متفاوت موازی کرد و بهینه سازی در پیاده سازی ساختار انجام داد.



شکل (۴): پیاده سازی ۸ تابع G در دو گروه ۴ تایی

در مرحله بعد پیغام ورودی به بلوک های ۱۶-کلمه ای m^i تقسیم می شود که در جدول (۲) با $m\ [0....15]$ مشخص شده است. m^i مشخص کننده i امین بلوک ۱۶-کلمه ای از n بلوک پیغام ورودی است. s نیز دیتای تصادفی است که به عنوان ورودی تابع فشرده-ساز مورد استفاده قرار می گیرد. IV بردار مقادیر اولیه تابع است که مقادیر مشابه الگوریتم SHA-512 دارد [۲۳] و در ابتدای الگوریتم در هم سازی تابع کار خود را با این مقادیر اولیه آغاز می نماید. با توجه به شکل (۴ تابع F با شروع از این مقدار اولیه و با در نظر گرفتن بلوک های ۱۶-کلمه ای m^i و همینطور دیتای تصادفی s و l^i که در جدول (۲) با tt نشان داده شده است و شمارنده تعداد بیت های پیغام جاری می باشد شروع می شود و به تعداد کل بلوک های پیغام ورودی الگوریتم را تکرار می کند و در هر مرحله مقادیر زنجیره[۱۸] h را به روز می کند.

$$h^0 \leftarrow IV$$
$$for\ i = o, \dots, N-1$$
$$h^{i+1} \leftarrow compress\ (h^i, m^i, s, l^i)$$
$$return\ h^N$$

شکل (۴): تابع تکرار در هم ساز پیغام لایه گذاری شده

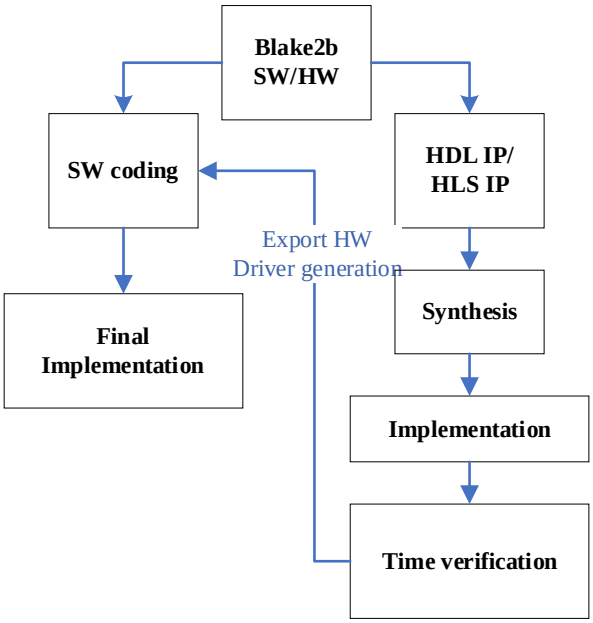
در مرحله آخر، الگوریتم در هم سازی برای محاسبه مقادیر جدید زنجیره $h'_0..h'_7$ که شامل XOR مقادیر اولیه زنجیره $(h_0..h_7)$ و مقادیر حالت های داخلی $(V\ [0....15])$ است که در رابطه (۱) نشان داده شده است.

$$h'_i = h_i \oplus v_i \oplus v_{i+8} \qquad i = 0, \dots, 7 \qquad (۱)$$

با توجه به توضیحات ارائه شده در این بخش در مورد قسمت های مختلف الگوریتم BLAKE2b، در بخش بعد در مورد مراحل طراحی بلوک های مختلف الگوریتم و روش های به کار گرفته شده برای بهینه سازی پیاده سازی توضیح داده می شود.

۳- طراحی سخت افزار ساختار BLAKE2b

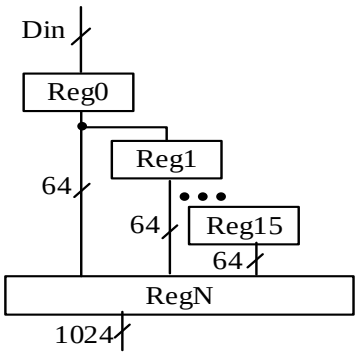
در این قسمت در مورد نحوه طراحی سخت افزار و پیاده سازی الگوریتم BLAKE2b توضیح داده می شود. فلوچارت از طراحی تا پیاده سازی الگوریتم روی برد های سیستم بر تراشه[۱۹] (SOC) در شکل ((۵) نشان داده شده است. همانطوری که مشخص است، این الگوریتم به دو قسمت برای پیاده سازی سخت افزار(HW) و نرم افزار(SW) تقسیم بندی می شود. برای قسمت های سخت افزاری به دو روش می توان ماژول آماده شده[۲۰] (IP) مورد نظر را پیاده سازی کرد. زبان توصیف سخت افزار[۲۱] (HDL) و کد نویسی سطح بالا[۲۲] (HLS) که با استفاده از ابزار سنتز سطح بالا برای پیاده سازی قسمت سخت افزار مورد استفاده قرار می گیرند. هر چند که استفاده از HLS به دلیل استفاده از کد نویسی سطح بالا سرعت پیاده سازی الگوریتم را بالا می برد ولی بدلیل متفاوت بودن ماهیت پیاده سازی سخت افزاری با الگوریتم های سطح بالا نمی توان با این روش بهینه سازی مطلوبی از طرح بدست آورد.



شکل (۶): فلوچارت طراحی و پیاده سازی الگوریتم BLAKE2b

با توجه به شکل (۶) در مرحله بعد از سنتز و پیاده سازی سخت افزار، عملیات انتقال به محیط توسعه نرم افزاری[۲۳] (SDK) انجام می پذیرد. در این محیط کدهای مورد نیاز برای پیاده سازی روی پردازنده SOC با اضافه کردن درایورهای سخت افزار ایجاد شده تولید می شوند. مراحل نوشتن در سخت افزار و نیز تولید خروجی های مورد نیاز طرح و در صورت نیاز نمایش مناسب توسط برد توسعه[۲۴] نیز در کد نویسی نرم افزار در نظر گرفته می شود و در پایان پیاده سازی طرح روی برد انجام می گردد.

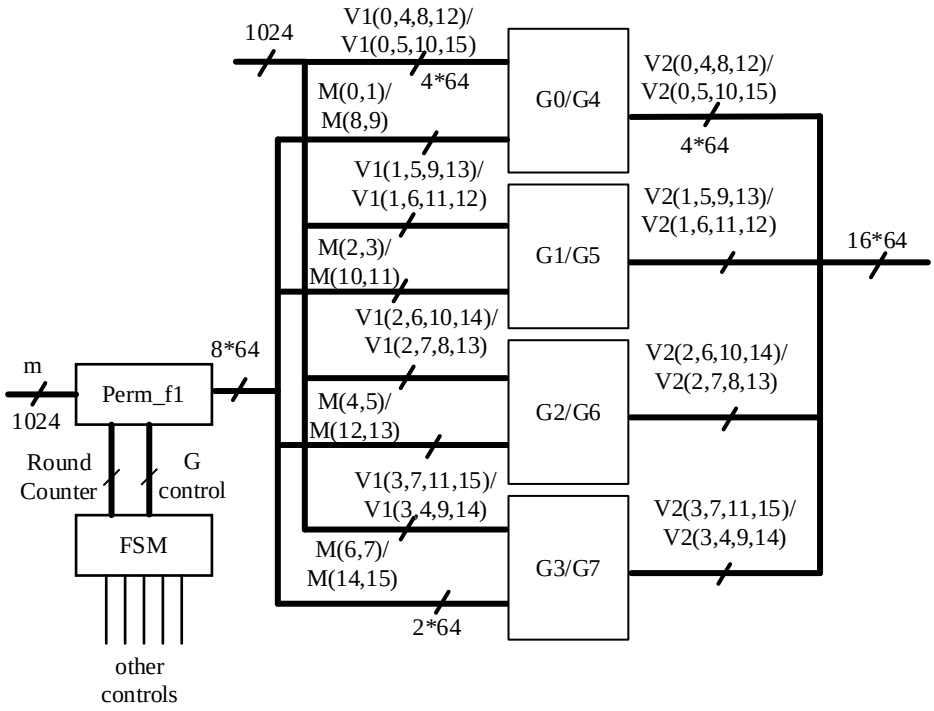
در مرحله بعد به منظور ایجاد قابلیت پردازش موازی مبدل سریال به موازی[۲۵] (SIPO) در مسیر دیتای ورودی استفاده شده است. ساختار این مبدل در شکل (۷) نشان داده شده است. همانطوری که در این شکل دیده می شود دیتای ۶۴ بیتی ورودی با هر کلاک به مبدل وارد و شیفت پیدا می کند. در مجموع ۱۶ خروجی ۶۴ بیتی (۱۰۲۴ بیت) شامل دیتای جدید ورودی و شیفت یافته های دیتاهای ورودی قدیم ایجاد می شود که در Reg N ذخیره می شود.



شکل (۷): مبدل سریال به موازی

در شکل (۸) ساختار سخت افزار پیشنهاد شده برای بلوک مربوط به تابع F نشان داده شده است. همانطور که در قسمت مربوط به ساختار الگوریتم توضیح داده شد وظیفه تابع Perm_f1 ایجاد جایگشت های مورد نیاز برای بلوک های پیغام ورودی و ثابت های الگوریتم *CM* است که در اینجا مجموع آنها با *M(i,j)* نشان داده شده است. در این طرح برای افزایش بازدهی الگوریتم و کاهش منابع با توجه به شکلهای (۴و۸) بجای ۸ تابع G همزمان، از ۴ تابع استفاده شده است. یک FSM نیز برای کنترل عملکرد بلوک های مختلف

تابع F و همین طور مقدار رند r مربوط به الگوریتم طراحی و پیاده سازی شد. با کمک این ماشین حالت جریان داده بین بلوک های مختلف کنترل و پیکربندی می شود.



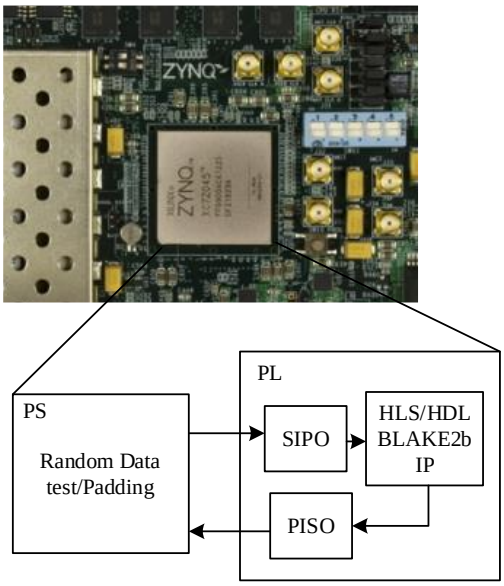
شکل (۸): پیاده سازی تابع G

در مرحله آخر برای تکمیل فرآیند فشرده سازی مرحله نهایی سازی که طبق فرمول (۱) بیان شد بصورت سخت افزاری پیاده سازی می شود تا مقادیر جدید زنجیره h_i' بروز شوند. خروجی های بلوک آخر بعنوان ورودی های تابع نهایی ساز به جای حالت های داخلی $V[0....15]$ مورد استفاده قرار می گیرند. در نهایت بدلیل اینکه خروجی ایجاد شده ۱۰۲۴ بیتی است و سازگار با تعداد پایه های I/O در دسترس قطعه در نظر گرفته شده برای پیاده سازی نیست، در خروجی از مبدل موازی به سریال[۲۶] (PISO) استفاده می شود. با استفاده از مبدل PISO با هر سیکل کلاک خروجی های ۶۴ بیتی تابع در هم ساز ایجاد می شوند.

۴- نتایج پیاده سازی

الگوریتم مورد نظر با برد ZC706 با کمک ابزار Vivado پیاده سازی شد. برد توسعه ZC706 [۲۴] شامل قطعه Zynq-7000 XC7Z045 از شرکت Xilinx می باشد. قسمت PL قطعه XC7Z045 شامل بلوک های منطقی قابل پیکربندی مجدد با ۲۱۸۶۰۰ جدول جستجو[۲۷] (LUT)، ۴۳۷۲۰۰ رجیستر[۲۸]، ۱۰۹۰ BRAM و ۹۰۰ بلوک DSP48E می باشد. قسمت PS این برد شامل دو هسته پردازشی ARM Coretex-.A9 است که با فرکانس 1.2G کار می کند. برای تست عملکرد الگوریتم از بردارهای تست تولید شده بصورت تصادفی در قسمت PS استفاده شده است. تقسیم بندی PS/PL انجام شده برای پیاده سازی این الگوریتم در شکل (۹) نشان داده شده است. تعداد کل منابع مورد نیاز برای پیاده سازی الگوریتم BLAKE2b پیشنهاد شده در جدول (۳) خلاصه شده است.

برای مقایسه، الگوریتم مورد نظر با استفاده از ابزار Vivado HLS نیز بصورت سطح بالا طراحی و شبیه سازی شد. IP ساخته شده با کمک ابزار HLS در محیط Vivado IDE مورد شبیه سازی قرار گرفت و نتایج خروجی مشابه آنچه در Vivado HLS بدست آمده بود ایجاد شد. برای استخراج پارامترهای HLS IP طراحی شده، IP های SIPOو PISO طراحی شده برای روش پیشنهادی در ورودی و خروجی های HLS IP قرار داده شدند تا تعداد پایه های ورودی و خروجی سازگار با تعداد I/O های قطعه XC7Z045 باشند. نتایج پیاده سازی طرح با HLS IP با کمک Vivado IDE با در نظر گرفتن برد توسعه ZC706 در جدول (۳) خلاصه شده است.



شکل (۹): تقسیم بندی نرم افزاری/ سخت افزاری الگوریتم روی تراشه XC7Z045 برد ZC706

با توجه به اینکه الگوریتم BLAKE2b شامل توابع G است که خروجی های واسط ایجاد شده در هر مرحله در تولید سایر خروجی های اصلی استفاده می شوند، بهینه سازی این توابع با استفاده از کد نویسی سطح بالا دشوار است. هر چند با اضافه کردن تعدادی دایرکتیو[۲۹] به طرح، تلاش بر ایجاد موازی سازی آرایه ها شد، با این وجود تعداد ۶۳ سیکل کلاک برای انجام پردازش الگوریتم BLAKE2b پیاده سازی شده با کد نویسی سطح بالا مورد نیاز است. با آنالیز طرح مشاهده می شود که کل این تعداد مرتبط با انجام محاسبات مربوط به توابع G است. ماکزیمم فرکانس کلاک IP ساخته شده با HLS نیز برابر 100M می باشد. تعداد کل منابع مورد نیاز برای پیاده سازی الگوریتم BLAKE2b با استفاده از کد نویسی سطح بالا نیز در جدول (۳) آورده شده است.

همچنین الگوریتم BLAKE2b بدون بهینه سازی توابع G فقط با در نظر گرفتن روش پایپلاین برای حصول فرکانس کلاک مطلوب نیز کد نویسی و با برد ZC706 با ابزار Vivado پیاده سازی شد. نتایج مربوط به تعداد منابع مصرفی این ساختار نیز در جدول (۳) گزارش شده است.

جدول (۳): تعداد منابع مصرفی مربوط به الگوریتم های مورد بررسی

Resources	*LUT*	*FF*	*DSP48E*	*BRAM_18k*
Proposed 64-bit BLAKE2b	6575	4726	0	0
Regular 64-bit BLAKE2b pipelined	10697	5233	0	0
HLS block	66835	49473	0	0
Available Resources	218600	437200	900	1090

همچنین در صد منابع مصرفی سه طرح مورد بررسی در جدول (۴) خلاصه شده است. با توجه به جدول (۴) پیاده سازی با استفاده از HLS IP حدود ۱۰ برابر طرح بهینه شده پیشنهاد شده منابع LUT و FF مصرف می کند.

جدول (۴): درصد منابع مصرفی مربوط به الگوریتم های مورد بررسی

Utilization%	*LUT*	*FF*
Proposed 64-bit balke2b	3	1
Regular 64-bit balke2b pipelined	5	1
HLS block	30	11

سایر مشخصات مربوط به طرح های مورد بررسی در جدول (۵) خلاصه شده است. نرخ انجام در هم سازی[۳۰] و همینطور توان مصرفی کل مربوط به سه الگوریتم نیز در جدول (۵) نشان داده شده است. تخمین توان مصرفی با استفاده از ابزار تخمین توان Vivado بدست آورده شده است. برای محاسبه نرخ انجام در هم سازی از رابطه (۲) استفاده شده است:

$$Hash\ rate = \frac{Clock\_frequency}{Number\ of\ Cycles} \quad (۲)$$

جدول (۵): مشخصات مربوط به سه ساختار بررسی شده در این مقاله

Specifications	Power Consumption(W)			Hash rate (MHash/s)	En.-Eff. (nJ/Hash)
	Dynamic	Static	Total		
Proposed 64-bit BLAKE2b	0.115	0.201	0.316	50	6.3
Regular 64-bit BLAKE2b pipelined	0.253	0.202	0.455	66	6.89
HLS IP block	0.252	0.202	0.454	1.6	283.75

۵- نتیجه‌گیری

از بین الگوریتم های در هم ساز برای کاربردهای ارزهای رمز نگاری شده الگوریتم BLAKE به عنوان یکی از الگوریتم های بهینه از لحاظ سرعت پیاده سازی در هر دو زمینه نرم افزاری و سخت افزاری در این مقاله مورد بررسی قرار گرفت. از بین ورژن های مختلف آن الگوریتم BLAKE2b که ۶۴ بیتی است در این مقاله بطور کامل توضیح داده شد. به منظور کاهش تعداد منابع مصرفی الگوریتم و بهبود توان مصرفی و سطح اشغال شده با هدف پیاده سازی های ASIC/FPGA بهینه سازی هایی در ساختار انجام شد. یکی از این بهینه سازی ها کاهش تعداد توابع G مورد استفاده در ساختار و کنترل دیتای ورودی و خروجی بلوک فشرده ساز الگوریتم در هم ساز با FSM است. الگوریتم پیشنهاد شده جدید با استفاده از Vivado با در نظر گرفتن برد ZC706 پیاده سازی شد. به منظور مقایسه، پیاده سازی دیگری با IP ساخته شده با ابزار HLS با کمک کد نویسی سطح بالا انجام گرفت. همینطور حالت عادی الگوریتم با تعداد ۸ تابع G بصورت موازی نیز کدنویسی و با ابزار Vivado پیاده سازی شد. نتایج پیاده سازی نشان می دهد که الگوریتم پیشنهاد شده نسبت به ساختارهای دیگر تعداد منابع کمتری استفاده کرده و توان مصرفی پایین تری دارد.

مراجع

[1] A. Kuznetsov, K. Shekhanin, A. Kolhatin, D. Kovalchuk, V. Babenko, and I. Perevozova, "Performance of hash algorithms on GPUs for use in blockchain," in *2019 IEEE International Conference on Advanced Trends in Information Theory (ATIT)*, 2019, pp. 166-170: IEEE. doi: 10.1109/ATIT49449.2019.9030442 .

[2] S. Shioiri, K. Yamamoto, H. Oshida, K. Matsubara, and H. Yaguchi, "Measuring attention using flash-lag effect," *J Vis*, vol. 10, no. 10, p. 10, Aug 13 2010. doi: 10.1167/10.10.10.

[3] J.-F. Têtu, L.-C. Trudeau, M. Van Beirendonck, A. Balatsoukas-Stimming, and P. Giard, "A standalone FPGA-Based miner for Lyra2REv2 cryptocurrencies," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 4, pp. 1194-1206, 2020. doi: 10.1109/TCSI.2020.2970923.

[4] D. Rachmawati, J. Tarigan, and A. Ginting, "A comparative study of Message Digest 5 (MD5) and SHA256 algorithm," in *Journal of Physics: Conference Series*, 2018, vol. 978, no. 1, p. 012116: IOP Publishing. doi: 10.1088/1742-6596/978/1/012116.

[5] S. Zhu, C. Zhu, and W. Wang, "A New Image Encryption Algorithm Based on Chaos and Secure Hash SHA-256," *Entropy (Basel)*, vol. 20, no. 9, p. 716, Sep 19 2018. doi: 10.3390/e20090716.

[6] S. Gupta, N. Goyal, and K. Aggarwal, "A review of comparative study of md5 and ssh security algorithm," *International Journal of Computer Applications*, vol. 104, no. 14, 2014. doi: 10.5120/18267-9305.

[7] G. Bertoni, J. Daemen, M. Peeters, G. V. Assche, R. V. Keer, and B. Viguier, "K angarooT welve: Fast Hashing Based on $${\textsc {Keccak}\text {-} p}{} $$," in *International Conference on Applied*

Cryptography and Network Security, 2018, pp. 400-418: Springer. doi: 10.1007/978-3-319-93387-0_21.

[8] F. Kahri, H. Mestiri, B. Bouallegue, and M. Machhout, "High speed FPGA implementation of cryptographic KECCAK hash function crypto-processor," *Journal of Circuits, Systems and Computers*, vol. 25, no. 04, p. 1650026, 2016. doi: 10.1142/S0218126616500262.

[9] J.-P. Aumasson, S. Neves, Z. Wilcox-O'Hearn, and C. Winnerlein, "BLAKE2: simpler, smaller, fast as MD5," in *International Conference on Applied Cryptography and Network Security*, 2013, pp. 119-135: Springer. doi: 10.1007/978-3-642-38980-1_8.

[10] J.-P. Aumasson, S. Neves, Z. Wilcox-O'Hearn, and C. Winnerlein, "BLAKE2 fast secure hashing," ed: Web Blog, Source: https://BLAKE2. net, 2021.

[11] V. Rao and K. Prema, "Light-weight hashing method for user authentication in Internet-of-Things," *Ad Hoc Networks*, vol. 89, pp. 97-106, 2019. doi: 10.1016/j.adhoc.2019.03.003.

[12] C. Zet and G.-C. Dumitriu, "Using blockchain technology for ensuring students results traceability for instrumentation classes," *Measurement: Sensors*, vol. 18, p. 100315, 2021. doi: 10.1016/j.measen.2021.100315.

[13] H. Cho, "ASIC-resistance of multi-hash proof-of-work mechanisms for blockchain consensus protocols," *IEEE Access*, vol. 6, pp. 66210-66222, 2018. doi: 10.1109/ACCESS.2018.2878895.

[14] Q. Aini, N. Lutfiani, N. P. L. Santoso, S. Sulistiawati, and E. Astriyani, "Blockchain for education purpose: essential topology," *Aptisi Transactions on Management (ATM)*, vol. 5, no. 2, pp. 112-120, 2021. doi: 10.33050/atm.v5i2.1506.

[15] J.-F. Têtu, L.-C. Trudeau, M. Van Beirendonck, A. Balatsoukas-Stimming, and P. Giard, "FPGA-based Mining of Lyra2REv2 Cryptocurrencies," *CoRR*, 2019. doi: 10.1109/TCSI.2020.2970923.

[16] R. J. Meijer, "MattockFS; Page-cache and access-control concerns in asynchronous message-based forensic frameworks on the Linux platform," *arXiv preprint arXiv:1703.00369*, 2017. doi: 10.13140/RG.2.2.35426.53440.

[17] M. Al-Zubaidie, Z. Zhang, and J. Zhang, "REISCH: incorporating lightweight and reliable algorithms into healthcare applications of WSNs," *Applied Sciences*, vol. 10, no. 6, p. 2007, 2020. doi: 10.3390/app10062007.

[18] J.-P. Aumasson, W. Meier, and R. C.-W. Phan, "The hash function family LAKE," in *International Workshop on Fast Software Encryption*, 2008, pp. 36-53: Springer. doi: 10.1007/978-3-540-71039-4_3.

[19] M. S. Mahdi, N. F. Hassan, and G. H. Abdul-Majeed, "An improved chacha algorithm for securing data on IoT devices," *SN Applied Sciences*, vol. 3, no. 4, pp. 1-9, 2021. doi: 10.1007/s42452-021-04425-7.

[20] A. Biryukov and D. Wagner, "Advanced slide attacks," in *International conference on the theory and applications of cryptographic techniques*, 2000, pp. 589-606: Springer. doi: 10.1007/3-540-45539-6_41.

[21] T. Peyrin, "Security analysis of extended sponge functions," in *Talk at the workshop Hash functions in cryptology: theory and practice*, 2008.

[22] J.-P. Aumasson, L. Henzen, W. Meier, and R. C.-W. Phan, "Sha-3 proposal BLAKE," *Submission to NIST*, vol. 92, 2008.

[23] S. Gueron, S. Johnson, and J. Walker, "SHA-512/256," in *2011 Eighth International Conference on Information Technology: New Generations*, 2011, pp. 354-358: IEEE. doi: 10.1109/ITNG.2011.69

[24] Xilinx, "ZC706 Evaluation Board for the Zynq-7000 XC7Z045 SoC (UG954), v1. 7," ed: Xilinx San José, CA, USA, 2018.

زیرنویس‌ها

[1] Hash
[2] Central Processing Unit
[3] Graphics Processing Unit
[4] Field Programmable Gate Array
[5] Application Specific Integrated Circuit
[6] Secure Hashing Algorithm
[7] Message Digest 5
[5] Finite State Machine
[9] Padding
[10] Big Endian
[11] Compressing
[12] Initialization
[13] Mixing Function
[14]Finalization
[15] Slide attacking
[16] Internal State
[17] Clock Frequency
[18] Chain Value
[19] System on Chip
[20] Intellectual Property
[21] Hardware Description Language
[22] High Level Synthesis
[23] Software Development Kit
[24] Evaluation Board
[25] Serial in/ Parallel out
[26] Parallel in/ Serial out
[27] Lookup Table
[28] Register
[29] Directive
[30] Hash rate