مروری بر پیاده‌سازی‌های سخت‌افزاری سبک‌وزن رمز AES

محسن جهانبانی*[1]، نصور باقری[2] و زین‌العابدین نوروزی[3]

[1] دانشجوی دکترای ریاضی-رمز، دانشکده و پژوهشکده مهندسی فن‌آوری اطلاعات و ارتباطات، دانشگاه جامع امام حسین (ع)، تهران، ایران
mjahanbani@ihu.ac.ir

[2] استادیار، دانشکده مهندسی برق، دانشگاه تربیت دبیر شهید رجائی، تهران، ایران
nbagheri@srttu.edu

[3] استادیار، دانشکده و پژوهشکده مهندسی فن‌آوری اطلاعات و ارتباطات، دانشگاه جامع امام حسین (ع)، تهران، ایران
znorozi@ihu.ac.ir

چکیده

وسایل با منابع محدود حافظه، توان و انرژی، مانند حس‌گرهای بی‌سیم شبکه و RFID–ها دارای نیازمندی‌های امنیتی هستند که پیاده‌سازی الگوریتم‌های رمزنگاری به‌صورت معمول روی این وسایل مناسب نبوده و منجر به مصرف زیاد منابع می‌شود. یک راه حل، طراحی الگوریتم‌های جدید رمز سبک‌وزن است که اغلب این الگوریتم‌ها نسبت به الگوریتم‌های استاندارد دارای سطح امنیتی پایین‌تری هستند. راه حل دوم پیاده‌سازی الگوریتم‌های استاندارد مانند رمز قالبی AES به‌صورت سبک‌وزن است که در این نوع پیاده‌سازی از روش‌هایی مانند به‌اشتراک‌گذاری منابع، پیاده‌سازی S-box با مدارات ترکیبی، انتقال محاسبات روی میدان متناهی از یک پایهٔ به پایه دیگر و محاسبات در پرواز استفاده می‌شود. با توجه به اینکه تاکنون پیاده‌سازی‌های فشرده AES از لحاظ میزان مصرف ناحیه، انرژی، توان و توان عملیاتی به‌صورت جامع بررسی نشده‌اند، در اینجا مهم‌ترین پیاده‌سازی‌های سخت‌افزاری سبک‌وزن ارائه‌شده برای رمز AES بررسی و ارزیابی شده‌اند. این معیارهای ارزیابی شامل مقدار گیت مصرفی، تعداد کلاک موردنیاز رمزنگاری/رمزگشایی، توان عملیات، مصرف توان، انرژی و ترکیب آن‌ها است. بررسی‌های ما نشان می‌دهد با تلاش‌های صورت‌گرفته از الگوریتم‌های استاندارد مانند AES رمز در کاربردهای با منابع محدود ناحیه در حد ۲۰۰۰ الی ۳۰۰۰ گیت و انرژی محدود در حد چند پیکو ژول به راحتی می‌توان استفاده کرد. برخی از این موفقیت‌ها نتیجه پیشرفت فناوری مدارات CMOS و برخی نیز نتیجه ارائه معماری مناسب سخت‌افزاری، خلاقیت در زمان‌بندی یک عملیات رمزنگاری و استفاده بهینه از منابع است.

واژگان کلیدی: پیاده‌سازی سبک‌وزن، AES، توان و انرژی، ناحیه مصرفی، کلاک

۱- مقدمه

امروزه کاربردهای رمزنگاری برای بسیاری از سامانه‌های نیازمند ارتباطات امن، احراز هویت و امضاهای رقمی ضروری شده است. در سال‌های اخیر وسایل هوشمند و نهفته در تمام جوانب زندگی ما نفوذ کرده‌اند. این وسایل اغلب برای کاربردهای حساس مانند کنترل دسترسی، عملیات بانکی و سلامتی استفاده می‌شوند. اتصال این وسایل به‌هم مفهوم اینترنت اشیا (IOT[1]) را ایجاد کرده است. نیازهای امنیتی این کاربردها منجر به ایجاد حوزهٔ پژوهشی در زمینه رمزنگاری سبک‌وزن شده است که هدف آن طراحی و پیاده‌سازی اولیه‌های امنیتی منطبق بر نیازهای وسایل با منابع بسیار محدود است که با دو رویکرد اصلی، به این هدف می‌توان دست یافت. رویکرد نخست طراحی الگوریتم‌های جدید جهت پیاده‌سازی در وسایل با منابع محدود و رویکرد دوم تلاش برای

[1] Internet of Things

پیاده‌سازی الگوریتم‌های استاندارد و شناخته‌شده به‌صورت سبک‌وزن است. رمزهای قالبی یکی از این اولیه‌های امنیتی است که پژوهش‌های گسترده‌ای برای طراحی و پیاده‌سازی آن با رویکردهای بالا صورت گرفته است. در این حالت در رویکرد اول نخست ساختن رمزهای قالبی دارای خاصیت سبک‌وزنی با بهینه‌سازی یک یا چندین پارامتر در حوزه طراحی است. از این طرح‌ها که در چند سال اخیر پیشنهاد شده است، مواردی مانند هایت[1] [۱]، کتان[2] [۲]، کلین[3] [۳]، لد[4] [۴]، نوکئون[5] [۵]، پرزنت[6] [۶]، پیکولو[7] [۷]، پرینس[8] [۸]، سایمون/اسپک[9] [۹] و توین[10] [۱۰] را می‌توان نام برد.

هدف در رویکرد دوم پیاده‌سازی رمزهای استاندارد مانند AES[11]، SHA-256[12] و SHA-3 به‌صورت سبک‌وزن است. با وجود اینکه در حوزه رمزهای قالبی تعدادی الگوریتم سبک‌وزن استاندارد وجود دارد، رمز AES هنوز یک انتخاب ارجح برای ایجاد امنیت در محیط‌هایی با محدودیت منابع است. منابع مورد نیاز در پیاده‌سازی‌های سخت‌افزاری به‌صورت ناحیه، توان و انرژی مصرفی است. منظور از ناحیه مصرفی مقدار سیلیکون اشغال‌شده برای پیاده‌سازی است. این ناحیه برای کاربردهایی که محدودیت فضا وجود دارد یا در وسایلی که حساس به هزینه هستند، دارای اهمیت است؛ به‌علاوه در وسایل هوشمند غیرفعال مانند برچسب‌های شناسایی با امواج رادیویی (RFID[13]) و کارت‌های هوشمند غیر تماسی محدودیت‌های توان نیز مهم است. بسته به برد انتقال برچسب‌ها محدودیت توان در حد چند میکرو وات است. بنابراین این محدودیت باید در هنگام پیاده‌سازی رعایت شود. وسایل هوشمند فعال مانند نودهای حسگر بی‌سیم، RFID-خوان‌ها و کارت‌های هوشمند تماسی که دارای منبع تغذیه مربوط به خود مانند باتری هستند از لحاظ محدودیت مصرف توان شرایط بهتری دارند. همچنین مصرف انرژی، متفاوت از مصرف توان است. یک پیاده‌سازی ممکن است دارای مصرف توان پایین باشد؛ اما به‌علت نیاز به زمان طولانی برای محاسبات، انرژی مصرفی آن بالا باشد. بنابراین معیار اصلی باید کاهش مصرف انرژی کل باشد.

چندین پیاده‌سازی سبک‌وزن در این حوزه انجام شده است. بعضی از نتایج مانند [۱۱] و [۱۲] با هدف پیاده‌سازی فشرده در بستر مدارات خاص منظوره (ASIC[14]) و آرایه‌های منطقی برنامه‌پذیر (FPGA[15]) به‌ترتیب انجام شده است. کارهای [۱۳] و [۱۴] با هدف رسیدن به مسیر بحرانی کوتاه‌تر و افزایش توان عملیاتی روی بستر ASIC صورت گرفته است. هدف کارهای [۱۵] و [۱۶] پیاده‌سازی مدار با مصرف انرژی پایین به‌ازای هر عملیات رمزنگاری است.

در بخش ۲ معیارها و روش‌های مورد استفاده در پیاده‌سازی فشرده الگوریتم‌های رمز ارائه خواهد شد. در بخش ۳ ساختار رمز AES شرح داده می‌شود. در بخش ۴ مروری بر مهم‌ترین کارهای صورت‌گرفته در راستای پیاده‌سازی فشرده الگوریتم AES انجام می‌گیرد. در بخش ۵ با کمک معیارهای ارائه‌شده بخش ۲، طرح‌های ارائه‌شده با یکدیگر مقایسه خواهند شد و در پایان جمع‌بندی و نتیجه‌گیری ارائه خواهد شد.

۲- پیاده‌سازی سخت‌افزاری فشرده الگوریتم‌های رمز

بستر پیاده‌سازی الگوریتم‌های رمزنگاری به دو دسته کلی نرم‌افزار و سخت‌افزار تقسیم‌بندی می‌شوند. انتخاب یک بستر به پارامترهای زیادی به‌خصوص حوزه کاربرد وابسته است. بسترهای نرم‌افزاری روی ریزپردازنده‌ها مانند کارت‌های هوشمند یا پردازنده‌های همه‌منظوره مانند رایانه‌ها می‌تواند باشد؛ اما این بسترها در مقایسه با بسترهای سخت‌افزاری بسیار کندتر است. بستر سخت‌افزاری به‌طور معمول شامل FPGA یا ASIC است. بستر ASIC مدارهای مجتمعی هستند که به‌منظور یک عملیات خاص طراحی و بهینه‌سازی شده‌اند. مانند پردازنده یک گوشی هوشمند؛ اما FPGA-ها بسترهایی هستند که قابلیت پیکربندی توسط طراح یا مشتری را دارند که با کمک یکی از زبان‌های توصیف‌گر سخت‌افزار (HDL[16]) مانند VHDL اجزای سازنده FPGA-ها به‌گونه‌ای تغییر می‌کنند که عملکرد موردنظر را پیاده‌سازی کنند. همچنین

[1] HIGHT
[2] KATAN
[3] Klein
[4] LED
[5] Noekeon
[6] Present
[7] Piccolo
[8] Prince
[9] Simon/Speck
[10] TWINE
[11] Advance Encryption Standard
[12] Secure Hash Algorithm
[13] Radio Frequency Identification
[14] Application Specific Integrated Circuit
[15] field-programmable gate array
[16] Hardware Description Language

FPGA–ها هزینه پایین‌تری در تولید یک محصول داشته و قابلیت بازپیکربندی بعد از تولید آن را نیز دارند؛ اما در کاربردهای تولید انبوه ASIC–ها دارای عملکرد سریع‌تر و دارای توان مصرفی کمتری هستند.

برای یک الگوریتم رمزنگاری، ابتدا یک معماری سخت‌افزاری ارائه می‌شود. این معماری برای اهداف پیاده‌سازی گذردهی بالا یا پیاده‌سازی فشرده متفاوت خواهد بود که برای هر کدام روش‌های مشخصی وجود دارد. این معماری به کدهای VHDL یا وریلاگ[1] تبدیل می‌شود؛ سپس صحت کارایی کد با کمک بردارهای آزمایش تولیدشده از پیاده‌سازی نرم‌افزاری مرجع، بررسی می‌شود. کدها به کمک ابزارهای کدنویسی و سنتز FPGA یا ASIC مانند ISE، ویوادو[2] و سیناپسیس[3] سنتز می‌شوند که این ابزارهای سنتز تا حدودی بهینه‌سازی‌ها را به‌صورت خودکار انجام می‌دهند. با توجه به نتایج به دست آمده مانند مسیر بحرانی و فرکانس کاری، هدف نهایی از پیاده‌سازی (گذردهی بالا یا پیاده‌سازی فشرده)، به‌صورت دستی بهینه‌سازی‌هایی انجام می‌شود. همچنین از خروجی به‌دست آمده و بردار آزمایش، بررسی زمانی صورت گرفته تا صحت پیاده‌سازی تأیید شود. درنهایت پس از نگاشت، جانمایی و مسیریابی[4] مدار روی سخت‌افزار هدف، مصرف منابع، فرکانس کاری و غیره گزارش می‌شود. شکل (۱) جریان توسعه و ارزیابی سخت‌افزاری یک طرح رمزنگاری را نشان می‌دهد[۱۷].

درصورتی‌که کاربرد طرح رمز در محیط‌های با محدودیت منابع باشد مرحله بهینه‌سازی در جریان پیاده‌سازی بسیار مهم خواهد بود. روش‌هایی وجود دارد که به کمک آن طرح را به‌صورت بهینه می‌توان پیاده‌سازی کرد. این نوع پیاده‌سازی‌ها نیز پیاده‌سازی فشرده نامیده می‌شوند. برای مقایسه و ارزیابی کارایی هر پیاده‌سازی معیارهایی وجود دارد که خروجی‌ها با کمک آن با هم مقایسه می‌شوند. در ادامه این دو موضوع شرح داده خواهد شد.

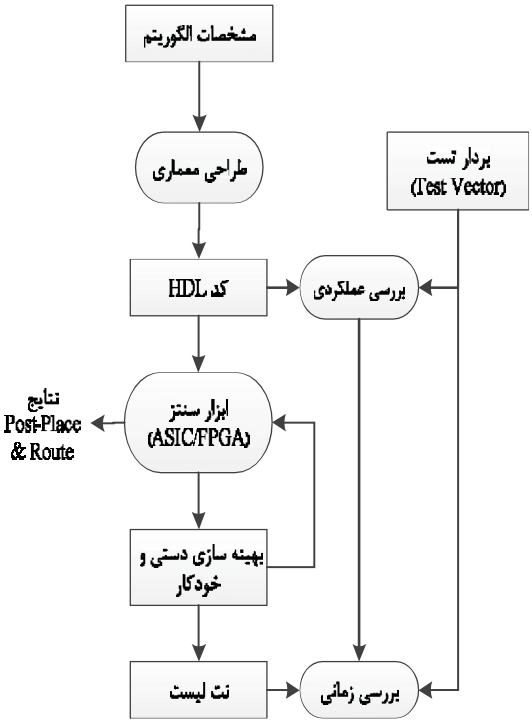
۱-۲- معیارهای سنجش پیاده‌سازی سخت‌افزاری الگوریتم‌های رمزنگاری

توان عملیاتی[5] و تأخیر:

توان عملیاتی یا گذردهی برابر تعداد بیت‌های رمزنگاری/رمزگشایی‌شده در واحد زمان تعریف می‌شود. به‌طور معمول توان عملیاتی رمزنگاری و رمزگشایی یکسان است و یکی از آن‌ها گزارش می‌شود. واحد سنجش آن نیز به‌طور معمول Mbit/s یا Gbit/s است. همچنین تأخیر زمان لازم برای رمزنگاری/ رمزگشایی در قالب متن اصلی/ متن رمزی تعریف می‌شود که واحد سنجش آن به‌طور معمول ns است. ارتباط توان عملیاتی با تأخیر به‌صورت زیر است:

$$\text{گذردهی} = \frac{\text{اندازه بلوک} \times \text{تعداد بلوک‌های پردازش شده هم زمان}}{\text{تأخیر}} \quad (۱)$$

در کاربردهایی که مقدار زیادی داده رمزنگاری/رمزگشایی می‌شود زمان کل رمزنگاری/ رمزگشایی این داده‌ها توسط توان عملیاتی تعیین می‌شود؛ درنتیجه این معیار خوبی برای سرعت یک رمز است. در حالتی که تعداد کمی متن اصلی/ متن رمزی پردازش می‌شود، زمان کل رمزنگاری این داده‌ها به هر دو مقدار تأخیر و توان عملیاتی وابسته است.



(شکل-۱): نمودار جریان توسعه و ارزیابی سخت‌افزاری یک طرح رمزنگاری [۱۷]

[1] Verilog
[2] Vivado
[3] Synopsis
[4] Place and Route
[5] Throughput

ناحیه سخت‌افزاری:

ناحیه سخت‌افزاری موردنیاز برای پیاده‌سازی یک رمز، به دلایل زیر شاخصهٔ مهمی است:

هزینه: مقدار ناحیه مصرفی یک مدار مجتمع عامل اصلی تعیین هزینه آن مدار است.

محدودیت روی بیشترین ناحیه: در محیط‌های سخت‌افزاری خاص برای یک واحد رمزنگاری محدودیت ناحیه وجود دارد. این محدودیت ممکن است به‌دلیل هزینه، فناوری‌های موجود ساخت، مصرف توان و یا ترکیب این عوامل تحمیل شود. برای مثال در کارت‌های هوشمند و ریزپردازنده‌ها هزینه و توان مصرفی ناحیه مصرفی واحد رمزنگاری نهفته را محدود می‌کند. در FPGA-ها فناوری ساخت و هزینه این محدودیت را تحمیل می‌کند. در پیاده‌سازی‌های ASIC ناحیه مصرفی به‌طور معمول بر حسب μm^2 بیان می‌شود. البته گاهی مساحت مصرفی بعد از سنتز منطقی مدار توسط ابزارهای نرم‌افزاری بر حسب گیت‌های منطقی موجود در کتابخانه استاندارد مانند گیت‌های XOR (جمع)، AND (ضرب)، OR (یا) و غیره بیان می‌شود. در کارهای پژوهشی و دانشگاهی برای مقایسه منصفانه طرح‌ها، فرض می‌شود که کل مدار با گیت NAND دو ورودی پیاده‌سازی شده است و آن‌گاه تعداد گیت مصرفی گزارش می‌شود که این مقدار معادل گیت یا GE[1] نامیده می‌شود. بدین‌منظور مساحت مصرفی کل طرح به مساحت مصرفی یک گیت NAND تقسیم می‌شود تا مقدار GE به‌دست آید.

در پیاده‌سازی FPGA ناحیه مصرفی توسط ابزار هر برند FPGA بر حسب منابع آن FPGA گزارش می‌شود. به‌عنوان مثال خانواده Xilinx دارای ابزار ISE یا ویوادو هستند. پس از سنتز مدار منابع مصرفی مانند اسلایس[2]، LUT[3]، حافظه رم بلوکی، فلیپ-فلاپ و DSP گزارش می‌شود.

اندازه‌گیری و مقایسه ناحیه مصرفی در FPGA-هایی که در آن‌ها از بلوک‌های آماده مانند RAM و DSP-ها استفاده شود، سخت است؛ زیرا هیچ معیاری برای تبدیل این بلوک‌ها به منابع اصلی FPGA مانند LUT وجود ندارد. بنابراین برای مقایسه منصفانه صرفاً از منابع اصلی سازنده FPGA-ها استفاده می‌شود.

انرژی و توان:

انرژی و توان سنجه‌های ضروری برای رمزهایی است که کاربرد آن‌ها استفاده در تجهیزات با منابع کم و محدودیت انرژی است. میانگین اتلاف توان، نرخ مصرف انرژی را نشان می‌دهد. مصرف توان وابسته به فرکانس کاری است و با کاهش آن مصرف توان هم کاهش می‌یابد. مصرف شامل دو بخش توان ایستا و پویا است. توان ایستا متناسب با ناحیه و فرآیند ساخت است و توان پویا متناسب با فعالیت سوییچینگ [4] مدار است. هر دو بخش توان به ولتاژ تغذیه نیز وابسته هستند. بازده انرژی که به‌صورت انرژی بر بیت بیان می‌شود، مهم‌ترین معیار در این بخش است. البته اگر اندازه قالب رمزها یکسان باشند، برای مقایسه فقط مصرف انرژی بر حسب ژول را جهت محاسبه یک قالب متن رمزی می‌توان در نظر گرفت.

توان عملیاتی به ناحیه:

از تقسیم توان عملیاتی طرح به ناحیه مصرفی به دست می‌آید.

بیشترین فرکانس:

مدت زمانی که طول می‌کشد یک ورودی به خروجی برسد مسیر بحرانی مدار می‌گویند. طولانی‌ترین مسیر بحرانی یا به اصطلاح کندترین مسیر، فرکانس کاری مدار را تعیین می‌کند. البته روش‌هایی مثل خط-لوله[5] وجود دارند که فرکانس کاری را افزایش داده؛ اما درعوض ناحیه و توان مصرفی را نیز افزایش می‌دهد.

۲-۲- روش‌های پیاده‌سازی فشرده سخت‌افزاری الگوریتم‌های رمز

برای پیاده‌سازی الگوریتم‌های رمز به‌خصوص در مواردی که با محدودیت منابع روبرو هستیم، روش‌هایی وجود دارد که منجر به کاهش حجم مصرفی، توان و انرژی می‌شود. به پیاده‌سازی‌هایی که با این روش‌ها انجام شوند، پیاده‌سازی‌های فشرده گویند. در ادامه برخی از این روش‌ها معرفی می‌شوند.

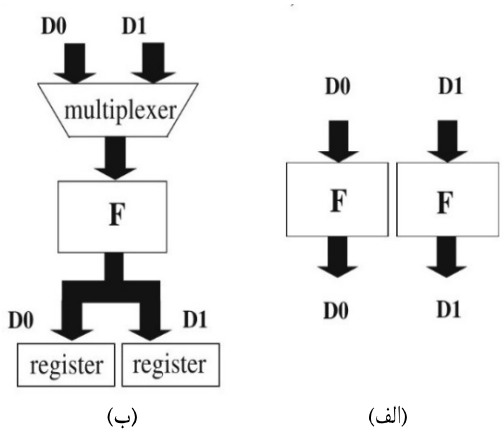
به اشتراک‌گذاری منابع[6]:

به اشتراک‌گذاری منابع شامل چند مفهوم است: در مفهوم نخست یک منبع در بخش‌های دیگر یک طرح یا حتی در ماژول‌های مختلف که در حال کار است، به اشتراک گذاشته می‌شود. مثال واضح این نوع به اشتراک‌گذاری که در بیشتر طرح‌ها وجود دارد، شمارنده‌ها هستند. در این حالت خروجی یک شمارنده در حال کار، در بخش‌های مختلفی استفاده می‌تواند شود و بدین ترتیب از پیاده‌سازی چندین شمارنده جلوگیری می‌شود.

[1] Gate Equivalent
[2] Slice
[3] Look-Up Table
[4] Switching
[5] Pipelining
[6] Resource Shairing

در مفهوم دوم بحث استفاده مجدد از یک ماژول پیاده‌سازی‌شده، مطرح است که در زمان‌های متفاوت ورودی‌های متفاوت می‌گیرد و نیازمند بخش کنترلی است.

در این روش از یک واحد عملکردی برای پردازش دو یا چند بخش قالب داده در کلاک‌های متفاوت استفاده می‌شود. در شکل (۲-الف) در حالت بدون به‌اشتراک‌گذاری منابع، دو بخش داده D_0 و D_1 با کمک دو واحد عملکردی F مستقل، به‌صورت موازی پردازش می‌شوند. در شکل (۲-ب) یک واحد F برای پردازش دو بخش از قالب داده به‌صورت سریال و در دو کلاک پشت سرهم استفاده شده است. در عمل از این روش برای کاهش عرض مسیر داده AES از حالت پایه ۱۲۸ به ۶۴ یا ۳۲ یا حتی ۸ بیت می‌توان استفاده کرد.



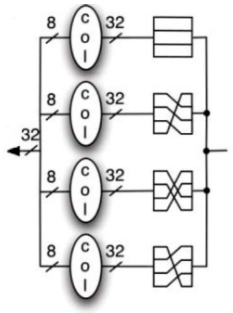
(شکل-۲): اشتراک‌گذاری منابع: الف) بدون به اشتراک‌گذاری تابع F، ب) با اشتراک‌گذاری تابع F [۱۷]

در مفهوم سوم، هنگامی‌که دو یا چند تابع متفاوت وجود دارد به جای اینکه توابع متفاوت جداگانه پیاده‌سازی شود از مفهوم به‌اشتراک‌گذاری منابع بین دو یا چند تابع برای کاهش ناحیه مصرفی استفاده می‌شود. برای این کار توابع تجزیه، بخش‌های مشترک حذف شده و یک تابع با چندین عملکرد پیاده‌سازی می‌شود که در آن عملکردهای متفاوت با یک انتخاب‌گر قابل استفاده است. برای مثال می‌توان به تابع مخلوط‌سازی ستونی در الگوریتم AES اشاره کرد. مخلوط‌سازی ستونی یک از ماژول‌های مصرف‌کننده زیاد منابع است و کارایی آن در کارایی نهایی مدار بسیار تأثیرگذار است. زمانی‌که به پیاده‌سازی هم‌زمان الگوریتم AES و معکوس در یک مدار نیاز است، به جای پیاده‌سازی دو مدار مجزا برای مخلوط‌سازی ستونی و معکوس آن، یک مدار ترکیبی را می‌توان طراحی کرد.

مخلوط‌سازی ستونی یک تبدیل ماتریسی است که از ضرب آن روی ستون‌های حالت تابع دور AES به‌دست می‌آید. برای این کار هر درایه ماتریس به‌عنوان یک عضو میدان $GF(2^8)$ در نظر گرفته می‌شود. این ماتریس به‌صورت زیر است:

$$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \quad (۲)$$

همان‌طور که مشاهده می‌شود، این ماتریس چرخشی است و یک مدار ضرب می‌توان طراحی کرد و سپس ورودی‌ها هر بار یک شیفت چرخشی داشته باشند. (شکل۶) این روند را نشان می‌دهد.



(شکل-۳): معماری مخلوط‌سازی ستونی [۱۸]

معکوس مخلوط‌سازی ستونی دارای ماتریس زیر است:

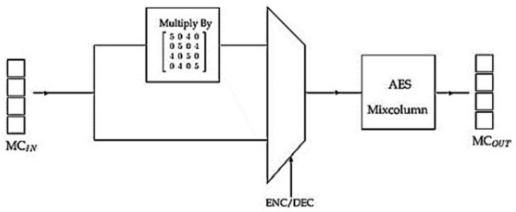
$$\begin{bmatrix} 14 & 11 & 13 & 9 \\ 9 & 14 & 11 & 13 \\ 13 & 9 & 14 & 11 \\ 11 & 13 & 9 & 14 \end{bmatrix} \quad (۳)$$

با توجه به پیچیده‌ترین ضرایب این ماتریس برای پیاده‌سازی آن ضرب‌کننده‌های بیشتری لازم است. یکی از راه‌های کاهش این پیچیدگی، تجزیه این ماتریس است به‌طوری‌که یک بخش از همان مدار مربوط به محاسبهٔ مستقیم مخلوط‌سازی ستونی باشد. یکی از بهترین تجزیه‌های ماتریس معکوس به‌صورت زیر است [۱۹]:

$$\begin{bmatrix} 14 & 11 & 13 & 9 \\ 9 & 14 & 11 & 13 \\ 13 & 9 & 14 & 11 \\ 11 & 13 & 9 & 14 \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \cdot \begin{bmatrix} 5 & 0 & 0 & 4 \\ 0 & 5 & 0 & 4 \\ 4 & 0 & 5 & 0 \\ 0 & 4 & 5 & 0 \end{bmatrix} \quad (۴)$$

معماری ارائه‌شده در شکل (۴) نشان می‌دهد عملکرد مخلوط‌سازی ستونی یا معکوس آن با سیگنال ENC/DEC

قابل انتخاب است، در حالت معکوس‌سازی یک ضرب اضافی در سطر (۵،۰،۴،۰) نیاز است. مابقی ضرب‌ها با چرخش ورودی روی این سطر انجام می‌شود.



(شکل-۴): معماری مخلوط‌سازی ستونی/معکوس مخلوط‌سازی ستونی [۱۹]

پیاده‌سازی S-box به‌صورت منطقی:

عملیات جانشینی بایتی[1] (S-box) یکی از مؤلفه‌های پرکاربرد در بیش‌تر رمزها برای ایجاد عملکرد غیرخطی کردن مدار به‌کار می‌رود. برای مثال هر S-box رمز AES دارای هشت بیت ورودی و هشت بیت خروجی است. یک راه حل ساده پیاده‌سازی S-box استفاده از یک جدول جستجوی ۲۵۶×۸ بیتی است؛ درنتیجه برای ذخیره یک S-box حافظه ۲۰۴۸ بیتی، برای معکوس آن همین مقدار حافظه موردنیاز است که برای طرح‌های فشرده اصلاً مناسب نیست. تلاش‌های زیادی برای پیاده‌سازی فشرده AES S-box در سال‌های اخیر انجام شده است. تمام آن‌ها از ساختار جبری S-box استفاده می‌کنند که ترکیبی از یک تبدیل آفینی[2] و محاسبه معکوس روی میدان متناهی است. برای این کار ورودی هر S-box به‌عنوان یک عضو روی میدان $GF(2^8)$ در نظر گرفته می‌شود که باید معکوس آن محاسبه شود. محاسبه معکوس عضو میدان دارای محاسبات پیچیده و زیادی است. ایده اصلی کاهش این پیچیدگی، استفاده از میدان مرکب[3] است که محاسبات از میدان اصلی $GF(2^8)$ به زیرمیدان‌های $GF(2^4)$ و $GF(2^2)$ و $GF(2)$ منتقل می‌شود. یعنی عملیات محاسبه معکوس روی $GF(2^8)$ به چندین عملیات روی زیرمیدان $GF(2^4)$ شامل جمع، ضرب و معکوس تبدیل می‌شود. به‌طور مشابه عملیات میدان $GF(2^4)$ با عملیات روی میدان $GF(2^2)$ جایگزین و درنهایت عملیات روی میدان $GF(2^2)$ با عملیات روی میدان $GF(2)$ تبدیل می‌شود. عملیات روی میدان $GF(2)$ با گیت‌های XOR و AND می‌تواند پیاده‌سازی شود.

معکوس عدد یک روی میدان $GF(2)$ برابر یک و معکوس صفر وجود ندارد. بنابراین محاسبه معکوس روی $GF(2^8)$ به یک مدار منطقی تشکیل‌شده تنها از گیت‌های XOR و AND می‌تواند تجزیه شود. پیچیدگی مداری تأخیر (مسیر بحرانی) این مدار به انتخاب نمایش هر میدان $GF(2^{2k})$ با کمک اعضای میدان پایه $GF(2^k)$ برای k= ۱، ۲، ۴ وابسته است. پیاده‌سازی کن‌رایت[4] [۲۰] یکی از این پیاده‌سازی‌ها با کمترین اندازه مداری برای محاسبه S-box، معکوس و ادغام هر دو است. در این پیاده‌سازی برای نمایش اعضای میدان $GF(2^{2k})$ از پایه نرمال استفاده شده و پس از بررسی تمام ۴۳۲ نمایش متفاوت هم‌ریخت و نمایش بهینه به‌صورت زیر ارائه شده است:

- میدان $GF(2^2)$ با الحاق یک ریشه W به چندجمله‌ای $w^2 + w + 1$ روی میدان $GF(2)$ با پایه نرمال (W^2, W) ساخته می‌شود.
- میدان $GF(2^4)$ با الحاق یک ریشه Z به چندجمله‌ای $z^2 + z + N$ روی میدان $GF(2^2)$ با پایه نرمال (Z^4, Z) ساخته می‌شود.
- میدان $GF(2^8)$ با الحاق یک ریشه Y به چندجمله‌ای $y^2 + y + \nu$ روی میدان $GF(2^4)$ با پایه نرمال (Y^{16}, Y) ساخته می‌شود.

اگر فرض شود عضو میدان $GF(2^8)$ به‌صورت $\gamma_1 Y^{16} + \gamma_0 Y$ باشد، آنگاه معکوس آن به‌صورت زیر است که به‌سادگی قابل بررسی است:

$$(\gamma_1 Y^{16} + \gamma_0 Y)^{-1} = [\theta^{-1}\gamma_0]Y^{16} + [\theta^{-1}\gamma_1]Y$$
$$\theta = \gamma_0\gamma_1 + (\gamma_1^2 + \gamma_0^2)\nu \qquad (۵)$$

بنابراین معکوس روی میدان $GF(2^8)$ شامل یک معکوس، سه ضرب و دو مربع‌کردن روی میدان $GF(2^4)$ است.

محاسبه معکوس روی میدان $GF(2^4)$ مشابه ساختار معکوس روی میدان $GF(2^8)$ است. یعنی اگر عضو میدان به‌صورت $\Gamma_1 Z^4 + \Gamma_0 Z$ و معکوس آن را به‌صورت $\Delta_1 Z^4 + \Delta_0 Z$ در نظر بگیریم، آن‌گاه مقادیر Δ_0 و Δ_1 برابر:

$$\Delta_1 = [\Gamma_1\Gamma_0 + (\Gamma_1^2 + \Gamma_0^2)N]^{-1}\Gamma_0$$
$$\Delta_0 = [\Gamma_1\Gamma_0 + (\Gamma_1^2 + \Gamma_0^2)N]^{-1}\Gamma_1 \qquad (۶)$$

درنتیجه معکوس روی میدان $GF(2^4)$ شامل یک معکوس، چند ضرب و دو مربع‌کردن روی میدان $GF(2^2)$ است.

معکوس روی میدان $GF(2^2)$ بسیار ساده و معادل جابه‌جای بیتی است. یعنی:

[1] SubBytes
[2] Affine
[3] Composite Field
[4] Canright

$$(g_1W^2+g_0W)^{-1}=(g_1W^2+g_0W)^2 \\ =g_0W^2+g_1W \qquad (۷)$$

ضرب روی میدان $GF(2^4)$ به‌صورت زیر محاسبه می‌شود:

$$(\Gamma_1Z^4+\Gamma_0Z)(\Delta_1Z^4+\Delta_0Z) \\ =\Phi_1Z^4+\Phi_0Z \\ \Phi_1=\Gamma_1\Delta_1+(\Gamma_1+\Gamma_0)(\Delta_1+\Delta_0)N \\ \Phi_0=\Gamma_0\Delta_0+(\Gamma_1+\Gamma_0)(\Delta_1+\Delta_0)N \qquad (۸)$$

همان‌طور که مشاهده می‌شود، ضرب روی میدان $GF(2^4)$ شامل تعدادی ضرب و تعدادی جمع روی میدان $GF(2^2)$ است.

ضرب روی میدان $GF(2^2)$ دارای ساختار مشابه با ضرب روی میدان $GF(2^4)$ است، به جز اینکه N=1 است، یعنی:

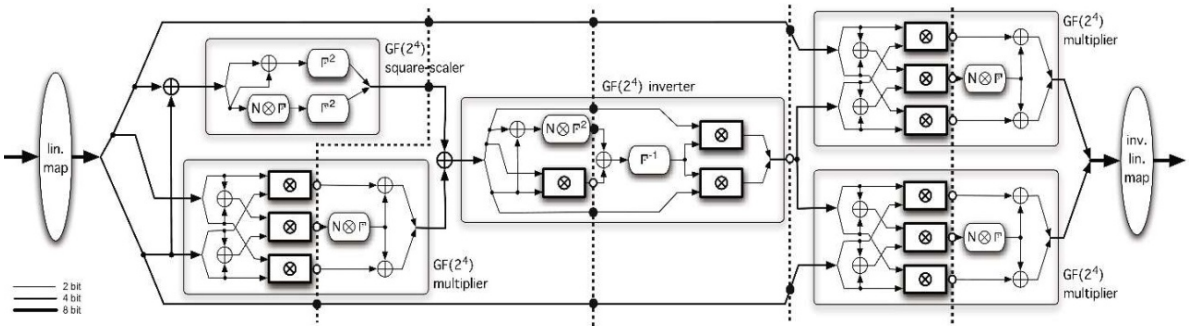
$$(g_1W^2+g_0W)((f_1W^2+f_0W) \\ =h_1W^2+h_0W \\ h_1=g_1f_1+(g_1+g_0)(f_1+f_0) \\ h_0=g_0f_0+(g_1+g_0)(f_1+f_0) \qquad (۹)$$

مربع‌کردن روی میدان $GF(2^4)$ همراه با ضرب ν با انتخاب $\nu=N^2Z$ به‌صورت زیر محاسبه می‌شود:

$$\nu(\Gamma_1Z^4+\Gamma_0Z)^2=[(\Gamma_1+\Gamma_0)^2]Z^4 \\ +[(N\Gamma_0)^2]\,Z \qquad (۱۰)$$

همچنین مربع‌کردن روی میدان $GF(2^2)$ مانند معکوس روی آن است و معادل جابه‌جایی بیتی است که درقبل توضیح داده شده است. در نهایت یک تبدیل خطی بهینه برای تغییر پایه از چندجمله‌ای به نرمال در ابتدا و بالعکس آن در انتها نیاز است. این تبدیل خطی با تبدیل آفینی S-box می‌تواند ترکیب شود. شکل (۵) نمایش S-box بر اساس معماری فشرده کن‌رایت را نشان می‌دهد. در این شکل مربع‌های ضخیم ضرب‌کننده روی میدان $GF(2^2)$ را نشان می‌دهد که تنها بخش غیرخطی مدار است. درنهایت برای پیاده‌سازی هر S-box به ۱۸۰ گیت و برای ترکیب با معکوس S-box به ۲۳۴ گیت نیاز است.

انتقال محاسبات روی میدان متناهی از یک پایه به پایه دیگر:
همان‌طور که در قسمت قبلی بیان شد، محاسبات بخش‌هایی از الگوریتم رمز مانند S-box روی میدان‌های متناهی صورت پذیرد. می‌توان اعضای میدان با استفاده از پایه چندجمله‌ای یا نرمال نمایش داد. ممکن است برخی عملیات در یک پایه کارآمدتر از پایه دیگر باشد. به‌طور مثال عملیاتی مانند مربع‌کردن روی پایه نرمال به‌طورتقریبی بدون هزینه و معادل جابه‌جایی بیتی است که منجر به مصرف کم‌تر منابع می‌شود.



(شکل-۵): معماری فشرده S-box بر اساس طرح کن‌رایت [۱۸]

استفاده از محاسبات در پرواز[1] در مقابل پیش محاسبه:
بخش فرانمای کلید[2] در الگوریتم‌های رمز کلید دور را تولید می‌کند. این تولید کلید می‌تواند به‌صورت در پرواز یا به‌صورت پیش محاسبه باشد. در حالت پیش محاسبه، کلیدهای دور در گام برپایی کلید درون حافظه کلید ذخیره می‌شود و هر زمان که لازم باشد فراخوانی می‌شود. در حالت در پرواز به جای اینکه کلیدهای دور از قبل تولید و ذخیره شوند، در حین عملیات هر دور، کلید همان دور نیز تولید می‌شود. این روش باعث عدم نیاز به استفاده از حافظه برای ذخیره‌سازی کلیدهای دور می‌شود.

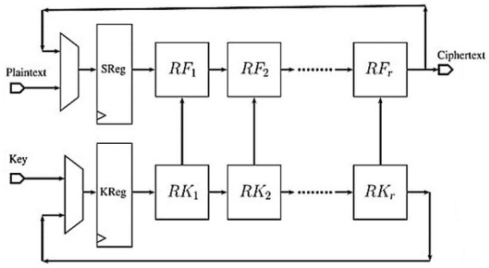
[1] On-the-fly
[2] Key Schedule

تکنیک کلیدزنی[1]:

این تکنیک به‌منظور کاهش مصرف توان و انرژی یک مدار به کار می‌رود. دو روش کلیدزنی که در پیاده‌سازی الگوریتم‌های رمز قابل اعمال است، کلیدزنی تابع دور و کلیدزنی کلاک است.

تکنیک کلیدزنی تابع دور [۲۱] در الگوریتم‌های رمز که به‌صورت حلقه‌باز[2] پیاده‌سازی شده باشند، مؤثر است. اگر یک نمونه از تابع دور و یک نمونه از فرانمای کلید پیاده‌سازی شده باشد طرح دارای معماری روند پایه است که اجرای کامل یک عملیات رمزنگاری نیاز به اجرای R مرتبه (R کلاک) تابع دور است. یک طراح بیشتر از یک نمونه تابع دور را می‌تواند پیاده‌سازی کند که به آن معماری حلقه‌باز گویند. اگر r نمونه پیاده‌سازی شود، آنگاه اجرای یک بار رمزنگاری در R/r کلاک انجام می‌پذیرد. و در معماری حلقه‌باز کامل R=r است؛ یعنی متن رمزی در یک کلاک تولید می‌شود.

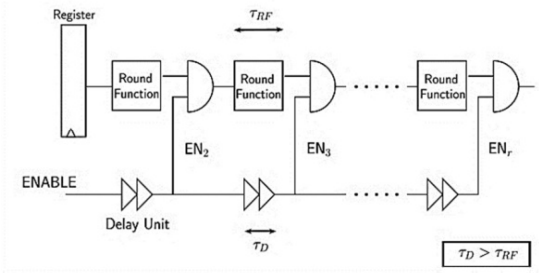
(شکل۶) یک معماری حلقه‌باز با r حلقه را نشان می‌دهد.



(شکل-۶): معماری رمز قالبی حلقه‌باز r- دوری [۲۱]

بر اساس مدل گیت‌های CMOS هنگامی‌که r نمونه از تابع دور به‌صورت سریال پشت سرهم باشند، خطاهای لحظه‌ای[3] بین تابع دورها به‌صورت سریال منتشر می‌شود و به‌خاطر این خطاها مصرف توان بالاتر می‌رود، یعنی تابع دور بعدی انرژی بیشتری از تابع دور قبلی مصرف می‌کند. به‌طور تقریبی مصرف توان تابع مربعی از r است؛ با این وجود انتشار خطاها از i-اُمین به (i+1)-اُمین تابع دور می‌تواند متوقف شود. این کار با خاموش‌کردن تابع (i+1)-اُم در هنگامی‌که تابع iاُم در حال محاسبه است، محقق می‌شود.

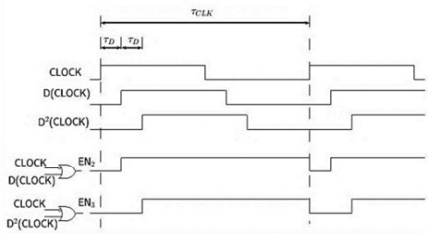
همان‌طورکه در شکل (۷) نشان داده شده است به جای اتصال مستقیم خروجی i-اُمین تابع دور به ورودی تابع (i+1)اُمین دور، از بانک گیت‌های AND دو ورودی جهت اتصال استفاده می‌شود و ورودی دیگر این گیت با سیگنال فعال‌ساز[4] کنترل می‌شود. مجموعه سیگنال‌های فعال‌ساز EN_i که کنترل ورودی هر تابع دور را بر عهده دارند، توسط زنجیره‌ای از واحدهای تأخیر تولید می‌شوند. سیگنال تأخیر τ_D باید بزرگ‌تر از تأخیر ایجاد شده τ_{RF} ناشی از هر تابع دور باشند. برای مثال دومین تابع دور برای مدت τ_D ثانیه بعد از لبه بالارونده کلاک خاموش است و چون $\tau_D > \tau_{RF}$ است، زمان کافی برای تولید خروجی صحیح و پایدار نخستین تابع دور قبل از ارسال به ورودی دومین تابع دور وجود دارد. بنابراین از انتشار خطای لحظه‌ای از دور ۱ به ۲ جلوگیری می‌شود. زنجیره تأخیر برای رسیدن به چنین عملکردی برای هر دو تابع دور متوالی، مدیریت می‌شود.



(شکل-۷): مدار کلیدزنی کلاک [۲۱]

شکل (۸) نشان می‌دهد که چگونه i-اُمین سیگنال فعال‌ساز EN_i از سامانهٔ کلاک می‌تواند تولید شود. سیگنال کلاک در میان i واحد تأخیر عبور می‌کند و $D^i(CLOCK)$ را تولید می‌کند. معادله منطقی سیگنال EN_{i+1} به‌صورت زیر است:

$$EN_{i+1} = \overline{CLOCK} \ or \ D_i(CLOCK) \qquad (۱۱)$$



(شکل-۸): شکل موج تولیدی کلیدزنی کلاک [۲۱]

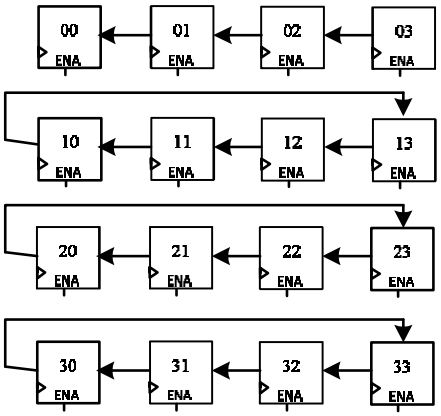
نتایج عملی [۲۱] نشان داد در حالتی که r کوچک باشد، این روش ممکن است، کارایی کمی داشته باشد؛ اما در حالتی که r≥ 4 باشد، در الگوریتم‌هایی مثل AES کاهش مصرف انرژی (توان پویای مدار) کاملاً مشهود است درحالتی‌که مدار حلقه‌باز کامل (R=r) پیاده‌سازی شده باشد،

[1] Gating
[2] Unrolled
[3] Glitch
[4] Enable

تا هشتاد درصد کاهش مصرف انرژی رخ دهد. البته ممکن است حدود ۱۰ تا ۱۵ درصد افزونگی سخت‌افزاری به خاطر مدارات تولید سیگنال‌های فعال‌ساز وجود داشته باشد.

یک روش دیگر کاهش مصرف توان، روش کلیدزنی کلاک است. در این روش انتشار کلاک در درخت کلاک محدود و باعث کاهش مصرف توان پویا می‌شود. در این روش در زمان‌های خاصی بخش‌هایی از مدار به‌خصوص به ثبات‌های کلاک نمی‌رسد و درنتیجه در آن بخش فعالیت سوییچینگ وجود نداشته و توان مصرف نمی‌شود.

برای مثال می‌توان به محدودسازی رسیدن کلاک به ثبات‌های انتقال و به اصطلاح منجمدکردن ثبات در عملیات جابه‌جایی سطری که در تابع دور الگوریتم AES کاربرد دارد، اشاره کرد. در عملیات جابه‌جایی سطری داده‌های i-اُمین سطر به اندازه i بایت که $0 \le i \le 3$ به سمت چپ جابه‌جا می‌شوند و در معکوس جابه‌جایی سطری داده‌های i-اُمین i بایت به سمت راست جابه‌جا می‌شوند. لازم به ذکر است، داده‌ها به‌صورت بایتی در نظر گرفته می‌شوند. در برخی پیاده‌سازی‌ها به‌خصوص با بستر ASIC این حالت به‌صورت شانزده ثبات هشت بیت که به‌صورت ماتریس ۴×۴ چیده شده‌اند، مرتب می‌شوند. هر ثبات حالت، با هشت فلیپ-فلاپ پیاده‌سازی می‌شود که دارای یک فعال‌ساز ENA و یک کلاک است. این ثبات‌ها مطابق شکل (۹) از سطر صفر تا سه شماره‌گذاری می‌شوند. سطر صفر در عملیات جابه‌جایی سطری تابع دور AES هیچ‌گونه جابه‌جایی داده‌ای ندارد؛ ولی در بقیه سطرها درنهایت سه جابه‌جایی موردنیاز است؛ درحقیقت به جز سطر صفر، هر سطری که x ستون به سمت چپ حرکت می‌کند. برای معکوس جابه‌جایی سطری، x-۴ ستون به سمت چپ حرکت می‌کند.



(شکل-۹): معماری تابع جابه‌جایی سطری تابع دور AES [۲۱]

بنابراین تنها با صرف سه کلاک عملیات جابه‌جایی سطری و معکوس آن با جابه‌جایی در یک جهت (به سمت چپ) می‌توان پیاده‌سازی کرد. در این حالت نیاز است تا برخی سطرها در طول عملیات جابه‌جایی سطری منجمد باشند. این کار به کمک روش کلیدزنی کلاک قابل انجام است. در این صورت ثبات‌هایی که منجمد هستند، مصرف توان پویا ندارد و مصرف توان کل هم کاهش می‌یابد. نحوه انجام این کار در جدول (۱) آمده است. مطابق جدول در عملیات رمزنگاری در کلاک صفر فقط ثبات‌های سطر سوم فعال است. در کلاک یک، ثبات‌های سطر دو و سه فعال و در کلاک دوم ثبات‌های سطر یک، دو و سه فعال هستند. مشابه همین روند برای عملیات رمزگشایی نیز در این جدول آمده است. بنابراین در طول عملیات جابه‌جایی سطری در طول سه کلاک فقط نیمی از ثبات‌ها فعال شده و مصرف انرژی داشته‌اند.

(جدول-۱): جریان داده ثبات‌های حالت در عملیات جابه‌جایی سطری و معکوس آن (F: منجمد، O: عملیاتی)

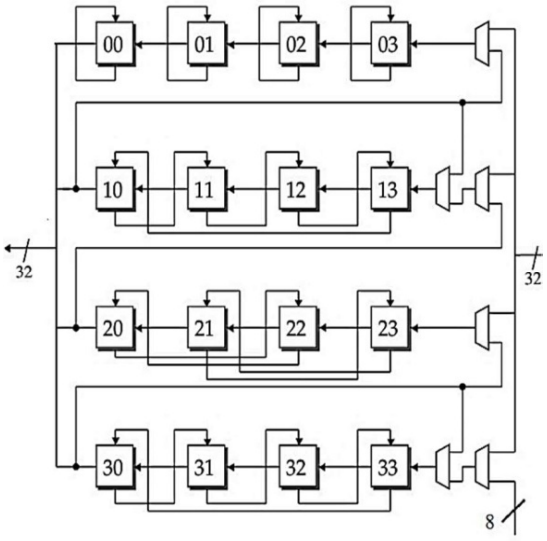
شماره سطر	کلاک‌های جابه‌جایی سطری			کلاک‌های معکوس جابه‌جایی سطری		
	0	1	2	0	1	2
0	F	F	F	F	F	F
1	F	F	O	O	F	O
2	F	O	O	O	O	F
3	O	F	O	O	F	O

بهینه‌سازی‌های نگاشت فناورانه[1]:

به‌طورکلی بهینه‌سازی مداری شامل دو گام است: سنتز منطقی و نگاشت فناوری. روش‌های بهینه‌سازی‌های منطقی شامل روش‌هایی است که در بخش ۲-۲ به آن‌ها اشاره شد و هدف آن کاهش تعداد گیت مصرفی است. همان‌طورکه در ابتدای بخش ۲ بیان شد، ابزارهای سنتز، کدهای HDL را سنتز کرده و به گیت‌های منطقی تبدیل می‌کنند. پس از آن، این گیت‌ها باید به منابع سخت‌افزار هدف نگاشت شوند که به آن نگاشت فناورانه گویند. برای فناوری با یک مقصد خاص مثل FPGA-ها یا ASIC ممکن است، مداری که از لحاظ منطقی بهینه باشد، از لحاظ فناورانه، بهینه نگاشت نشده باشد و بسیاری از منابع سخت‌افزاری را بدون استفاده بگذارد. در این حالت طراح با دانستن ویژگی‌های عناصر سخت‌افزاری مقصد، طرح را برای استفاده از آن عناصر بهینه کند.

[1] Technology Mapping Optimization

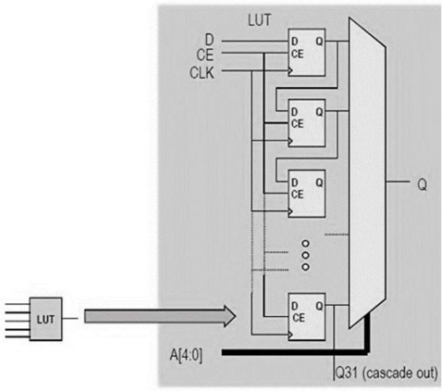
برای مثال در کتابخانه سلول‌های استاندارد ASIC، یک نوع فلیپ-فلاپ خاص به نام فلیپ-فلاپ پویشی[1] وجود دارد. این فلیپ-فلاپ‌ها دارای یک ورودی اضافی هستند که با یک خط انتخاب‌گر یکی از دو ورودی را می‌توان انتخاب کرد. این فلیپ-فلاپ‌ها را برای ساخت ثبات‌های هشت بیتی حالت و کلید تابع دور AES که به‌صورت ماتریس ۴×۴ چیده شده‌اند، می‌توان استفاده کرد. این ثبات‌های دارای دو ورودی، به چند صورت می‌توانند به هم متصل شوند. شکل (۱۰) نحوه اتصال ثبات‌ها را نشان می‌دهد. در یک وضعیت ورودی از ثبات کناری می‌تواند باشد. این سازوکار بارگذاری سریال داده‌های بایتی را تسهیل می‌کند. در وضعیت دوم اتصال، تابع جابه‌جایی سطری را می‌توان تسهیل کرد. به همین ترتیب در بخش فرانمای کلید، ثبات‌ها در دو حالت به هم متصل شده‌اند. یک حالت برای بارگذاری سریال داده و یک وضعیت برای محاسبه کلید دور به کار می‌رود.



(شکل-۱۰): اتصال ثبات‌های پویشی [۱۹]

در حالتی که فناوری مقصد FPGA باشند LUT-ها به یک ثبات انتقال[2] می‌توانند تبدیل شوند که به آن‌ها SRL[3] می‌گویند. در بیشتر FPGA-ها هر LUT یک ثبات انتقال ۱۶ یا ۳۲ بیتی را می‌تواند پیاده‌سازی کند. استفاده از این SRL-ها می‌تواند عملکرد جابه‌جایی بایتی را در بخش‌هایی از الگوریتم AES تسهیل کند [۲۲]. شکل (۱۱) ساختار یک SRL را نشان می‌دهد.

[1] Scan Flip-Flop
[2] Shift Register
[3] Shift Register LUT



(شکل-۱۱): ساختار یک SRL [۲۲]

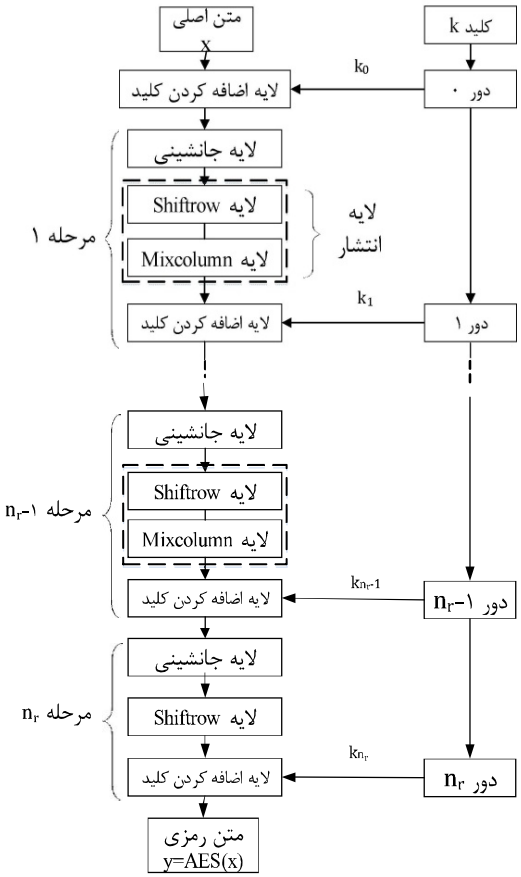
۳- مروری بر الگوریتم رمز AES

الگوریتم رمز رایندال[4] [۲۳] در یک مسابقه انتخاب رمز متقارن که توسط مؤسسه ملی فناوری و استانداردها (NIST[5]) در سال ۲۰۰۰ میلادی برگزار شد، به‌عنوان استاندارد رمزنگاری پیشرفته (AES[6]) انتخاب شد. این رمز، داده‌ها را در قالب‌های ۱۲۸ بیتی پردازش می‌کند و کلیدهایی با طول ۱۲۸، ۱۹۲ و ۲۵۶ بیت را پشتیبانی و تابع دور به‌ترتیب ۱۰، ۱۲ و ۱۴ تکرار می‌شود. در هر دور اجرای الگوریتم، داده‌ها با کلید دوری که از کلید اصلی ساخته شده است، ترکیب می‌شوند. طرح کلی عملیات رمزنگاری AES در شکل (۱۲) نشان داده شده است. حالت داخلی الگوریتم به‌صورت ماتریس ۴×۴ از بایت‌ها است که عملیات روی آن‌ها انجام می‌شود. حالت اولیه با XOR کردن کلید اصلی و قالب داده ورودی پر می‌شود. دورها شامل چهار عملیات S-box، جابه‌جایی سطری[7]، مخلوط‌سازی ستونی[8] و جمع با کلید دور[9] است. دور آخر فاقد مخلوط‌سازی ستونی است. در رمزگشایی این عملیات به‌صورت معکوس انجام می‌شود.

جانشینی بایتی یک تبدیل غیرخطی و معکوس‌پذیر است که برای نگاشت هر بایت حالت به بایت دیگر از شانزده جدول جانشینی ۲۵۶ بایتی مشابه، تشکیل شده است. مقادیر جداول S-box با محاسبهٔ معکوس ضربی در میدان متناهی $GF(2^8)$ و اعمال یک تبدیل آفینی تولید می‌شود. جانشینی بایتی می‌تواند به‌صورت جدول جستجو یا محاسباتی پیاده‌سازی شود. جابه‌جایی سطری، یک جابه‌جایی دوری به

[4] Rijndael
[5] National Institute of Standards and Technology
[6] Advanced Encryption Standard
[7] ShiftRows
[8] MixColumns
[9] Add Round Key

سمت چپ سطرهای دوم، سوم و چهارم حالت به‌ترتیب به اندازه یک، دو و سه بایت است. در مخلوط‌سازی ستونی هر ستون چهار بایتی به‌عنوان ضرایب یک چهار جمله‌ای a(x) روی میدان $GF(2^4)$ تعبیر می‌شود. هر ستون در چندجمله‌ای ثابت $c(x) = 03.x^3 + 01.x^2 + 01.x + 02$ به پیمانه $x^4 + 1$ ضرب می‌شود. در هر دور مرحله جمع با کلید دور به‌صورت XOR حالت و کلید آن دور انجام می‌شود. کلید هر دور از کلید اصلی مشتق شده و تابع اشتقاق آن شامل عملیات S-box، چرخش کلمه‌ای و عملیات XOR با ثابت آن دور، است.



(شکل-۱۲): طرح رمزنگاری AES [۲۴]

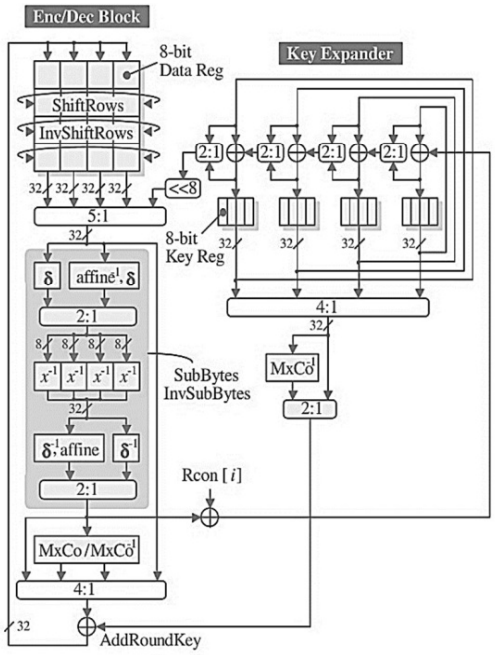
۴- مروری بر پیاده‌سازی‌های فشرده الگوریتم AES

در این بخش مهم‌ترین معماری‌های فشردهٔ ارائه‌شده از گذشته تا حال حاضر مرور خواهند شد و در هر معماری روش‌های استفاده‌شده برای پیاده‌سازی فشرده به‌صورت مختصر شرح داده خواهد شد.

ساتو[1] و همکاران در [۲۵] نخستین معماری فشرده را با مسیر داده ۳۲ بیتی در سال ۲۰۰۱ پیشنهاد دادند که در یک هسته از رمزنگاری و رمزگشایی پشتیبانی می‌شود (شکل ۱۳). این معماری با اندازه مدار ۵۴۰۰ GE و تأخیر ۵۴ سیکل برای هر عملیات رمزنگاری یا رمزگشایی و توان عملیاتی ۳۱۱ Mbps یک معماری فشرده و در عین حال با سرعت بالا است. در این معماری به‌منظور کمینه‌سازی مصرف سخت‌افزار از روش به‌اشتراک‌گذاری منابع و استفاده از یک مسیر داده برای رمزنگاری و رمزگشایی استفاده شده است. ترتیب مسیر داده برای رمزنگاری و رمزگشایی به‌صورت زیر است.

$$\delta \rightarrow x^{-1} \rightarrow \delta^{-1}\ and\ affine\ \rightarrow MixColumn$$
$$affine^{-1}\ and\ \delta^{-1} \rightarrow x^{-1} \rightarrow \delta \rightarrow InvMixColumn \quad (۱۲)$$

δ^{-1} و δ یک تابع هم‌ریختی درون میدان $GF(2^8)$ است.



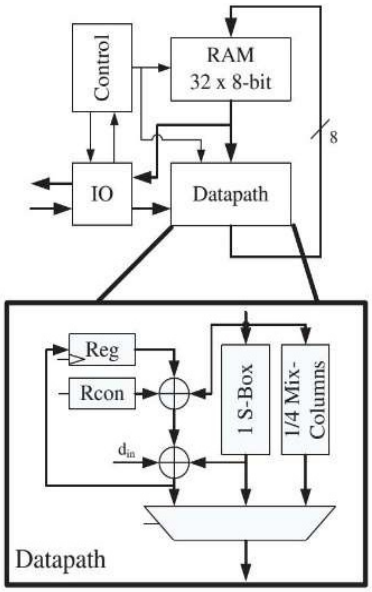
(شکل-۱۳): معماری مسیر داده [۲۵]

در این معماری برای بهینه‌سازی S-box از محاسبات روی میدان مرکب استفاده شده است. برای پیاده‌سازی S-box یک استراتژی سه مرحله‌ای به کار برده شده است. در گام نخست تمام اعضای میدان $GF(2^8)$ به میدان مرکب با تابع هم‌ریختی δ نگاشت می‌یابند. در گام دوم معکوس روی میدان مرکب محاسبه می‌شود و در گام سوم با تابع δ^{-1} نتیجه دوباره به میدان اصلی نگاشت می‌یابد. با انتقال عملیات معکوس

[1] Satoh

مخلوط‌سازی ستونی به بعد از S-box مخلوط‌سازی ستونی و معکوس آن در هم ادغام شده است.

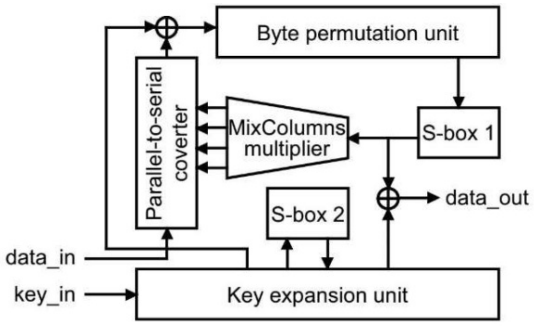
معماری ارائه‌شده در [۲۶] یک معماری فشرده با مسیر دادهٔ هشت بیتی با پشتیبانی رمزنگاری و رمزگشایی در یک هسته با اندازه مدار ۳۴۰۰ GE معادل یک دانه شن است (شکل ۱۴). این معماری دارای تأخیر ۱۰۳۲ سیکل برای عملیات رمزنگاری و ۱۱۶۵ برای عملیات رمزگشایی و توان عملیاتی ۹/۹ Mbps است. همچنین این معماری برای مصرف توان پایین در کاربردهای با توان عملیاتی کم مناسب است. معماری شامل چهار بخش کنترلی، RAM، مسیر داده و ماژول IO است. ماژول IO دارای واسط میکروکنترلی است که ماژول AES به‌عنوان کمک‌پردازندهٔ آن است. بخش کنترلی دستورهایی از ماژول IO پذیرفته و مسیر داده را تولید می‌کند. کنترلر به‌صورت ماشین حالت متناهی پیاده‌سازی شده و شامل یک شمارنده چهار بیتی و نشانی ثبات‌ها برای نشان‌دهی RAM است. RAM شامل ۳۲ سطر هشت بیتی است که ۱۲۸ بیت نخست برای ذخیره حالت و ۱۲۸ بیت دوم برای ذخیره کلید دور به کار می‌رود. در مسیر دادهٔ ماژول AES عملیات جابه‌جایی دوری و معکوس آن با نشان‌دهی صحیح RAM انجام می‌پذیرد.



(شکل-۱۴): معماری هشت بیت AES [۲۶]

با توجه به معماری هشت بیت برای پیاده‌سازی لایه جانشینی از یک S-box استفاده شده است. این S-box به‌صورت محاسباتی پیاده‌سازی شده و دارای یک مرحله خط-لوله برای کاهش مسیر بحرانی است. همچنین برای کاهش مصرف توان از حالت بیکاری برای مواقعی که به خروجی S-box نیاز نباشد، استفاده شده است.

همالاینن[1] و همکاران در [۱۲] یک هسته سخت‌افزاری برای کاهش مصرف توان و ناحیه ارائه داده‌اند (شکل ۱۵). این هسته دارای معماری هشت بیت بوده و فقط رمزنگاری را پشتیبانی می‌کند. در این هسته از فناوری ۰/۱۳ μm CMOS استفاده شده که دارای مصرف ناحیه ۳۱۰۰GE با توان عملیاتی ۱۲۱ Mbps است. این معماری دارای تأخیر ۱۶۰ سیکل و مصرف انرژی ۵/۹ nJ برای پردازش هر قالب داده است. معماری سطح بالا دارای پنج واحد مبدل موازی به سریال، جایگشت بایتی، واحد ضرب‌کنندهٔ مخلوط‌سازی ستونی، واحد فرانمای کلید و دو عدد S-box است. واحد جایگشت بایتی عملیات جابه‌جایی سطری و ذخیره‌سازی حالت‌ها صورت می‌پذیرد. در این معماری برای کاهش تعداد کل سیکل‌ها و افزایش توان عملیاتی، فرانمای کلید به‌صورت موازی با عملیات دور انجام می‌شود و درنتیجه برای هر بخش یک S-box پیاده‌سازی شده است.



(شکل-۱۵): معماری سطح بالای هسته AES [۱۲]

پیاده‌سازی سخت‌افزاری AES توسط مرادی و همکاران در [۱۸] با اندازه مداری ۲۴۰۰ GE و تأخیر ۲۵۶ سیکل یکی از کوچک‌ترین معماری‌های شناخته شده است. در این معماری فقط عملیات رمزنگاری پیاده‌سازی شده و دارای توان عملیاتی ۶۱ Kbs در فرکانس کاری ۱۰۰ KHz است. این معماری از ترکیب مسیر دادهٔ ۸ و ۳۲ بیتی به‌منظور کاهش بیشتر ناحیه مصرفی استفاده می‌کند. این طرح از فلیپ-فلاپ پویشی برای ایجاد ثبات‌های مورد استفاده در به‌روزرسانی حالت و فرانمای کلید، استفاده می‌شود. این روش باعث صرفه‌جویی ۱ GE به ازای هر فلیپ-فلاپ می‌شود. در بخش مخلوط‌سازی ستونی به جای استفاده از مسیر داده

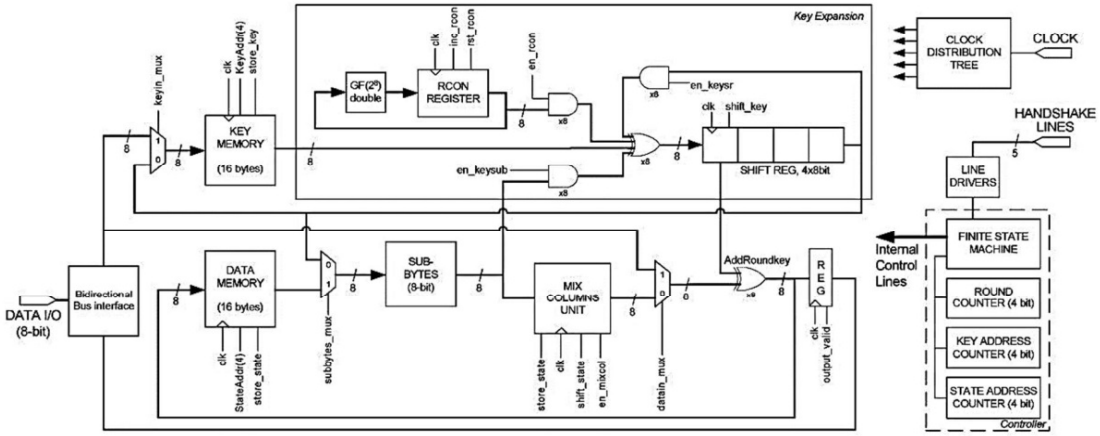
[1] Hamalainen

هشت بیتی که در ساختارهای سریال رایج است، از مدار ۳۲ بیتی استفاده شده است. به ادعای مؤلف مقدار ناحیه صرفه‌جویی شده در معماری هشت بیت توسط ثبات‌های مورد نیاز برای ذخیره‌سازی خروجی خنثی می‌شود و درحالی‌که به دو کلاک بیشتر نیاز دارد. بنابراین معماری ۳۲ بیت در این بخش ارجح است. درنهایت چون تابع دور در ۲۱ کلاک محاسبه می‌شود، سامانهٔ کنترلی از یک LFSR پنج بیتی برای تولید تمام سیگنال‌های زمان‌بندی استفاده می‌کند.

پژوهش‌های صورت‌گرفته در [۲۷] باعث ارائه یک معماری فشرده رمزنگاری/رمزگشایی با هدف کاهش هم‌زمان توان مصرفی، ناحیه و تأخیر برای کاربردهای در فرکانس کاری پایین شده است (شکل ۱۶). با درنظرگرفتن این هدف برای مقایسه کارآمدی طرح از معیار حاصل‌ضرب توان-ناحیه-تأخیر استفاده شده است. این معماری دارای مسیر داده هشت بیتی با فناوری CMOS μm ۰/۱۳ دارای ناحیه مصرفی GE ۵۶۰۰ است. این پیاده‌سازی در فرکانس کاری KHz ۱۰۰ دارای مصرف توان nW ۶۹۲ و تأخیر ۳۵۶ سیکل است. کمینه‌سازی مصرف منابع با کمک روش‌های به اشتراک‌گذاری منابع، معماری فشرده برای حافظه، بهینه‌سازی محاسبات روی میدان، اجتناب از فعالیت‌های سوییچینگ غیر لازم، کاهش در انتقال‌های حافظه و بهینه‌سازی بخش کنترلی، انجام شده است. اغلب توان مصرفی در مدارت CMOS شامل توان مصرفی پویا است که ناشی از فعالیت‌های سوییچینگ مدارات است. در طرح‌هایی مانند AES به مراتب از عملیات XOR استفاده می‌شود که به‌علت وجود مسیرهای با طول متفاوت، این عملیات فعالیت‌های سوییچینگ ناخواسته ایجاد می‌کند. در اینجا طراحان با قراردادن ثبات انتقال بین واحد S-box و مخلوط‌سازی ستونی به همراه سیگنال فعال‌ساز AND از این فعالیت‌های سوییچینگ جلوگیری کرده و درنتیجه مصرف توان کاهش یافته است. همچنین برای کاهش تعداد سیکل‌های محاسبه به جای استفاده از یک حافظه ۳۲ بایتی از دو حافظه مجزای ۱۶ بایتی برای ذخیره‌سازی مقادیر کلید و حالت استفاده شده است. این کار داده‌های دو بخش به‌صورت هم‌زمان را قابل دسترس می‌کند و بنابراین بخش فرانمای کلید در پرواز می‌تواند محاسبه شود.

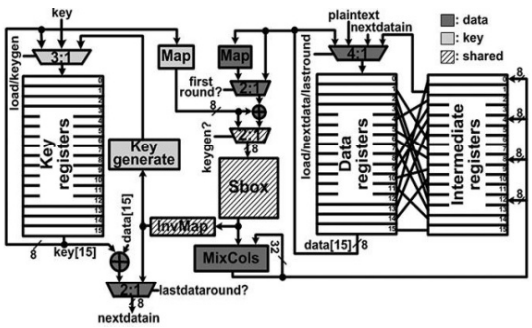
طرح ارائه‌شده در [۲۸] یک کمک‌پردازنده رمزنگاری با ولتاژ کاری پایین و فناوری ۶۵ nm LP CMOS با مصرف گیت GE ۳۵۰۰ مناسب برای برچسب‌های RFID غیرفعال است. این معماری فشرده هشت بیتی بر اساس معماری [۲۶] بوده و تنها برای کار با ولتاژهای پایین تغییر یافته است تا مصرف توان را کاهش دهد. توان اندازه‌گیری در ولتاژ کاری V ۰/۳۵ برابر μW ۰/۲۱ و توان عملیاتی kbps ۳۱ است. این کار نشان داد که پیاده‌سازی با منطق ولتاژهای بسیار پایین و استفاده از فناوری CMOS نانومتری توان مصرفی را بسیار کاهش داده و بدون صرف زمان برای بهینه‌سازی معماری، رمز AES را برای استفاده در RFIDهای غیرفعال می‌توان مناسب کرد.

ژائو و همکاران [۲۹] یک پیاده‌سازی سبک‌وزن AES با مصرف انرژی بسیار کم ارائه دادند. این طرح دارای مصرف انرژی nJ ۰/۱ در هر قالب داده با ولتاژ V ۰/۳۲ است. در این معماری ۸ بیت ارائه شده از دو S-box به‌منظور محاسبه تابع دور و فرانمای کلید استفاده شده است که به تعداد ۱۶۰ کلاک برای هر قالب داده که همان حد پایین تئوری است، دست یافته است.



(شکل-۱۶): معماری مسیر داده ۸ بیتی AES [۲۷]

در این طرح از مزیت S-box طبیعی[1] روی میدان مرکب استفاده شده است. در S-box طبیعی نیازی به نگاشت از میدان $GF(2^8)$ به میدان مرکب نبوده و بنابراین دارای سرعت و کارآمدی ناحیه بهتری است. نتایج مقایسه در این کار نشان داد که استفاده از S-box طبیعی در دو بخش تابع دور و فرانمای کلید منجر به کاهش ۲۸٪ در ناحیه می‌شود. ذخیره‌سازی نتایج محاسبات تابع دور و فرانمای کلید نیاز به حافظه دارد. در این طرح انواع حافظه‌های مورد استفاده در پیاده‌سازی‌ها مقایسه شده و از میان انواع حافظه‌های RAM یکپارچه، RAM دوتکه، ثبات انتقال و آرایه‌ای، نوع حافظه ثبات انتقال انتخاب شده است. این نوع حافظه اگر چه کمی ناحیه اشغال‌شدهٔ بالاتری دارد، ولی به‌دلیل کارآمدی عملیات انتقال دوری دارای سرعت بالاتر و به‌تبع آن مصرف انرژی پایین‌تر است.



(شکل–۱۷): AES نانو با مسیر داده هشت بیتی [۳۰]

طرح ارائه‌شده در [۳۰] که در (شکل ۱۷) نمایش داده شده، یک پیاده‌سازی فشرده AES به نام AES نانو[2] است که دارای دو هسته مجزا برای رمزنگاری و رمزگشایی است. این هسته‌ها به‌ترتیب دارای مصرف گیت ۱۹۴۷ و ۲۰۹۰ بوده و بر اساس فناوری ۲۲ نانومتری Tri-gate ساخته شده‌اند. این طرح دارای معماری هشت بیت و یک S-box روی میدان مرکب طبیعی $GF(2^4)^2$ است که تمام محاسبات تابع دور هم روی این میدان انجام می‌پذیرد. برای بهینه‌سازی ناحیه مسیر داده، تمامی چندجمله‌ای‌های ممکن روی میدان مرکب بررسی شده‌اند و یک چندجمله‌ای پایه و توسعه‌ای انتخاب شده است که ناحیه مصرفی را کمینه کند. همچنین این طرح نخستین طرحی است که از دو چندجمله‌ای متفاوت برای رمزنگاری و رمزگشایی استفاده می‌کند که منجر به کاهش ناحیه بیشتر نسبت به دیگر طرح‌ها شده است. این طرح دارای مصرف توان بسیار پایین ۳/۹ nJ برای رمزنگاری و ۲/۵ nJ برای رمزگشایی یک بلوک داده است. این طرح دارای کمینه نرخ توان عملیاتی ۴۳۲ Mbps است.

بنیک[3] و همکاران در [۱۹] طرحی را به نام AES اتمی[4] بر اساس طرح مرادی و همکاران [۱۸] ارائه دادند (شکل ۱۸). با اینکه طرح مرادی یکی از کوچک‌ترین طرح‌های پیاده‌سازی AES از لحاظ تعداد گیت (حدود ۲۴۰۰ GE) است؛ اما فقط از رمزنگاری پشتیبانی می‌کند که برای مدهایی مثل CBC که نیاز به دسترسی به هر دو ماژول رمزنگاری و رمزگشایی دارند، مناسب نیست. تفاوت این طرح با طرح [۱۸] به شرح زیر است:

- در این طرح S-box و معکوس آن در یک مدار پیاده‌سازی شده است که با یک انتخاب‌گر نوع عملیات قابل انتخاب است.
- مخلوط‌سازی ستونی و معکوس آن در یک مدار پیاده‌سازی شده است. برای این کار ماتریس معکوس مخلوط‌سازی ستونی طوری تجزیه شده است که یکی از ماتریس‌های حاصله، ماتریس مخلوط‌سازی ستونی مستقیم باشد.

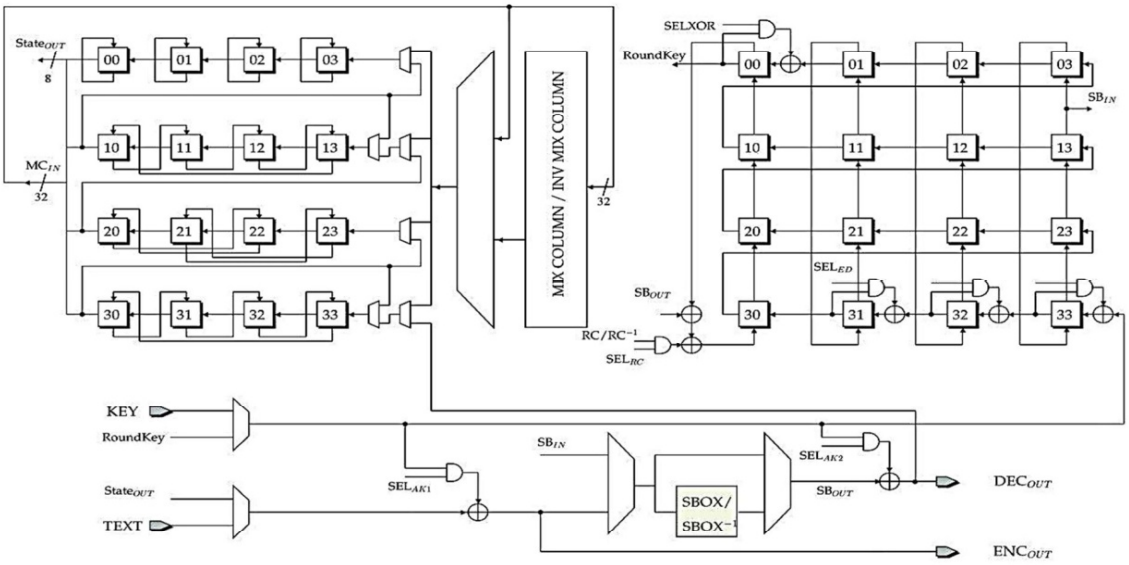
این طرح دارای مصرف کلاک ۲۲۶، مصرف انرژی ۳/۳ nJ و توان عملیاتی حداکثر ۹۴/۴ Mbps است.

بنیک و همکاران یک بهینه‌سازی بیشتری برای طرح قبلی خود در [۳۱] ارائه کردند. این طرح دارای قابلیت دوگانه رمزنگاری/رمزگشایی و مصرف گیت ۲۲۶۳ GE و تأخیر ۲۴۶ سیکل برای رمزنگاری و ۳۲۶ برای رمزگشایی است. تفاوت‌های این طرح جدید با طرح قبلی به‌شرح زیر است:

- استفاده از روش کلیدزنی کلاک[5]، عملیات جابه‌جایی دوری و معکوس آن، به جای یک کلاک در سه کلاک محاسبه می‌شود. این مسئله امکان جایگزینی بسیاری از فلیپ–فلاپ‌های پویشی را با فلیپ–فلاپ‌های معمولی که دارای ناحیه مصرفی کمتری به اندازه ۱ GE هستند، فراهم می‌کند. البته این موضوع منجر به افزایش تعدادی کلاک (۲۰ کلاک برای رمزنگاری و ۱۰۰ کلاک در رمزگشای) می‌شود.
- پیاده‌سازی معکوس مخلوط‌سازی ستونی به روش متفاوت صورت گرفته است. درحقیقت ماتریس معکوس مخلوط‌سازی ستونی AES برابر توان سوم ماتریس

[1] Native
[2] Nano-AES
[3] Banik
[4] Atomic-AES
[5] Clock Gating

مستقیم آن است؛ بنابراین با اجرای سه بار مخلوط‌سازی ستونی، معکوس آن محاسبه می‌شود و دیگر نیاز به پیاده‌سازی مداری که عملکرد مستقیم و معکوس مخلوط‌سازی ستونی را ترکیب کند، نبوده که این امر منجر به کاهش ناحیه مصرفی می‌شود. در نهایت این طرح دارای مصرف انرژی ۳/۲ nJ برای رمزنگاری و ۴/۳ nJ برای رمزگشایی است. همچنین نرخ توان عملیاتی ۸۴/۴ Mbps برای رمزنگاری و ۶۶/۷ Mbps برای رمزگشایی است. در بخش بعدی طرح‌های بررسی‌شده از لحاظ معیارهای مختلف بررسی و مزیت یا نقطه ضعف هر یک بیان خواهد شد.



(شکل-۱۸): معماری ۸ بیت AES برای Atomic AES [۱۹]

۵- مقایسهٔ نتایج پیاده‌سازی‌های فشرده الگوریتم AES

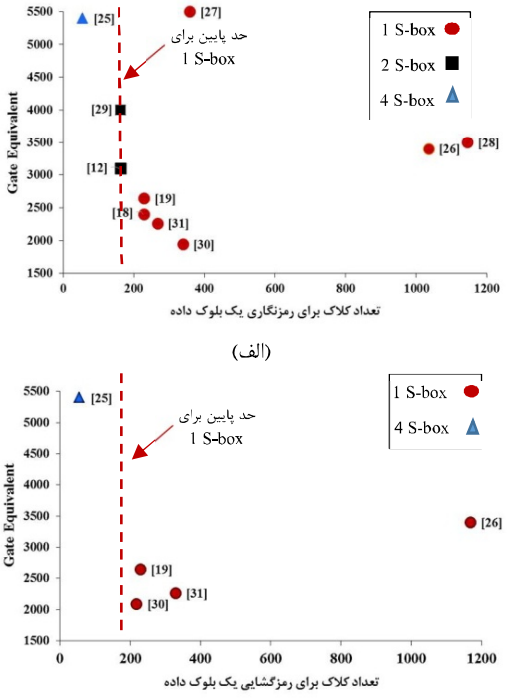
در این بخش برخی ویژگی‌های یکسان در طرح‌های مختلف مقایسه و سپس تحلیل و نتیجه‌گیری می‌گردد.

۵-۱- مقایسه ناحیه مصرفی بر اساس تعداد S-box پیاده‌سازی شده و تعداد کلاک رمزگذاری/رمزگشایی

کمینه تعداد کلاک مورد نیاز برای پیاده‌سازی فشرده طرح رمزنگاری یا رمزگشایی AES تابعی از تعداد S-box مورد استفاده است. بیشتر پیاده‌سازی‌های فشرده، دارای معماری ۸ بیت بوده و از یک S-box استفاده می‌کنند که این S-box در هر دو بخش تابع دور و فرانمای کلید به‌صورت اشتراکی استفاده می‌شود. در این حالت در هر دور یک S-box شانزده بار فراخوانی می‌شود؛ پس برای هر قالب داده ده بار تابع دور تکرار شده و درمجموع ۱۶۰ بار S-box از استفاده می‌شود. در بخش فرانمای کلید برای تولید کلید دور چهار بار از S-box استفاده می‌شود که برای تولید ده کلید دور چهل بار از آن استفاده می‌شود؛ بنابراین در حالت زمان‌بندی ایده‌آل جریان داده، کمینه تعداد کلاک مورد نیاز با به‌کارگیری یک S-box برابر دویست است.

برخی دیگر طرح‌ها از دو S-box استفاده کرده‌اند و S-box دوم در بخش فرانمای کلید استفاده می‌شود. بنابراین امکان محاسبه هم‌زمان محاسبات تابع دور و فرانمای کلید را فراهم می‌کند که در این حالت دست‌کم تعداد ۱۶۰ کلاک مورد نیاز خواهد بود. این کار باعث افزایش توان عملیاتی و ناحیه مصرفی می‌شود.

شکل (۱۹) که مستخرج از بررسی‌های انجام‌شده روی جدیدترین پیاده‌سازی‌های فشرده AES است، ارتباط تعداد کلاک، تعداد گیت مصرفی و تعداد S-box را نشان می‌دهد. طرح [۲۵] از چهار S-box استفاده کرده که برای محاسبه هر دور ۵۴ کلاک صرف شده ولی دارای مصرف گیت بسیار بالای ۵۴۰۰ GE است.



(شکل-۱۹): مقایسه مصرف گیت بر حسب تعداد کلاک و ارتباط آن با تعداد S-box برای طرح‌های فشرده AES الف)رمزنگاری، ب) رمزگشایی

در شکل (۱۹-الف) طرح‌های [۱۲] و [۲۹] با دو S-box به حد پایین تئوری ۱۶۰ کلاک دست یافته‌اند و به‌ترتیب مصرف گیت GE ۳۱۰۰ و GE ۴۰۰۰ در رمزنگاری دارند. همچنین هیچ طرحی با یک S-box به حد پایین تئوری تاکنون دست نیافته است. طرح‌های [۲۶] و [۱۸] به‌دلیل استفاده از حافظه یکپارچه برای کلید و داده و دسترسی ناکارآمد و مکرر به داده‌ها بیش از هزار کلاک برای محاسبه نیاز دارند. طرح‌های [۱۹] و [۱۸] در رمزنگاری و [۳۰] در رمزگشایی با یک S-box دارای کم‌ترین تعداد کلاک یعنی ۲۲۶ هستند. طرح [۱۹] با توجه به اینکه در یک هسته هم رمزنگاری و هم رمزگشایی را پشتیبانی می‌کند و مصرف گیتی برابر GE ۲۶۴۵ دارد، بهترین طرح شناخته‌شده از لحاظ تعداد کلاک و گیت مصرفی است. همچنین طرح [۳۱] دارای تعداد کلاک مصرفی۲۶۴ با پایین‌ترین گیت مصرفی GE ۲۲۶۳ است.

۲-۵- مقایسه مصرف ناحیه به تفکیک اجزا

پیاده‌سازی‌های فشرده به‌منظور دستیابی به مصرف گیت کم به‌طور معمول از مسیر داده و بخش کنترلی متناسب با آن استفاده می‌کنند. معماری‌های فشرده دارای برخی مشخصات اختصاصی هستند که آن‌ها را از طرح‌های با گذردهی بالا متمایز می‌کنند. جدول (۲) مصرف گیت طرح‌های فشرده را به‌صورت تفکیک اجزا نشان می‌دهد. همان‌طور که مشاهده می‌شود، سهم S-box برخلاف طرح‌های با گذردهی بالا، در طرح‌های فشرده غالب نبوده، ولی حافظه جهت ذخیره‌سازی بلوک داده و کلید ۱۲۸ بیتی، بزرگ‌ترین سهم را دارد.

در این طرح‌ها از چهار نوع حافظه RAM یک‌پارچه، RAM دوتکه، ثبات انتقال و آرایه‌ای استفاده شده است. در حافظه RAM یک‌پارچه، کلید و داده به‌صورت ترتیبی از یک حافظه RAM واکشی و پردازش‌شده و داخل یک بلوک RAM ذخیره می‌شوند. طرح‌های [۲۶] و [۲۸] از این نوع حافظه استفاده می‌کنند که به‌علت ساختار حافظه، دسترسی به داده‌ها ناکارآمد بوده و تعداد کلاک بالا برای پردازش نیاز است. درنتیجه طرح دارای توان عملیاتی کم است.

در حافظه RAM دوتکه، دو حافظه مجزای ۱۲۸ بیتی برای ذخیره کلید و داده وجود دارد که امکان انجام فرانمای کلید در حین محاسبه تابع دور با استفاده از یک S-box را میسر می‌کند که در این حالت تعداد کلاک مورد نیاز کاهش یافته و درنتیجه توان عملیاتی افزایش می‌یابد.

طرح [۲۷] دارای این نوع حافظه است که باعث شده تعداد کلاک مورد نیاز از ۱۰۳۲ به ۳۵۶ کلاک کاهش یابد؛ ولی ۳۸ درصد ناحیه افزایش یافته است.

در پیاده‌سازی حافظه به‌صورت ثبات انتقال عملیات جابه‌جایی دوری به‌صورت بسیار کارآمدی انجام می‌پذیرد. ثبات انتقال‌ها به‌طورمعمول به‌صورت فلیپ-فلاپ نوع D پیاده‌سازی می‌شوند. طرح‌های [۱۲]، [۲۵] و [۳۰] از این نوع حافظه بهره می‌برند. در طرح [۱۲] از دو S-box استفاده شده که باعث افزایش دوازده برابری توان عملیاتی و در عین حال کاهش هفده درصدی ناحیه نسبت به [۲۶] شده است.

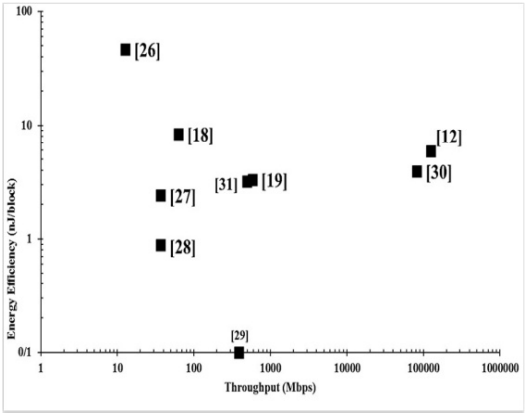
حافظه از نوع آرایه‌ای، مشابه حافظه ثبات انتقالی بوده ولی حافظه به‌صورت یک آرایه ۴×۴ چیده شده که هر درایه یک ثبات انتقال هشت بیتی است. اطلاعات در آرایه می‌تواند به‌صورت عمودی یا افقی جابه‌جا شود. طرح‌های [۱۸]، [۱۹] و [۳۱] از این نوع حافظه بهره می‌برند. طرح [۱۸] نخستین بار از این نوع حافظه استفاده کرده که به مصرف گیت GE ۲۴۰۰ دست یافت. طرح‌های [۱۹] و [۳۱] با اضافه کردن قابلیت رمزگشایی و انجام بهینه‌سازی روی این طرح به‌ترتیب به مصرف گیتKGE ۲/۶ و KGE ۲/۲ دست یافتند.

(جدول-۲): مصرف گیت تفکیک شده طرح‌های رمزنگاری فشرده AES

نوع حافظه	مرجع	حافظه	S-box	مخلوط‌سازی ستونی	کنترلی و ...	تعداد کلاک/بلاک	گیت مصرفی (GE)
RAM یکپارچه	[۲۶] و [۲۸]	۶۰٪	۱۲٪	۹٪	۲۹٪	۱۰۳۲	۳/۴ K
RAM دو تکه	[۲۷]	۵۵٪	۷٪	۸٪	۳۰٪	۳۵۶	۵/۵ K
ثبات انتقال	[۱۲]	۵۹٪	۱۶٪	۹٪	۱۶٪	۱۶۰	۳/۱ K
	[۲۵]	۳۳٪	۲۱٪/۸	۱۱٪/۹	۳۴٪/۳	۵۴	۵/۴ K
	[۳۰]	۵۰٪				۳۳۶	۱/۹ K
آرایه‌ای	[۱۸]	۶۴٪	۹٪	۱۴٪	۱۳٪	۲۲۶	۲/۴ K
	[۱۹]	۵۵٪/۴	۹٪/۶	۱۲٪/۲	۲۲٪/۸	۲۲۶	۲/۶ K
	[۳۱]	۵۳٪/۹	۱۱٪/۲	۸٪/۵	۲۶٪/۴	۲۶۴	۲/۳ K

۳-۵- مقایسه موازنه انرژی و توان عملیاتی

بین توان عملیاتی و مصرف انرژی ارتباط مستقیمی وجود دارد. یعنی اگر بخواهیم توان عملیاتی افزایش یابد، مصرف توان نیز افزایش می‌یابد و یا اگر بخواهیم توان مصرفی کم شود توان عملیاتی هم کاهش می‌یابد؛ درنتیجه باید یک موازنه بین این دو برقرار کنیم. شکل (۲۰) موازنه مصرف انرژی و توان عملیاتی را برای طرح‌های بررسی‌شده در این پژوهش، نشان می‌دهد.

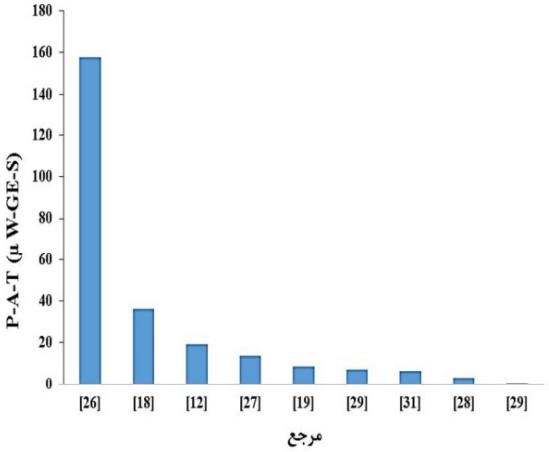


(شکل-۲۰): مقایسه مصرف انرژی در برابر توان عملیاتی طرح‌های فشرده AES

طرح [۲۹] با مصرف انرژی ۰/۱ nJ به‌ازای هر قالب داده با توان عملیاتی ۳۷۶ kbps کم‌مصرف‌ترین طرح ارائه‌شده تاکنون است که این مصرف توان پایین، بیشتر نتیجه به‌کارگیری روش‌های مداری CMOS است. از طرف دیگر طرح [۱۲] با مصرف انرژی ۵/۹ nJ دارای بیشترین توان عملیاتی با مقدار ۱۲۱ Mbps است. این توان عملیاتی بالا به‌دلیل به‌کارگیری دو S-box که منجر به کاهش تعداد کلاک می‌شود، حاصل شده است. طرح [۱۹] نیز با مصرف انرژی nJ ۳/۳ با توان عملیاتی ۵۶۶ kbps دارای بهترین موازنه انرژی و توان عملیاتی است.

۴-۵- مقایسه طرح‌ها با معیار P-A-T

در برخی مراجع مانند [۲۷] از معیار حاصل‌ضرب توان مصرفی، ناحیه و زمان (P-A-T[1]) برای مقایسه و ارزیابی استفاده شده است. در این صورت طرحی که حاصل‌ضرب کمتری نسبت به بقیه داشته باشد، مطلوب‌تر است؛ زیرا کمینه‌شدن این مقدار بدین معنی است که یک، دو یا هر سه مقدار توان، ناحیه و زمان نسبت به بقیه طرح‌ها پایین‌تر است که این در یک طرح فشرده مطلوب است. در شکل (۲۱) طرح‌های مرورشده در اینجا از لحاظ معیار P-A-T مقایسه شده‌اند. در این مقایسه، طرح [۲۹] از این لحاظ این معیار دارای کمترین مقدار یعنی ۰/۴ است. اگر چه این طرح از لحاظ ناحیه، از برخی طرح‌ها بزرگ‌تر است، اما به‌دلیل توان مصرفی بسیار پایین این طرح، مقدار P-A-T آن بسیار پایین است.



(شکل-۲۱): مقایسه P-A-T طرح‌های فشرده AES

[1] Power-Area-Time

۶- نتیجه‌گیری

پیاده‌سازی‌های فشرده AES از لحاظ میزان مصرف ناحیه، انرژی، توان و توان عملیاتی تاکنون به‌صورت جامع بررسی نشده‌اند. بنابراین در این مقاله مروری بر برجسته‌ترین پیاده‌سازی‌های فشرده AES صورت گرفت. نتایج بررسی نشان داد که در طرح‌های فشرده برای کاهش ناحیه مصرفی یک یا دو S-box پیاده‌سازی می‌شود که در این حالت تعداد کلاک برای محاسبه یک عملیات رمزنگاری یا رمزگشایی در حالت تئوری به ترتیب ۲۰۰ و ۱۶۰ است که تاکنون طرحی که با یک S-box با ۲۰۰ کلاک عملیات رمزنگاری را محاسبه کند ارائه نشده که در این زمینه جای پژوهش بیشتری وجود دارد؛ همچنین نوع حافظه پیاده‌سازی‌شده برای ذخیره‌سازی مقادیر میانی داده و کلید بر ناحیه مصرفی و تعداد کلاک مصرفی تأثیرگذار است؛ به‌طوری‌که حافظه از نوع RAM یکپارچه بدترین انتخاب است و به نظر می‌رسد حافظه از نوع آرایه‌ای برای این منظور مناسب است.

با توجه به این‌که مصرف کم انرژی در طرح‌های فشرده بسیار مهم است، باید یک موازنه بین مصرف انرژی و توان عملیاتی صورت پذیرد تا یک طرح بتواند با توان عملیاتی مناسب طرح‌های فشرده دارای مصرف انرژی کم باشد. همچنین برای مقایسه هم‌زمان توان مصرفی، ناحیه و زمان تأخیر از معیار P-A-T استفاده شد که طرحی که بتواند هر سه مقدار آن نسبت به دیگر طرح‌ها کمتر باشد، مناسب‌تر خواهد بود.

۷- مراجع

[1] D. Hong, J. Sung, S. Hong, J. Lim, S. Lee, B. Ko, C. Lee, D. Chang, J. Lee- K. Jeong, H. Kim, J. Kim, S. Chee. HIGHT: A New Block Cipher Suitable for Low-Resource Device. In CHES 2006, LNCS, vol. 4249, pp. 46-59, 2006.

[2] C. De Canniere, O. Dunkelman, M. Knezevic. KATAN and KTANTAN - a family of small and ecient hardware-oriented block ciphers. In CHES 2009, LNCS, vol.5747, pp. 272-288, 2009.

[3] Z. Gong, S. Nikova, Y.W. Law. KLEIN: a new family of lightweight block ciphers. In RFIDSec 2011, LNCS, vol. 7055, pp. 1-18, 2011.

[4] J. Guo, T. Peyrin, A. Poschmann, M. J. B. Robshaw. The LED Block Cipher. In CHES 2011, LNCS, vol. 6917, pp. 326-341, 2011.

[5] J. Daemen, M. Peeters, G. V. Assche, V. Rijmen. Nessie Proposal: NOEKEON. Available at http://gro.noekeon.org/Noekeon-spec.pdf.

[6] A. Bogdanov, L. Knudsen, G. Leander, C. Paar, A. Poschmann, M. Robshaw, Y. Seurin, C. Vikkelsoe. PRESENT: An Ultra-Lightweight Block Cipher. In CHES 2007, LNCS, vol. 4727, pp. 450-466, 2007.

[7] K. Shibutani, T. Isobe, H. Hiwatari, A. Mitsuda, T. Akishita, T. Shirai. Piccolo: An Ultra-Lightweight Blockcipher. In CHES 2011, LNCS, vol. 6917, pp. 342-357, 2011.

[8] J. Borgho, A. Canteaut, T. Guneysu, E. B. Kavun, M. Knezevic, L. R. Knudsen, G. Leander, V. Nikov, C. Paar, C. Rechberger, P. Rombouts, S. S. Thomsen, T. Yalcin. PRINCE - A Low-Latency Block Cipher for Pervasive Computing Applications - Extended Abstract. In Asiacrypt 2012, LNCS, vol. 7658, pages 208-225, 2012.

[9] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, L. Wingers. The Simon and Speck Families of Lightweight Block Ciphers. In IACR eprint archive. Available at https://eprint.iacr.org/2013/404.pdf.

[10] T. Suzaki, K. Minematsu, S. Morioka, E. Kobayashi. TWINE: A Lightweight Block Cipher for Multiple Platforms. In SAC 2012, LNCS, vol. 7707, pp. 339-354, 2012.

[11] P. Chodowiec, K. Gaj. Very Compact FPGA Implementation of the AES Algorithm. In CHES 2003, LNCS, vol. 2779, pp. 319-333, 2003.

[12] P. Hamalainen, T. Alho, M. Hannikainen, and T. D. Hamalainen. Design and Implementation of Low-Area and Low-Power AES Encryption Hardware Core. In DSD, pages 577-583, 2006.

[13] A. Lutz, J. Treichler, F. Gurkaynak, H. Kaeslin, G. Basler, A. Erni, S. Reichmuth, P. Rommens, S. Oetiker, W. Fichtner. 2Gbit/s hardware realizations of RIJNDAEL and SERPENT: A comparative analysis. In CHES 2002, LNCS, vol. 2523, pp. 144158, 2002.

[14] R. Ueno, S. Morioka, N. Homma, T. Aoki. A High Throughput/Gate AES Hardware Architecture by Compressing Encryption and Decryption Datapaths – Toward Efficient CBC-Mode Implementation. In CHES 2016, LNCS, vol. 9813, pp. 538-558, 2016.

[15] S. Banik, A. Bogdanov, F. Regazzoni. Exploring Energy Efficiency of Lightweight Block Ciphers. In SAC 2015, LNCS, vol. 9566, pp. 178-194, 2015.

[16] S. Banik, A. Bogdanov, F. Regazzoni, T. Isobe, H. Hiwatari, T. Akishita. Round gating for low energy block ciphers. In IEEE Hardware Oriented Security and Trust (HOST), pp. 55-60, 2016.

[17] Ç. K. Koç, cryptographic engineering, New York, Springer, 2009.

[18] A. Moradi, A. Poschmann, S. Ling, C. Paar, H. Wang. Pushing the Limits: A Very Compact and a Threshold Implementation of AES. In Eurocrypt 2011, LNCS, vol. 6632, pp. 69-88, 2011.

[19] S. Banik, A. Bogdanov, F. Regazzoni, Atomic-AES: A Compact Implementation of the AES Encryption/Decryption Core, In IACR eprint archive. Available at http://epr-int.ia-cr.org/2016/927.pdf.

[20] D. Canright, A Very Compact S-Box for AES. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 441–455, 2005.

[21] S. Banik, A. Bogdanov, F. Regazzoni, Round gating for low energy block ciphers, In Hardware Oriented Security and Trust (HOST), IEEE International Symposium, 2016.

[22] M. Khairallah, A. Chattopadhyay, T. Peyrin, Looting the LUTs : FPGA Optimization of AES and AES-like Ciphers for Authenticated Encryption, In IACR eprint archive. Available at http://eprint.iacr.org/2017/1019.

[23] J. Daemen, V. Rijmen. The design of Rijndael: AES - the Advanced Encryption Standard. Springer-Verlag, 2002.

[24] V. L. Dao, V. P. Hoang, A.T. Nguyen, and Q. M. Le, Eds., A compact, low power AES core on 180nm CMOS process, in International Conference on IC Design and Technology (ICICDT), IEEE, 2016.

[25] A. Satoh, S. Morioka, K. Takano, S. Munetoh. A Compact Rijndael Hardware Architecture with S-Box Optimization, In Asiacrypt 2001, LNCS, vol. 2248, pp. 239-254, 2001.

[26] M. Feldhofer, J. Wolkerstorfer, V. Rijmen, AES Implementation on a Grain of Sand. In IEEE Proceedings of Information Security, vol. 152(1), pp. 13-20, 2005.

[27] T. Good, M. Benaissa, 692-nW advanced encryption standard (AES) on a 0.13-µm CMOS, IEEE Transaction Very Large Scale Integration (VLSI) Syst., vol. 18, no. 12, pp. 1753-1757, 2010.

[28] C. Hocquet, D. Kamel, F. Regazzoni, J. D. Legat, D. Flandre, D. Bol, F. X. Standaert, Harvesting the potential of nano-CMOS for lightweight cryptography: an ultra-low-voltage 65 nm AES coprocessor for passive RFID tags, Springer Journal of Cryptography Engineering, vol. 1, no. 1, pp. 79-89, 2011.

[29] W. Zhao, Y. Ha, M. Alioto, AES Architectures for Minimum-Energy Operation and Silicon Demonstration in 65nm with Lowest Energy per Encryption, in IEEE International Symposium on Circuits and Systems (ISCAS), 2015.

[30] S. Mathew, S. Satpathy, V. Suresh, M. Anders, H. Kaul, A. Agarwal, S. Hsu, G. Chen, R.K. Krishnamurthy. 340 mV{1.1V, 289 Gbps/W, 2090-gate nanoAES hardware accelerator with area-optimized encrypt/decrypt GF(24)2 polynomials in 22 nm tri-gate CMOS. In IEEE Journal of Solid-State Circuits, vol. 50, pp. 1048-1058, 2015.

[31] S. Banik, A. Bogdanov, F. Regazzoni, Atomic-AES v2.0, In IACR eprint archive. Available at http://eprint.iacr.org/2016/1005.

محسن جهانبانی هم‌اکنون به‌عنوان دانشجوی دکترای ریاضی-رمز در دانشگاه جامع امام حسین (ع) مشغول به تحصیل است. مدارک کارشناسی و کارشناسی ارشد را به‌ترتیب در رشته مهندسی برق الکترونیک و مهندسی مخابرات-رمز در سال‌های ۸۴ و ۸۸، دریافت کرده است. علایق پژوهشی ایشان پیاده‌سازی سخت‌افزاری الگوریتم‌های رمزنگاری است.

دکتر نصور باقری هم‌اکنون به‌عنوان استادیار در دانشکده مهندسی برق دانشگاه شهیدرجایی فعالیت می‌کند. ایشان کارشناسی خود را در رشته مهندسی برق از دانشگاه مازندران و کارشناسی ارشد و دکترای خود را در همین رشته از دانشگاه علم و صنعت دریافت کرده است. علایق پژوهشی ایشان شامل تحلیل و طراحی طرح‌های رمزنگاری متقارن، فناوری RFID و پروتکل‌های امنیتی است.

دکتر زین‌العابدین نوروزی هم‌اکنون به‌عنوان استادیار در دانشکده فناوری اطلاعات و ارتباطات دانشگاه جامع امام حسین (ع) فعالیت می‌کند. ایشان مدرک دکترای ریاضی گرایش رمز را از دانشگاه خوارزمی دریافت کرده است. علایق پژوهشی ایشان در حوزهٔ الگوریتم‌ها و پروتکل‌های رمزنگاری و پنهان‌نگاری است و مقالات متعددی را در این حوزه منتشر کرده است.