پیاده‌سازی الگوریتم رایندال مبتنی بر معماری ضربه‌ای

علی فانیان، مهدی برنجکوب و شادرخ سماوی
دانشکده‌ی برق و کامپیوتر، دانشگاه صنعتی اصفهان

نویسنده‌ی عهده‌دار مکاتبات: علی فانیان

چکیده :

پس از معرفی رایندال به‌عنوان الگوریتم رمز استاندارد در اکتبر سال ۲۰۰۰ میلادی توسط NIST، استفاده از آن به‌طور گسترده در کاربردهای مختلف سخت‌افزاری و نرم‌افزاری مورد توجّه قرار گرفت. در این مقاله دو پیشنهاد بر مبنای مدل ضربه‌ای برای پیاده‌سازی الگوریتم رایندال برروی FPGA ارائه خواهد شد، که از نظر حجم سخت‌افزار مصرفی و نرخ گذردهی کارآمد است. در روش اول یک واحد سخت‌افزاری مبتنی بر مدل ضربه‌ای طراحی شده که دورهای مختلف الگوریتم رایندال در آن با استفاده از مکانیزم فیدبک پیاده‌سازی می‌شود و از این رمزکننده می‌توان در سرعت‌های متوسط استفاده نمود. این رمزکننده قادر است دو قطعه را به‌طور همزمان دریافت نماید تا عملیات رمزگذاری بر روی آن‌ها به موازات یکدیگر و با استفاده اشتراکی از منابع انجام گیرد. روش دوم برای کاربردهای پرسرعت طراحی شده است که در آن توأمان از مدل ضربه‌ای و معماری خط لوله استفاده شده است. نتایج سنتز رمزکننده‌های پیشنهادی،گویای صحت عملکرد و کارآیی مناسب روش‌ها می‌باشد و حداکثر نرخ گذردهی 38.272 Gbps را برآورده می‌سازد.

واژه‌های کلیدی: رمزنگاری، رایندال ، AES ، FPGA، مدل ضربه‌ای

۱- مقدمه

رمزنگاری[1] نقش مهمی در زمینه‌ی امنیت اطلاعات ایفا می‌کند. اطلاعات حساسی را که قرار است در محیط‌های باز و ناامن مبادله و یا ذخیره شوند، می‌توان رمز نمود. الگوریتم‌های رمزنگاری در دو دسته‌ی کلی الگوریتم‌های متقارن(کلید پنهان) و الگوریتم‌های نامتقارن(کلید عمومی) دسته‌بندی می‌شوند[۱]. الگوریتم‌های رمزنگاری متقارن سریع‌تر بوده و معمولاً برای رمزگذاری داده‌های پرحجم استفاده می‌شوند؛ درحالی‌که الگوریتم‌های نامتقارن معمولاً برای رمزکردن کلیدهای جلسه و یا انجام امضا، مورد استفاده قرار می‌گیرند. تا قبل از سال ۲۰۰۰ میلادی الگوریتم DES[2] به‌عنوان الگوریتم استاندارد به‌طور گسترده در کاربردهای امنیتی استفاده می‌گردید.

با توجه به این‌که طول کلید در این الگوریتم ۵۶ بیت است، سطح مطلوبی از امنیت برای برخی از کاربردها فراهم نمی‌شد. برای دستیابی به یک الگوریتم مناسب رمزنگاری به‌دنبال فراخوانی که NIST[3] انجام داد، الگوریتم‌های مختلفی به رقابت پرداختند[۲]، که درنهایت الگوریتم *رایندال*[4] به‌عنوان الگوریتم رمزنگاری جدید انتخاب شد[۳]. پس از آن پیاده‌سازی این الگوریتم بر روی سکوهای مختلف آغازشد. در یک دسته‌بندی کلی پیاده‌سازی‌های رایندال به دو دسته پیاده‌سازی نرم‌افزاری و پیاده‌سازی سخت‌افزاری تقسیم می‌شوند. با انجام برخی ملاحظات در پیاده‌سازی نرم‌افزاری، می‌توان سرعت پردازش را تا حدی افزایش داد. برای مثال جهت کاهش زمان پردازش در محاسبه برخی توابع منطقی، می‌توان بجای پیاده‌سازی توابع، از جدول‌های از پیش آماده استفاده نمود. در این روش کافی است به‌ازای

[1] Cryptography
[2] Data Encryption Standard
[3] National Institute of Standards and Technology
[4] Rijndeal

کلیه‌ی حالات ورودی، خروجی‌ها را به‌دست آورد و در جدول‌های مربوطه قرار داد. اما در افزایش سرعت الگوریتم با محدودیت‌های سخت‌افزاری کامپیوتر روبه‌رو خواهیم شد که این محدودیت‌ها مانع دستیابی به سرعت‌های بالا هستند. محدودیت‌هایی مانند وابستگی داده‌ها در طول پردازش، در اختیار نداشتن منابع مورد نیاز در هر لحظه و خاصیت تک‌پردازنده‌ای و اجرای پی‌درپی دستورها باعث می‌شود که نتوان در پیاده‌سازی‌های نرم‌افزاری، به سرعت‌های بالایی دست یافت. با توجّه به این محدودیت‌ها در پیاده‌سازی نرم‌افزاری الگوریتم رایندال، سرعتی معادل 100Mbps بر روی پردازنده‌ی Pentium200-MHz به‌دست آمده است[۳]. در پیاده‌سازی دیگری که با زبان اسمبلی بر روی پردازنده‌ی Pentium 458-MHz [۴] انجام شده است، سرعت 243Mbps به‌دست آمده است.

اما از آنجا که سرویس‌های محرمانگی از سطح برنامه‌های کاربردی فراتر رفته و در حوزه‌ی امنیت بستر نیز گسترش پیدا کرده‌اند، نمی‌توان تنها به پیاده‌سازی‌های نرم‌افزاری بسنده نمود. از این‌رو استفاده از الگوریتم‌های رمزنگاری در محیط‌های پرسرعتی همانند خطوط ارتباطی شبکه‌ها، سیستم‌های VPN[1] و مسیریاب‌ها به یک چالش جدی پژوهشی تبدیل شده است[۵]. در این راستا در کاربردهای پرسرعت می‌توان با بهره‌گیری از شتاب دهنده‌های سخت‌افزاری[2] به سرعت‌های بالایی دست یافت. در طراحی شتاب‌دهنده‌های سخت‌افزاری، معمولاً از سخت‌افزارهای قابل برنامه‌ریزی[3] استفاده می‌گردد.

سخت‌افزارهای قابل برنامه‌ریزی در خانواده‌های مختلفی قابل دسترس هستند که معروف‌ترین آن‌ها خانواده‌ی FPGA[4] است. در FPGAها مدارهای منطقی پایه و پرکاربردی مانند فلیپ‌فلاپ‌ها، دروازه‌های منطقی، LUT[5] و حافظه‌های RAM وجود دارند که امکان پیاده‌سازی سخت‌افزارهای مختلف را با معماری دلخواه فراهم می‌نمایند. برای پیاده‌سازی سخت‌افزار بر روی FPGAها، می‌توان از امکانات محیط‌های توسعه‌ی این تراشه‌ها استفاده کرد و عملکرد سخت‌افزار را قبل از پیاده‌سازی فیزیکی، شبیه‌سازی[6] نمود.

در این مقاله پس از مقدمه، ابتدا ساختار و معماری الگوریتم رایندال مرور خواهد شد؛ پس از آن، در بخش ۳ به بررسی معماری‌های مختلفی می‌پردازیم که در پیاده‌سازی الگوریتم رایندال ارائه شده است. در بخش(۴) معماری ضربه‌ای پیشنهادی برای الگوریتم رایندال ارائه خواهد شد. در بخش(۵) رمزکننده‌ی پیشنهادی مبتنی بر فیدبک ارائه می شود و عملکرد آن مورد ارزیابی قرار خواهد گرفت. پس از آن در بخش(۶)، معماری پیشنهادی مبتنی بر معماری خط لوله ارائه می‌گردد و در انتها نیز به‌عنوان نتیجه‌گیری مطالب، مقاله مرور می‌گردند.

۲- ساختار الگوریتم رایندال

الگوریتم رایندال یک الگوریتم رمز قطعه‌ای با طول کلیدهای مختلف متغیر بین ۱۲۸ ، ۱۹۲ و ۲۵۶ بیت و تعداد دورهای ۱۰ ، ۱۲ و ۱۴ است. اگرچه ارزیابی‌های انجام گرفته توسط NIST برای طول قطعه، ۱۲۸ بیتی بوده است اما در رایندال می‌توان از طول کلیدها و قطعه‌های ۱۹۲ و ۲۵۶ بیتی نیز استفاده کرد. در ادبیات مربوط به این الگوریتم، طول قطعه و کلید ۱۲۸ بیت و تعداد دور ۱۰، مورد توجه است. ساختار الگوریتم رایندال برای عملیات رمزگذاری و رمزگشایی در شکل (۱)نشان داده شده است. در این الگوریتم قطعه ورودی در هر دور به‌صورت یک ماتریس ۴×۴ بایتی در نظر گرفته می‌شود تا پردازش هر واحد بر روی این ماتریس انجام گیرد. در ادامه، واحدهای تشکیل دهنده‌ی این الگوریتم مرور می‌شوند.

۲-۱- واحدهای ByteSubstitution و InvByteSubstitution

واحد ByteSubstitution شامل ۱۶ جعبه‌ی جانشینی[7] مشابه مشابه ۸×۸ بیت است که هر جعبه‌ی جانشینی مطابق با (شکل ۲) بر روی یک بایت از داده‌ها عمل می‌کند. وظیفه این واحد، انجام عملیات غیرخطی بر روی داده‌های ورودی است. برای به‌دست آوردن خروجی جعبه‌ی جانشینی، ابتدا معکوس ضربی ۸ بیت ورودی در $GF(2^8)$ و در پیمانه‌ی چندجمله‌ای مولد $m(x)$ که در رابطه‌ی (۱) آمده است، محاسبه می‌شود و پس از آن از یک تبدیل مستوی[8] عبور داده می‌شود. از آنجا که در الگوریتم رایندال هر ۱۶ واحد

[1] Virtual Private Network
[2] Hardware Accelerator
[3] Reconfigurable Hardware
[4] Field Programmable Gate Array
[5] Look Up Table
[6] Simulation
[7] Substitution Box(S-Box)
[8] Affine

جعبه‌ی جانشینی مشابه هستند در پیاده‌سازی‌ها معمولاً به‌جای انجام عملیات ریاضی و منطقی از جدول‌های از پیش آماده شده استفاده می‌کنند. در این جدول، خروجی واحد جعبه‌ی جانشینی را برای تمامی ۲۵۶ حالت ورودی، محاسبه می‌کنند تا در مراحل بعدی با مراجعه به جدول، مقدار متناظر ورودی را به‌دست آورند.

$$m(x) = x^8 + x^4 + x^3 + x + 1 \qquad (۱)$$

برای معکوس‌سازی جعبه‌های جانشینی در عملیات InvByteSubstitution، ابتدا ۸ بیت ورودی از معکوس تبدیل مستوی عبور داده شده و پس از آن در $GF(2^8)$ عمل معکوس‌گیری انجام می‌شود. در این جا نیز می‌توان از جدول‌های از پیش آماده شده برای به‌دست آوردن خروجی استفاده نمود.

۲-۲- واحدهای ByteRotation و InvByteRotation

واحدهای ByteRotation و InvByteRotation برای انجام انتقال بایتی بر روی ماتریس ورودی مورد استفاده قرار می‌گیرد. شکل‌های ۳ و ۴ نحوه‌ی انجام انتقال ورودی در این واحدها را نشان می‌دهند. در پیاده‌سازی‌های سخت‌افزاری این واحدها، فقط کافی است مسیر سیمی مناسبی با توجّه به موقعیت هر بایت پس از انجام شیفت، از ورودی به خروجی واحد اعمال نمود و بنابراین نیازی به استفاده از عناصر منطقی نیست.

۲-۳- واحدهای MixColumns و InvMixColumns

واحد MixColumns برای انجام عملیات جایگشت[1] بر روی ماتریس ورودی آن مورد استفاده قرار می‌گیرد. برای انجام جایگشت بر روی داده‌ها، ماتریس انتقال در ستون‌های ماتریس ورودی در میدان $GF(2^8)$ ضرب می‌شوند. در (شکل‌های ۵ و ۶) ماتریس انتقال به ترتیب برای عملیات رمزگذاری و رمزگشایی آمده است.

عملیات ضرب ماتریس جایگشت با ستون‌ها در پیمانه‌ی چندجمله‌ای مولد m(x) (رابطه (۱)) انجام می‌شود. معمولاً در پیاده‌سازی واحد MixColumn و معکوس آن به‌جای استفاده از ضرب کننده از تابع خطی xTime [۳] استفاده می‌شود که در این تابع، عملیات ضرب بایت ورودی در بایت 02 (متناظر باچند جمله‌ای $p(x) = x$) انجام می‌گیرد. برای مثال اگر بایت b در چندجمله‌ای x ضرب گردد، نتیجه‌ی حاصل‌ضرب آن‌ها در رابطه‌ی (۲) نشان داده شده است.

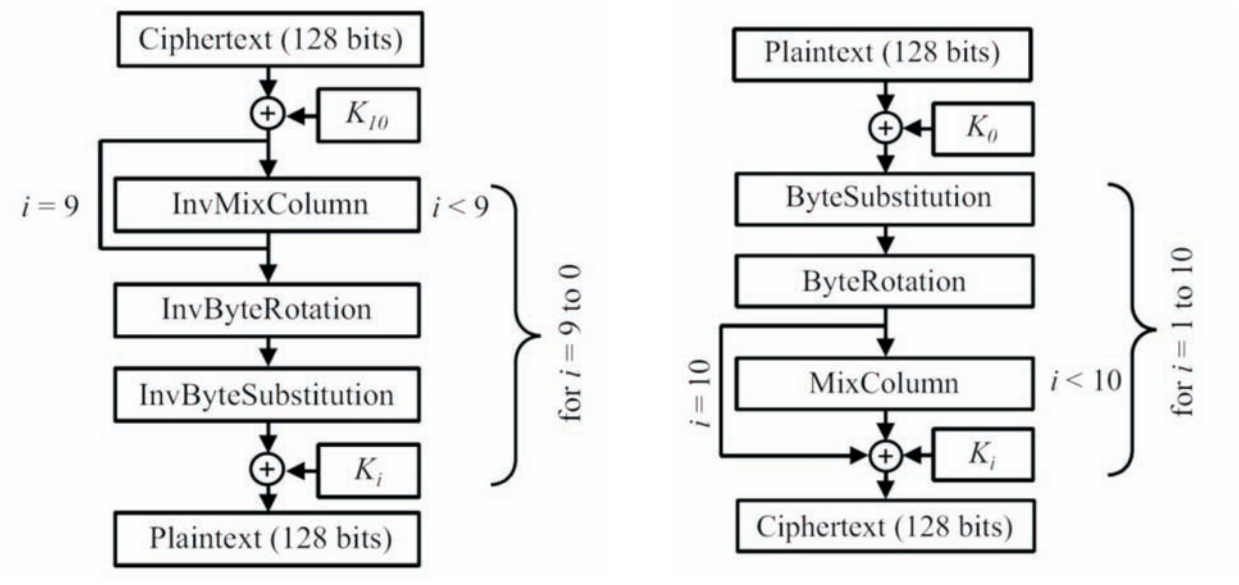
در پیاده‌سازی تابع xTime در سخت‌افزار می‌توان بایت ورودی را یک بیت، شیفت به سمت چپ داد و درصورتی‌که بیت خارج شونده، یک باشد حاصل شیفت را با بایت $0x1b$ جمع انحصاری نمود. به‌همین صورت می‌توان با استفاده از تابع خطی xTime و عملیات منطقی، حاصل‌ضرب ماتریس‌ها را به‌دست آورد؛ برای مثال اگر ستونی از ماتریس ورودی را a بنامیم، حاصل ضرب ماتریس جایگشت رمزگذار در ستون a به‌صورت زیر به‌دست می‌آید. در رابطه‌ی (۳) نتایج به‌دست آمده نشان داده شده است.

$$\begin{aligned} b &= b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0 \\ b.x &= b_7x^8 + b_6x^7 + b_5x^6 + b_4x^5 + b_3x^4 + b_2x^3 + b_1x^2 + b_0x \mod m(x) \\ &= b_6x^7 + b_5x^6 + b_4x^5 + (b_3 \oplus b_7)x^4 + (b_2 \oplus b_7)x^3 + b_1x^2 + (b_0 \oplus b_7)x + b_7 \end{aligned} \qquad (۲)$$

$$\begin{bmatrix} b[0] \\ b[1] \\ b[2] \\ b[3] \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \times \begin{bmatrix} a[0] \\ a[1] \\ a[2] \\ a[3] \end{bmatrix} \Rightarrow \begin{aligned} b[0] &= 02 \times a[0] \oplus 03 \bullet a[1] \oplus 01 \bullet a[2] \oplus 01 \bullet a[3] \\ b[0] &= xTime(a[0]) \oplus xTime(a[1]) \oplus a[1] \oplus a[2] \oplus a[3] \\ b[0] &= xTime(a[0] \oplus a[1]) \oplus a[1] \oplus a[2] \oplus a[3] \\ v &= a[0] \oplus a[1]; v = xTime(v); b[0] = a[1] \oplus a[2] \oplus a[3] \oplus v \end{aligned} \qquad (۳)$$

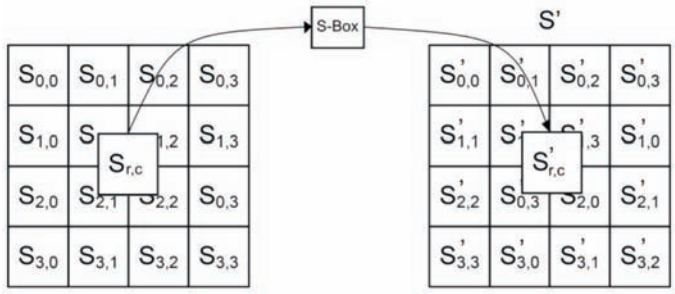
و به‌طور مشابه:

$$\begin{aligned} v &= a[1] \oplus a[2]; v = xTime(v); b[1] = a[0] \oplus a[2] \oplus a[3] \oplus v \\ v &= a[2] \oplus a[3]; v = xTime(v); b[2] = a[0] \oplus a[1] \oplus a[3] \oplus v \\ v &= a[3] \oplus a[0]; v = xTime(v); b[3] = a[0] \oplus a[1] \oplus a[2] \oplus v \end{aligned}$$

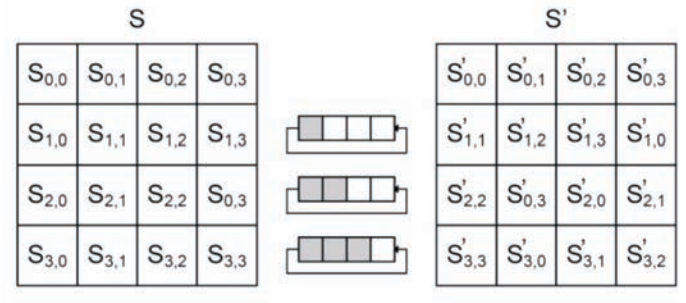


(الف) (ب)

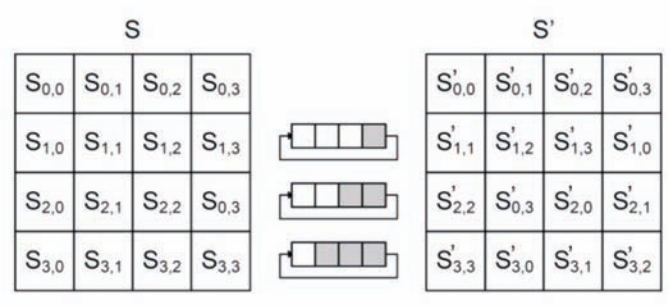
شکل (۱) : نمایش بلوکی الگوریتم رایندال (الف) رمزگذاری ، (ب) رمزگشایی



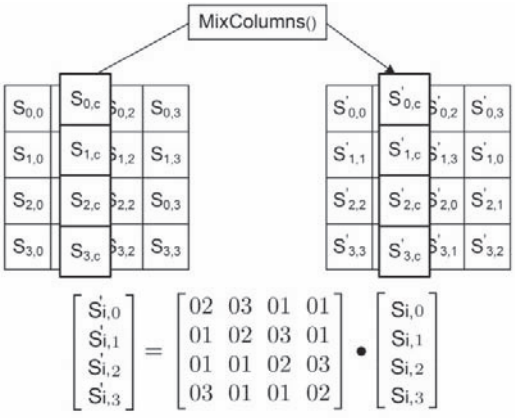
شکل (۲) : عملکرد ByteSubstitution



شکل(۳) : نحوه عملکرد واحد ByteRotation



شکل(۴) : نحوه عملکرد واحد InvByteRotation



شکل (۵) : عملکرد MixColumns

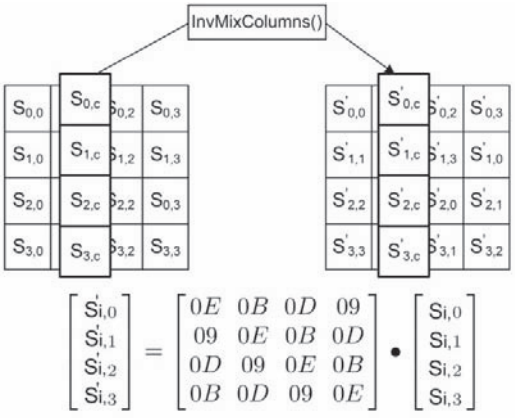
۲-۴- ترکیب زیرکلید

هر دور الگوریتم رایندل از زیرکلید مخصوص به خود استفاده می‌کند که توسط واحد تولید زیرکلیدها، تولید شده است. در این عملیات زیرکلید هر دور با قطعه‌ی ورودی در آن دور، XOR می‌شود.

۲-۵- اصلاح در ساختار رمزگشا

در طرح‌های ارائه شده برای پیاده‌سازی الگوریتم رایندال، عموماً تنها ساختار رمزگذار این الگوریتم، ارائه می‌شود. یک علت برای این موضوع آن است که در برخی از نحوه‌های به‌کارگیری الگوریتم‌های قطعه‌ای[1]، مثل CFB[2]، OFB[3] و CM[4]، تنها ساختار رمزگذار مورد استفاده قرار می‌گیرد؛ اما علت دیگر این موضوع آن است که امکان ارائه ساختار دیگری از رمزگشای رایندال که کاملاً مشابه ساختار رمزگذار است وجود دارد[۱۰]. به‌دلیل بایتی بودن عملیات در واحدهای InvByteSubstitution و InvByteRotation، همان‌طور که در شکل(۷) مشاهده می‌شود، ترتیب دو واحد مذکور در ساختار رمزگشای رایندال می‌تواند جابه‌جا شود. با توجه به خاصیت تکرری ساختار رمزگشا، انتقال واحد InvMixColumns از ابتدای حلقه به انتهای حلقه در شکل مذکور بلامانع است (دقّت شود همانطور که در شکل (۱)-ب قابل مشاهده است در دور اوّل اجرای حلقه، قطعه‌ی ورودی از واحد InvMixColumns عبور نمی‌کند). با مقایسه‌ی ساختار جدید رمزگشا با ساختار رمزگذار رایندال در شکل (۱)-الف، ملاحظه می‌شود که تنها تفاوت باقیمانده، ترتیب متفاوت اعمال InvMixColumns و ترکیب زیر کلید است. اما به دلیل خاصیت خطی داشتن

[1] Modes of operation
[2] Cipher FeedBack
[3] Output FeedBack
[4] Counter Mode



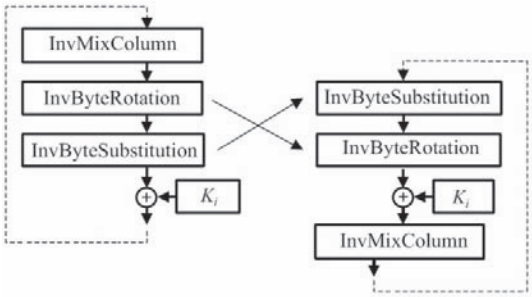
شکل (۶) : عملکرد InvMixColumns

توأم عملیات جایگشت و عملیات XOR، جابه‌جایی دو عملیات مذکور امکان‌پذیر خواهد بود:

بدین ترتیب ساختار نهایی رمزگشای جدید، مطابق شکل (۸)، حاصل می‌شود که از ساختاری کاملاً مشابه ساختار رمزگذار رایندال برخوردار است و در آن زیر کلید به‌کار رفته، صورت جابه‌جا شده‌ای از زیرکلید اصلی است[۱۱].

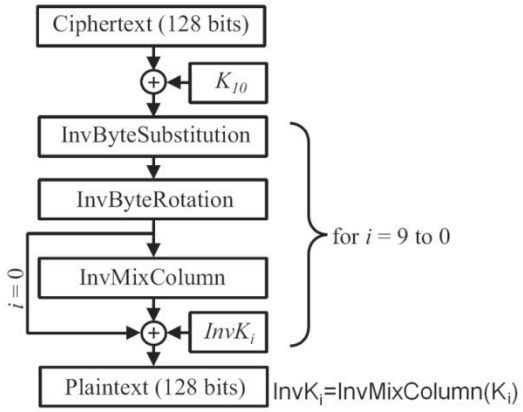
(۴)

$$InvMixColumn(d \oplus K) = InvMixColumn(d) \oplus InvMixColumn(K)$$



شکل (۷) : نحوه‌ی جابه‌جایی دو واحد در قسمت رمزگشای الگوریتم رایندال[۱۱]

البته به‌دلیل پیچیدگی عملیات پردازش در واحد InvMixColumns نسبت به واحد MixColumns (به شکل‌های ۵ و ۶ رجوع گردد)، کماکان نرخ گذردهی رمزگشا از نرخ گذردهی رمزگذار پایین‌تر خواهد بود[۲۶،۲۴،۱۱]. با توجه به بحث فوق، برای پیاده‌سازی رمزگشای رایندال می‌توان عیناً از معماری پیشنهادی برای رمزگذار استفاده کرد. بنابراین، در این مقاله بحث خود را به ارائه‌ی معماری مناسب برای رمزگذار الگوریتم رایندال، محدود نموده و از این پس، از واحد رمزگذار با عنوان کلی‌تر "واحد رمزکننده" یاد می‌کنیم.



شکل (۸): نمایش ساختار تغییر یافته رمزگشای الگوریتم رایندال[۱۱]

۳- معرفی مدلهای مختلف معماری در پیاده‌سازی الگوریتم رایندال

با توجّه به کاربرد گسترده‌ی الگوریتم‌های رمزنگاری در سیستم‌های امنیتی و به‌دنبال معرفی الگوریتم رایندال به‌عنوان الگوریتم استاندارد، پیاده‌سازی‌های سخت‌افزاری مختلفی از آن برای کاربردهای متنوع انجام شده است. طراحان با در نظر گرفتن نوع کاربردها و محدودیت‌های آن‌ها سعی فراوانی در یافتن معماری مناسبی برای پیاده‌سازی الگوریتم مذکور کرده‌اند. برای مثال در برخی از کاربردها همانند کارت‌های هوشمند، محدودیت حجم کد، مسئله‌ی مهمی در پیاده‌سازی الگوریتم محسوب می‌شود. از این‌رو در پیاده‌سازی الگوریتم سعی می‌شود تا حد امکان فضای مورد نیاز برای دستورالعمل‌ها کاهش یابد که البته این مسئله باعث کاهش نرخ رمزکننده می‌گردد. در مقابل در برخی از کاربردها سرعت پردازش از حجم سخت‌افزار مصرفی مهم‌تر است. در این گونه سیستم‌ها مسئله‌ی زمان پاسخ بسیار مهم می باشد و طراحی باید به‌گونه‌ای باشد که ضمن صرفه‌جویی در مصرف سخت‌افزار، به محدودیتهای زمانی مورد نیاز نیز پاسخ مناسب داده شود. از این‌رو پیاده‌سازی‌های مختلفی به‌منظور افزایش کارایی الگوریتم رایندال ارائه شده است که در ادامه، چندین معماری عمده که در سخت‌افزارهای برنامه‌پذیر FPGA مورد استفاده قرار گرفته‌اند بررسی می‌شوند.

۳-۱- پیاده‌سازی یک دور الگوریتم و استفاده‌ی مکرر از آن در دورهای مختلف به‌صورت فیدبک

در این روش با توجه به مشابه‌بودن عملیاتی که در هر دور انجام می‌شود (مطابق شکل ۱)، تنها یک دور الگوریتم به‌صورت کامل در یک واحد سخت‌افزاری پیاده‌سازی می‌شود. در این پیاده‌سازی نتیجه‌ی تولید شده در هر دور با فیدبک، به ورودی واحد پیاده‌سازی شده ارسال می‌گردد تا پردازش دور بعدی انجام گیرد[۷]. درصورتی‌که در پیاده‌سازی یک دور الگوریتم از معماری مناسب و بهینه‌ای استفاده شود، می‌توان به نرخ‌های مناسبی دست یافت. در پیاده‌سازی‌هایی که با این روش انجام گرفته است، معمولاً سعی می‌شود تا عملیات یک دور الگوریتم در یک پالس ساعت انجام شود. برای تحقق این هدف، در طراحی واحدهای جانشینی رمزگذار و یا رمزگشا، با تکرار ۱۶ واحد جعبه‌ی جانشینی مشابه، عملیات Bytesubstitution به‌صورت موازی انجام می‌گیرد. در این‌صورت نتیجه‌ی عملیات رمزگذاری پس از ۱۰ پالس ساعت، تولید خواهد شد[۶،۸،۹،۱۲،۱۳].

۳-۲- استفاده از خطلوله در پیاده‌سازی الگوریتم

یک ایده‌ی مناسب برای دستیابی به سرعت‌های بالا استفاده از معماری خطلوله در پیاده‌سازی الگوریتم است. در این روش واحد سخت‌افزاری پیاده‌سازی شده برای هر دور را به تعداد دورها تکرار می‌کنند تا خروجی تولید شده در هر دور را به دور بعد انتقال دهند. در این‌گونه پیاده‌سازی، حجم سخت‌افزار مصرفی به‌طور چشم‌گیری افزایش پیدا می‌کند[۱۵]. برای مثال در این روش جهت انجام رمزگذاری و یا رمزگشایی یک قطعه ۱۶۰ واحد جعبه‌ی جانشینی مورد نیاز است. در این روش داده‌ها یکی پس از دیگری با هر پالس ساعت وارد سیستم می‌شوند و پس از گذشت ۱۰ پالس ساعت از زمان وارد شدن اولین ورودی، نتایج، پشت سرهم، پالس به پالس تولید خواهند شد. [۱۶،۱۷،۱۸،۱۹].

۳-۳- استفاده از خط لوله در پیاده‌سازی یک دور الگوریتم و استفاده‌ی مکرر از آن در دورهای مختلف به‌صورت فیدبک

این روش مشابه روش اول می‌باشد منتها در پیاده‌سازی یک دور الگوریتم، از خطلوله استفاده می‌گردد. در این معماری عملیات هر دور به چند قسمت تقسیم بندی می‌شود به طوری که حجم عملیات قسمت‌ها با یکدیگر مشابه باشند. در این صورت هر قسمت را به‌عنوان

یک مرحله خط‌لوله در نظر گرفته و رمزکننده را پیاده‌سازی می‌نمایند. برای مثال با قرار دادن مراحل ByteSubstitution ، MixColumn و ترکیب کلید در سه مرحله خط‌لوله عملیات‌های آن‌ها را به‌صورت پشت سرهم در سه پالس انجام می‌دهند. با این روش می‌توان فرکانس کاری رمزکننده را با توجه به کاهش عملیات در هر پالس افزایش داد. البته در این روش، تا پردازش سه قطعه‌ی وارد شده‌ی متوالی(طی ۱۰ دور) کامل نشود، نمی‌توان ورودی بعدی را وارد نمود[۶،۲۴].

۴- ارائه‌ی معماری ضربه‌ای[1] پیشنهادی

در این بخش معماری پیشنهادی که مبتنی بر معماری ضربه‌ای است، ارائه خواهد شد. در طراحی و پیاده‌سازی پردازنده‌های محاسباتی می‌توان از معماری ضربه‌ای استفاده نمود[۲۳]. بر اساس این معماری می‌توان از هر دو لبه پالس ساعت برای پردازش بهره برد. در طراحی معماری پیشنهادی مسائل مختلفی از جمله هزینه‌ی سخت افزار و سرعت پردازش مورد توجه قرار گرفته است که در ادامه، عوامل مختلف تأثیرگذار در انتخاب معماری پیشنهادی ارائه خواهد شد.

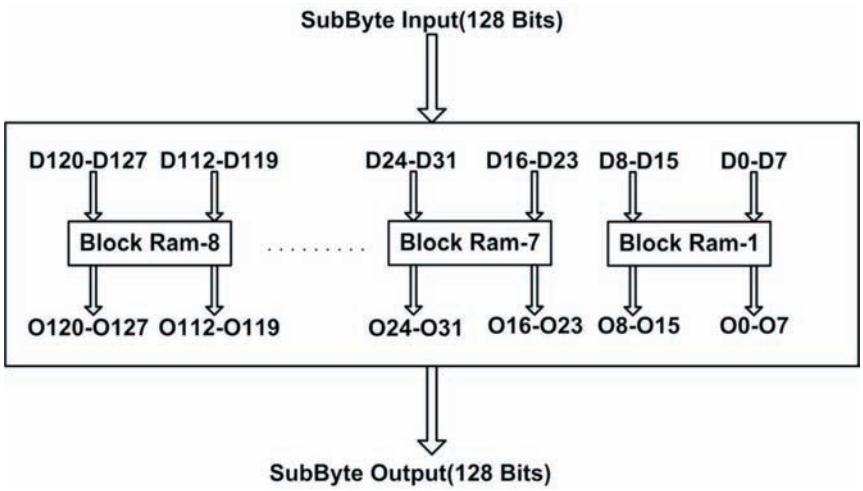
۴-۱- انتخاب معماری کلی

راه‌کار کلی که برای طراحی رمزکننده، مورد توجه قرار گرفته است، مشتمل بر موازی‌سازی واحدهای مختلف رمزکننده و استفاده‌ی اشتراکی از منابع پرهزینه است. تحقق این راه‌کار، ضمن اینکه صرفه‌جویی در سخت‌افزار مصرفی را موجب می‌شود گذردهی مناسبی را نیز به بار می‌آورد. همانطور که در (شکل ۱) نمایش داده شده است الگوریتم رایندال از واحدهای مختلفی تشکیل شده است که با بررسی وظیفه‌ی هر واحد می‌توان به تخمینی از حجم سخت‌افزار مصرفی و سرعت پردازش آن واحد رسید و نقاط گلوگاه الگوریتم را مشخص نمود. برای مثال در واحد ByteRotation با توجه به طول قطعه، جایگشت‌هایی روی ماتریس داده‌ها انجام می‌گیرد. در این عملیات نیازی به استفاده از عناصر منطقی نیست و فقط می‌توان با نحوه‌ی سیم‌بندی مناسب از ورودی به خروجی، عملیات ByteRotation را انجام داد. واحد MixColumn نیازمند عملیات ضرب در میدان $GF(2^8)$ است که معمولاً در پیاده‌سازی‌های انجام گرفته از توابع xTime و XOR استفاده می‌شود. در واحد ByteSubstitution که عملیات غیرخطی بر روی داده‌ها انجام می‌گردد، برای کاهش تأخیر بجای استفاده از مدارهای منطقی، از جدول‌های[2] از پیش آماده شده استفاده می‌شود. هر بایت قطعه‌ی ورودی نقش آدرس را برای جدول ایفا نموده و محتوای متناظر با آدرس ورودی، نتیجه‌ی عملیات غیرخطی بر روی آن بایت است. اگر این عملیات به‌صورت ترتیبی با استفاده از یک جدول انجام شود نیاز به صرف ۱۶ پالس ساعت در هر دور الگوریتم است که این به مفهوم صرف هزینه‌ی زمانی بالا و کاهش سرعت رمزکننده است.

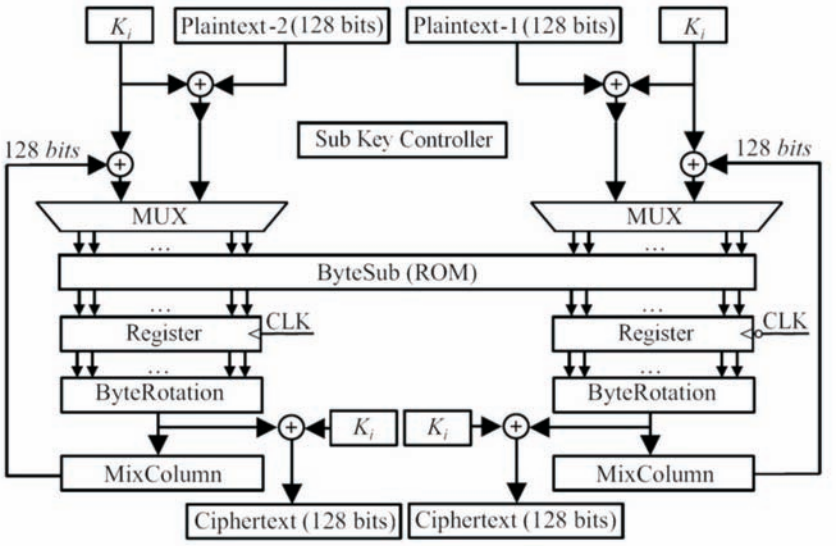
برای افزایش سرعت رمزکننده می‌توان ۱۶ واحد جعبه‌ی جانشینی را به‌صورت جدول‌های از پیش آماده شده تکرار نمود تا ۱۶ بایت ورودی در یک لحظه وارد جدول‌ها شود و نتیجه‌ی‌نهایی به صورت موازی و همزمان تولید گردند. در این‌صورت امکان پردازش هر دور در یک پالس ساعت فراهم می‌شود. در پیاده‌سازی واحد جانشینی برروی FPGA می‌توان از بلوک‌های حافظه‌ی دو درگاهه[3] استفاده نمود. در این بلوک‌ها دسترسی همزمان به مقادیر واحد جانشینی از طریق دو درگاه امکان پذیر است. این بلوک‌های حافظه از نوع حافظه‌های پایدارِ سریع[4] بوده و تقریباً در تمامی FPGA ها قابل دسترس هستند. در این‌صورت تعداد بلوکهای مورد نیاز برای نگهداری جعبه‌های جانشینی از ۱۶ بلوک به ۸ بلوک کاهش می‌یابد. شکل ۹ ساختار واحد جانشینی مورد استفاده در معماری پیشنهادی را نشان می‌دهد. بلوک‌های حافظه در FPGA ها محدود بوده و در استفاده از آن‌ها باید بیشترین دقت و صرفه‌جویی را به‌کار بست. برای مثال در تراشه‌ی Spartan IIE2S200 که شامل حدود ۲۰۰۰ قطعه سخت‌افزار[5] است، تعداد بلوک‌های حافظه‌ی آن، ۱۴ واحد است. بنابراین با توجه به تکرار جعبه‌های جانشینی برای کاهش زمان پاسخ، حجم سخت‌افزار مصرفی به‌شدت افزایش پیدا می‌کند. با این توصیف می‌توان گفت که پیاده‌سازی واحد ByteSubstitution پرهزینه خواهد بود و نیاز به اتخاذ روشی مناسب است که هم از نظر کارآیی و هم از نظر حجم سخت‌افزار مصرفی مناسب باشد.

[1] Multibeat

[2] Look-UP Tables
[3] Dual Port
[4] Fast Static RAM
[5] Slice



شکل(۹) ساختار جعبه‌های جانشینی



شکل(۱۰): ساختار پیشنهادی برای رمزگذار الگوریتم رایندال مبتنی بر معماری ضربه‌ای

۴-۲- پیشنهاد رمزکننده‌ای مبتنی بر معماری ضربه‌ای

از آنجا که حجم عمده‌ی سخت‌افزار در پیاده‌سازی مربوط به واحد جعبه‌ی جانشینی مصرف می‌شود، بنابراین برای افزایش کارآیی رمزکننده و صرفه‌جویی در سخت‌افزار مصرفی، در معماری پیشنهادی دو واحد رمز کننده به‌صورت مشترک از یک جعبه‌ی جانشینی استفاده می‌کنند. طرح پیشنهادی که مبتنی بر معماری ضربه‌ای است، از هر دو لبه‌ی پالس ساعت برای پردازش استفاده می‌کند. برای استفاده‌ی مناسب از سخت‌افزار و برخورداری از سرعت بیشتر، از همپوشانی در پردازش نیز بهره گرفته می‌شود، که در ادامه، جزییات آن شرح داده خواهد شد.

همان‌طور که در شکل (۱۰) نشان داده شده است در معماری پیشنهادی دو قطعه‌ی ورودی به‌طور همزمان وارد رمزکننده می شود و به موازات یکدیگر عملیات رمزگذاری بر روی آنها انجام می‌گردد. نحوه‌ی پردازش به این صورت است که قطعه‌ی ورودی اول در سطح high پالس ساعت با زیرکلید مربوط به دور، XOR شده و به‌عنوان آدرس، وارد جداول از پیش آماده شده می‌شوند.

نتایج به‌دست آمده از این جدول در لبه‌ی پایین رونده پالس ساعت در یک ثبات ۱۲۸ بیتی ذخیره می‌شود، تا ادامه‌ی پردازش در ماژول‌های ByteRotation و MixColumn در سطح Low پالس ساعت انجام شود. در لبه‌ی پایین‌رونده‌ی پالس ساعت، داده‌های قطعه‌ی دوم با زیرکلید مربوطه XOR شده و سپس وارد جداول از پیش

آماده شده می‌شوند (واحد ByteSubstitution بین ورودی‌های قطعات اول و دوم براساس سطح پالس ساعت تمایز قائل می‌شود و در هر سطح فقط یکی از رمزکننده‌ها به جداول مربوطه دسترسی دارند).

۵- پیاده‌سازی الگوریتم رایندال مبتنی بر روش پیشنهادی با استفاده از مدل فیدبک

در این بخش رمزکننده‌ای ارائه می‌شود که در طراحی آن از روش پیشنهادی و مدل فیدبک استفاده شده است. در این رمزکننده یک دور الگوریتم منطبق بر معماری ضربه‌ای که در شکل (۱۰) نمایش داده شده است، پیاده‌سازی می‌گردد و سپس با فیدبک گرفتن از خروجی به ورودی، عملیات رمزگذاری تکمیل خواهد شد. از آنجا که در این روش پیاده‌سازی دقیقاً از معماری ضربه‌ای معرفی شده در شکل (۱۰) استفاده شده است، در اینجا فقط به نحوه تولید زیر کلیدها و استفاده از آنها می‌پردازیم.

از ویژگی‌های الگوریتم رایندال، امکان تولید زیرکلیدها در حین عملیات رمزگذاری است[۳]. در صورتی که از این روش برای تولید زیرکلیدها استفاده گردد باید در هر دور برای تولید زیرکلید دور بعد، از واحد جعبه جانشینی استفاده شود. این مسئله باعث می‌شود که یا زمان تأخیر برای پردازش در یک دور افزایش یابد(دسترسی دوبار به جعبه‌های جانشینی در یک دور به صورت ترتیبی) و یا اینکه با صرف هزینه سخت‌افزار اضافی از جعبه جانشینی دیگری برای تولید زیرکلیدها استفاده شود. هر دو روش تأثیر بسیار منفی بر روی کارایی رمزکننده‌ی پیشنهادی خواهند داشت. از طرفی اگر به کاربردهای واقعی دقت شود ملاحظه می‌گردد که معمولا نرخ رمزگذاری به مراتب بیش از نرخ تعویض کلیدها است. برای مثال در یک محیط پر تراکنش مانند شبکه اینترنت، طول متوسط بسته‌های IP در شبکه اینترنت حدود ۵۰۰ بایت است که برای رمزگذاری هر بسته از کلید یکسان بهره‌برداری می‌شود. در بدترین حالت در این شبکه با بسته‌های کوچک ACK سروکارداریم که به ازای یک بار تعویض کلید همچنان ۳ قطعه برای رمزگذاری وجود دارد.

از آنجا که در این روش قصد داریم رمزگذاری‌ای با سرعت و هزینه‌ی مناسب طراحی کنیم، از تولید زیرکلیدها در حین پردازش صرف‌نظر خواهیم نمود و تولید زیرکلیدها را با استفاده از واحدهای پیاده‌سازی شده‌ی موجود و در ابتدای عملیات رمزنگاری انجام خواهیم داد. در غالب طرح‌های ارائه شده نیز راه‌حلی در ارتباط با نحوه‌ی تولید زیرکلیدها ارائه نشده است و فرض بر آماده بودن زیرکلیدها قبل از شروع عملیات رمزنگاری است.

واحد SubKeyController در شکل (۱۰) به این صورت طراحی می‌شود که با دریافت فرمان تعویض کلید و ۱۲۸ بیت کلید اصلی، تولید زیرکلیدها و ذخیره‌ی آنها را با استفاده از منابع موجود انجام خواهد داد. بنابراین عملیات تولید زیرکلیدها، فقط در زمان تعویض کلید انجام می‌شود و در طول این مدت (۱۰ پالس ساعت) رمزگذار قادر به ارائه‌ی سرویس رمزگذاری نیست.

همان‌طور که در شکل(۱) نشان داده شده است در الگوریتم رایندال در ابتدای هر دور، نیاز به زیرکلید مربوط به همان دور است؛ بنابراین در ذخیره‌ی زیرکلیدها باید به‌گونه‌ای عمل نمود که دسترسی به ۱۶ بایت کلید در هر دور تأثیر نامطلوبی بر روی سرعت رمزگذار نداشته باشد. یک راه حل ساده برای ذخیره‌ی زیرکلیدها استفاده از فلیپ‌فلاپ‌ها است که در این‌صورت نیاز به ۲۶۰۰ فلیپ‌فلاپ‌ها برای ذخیره‌ی بیت‌های زیرکلید داریم. از آنجا که این عناصر به‌صورت گسترده در پیاده‌سازی مدارهای سخت‌افزاری مورد استفاده قرار می‌گیرد و تعداد آن در FPGA ها محدود است، در رمزکننده‌ی فیدبکی پیشنهادی برای ذخیره‌ی زیرکلیدها از آنها استفاده نمی‌شود. از طرفی اگر از یک حافظه‌ی بلوکی[1] برای ذخیره‌ی زیرکلیدها استفاده شود، در هر دور برای دسترسی به زیرکلیدها باید ۱۶ پالس ساعت منتظر بمانیم. برای حفظ سرعت و همچنین صرفه‌جویی در حجم سخت‌افزار از ۱۶ قطعه حافظه با ظرفیت ۱۱ بایت استفاده می‌شود. در پیاده‌سازی این ساختار از حافظه‌های توزیع شده[2] که به‌صورت گسترده و با ابعاد قابل تنظیم در سطح FPGA وجود دارد، استفاده خواهیم نمود.

در این ساختار برای ذخیره‌ی زیرکلید دور صفر، تمامی۱۶ بایت زیرکلید این دور را در بایت‌های با موقعیت صفر، ۱۶ قطعه حافظه قرار می‌دهیم. به‌همین‌صورت برای ذخیره‌ی زیرکلیدهای دور i ام، از آدرس i ام هر بلوک استفاده می‌شود. به‌عبارت دیگر ساختار حافظه زیرکلیدها را به‌طور ستونی مقداردهی می‌نماییم. بنابراین با توجه به تعداد دورهای الگوریتم، نیاز به قطعه‌های حافظه با عمق ۱۱ بایت و با توجه به تعداد بایت‌های زیرکلید هر دور از ۱۶ قطعه حافظه استفاده می‌شود. بنابراین K_{i-j} در (شکل ۱۱) نشان

[1] Block Ram
[2] Distributed Memory

دهنده‌ی زیرکلید مربوط به i امین دور و j امین بایت از زیرکلید مربوط به آن دور است. با توجه به حجم حافظه‌های توزیع شده در FPGA و قابلیت برنامه‌ریزی آنها با ابعاد دلخواه، استفاده از این ساختار حجم ناچیزی از سخت‌افزار را به‌خود اختصاص می‌دهد. در این ساختار طی یک دسترسی، می‌توان زیرکلیدهای هر دور را به‌دست آورد. همان‌طور که در (شکل ۱۱) نشان داده شده است، شماره‌ی دور به‌عنوان آدرس بلوک‌های ۱۱ بایتی، به حافظه‌ی زیرکلیدها اعمال می‌شود.

شکل(۱۱): ساختار ذخیره زیرکلیدها

جدول (۱) : نتایج پیاده‌سازی‌های مختلف الگوریتم رایندال با استفاده از روش فیدبک

	نوع سخت‌افزار	فرکانس ساعت(MHz)	نرخ خروجی(Gbps)	Block RAM	مقدار Slice مصرفی	Mbps/Slice
رمزکننده فیدبکی پیشنهادی	XC2S200e	120	3.072	8	900	1.596
Francis C.[12]	XC2V400	40	0.465	-	2688	0.172
Pramstaller [14]	XCV1000E	101	0.215	-	1125	0.191
Attaie[24]	XC2V2000	195	2.5	10	654	1.292
Attaie[24]	XC2V2000	107	1.35	-	1844	0.732
Li[6]	Virtex II	135.7	1.57	-	2052	0.765

این طرح بر روی تراشه‌ی XC2S200e-6 از خانواده‌ی Xilinx در نرم‌افزار ISE 6.1 پیاده‌سازی شده است. شبیه‌سازی مدار پیشنهادی در نرم‌افزار ModelSim و همچنین سنتز آن در نرم‌افزار Synplify انجام گرفته است. تراشه‌ی XC2S200e-6 در رده‌ی تراشه‌های ارزان قیمت Xilinx بوده و دارای ۱۴ واحد حافظه‌ی بلوکی ، 56 Kbit حافظه توزیع شده ، ۲۳۵۲ واحد Slice و درجه‌ی سرعت[1] آن ۶ است.

در جدول(۱) مشخصات طرح پیشنهادی با مشخصات سایر طرح‌ها مقایسه شده است. در مقایسه‌ی طرح‌های مختلف بر روی FPGA حجم سخت‌افزار و نرخ‌گذردهی، دو معیار اصلی برای مقایسه هستند که معمولاً به ترتیب برحسب میزان مصرف Slice و Mbps مورد مقایسه قرار می‌گیرند. بنابراین درمجموع، معیار Mbps/Slice می‌تواند پارامتر مناسبی برای مقایسه‌ی طرح‌های مختلف باشد؛ امّا از آنجا که برخی طرح‌ها از حافظه‌های بلوکی در معماری خود استفاده کرده‌اند و در گزارش‌های سنتز، مقدار Slice مصرفی و تعداد حافظه‌ی بلوکی مصرفی عموماً به‌طور مجزا داده می‌شود، باید در مقایسه‌ی طرح‌ها، مقدار حافظه‌ی بلوکی مصرف شده را نیز مورد توجه قرار داد. در همین راستا در [۱۹] با معادل گرفتن هر واحد حافظه‌ی بلوکی با ۱۲۸ Slice امکان مقایسه‌ی طرح‌های مختلف را فراهم نموده است. بر همین اساس، برای محاسبه Mbps/Slice (جدول ۱)، از این ملاک برای مقایسه‌ی طرح‌های مختلف استفاده شده است (در منابع [۲۲] و [۲۸] نیز عیناً همین

[1] Speed Grade

ملاک مورد توجه قرار گرفته است). همان‌طور که ملاحظه می‌شود، طرح پیشنهادی در مقایسه با طرح‌هایی که از روش فیدبک استفاده کرده‌اند کارآیی مناسبی دارد. درصورتی‌که رمزکننده‌ی مبتنی بر فیدبک بر روی ASIC پیاده‌سازی شود، به‌سرعت بالاتری نیز می‌توان دست یافت. در یک پیاده‌سازی نمونه بر روی ASIC [۸]، فرکانس کاری رمزگذار 224 MHz به‌دست آمده است که در مقایسه با پیاده‌سازی‌های انجام گرفته بر روی FGPA، همان‌طور که قابل انتظار است، افزایش قابل توجهی را در فرکانس ساعت نشان می‌دهد.

۶- پیاده‌سازی الگوریتم رایندال مبتنی بر روش پیشنهادی و خط لوله

در این بخش، رمزکننده‌ای که در طراحی آن از معماری خط لوله و معماری ضربه‌ای به‌طور توأم استفاده شده است، ارائه می‌شود. در معماری خط لوله، کل عملیات به قسمت-های مختلفی تقسیم‌بندی می‌شود که به هر قسمت، یک مرحله‌ی[1] خط لوله می‌گویند. در طراحی مراحل خط لوله سعی می‌شود تا حجم عملیات مراحل مختلف، یکسان باشد تا خط لوله‌ی متقارنی در سیستم ایجاد گردد. معمولاً در این معماری در هر پالس ساعت، نتیجه‌ی تولید شده از دور قبل را در ثبات‌های میانی هر مرحله قرار می‌دهند تا در پردازش دور بعد از آن‌ها استفاده شود. در معماری خط لوله، اطلاعات ورودی بدون وقفه وارد رمزکننده می‌گردند و بنابراین گذردهی بسیار بالایی حاصل می‌شود که هزینه‌ی آن حجم زیاد سخت‌افزار مصرفی در آن است.

استفاده از ثبات‌های میانی که برای نگهداری نتایج هر مرحله به‌کار گرفته می‌شوند، باعث افزایش تأخیر در اجرای هر مرحله از خط لوله است و باعث کاهش فرکانس پالس ساعت نسبت به حالت بدون خط لوله می‌گردد، امّا در معماری ضربه‌ای پیشنهادی، نیازی به قرار دادن چنین ثبات‌هایی در بین مراحل خط لوله نیست. اگر به ساختار شکل (۱۰) دقت شود ملاحظه می‌شود که در معماری ضربه‌ای برای ذخیره‌ی نتیجه‌ی تولید شده در واحد ByteSubsitution از ثبات‌هایی استفاده شده است که محتوای آن‌ها برای یک پالس ساعت، ثابت است. بنابراین نیازی به استفاده از ثبات‌های اضافی برای ذخیره‌ی نتیجه‌ی نهایی یک دور نیست. این مسئله باعث می‌شود تا فرکانس کاری در روش مبتنی بر خط لوله نسبت به روش بدون استفاده از آن، کاهش پیدا نکند، از طرفی معماری پیشنهادی که در بخش(۴) ارائه شد، به‌گونه‌ای طراحی شده است که امکان استفاده از آن در تمامی دورها وجود داشته باشد؛ درصورتی که در مدل خط لوله با پیاده‌سازی سخت‌افزارهای مجزا برای هر دور، می‌توان سخت‌افزار هر دور را با توجه به نیازمندیهای همان دور پیاده‌سازی نمود. از این رو می‌توان معماری پیشنهادی را برای استفاده در خط لوله، بهینه نمود. برای مثال می‌توان انتخاب، کننده‌ای[2] را که در ابتدای هر دور وجود دارد، حذف نمود و یا در دور آخر نیازی به پیاده‌سازی واحد MixColumn نیست. شکل(۱۲) ساختار بهینه، شده‌ای از معماری پیشنهادی را در مدل خط لوله نشان می‌دهد.

برای تولید زیرکلیدها در روش خط لوله، بدین صورت عمل می‌شود که کلید اصلی به‌همراه دستور تعویض کلید به دور اول وارد می‌گردد تا زیرکلید مربوط به این دور تولید شود. پس از آن در پالس بعدی نتیجه‌ی تولید شده در دور اول به‌همراه دستور تولید زیرکلید، به دور بعد ارسال می‌شود و قطعه‌ی ورودی برای پردازش عملیات رمزگذاری وارد خط لوله می‌گردد. بنابراین در هنگام تعویض کلید رمزکننده فقط به یک پالس تأخیر در ورود اطلاعات جدید نیاز دارد. از این‌رو در کاربردهایی که نرخ تعویض کلید بالایی دارند، تعویض کلید، تأخیر قابل ملاحظه‌ای ایجاد نمی‌کند (بر خلاف روش مبتنی بر فیدبک که تعویض کلید، تأخیری معادل با ۱۰ پالس ساعت را در روند رمزنگاری داده‌ها ایجاد می‌نمود).

برای پیاده‌سازی الگوریتم رایندال به‌صورت خط لوله و براساس معماری پیشنهادی، دو طرح ارائه شده است. در طرح اول رمزکننده‌ای طراحی می‌شود که طول خط لوله آن ۵ می‌باشد و یک قطعه را برای عملیات رمزگذاری دریافت می‌نماید.

در این روش، ضمن بهره‌گرفتن از معماری خط لوله، از حجیم شدن بی‌رویه‌ی سخت‌افزار پرهیز شده است. در روش دوّم رمزکننده‌ای طراحی شده است که طول خط لوله آن ۱۰ می باشد و قابلیت دریافت دو قطعه را به‌طور همزمان دارد. روش اخیر از نظر سرعت، نسبت به روش اول بهتر است؛ اما هزینه‌ی سخت‌افزاری آن دو برابر روش قبل است. در ادامه، جزییات پیاده‌سازی این رمزکننده‌ها، شرح داده می‌شود.

۶-۱- پیاده‌سازی الگوریتم رایندال با طول خط لوله

در این روش سعی بر آن است تا ضمن استفاده از معماری خط لوله، هزینه‌ی سخت‌افزار مصرفی، به نسبت زیاد نباشد.

[1] Stage

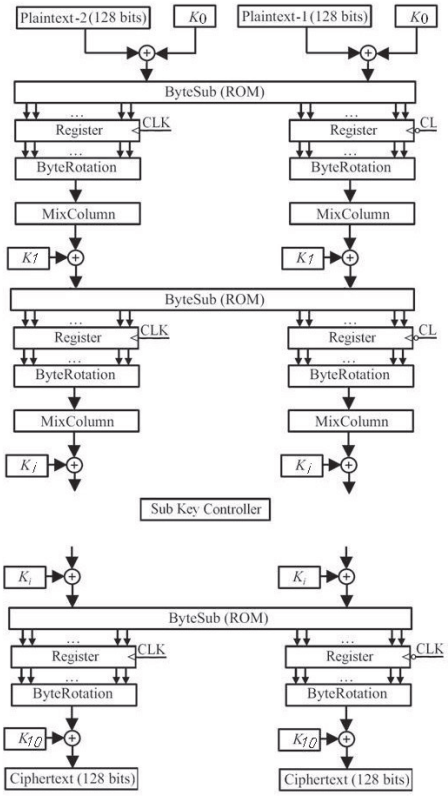
[2] Multiplexer

از این‌رو خط لوله به‌طور کامل پیاده‌سازی نمی‌شود. در مدارهای مبتنی بر خط لوله درصورتی‌که طول خط لوله کاهش یابد، منجر به حجیم شدن عملیات در هر مرحله می‌گردد و یا اینکه باعث تأخیر در پذیرش ورودی‌های جدید می‌شود، که هر دو پیامد برای دستیابی به سرعت‌های بالا نامناسب خواهند بود؛ اما با استفاده از معماری ضربه‌ای می‌توان رمزکننده‌ای پیشنهاد داد که تنها با پیاده‌سازی ۵ مرحله از ۱۰ مرحله‌ی خط لوله، هیچکدام از پیامدهای نامناسب ذکر شده رخ ندهند.

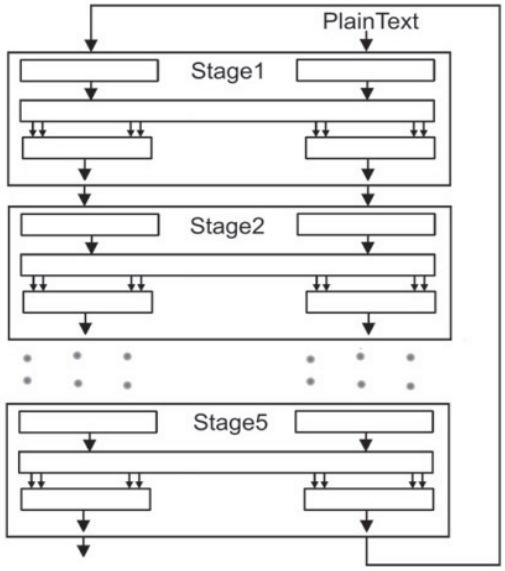
شکل(۱۳) ساختار این طرح را نشان می‌دهد. همان‌طور که در این شکل نشان داده شده است در این طرح رمزگذاری با طول خط لوله‌ی ۵ و منطبق بر معماری ضربه‌ای بهبود یافته، بدین‌صورت پیاده‌سازی شده است که در ابتدا قطعه‌ی ورودی به رمزکننده‌ی سمت راست دور اول وارد می‌شود. این رمزکننده پس از انجام پردازش خود در یک پالس، نتیجه را به رمزکننده‌ی هم‌ستونی خود در دور بعد تحویل می‌دهد.

پس از طی ۵ مرحله، نتیجه‌ی تولید شده در مرحله‌ی پنجم با فیدبکی به ورودی دور اول در رمزکننده‌ی سمت چپ ارسال می‌شود تا ادامه‌ی پردازش بر روی قطعه‌ی مذکور در رمزکننده‌های سمت چپ (واحدهای ۱ تا ۵) دنبال می‌شود. با توجه به این نحوه‌ی استفاده از معماری ضربه‌ای در خط لوله، رمزکننده قادر به دریافت یک قطعه‌ی ورودی در هر پالس ساعت است. در این روش تعداد حافظه‌ی بلوکی مورد استفاده، ۴۰ بلوک است.

برای ذخیره‌ی زیرکلیدها نمی‌توان از طرحی که در بخش(۵)ارائه شد استفاده نمود، چون در روش خط لوله، تمام دورها به‌طور همزمان به زیرکلید مربوط به دور خود نیاز دارند و در آن روش دسترسی همزمان دورهای مختلف به ساختار طراحی شده، پشتیبانی نمی‌شد. با توجّه به این‌که حافظه‌های توزیع شده را می‌توان با ساختارهای قابل تعریف استفاده نمود، به‌طور جایگزین برای این طرح در هر دور، دو بلوک ۱۲۸ بیتی از این حافظه‌ها در نظر گرفته می‌شود. این طرح علاوه بر سنتز بر روی XCV1000-6 بر روی تراشه‌ی XCV1000e-8 نیز سنتز شده است و با توجه به درجه‌ی سرعت ۸ این FPGA فرکانس 149.5 MHz و نرخ گذردهی 19.136 Gbps برای آن ثبت گردید (با افزایش هر درجه، سرعت بین ۵ تا ۲۰ درصد فرکانس پالس ساعت می‌تواند افزایش پیدا کند[۲۲،۲۴]). در(جدول ۲) اطلاعات بیشتری در مورد این رمزکننده به‌همراه مقایسه‌ی آن با طرح‌های مشابه در دسترس آورده شده است.



شکل(۱۲): بهینه‌سازی معماری پیشنهادی در هنگام استفاده از خط‌لوله



شکل (۱۳): پیاده‌سازی الگوریتم رایندال با خط لوله به طول۵

۶-۲- پیاده‌سازی الگوریتم رایندال با طول خط لوله کامل

در این روش رمزکننده‌ای مبتنی بر معماری خط‌لوله که طول آن ۱۰ است، مطابق شکل(۱۲)، پیاده‌سازی می‌شود. بدین ترتیب، رمزکننده قابلیت دریافت همزمان دو قطعه ورودی ۱۲۸ بیتی را دارد. سخت‌افزار مصرفی و گذردهی در این روش تقریباً دو برابر روش قبل است. برای مثال تعداد حافظه‌ی بلوکی مورد استفاده در این روش۸۰ بلوک است. نحوه‌ی ذخیره‌ی زیرکلیدها در این روش همانند روش معرفی شده در بخش ۶-۱ است. این طرح نیز بر روی تراشه‌های XCV1000-6 و XCV1000e-8 سنتز شده است و نرخ گذردهی 38.272 Gbps برای گونه دارای درجه‌ی سرعت ۸ آن ثبت گردید. جدول(۲)کارایی این طرح را نیز نشان داده است.

همان‌طور که در جدول(۲)مشاهده می‌شود طرح پیشنهادی براساس تراشه‌ی XCV1000e-8، هم از نظر نرخ گذردهی و هم از نظر مقدار Mbps/Slice از سایر طرح‌ها در وضعیت بهتری قرار دارد. تنها طرح قابل مقایسه با طرح پیشنهادی طرح[۲۲] است که بر روی تراشه‌ی یکسانی سنتز شده و از لحاظ پارامتر Mbps/Slice نزدیک به طرح پیشنهادی است. با این حال طرح پیشنهادی نسبت به طرح مذکور دارای مزایای دیگری نیز هست که ازجمله می‌توان به پالس ساعت کمتر، نرخ گذردهی بیشتر و استفاده‌ی متوازن‌تر از منابع FPGA اشاره نمود.

جدول (۲) نتایج پیاده‌سازی‌های مختلف الگوریتم رایندال به‌صورت خط لوله

	نوع FPGA	فرکانس ساعت(MHz)	نرخ خروجی(Gbps)	Block Ram	مقدار Slice مصرفی	Mbps/Slice
طرح پیشنهادی خطلوله به طول ۵	XCV1000e-6	125.5	16.064	40	4887	1.605
طرح پیشنهادی خطلوله به طول ۱۰	XCV1000e-6	125.5	32.128	80	9196	1.653
طرح پیشنهادی خطلوله به طول ۵	XCV1000e-8	149.5	19.136	40	4912	1.907
طرح پیشنهادی خطلوله به طول ۱۰	XCV1000e-8	149.5	38.272	80	9227	1.965
Elbirt[17]	XCV1000-4	31.8	1.937	0	10992	0.176
McLoone [15]	XCV812E -8	54.35	6.9	100	2222	0.459
Mcloone[18]	XCV812E	93.9	12.02	244	2000	0.362
Saggese[19]	XCV2000e-8	158	20.30	100	5810	1.09
Standaert[20]	XCV3200e-8	145	18.56	0	15112	1.228
Jarvinen[21]	XCV1000e-8	129.2	16.5	0	11719	1.407
Zhang[22]	XCV1000-6	125.3	16.03	0	11014	1.456
Zhang[22]	XCV1000e-8	168.4	21.56	0	11022	1.956
Hodjat [27]	XC2VP20 -7	168.3	21.54	84	5177	1.352

۷- نتیجه‌گیری

موضوع این مقاله بررسی پیاده‌سازی الگوریتم رایندال بر روی FPGA بود. برای این منظور پس از مرور الگوریتم رایندال مدل‌های مختلفی که در پیاده‌سازی‌های این الگوریتم از آن‌ها استفاده می‌شود، معرفی شدند و معماری جدیدی برای پیاده‌سازی این الگوریتم ارائه گردید. معماری پیشنهادی مبتنی بر مدل ضربه‌ای است که در آن از هر دو سطح پالس ساعت برای دسترسی به منابع ارزشمندی

همچون واحد جعبه جانشینی استفاده شد. در این معماری با موازی‌سازی بخش‌های مختلف اعم از پیاده‌سازی جعبه‌های جانشینی، استفاده اشتراکی از آن‌ها در دو رمز کننده، ساختاری موازی‌گونه برای دسترسی به زیرکلیدها رمزکننده‌ی فیدبکی و استفاده از روش‌های بهبوددهنده، توانستیم رمزکننده‌هایی با کارآیی مناسبی برای استفاده در کاربردهای مختلف طراحی کنیم. طرح‌های ارائه شده را که مبتنی بر معماری ضربه‌ای هستند، می‌توان در کاربردهایی با نرخ‌های متوسط، بالا و بسیار بالا استفاده نمود. در طرحی که برای نرخ‌های متوسط ارائه شد از معماری ضربه‌ای به‌همراه ساختاری برای ذخیره‌سازی زیرکلیدها استفاده شد، که حاصل آن رمزکننده‌ای با نرخ 3.057Gbps با حجم سخت‌افزار مناسب بود. پس از آن ملاحظه شد که با بهینه کردن طرح پیشنهادی برای ساختار خط لوله و استفاده از FPGA با درجه‌ی سرعت بالاتر می‌توان به سرعت‌های بالایی دست یافت. بر این اساس دو رمزکننده مبتنی بر خط لوله پیشنهاد شد. در رمزکننده‌ی اول با تکرار ۵ واحد از معماری پیشنهادی، سرعت 19.136 Gbps به‌دست آمد. با صرف هزینه‌ی سخت‌افزار بیشتر و پیاده‌سازی کامل خط لوله (به‌طول ۱۰) رمزکننده‌ی دیگری طراحی گردید که نرخ گذردهی 38.272 Gbps را به ارمغان آورد. مقایسه‌ی طرح‌های پیشنهادی با طرح‌های مشابه در دسترس، نشان از مزیت نسبی معماری و طرح‌های ارائه شده در این مقاله دارد. از معماری ضربه‌ای پیشنهادی می‌توان برای پیاده‌سازی کارآمد دیگر الگوریتم‌های رمزنگاری قطعه‌ای که دارای جعبه‌ی جانشینی باشد نیز استفاده نمود.

قدردانی

نویسندگان مقاله بر خود لازم می‌بینند که از داوران محترمی که با ارائه نظرات ارزشمندشان باعث ارتقای مقاله شدند تشکر کنند.

مراجع

[1] Schneier, B., "Applied Cryptography: Protocols, Algorithms, and Source Code in C," John Wiley & Sons, 1996.

[2] Nechavatal, J., "Report on the Development of Advanced Encryption Standard (AES)," NIST, October 2000.

[3] Daemen, J. and Rijmen, V., "The Rijndeal Block Cipher: AES Proposal," *First AES Candidate Conference (AES1)*, August 1998, pp.20-22.

[4] Aoki, K. and Lipmaa, H., "Fast Implementation of AES Candidates," *3rd AES Conference*, April 2000, pp.106-120, http://www.nist.gov/aes.

[5] Hodjat, A., "Area-Throughput Trade-Offs for Fully Pipelined 30 to 70 Gbits/sec AES Processors," *IEEE Transactions on Computers*, VOL. 55, NO. 4, April, 2006.

[6] Li, h. and Li, j., "A High Performance Sub-Pipelined Architecture for AES," *Proceedings of the 2005 International Conference on Computer Design (ICCD'05)*.

[7] Kuo, H., Verbauwhede, I. and Schaumont, P., "A 2.29 Gbits/sec, 56 mW Non-Pipelined Rijdeal AES Encryption IC in a 1.8V, 0.18 μm CMOS Technology," *IEEE Custom Integrated Circuits Conference*, 2002, pp. 147-150.

[8] Gurkaynak, F. K., Gasser, D. and Hug, D., "A 2 Gbits/sec Balanced AES Crypto-Chip Implementation," *in Proc. 14th ACM Great Lakes Symp. VLSI*, Boston, Apr. 2004, pp. 39–44.

[9] Kim, N. S., Mudge, T. and Brown, R., "A 2.3 Gbits/sec Fully Integrated and Synthesizable AES Rijndael Core," *in Proc. IEEE Custom Integrated Circuits Conference*, September. 2003, pp. 193–196.

[10] Daemen, J. and Rijmen, V., "AES Proposal: The Rijndael Block Cipher. Version 2," September, 1999, available at *http://www.nist.gov/aes/*

[11] Fischer, V. and Drutarovsky, M., "Two Methods of Rijndael Implementation in Reconfigurable Hardware," Cryptographic Hardware and Embedded Systems, 3rd *International Workshop*, CHES 2001.

[12] Hsiao, S. Chen, M.C. and Tu, C.S., "Memory-Free Low-Cost Designs of Advanced Encryption Standard Using Common Subexpression Elimination for Subfunctions in Transformations," *IEEE Transactions on Circuits and Systems*, VOL. 53, NO. 3, MARCH 2006.

[۱۳] فانیان, ع. سماوی، ش. برنجکوب، م. "معماری جدید برای پیاده‌سازی الگوریتم رایندال با نرخ پردازش *6.14Gbps*." *مجموعه مقالات سومین کنفرانس انجمن رمز ایران*، صفحه ۳۴۱-۳۴۸، ۱۳۸۴.

[14] Pramstaller, N. and Wolkerstorfer, J., "A Universal and efficient AES Co-processor for Field Programmable Logic Arrays," *LNCS Vol. 3203*, pp. 565-574, 2004.

[15] McLoone, M. and McCanny, J.V., "Rijndael FPGA Implementations Utilizing Look-Up

Tables," *Journal of VLSI Signal Processing Systems*, pp.261-275, 2003.

[16] Alam, J.V., Badaway, J.V. and Jullien, G., "A Novel Pipelined Threads Architecture for AES Encryption Algorithm Department of Electrical," *13th IEEE International Conference*, 17-19 July 2002, pp. 296-302.

[17] Elbirt, A., "An FPGA Implementation and Performance Evaluation of the AES Block Cipher Candidate Algorithm Finalists," *Proc. of the Third Advanced Encryption Standard Candidate Conference*, NIST, Gaithersburg, MD, April 13-14, 2000.

[18] McLoone, M. and McCanny, J.V., "Rijndael FPGA Implementation Utilizing Look-up Tables," *in IEEE Workshop on Signal Processing Systems*, September, 2001, pp. 349–360.

[19] Saggese, G. P., Mazzeo, A., Mazocca, N. and Strollo, A.G.M., "An FPGA Based Performance Analysis of the Unrolling Tiling and Pipelining of the AES Algorithm," *in Proceeding FPL 2003*, Portugal, September 2003.

[20] Standaert, F., Rouvroy, G., Quisquater, J. and Legat, J., "Efficient Implementation of Rijndael Encryption in Reconfigurable Hardware: Improvements & Design Tradeoffs," *in Proc. CHES 2003*, Cologne, Germany, September. 2003.

[21] Jarvinen, K. U., Tommiska, M.T. and Skytta, J.O., "A Fully Pipelined Memory Less 17.8 Gbps AES-128 Encryptor," *in Proc. Int. Symp. Field Programmable Gate Arrays (FPGA 2003),* Monterey, CA, February, 2003, pp. 207–215.

[22] Zhang, X. and Parhi, K.K, "High-Speed VLSI Architectures for the AES Algorithm, "*IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, VOL. 12, NO. 9, September 2004.

[23] Parhami, B., "Computer Arithmetic: Algorithms and Hardware Designs," *Oxford University Press*, 2000.

[۲۴] عطایی, م. افتخاری، ی. سعیدی، ح. برنجکوب، م. "طراحی و پیاده‌سازی رمزگذار و رمزگشای AES با طرح خط لوله فیدبک‌دار،" *مجموعه مقالات سومین کنفرانس انجمن رمز ایران*، صفحه ۳۷۱-۳۶۰، ۱۳۸۴.

[25] Xilinx, "Using Block RAM in Spartan-3 FPGAs," *available at www.xilinx.com/xapp/xapp463.pdf*

[26] Mali, M., Novak, F. and Biasizzo, A., "Hardware Implementation of AES Algorithm," *Journal of Electrical Engineering*, VOL. 56, NO. 9-10, 2005, pp. 265–269.

[27] Hodjat, A. and Verbaushede, I., "A 21.54 Gbits/s fully pipelined AES processor on FPGA," 12th Annual IEEE Symposium on *Field-Programmable Custom Computing Machines*, pp.308–309, 20-23 April 2004,

[28] Good, T. and Benaissa, M., "AES on FPGA from the Fastest to the Smallest," *CHES 2005, LNCS 3659*, pp.427-440, 2005.

علی فانیان دانشجوی دکتری کامپیوتر در دانشگاه صنعتی اصفهان می‌باشد. ایشان مدرک کارشناسی و کارشناسی ارشد را در مهندسی کامپیوتر-سخت افزار را به ترتیب در سالهای ۱۳۷۸ و ۱۳۸۱ از دانشگاه صنعتی اصفهان دریافت کرده است. زمینه‌های تحقیقاتی مورد علاقه‌ی علی فانیان پیرامون امنیت شبکه‌های بی‌سیم است. ایشان همچنین در زمینه طراحی الگوریتم‌های رمزنگاری بر روی سکوی‌های سخت‌افزاری و نرم‌افزاری نیز فعالیت دارد.

نشانی (رایانامک) پست الکترونیکی ایشان عبارت است از:
Samavi96@cc.iut.ac.ir

مهدی برنجکوب، دکترای مهندسی برق خود را در سال ۱۳۷۸ از دانشگاه صنعتی اصفهان اخذ نمود و بلافاصله در دانشکده برق و کامپیوتر همین دانشگاه به عنوان استادیار فعالیت آموزشی و پژوهشی خود را آغاز کرد که کماکان این همکاری ادامه دارد. دروس تحصیلات تکمیلی ارائه شده توسط وی عبارتند از: اصول رمزنگاری، پروتکل‌های رمزنگاری، امنیت شبکه و پردازش گفتار. وی قریب به بیست پروژه‌ی تحصیلات تکمیلی را در زمینه‌های رمزنگاری و امنیت شبکه هدایت کرده‌است. وی همچنین یکی از اعضای هیئت مؤسس انجمن رمز ایران در سال ۱۳۷۹ بوده و هم‌اکنون عضو شورای اجرایی و مسئول کمیته‌ی فناوری و صنعت این انجمن است. از دیگر مسئولیت‌های وی مدیریت گروه پژوهشی امنیت شبکه و سیستم دانشکده‌ی برق و کامپیوتر و مدیریت مرکز تخصصی آپا دانشگاه صنعتی اصفهان است و پروژه‌های متعددی را اجرا و هدایت کرده‌است. عناوین پژوهشی مورد علاقه‌ی وی در حال حاضر امنیت در شبکه‌های بی‌سیم و پروتکل‌های احراز اصالت می‌باشد.

نشانی (رایانامک) پست الکترونیکی ایشان عبارت است از:
Brnjkb@cc.iut.ac.ir

شادرخ سماوی استاد دانشکده برق و کامپیوتر دانشگاه صنعتی اصفهان می باشد. ایشان مدرک کارشناسی در تکنولوژی صنعتی و مدرک کارشناسی در مهندسی برق را به ترتیب در سال‌های ۱۳۵۹ و ۱۳۶۱ از دانشگاه ایالتی کالیفرنیا دریافت کرده است. نامبرده مدرک کارشناسی ارشد را در مهندسی کامپیوتر در سال ۱۳۶۴ از دانشگاه ممفیس آمریکا و مدرک دکتری در مهندسی برق را در سال ۱۳۶۸ از دانشگاه ایالتی می‌سی‌سی‌پی اخذ نمود. زمینه‌های تحقیقاتی مورد علاقه‌ی ایشان، پردازش تصاویر و پیاده‌سازی سخت افزاری الگوریتم‌های مربوطه است. ایشان همچنین در زمینه‌ی طراحی الگوریتم‌های پنهان‌نگاری در تصاویر نیز فعالیت دارد.

نشانی (رایانامک) پست الکترونیکی ایشان عبارت است از:
Fanian@ec.iut.ac.ir