مروری بر عملیات معکوس در میدان‌های متناهی دودویی و اول

رضا روح قلندری، حاتمه مثنایی بورانی و سیاوش بیات سرمدی

چکیده: رمزنگاری کلید عمومی یکی از روش‌های مرسوم رمزنگاری است که به دلیل عدم نیاز به تبادل کلید، در سال‌های اخیر بسیار مورد توجه قرار گرفته است. روش‌هایی مانند توان‌رسانی، جفت‌سازی، رمزنگاری خم بیضوی و همگونی در این دسته قرار می‌گیرند که تاکنون پژوهش‌های زیادی برای کاهش پیچیدگی زمانی و مساحت این روش‌ها صورت گرفته است. عملیات معکوس به عنوان یکی از اصلی‌ترین اعمال موجود در روش‌های رمزنگاری کلید عمومی است که بخش زیادی از پیچیدگی محاسباتی و زمانی را در این پردازنده‌های رمزنگاری به خود اختصاص می‌دهد. بنابراین برای افزایش کارایی و سرعت پردازنده‌های رمزنگاری کلید عمومی، بهبود سرعت و مساحت عملیات معکوس در میدان‌های متناهی بسیار ضروری به نظر می‌رسد.

در این مقاله، روش‌های موجود برای انجام عملیات معکوس بر روی میدان‌های دودویی و اول مورد بررسی قرار گرفته است. در سامانه‌های رمزنگاری امروزی، میدان‌های دودویی به دلیل سازگاری با سخت‌افزار، بسیار پرکاربرد هستند. در ابتدای این نوشتار، روش‌های موجود برای انجام عملیات معکوس در میدان‌های دودویی بررسی، الگوریتم‌های موجود بیان و از نظر پیچیدگی زمانی و منابع مورد نیاز با هم مقایسه شده است. سپس بهترین پیاده‌سازی‌های موجود روی بستر FPGA و به صورت مدار مجتمع معرفی می‌گردد. هدف اصلی در این روش‌ها کاهش تعداد ضرب مورد نیاز برای انجام عملیات معکوس در میدان دودویی و افزایش امکان توازی برای پیاده‌سازی هرچه بهتر این روش‌ها است. میدان‌های اول به دلیل پیچیدگی ساختاری و محاسباتی بیشتر نسبت به میدان‌های دودویی، تنوع و گستردگی کمتری دارند ولی در سالیان اخیر به دلیل ظهور کاربردهای جدیدی در رمزنگاری نظیر روش همگونی، مورد توجه بیشتری قرار گرفته‌اند. محققان این حوزه در تلاش هستند تا ضمن از بین بردن وابستگی زمان اجرای روش‌های موجود به مقدار ورودی، پیچیدگی زمانی و مساحت الگوریتم‌های محاسبه معکوس را تا حد امکان کاهش دهند. ارائه ساختارهایی نظیر آرایه ضربانی در همین راستا صورت گرفته که این مقاله به بررسی این روش‌ها می‌پردازد و در انتها روش‌های انجام عملیات معکوس را در میدان‌های اول از نظر پیچیدگی زمانی و محاسباتی با هم مقایسه می‌کند.

کلیدواژه: **الگوریتم‌های معکوس، میدان‌های متناهی، قضیه کوچک فِرما، الگوریتم اقلیدسی تعمیم‌یافته.**

۱- مقدمه

سامانه‌های رمزنگاری در دنیای امروز از اهمیت ویژه‌ای برخوردار است و در حال حاضر تلاش‌های فراوانی برای افزایش کارایی، امنیت و پیاده‌سازی بهتر این روش‌ها در حال انجام است. روش‌های رمزنگاری در دو دسته رمزنگاری متقارن[1] و نامتقارن[2] (کلید عمومی[3]) جای می‌گیرند. در این بین، روش رمزنگاری کلید عمومی به دلیل عدم نیاز به کانال امن برای تبادل کلید، در سال‌های اخیر بسیار مورد توجه قرار گرفته است. رمزنگاری خم بیضوی[4]، جفت‌شدن[5]، توان‌رسانی[6] (RSA) و همگونی[7] از جمله مهم‌ترین روش‌های این دسته هستند. محاسبات رمزنگاری در این روش‌ها روی میدان‌های متناهی[8] انجام می‌شود که عملیاتی نظیر جمع، ضرب و معکوس در این میدان‌ها به روش خاصی انجام می‌شوند. میدان‌های متناهی را می‌توان به دو دسته میدان‌های اول[9] و دودویی[10] تقسیم کرد که عملیات معکوس در هر دوی اینها نسبت به سایر اعمال هزینه زمانی و مساحت بیشتری دارد، به صورتی که عملیات معکوس در پردازنده‌های مبتنی بر خم بیضوی یا همگونی، بخش زیادی از هزینه زمانی و محاسباتی را به خود اختصاص می‌دهد. بنابراین برای افزایش کارایی و سرعت پردازنده رمزنگاری مبتنی بر خم بیضوی یا همگونی، بهبود سرعت و کارایی عملیات معکوس بسیار ضروری به نظر می‌رسد.

در روند این مقاله، ابتدا به معرفی سامانه‌های رمزنگاری نامتقارن پرداخته می‌شود. سپس انواع میدان‌های متناهی، عملیات معکوس، پایه‌های مورد استفاده برای نمایش عناصر در میدان متناهی، زنجیره جمعی[11] و قضیه کوچک فِرما معرفی می‌شوند. در بخش بعدی، روش‌های موجود برای انجام عملیات معکوس روی میدان دودویی معرفی و از نظر پیچیدگی زمانی و مساحت با هم مقایسه می‌شوند. در انتهای این بخش، بهترین پیاده‌سازی‌های موجود روی بستر FPGA و به صورت مدار مجتمع مشخص می‌شوند. سپس به معرفی روش‌های موجود برای محاسبه معکوس روی میدان‌های اول پرداخته می‌شود و در انتها پیچیدگی زمانی این روش‌ها گزارش می‌گردد.

۱-۱ رمزنگاری نامتقارن

این رمزنگاری موسوم به رمزنگاری کلید عمومی، اولین بار در سال ۱۹۷۶ میلادی توسط دیفی و هلمن ارائه شده که برخلاف رمزنگاری متقارن، دارای دو نوع کلید خصوصی و عمومی است. کلید عمومی برای عملیات رمزگذاری به کار می‌رود و همه از آن اطلاع دارند. عملیات رمزگذاری در رمزنگاری نامتقارن امری همگانی است و نیاز به اطلاعات محرمانه ندارد. کلید خصوصی را تنها گیرنده پیام در اختیار داشته و برای عملیات رمزگشایی از آن استفاده می‌کند.

این مقاله در تاریخ ۱۴ شهریور ماه ۱۳۹۷ دریافت و در تاریخ ۷ خرداد ماه ۱۳۹۸ بازنگری شد.

رضا روح قلندری، دانشکده مهندسی کامپیوتر، دانشگاه صنعتی شریف ، تهران، ایران، (email: ghalandari@ce.sharif.edu).

حاتمه مثنایی بورانی، دانشکده مهندسی کامپیوتر، دانشگاه صنعتی شریف ، تهران، ایران، (email: mosanaei@ce.sharif.edu).

سیاوش بیات سرمدی (نویسنده مسئول)، دانشکده مهندسی کامپیوتر، دانشگاه صنعتی شریف ، تهران، ایران، (email: sbayat@sharif.edu).

1. Symmetic Encryption
2. Asymmetric Encryption
3. Public-Key Cryptography
4. Elliptic-Curve Cryptography
5. Pairing
6. Rivest, Shamir, and Adleman
7. Isogeny
8. Finite Field (Galois Field)
9. Prime Field
10. Binary Field
11. Additional Chain

از دیگر کاربردهای مهم این نوع رمزنگاری، امضای دیجیتال[1] است که که در آن امضاکننده، کلید خصوصی و بررسی‌کننده[2] صحت امضا، کلید عمومی را نیاز دارند. بنابراین در این کاربردها نیازی به یک کانال امن برای انتقال کلید خصوصی نخواهد بود. در رمزنگاری نامتقارن، از نظر محاسباتی دستیابی به اطلاعات خصوصی با دانستن کلید عمومی در زمان معقول، غیر ممکن است.

الگوریتم‌هایی نظیر تعویض کلید دفی- هلمن، RSA، جفت‌سازی و رمزنگاری خم بیضوی در این گروه قرار می‌گیرند. به دلیل طول کلید کمتر، رمزنگاری خم بیضوی در سطح امنیت یکسان به مساحت، توان و فضای ذخیره‌سازی کمتری نسبت به سایر روش‌های نامتقارن نیاز دارد. از طرفی با معرفی محاسبات کوانتومی، امنیت روش‌های رمزنگاری کلاسیک با چالش‌هایی نظیر کاهش سطح امنیت روبه‌رو گردیده که این امر سبب تمرکز محققان بر روی الگوریتم‌های رمزنگاری پساکوانتومی شده است. روش رمزنگاری همگونی که به تازگی معرفی شده، یکی از روش‌های رمزنگاری پساکوانتومی است که جزء روش‌های کلید عمومی محسوب می‌شود. عملیات معکوس یک عملیات اصلی در پردازنده‌های رمزنگاری خم بیضوی و همگونی است که هزینه زمانی و مساحت زیادی را در این پردازنده‌ها به خود اختصاص می‌دهد. بنابراین کاهش هزینه زمانی و مساحت آن بسیار حایز اهمیت است.

۱-۲ میدان‌های متناهی

میدانی با تعداد اعضای محدود را میدان متناهی می‌نامند که میدان‌های پیمانه‌ای یکی از انواع این میدان‌ها هستند. میدان‌های متناهی را به صورت $GF(p)$ نمایش می‌دهند که p مشخصه میدان نامیده می‌شود. اگر $p=2$ باشد آن را میدان دودویی و در غیر این صورت میدان اول می‌نامند. میدان $GF(2)$ تنها دارای دو عضو $\{0,1\}$ است. در میدان دودویی چون محاسبات به پیمانه ۲ انجام می‌شود، در عمل جمع، رقم نقلی وجود ندارد. در واقع هنگامی که دو بیت یک به پیمانه ۲، با هم جمع می‌شوند، حاصل برابر صفر خواهد بود. بنابراین عمل جمع در این میدان معادل عملگر XOR است. برای ضرب دو عنصر در این میدان نیز تنها به یک عمل AND نیاز است. میدان تعمیم‌یافته[3] را با $GF(p^m)$ نشان می‌دهند و از مرتبه p^m می‌باشد که m عددی مثبت و p مشخصه میدان است.

برای مثال در میدان تعمیم‌یافته دودویی $GF(2^m)$ عناصر عضو میدان m بیتی بوده و برای عمل جمع در این میدان به m گیت XOR نیاز است. در واقع چون در میدان‌های دودویی محاسبات به پیمانه ۲ انجام می‌شود، این میدان‌ها برای پیاده‌سازی روی سخت‌افزار مناسب‌تر هستند [1] و پیاده‌سازی اعمال محاسباتی نظیر جمع، مجذور و ضرب در این میدان‌ها به صورت سخت‌افزاری سربار مساحت و زمان کمتری را در پی خواهد داشت. برای نمایش هر عنصر در میدان متناهی از پایه‌های گوناگونی استفاده می‌شود که پایه نرمال[4] و چندجمله‌ای[5] معروف‌ترین آنها می‌باشند. نحوه انجام اعمالی نظیر ضرب و مجذور بر حسب پایه مورد استفاده متفاوت است که در ادامه این پایه‌ها به صورت مختصر معرفی می‌شوند.

1. Digital Signature
2. Verifier
3. Extended Field
4. Normal Basis
5. Polynomial Basis

۱-۳ پایه نرمال

اگر β عنصری از میدان $GF(p^m)$ باشد، مجموعه m عنصری $\{\beta,\beta^p,\beta^{p^2},\ldots,\beta^{p^{m-1}}\}$ مستقل خطی بوده و پایه نرمال را بر روی $GF(p)$ برای $GF(p^m)$ تشکیل می‌دهد. نمایش عناصر متعلق به $GF(p^m)$ با استفاده از این پایه مطابق (۱) خواهد بود. مهم‌ترین مزیت این پایه آن است که مجذور در آن یک جابه‌جایی[6] چرخشی به راست است [۲] و هزینه زمانی و مساحت ندارد اما هزینه عملیات ضرب این پایه از پایه چندجمله‌ای بیشتر است. پایه نرمال گوسی[7] (GNB) [۳] یکی از انواع پایه‌های نرمال است که در بیشتر پژوهش‌های موجود در حوزه عملیات معکوس مورد استفاده قرار گرفته و مزیت آن سادگی طراحی و کارایی بالاتر آن نسبت به پایه نرمال عادی است

$$A\in GF(p^m)\ ,\ a_i\in GF(p)$$
$$(A)_{NB}=a_{m-1}\beta^{p^{m-1}}+\ldots+a_1\beta^p+a_0\beta \qquad (1)$$

۱-۴ پایه چندجمله‌ای

اگر α ریشه چندجمله‌ای اولیه از درجه m باشد آن گاه $\{1,\alpha,\alpha^2,\ldots,\alpha^{m-1}\}$ یک مجموعه m عنصری مستقل خطی روی $GF(p^m)$ است که پایه چندجمله‌ای را تشکیل می‌دهد. بنابراین هر عنصر از $GF(p^m)$ را می‌توان توسط پایه چندجمله‌ای مطابق (۲) نمایش داد

$$A\in GF(p^m)\ ,\ a_i\in GF(p)$$
$$(A)_{PB}=a_{m-1}.\alpha^{m-1}+\ldots+a_1.\alpha+a_0 \qquad (2)$$

۱-۵ زنجیره جمعی

یک زنجیره جمعی برای عدد طبیعی n یک دنباله از اعداد طبیعی به صورت $v(v_0,\ldots,v_s)$ است که از $v_0=1$ آغاز شده و به $v_s=n$ ختم می‌شود. سایر اعضای زنجیره مطابق (۳) از جمع دو عنصر قبل از خود ساخته می‌شوند

$$\text{for } 0<i\le s: v_i=v_x+v_y\ ,\ 0\le x,y<i \qquad (3)$$

زنجیره جمعی بهینه به زنجیره‌ای گفته می‌شود که از کمترین عناصر ممکن تشکیل شود. این زنجیره برای هر عدد طبیعی با استفاده از الگوریتم عقبگرد[8] بهینه قابل یافتن است. زنجیره بَرِر[9] [۴]، نوع خاصی از زنجیره‌های جمعی است که کاربرد زیادی در پیاده‌سازی‌های سخت‌افزاری عملیات معکوس دارد. در این زنجیره، لزوماً یکی از اعداد v_x یا v_y برابر آخرین عدد موجود در زنجیره جمعی خواهد بود. به دلیل استفاده از عنصر قبلی در هر مرحله، این زنجیره به فضای ذخیره‌سازی کمتری در حین محاسبه نیاز دارد.

۱-۶ معکوس روی میدان‌های متناهی

معکوس پرهزینه‌ترین عملیات در میدان‌های متناهی است. در میدان متناهی با چندجمله‌ای مشخصه $f(x)$، معکوس یک عنصر نظیر $A\in GF(2^m)$، عنصر دیگری از همان میدان مانند $A^{-1}\in GF(2^m)$ است که در رابطه $A\times A^{-1}=1 \bmod f(x)$ صدق می‌کند. یک راه ابتدایی

6. Shift
7. Gaussian Normal Basis
8. Backtracking Algorithm
9. Brauer

برای محاسبه معکوس یک عنصر، بررسی همه اعضای میدان برای برقراری رابطه فوق است که معقول به نظر نمی‌رسد.

برای میدان $GF(۲^m)$ حداکثر به ۲^m بار آزمایش نیاز است. در راستای بهبود روش محاسبه معکوس در میدان متناهی، دو الگوریتم اقلیدسی تعمیم‌یافته (EEA)[۱] و الگوریتم مبتنی بر قضیه کوچک فِرما (FLT)[۲] برای یافتن معکوس یک عنصر در میدان متناهی معرفی شده است. روش الگوریتم اقلیدسی تعمیم‌یافته بر مبنای الگوریتم یافتن بزرگ‌ترین مقسوم‌علیه مشترک (GCD)[۳] عمل می‌کند که در ادامه به آن پرداخته خواهد شد. طبق قضیه کوچک فِرما در میدان‌های متناهی برای هر عنصر (۴) برقرار است که روش‌های مبتنی بر قضیه کوچک فِرما از این رابطه بهره می‌برند

$$A \in GF(p) \Rightarrow A = A^p \pmod p \Rightarrow \\ A^{-۱} = A^{p^m - ۲} \pmod p \qquad (۴)$$

در ادامه جدیدترین کارهای انجام‌شده در میدان‌های دودویی و اول به تفکیک بررسی شده و به تشریح دقیق‌تر هر کدام از روش‌های موجود پرداخته خواهد شد.

۲- عملیات معکوس روی میدان دودویی

الگوریتم‌های معکوس روی میدان‌های دودویی به دو دسته کلی مبتنی بر الگوریتم تعمیم‌یافته اقلیدسی و روش مبتنی بر قضیه کوچک فِرما تقسیم می‌شوند. در الگوریتم‌های معکوس روی میدان دودویی، می‌توان از پایه‌های مختلف استفاده کرد که مهم‌ترین و متداول‌ترین این پایه‌ها، پایه نرمال و چندجمله‌ای است که در بخش بعد، کارهای انجام‌شده در هر کدام از این پایه‌ها به تفکیک مورد بررسی قرار می‌گیرند.

۲-۱ روش اقلیدسی تعمیم‌یافته

در دامنه اقلیدسی برای هر دو عنصر عضو دامنه اقلیدسی نظیر (a,b)، بزرگ‌ترین مقسوم‌علیه مشترک از ترکیب خطی این دو عنصر، طبق (۵) حاصل می‌شود. به این روش، روش نردبانی گفته می‌شود

$$\text{if } a,b \in D \text{ then } GCD(a,b) = GCD(a, b - r \times a) \qquad (۵)$$

با تعمیم این روش می‌توان علاوه بر محاسبه بزرگ‌ترین مقسوم‌علیه مشترک، یک رابطه خطی نیز بین دو عدد ورودی پیدا کرد

$$d = GCD(a,b)\ ,\ a',b' \rightarrow a \times a' + b \times b' = d \qquad (۶)$$

برای محاسبه معکوس یک عنصر در میدان اول $(a \in GF(p^m))$، با توجه به (۷) امکان استفاده از الگوریتم تعمیم‌یافته اقلیدسی برای یافتن بزرگ‌ترین مقسوم‌علیه مشترک وجود دارد. با توجه به اول‌بودن p، رابطه $GCD(a,p) = ۱$ همواره برقرار است. در نتیجه با کمک الگوریتم یافتن بزرگ‌ترین مقسوم‌علیه مشترک می‌توان $b = a^{-۱}$ را محاسبه کرد

$$\left.\begin{array}{l} a \in GF(p) \rightarrow \gcd(a,p) = ۱ \\ a \times b + p \times c = ۱ \bmod p \end{array}\right\} a \times b = ۱ \bmod p \Rightarrow \\ b = a^{-۱} \bmod p \qquad (۷)$$

۲-۱-۱ روش اقلیدسی تعمیم‌یافته برای میدان دودویی

الگوریتم یافتن بزرگ‌ترین مقسوم‌علیه مشترک و به دنبال آن الگوریتم محاسبه معکوس به روش اقلیدسی تعمیم‌یافته به صورت کلی مطرح شده و به ازای هر عنصری عضو میدان متناهی قابل استفاده است اما نسخه‌ای از این روش معرفی گردیده که برای پیاده‌سازی در میدان‌های دودویی مناسب است. الگوریتم ۱، الگوریتم محاسبه معکوس به روش اقلیدسی تعمیم‌یافته را روی میدان دودویی نشان می‌دهد. در الگوریتم ۱ به جای استفاده از اعداد صحیح که در الگوریتم اولیه استفاده می‌شود، از معادل آنها در پایه چندجمله‌ای استفاده شده است.

الگوریتم ۱: محاسبه معکوس به روش اقلیدسی تعمیم‌یافته روی میدان دودویی [۵]

Input: $a(x), p(x), \deg(a(x)) < \deg(p(x))\ ,\ a \neq 0$

Output: $\beta = a^{-1}(x)$

1. $b(x) = 1\ ,\ c(x) = 0\ ,\ u(x) = a(x)\ ,\ v(x) = p(x)$
2. while $\deg(u) \neq 0$ do

1-2. $q(x) = \dfrac{u(x)}{v(x)}$

2-2. $v(x) = u(x)\ ,\ u(x) = v(x) - q(x) \times u(x)$

3-2. $c(x) = b(x)\ ,\ b(x) = c(x) - q(x) \times b(x)$

3. end while
4. return $c(x)$

این الگوریتم نیز مشابه الگوریتم اولیه است و تنها محاسبات آن روی میدان دودویی انجام می‌شود. حلقه اصلی برنامه شامل یک تقسیم، دو ضرب و دو تفریق است که در میدان دودویی معادل جمع است. همچنین به یک مقایسه‌کننده کوچک برای مقایسه بزرگ‌ترین درجه دو چندجمله‌ای نیاز است.

۲-۱-۲ روش اقلیدسی تعمیم‌یافته دودویی (b-EEA)

یکی از چالش‌های موجود در روش الگوریتم اقلیدسی تعمیم‌یافته تعداد زیاد عملیات ضرب و تقسیم است که هزینه زیادی در میدان‌های دودویی دارد. برای رفع این مشکل، الگوریتم اقلیدسی تعمیم‌یافته دودویی[۴] برای این میدان‌ها معرفی شده است [۵]. در این الگوریتم، ضرب و تقسیم به کمک ضرب و تقسیم‌های متوالی بر دو پیاده می‌شود که هزینه بسیار کمی دارند. چون در پیاده‌سازی سخت‌افزاری معادل جابه‌جایی به چپ و راست هستند.

همان طور که در الگوریتم ۲ مشاهده می‌کنید چون این الگوریتم از عملگرهای جمع و تفریق استفاده می‌کند، در مقایسه با الگوریتم اقلیدسی تعمیم‌یافته برای پیاده‌سازی روی سخت‌افزار مناسب‌تر است. در واقع در این الگوریتم، ضرب و تقسیم پرهزینه قبلی با چندین عملیات جابه‌جایی به صورت متوالی جایگزین شده است.

الگوریتم ۲: محاسبه معکوس به روش اقلیدسی تعمیم‌یافته دودویی در میدان دودویی [۵]

Input: $a(x), p(x), \deg(a(x)) < \deg(p(x))\ ,\ a \neq 0$

Output: $r(x) \Rightarrow r(x) = a(x)^{-1} \bmod p(x)$

1. $u(x) = p(x)\ ,\ v(x) = a(x)\ ,\ r(x) = 0\ ,\ s(x) = 1$
2. while $v(x) \neq 0$
3. while $u_0 = 0$ do
4. $u(x) = \dfrac{u(x)}{x}$
5. if $r_0 = 0$ then $r(x) = \dfrac{r(x)}{x}$ else $\dfrac{r(x) + p(x)}{x}$

1. Extended Euclidean Algorithm
2. Fermat's Little Theorem
3. Greatest Common Divisor
4. Binary Extended Euclidean Algorithm

6. end while
7. while $v_0 = 0$ do
8. $v(x) = \frac{u(x)}{x}$
9. if $s_0 = 0$ then $s(x) = \frac{s(x)}{x}$ else $s = \frac{s(x)+p(x)}{x}$
10. end while
11. if $\deg(u(x)) > \deg(v(x))$ then
12. $u(x) = u(x) - v(x)$, $r(x) = r(x) - s(x)$
13. else $v(x) = v(x) + u(x)$, $s(x) = s(x) + r(x)$
14. end while
15. return $r(x) \bmod p(x)$

۲-۱-۳ الگوریتم معکوس مونتگمری

در ضرب به روش مونتگمری عملیات پرهزینه کاهش با عملیاتی کم‌هزینه‌تر جایگزین می‌شود اما در این رابطه به جای محاسبه حاصل در دامنه عادی، حاصل‌ضرب دو عدد در دامنه مونتگمری محاسبه می‌شود. بنابراین ابتدا باید عناصر به دامنه مونتگمری منتقل شوند [۶]. در الگوریتم ضرب مونتگمری، برای ضرب دو عدد از میدان اول p به جای محاسبه $c = a \times b \pmod p$ از $c = a \times b \times 2^{-n} \pmod p$ استفاده می‌گردد.

الگوریتم معکوس مونتگمری نیز مشابه همین ضرب عمل کرده و معکوس یک عنصر از میدان اول را محاسبه می‌کند اما به دلیل انجام محاسبات در دامنه مونتگمری و کاهش سربار عملیات کاهش سبب افزایش کارایی عملیات معکوس می‌گردد. این الگوریتم شامل دو فاز است. الگوریتم ۳ روش محاسبه معکوس مونتگمری را در میدان دودویی نشان می‌دهد.

در فاز اول مقدار $r(x) = a(x)^{-1} x^k \bmod p(x)$ محاسبه می‌شود که مقدار k در بازه $n+1 \le k \le \deg(a(x)) + n + 1$ قرار دارد. در فاز دوم نیز مقدار $r(x) = a(x)^{-1}$ محاسبه می‌گردد. یکی از مشکلات این روش زیادبودن تعداد سیکل کلاک به دلیل ساختار ترتیبی این روش است.

یکی از مشکلات این روش زیادبودن تأخیر مسیر بحرانی به دلیل ساختار ترتیبی این روش می‌باشد. مشکل دیگر آن مساحت بالای آن در پیاده‌سازی سخت‌افزاری به علت ساختار دوبخشی آن است. در [۷] معماری کارای آن به صورت مدار مجتمع کلان مقیاس (VLSI)[1] پیاده‌سازی شده است. در [۸] بر روی کاهش تأخیر مسیر بحرانی و نویسندگان [۹] برای کاهش سربار مساحت در این روش تلاش کرده‌اند. مقاله [۸] از محاسبات $n/2$ بیتی بهره می‌برد که موفق به بهبود فرکانس کاری و افزایش برون‌دهی کلی شود. در [۸] یک نمایش مبنای دو برای اعداد معرفی شده که با کمک آن وابستگی داده‌ای موجود در مسیر بحرانی را کاهش می‌دهد.

الگوریتم ۳: محاسبه معکوس به روش مونتگمری در میدان دودویی [۸]

Input: $a(x), p(x), \deg(a(x)) < \deg(p(x)) , a \neq 0$

Output: $r(x) \Rightarrow r(x) = a(x)^{-1} \bmod p(x)$

1. $u(x) = p(x)$, $v(x) = a(x)$, $r(x) = 0$, $s(x) = 1$, $k = 1$
2. while $v(x) > 0$
3. if $u_0 = 0$ then $u(x) = u(x)/x$, $s = x\,s(x)$
4. else if $v_0(x) = 0$ then $v(x) = \frac{v(x)}{x}$, $r = x.r(x)$
5. else if $\deg(u(x)) > \deg(v(x))$ then
6. $u = \frac{(u(x) - v(x))}{x}$, $r(x) = r(x) + s(x)$, $s(x) = x.s(x)$
7. else $v(x) = [v(x) - u(x)]/x$, $s(x) = r(x) + s(x)$, $r(x) = x.r(x)$
8. $k = k + 1$
9. end while
10. if $\deg(r(x)) \ge \deg(p(x))$ then
11. $r(x) = (r(x) - p(x))$ end if
12. return $r(x) = p(x) - r(x)$, k

در ادامه می‌توان روش اقلیدسی تعمیم‌یافته دودویی را با روش مونتگمری دودویی بر اساس هزینه عملیاتی مقایسه کرد. از آنجایی که روش‌های مبتنی بر الگوریتم اقلیدسی تعمیم‌یافته بر اساس ورودی هزینه زمانی و مساحت متفاوتی دارند، شکل ۱ بعد از ۱۰۰۰۰۰ بار اجرای الگوریتم‌های فوق با استفاده از ۱۰۰ عدد ورودی متفاوت و ۱۰۰ عدد اول متفاوت حاصل شده است. با توجه به شکل ۱، در مجموع از نظر تعداد عملیات، روش مونتگمری به وضوح هزینه کمتری نسبت به روش اقلیدسی تعمیم‌یافته دارد. پس با توجه به ساختار ساده و هزینه بسیار کمتر، این روش برای پیاده‌سازی سخت‌افزاری مطلوب‌تر به نظر می‌رسد.

۲-۱-۴ روش برِنت- کونگ و معماری آرایه‌ای ضربانی

یکی از قسمت‌های مهم در الگوریتم ۲ و الگوریتم ۳، مقایسه بزرگ‌ترین درجه دو چندجمله‌ای در حلقه اصلی برنامه است. این عمل، علاوه بر هزینه محاسباتی بالا، باعث وابستگی زمان اجرای الگوریتم به مقدار ورودی آن می‌شود. در پیاده‌سازی‌های سخت‌افزاری وابستگی زمان اجرای الگوریتم به ورودی می‌تواند آن را در برابر حملات کانال جانبی آسیب‌پذیر کند و سطح امنیت سامانه را کاهش دهد.

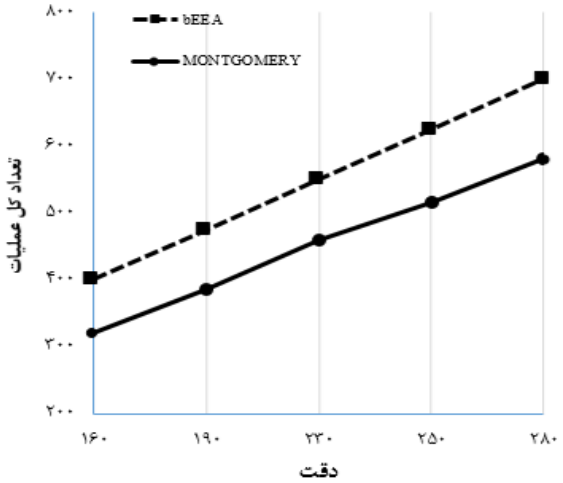
در [۱۰] آقایان برِنت و کونگ[2] روشی جدید برای حل این مشکل معرفی کردند. در این مقاله به کمک ایده‌ای ساده و جالب، اختلاف درجه‌های دو چندجمله‌ای در متغیری ذخیره می‌شود. این عمل به دلیل کاهش هزینه محاسباتی در پیاده‌سازی سخت‌افزاری بسیار مفید خواهد بود و باعث بهبود زمان اجرا و مساحت معکوس‌کننده خواهد شد. همچنین با این روش واحد کنترلی مدار فقط شامل سیگنال‌های داخلی بوده و از سربار پیاده‌سازی آن کاسته می‌شود. این ایده، مسیر جدیدی را برای پیاده‌سازی عملیات معکوس به روش آرایه‌ای ضربانی[3] ایجاد نموده است.

این معماری به علت خواصی مثل ساختار منظم[4]، پیمانه‌ای[5] و هم‌زمانی[6]، یکی از بهینه‌ترین روش‌های پیاده‌سازی به صورت مدار مجتمع خاص‌منظوره است [۱۱]. این ساختار به علت برون‌دهی[7] بالا در کاربردهایی بیشتر مورد استفاده قرار می‌گیرد که باید تعداد زیادی عملیات معکوس به طور متوالی انجام شود.

تبدیل الگوریتم‌های موجود در میدان دودویی به معماری آرایه‌ای ضربانی نسبت به الگوریتم‌های مربوط به میدان اول، پیچیدگی کمتری دارد و به همین دلیل این معماری در میدان دودویی بیشتر مورد استفاده قرار گرفته و مقاله‌های ارائه‌شده برای بهبود این معماری، بیشتر روی میدان دودویی تمرکز دارند. به طور کلی می‌توان روش‌های مبتنی بر ساختار آرایه‌ای ضربانی را به سه دسته تقسیم کرد.

1. Very Large Scale Integration

2. Barent and Kung
3. Systolic Arrays
4. Regularity
5. Modularity
6. Concurrency
7. Throughput



شکل ۱: مقایسه الگوریتم اقلیدسی تعمیم‌یافته دودویی و الگوریتم معکوس مونتگمری از نظر تعداد عملیات.

الف) روش آرایه‌ای ضربانی اِستین

این روش در [۱۲] معرفی شده و در [۱۳] بهبود و برای عملیات تقسیم نیز گسترش داده شده است. این روش یکی از بهترین روش‌های پیاده‌سازی آرایه‌ای ضربانی برای الگوریتم اقلیدسی دودویی است. همان طور که در الگوریتم ۴ نشان داده شده است این الگوریتم فقط از جابه‌جایی به راست بهره می‌برد و برای استفاده در عملیات تقسیم نیز مناسب است.

یکی از کاربردهای عملیات معکوس، استفاده برای اجرای عملیات تقسیم است. در واقع وقتی نیاز به محاسبه تقسیم پیمانه‌ای $x = A/B$ باشد، در ابتدا معکوس B محاسبه می‌شود. سپس از ضرب معکوس حاصل در A مقدار x حاصل می‌شود. البته می‌توان هر روش محاسبه معکوس را به طور مستقیم با اضافه‌کردن یک ضرب به عملیات تقسیم تبدیل کرد.

یکی از مشکلات استفاده از جابه‌جایی به چپ در روش‌های محاسبه معکوس، این است که اگر هدف استفاده از عملیات معکوس، محاسبه تقسیم باشد، تعداد سیکل کلاک آن بیشتر خواهد شد. چون ممکن است که درجه چندجمله‌ای هنگام استفاده از جابه‌جایی به چپ، از درجه چندجمله‌ای مشخصه میدان بیشتر شده و در نتیجه برای محاسبه مقدار صحیح حاصل، به عملیات کاهش[1] نیاز باشد که این روش با استفاده از جابه‌جایی به راست این مشکل را حل کرده است.

همچنین این روش مشکل ثابت‌نبودن زمان اجرا در روش اقلیدسی تعمیم‌یافته را حل کرده است. این روش در هر اجرا به $2m-1$ گام نیاز دارد که m درجه چندجمله‌ای مشخصه میدان است. ثابت‌بودن زمان اجرا همچنین برای پیاده‌سازی آرایه‌ای ضربانی بسیار مهم است زیرا تعداد سطرها در این معماریِ دوبعدی برابر با تعداد تکرار حلقه اصلی برنامه است. در واقع هر سطر در این معماری به منزله یک بار اجرای حلقه اصلی برنامه است. همچنین یک سلول در هر سطر برای مدیریت سیگنال‌های داخلی استفاده می‌شود. طبق الگوریتم ۴ این معماری به سه سیگنال کنترلی نیازمند است. سیگنال اول برای خط دوم الگوریتم ۴ که برای شمردن دفعات اجرای برنامه از صفر تا $2m-1$ به کار می‌رود. دیگری برای بررسی مقدار v_0 و آخرین سیگنال هم برای بررسی مقدار δ به کار می‌رود.

الگوریتم ۴: محاسبه معکوس به روش اقلیدسی تعمیم‌یافته اِستین [۱۳]

Input: $a(x), p(x)$, $a \neq 0$

Output: $r(x) = a^{-1}(x) \bmod p(x)$

1. $u(x) = p(x)$, $v(x) = a(x)$, $r(x) = 0$, $s(x) = 1$, $\delta = -1$
2. for $i = 1$ to $2m$
3. if $v_0 = 1$ then
4. if $\delta < 0$ then
5. $(u(x), v(x), r(x), s(x)) \leftarrow (v(x), u(x) + v(x), s(x), r(x) + s(x))$
6. $, \delta = -\delta$
7. else
8. $v(x) = v(x) + u(x)$, $s(x) = s(x) + r(x)$
9. end if
10. $v(x) = \frac{v(x)}{x}$, $s(x) = \frac{s(x) + s_0 . p(x)}{x}$, $\delta = \delta - 1$
11. end for
12. return $r(x)$

ب) روش آرایه‌ای ضربانی برونر

این روش اولین بار توسط برونر[2] و همکارانش در [۱۴] مطرح شد. مقاله‌های [۱۵] و [۱۶] پیاده‌سازی‌های خوبی از این روش به صورت ساختار آرایه‌ای ضربانی ارائه داده‌اند و نتایجی مشابه به روش اِستین[3] دارند. البته این روش برخلاف روش اِستین، الگوریتمی بر مبنای جابه‌جایی به چپ دارد و در نتیجه برای عمل تقسیم کارایی خوبی ندارد.

حلقه اصلی این الگوریتم مشابه با روش اِستین $2m$ بار تکرار می‌شود و از نظر ساختار الگوریتم مشابهت بسیاری به آن دارد ولی به جای بیت کم‌ارزش، بیت پرارزش را بررسی می‌کند. بنابراین همان طور که اشاره شد، الگوریتم مذکور بر مبنای جابه‌جایی به چپ است. در کاربردهای زیادی، هدف استفاده از عملیات معکوس در سامانه یا الگوریتم مورد نظر، انجام یک عملیات تقسیم است که این روش، کارایی خوبی برای این کاربردها ندارد.

ج) روش آرایه‌ای ضربانی مونتگمری

الگوریتم مونتگمری برای پیاده‌سازی در میدان دودویی مناسب است اما ثابت‌نبودن زمان اجرا در این الگوریتم سبب می‌شود که قابلیت پیاده‌سازی به صورت آرایه‌ای ضربانی را نداشته باشد. استفاده از متغیر δ باعث می‌شود که دیگر نیازی به مقایسه بزرگ‌ترین درجه دو چندجمله‌ای در این روش وجود نداشته باشد. همچنین نیازی به فاز دوم الگوریتم مونتگمری نیست چون الگوریتم ۵، در فاز اول دقیقاً $2m$ بار اجرا شده و مقدار حاصل در انتهای فاز اول برابر $r(x) = a(x)^{-1} x^{2m}$ است. همان طور که در الگوریتم ۵ مشاهده می‌شود، روال الگوریتم مانند روش برِنت–کونگ عمل می‌کند. این روش نیازمند عمل جابه‌جایی به چپ بوده و همان طور که قبل‌تر مطرح شد برای عمل تقسیم مناسب نیست.

برای مقایسه روش‌هایی که در این دسته هستند باید علاوه بر مواردی که به طور معمول مورد ارزیابی قرار می‌گیرند، بروندهی هر کدام از الگوریتم‌ها نیز لحاظ شود. بروندهی در این ساختار به تعداد عملیات معکوس انجام‌شده در هر سیکل زمانی گفته می‌شود. کلیه الگوریتم‌های معرفی‌شده، یک عملیات معکوس را در یک سیکل زمانی انجام می‌دهند، یعنی بروندهی آنها برابر یک است. پیچیدگی زمانی نیز به دلیل استفاده از δ برای هر سه الگوریتم به صورت $O(1)$ است.

1. Reduction
2. Brunner
3. Stein

الگوریتم ۵: محاسبه معکوس مونتگمری به روش برِنت- کونگ روی میدان‌های دودویی [۷]

Input: $a(x), p(x)\ ,\ a \neq 0$

Output: $r(x) = a^{-1}(x)x^{2m} \bmod p(x)$

1. $u(x) = p(x)\ ,\ v(x) = a(x)\ ,\ r(x) = 0\ ,\ s(x) = 1\ ,\ \delta = -1$
2. for $i = 1$ to $2m$
3. if $u_0 = 1$ then $u(x) = \frac{u(x)}{x}\ ,\ s(x) = x.s(x)\ ,\ \delta = \delta + 1$
4. else if $v_0 = 0$ then $v(x) = \frac{v(x)}{x}\ ,\ r(x) = x.r(x)\ ,\ \delta = \delta - 1$
5. else if $\delta < 0$ then
6. $u(x) = \frac{u(x) + v(x)}{x}\ ,\ r(x) = r(x) + s(x)$
7. $s(x) = x.s(x)\ ,\ \delta = \delta + 1$
8. else $v(x) = \frac{v(x) + u(x)}{x}\ ,\ s(x) = s(x) + r(x)\ ,\ r(x) = x.r(x)\ ,\ \delta = \delta - 1$
9. end if
10. end for
11. return $r(x)$

معماری آرایه ضربانی به روش مونتگمری و اِستین، به دلیل نیاز کمتر به ثبات، تأخیر بهتری نسبت به الگوریتم برونر دارند و همچنین الگوریتم اِستین به تعداد کمتری فیلیپ‌فلاپ نیاز دارد. اگر هدف استفاده از عملیات معکوس، انجام عملیات تقسیم باشد، روش اِستین به دلیل استفاده از عملیات جابه‌جایی به راست بهترین کارایی را دارد. به طور کلی معماری هر سه دسته متشکل از $2m$ سطر و در هر سطر m سلول است. بنابراین پیچیدگی مساحت آن برابر $O(m^2)$ خواهد بود که این مساحت بسیار زیاد تنها در کاربردهایی قابل چشم‌پوشی است که بروندهی معکوس‌کننده بسیار مهم باشد. در واقع روش‌های مبتنی بر اقلیدسی تعمیم‌یافته، به علت سرعت بالا مورد توجه هستند که مشکل ثابت‌نبودن زمان با ورودی‌های مختلف از یک میدان ثابت را دارند. معماری آرایه‌ای ضربانی این مشکل را حل کرده است ولی باعث کاهش این امتیاز بزرگ یعنی سرعت بالای این روش‌ها شده‌اند.

۲-۲ مروری بر جدیدترین معکوس‌کننده‌های موجود مبتنی بر الگوریتم اقلیدسی تعمیم‌یافته

در [۱۷] یک معماری جدید برای محاسبه معکوس در میدان دودویی به کمک الگوریتم تعمیم‌یافته اقلیدسی معرفی شده که تمرکز اصلی آن روی کشف خطا است. در این مقاله یک معماری قابل اعتماد کشف خطای همروند (CED)[1] ارائه شده است. در [۱۸] یک روش جدید ارائه شده که پیچیدگی ساختاری کمی دارد و برای میدان‌های دودویی با اندازه میدان کوچک بسیار مناسب است.

نویسندگان در [۱۹] یک معماری آرایه‌ای ضربانی برای محاسبه معکوس در میدان دودویی معرفی کرده‌اند. آنها در این مقاله ابتدا یک روش محاسبه معکوس برای پیاده‌سازی سخت‌افزاری بر مبنای الگوریتم اقلیدسی یافته ارائه داده‌اند و سپس در ادامه یک ساختار بیت- سریال موازی برای آن طراحی کرده‌اند. این ساختار از $2m-1$ سلول موازی تشکیل شده که هر کدام از این سلول‌ها شامل زیرسلول‌های دیگری است. گذردهی این روش $1/m$ و پیچیدگی زمانی آن $5m-3$ سیکل کلاک می‌باشد. همین گروه تحقیقاتی را در [۲۰] منتشر کرده‌اند که یک معماری آرایه‌ای ضربانی جدید را بر اساس ساختار رقم- سریال معرفی می‌کند.

در [۲۱] مشابه با [۱۹] یک معماری آرایه‌ای ضربانی بیت- سریال یک‌طرفه طراحی شده است. این روش که بر پایه روش اقلیدسی بیان گردیده است با بهره‌بردن از یک شمارنده حلقوی[2] موفق به کاهش مساحت معکوس‌کننده شده است.

در [۱۵] با تغییر در ساختار روش برونر، با به تعویق انداختن عملیات $u(x)/x$ یک معماری آرایه‌ای ضربانی بیت- سریالی طراحی شده است. این مقاله به علت استفاده از ساختار غیر یکنواخت در سلول‌های معماری و جمع/تفریق‌کننده مبتنی بر سامانه مکمل دو، پیچیدگی مساحتی برابر $O(m\log_2^m)$ دارد که در [۲۲] با استفاده از تغییر در ساختار کنترلی معماری سیستولیک به وسیله توزیع همگون درجه‌ها پیچیدگی مساحت را به $O(m)$ کاهش داده است.

در [۲۳] دو الگوریتم، یکی بر پایه جابه‌جایی به راست و دیگری بر مبنای جابه‌جایی به چپ ارائه شده است. الگوریتم مبتنی بر جابه‌جایی به راست از هیچ عملیات پیمانه‌ای استفاده نمی‌کند. این الگوریتم با کمک یک شمارنده چرخشی، تأخیر مسیر بحرانی را کم کرده و سرعت بیشتری نسبت به روش دوم دارد. روش دوم نیز از یک جمع‌کننده رقم نقلی موجی[3] استفاده می‌کند که مساحت کمتری نسبت به الگوریتم اول دارد. این روش برای کاربردهایی با محدودیت مساحت و توان مصرفی مناسب است.

نویسندگان [۲۴] یک تکنیک سخت‌افزاری در میدان دودویی بر مبنای الگوریتم تعمیم‌یافته اقلیدسی برای محاسبه عملیات معکوس و تقسیم ارائه کردند. این الگوریتم عملیات کاهش پیمانه‌ای را به صورت موازی با حلقه اصلی انجام می‌دهد. در نتیجه تعداد سیکل‌های کلاک مورد نیاز از $2m-1$ به m کاهش یافته است.

در [۲۵] همانند روش دوم موجود در [۲۳]، روشی برای کاربردهایی با محدودیت مساحت و توان مصرفی ارائه شده است. این مقاله نیز همانند سایر مقاله‌های موجود در این حوزه مبتنی بر الگوریتم تعمیم‌یافته اقلیدسی است. اگرچه این مقاله الگوریتم جدیدی ارائه نداده است اما یک ساختار آرایه‌ای ضربانی جدید در آن معرفی شده که دارای پیچیدگی مساحت $O(m)$ می‌باشد. در این مقاله، ابتدا الگوریتم موجود تبدیل به یک الگوریتم تکراری منظم[4] می‌شود و سپس به کمک گراف وابستگی داده و یک تابع زمان‌بندی وابسته، عملیات معکوس انجام می‌شود. این ایده کمک می‌کند که ساختار آرایه‌ای ضربانی به صورت ساده‌تر و کم‌هزینه‌تر پیاده شود.

۲-۳ روش مبتنی بر قضیه کوچک فِرما در پایه نرمال

همان طور که در (۴) بیان شد می‌توان معکوس یک عدد را به کمک قضیه کوچک فِرما محاسبه کرد. بنابراین روش عملیات معکوس در میدان دودویی در واقع فاکتورگیری و تجزیه مطابق (۸) است. در (۸)، $m-2$ ضرب و $m-1$ مجذور مورد نیاز است [۲۶]. در واقع هر جمع در توان، معادل یک ضرب بین دو عدد است. این تعداد عملیات ضرب و مجذور به زمان زیادی نیاز خواهند داشت و بنابراین با ارائه یک تجزیه بهتر برای جمع فوق، می‌توان تعداد ضرب‌های مورد نیاز را کاهش داد

1. Reliable Concurrent Error Detection
2. Ring-Counter
3. Ripple-Carry
4. Regular Iterative Algorithm

جدول ۱: محاسبه معکوس در میدان $GF(2^{11})$ با روش ITA.

	$\beta_{v_i}(a)$	$\beta_{v_i+u_k}(a)$	تشریح
۱	$\beta_1(a)$	–	a
۲	$\beta_2(a)$	$\beta_{1+1}(a)$	$(\beta_1)^{2^1}\times\beta_1=a^{2^2-1}$
۳	$\beta_4(a)$	$\beta_{2+2}(a)$	$(\beta_2)^{2^2}\times\beta_2=a^{2^4-1}$
۴	$\beta_8(a)$	$\beta_{4+4}(a)$	$(\beta_4)^{2^4}\times\beta_4=a^{2^8-1}$
۵	$\beta_{10}(a)$	$\beta_{8+2}(a)$	$(\beta_8)^{2^2}\times\beta_2=a^{2^{10}-1}$
			$a^{-1}=\beta_{10}^{\,2}=a^{2^{11}-2}$

$$A^{-1}=A^{2^m-2}=A^{2(2^{m-1}-1)}=A^{\overbrace{11\ldots1}^{m-1}}$$
$$A^{2(1+2+\ldots+2^{m-2})}=B^{\sum_{i=0}^{m-2}2^i}=B\times B^2\times\ldots\times B^{2^{m-2}} \qquad (8)$$
$$B=A^2$$

در سال ۱۹۸۸ میلادی آقایان Itoh و Tsujii [۲۷] الگوریتم جدید (ITA) مبتنی بر قضیه کوچک فِرما را برای محاسبه معکوس ارائه دادند. الگوریتم ۶ این روش را نشان می‌دهد که مطابق (۹) عمل می‌کند. برای محاسبه عملیات معکوس به روش ITA به ازای هر عضو $GF(2^m)$، یک زنجیر جمعی برای $m-1$ نوشته و طبق (۹) با استفاده از اعضای زنجیره جمعی، می‌توان معکوس را محاسبه کرد

$$\forall k\in N,\ \beta_k(a)=a^{2^k-1}\rightarrow\beta_{j+k}=(\beta_j)^{2^k}.\beta_k=$$
$$(\beta_k)^{2^j}.\beta_j \qquad (9)$$
$$\text{then } a^{-1}=[\beta_{m-1}(a)]^2$$

الگوریتم ۶: محاسبه معکوس به روش ITA کلاسیک در میدان دودویی [۲۷]

Input: $a\in GF(2^m)$, $a\neq 0$, $m-1=\langle m_{q-1}\ldots m_0\rangle_2$

Output: $\beta=a^{-1}\in GF(2^m)$

1. for $i=q-2$ downto 0
2. $\beta=\beta.(\beta)^{2^t}$
3. $t=2t$
4. if $m_i=1$ then
5. $\beta=\alpha.\beta^2$
6. $t=t+1$
7. end if
8. end for
9. $\beta=\beta^2$
10. return β

مثلاً می‌توان زنجیره جمعی $m-1=10=\{1,2,4,8,10\}$ را برای $a\in GF(2^{11})$ ساخت و مطابق جدول ۱، عملیات معکوس را تنها با استفاده از ۴ عملیات ضرب انجام داد. در واقع به تعداد هر بیت در مبنای دو یک ضرب داریم و به ازای بیت‌های یک، یک ضرب اضافه‌تر خواهیم داشت.

به دلیل مزیت زنجیره جمعی بَرِر در پیاده‌سازی سخت‌افزاری، غالباً از این زنجیره جمعی استفاده می‌شود. از آنجا که همه پژوهش‌های انجام‌شده مبتنی بر این روش، دنباله‌روی این مقاله هستند روش معکوس مبتنی بر قضیه کوچک فِرما، الگوریتم IT نیز نامیده می‌شود.

در واقع هر روشی که موفق به یافتن تجزیه بهتری برای رابطه $1+2+\ldots+2^{m-2}$ شود، می‌تواند عملیات معکوس به روش ITA را بهبود دهد. در همه روش‌های مبتنی بر تجزیه فوق، زنجیره جمعی را برای عدد $m-1$ تشکیل می‌دهند و طبق روابط معرفی‌شده، زنجیره جمعی را محاسبه می‌کنند. در [۲۸] روش TYT[1] با همین هدف ارائه شد. ایده اصلی این مقاله، تجزیه $m-1$ به چند عامل به همراه یک باقیمانده کوچک بود ولی این روش برای میدان‌های استاندارد، کاهش تعداد ضرب را به دنبال نداشت. در [۲۹] چند مورد از این تجزیه‌ها معرفی شده‌اند که از تعداد ضرب‌های لازم برای محاسبه معکوس کاسته است. الگوریتم DB که در همین مقاله ارائه شده، مبتنی بر (۱۰) است

$$1+2+2^2+2^3+\ldots+2^{m-2}=$$
$$\begin{cases}(1+2^1+2^2)(1+2^3+2^6+\ldots+2^{m-4}) & m-1\overset{3}{\equiv}0\\ (1+2)(1+2^2+2^4+\ldots+2^{m-3}) & m-1\overset{3}{\equiv}2\\ 1+2(1+2)(1+2^2+2^4+\ldots+2^{m-4}) & m-1\overset{3}{\equiv}1\end{cases} \qquad (10)$$

برای پیاده‌سازی بهتر عملیات معکوس علاوه بر زنجیره جمعی بهینه، به واحدهای ضرب‌کننده با سرعت و کارایی بالا نیاز است. برای انجام عملیات ضرب $A\times B\times C$ با استفاده از ضرب‌کننده دو ورودی به دو عملیات ضرب نیاز است. در [۳۰] ضرب‌کننده سه ورودی جدیدی در پایه نرمال معرفی شده که حاصل‌ضرب $A\times B\times C$ را با تأخیری تقریباً برابر با یک ضرب دو ورودی انجام می‌دهد. ضرب‌کننده معرفی‌شده از دو طبقه تشکیل شده است. در طبقه اول، ورودی به صورت موازی وارد و خروجی به صورت سریال تولید (PISO)[2] می‌شود. در طبقه دوم، عملیات ضرب بر روی ورودی‌های سریالی انجام و خروجی به صورت موازی تولید می‌شود (SIPO)[3]. این ضرب‌کننده مختلط دوگانه (HD)[4] در پژوهش‌های [۳۱] تا [۳۳] نیز مورد استفاده قرار گرفته است. زنجیره سه‌تایی جمعی به زنجیره‌ای گفته می‌شود که در آن هر عدد از جمع سه عدد قبلی به دست آمده باشد

$$1+2^2+2^3+\cdots+2^{m-2}=$$
$$\begin{cases}(1+2^n+2^{2n})\times(1+2^{3n}+2^{6n}+\cdots+2^{(k-1)n}) & k-1\overset{3}{\equiv}0\\ 1+2\times(1+2^n+2^{2n})\times(1+2^{3n}+2^{6n}+\cdots+2^{(k-5)n}) & k-1\overset{3}{\equiv}1\\ 1+2^n+2^{2n}\times(1+2^n+2^{2n})\times & \\ \quad(1+2^{3n}+2^{6n}+\cdots+2^{(k-6)n}) & k-1\overset{3}{\equiv}2\end{cases} \qquad (11)$$

در [۳۱] با استفاده از زنجیره جمعی IT سه‌تایی (TIT)[5] و ضرب‌کننده مختلط دوگانه تعداد ضرب‌های مورد نیاز برای عملیات معکوس کاهش یافته است. زنجیره جمعی مورد نظر با استفاده از (۱۱) قابل محاسبه است. $\left\lfloor\log_2^{m-1}\right\rfloor+H_2(m-1)+c-1$، تعداد ضرب‌های مورد نیاز در روش مطرح‌شده است که H_2 وزن همینگ[6] عدد در مبنای سه می‌باشد. در (۱۱) اگر بزرگ‌ترین رقم عدد مورد نظر در نمایش مبنای سه آن برابر یک باشد، آن گاه $c=0$ است. برای مثال، روش فوق تعداد ضرب‌های مورد نیاز برای محاسبه معکوس در میدان $GF(2^{163})$ را مطابق (۱۲) به ۵ ضرب کاهش می‌دهد

1. Takagi, Yoshiki, and Takagi
2. Parallel In Serial Out
3. Serial In Parallel Out
4. Hybrid Double
5. Ternary Itoh-Tsujii Algorithm
6. Hamming

$$\sum_{i=0}^{m-1} 2^i = 1+2+2^2+2^3+\ldots+2^{161} = (1+2+2^2)\times (1+2^3+2^6)(1+2^9+2^{18})(1+2^{27}+2^{54})(1+2^{81}) \quad (12)$$

مشابه با روش موجود در پژوهش [۳۱]، در [۳۲] نیز به کمک روابط ریاضی، مطابق با (۱۳) روشی ارائه شده که نیازی به فرایند تجزیه ندارد. به همین دلیل این روش برای پیاده‌سازی سخت‌افزاری مناسب‌تر است و به آن ITA سه‌تایی[1] گفته می‌شود

$$\begin{aligned} &\forall k_1,k_2,k_3 \in N \text{ if } \alpha \in GF(2^n) \\ &\text{where } I_{k_1}(\alpha)=\alpha^{2^{k_1}-1},\ I_{k_2}(\alpha)=\alpha^{2^{k_2}-1}, \\ &\quad I_{k_3}(\alpha)=\alpha^{2^{k_3}-1} \\ &\text{then } I_{k_1+k_2+k_3}(\alpha)=I_{k_1}(\alpha).[I_{k_2}(\alpha)]^{2^{k_1}}.[I_{k_3}(\alpha)]^{2^{k_1+k_2}} \end{aligned} \quad (13)$$

در [۳۳] نیز مطالعه جامعی روی زنجیره‌های جمعی صورت گرفته و روشی برای یافتن کوتاه‌ترین زنجیره جمعی ارائه شده است. در این پژوهش روشی با نام زنجیره جمعی بهینه (OAC)[2] ارائه شده که با استفاده از ضرب‌کننده‌های دوتایی نیز قابل پیاده‌سازی است و نیاز به ضرب‌کننده مختلط دوگانه ندارد. در این مقاله نشان داده شده است که به ازای هر زنجیره جمعی سه‌تایی می‌توان با استفاده از (۱۴) عملیات معکوس را انجام داد

$$\begin{aligned} &\text{if } V(i)=V(i_0)+V(i_1)+V(i_2) \text{ and } V_{i_j}=B^{2^{v(i_j)}-1} \\ &\text{then } V_i=V_{i_0}\times V_{i_1}^{2^{v(i_0)}}\times V_{i_2}^{2^{v(i_0)+v(i_1)}}=B^{2^{v(i)}-1} \end{aligned} \quad (14)$$

دسته دیگری از مقالات بر روی موازی‌سازی بیشتر عملیات معکوس تمرکز دارند. مقاله [۳۴] نشان داده که یک زنجیره جمعی دودویی می‌تواند به یک روش موازی با عمق $s=\lceil \log_2^m \rceil$ تبدیل شود. عملیات معکوس در این روش توسط دو ضرب‌کننده موازی قابل انجام خواهد بود. در روش ITA پایه در حالتی که بیت مورد نظر یک باشد، باید اعمال مشخصی به صورت ترتیبی انجام شود. مقاله [۳۲] این وابستگی داده‌ای را کاهش داده است. با این روش و به کمک دو ضرب‌کننده میدان متناهی می‌توان عملیات معکوس را انجام داد. در [۳۳] نیز با استفاده از دو ضرب‌کننده مختلط دوگانه عملیات معکوس را به صورت موازی انجام داده است. بنابراین مقاله به ازای $k=3$ می‌توان از ضرب‌کننده مختلط دوگانه استفاده کرد. برای $m-1=\langle n_{q-1},\ldots,n_1,n_0\rangle_2$ دو زنجیره جمعی است که ضرب‌کننده مختلط دوگانه اول مقادیر $V_0(i)=3^i, 0\le i\le \lceil \log_2^n \rceil$ را محاسبه می‌کند و ضرب‌کننده مختلط دوگانه دوم از حاصل این ضرب‌کننده استفاده می‌کند و طبق رابطه زیر معکوس را محاسبه می‌نماید

$$V_1(i)=v_1(i-1)+n_{i-1}v_0(i-1),\ 1\le i\le [\log_2^n] \quad (15)$$

برای مثال عملیات معکوس در میدان $GF(2^{163})$ با استفاده از این روش و با توجه به این که $162=(2,0,0,0,0)_3$ است به صورت زیر قابل انجام خواهد بود. سطر اول در این ماتریس مربوط به ضرب‌کننده اول و سطر دوم مربوط به ضرب‌کننده دوم خواهد بود

$$v=\begin{pmatrix} 1 & 3 & 9 & 27 & 81 & \\ 0 & 0 & 0 & 0 & 162 & \end{pmatrix}_2 \quad (16)$$

در [۳۵] یک ضرب‌کننده مختلط دوگانه در پایه نرمال گوسی طراحی شده که برای پیاده‌سازی الگوریتم ITA مورد استفاده قرار گرفته و سبب بهبود آن شده است. در این پژوهش همچنین یک زنجیره جدید معرفی گردیده که توانسته تعداد سیکل‌های کلاک را با سربار کمی کاهش دهد.

۲-۴ روش مبتنی بر قضیه کوچک فِرما در پایه چندجمله‌ای

الگوریتم IT در نسخه اصلی آن [۲۷] برای پایه نرمال ارائه شده بود اما بعدها در [۳۶] برای پایه چندجمله‌ای نیز معرفی و پیاده‌سازی شد. در [۳۷] از زنجیره جمعی و مجذور پی‌درپی استفاده شده تا روش ITA بهبود داده شود. مجذورهای پی‌درپی باعث کاهش تعداد سیکل‌های کلاک مورد نیاز برای انجام مجذورها می‌شود. در [۳۷] تلاش‌هایی برای موازی‌سازی روش قبل انجام شده است. نویسندگان این مقاله مطابق (۱۷) ثابت کرده‌اند که با استفاده از جذرهای متوالی به جای مجذورهای متوالی نیز می‌توان معکوس یک عنصر را محاسبه کرد

$$\begin{aligned} &\forall k\in N: \gamma(a)=a^{1-2^k} \rightarrow \gamma_{k+j}(a)=\gamma_k^{2^{-j}}(a).\gamma_j(a) \\ &\text{then } a^{-1}=[\gamma_{m-1}(a)]^2 \end{aligned} \quad (17)$$

در این روش به کمک مجذور و جذر به طور موازی، طول زنجیره جمعی و به دنبال آن تعداد سیکل‌های کلاک مورد نیاز برای محاسبه معکوس کاهش می‌یابد. در واقع این روش طول زنجیره را نصف می‌کند. سپس نیمی از آن را به کمک مجذورهای متوالی و نیمی دیگر را با جذرهای متوالی محاسبه می‌کند. در [۳۸] به جای استفاده از مجذور، از توان چهار[3] استفاده شده است. در پیاده‌سازی عملیات معکوس روی بستر FPGA، تعداد LUT لازم برای انجام عملیات توان چهار رسانی ۱/۵ برابر تعداد LUT لازم برای انجام عملیات مجذور است. این روش به دلیل استفاده بهتر از LUTها، سبب بهبود سرعت و مساحت عملیات معکوس در پیاده‌سازی سخت‌افزاری آن شده است. در روش قبل، پیش از حلقه اصلی الگوریتم، برای توان سه رسانی ورودی، به یک ضرب نیاز بود که در [۳۹] این مشکل بر طرف شده است. مقاله [۴۰] نیز روش موجود در [۳۸] را به جای توان چهار، به ازای هر توانی از دو تعمیم داده است.

مقاله [۴۱] روش موازی‌سازی موجود در [۴۲] و توان چهار رسانی به کار رفته در [۳۸] را با هم ترکیب کرده تا از مزایای هر دو استفاده کند. در نتیجه تعداد سیکل‌های کلاک مورد نیاز در این روش از دو روش قبل کمتر است. این مقاله این روش را تعمیم داده و به جای محاسبه 2^2 و 2^{-2} از 2^n و 2^{-n} استفاده کرده تا طول زنجیره جمعی را کمتر و به دنبال آن تعداد سیکل‌های کلاک مورد نیاز را کاهش دهد. در واقع این روش به ازای یک مقدار مشخص برای n زنجیره جمعی را برای $\lfloor m-1/2n \rfloor$ به دست می‌آورد. معماری ارائه‌شده در این مقاله از یک واحد جذر، یک واحد مجذور و یک واحد ضرب‌کننده میدان متناهی تشکیل شده است.

یکی از چالش‌های پیش رو برای انجام عملیات معکوس در پایه چندجمله‌ای کاهش تعداد سیکل‌های کلاک مورد نیاز برای انجام مجذور است. در پایه نرمال مجذورهای متوالی هزینه‌ای ندارند اما در پایه چندجمله‌ای هزینه این عمل با توجه به تعداد بالای آن، بسیار قابل توجه خواهد بود. برای انجام عملیات معکوس در پایه چندجمله‌ای با استفاده از یک واحد مجذور، باید در هر مرحله به تعداد u_i-u_{i-1} سیکل کلاک

1. Ternary-ITA
2. Optimal Addition Chain
3. Quad

زمان صرف کرد. یکی از راه‌های سرعت‌بخشیدن به این مجذورها، در [۳۷] معرفی شده که به آن مجذور نردبانی گفته می‌شود. در این روش یک واحد مجذور وجود دارد که می‌تواند تعداد u_s مجذور را در یک سیکل کلاک انجام دهد. در نتیجه برای محاسبه $\beta_p^{۲^q}$ که $q = u_i - u_{i-۱}$ است در هر مرحله به تعداد $\lceil (u_i - u_{i-۱})/u_s \rceil$ سیکل کلاک نیاز است. تعداد کل سیکل‌های کلاک مورد نیاز برای انجام عملیات معکوس در این روش برابر $\#C = l + \sum_{i=۲}^{l} \lceil (u_i - u_{i-۱})/e_s \rceil$ است.

در [۴۳] سه روش برای پیاده‌سازی ITA معرفی شده که عملیات معکوس را با کمترین تعداد سیکل کلاک مورد نیاز انجام می‌دهد. یافتن کوتاه‌ترین زنجیره جمعی جزو مسایل NP-hard است اما در صورت مشخص‌بودن سایز میدان می‌توان این مسأله را با الگوریتم‌هایی نظیر الگوریتم معرفی‌شده در [۳] حل کرد. حال از بین کوتاه‌ترین زنجیره پیداشده، مناسب‌ترین زنجیره با توجه به شرایط مورد نظر انتخاب می‌شود.

مطابق روش ITA پایه برای هر مرحله به یک ضرب و u_j مجذور نیاز است. همچنین در صورت یک‌بودن بیت مربوط، به یک ضرب و یک مجذور دیگر نیز نیاز خواهد بود. این مراحل به صورت متوالی انجام می‌شوند اما در [۴۴] روش دیگری برای ITA ارائه شده که قابلیت موازی‌سازی بیشتری دارد.

طبق (۱۸)، زنجیره جمعی نیز به شکل دیگری رشد خواهد کرد. از آنجایی که رابطه $I_{u_j}^{۲^{u_i+u_j}} = (I_{u_j}^{۲^{u_i}})^{۲^{u_j}}$ برقرار است، پس در هنگام یک‌بودن بیت فقط به u_j بار مجذورکردن $I_{u_j}^{۲^{u_i}}$ نیاز دارد. مجذورهای فوق مستقل از $I_{u_{j+۱}}$ بوده و می‌تواند موازی با آن محاسبه شود. در مرحله $j+۱$، به تعداد $\min(C_m, \lceil u_j/e_N \rceil)$ سیکل کلاک همپوشانی خواهد داشت که می‌توان برای هر مرحله تعداد سیکل‌های کلاک همپوشان را محاسبه کرد و تعداد کل سیکل کلاک مورد نیاز برای این روش را به دست آورد. با توجه به رابطه فوق باید در بین کوتاه‌ترین زنجیره‌های جمعی موجود برای یک میدان، زنجیره‌ای انتخاب شود که به کمترین تعداد سیکل کلاک نیاز داشته باشد (OAC). تعداد ضرب و توان‌رسانی مورد نیاز در این روش کاهش نیافته است اما این روش توانسته تعداد سیکل کلاک مورد نیاز و زمان کل اجرا را با ایده فوق کاهش دهد

$$m-۱ = \langle n_{q-۱} \cdots n_۱ n_۰ \rangle_۲ \rightarrow \text{if } (m_i = ۱) \text{ then}$$
$$I_{u_j+u_۰} = I_{u_۰} . I_{u_j}^{۲^{u_۰}} \ , \ I_{۲u_j+u_۰} = I_{u_۰+u_j} . I_{u_j}^{۲^{u_۰+u_j}} \qquad (۱۸)$$

مقاله [۴۵] به بررسی پیاده‌سازی‌های نرم‌افزاری معکوس بر روی میدان دودویی پرداخته و بیان می‌کند که در معماری‌های مدرن روش اقلیدسی تعمیم‌یافته به دلیل تعداد زیاد عملیات پرش و جابه‌جایی قابل رقابت با روش مبتنی بر قضیه کوچک فِرما نیست. روش‌های مبتنی بر الگوریتم اقلیدسی تعمیم‌یافته بیشتر در پیاده‌سازی‌های نرم‌افزاری مورد استفاده قرار می‌گیرند. در حالی که روش‌های مبتنی بر قضیه کوچک فِرما برای پیاده‌سازی‌های سخت‌افزاری مناسب‌تر هستند [۴۶]. واحدهای مورد نیاز برای پیاده‌سازی روش‌های مبتنی بر قضیه کوچک فِرما، واحد ضرب‌کننده و مجذور میدان متناهی می‌باشند. با توجه به کاربرد عملیات معکوس در پردازنده رمزنگاری خم بیضوی، عملیات معکوس مبتنی بر قضیه کوچک فِرما در این کاربردها به سخت‌افزار اضافه‌ای نیاز نخواهد داشت [۴۷].

لازم به ذکر است که مقالات جدیدتری نظیر [۴۸] نیز وجود دارند که الگوریتم جدیدی برای محاسبه معکوس ارائه نداده‌اند و در آنها تنها معماری و بستر مورد استفاده تغییر یافته است. با توجه به این که هدف این مقاله معرفی روش‌های موجود از نقطه نظر الگوریتم مورد استفاده است، از ذکر جزئیات بیشتر در مورد این روش‌ها خودداری شده است.

۳- تحلیل پیچیدگی عملیات معکوس در میدان دودویی

در این بخش سربار عملیاتی واحد معکوس‌کننده بر حسب نوع الگوریتم به کار رفته مورد بررسی قرار می‌گیرد. در ابتدا به الگوریتم معکوس اقلیدسی تعمیم‌یافته پرداخته شده است. همان طور که پیش‌تر بیان شد، الگوریتم اقلیدسی تعمیم‌یافته مبتنی بر یافتن بزرگ‌ترین مقسوم‌علیه مشترک است. در روند محاسبه بزرگ‌ترین مقسوم‌علیه مشترک در هر مرحله رابطه $n_{j-۱} = q_{j-۱} \times n_j + n_{j+۱}$ برقرار است که خارج قسمت مراحل بعدی $(n_{j+۱})$، باقیمانده تقسیم خارج قسمت‌های دو مرحله قبل است. در روند محاسبه بزرگ‌ترین مقسوم‌علیه مشترک دو عدد همواره (۱۹) بین دو مقدار متوالی تقسیم m/n برقرار است

$$\frac{n_j}{n_{j+۱}} = q_j + \frac{۱}{\frac{n_j+۱}{n_j+۲}} \Rightarrow \frac{m}{n} = q_۰ + \frac{۱}{q_۱ + \frac{۱}{q_۲ + \frac{۱}{q_k}}} \qquad (۱۹)$$

در حالتی که همه q_iهای موجود در (۸) برابر با یک باشند، الگوریتم دارای بیشترین زمان اجرا خواهد بود. در این صورت مقادیر متوالی خارج قسمت در این روش یک دنباله فیبوناچی[1] را تشکیل خواهند داد. برای مثال در حین محاسبه بزرگ‌ترین مقسوم‌علیه مشترک عدد ۸ و ۵ دنباله زیر تشکیل می‌شود

$$GCD(۸,۵) = GCD(۵,۳) = GCD(۳,۲) = GCD(۳,۲) = GCD(۲,۱) \qquad (۲۰)$$

بنابراین بدترین حالت اجرای الگوریتم اقلیدسی تعمیم‌یافته طبق (۲۱) برابر با $\mathrm{O}(m)$ خواهد بود

$$K = \mathrm{O}(\log_\rho^P) \rightarrow \rho = \frac{\sqrt{۵}+۱}{۲} \ , \quad P = ۲^m \Rightarrow K = \mathrm{O}(m \log_\rho^۲) = \mathrm{O}(m) \qquad (۲۱)$$

در الگوریتم اقلیدسی تعمیم‌یافته دودویی برای میدان دودویی، زمان اجرای الگوریتم در حدود زمان اجرای الگوریتم اقلیدسی تعمیم‌یافته در حالت پایه است. خانم ولی در [۴۹] رابطه‌ای بین الگوریتم فوق و الگوریتم یافتن بزرگ‌ترین مقسوم‌علیه مشترک به روش تقسیم‌های متوالی پیدا کرده است. مطابق (۲۲) بدترین زمان اجرای الگوریتم اقلیدسی تعمیم‌یافته دودویی برای میدان دودویی برابر $\log_۲^P$ می‌باشد که در آن $P = ۲^m$ است

$$a_j \text{ is odd }, ۰ < a_j < ۲^{k_j} \Rightarrow \frac{m}{n} = \frac{۱}{a_۱} + \frac{۲^{k_۱}}{a_۲} + \cdots + \frac{۲^{k_{r-۱}}}{a_r} + ۲^{k_r} \qquad (۲۲)$$

همچنین خانم ولی در [۵۰] اثبات می‌کند که الگوریتم دودویی یافتن بزرگ‌ترین مقسوم‌علیه مشترک حدوداً ۶۰٪ در سرعت و هزینه عملیاتی از الگوریتم یافتن بزرگ‌ترین مقسوم‌علیه مشترک بهتر عمل می‌کند. همچنین ثابت می‌شود که زمان اجرای فاز اول الگوریتم مونتگمری نیز از (۲۳) قابل محاسبه است

$$\log^P \le k \le ۲\log^P \Rightarrow p = ۲^m \Rightarrow m \le k \le ۲m \qquad (۲۳)$$

در مقالات موجود معمولاً پیچیدگی زمانی و مساحت روش‌های مبتنی بر الگوریتم تعمیم‌یافته اقلیدسی بر حسب تأخیر مسیر بحرانی، تعداد سیکل

1. Fibonacci

جدول ۲: مقایسه پیچیدگی الگوریتم اقلیدسی تعمیم‌یافته در میدان دودویی.

منابع	تعداد NOT	تعداد AND	تعداد XOR	تعداد مالتی‌پلکسر	تعداد سیکل	تأخیر مسیر بحرانی
[۱۵]	$۲m$	$۴m$	$۸۰m$	$۲m$	$۵m-۴$	$۲T_A+T_X+T_{Mux}$
[۲۱]	۲	$۶m$	$۶m$	$۱۸m$	$۵m-۴$	$۲T_A+T_{MUX}$
[۱۹]	$۴m$	$۶m$	$۴m$	$۱۲m$	$۵m-۲$	$۲T_{MUX}$
[۱۷]	۱	$۳m+\log(m+۱)$	$۲m+۳$	$۴m+\log(m+۱)$	$۲m-۱$	$۲T_A+۳T_X$
[۱۸]	۰	$۹m-۳$	$۶m+۳$	$۹m$	$۱/۴۵m$	$۲T_X+۳T_{MUX}$
[۲۴]	۰	$۶m+۹$	$۶m+۳$	$۶m+۴$	m	$۲T_A+۳T_X$
[۲۳]	۱	$۲m+۳$	$۲m+۲$	$۵m+۴$	$۲m-۱$	$۲T_{MUX}$
[۲۳]	۲	$۲m+۲$	$۲m-۲$	$۴m$	$۲m-۱$	$۲T_{MUX}$

جدول ۳: مقایسه پیچیدگی الگوریتم ITA در میدان دودویی.

منابع	دسته‌بندی	تعداد ضرب	تعداد توان‌رسانی	قابلیت موازی‌سازی
[۲۶]	–	$m-۲$	$m-۱$	×
[۲۷]	ITA	$\log_۲(m-۱)+H_۲(m-۱)-۱$	$m-۱$	×
[۳۱] و [۳۲]	ITA-N	$۲(\log_۲(m-۱)+H_۲(m-۱)+c)$	$\le ۳(m-۱)$	×
[۳۳]	ITA-N	$<۲\log_۲(m-۱)+\frac{d_۲(m-۱)}{۲}$	$\le ۳(m-۱)$	×
[۳۲]	ITA-N	$۲\log_۲(m-۱)-۱$	$\le ۲(m-۱)$	✓
[۳۳]	ITA-N	$۴\log_۲(m-۱)$	$\le ۳(m-۱)$	✓
[۳۶] و [۴۲]	ITA-P	$\log_۲(m-۱)+H_۲(m-۱)+۱$	$m-۱$	×
[۳۸]	ITA-P	$\log_۲(\frac{m-۱}{۲})+\frac{H_۲(m-۱)}{۲}-۱$	$\ge m-۱$	×
[۴۲]	ITA-P	$\log_۲(m-۱)+H_۲(m-۱)-۱$	$\ge m-۱$	✓
[۴۱]	ITA-P	$\log_۲(\frac{m-۱}{۲n})+\frac{H_۲(m-۱)}{۲n}-۱$	$\ge m-۱$	✓
[۴۵]	ITA-P	$\log_۲(m-۱)+H_۲(m-۱)-۱$	$m-۱$	✓

کلاک و تعداد گیت‌های مورد نیاز گزارش شده که در جدول ۲ به تفکیک قابل مشاهده است. در این جدول T_X، T_A و T_{MUX} به ترتیب تأخیر گیت XOR، AND و مالتی‌پلکسر و همچنین m مرتبه میدان است. حاصل‌ضرب تعداد سیکل در تأخیر مسیر بحرانی پیچیدگی زمانی الگوریتم مورد نظر را نشان خواهد داد اما فرکانس کاری بر حسب واحد ثانیه و توان مصرفی این روش وابسته به تکنولوژی مورد استفاده بوده و در حالت کلی قابل مقایسه نیست.

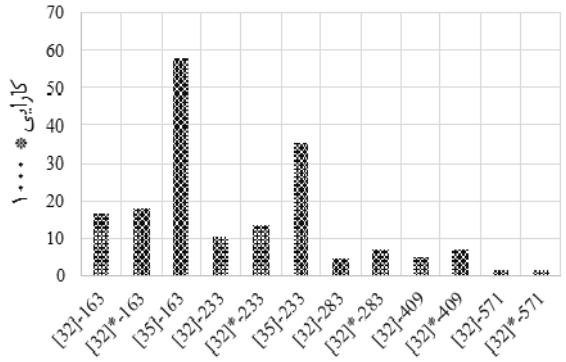
بنابراین در حالت کلی و مطابق جدول ۲، [۲۴] در کمترین تعداد سیکل کلاک عملیات معکوس را انجام می‌دهد. همچنین به دلیل تأخیر مسیر بحرانی کم، مقاله فوق دارای کمترین پیچیدگی زمانی برای محاسبه معکوس می‌باشد. همچنین به نظر می‌رسد که واحد معکوس طراحی‌شده در [۲۵] کمترین پیچیدگی مساحت را دارا است.

همان طور که پیش‌تر به آن پرداخته شد الگوریتم‌های معکوس مبتنی بر قضیه کوچک فرما از تعدادی عملیات ضرب و مجذور میدان متناهی تشکیل شده‌اند. به همین دلیل پیچیدگی عملیاتی این روش‌ها در مقالات موجود بر حسب تعداد عملیات ضرب و مجذور بیان شده است. جدول ۳ پیچیدگی عملیات معکوس مبتنی بر ITA را نشان می‌دهد که $H_۲$ وزن همینگ عدد در مبنای دو، $H_۳$ وزن همینگ عدد در مبنای سه و $d_۳$ جمع ارقام عدد در مبنای سه می‌باشد.

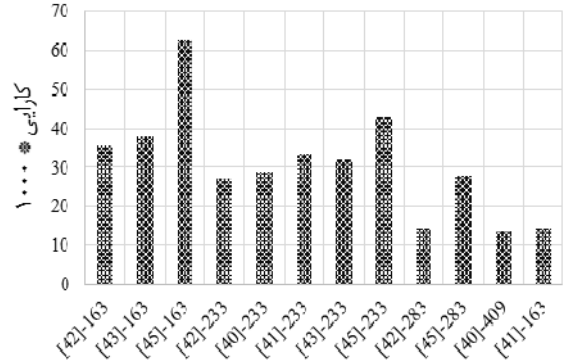
مقاله [۴۳] از لحاظ پیچیدگی بهترین نتایج را در کارهای انجام‌شده روی میدان دودویی در بین روش‌های مبتنی بر ITA دارد. این مقاله با کمک محاسبه معکوس به روش جذر و مجذور همزمان و موازی‌کردن آنها، زنجیره جمعی را با $(m-۱)/۲n$ عنصر تشکیل می‌دهد و به ازای هر m با پیداکردن بهترین n، تعداد سیکل کلاک را نیز به کمترین میزان کاهش می‌دهد. حال اگر فرض کنید که واحد معکوس مبتنی بر ITA شامل یک واحد ضرب‌کننده و یک واحد مجذور میدان متناهی باشد. هر ضرب‌کننده میدان دودویی تک‌مرحله‌ای به $m^۲$ گیت AND و $O(m^۲)$ گیت XOR نیاز دارد و همچنین زمان اجرای آن برابر $T_A+\log ۴m\times T_X$ است. از طرفی هر واحد مجذور تقریباً به $m/۲$ گیت XOR نیاز دارد و تأخیر آن در بدترین حالت $۳T_X$ است. حال برای مقایسه دو روش ITA و تعمیم‌یافته اقلیدسی در میدان دودویی می‌توان بهترین روش موجود در هر کدام را با هم مقایسه کرد. با توجه به فرض صورت‌گرفته واحد طراحی‌شده در [۴۱] نسبت به الگوریتم تعمیم‌یافته اقلیدسی موجود در [۲۴] سربار مساحت و زمان بیشتری خواهد داشت.

برای مقایسه روش‌های مبتنی بر الگوریتم تعمیم‌یافته اقلیدسی و مبتنی بر ITA باید اذعان داشت که زمان اجرای الگوریتم اقلیدسی تعمیم‌یافته به ورودی وابسته است. البته همان طور که اشاره شد، تکنیک‌هایی مانند معماری آرایه‌ای ضربانی برای ثابت‌کردن این زمان وجود دارد اما روش‌های مبتنی بر ITA دارای زمان ثابت بوده و نسبت به حملات کانال جانبی مقاوم‌تر هستند. الگوریتم ITA نیز ماهیتاً سریالی بوده و قابلیت موازی‌سازی چندانی ندارد ولی می‌توان با کمک تکنیک‌هایی مانند موازی‌سازی جذر و مجذور، یا موازی‌سازی ضرب و توان‌رسانی به بهبودهای خوبی رسید. همچنین کارهای انجام‌شده روی زنجیره جمعی توانسته هزینه این روش را به صورت مؤثر کاهش دهد.

البته باید به این نکته توجه داشت که به علت متغیربودن زمان اجرای الگوریتم تعمیم‌یافته اقلیدسی بر حسب ورودی، این روش برای پیاده‌سازی



شکل ۲: مقایسه کارایی واحدهای معکوس پیاده‌شده در پایه نرمال و روی بورد Virtex -۴.



شکل ۳: مقایسه کارایی واحدهای معکوس پیاده‌شده در پایه چندجمله‌ای و روی بورد Virtex -۴.

سخت‌افزاری مناسب نمی‌باشد. به همین علت در تحقیقات انجام‌شده، تغییراتی در آن داده می‌شود تا زمان آن مستقل از ورودی و ثابت شود. مثلاً معماری آرایه‌ای ضربانی که بر پایه الگوریتم تعمیم‌یافته اقلیدسی است دارای زمان ثابت است و زمان اجرای آن از $O(m)$ به مقدار دقیق m می‌رسد. یعنی زمان اجرا از حالتی که می‌تواند عددی در بازه صفر تا m باشد دقیقاً به مقدار m برسد. این نکته در محاسبه پیچیدگی زمانی تأثیر چندانی ندارد ولی در پیاده‌سازی واقعی اهمیت و اثربخشی خود را نشان می‌دهد.

در حالت کلی زمان اجرای عملیات معکوس برای روش موجود در [۲۵] از (۲۴) و زمان کل روش موجود در [۴۳] نیز از (۲۵) حاصل می‌شود. پیچیدگی زمانی روش ITA برابر $O(m)$ و پیچیدگی زمانی الگوریتم تعمیم‌یافته اقلیدسی به روش آرایه‌ای ضربانی برابر $O(\log m)^2$ است و بنابراین روش ITA نسبت به روش‌های تعمیم‌یافته اقلیدسی با زمان ثابت سریع‌تر می‌باشد

$$T_1 = (m)\times(2T_A + 3_X) \approx m \times 4T_x \qquad (۲۴)$$

$$T_2 = (2\log_2^{\frac{m-1}{4}})\times(2T_{mux} + T_{mul}) \leq (\log_2^{\frac{m-1}{4}})\times(6 + \log_2^{4m})T_x \qquad (۲۵)$$

٤- بررسی نتایج پیاده‌سازی عملیات معکوس در میدان دودویی

در این قسمت به بررسی بهترین پیاده‌سازی‌های انجام‌شده عملیات معکوس در میدان دودویی پرداخته می‌شود. در ابتدا نتایج پیاده‌سازی روی FPGA و سپس به صورت ASIC با استفاده از کتابخانه استاندارد ۴۵ و ۶۵ نانومتر بررسی می‌شود.

٤-١ پیاده‌سازی روی بستر FPGA

در این قسمت پیاده‌سازی‌های مختلف موجود بر اساس دستگاه و پایه مورد استفاده تقسیم‌بندی شده است.

متأسفانه نتایج موجود برای پیاده‌سازی روی بستر FPGA تنها منحصر به روش‌های مبتنی بر قضیه کوچک فرما است. جدول ۴ نتایج پیاده‌سازی روش ITA را بر روی دستگاه Virtex -۴ در پایه نرمال نشان می‌دهد. همان طور که مشاهده می‌کنید واحد معکوس طراحی‌شده در [۳۵] روی میدان $GF(2^{163})$ و $GF(2^{233})$ سریع‌ترین واحد معکوس پیاده‌شده در پایه نرمال است. همچنین [۳۲] سریع‌ترین پیاده‌سازی را برای میدان $GF(2^{283})$، $GF(2^{409})$ و $GF(2^{571})$ انجام داده است.

برای مقایسه پیاده‌سازی‌های انجام‌شده روی بستر FPGA و مستقل از اندازه میدان مورد نظر می‌توان از معیاری به نام کارایی استفاده کرد. برای محاسبه کارایی از رابطه $m/(A\times T)$ استفاده می‌شود که در آن m اندازه میدان، A مساحت واحد مورد نظر بر حسب قطعه و T زمان اجرای عملیات معکوس می‌باشد [۵۱]. شکل ۲ کارایی طرح‌های پیاده‌شده روی بستر FPGA و در پایه نرمال را نشان می‌دهد. همان طور که مشاهده می‌شود، واحد معکوس پیاده‌شده در [۳۵] بیشترین کارایی را نسبت به سایر طرح‌های موجود دارد. استفاده از ضرب‌کننده مختلط دوگانه و ارائه یک زنجیره جمعی جدید این افزایش کارایی را به همراه داشته است.

جدول ۵ نتایج پیاده‌سازی عملیات معکوس را در پایه چندجمله‌ای بر روی دستگاه Virtex -۴ نشان می‌دهد. در این پایه روش موجود در [۴۴] سریع‌ترین روش برای میدان‌های استاندارد است.

مقایسه بین دو پایه نرمال و چندجمله‌ای، به علت متفاوت‌بودن ساختار این دو پایه ممکن نمی‌باشد. در پایه نرمال توان‌رسانی هزینه چندانی ندارد ولی در پایه چندجمله‌ای این عمل هزینه زمانی و مساحتی قابل توجهی را به همراه دارد. در مقابل، هزینه ضرب در پایه نرمال تقریباً ۱/۵ برابر هزینه آن در پایه چندجمله‌ای است. بنابراین با توجه به کاربرد و ماهیت سامانه رمزنگاری استفاده‌شده می‌توان بهترین روش را برای هر پایه انتخاب کرد.

با این وجود می‌توان بهترین کار انجام‌شده در هر دو دسته را مقایسه کرد. مقاله [۴۴] بهترین کار در پایه چندجمله‌ای است که الگوریتم خود را با [۳۲] مقایسه کرده است. مقاله [۳۲] در زمره بهترین کارها در پایه نرمال بوده و از دو ضرب‌کننده پایه نرمال به صورت موازی استفاده می‌کند. برای عادلانه‌بودن مقایسه در [۴۴] به جای استفاده از ضرب‌کننده ترکیبی[1] از یک ضرب‌کننده رقم- سریالی استفاده شده است. نتایج نشان می‌دهد که در هر دو عاملِ زمان و مساحت [۴۴] نتایج بهتری را دارا بوده و به طور میانگین ۶۱٪ بهبود زمان و ۶۹٪ بهبود مساحت را به دنبال داشته است.

شکل ۳ کارایی واحدهای معکوس‌کننده موجود را در پایه چندجمله‌ای و روی بستر FPGA نشان می‌دهد. در این پایه اختلاف کارایی واحدهای پیاده‌شده کمتر بوده و واحد معکوس پیاده در [۴۴] بیشترین کارایی را دارد. همچنین با توجه به دو نمودار فوق واحدهای معکوس موجود در پایه نرمال و چندجمله‌ای اختلاف چندانی از نظر کارایی ندارند.

1. Combinational

جدول ۴: نتایج پیاده‌سازی روش ITA در پایه نرمال روی دستگاه VIRTEX-۴.

منابع	الگوریتم	میدان	تعداد کلاک	تأخیر مسیر بحرانی	زمان (ns)	مساحت × 10^3	کارایی × 10^3
[۳۲]	T-ITA	۱۶۳	۳۶	۱۰٫۴۸	۳۷۷٫۲	۲۶	۱۷
[۳۲]	Parallel-ITA	۱۶۳	۵۷	۹۲٫۸	۵۰۸٫۳	۱۸	۱۸
[۳۵]	–	۱۶۳	۸۰	۴٫۳۹	۳۵۱٫۳	۸	۵۸
[۳۲]	T-ITA	۲۳۳	۸۲	۸٫۲۷	۶۶۷٫۹	۳۴	۱۰
[۳۲]	Parallel-ITA	۲۳۳	۶۴	۱۰٫۰۴	۶۴۲٫۲	۲۷	۱۳
[۳۵]	–	۲۳۳	۳۵	۵٫۱	۱۷۸٫۶	۳۷	۳۵
[۳۲]	T-ITA	۲۸۳	۸۱	۱۱٫۵۳	۹۳۴٫۱	۶۵	۵
[۳۲]	Parallel-ITA	۲۸۳	۸۱	۱۱٫۷۳	۹۴۹٫۷	۴۳	۷
[۳۲]	T-ITA	۴۰۹	۱۴۱	۱۲٫۵۲	۱۷۶۵٫۳	۴۵	۵
[۳۲]	Parallel-ITA	۴۰۹	۱۶۱	۱۰٫۶۰	۱۷۰۶٫۹	۳۶	۷
[۳۲]	T-ITA	۵۷۱	۱۹۳	۱۳٫۷۱	۲۶۴۵٫۶	۱۵۲	۱
[۳۲]	Parallel-ITA	۵۷۱	۲۱۷	۱۳٫۰۹	۲۸۴۰٫۳	۹۳	۲

جدول ۵: نتایج پیاده‌سازی روش ITA در پایه چندجمله‌ای روی دستگاه VIRTEX-۴.

منابع	میدان	تعداد کلاک	تأخیر مسیر بحرانی	زمان (ns)	مساحت × 10^3	کارایی × 10^3
[۴۰]	۱۶۳	۲۸	۱۳٫۵۸	۳۸۰٫۳	۱۲	۳۶
[۴۱]	۱۶۳	۲۵	۱۱٫۴	۲۸۵	۱۵	۳۸
[۴۴]	۱۶۳	۲۷	۸٫۷۸	۲۳۷٫۱	۱۱	۶۲
[۴۰]	۲۳۳	۳۰	۱۳٫۵۷	۴۰۷٫۱	۲۱	۲۷
[۳۸]	۲۳۳	۳۰	۱۰٫۳	۳۰۹	۲۶	۲۹
[۳۹]	۲۳۳	۲۹	۱۰٫۰۶	۲۹۰	۲۴	۳۳
[۴۱]	۲۳۳	۲۳	۱۳٫۸	۳۱۷٫۴	۲۳	۳۲
[۴۴]	۲۳۳	۲۶	۱۱٫۰۳	۲۸۷٫۳	۱۹	۴۳
[۴۰]	۲۸۳	۴۳	۱۴٫۸۲	۶۳۷٫۲۶	۳۱	۱۴
[۴۴]	۲۸۳	۳۵	۱۱٫۶۰	۴۰۶	۲۵	۲۸
[۳۸]	۴۰۹	۳۲	۱۵٫۸	۵۰۵٫۶	۶۰	۱۳
[۳۹]	۴۰۹	۳۱	۱۶٫۲۷	۵۰۲٫۲	۵۷	۱۴

جدول ۶: نتایج پیاده‌سازی الگوریتم اقلیدسی تعمیم‌یافته در میدان $GF(2^{233})$ با استفاده از کتابخانه استاندارد ۴۵ نانومتر.

منابع	تعداد کلاک	زمان (ns)	مساحت (Kgates)	انرژی (PJ)	برون‌دهی (m/T)
[۲۱]	۱۱۶۱	۰٫۳۹۹	۱۱٫۰۸	۱٫۴	۵۸۳٫۹
[۱۹]	۱۱۶۳	۰٫۳۹۸	۱۰٫۲۸	۳٫۵	۸۵٫۴
[۱۷]	۴۶۵	۰٫۲۰۰	۴٫۷۳	۱٫۲۲	۱۱۶۵٫۰
[۲۳]	۴۶۵	۰٫۱۵۹	۵٫۵۳	۱٫۰۵	۱۴۶۵٫۴
[۲۵]	۴۶۵	۰٫۱۵۹	۴٫۰۶	۰٫۷۴	۱۴۶۵٫۴

۴-۲ نتایج پیاده‌سازی به صورت ASIC

جدول ۶ نتایج پیاده‌سازی بهترین روش‌های انجام عملیات معکوس مبتنی بر الگوریتم اقلیدسی تعمیم‌یافته را نشان می‌دهد. همان طور که مشاهده می‌کنید، روش موجود در [۲۳] و [۲۵] سریع‌ترین پیاده‌سازی را با استفاده از کتابخانه استاندارد ۴۵ نانومتر ارائه داده که [۲۵] مساحت و انرژی مصرفی کمتری نسبت به سایرین دارد.

جدول ۷ نتایج پیاده‌سازی عملیات معکوس مبتنی بر ITA را با استفاده از کتابخانه سلولی استاندارد ۶۵ نانومتر نشان می‌دهد که مطابق آن روش موجود در [۳۳] با استفاده از ضرب‌کننده مختلط (مختلط دوگانه) سریع‌ترین روش روی میدان $GF(2^{233})$ و روش TIT موجود در [۳۱] سریع‌ترین روش در سایر میدان‌های استاندارد است.

متأسفانه پیاده‌سازی‌های انجام‌شده دو روش مبتنی بر الگوریتم تعمیم‌یافته اقلیدسی و روش ITA به صورت ASIC با استفاده از دو کتابخانه سلولی متفاوت انجام شده و امکان مقایسه دقیق وجود ندارد اما همان طور که جداول فوق نشان می‌دهد، زمان اجرای الگوریتم اقلیدسی تعمیم‌یافته در حالت میانگین از روش مبتنی بر ITA کمتر است. همچنین با توجه به توان مصرفی گزارش‌شده برای آن، به نظر می‌رسد که این روش برای پیاده‌سازی در سنسورها و گره‌های موجود در اینترنت اشیا مناسب است.

خلاصه‌ای از بهترین کارهای انجام‌شده از نقطه‌نظرهای مختلف در جدول ۸ ارائه شده است.

۵- الگوریتم‌های معکوس بر روی میدان‌های اول

پژوهش‌های انجام‌شده بر روی میدان اول بر مبنای الگوریتم اقلیدسی تعمیم‌یافته بنا شده است. در ادامه روش‌های موجود به چند دسته تقسیم گردیده و هر یک به تفصیل مورد بررسی قرار گرفته است.

۵-۱ الگوریتم اقلیدسی تعمیم‌یافته دودویی (b-EEA)

الگوریتم ۷ به علت تعداد زیاد عملیات تقسیم مورد نیاز در الگوریتم اقلیدسی تعمیم‌یافته، برای میدان‌های اول معرفی شده است. مشابه با

جدول ۷: نتایج پیاده‌سازی روش ITA در پایه نرمال با کتابخانه استاندارد ۶۵ نانومتر.

منابع	الگوریتم	میدان	تعداد کلاک	تأخیر مسیر بحرانی	زمان (ns)	مساحت $\times 10^2$
[۳۱]	ITA + HD	۱۶۳	۶۵	۲٫۶۵	۱۴۶٫۲	۱۹۹
[۳۱]	TIT	۱۶۳	۳۷	۲٫۲۵	۸۳٫۲	۱۹۹
[۳۳]	OAC	۲۳۳	۷۲	۳٫۶۳	۲۶۱٫۴	۱۱۴
[۳۳]	۳chain + HD	۲۳۳	۸۶	۲٫۶۶	۲۲۸٫۸	۱۵۰
[۳۱]	ITA + HD	۲۸۳	۱۳۴	۲٫۵۵	۳۴۱٫۷	۴۲۳
[۳۱]	TIT	۲۸۳	۹۸	۲٫۵۵	۲۴۹٫۹	۴۲۳
[۳۳]	OAC	۲۸۳	۱۰۱	۴٫۶	۴۶۴٫۶	۱۹۰
[۳۳]	TIT	۲۸۳	۹۸	۴٫۱۲	۴۰۳٫۸	۳۷۰
[۳۳]	۳chain + HD	۲۸۳	۷۴	۴٫۱۶	۳۰۷٫۸	۳۷۰
[۳۱]	ITA + HD	۴۰۹	۱۵۲	۱٫۹۱	۲۹۰٫۱	۳۵۴
[۳۱]	TIT	۴۰۹	۱۰۷	۱٫۹۱	۲۰۱٫۴	۳۵۵
[۳۱]	ITA + HD	۵۷۱	۱۷۱	۲٫۶۳	۴۴۹٫۸	۱۱۸۱
[۳۱]	TIT	۵۷۱	۱۰۶	۲٫۶۳	۲۷۸٫۸	۵۵۹
[۳۱]	OAC	۵۷۱	۲۱۸	۴٫۹۳	۱۰۷٫۸	۴۶۸
[۳۳]	TIT	۵۷۱	۱۳۰	۶٫۲۹	۸۱۷٫۷	۱۵۳۵
[۳۳]	۳chain + HD	۵۷۱	۱۱۴	۶٫۳۲	۷۲۰٫۴۸	۱۵۳۵

جدول ۸: بهترین کارهای انجام‌شده از نقطه‌نظرهای مختلف.

مقالات	دلیل برتری
[۲۴]	سریع‌ترین روش مبتنی بر الگوریتم تعمیم‌یافته اقلیدسی از نظر پیچیدگی زمانی در میدان دودویی
[۲۵]	کم‌مساحت‌ترین روش مبتنی بر الگوریتم تعمیم‌یافته اقلیدسی از نظر پیچیدگی مساحت در میدان دودویی
[۴۱]	سریع‌ترین روش مبتنی بر ITA از نظر پیچیدگی زمانی در میدان دودویی
[۳۵]	سریع‌ترین روش مبتنی بر ITA پیاده‌شده در میدان $GF(2^{163})$ و $GF(2^{233})$ روی بورد Virtex-۴ و در پایه نرمال
[۳۲]	سریع‌ترین روش مبتنی بر ITA پیاده‌شده در میدان $GF(2^{283})$، $GF(2^{409})$ و $GF(2^{571})$ روی بورد Virtex-۴ و در پایه نرمال
[۴۴]	سریع‌ترین روش مبتنی بر ITA پیاده‌شده در کلیه میدان‌های دودویی استاندارد روی بورد Virtex-۴ و در پایه چندجمله‌ای
[۲۵]	بهترین پیاده‌سازی انجام‌شده مبتنی بر الگوریتم تعمیم‌یافته اقلیدسی روی میدان $GF(2^{233})$ با استفاده از کتابخانه سلولی ۴۵ نانومتر از نظر مساحت، زمان و توان مصرفی
[۳۳]	سریع‌ترین روش مبتنی بر ITA پیاده‌شده در میدان $GF(2^{233})$ با استفاده از کتابخانه سلولی ۶۵ نانومتر
[۳۳]	سریع‌ترین روش مبتنی بر ITA پیاده‌شده در میدان $GF(2^{163})$، $GF(2^{283})$، $GF(2^{409})$ و $GF(2^{571})$ با استفاده از کتابخانه سلولی ۶۵ نانومتر

بخش میدان‌های دودویی در این روش نیز، تقسیم پرهزینه قبلی، با چندین عملیات تقسیم بر دو به صورت متوالی جایگزین شده است. مقاله‌های [۵۲] تا [۵۴] پیاده‌سازی‌های خوبی از این روش را ارائه داده‌اند. زمان اجرای این الگوریتم نیز مشابه قبل برابر $O(\log_2^P)$ می‌باشد.

یکی از مشکلات مهم روش‌های مبتنی بر الگوریتم تعمیم‌یافته اقلیدسی نظیر الگوریتم اقلیدسی دودویی، وابسته‌بودن زمان اجرا به مقدار ورودی است. در [۵۳] الگوریتمی ارائه شده که باعث ثابت‌شدن زمان اجرای الگوریتم فوق $(\log_2^P)$ می‌شود. در این مقاله برای ثابت‌کردن مسیر در همه حالت‌های شرطی چند دستور اضافه شده که هیچ کاربردی در روند انجام عملیات معکوس ندارند و صرفاً برای از بین بردن وابستگی زمانی الگوریتم به ورودی، اضافه شده‌اند.

۵-۲ الگوریتم معکوس مونتگمری

همان طور که برای میدان دودویی بیان شد، الگوریتم معکوس مونتگمری بر مبنای ضرب مونتگمری بیان گردیده و از دو فاز تشکیل شده است. در فاز اول مقدار $r = a^{-1}2^k \bmod p$ به دست می‌آید که $n \le k \le 2n$ است.

در فاز دوم مقدار r تصحیح می‌شود تا مقدار دقیق معکوس حاصل شود [۵۵]. در این فاز به این شکل عمل می‌شود که اگر r زوج باشد، $r = r/2$ است و اگر فرد باشد، $r = (r+p)/2$ است. در این الگوریتم، عملیات فوق $2n-k$ بار تکرار می‌شود تا نتایج موجود در (۲۶) حاصل شود. همان طور که در بخش میدان‌هایی دودویی اشاره شد، زمان اجرای فاز اول این الگوریتم نیز از رابطه $\log P \le k \le 2\log P$ قابل محاسبه است

$$b = a^{-1}2^n \bmod p$$
$$n = [\log_2^p] \qquad (۲۶)$$

مشکل وابستگی زمان اجرا به ورودی و ثابت‌نبودن زمان برای الگوریتم مونتگمری نیز وجود دارد چون به ازای اعضای یک میدان زمان‌های متفاوت دارند که برای پیاده‌سازی سخت‌افزاری مشکل بزرگی تلقی می‌شود.

الگوریتم ۷: محاسبه معکوس به روش اقلیدسی تعمیم‌یافته دودویی در میدان اول [۵۲]

Input: $a \in [1, p-1]$, $a \neq 0$

Output: $r = [1, p-1]$ where $r = a^{-1} \bmod p$

1. $u = p$, $v = a$, $r = 0$, $s = 1$
2. while $v > 0$

3. while u is even do
4. $u = \frac{u}{2}$
5. if r is even then $r = \frac{r}{2}$ else $\frac{r+p}{2}$, end if
6. end while
7. while v is even do
8. $v = \frac{v}{2}$
9. if s is even then $s = \frac{s}{2}$ else $s = \frac{s+p}{2}$, end if
10. end while
11. if $u > v$ is even then $u = u - v$, $r = r - s$
12. else $v = v - u$, $s = s - r$
13. end if
14. end while
15. if $r \geq p$ then $r = r - p \bmod p$, end if
16. if $r < 0$ then $r = r + p \bmod p$, end if
17. return r

مقاله [۵۶] الگوریتم مونتگمری را به نحوی تغییر داده تا این مشکل را رفع کند. الگوریتم موجود در این مقاله که به ازای یک میدان مشخص، زمان ثابت دارد در برابر بخش مهمی از حملات کانال جانبی مصون است. حلقه اصلی روش مونتگمری $2n-1$ بار اجرا می‌شود که هر بار چهار حالت شرطی در آن وجود دارد که هر کدام شامل دستورات متفاوت هستند. نویسندگان این مقاله تلاش کرده‌اند که مسیر اجرای الگوریتم در صورت اجرای هر کدام از این شرطها یکسان باشد. به عبارتی در هر بار اجرای حلقه اصلی برنامه، یک جمع، یک تفریق و ۶ جابه‌جایی استفاده می‌شود.

مقاله [۵۵] روش [۵۶] را بهبود داده است. مقاله [۵۵] دو مزیت نسبت به مقاله فوق دارد. مزیت اول کم‌کردن تعداد عملیاتی است که در هر بار اجرای حلقه اصلی برنامه اجرا می‌شوند. این مسیر شامل یک جمع، یک تفریق و دو جابه‌جایی می‌باشد. مزیت دوم این است که در انتهای الگوریتم معکوس عدد ورودی به طور کامل به دست می‌آید و برخلاف مقاله قبلی نیاز به فاز دوم برای تصحیح مقدار معکوس که در الگوریتم مونتگمری به آن اشاره شده است ندارد.

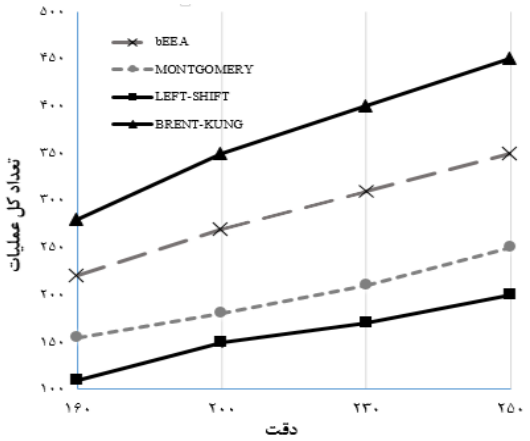
۳-۵ عملیات معکوس مبتنی بر جابه‌جایی به چپ

دو دسته قبلی از الگوریتم‌های معرفی‌شده، مبتنی بر جابه‌جایی به راست بودند. در [۵۷] الگوریتم به نحوی تغییر داده شده که در آن فقط از جابه‌جایی به چپ استفاده می‌شود. این الگوریتم ساختار بهتری برای پیاده‌سازی سخت‌افزاری دارد زیرا به یک مقایسه‌کننده اعداد صحیح در حلقه اصلی برنامه نیاز ندارد. مزیت دیگر این روش، تعداد کمتر عمل جمع و تفریق است که هزینه عملیاتی معکوس را کاهش می‌دهد.

۴-۵ الگوریتم اقلیدسی تعمیم‌یافته به روش برِنت- کونگ

همان طور که برای میدان‌های دودویی بحث شد، در اغلب الگوریتم‌های محاسبه معکوس در حلقه اصلی برنامه یک مقایسه برای یافتن بزرگ‌ترین درجه چندجمله‌ای نیاز است که این عملیات محاسباتی در میدان اول پرهزینه می‌باشد. به منظور حل این مشکل، آقایان برِنت و کونگ در [۱۰] راهکار جدیدی را ارائه داده‌اند که مسیر جدیدی برای محاسبه معکوس در میدان‌های اول و دودویی به کمک معماری آرایه‌ای ضربانی ایجاد کرد.

زمان اجرای این روش برابر $\log_2^P$ است که با استفاده از $\log_2^P$ واحد



شکل ۴: مقایسه روش‌های محاسبه معکوس روی میدان‌های اول.

پردازشی تک‌بیتی نتیجه را محاسبه می‌کند و کارایی کلی این روش برابر $(\log_2^P)^2$ می‌باشد. همچنین یکی از مشکلات این الگوریتم احتمال منفی‌شدن پارامتر δ است که این رخداد، باعث طولانی‌شدن حلقه اصلی برنامه نسبت به سایر الگوریتم‌های این دسته می‌شود. البته به دلیل عدم نیاز به مقایسه‌کننده، این روش برای محاسبات بدون رقم نقلی مناسب است.

۶- مقایسه الگوریتم‌های معکوس روی میدان اول

در این بخش برای مقایسه، چهار روش فوق از نظر تعداد متوسط هزینه عملیاتی مورد بررسی قرار می‌گیرد. نتایج شکل ۴ بعد از ۱۰۰۰۰۰ بار اجرای الگوریتم‌های فوق با استفاده از ۱۰۰ عدد صحیح متفاوت و ۱۰۰ عدد اول متفاوت حاصل شده است. بر اساس نتایج موجود در شکل ۴، الگوریتم جابه‌جایی به چپ و با اختلاف کمی نسبت به الگوریتم مونتگمری کمترین هزینه عملیاتی را دارد. مشکل مهم در ساختار الگوریتم جابه‌جایی به چپ، پیچیده‌بودن حالت‌های شرطی داخل آن است که برای پیاده‌سازی سخت‌افزاری مناسب به نظر نمی‌رسد و بنابراین الگوریتم مونتگمری می‌تواند بهترین روش برای پیاده‌سازی سخت‌افزاری باشد.

الگوریتم اقلیدسی تعمیم‌یافته دودویی با روش برِنت- کونگ نیز به وضوح نتایج بدتری را در عمل نسبت به سایر روش‌ها به دست آورده است. یکی از دلایل مهم آن، عدم استفاده از مقایسه‌کننده موجود در سایر الگوریتم‌ها است و همین باعث زیادشدن تعداد گام‌های حلقه اصلی آن شده ولی این به معنی عدم استفاده از این الگوریتم در پیاده‌سازی‌های سخت‌افزاری نیست. چون به علت امکان استفاده از این روش در ساختارهای آرایه‌ای ضربانی و خواص مهم این ساختار، می‌تواند بسیار مورد توجه قرار بگیرد.

همانند میدان دودویی در میدان اول نیز طبق (۴) می‌توان معکوس یک عضو میدان را با ضرب و توان‌رسانی‌های متوالی در زمان ثابت محاسبه کرد. این روش در حالت کلی به $n = \log_2^P$ ضرب و $n/2$ توان‌رسانی نیاز دارد. یافتن بهترین زنجیره برای این روش در میدان‌های اول بسیار دشوار اما برای یک میدان خاص، به شدت قابل بهبود است. مثلاً برای میدان $p = 2^{255} - 19$ عملیات معکوس تنها با ۱۱ ضرب و ۲۵۴ مجذور قابل انجام است [۵۸]. پیچیدگی زمانی روش‌های مبتنی بر الگوریتم اقلیدسی تعمیم‌یافته برابر $O(\log p)^2$ و روش‌های مبتنی بر قضیه کوچک فِرما برابر $O(\log p)^2$ می‌باشد. بنابراین می‌توان استنباط کرد که در حالت کلی پیچیدگی زمانی روش‌های مبتنی بر الگوریتم اقلیدسی

تعمیم‌یافته از روش‌های مبتنی بر قضیه کوچک فِرما در میدان‌های اول کمتر است.

۷- جمع‌بندی و کارهای آتی

عملیات معکوس در سامانه‌های رمزنگاری کلید عمومی از اهمیت ویژه‌ای برخوردار است و هزینه محاسباتی و مساحت زیادی را به خود اختصاص می‌دهد. در این مقاله، روش‌های موجود برای انجام عملیات معکوس بر روی میدان‌های دودویی و اول مورد بررسی قرار گرفته است. در این راستا روش‌های موجود برای انجام عملیات معکوس در میدان‌های دودویی بررسی، الگوریتم‌های موجود مطرح و از نظر پیچیدگی زمانی و منابع مورد نیاز با هم مقایسه شده است. سپس به معرفی بهترین روش‌ها و پیاده‌سازی‌های موجود روی بستر FPGA و به صورت مدار مجتمع پرداخته شد. در ادامه به بررسی میدان‌های اول پرداخته شده و روش‌های موجود معرفی گردید. در انتها نیز روش‌های موجود در میدان اول از نظر پیچیدگی محاسباتی و هزینه زمانی مورد مقایسه قرار گرفته و بهترین روش معرفی شد.

مراجع

[1] E. Wenger and M. Hutter, "Exploring the design space of prime field vs. binary field ECC-hardware implementations," in *Proc. Nordic Conf. on Secure IT Systems*, pp. 256-271, Tallinn, Estonia, 26-28 Oct. 2011.

[2] J. P. Deschamps, J. L. Imana, and G. D. Sutter, *Hardware Implementation of Finite-Field Arithmetic*, New York: McGraw-Hill, 2009.

[3] Q. Liao, "The Gaussian normal basis and its trace basis over finite fields," *Number Theory*, vol. 132, no. 7, pp. 1507-1518, Jul. 2007.

[4] A. Brauer, "On addition chains," *Bulletin of the American Mathematical Society*, vol. 45, no. 10, pp. 736-739, 1939.

[5] D. E. Knuth, The Art of Computer Programming: Sorting and Searching: Volume 3, 1973.

[6] P. L. Montgomery, "Modular multiplication without trial division," *Mathematics of Computation*, vol. 44, no. 170, pp. 519-521, Dec. 1985.

[7] A. A. Gutub, A. F. Tenca, and C. K. Koc, "Scalable VLSI architecture for GF(p) montgomery modular inverse computation," in *Proc. of IEEE Computer Society Annual Symp.on VLSI*, pp. 53-58, Pittsburgh, PA, USA, 25-26 Apr. 2002.

[8] A. Daly, W. Marnane, T. Kerins, and E. Popovici, "Fast modular division for application in ECC on reconfigurable logic," *Lecture Notes in Computer Science*, pp. 786-795, Sept. 2003.

[9] J. Bucek and R. Lorencz, "Comparing subtraction-free and traditional AMI," in *Proc. of 9th IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems*, *DDECS'06*, pp. 95-97, Prague, Czech Republic, 18-21 Apr. 2006.

[10] R. P. Brent and H. T. Kung, "Systolic VLSI arrays for polynomial GCD computation," *IEEE Trans. on Computers*, vol. 33, no. 8, pp. 731-736, Aug. 1984.

[11] H. T. Kung, "Why systolic arrays?" *Computer*, vol. 15, pp. 37-46, Jan. 1982.

[12] J. Stein, "Computational problems associated with Racah Algebra," *J. of Computational Physics*, vol. 1, no. 3, pp. 397-405, Feb. 1967.

[13] C. H. Wu, C. M. Wu, M. D. Shieh, and Y. T. Hwang, "High-speed, low complexity systolic designs of novel iterative division algorithms in GF (2^m)," *IEEE Trans. on Computers*, vol. 53, no. 3, pp. 375-380, Mar. 2004.

[14] H. Brunner, A. Curiger, and M. Hofstette, "On computing multiplicative inverses in GF (2^m)," *IEEE Trans. on Computers*, vol. 42, no. 8, pp. 1010-1015, Aug. 1993.

[15] J. H. Guo and C. L. Wang, "Systolic array implementation of Euclid's algorithm for inversion and division in GF (2^m)," *IEEE Trans. on Computers*, vol. 47, no. 10, pp. 1161-1167, May 1998.

[16] Z. Yan and D. B. Sarwate, "High-speed systolic architectures for finite field inversion," *Integration: The VLSI J.*, vol. 38, no. 3, pp. 383-398, Jan. 2005.

[17] M. Mozaffari-Kermani, R. Azarderakhsh, C. Y. Lee, and S. Bayat-Sarmadi, "Reliable concurrent error detection architectures for extended euclidean-based division over GF (2^m)," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 5, pp. 995-1003, May 2014.

[18] S. Kim, N. S. Chang, C. H. Kim, Y. H. Park, and J. Lim, "A fast inversion algorithm and low-complexity architecture over GF (2^m)," In: Hao Y. et al. (eds) *Computational Intelligence and Security*, *CIS'05*, 8 pp., Lecture Notes in Computer Science, vol 3802. Springer, Berlin, Heidelberg ,2005.

[19] Z. Yan, D. V. Sarwate, and Z. Liu, "Hardware-efficient systolic architectures for inversions in GF (2^m)," *IEE Proceedings-Computers and Digital Techniques.*, vol. 145, no. 4, pp. 272-278, Jul. 1998.

[20] Z. Yan, "Digit-serial systolic architectures for inversions over GF (2^m)," in *Proc. IEEE Workshop on Signal Processing Systems Design and Implementation*, pp. 77-82, Banff, Canada, 2-4 Oct. 2006.

[21] A. K. Daneshbeh and M. A. Hasan, "A class of unidirectional bitserial systolic architectures for multiplicative inversion and division over GF (2^m)," *IEEE Trans. Computer*, vol. 54, no. 3, pp. 370-380, Mar. 2005.

[22] J. H. Guo and C. L. Wang, "Bit-serial systolic array implementation of Euclid's algorithm for inversion and division in GF (2^m)," in *Proc. of Technical Papers. Int. Symp. on VLSI Technology, Systems, and Applications*, pp. 113-117, Taipei, Taiwan, 3-5 Jun. 1997.

[23] J. Fan, L. Batina, and I. Verbauwhede, "Design and design methods for unified multiplier and inverter and its application for HECC," *Integration, VLSI J.*, vol. 44, no. 4, pp. 280-289, Sept. 2011.

[24] K. Kobayashi and N. Takagi, "Fast hardware algorithm for division in G (2^m) based on the extended euclid's algorithm with parallelization of modular reductions," *IEEE Trans. on Circuits and Systems II: Express Briefs*, vol. 56, no. 8, pp. 644-648, Aug. 2009.

[25] A. Ibrahim, T. F. Al-Somani, and F. Gebali, "New systolic array architecture for finite field inversion," *Canadian J. of Electrical and Computer Engineering*, vol. 40, no. 1, pp. 23-30, Winter 2017.

[26] C. C. Wang, *et al*., "VLSI architectures for computers multiplications and inverses in GF (2^m)," *IEEE Trans. on Computers*, vol. 34, no. 8, pp. 709-717, Aug. 1985.

[27] T. Itoh and S. Tsujii, "A fast algorithm for computing multiplicative inverses in GF (2^m) using normal bases," *Information and Computation*, vol. 78, no. 3, pp. 171-177, Sept. 1988.

[28] N. Takagi, "A fast algorithm for multiplicative inversion in GF (2^m) using normal basis," *IEEE Trans. on Computers*, vol. 42, no. 9, pp. 1141-1146, May 1993.

[29] V. Dimitrov and K. Jarvinen, "Another look at inversions over binary fields," in *Proc. IEEE Symp. on Computer Arithmetic, ARITH'13*, pp. 211-218, Austin, TX, USA, 7-10 Apr. 2013.

[30] R. Azarderakhsh and A. Reyhani-Masoleh, "Low-complexity multiplier architectures for single and hybrid-double multiplications in Gaussian normal bases," *IEEE Trans. on Computers*, vol. 62, no. 4, pp. 744-757, Apr. 2013.

[31] R. Azarderakhsh, K. Jarvinen, and V. Dimitrov, "Fast inversion in GF (2^m) with normal basis using hybrid-double multipliers," *IEEE Trans. on Computers*, vol. 63, no. 4, pp. 1041-1047, Apr. 2014.

[32] J. Hu, W. Guo, J. Wei, and R. C. Cheung, "Fast and generic inversion architectures over GF (2^m) using modified Itoh-Tsujii algorithms," *IEEE Trans. on Circuits and Systems II: Express Briefs*, vol. 62, no. 4, pp. 367-371, Apr. 2015.

[33] K. Jarvinen, V. Dimitrov, and R. Azarderakhsh, "A generalization of addition chains and fast inversions in binary fields," *IEEE Trans. on Computers*, vol. 64, no. 9, pp. 2421-2432, Sept. 2015.

[34] M. Nocker, Data Structures for Parallel Exponentiation in Finite Fields, Ph.D Diss., 2001.

[35] B. Rashidi, "High-speed hardware implementation of Gaussian normal basis inversion algorithm over $F_2{}^m$," *Microelectronics J.*, vol. 63, pp. 138-147, May 2017.

[36] J. Guajardo and C. Paar, "Itoh-Tsujii inversion in standard basis and its application in cryptography and codes," *Designs, Codes and Cryptography*, vol. 25, no. 2, pp. 207-216, Nov. 2002.

[37] F. Rodriguez-Henriquez, N. Cruz-Cortes, and N. A. Saqib, "A fast implementation of multiplicative inversion over GF (2^m)," in *Proc. Int. Conf. on Information Technology: Coding and Computers, ITCC'05*, vol. 1, pp. 574-579, Covtat, Croatia, 20-23 Jun. 2005.

[38] F. Rodriguez-Henriquez, G. Morales-Luna, N. A. Saqib, and N. Cruz-Cortes, "Parallel Itoh-Tsujii multiplicative inversion algorithm for a special class of trinomials," *Designs, Codes and Cryptography*, vol. 45, no. 1, pp. 19-37, Jan. 2007.

[39] C. Rebeiro, S. S. Roy, S. S. Reddy, and D. Mukhopadhyay, "Revisiting the Itoh-Tsujii inversion algorithm for FPGA platforms," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 19, no. 8, pp. 1508-1512, Aug. 2011.

[40] V. R. Venkatasubramani, N. Murali, and S. Rajaram, "An improved quad Itoh-Tsujii algorithm for FPGAs," *IEICE Electronics Express*, vol. 10, no. 18, 10 pp., Article -20130612, Jan. 2013.

[41] S. S. Roy, C. Rebeiro, and D. Mukhopadhyay, "Theoretical modeling of the Itoh-Tsujii inversion algorithm for enhanced performance on k-LUT based FPGAs," in *Proc.Design, Automation and Test in Europe Conf. and Exhibition, DATE'16*, 6 pp., Grenoble, France, 14-18 Mar. 2011.

[42] S. S. Roy, C. Rebeiro, and D. Mukhopadhyay, "Generalized high speed Itoh-Tsujii multiplicative inversion architecture for FPGAs," *Integration, The VLSI J.*, vol. 45, no. 3, pp. 307-315, Jun. 2012.

[43] L. Parrilla, A. Lloris, E. Castillo, and A. Garcia, "Minimum-clock-cycle Itoh-Tsujii algorithm hardware implementation for cryptography applications over GF (2^m) fields," *Electronics Letters*, vol. 48, no. 18, pp. 1126-1128, Aug. 2012.

[44] L. Li and S. Li, "Fast inversion in GF (2^m) with polynomial basis using optimal addition chains," in *Proc. IEEE Int Symp. in Circuits and Systems, ISCAS'17*, 4 pp., Baltimore, MD, USA, 28-31 May 2017.

[45] J. Maitin-Shepard, "Optimal software-implemented Itoh-Tsujii inversion for GF (2^m)," *Designs, Codes and Cryptography*, vol. 82, no. 2, pp. 301-318, Aug. 2017.

[46] F. Rodriguez-Henriquez, N. A. Saqib, A. D. Perez, and C. K. Koc, *Cryptographic Algorithms on Reconfigurable Hardware*, Springer-Verlag US, 2007.

[47] J. Li, Z. Li, C. Xue, J. Zhang, W. Gao, and S. Cao, "A fast modular inversion FPGA implementation over GF (2^m) using modified x^{2n} unit," in *Proc. IEEE Int. Symp. on Circuits and Systems, ISCAS'18*, 5 pp., Florence, Italy, 27-30 2018.

[48] G. M. de Dormale and J. J. Quisquater, "High-speed hardware implementations of elliptic curve cryptography: a survey," *J. of Systems Architecture*, vol. 53, no. 2-3, pp. 72-84, Feb./Mar. 2007.

[49] B. Vallee, "Dynamics of the binary Euclidean algorithm: functional analysis and operators," *Algorithmic*, vol. 22, no. 4, pp. 660-685, Mar. 1998.

[50] A. Akhavi and B. Vallee, "Average bit-complexity of Euclidean algorithms," in *Proc. Int. Colloquium on Automata, Languages, and Programming*, pp. 373-387, Ceneva, Switzerland, 14-21 Jul. 2000.

[51] H. Mahdizadeh and M. Masoumi, "Novel architecture for efficient FPGA implementation of elliptic curve cryptographic processor over GF (2^{163})," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 12, pp. 2330-2333, Dec. 2013.

[52] N. Takagi, "A VLSI algorithm for modular division based on the binary GCD algorithm," *IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E81-A, no. 5, pp. 724-728, 1998.

[53] L. A. Tawalbeh, A. F. Tenca, S. Park, and C. K. Koc, "A dual-field modular division algorithm and architecture for application specific hardware," in *Proc. IEEE the 38th Asilomar on Signals, Systems and Computers Conf.*, vol. 1, pp. 483-487, Nov. 2004.

[54] A. Mrabet, N. El-Mrabet, B. Bouallegue, S. Mesnager, and M. Machhout, "An efficient and scalable modular inversion/division for public key cryptosystems," in *Proc. Int. Conf. on Engineering & MIS, ICEMIS'17*, 6 pp., Astana, Kazakhstan, 6-8 Jun. 2017.

[55] E. Savas and C. K. Koc, "Montgomery inversion," *J. of Cryptographic Engineering*, vol. 8, no. 3, pp. 201-210, Apr. 2018.

[56] J. W. Bos, "Constant time modular inversion," *J. of Cryptographic Engineering*, vol. 4, no. 4, pp. 275-281, Aug.. 2014.

[57] D. J. Bernstein, "Curve25519: new Diffie-Hellman speed records," in *Proc. Int. Workshop on Public Key Cryptography*, pp. 207-228, New York, NY, USA, 5-7 Nov. 2006.

رضا روح قلندری تحصیلات خود را در مقاطع کارشناسی و کارشناسی ارشد در رشته مهندسی کامپیوتر به‌ترتیب در سال‌های ۱۳۹۵ و ۱۳۹۷ به‌ترتیب از دانشگاه شهید بهشتی و دانشگاه صنعتی شریف به پایان رسانده است. زمینه‌های تحقیقاتی مورد علاقه ایشان عبارتند از: مهندسی رمزنگاری و معماری کامپیوتر.

حاتمه مثنایی بورانی تحصیلات خود را در مقاطع کارشناسی و کارشناسی ارشد و دکتری در رشته مهندسی کامپیوتر به‌ترتیب در سال‌های ۱۳۹۲ و ۱۳۹۶ از دانشگاه صنعتی اصفهان و دانشگاه صنعتی شریف به پایان رسانده است و هم‌اکنون در آزمایشگاه تحقیقاتی سامانه‌های هوشمند و امن در دانشگاه صنعتی شریف مشغول به کار می‌باشد. زمینه‌های تحقیقاتی مورد علاقه ایشان عبارتند از: مهندسی رمزنگاری، امنیت و اعتماد سخت‌افزار و طراحی مدارهای مجتمع.

سیاوش بیات سرمدی تحصیلات خود را در مقاطع کارشناسی و کارشناسی ارشد و دکتری در رشته مهندسی کامپیوتر به ترتیب در سال‌های ۱۳۷۹، ۱۳۸۱ و ۱۳۸۶ از دانشگاه تهران، دانشگاه صنعتی شریف و دانشگاه واتراو (کانادا) به پایان رسانده است و هم‌اکنون دانشیار دانشکده مهندسی کامپیوتر دانشگاه صنعتی شریف می‌باشد. نام‌برده قبل از پیوستنش به دانشگاه صنعتی شریف در سال‌های ۱۳۸۶ الی ۱۳۹۲ در شرکت ای ام دی (AMD Inc.) مشغول به کار بوده است. زمینه‌های تحقیقاتی مورد علاقه ایشان عبارتند از: مهندسی رمزنگاری، امنیت و اعتماد سخت‌افزار، پیاده‌سازی‌ها و معماری‌های امن، کارا و قابل اطمینان در اینترنت اشیا و سیستم‌های سایبر- فیزیکی.