Archvie of SID

هشتمین کنفرانس بین المللی انجمن رمز ایران

۲۴ و ۲۵ شهریورماه ۱۳۸۹، دانشگاه صنعتی خواجه نصیرالدین طوسی

انجمن رمزایران

دانشگاه صنعتی خواجه نصیرالدین طوسی

# بهبود الگوریتم رمز منحنی‌های بیضوی با استفاده از کدگذاری داده

زهرا میرمحمدی[۱]، سعادت پورمظفری[۲]

دانشگاه صنعتی امیرکبیر۱

mirmohammadi@itrc.ac.ir

تهران، دانشگاه صنعتی امیرکبیر، گروه مهندسی کامپیوتر [۲و۳]

saadat@ce.aut.ac.

## چکیده

معماری‌های مختلفی برای پیاده‌سازی سخت‌افزاری الگوریتم رمز منحنی های بیضوی پیشنهاد شده است که هدف آنها بالا بردن کارایی و سطح امنیت می باشد. افزایش کارایی یعنی در عین بالا بردن سرعت پردازش به فضای مصرفی نیز توجه کرده‌اند. در بسیاری از معماری ها ی مطرح در الگوریتم رمز منحنی های بیضوی، الگوریتم ضرب پیمانه ای مونتگمری با استفاده از جمع کننده CSA بکار برده می شود.. در این مقاله برای بهبود الگوریتم رمز منحنی های بیضوی از الگوریتم بهبود یافته مونتگمری با بکارگیری جمع کننده سریع که از CSA نیزمناسب‌تر می‌باشد،استفاده شده است. این روش علاوه برکاهش تاخیر ( افزایش سرعت پردازش) فضای مصرفی را نیز کاهش داده و کارایی بیشتری حدود ۱۸٪ در مقایسه با معماری‌های قبلی دارد. [*۱]

## واژه های کلیدی

سیستم رمز، رمزنگاری، ECC ، رمزگشایی، ضرب مونتگمری، تقسیم پیمانه ای و حملات کانال جانبی

طراحی و تحلیل رمزها ایفا می‌کند. از آنجا که تحلیل جبری این عملگر برای $r = 2^n$ عملوند، در مقاله‌ای به تألیف مؤلفان همین مقاله، در ششمین کنفرانس رمز ایران انجام شده است، در این مقاله با استفاده از نتایج پژوهش‌های پیشین، به بررسی جبری جمع پیمانه‌ای به پیمانه‌ی $2^t$ پرداخته، درجات توابع بولی مؤلفه‌ای این عملگر را ( به عنوان یک نگاشت بولی ) در حالت کلی، به دست می‌آوریم. همچنین به عنوان یک کاربرد، درجه‌ی توابع مؤلفه‌ای مذکور را برای یکی از پرکاربردترین حالات، یعنی تبدیل‌های PHT به کار رفته در رمزهای قالبی و دنباله‌ای به دست می‌آوریم.

## واژه های کلیدی

جمع پیمانه‌ای به پیمانه‌ی $2^t$، توابع بولی، فرم نرمال جبری، درجه‌ی جبری.

[*]این مقاله تحت حمایت مرکز تحقیقات مخابرات ایران می باشد.

www.SID.ir

Archvie of SID

## ۱- مقدمه

در سالهای اخیر به ارتباطات هوشمند در بستر الکترونیک بیشتر از گذشته توجه شده و بنابراین حفاظت از اطلاعات و امنیت آنها اهمیت یافته است، برای حفظ امنیت داده ها می توان قبل از ارسال آنها را با الگوریتم های مختلف رمزنگاری کرد. ویژگی مهم یک الگوریتم رمز پیچیدگی مناسب برای کاهش احتمال رمزگشایی توسط غیر است. و در نتیجه بالا بود سطح امنیت است. امنیت بالا در واقع از مقاومت در برابر حملات ایجاد می شود.

از اینرو الگوریتم‌های متفاوتی برای رمزنگاری مطرح می‌شود. یکی از مهمترین آن‌ها سیستم رمزنگاری منحنی‌های بیضوی[۲] با طول کلید۱۶۰ بیت وسطح امنیتی بالا است[۱] در این سیستم رمز، حتی با استفاده از کلیدهای کوتاه می-توان به سطح امنیتی مناسبی دست یافت. این نوع رمزنگاری در برابر حملات شناخته شده از جمله حملات پرکاربرد کانال جانبی مقاوم می باشد. کوتاه بودن طول کلید در سیستم‌های رمزنگاری دارای اهمیت بسزایی می‌باشد و سبب کاهش توان مصرفی سیستم، پهنای باند سیستم، میزان پردازش و حافظه مورد نیاز می شود. به دلیل کمبود این منابع در سیستم‌های قابل حمل[۳]، از قبیل تلفن‌های همراه و کارت‌های هوشمند، محدودیت‌هایی در استفاده از الگوریتم‌های رمزنگاری وجود دارد. سیستم رمزنگاری مبتنی بر خم بیضوی به عنوان یک راهکار مناسب جهت رفع محدودیت‌های مذکور، استفاده می شود[۲،۳]. البته مشکل این سیستم، زمانبر بودن محاسبات بدلیل پیچیدگی بالای عملیات می‌باشد که بهتر است رفع شود. یکی از روش‌های مطرح برای سرعت بخشیدن به محاسبات در این سیستم، ضرب‌های پیمانه‌ای مونتگمری است[۶،۵،۴] که در این مقاله با کمک الگوریتم سریع جمع بدون انتقال رقم نقلی[۴] و الگوریتم ضرب مونتگمری با منطق تقسیم پیمانه‌ای به آن پرداخته شده است. طبق ادعای نویسنده مقاله‌ی[۷] با تلفیق تقسیم پیمانه ای و ضرب مونتگمری به فضا و زمان کمتری نسبت به روش‌های مشابه جهت پیاده سازی نیازمند است، از طرف دیگر با کمک گرفتن از مقاله[۸] نیز که یک کدینگ جدید برای جمع ارائه نموده، فضای مصرفی به میزان ۸٪ و زمان محاسبات ۷٪ قابل کاهش است. بنابراین می توان پیش بینی کرد که با تلفیق این دو روش بهبود مناسبی برای ضرب پیمانه‌ای مونتگمری مورد استفاده در الگوریتم رمز ECC خواهیم داشت.

در ادامه، در بخش۲ مروری بر سیستم رمز منحنی های بیضوی خواهیم داشت، در بخش۳ الگوریتم ضرب پیمانه ای که مورد استفاده سیستم های رمزنگاری مانند ECC می باشد شرح داده می شود، دربخش۴، برای درک بهتر عملکرد الگوریتم بهبود یافته ضرب پیمانه‌ای، به منطق الگوریتم تقسیم پیمانه‌ای که یکی از موثرترین روش‌ها جهت بهبود ضرب پیمانه‌ای مونتگمری است اشاره ای داشته، در بخش۵، به بررسی الگوریتم ضرب پیمانه‌ای مونتگمری با منطق تقسیم پیمانه‌ای پرداخته شده و سپس در بخش۶، به ارائه راه کاری برای سرعت بخشیدن به جمع مورد استفاده در آن، با کدینگ جدید خواهد پرداخت و در آخر الگوریتم مونتگمری بهبود یافته ارائه و صحت آن اثبات خواهد شد.

## ۲- سیستم رمز منحنی های بیضوی

یکی از سیستم‌هایی که در سال‌های اخیر، برای انجام عملیات رمزنگاری مورد استفاده قرار می گیرد، سیستم رمزنگاری مبتنی بر خم بیضوی است. این سیستم رمزنگاری برای نخستین بار در اواخر دهه ۸۰ توسط کوبلیتز[۵] ارائه شد. به دلیل کوتاه بودن طول کلید و سطح بالای امنیتی آن، یکی از بهترین سیستم‌های رمزنگاری می باشد. از مزایای این سیستم نسبت به دیگر سیستم‌های رمزنگاری، می توان به مواردی چون دارا بودن بالاترین درجه محرمانگی به ازای هر بیت، نیاز به حافظه کمتر، توان مصرفی کمتر و کاهش پهنای باند مورد نیاز سیستم اشاره کرد.

از مهمترین و بارزترین ویژگی این رمز نگاری انجام عملیات روی میدان های متناهی می باشد. فرض کنید p یک عدد اول و میدان متناهی $F_p$ شامل مجموعه ای از اعداد کوچکتر از p می باشد، خم بیضوی E با مختصات دو بعدی[۶] به صورت زیر تعریف می شود:

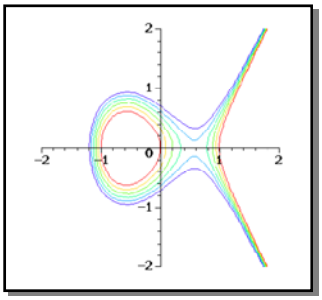
$$y^2=x^3+ax+b \qquad (1)$$

فرض در این رابطه $a,b\in F_p$ و $4a^3+27b^2\neq 0 \pmod p$ می‌باشد، اگر مختصات(x,y) یک نقطه در رابطه (۱) صدق کند آن نقطه متعلق به خم بیضوی است. باید توجه داشت که،$E(Fp)$[۷] مجموعه تمام نقاط بر روی خم بیضوی (نقاط صحیح) و P نقطه ای بر روی E است.

[2] Elliptic curve cryptography
[3] Mobile Device
[4] Carry propagat
[5] Koblitz
[6] Affin
[۷] E یک خم بیضوی تعریف شده بر روی میدان $F_p$

www.SID.ir

Archvie of SID



**شکل۱.** منحنی بیضوی

برای رمزنگاری در این سیستم، عدد تصادفی k در بازه [1,n-1] و متعلق به میدان $F_p$ انتخاب کرده و آنرا به عنوان کید خصوصی در نظر می گیرد. سپس کلید عمومی Q را به صورت Q=kP محاسبه می کند، P نقطه‌ای روی خم بیضوی می باشد.

رمزنگاری خم‌های بیضوی بر اساس دشواری مسأله لگاریتم گسسته است. به بیان دیگر، محاسبه k از روی نقاط P و Q دارای پیچیدگی نمایی است. این خصوصیت، مزیت اصلی سیستم رمزنگاری مبتنی بر خم‌بیضوی می‌باشد.

در این سیستم رمزنگاری از عملیات ضرب عددی استفاده می شود. عملیات ضرب عددی به صورت زیر بیان می شود:

$$Q = k.P = P + P + \ldots + P \quad (۲)$$

از رابطه بالا مشخص است که فقط از عملیات جمع، ضرب، توان برای محاسبه استفاده شده، زمان‌برترین آن‌ها ضرب است . با بهبود ضرب می توان کارایی کل محاسبات سیستم رمز منحنی های بیضوی را بالا برد. با توجه به اینکه عملیات در این سیستم مبتنی بر میدان است، استفاده از الگوریتم-های ضرب پیمانه‌ای به بهبود کارایی کمک زیادی می‌کند.

یکی از سریع‌ترین الگوریتم ها برای ضرب‌های پیمانه‌ای، الگوریتم ضرب مونتگمری است. که در بخش بعدی به آن پرداخته شده است.

## ۳- الگوریتم ضرب پیمانه ای مونتگمری

برای سرعت بخشیدن به محاسبات ضرب پیمانه‌ای به صورت $Z = X.Y(modM)$ از روش مونتگمری استفاده می‌شود.

معماری های مختلفی برای پیاده سازی ضرب پیمانه‌ای مونتگمری پیشنهاد شده است. منطق تقسیم پیمانه‌ای یکی از این روش‌ها می باشد این الگوریتم با کمک محاسبه بزرگترین مقسوم علیه مشترک اعداد[۸] باعث کوچک شدن طول بیت ورودی های الگوریتم ضرب شده و در بهبود محاسبات ضرب پیمانه‌ای بسیار مفید است. در مقاله[ ۴] با استفاده از منطق تقسیم پیمانه ای ضرب پیمانه ای مونتگمری، به فضا و زمان مصرفی کمتری جهت پیاده سازی، نسبت به روش های مشابه نیازمند است، از طرف دیگر در مقاله[۵] با کمک الگوریتم سریع جمع بدون انتقال رقم نقلی[۹] نیز فضای مصرفی به میزان ۸٪ و زمان محاسبات ۷٪ قابل کاهش است. در این مقاله با کمک الگوریتم سریع جمع و تقسیم پیمانه ای، الگوریتم بهبودیافته‌ای برای ضرب پیمانه‌ای مونتگمری پیشنهاد می‌شود. در این مقاله با تلفیق این دو روش بهبود ۱۸٪ برای ضرب پیمانه‌ای مونتگمری مورد استفاده در الگوریتم های رمز خواهیم داشت.

## ٤- الگوریتم تقسیم پیمانه ای

با استفاده از تقسیم پیمانه‌ای با کمک محاسبه بزرگترین مقسوم علیه مشترک اعداد[۱۰] که یکی از موثرترین روش‌ها است ضرب پیمانه‌ای که در اکثر محاسبات رمزنگاری کاربرد دارد را بهبود می‌بخشد [۴] . همانطور که می‌دانیم دربیشتر الگوریتم‌های رمز نگاری، پیمانه عددی اول و فرد[۱۱] در نظر گرفته می‌شود. برای محاسبه حاصل تقسیم پیمانه‌ای دو عدد مخالف صفر در میدان دودویی[۱۲] بزرگترین مقسوم علیه مشترک بین مقسوم علیه[۱۳] و پیمانه را محاسبه می کند.

در این روش از چهار متغیر کمکی A,B,U وV استفاده می شود، متغیرهای A,B برای محاسبه بزرگترین مقسوم علیه مشترک بین مقسوم علیه[۱۴] و پیمانه . متغیرهایV,U نیز تقسیم پیمانه‌ای نهایی را محاسبه می‌کند.

چون پیمانه عددی فرد و اول است، اگر Y عددی زوج باشد بزرگترین مقسوم علیه مشترک بین مقسوم علیه و پیمانه برابر بزرگترین مقسوم علیه مشترک بین نصف مقسوم علیه[۱۵] و پیمانه است و اگر عددی فرد باشد حاصل جمع یا حاصل تفریق آن با پیمانه بر ۴ بخش پذیر خواهد بود و داریم:

$$\begin{aligned} &\text{if } (A-B)(\text{mod } 4)=0 \text{ then} \\ &\gcd(((A-B)/(4,M)=\gcd(A,M \quad (۳) \\ &\text{if } (A+B)(\text{mod } 4)=0 \text{ then} \\ &\gcd((A+B)/(4,M))=\gcd(A,M) \end{aligned}$$

این دستورات تا زمانی که A=۰ شود ادامه پیدا می‌کند. برای این منظور ρ را برابر تعداد بیت‌های عدد A تعریف کرده و تا

[8] Greatest Common Divisor(gcd)

[9] Carry propagat

[10] gcd

[11] M is odd&prim

[12] $X,Y \in F_{2^P}, Z = X/Y(modM)$

[13] Denominator,$Y$ $in$: $X/Y$

[14] Gcd(Y,M), in: $X/Y(modM)$

[15] Gcd( $Y/2, M$) = Gcd(Y,M)

www.SID.ir

Archvie of SID

زمانی‌که مخالف صفر باشد دستورات بالا تکرار می‌شوند. در زیر به شبه کد الگوریتم اشاره شده است:

Algorithm 1 (Algorithm for Modular Division)
Inputs: M: 2^(n-1)<M>2^n& gcd(M,2)=1
X,Y: 0≥X,Y<M
Outputs: Z=X⁄Y(mod M)
Algorithm:
A:=Y;B=M;U:=X;V:=0;ρ:=n;
While ≠ρ0 do
While A mod2=O do
A:=A⁄(2;U:=U⁄(2 modM;))
ρ:=ρ-1;
End while
if (A+B)(mod 4)=0 then q:=1 else q:=-1;
A:=((A+qB))⁄4;U:=((U+qV))⁄(4 (mod M));
ρ:=ρ-1;
End while
if B=1 then Z:=V else Z:=M-V;

با توجه به الگوریتم بالا به راحتی می‌توان دید که $U \times Y = A \times X(mod\ M)$ وV×Y=B×X$(mod\ M)$

همانطور که در فرضیات الگوریتم مشخص است $gcd\ (Y, M) = 1$ پس اگر A=0 باشد۱- یاB=۱ خواهد بود. در مرحله نهایی الگوریتم

$$\boldsymbol{X(mod\ M)} \quad (۴)$$
$$Z = X\ /\ (Y(mod\ M))$$

می باشد.

## ۵- بهبود الگوریتم ضرب پیمانه ای مونتگمری با منطق تقسیم

برای سرعت بخشیدن به محاسبات به صورت= Z X.Y(modM) از روش مونتگمری استفاده می‌شود[ ۶و۷] . در این روش به جای محاسبه در پیمانه‌ی M یک عدد مثبت و حقیقی R بزرگتر از M را در نظر می‌گیرند، طوریکه R نسبت به M اول باشد. مقدارمثبت R معمولا به صورت عددی ازتوان دو($2^k$) درگرفته می‌شود وبا توجه به اینکه می‌توان ضرب و تقسیم برآن را با کمک جا به جایی یا انتقال[۱۶] بیتی و عملیات منطقی[۱۷] به راحتی انجام داد، محاسبات سریع و ساده خواهد شد.

با دانستن gcd (R, M) = 1 و با کمک گرفتن از دو عدد $R^{-1}$ ( معکوس R در پیمانه‌ی M ) و$\acute{M}$ (یک پیمانه ی کمکی) به طوریکه:

0≤R^(-1)≤M , 0≤M´≤R , R^(-1).R-M(M=1)´
(۵)

اگر عدد صحیح X کوچکتر از M انتخاب شود، آن‌گاه تبدیل یافته ی مونتگمری آن به صورت زیرمی‌شود:

X¯=X. R(modM), X<M
(۶)

می‌توان ثابت کرد که مجموعه‌ی{i.R(modM) ┤|0≤i≤M-1} ┤ مجموعه کامل باقیمانده‌ها است. بنابراین یک رابطه‌ی یک‌به‌یک بین عددی‌که بین۰ تاM-1 است و مجموعه فوق وجود دارد. الگوریتم‌مونتگمری از این‌خاصیت استفاده‌کرده و روش سریعتری را برای ضرب پیمانه‌ای ارائه می‌کند. اگر$\bar{X}$ و $\bar{Y}$را در نظر بگیریم ، رابطه ضرب مونتگمری به‌صورت زیر بیان می‌شود:

$$\bar{R} = \bar{X}.\bar{Y}.R^{-1}(modM) \Rightarrow$$
$$R = X.Y(modM) = \bar{X}.\bar{Y}.R^{-1}(modM).R$$
(۷)

و بادانستن قوانین

$$1.\ (X+Y)R = XR + YR$$
$$2.\ (X.Y)R = (XR).(YR)R^{-1}$$
(۸)

شبه کد الگوریتم مونتگمری پیمانه‌ای‌دودویی به صورت زیر می باشد:

Algorithm 2 (Algorithm for Modular Montgomery Multiplication)
Inputs: M: $2^{n-1} < M < 2^n$& gcd(M,2)=1
X,Y: $0 \le X, Y < M$
Outputs: $Z = XY2^{-n} modM$
Algorithm:
$A := Y; U := 0; V := X; \rho := n;$
While $\rho \neq 0$ do
$if\ A\ mod2 = 0\ then\ q = 0\ else\ q = 1$
A:= $(A - q)/2; U := (U - qV)/2\ modM;$
$\rho := \rho - 1;$
End while
$if\ U \ge M\ then\ Z := U - M\ else\ Z := U;$

حلقه while، پس از n بار تکرار پایان می‌یابد. برای بهبود الگوریتم اول، می‌توان از منطق استفاده شده در الگوریتم تقسیم پیمانه‌ای کمک گرفت. بنابراین یک شرط لازم است، در واقع نیاز به چک کردن بیت های کم ارزش A است. در صورتیکه دوبیت کم‌ارزش آن برابر صفر بود بدین معنی است که، A بر ۴ بخش‌پذیر است. در غیر اینصورت، اگر فقط کم ارزش‌ترین بیت آن صفر (عددزوج) بودA بر ۲ بخش پذیر است و درحالت سوم، کم ارزش‌ترین بیت آن برابر یک (عدد فرد)است.در اعداد زوج $A := A/2$ یا $A := A/4$ خواهد شد.در صورت فرد بودن عدد با قرار دادن معادل حقیقی[۱۸]آن، به حالات زیر می رسد:

[16] Shift
[17] Logical operation
[18] Integer

www.SID.ir

$$(a_1a_0) = 2 \times a_1 + a_0 = \begin{cases}1 \\ 3\end{cases} \ or \ = \begin{cases}-1 \\ -3\end{cases} \quad (۹)$$

برای حالاتی‌که مقدار حقیقیA برابر۱و۳- است، درصورت جمع با عدد منفی یک و در غیر این‌صورت با کم کردن از عدد ۱ مضربی‌از ۴ خواهند شد. با قرار دادن عدد ۱- در کم ارزش‌ترین بیت متغیر کمکیB می توان این قسمت از الگوریتم را پیاده سازی نمود.

Algorithm 3 (Fast Algorithm for Modular Montgomery Multiplication)
Inputs: M: $2^{n-1} < M < 2^n$& gcd(M,2)=1
X,Y: $-M \le X, Y < M$
Outputs: $Z = XY2^{-(n+1)} mod M \ (-M < Z < M)$
Algorithm:
$A := Y; B := -1; U := 0; V := X; M := M; \rho := n;$
While $\rho \neq 0$ do

$if\ [a_1a_0] = 0\ then$ A := $(A)/4; U := (U)/4\ modM;$ else$if\ [a_0] = 0\ then$ A := $(A)/2; U := (U)/2\ modM;$
else $if\ ([a_1a_0] + [b_1b_0])mod4 = 0\ then\ q := 1\ else\ q := -1;$
A:= $(A + qB)/4; U := (U + qV)/4\ modM;$
$\rho := \rho - 1;$
$endwhile$
$if\ U \ge M\ then\ Z := U - M\ else\ Z := U;$

## ٦- بهبود پیاده سازی الگوریتم ضرب پیمانه ای مونتگمری با کدینگ داده

در الگوریتم ضرب مونتگمری بالا، محاسبات تبدیل به یکسری عملیات جمع و تفریق و شیفت شد. با افزایش سرعت جمع می توان الگوریتم ضرب مونتگمری را بهبود بخشید.

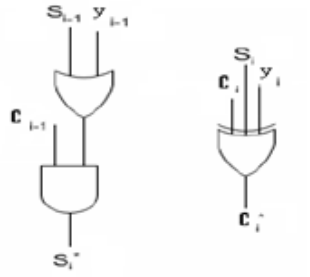
الگوریتمی برای جمع بیتی بدون انتشار رقم نقلی[19] مطرح می باشد[۵]، در این روش اعداد به صورت بیتی $A = \{a_n a_{n-1} \dots a_0\}$ و $a_i \in \{-1,0,1\}$ نمایش داده می‌شوند و انجام عمل جمع هم به صورت بیت به بیت دو عدد بوده. در حالتیکه دوبیت برابر باشند جمع معمولی و در غیر اینصورت یعنی حالتی‌که یکی از آنها صفر باشد با کمک متغیر کمکی $\dot{S}, \dot{Y}, \dot{C}$ و قرار دادن مقدار اولیهC=0 ، C,S,Y, Y=Y و S=X. حاصل جمع S=X+Y به صورت جدول۱ محاسبه می شود.

$for\ \forall i\ if\ a_i = b_i\ then\ s_i = x_i + y_i$ (۱۰)

**جدول۱. نمایش بیتی جمع سریع**

| i | xi | yi | S'i | Y'i | ci | S'i+y'i | si |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 1 | 10 | -1 |
| 2 | 1 | 0 | 1 | 1 | 1 | 10 | -1 |
| 3 | 1 | 1 | 1 | 1 | 0 | 10 | 0 |
| 4 | -1 | 0 | -1 | 0 | 0 | -1 | -1 |
| 5 | -1 | 1 | -1 | 1 | 0 | 0 | 0 |

در صورتیکه دوبیت ورودی جمع کننده با هم مساوی (طبق خاصیت CSA )رقم نقلی فقط در مرحله‌ی بعدی اثر گذاشته و در کل مدار منتشر نمی‌شود، در نتیجه تغییر خاصی ایجاد نکرده و جمع بیتی معمولی انجام می‌شود( فرمول۱۰). ولی اگر این دو بیت با هم برابر نباشند( یکی از دو بیت صفر و دیگری یک و یا منفی یک است) تنها فقط وقتی تغییر روش داریم که یکی از دوبیت صفر و دیگری یک باشد، که در این حالت هر دو بیت و بیت متناظر در $\dot{S}, \dot{Y}, \dot{C}$ ها برابر یک شده و حاصل جمع در این مرحله نیز برابر جمع دو بیت( $\dot{S_i} + \dot{Y_i}$ ) تفریق بیتی با $\dot{c}_i$ خواهد بود. برای واضح تر شدن روش کار مدار این جمع کننده در شکل۱ آورده شده است:




**شکل۲. جمع کننده سریع**

طبق توضیحات بالا و باتوجه به جدول۱، شبه کد جمع سریع به صورت زیر می باشد:

Algorithm 4 (Algorithm for Fast Multi-Adder)
Inputs: S,Y,C
X,Y: $0 \le X, Y < M$
Outputs: $\dot{S}, \dot{Y}, \dot{C}$
/ * $\emptyset$ is bitwise * / $S = (\dot{S} + Y)\dot{\emptyset}\dot{C}$
Algorithm:
$S := X; C := 0; Y := Y; \rho := n;$
For i=o to $\rho$ do
$if\ s_i = y_i\ then\ \dot{s}_i = \dot{y}_i, \dot{c}_i = 0;$,
Elseif $[s_i y_i] = [01\ or\ 10]\ then$
$[\dot{s}_i \dot{y}_i] = [11]\ [\dot{c}_i] = [1];$
Elseif $[s_i y_i] = [-10]\ then\ \dot{s}_i = \dot{y}_i;$
$\dot{c}_i = 0;$
$\rho := \rho + 1;$

[19] Carry propagate

www.SID.ir

Archvie of SID

$endif$

$S = (\dot{S} + Y)\dot{\phi}\dot{C}$

$endfor.$

برای سریع‌تر شدن الگوریتم ۳، که در بخش های قبل بحث شد. از جمع کننده سریع (الگوریتم۴)کمک خواهیم گرفت. همانطورکه در بالا توضیح داده شد، تصمیم‌گیری مبتنی بر کم‌ارزش‌ترین بیت A می‌باشد. در صورتیکه $a_0$ برابر صفر باشد(عددزوج است) دو حالت داریم:

$$a_0 = 0 \rightarrow a_1 = \begin{cases} 0 \rightarrow A\ (mod 4) = 0\ , (A := A/4\ ) \\ 1/-1 \rightarrow A(mod\ 2) = 0\ , (A := A/2) \end{cases}$$

**(۱۱)**

ودر غیر اینصورت و برای حالتی که A عددی فرد است(با توجه به فرمول ۱۱ مقدار معادلA محاسبه شده است)، به مجموعه حالات زیر

می رسد:

$$a_0 = 1 \rightarrow a_1 = \begin{cases} -1 \rightarrow A = -1 \\ 0 \rightarrow A = 1 \\ 1 \rightarrow A = 3 \end{cases}$$

$$\rightarrow A = \begin{cases} -3 \\ -1 \\ 1 \\ 3 \end{cases}$$

$$a_0 = -1 \rightarrow a_1 = \begin{cases} -1 \rightarrow A = -3 \\ 0 \rightarrow A = -1 \\ 1 \rightarrow A = 1 \end{cases}$$

**(۱۲)**

درحالتیکه A=-1,3 باشد، برای اینکه عدد مضربی از چهار باشد کافی است با عددی یک جمع شود. وقتی عدد A=1,-3 است عدد با منفی یک جمع خواهد شد. با توجه عملکرد الگوریتم جمع سریع (جدول۱والگوریتم۴) الگوریتم بهبودیافته ضرب پیمانه‌ای مونتگمری را خواهیم داشت.

در الگوریتم ضرب سریع، با توجه به اینکه در جمع‌های متوالی مقدار حاصل جمع هر مرحله عدد اول در حاصل جمع مرحله بعد بوده، هر بیت آن می‌تواند متعلق به مجموعه حالت$\{-1,0,1\}$ باشد و عدد دوم در این فرمول عددی است که در این مرحله می‌خواهد به حاصل جمع مرحله قبل اضافه شود و عددی باینری $\{0,1\}$ است(جدول ۱). لذا برای حالاتی از فرمول ۱۲که بیت های A مثبت بوده و قرار است از ۱ کم شود عدد اول را معادل 1- گرفته( $b_0 = -1$ و مابقی بیت های B برابر صفر مقدار دهی می‌شود) بنابراین جمع (A-۱-) را محاسبه می‌کنیم و اگر بیت‌های A برابر 1- باشند و لازم به محاسبه(A-1 ) باشد امکان پذیر نبوده، برای حل این مشکل کافی است هر دو عدد A,B را منفی نمود و محاسبه ی نهایی را نیز منفی کرد.

$$[a_0 a_1] = \begin{cases} [1-1] \\ [-1-1] \end{cases} \rightarrow (\mathrm{A}-1) \xrightarrow{\boldsymbol{Exchange}} -(-\boldsymbol{A}+\boldsymbol{1})$$

**(۱۳)**

می توان برای درک صحیح و کامل الگوریتم بهبود یافته ضرب مونتگمری با کمک جمع سریع مجموعه حالت و عملکرد متناسب هر یک از انها را در جدول زیر مشاهده نمود:

**جدول۲. جمع سریع در مونتگمری پیمانه ای**

| A | a | b | New s | New b | New c | s | New A | New c |
|---|---|---|---|---|---|---|---|---|
| 1 | [01] | [0$\bar{1}$] | - | - | - | [00] | $(a_0 = 0)$ | **0** |
| -1 | [0$\bar{1}$] | [01] | - | - | - | [00] | $(a_0 = 0)$ | **0** |
| 3 | [11] | [01] | - | [11] | [01] | [00] | $(a_1 a_0 = 00)$ | **1** |
| -1 | [ | [01] | - | [11] | [01] | [00] | $(a_1 a_0 = 00)$ | **0** |
| 1 | [1$\bar{1}$] → [$\bar{1}$1] | [0$\bar{1}$] → [01] | - | - | - | [00] | $(a_1 a_0 = 00)$ | **0** |
| -3 | [$\bar{1}$ $\bar{1}$] → [11] | [0$\bar{1}$] → [01] | - | [11] | [01] | [00] | t $(a_1 a_0 = 00)$ | 1 |

با توجه به جدول۲ می بینیم این روش پیاده سازی، منجر به تولید بیت صفر در کم ارزش ترین بیت‌های عدد ورودی می-شود. در واقع عددA به ۲A یا ۴Aتبدیل خواهدشد. این تبدیل برای سیستم رمز ECC قابل قبول می‌باشد و مشکلی در الگوریتم رمز ایجاد نمی‌شود، زیرا ورودی‌های الگوریتم مونتگمری می‌توانند بیش از M و کمتر از ۲M(فرمول۱۴)باشند و طبق فرمول ۶ همواره از M کوچکتر است [۸] در صورت دوبرابر شدن نیز از ۲M بزرگتر نخواهد شد. در الگوریتم بهبود یافته چون طول ورودی‌ها (زوج می‌شوند) یک بیت افزایش می‌یابد، بنابراین تعداد سیکل‌های مورد نیاز الگوریتم یکی اضافه می‌شود.

$$\boldsymbol{M \leq X, Y < 2M} \qquad \textbf{(۱۴)}$$

## ۷- بررسی صحت الگوریتم

از آنجا که با تغییر یک یا دو بیت از اعداد ورودی(طبق جدول۳)، به ترتیب یک یا دو سیکل و به طبع آن یک تقسیم بر دو یا چهار اضافه می شود. می توان طبق فرمول۱۵ صحت عملکرد الگوریتم را اثبات نمود

$$2^2 X \times Y \times 2^k \times 2^{-2} = X \times Y \times 2^k$$
$$2^1 X \times Y \times 2^k \times 2^{-1} = X \times Y \times 2^k$$

**(۱۵)**

www.SID.ir

همانطور که مشاهده می‌شود، نتیجه مورد انتظار، بدست آمده است. بنابراین الگوریتم جدید مونتگمری بدون تغییر در سیستم‌ها قابل استفاده می‌باشد

## ۸- برسی و مقایسه نتایج پیاده سازی الگوریتم بهبود یافته

برای پیاده سازی الگوریتم بهبودیافته‌ی مونتگمری، همانطور که در بخش های قبل بیان شد از الگوریتم جدید جمع سریع استفاده می شود. در این الگوریتم، جمع بیتی دو عدد X+Y با کمک متغیرهای کمکیS,C و به صورت $X + Y = (\dot{S} + \dot{Y})\emptyset\dot{C}$ امحاسبه می شود.

معماری که برای جمع کننده به جای CSA استفاده می شود که همان جمع سریع با کدینگ جدید منطبق با جدول۱ در شکل۱ نشان داده شده است. تلفیق این معماری با الگوریتم ضرب مونتگمری در منطق تقسیم پیمانه‌ای منجر به بهبود کارایی در حدود ۲۸٪ شده که نتایج این پیاده سازی در جداول زیر از نظر حافظه ی مصرفی و زمان پردازش در مقایسه با الگوریتم های مشابه آورده شده است.

**جدول۳. مقایسه زمان مصرفی الگوریتم بهبود یافته با الگوریتم مشابه**

| # of bit | Delay (ns) Sd adder | Delay (ns) booth adder | Time Improve % |
|---|---|---|---|
| 32 | 10.086 | 9.38 | 11.7 |
| 64 | 12.572 | 10.288 | 12.2 |
| 128 | 16.124 | 11.577 | 13.9 |
| 160 | 20.2 | 13.353 | 15.1 |

همانطور که ملاحظه شد، در بهبود الگوریتم ضرب مونتگمری با منطق تقسیم پیمانه ای و اعمال کدینگ جدید(جمع سریع)، سرعت پردازش حدود ۱۵٪ و فضای مصرفی ۲۱٪ بهبود داشته است.

**جدول۴. مقایسه حافظه مصرفی الگوریتم بهبود یافته با الگوریتم مشابه**

| # of bit | Slice in Sd adder | Slice in booth adder | Memory Improve % |
|---|---|---|---|
| 32 | 552 | 591 | 7 |
| 64 | 815 | 1023 | 19 |
| 128 | 1682 | 2050 | 18 |

## ۹- نتیجه گیری

با توجه به اینکه ویژگی مهم یک الگوریتم رمز پیچیدگی مناسب برای کاهش احتمال رمزگشایی توسط غیر است الگوریتم‌های رمز از کلیدهایی با طول بزرگ استفاده می کنند . این عمل منجر به افزایش حافظه و زمان مصرفی می شود و با کاهش طول کلید سطح امنیتی کاهش بیشتری پیدا می کند. سیستم رمزنگاری منحنی های بیضوی، حتی با استفاده از کلیدهای کوتاه سطح امنیتی مناسبی داشته و پیچیدگی در حد مسئله‌ی لگاریتمی دارد. به دلیل همین پیچیدگی بالای محاسبات، زمان مصرفی زیادی دارد و از طرفی دیگر چون محاسبات در خم های بیضوی مبتنی بر میدان های متناهی است.تمام عملیات پیمانه ای است. لذا استفاده از ضرب پیمانه ای مونتگمری در کاهش این زمان بسیار موثر می باشد.

دراین مقاله روش جدیدی برای ضرب پیمانه ای مونتگمری پیشنهاد شد که درالگوریتم جدید ضرب، فقط به شیفت و جمع نیاز داشت و برای محاسبه سریع تر ضرب از الگوریتم جمع سریع استفاده شد. در اکثر روش های پیاده سازی سخت افزاری برای جمع از معماریCSA که سریع ترین معماری موجود است، ( باعث کاهش اتلاف وقت برای دریافت رقم نقلی از واحدهای محاسباتی قبل می شود) استفاده می شود. ولی در این مقاله از روش جمعی با کدینگ جدید استفاده کردیم که علاوه بر کاهش فضای مصرفی به طور چشمگیر، افزایش سرعت پردازش را نیز خواهیم داشت. و در الگوریتم های رمزنگاری که سرعت محاسبات وابستگی زیادی به زمان پردازش ضرب دارد و از طرفی دیگر عمل ضرب یک سری جمع متوالی است با کاهش زمان محاسبات جمع، کارایی الگوریتم ضرب که طبق فرمول۱۶ محاسبه می‌شود، نیز بهبود می‌یابد. . برای این منظور، از روشی استفاده شد که علاوه بر داشتن ویژگی CSA که همان جلوگیری ازانتشار رقم نقلی و در نتیجه کاهش تاخیر است، کاهش فضای مصرفی را نیز دارد، استفاده شده است. در این روش کارایی که همان ضرب فضای مصرفی در زمان پردازش است کاهشی حدود ۲۸٪ داشته است. از ضرب کننده جدید می توان در سیستم های رمز نگاری که نیاز به ضرب های پیمانه ای دارند، مانندسیستم رمز منحنی های بیضوی(ECC) استفاده کرد..

$$\text{PERFORMANCE}={}^{1}/_{(area \times time)} \quad (۱۶)$$

## مراجع

[1] Z. Dyka, P. Langendoerfer "Efficient Hardware Implementation of Elliptic Curve Cryptography by Iteratively Applying Karatsuba's Method" Proceedings of the conference on Design, Automation and Test in Europe, Volume 3, Pages: 70 – 75, 2005.
[۲] T. Wollinger, J. Pelzl, V. Wittelsberger, C. Paar , "Elliptic and Hyperelliptic Curves on Embedded μP, ACM Transactions on Embedded Computing Systems." (TECS), August 2004.
[3] J.C.Bajard, S.Duquesne,M.Ercegovac, "Combining Montgomery Ladder for elliptic curves

www.SID.ir

Archvie of SID

defined over $F_p$ and RNS representation", UCLA,Computer Science Dep , Los Angeles,US, September 24, 2007.
[4] S. Park, "Hardware design of scalable and unified modular division and Montgomery multiplication", Oregon State University, Oregon, USA, June 2005.
[5] K.Manochehri , B.Sadeghian & S.Pourmozafari, "Very Fast Multi-Operand Addition Method by Bitwise subtraction" , EEE Information Tecnology, April 2008.
[6] P.L.Montgomery, "Modular Multipication without trial division, Math.Computation", VOL.44,1985.
[7] D. J. Guan, "Montgomery Algorithm for Modular Multiplication" , Math.Computation ,August 25, 2003.
[8] Al. F. Tenca, and K. Koc, "A Scalable Architecture for Modular Multiplication Based on Montgomery's Algorithm", IEEE Transaction on computer, VOL. 52, NO. 9, september 2003.

www.SID.ir