هشتمین کنفرانس بین المللی انجمن رمز ایران
دانشگاه فردوسی مشهد - ۲۳ و ۲۴ شهریور ۱۳۹۰

تسریع رمزنگاری تصویر با استفاده از الگوریتم های موازی آشوبی بر روی پردازش گرهای گرافیکی

احسان خان میرزا[۱]، محسن نساجی[۲]، عبداله چاله چاله[۳]، مهرداد احمدزاده راجی[۴]

[۱] دانشجوی ارشد، گروه معماری کامپیوتر، دانشگاه رازی، کرمانشاه،
Khanmirzaa@gmail.com

[۲] دانشجوی ارشد، گروه معماری کامپیوتر، دانشگاه رازی، کرمانشاه،
Mohsen_nassaji@ec.iut.ac.ir

[۳] استاد یار، گروه معماری کامپیوتر، دانشگاه رازی، کرمانشاه،
Chalechale@razi.ac.ir

[۴] استادیار، گروه ریاضی، دانشگاه رازی، کرمانشاه
mraji@razi.ac.ir

چکیده

در سال های اخیر الگوریتم‌های متنوعی در زمینه رمزنگاری تصویر، بر پایه توابع آشوب پیشنهاد شده‌اند. دو نکته اساسی در الگوریتم های رمزنگاری تصویر، سرعت و امنیت می باشند. تا به حال سرعت اجرای الگوریتم‌های رمزنگاری تصویر، بر روی پردازنده‌های گرافیکی محاسبه و مقایسه نشده است. در این مقاله الگوریتم رمزنگاری بر پایه توابع آشوب، که ویژگی‌های لازم برای اجرا بر روی پردازنده‌های گرافیکی را داشته باشد، پیشنهاد شده است.در این الگوریتم، از نگاشت Kolmogorov برای به هم ریختن مکان پیکسل‌های تصویر و از تابع لجستیک برای جایگزینی مقدار پیکسل‌های تصویر استفاده شده‌است. الگوریتم پیشنهادی، بر روی پردازنده گرافیکی پیاده‌سازی شده است. نتایج نشان می دهد پیاده سازی این الگوریتم بر روی پردازنده گرافیکی با ابزار برنامه نویسی CUDA، در مقایسه با پیاده‌سازی الگوریتم‌های ترتیبی بر روی پردازنده دو هسته ای، ۱۰۰۰۰ برابر سریع‌تر می باشد که به دلیل اجرای همزمان هزار ها ریسمان می باشد. در این مقاله، فاکتور های امنیتی جهت ارزیابی الگوریتم پیشنهادی در نظر گرفته شده است.

کلمات کلیدی

امنیت، پردازشگر گرافیکی، توابع آشوب، رمزنگاری تصویر، سرعت.

۱- مقدمه

با رشد سریع تصاویر دیجیتال و پخش گسترده آن بر روی اینترنت، محافظت از اطلاعات دیجیتال در برابر کپی و توزیع غیر مجاز، اهمیت بیشتری پیدا می کند. در بسیاری از موارد، کانال ارتباطی، قادر به ایجاد امنیت در مقابل نفوذ گرها نمی باشد. در همین راستا روش‌های متنوعی برای رمزنگاری تصویر بر پایه آشوب پیشنهاد شده اند[1,2].

رفتار شبه‌نویز، حساسیت زیاد نسبت به کلید و سادگی پیاده سازی، از جمله مزیت‌های سیگنال‌های آشوب در رمزنگاری تصویر می باشند.علاوه بر مزیت‌های بیان شده، یکسری نگاشت‌های آشوبی دو بعدی وجود دارند، که بسیار مناسب برای جایگشت پیکسل های تصویر می باشند. Picher و Scharinger روشی برای جایگشت پیکسل های تصویر با استفاده از نگاشت Kolmogorov ارائه داده‌اند[3].

Fridrich روش قبلی را برای کاربردهای عمومی توسعه داد. سپس Chen et al روشی جدید برای رمزنگاری تصویر، با استفاده از نگاشت سه بعدی Cat Map ارائه داد. Lian et al نیز روش دیگری بر پایه الگوریتم نگاشت استاندارد ارائه داد[4].

همه الگوریتم‌های رمزنگاری‌آشوبی تصویر، شامل دو قسمت می باشند: جایگشت پیکسل های تصویر: مکان پیکسل‌ها بر پایه توابع آشوب تغییر می کنند.

عمل جایگزینی: مقدار پیکسل های تصویر بر پایه توابع آشوب یکی پس از دیگری و وابسته به مقدار قبلی پیکسل تغییر می کند.

این دو مرحله آنقدر تکرار می شوند تا تصویر رمزنگاری شده، تولید شود.

دو مرحله اشاره شده، توان بسیار مناسبی جهت رمزنگاری تصویر دارند، اما قابلیت پیاده‌سازی موازی را ندارند و این به این دلیل است که مقدار جدید پیکسل وابسته به مقدار قبلی پیکسل می باشد. حتی اگر پردازنده‌ی چندهسته‌ای هم وجود داشته باشد، محدودیت هایی که توابع آشوب ایجاد می کنند، باعث می‌شود برنامه به صورت ترتیبی و روی یک هسته اجرا شود. نکته مهم دیگری که می تواند سرعت اجرای رمزنگاری تصویر را کاهش دهد، تعداد زیاد ضرب و تقسیم های ممیز شناور می باشد.

برای حل مشکلات فوق، Qing Zhou et al ایده بلاک‌بندی تصویر را پیشنهاد داد. بااستفاده از این ایده، ابتدا از نگاشت Kolmogorov برای جایگشت پیکسل‌های تصویراستفاده می‌شود، سپس تصویر به تعدادی بلاک تقسیم می‌شود، و از S-box برای جایگزینی پیکسل های داخل هر بلاک استفاده می شود[4].

Kwok-ko نیز با استفاده از عمل Add and Shift هنگام جایگشت پیکسل‌های تصویر باعث تسریع در عملیات رمزنگاری شد[5]. در این مقاله از نگاشت استاندارد برای جایگشت پیکسل ها ی تصویر استفاده شده است. سپس قبل از ورود به مرحله جایگزینی عملیات ساده Add and Shift هر پیکسل با مقدار قبلی در همان مکان انجام می شود. سپس عمل جایگزینی با استفاده از نگاشت لجستیک انجام می شود. روش ارائه شده، زمان مورد نیاز برای برای رسیدن به فاکتور های امنیتی مناسب را نسبت به روش Lian et al [5]، ۱/۳ برابر کرده است.

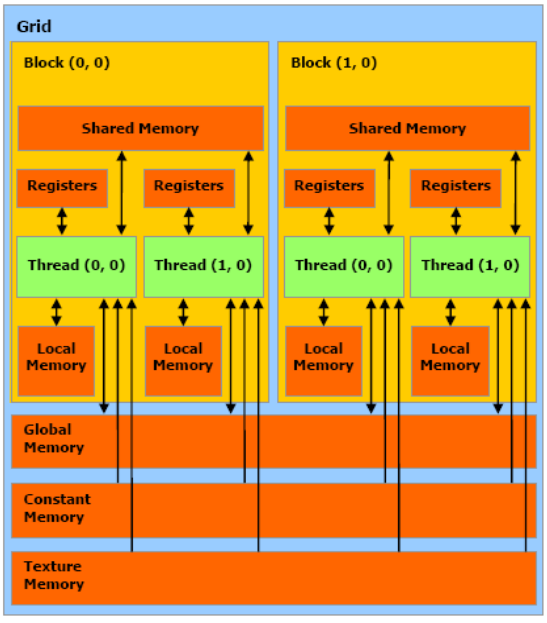
Yong wong نیز روشی را پیشنهاد کرد که عمل جایگشت و جایگزینی تصویر، در یک بار پیمایش تصویر انجام شود[6]. در این روش ابتدا تصویر به بلاک هایی تقسیم می شود. سپس با استفاده از تابع آشوب فضایی، همزمان هم جایگشت بلاک ها و جایگزینی پیکسل های تصویر انجام می شود. عامل دیگر تسریع روش پیشنهادی استفاده از تابع آشوب فضایی برای تولید اعداد تصادفی می باشد.

در این مقاله، الگوریتمی جهت رمزنگاری تصویر ارائه شده‌است که در عین استفاده از توابع آشوب، قابلیت اجرا بر روی پردازنده‌های گرافیکی را دارد. در مرحله جایگشت پیکسل های تصویر از تابع آشوب Kolmogorov به دلیل قابلیت توازی استفاده شده است. با استفاده از قابلیت پردازنده های گرافیکی برای هر پیکسل نیز یک ریسمان تعریف شده است. در مرحله جایگزینی پیکسل های تصویر تابع لجستیک به کار گرفته شده که در این مرحله هر ریسمان مسئول اجرای هر بلاک می باشد.

به دلیل نبودن معیار مناسب در اندازه‌گیری سرعت در رمزنگاری تصویر، زمان اجرای الگوریتم فوق بر روی پردازنده‌گرافیکی و همچنین، زمان اجرای الگوریتم ترتیبی بر روی پردازنده‌ی مرکزی، اندازه‌گیری و مقایسه شده‌است. در بخش دوم پردازنده‌های‌گرافیکی مورد بحث قرار خواهند گرفت. در بخش سوم الگوریتم پیشنهادی ارائه خواهد شد و در بخش نتایج تجربی، کارایی روش پیشنهادی از نظر امنیت و سرعت مورد ارزیابی قرار خواهد گرفت.

۲- نقش پردازنده گرافیکی در پردازش تصویر

پردازنده‌های گرافیکی (GPU) با معماری SIMD امکان اجرای همزمان صدها ریسمان را با ابزار برنامه نویسی CUDA فراهم می کنند[7]. پیاده سازی‌های برداری یکسان با CUDA، به علت توان عملیاتی بالای واحد اجرای برداری، بسیار کارا است [8]. هر پردازنده گرافیکی شامل چندین واحد پردازش مستقل SIMD به نام MP است. CUDA جهت اجرای کد بر روی پردازنده‌های گرافیکی، ریسمان‌ها را به صورت یک Grid، به پردازنده گرافیکی ارسال می‌کند. هر Grid، متشکل از چندین بلاک و هر بلاک شامل چندین ریسمان می‌باشد[9]. CUDA به هر بلاک یک شناسه یکتا نسبت می دهد (این شناسه در زمان اجرا قابل دسترسی است). هر MP مجموعه ای از رجیستر فایل ها، حافظه مشترک، کش داده و کش فقط خواندنی می باشد. ریسمان‌های داخل بلاک می‌توانند داده‌هایشان را بوسیله حافظه‌ی‌مشترک با سرعت دسترسی بالا به اشتراک بگذارند. شکل ۱ معماری داخلی پردازنده گرافیکی را نشان می دهد .



شکل ۱: معماری داخلی پردازنده گرافیکی

ابزار برنامه نویسی CUDA، بستر بسیار مناسبی برای پردازش تصویر ایجاد می‌کند. می‌توان پیکسل‌های تصویر را بر روی هر ریسمان نگاشت‌کرد و ریسمان‌ها به صورت موازی عملیات مربوط به هر پیکسل را اجرا کنند. در [10] روشی برای تسریع در فشرده سازی تصویر بر روی پردازنده‌های

گرافیکی پیشنهاد شده‌است. در [11] نیز احراز هویت بر پایه امضای دیجیتالی و نهان‌نگاری‌تصویر بر روی پردازنده‌گرافیکی NVIDIA's Tesla C1060 پیاده‌سازی شده‌است. با توجه به مستقل بودن پیکسل‌های تصویر در بیشتر الگوریتم‌های پردازش تصویر، می‌توان محاسبات با حجم سنگین را به طور موازی و در زمان کمتری انجام داد.

۳- الگوریتم پیشنهادی

در این الگوریتم، معماری رمزنگاری تصویر با استفاده از توابع آشوب، شامل دو بخش جایگشت و جایگزینی پیکسل‌های تصویر می باشد.

در مرحله جایگشت، از نگاشت Kolmogorov استفاده می‌شود، که مکان پیکسل‌های تصویر را به هم می‌ریزد. مدل گسسته تابع Kolmogorov در معادله ۱ بیان شده است.

$$T_{n,S}(x,y) = (q_s(x - F_s) + (y \bmod q_s), F_s + (y \operatorname{div} q_s))$$

$$F_s = \begin{cases} 0 & s=1 \\ n_1 + n_2 + \cdots, +n_k & s = 2,3,\ldots,t\text{-}1 \end{cases}$$

(۱)

x و y مختصات جدید پیکسل هستند وبا استفاده از مکان فعلی پیکسل بدست می آیند، S=1,2,3,... ، n_s اعداد صحیح مثبت می‌باشند که مجموع n_s ها برابر N می‌شود (اندازه تصویر N*N است)، $q_s=1/n_s$ می‌باشد و F_s از رابطه‌ی بالا بدست می‌آید.

در این الگوریتم، همان‌طور که در معادله ۲ ارائه شده است یک تابع Logistic Map اعداد تصادفی بین ۰ تا ۱ را تولید می‌کند. اعداد تصادفی بدست آمده با مقدار فعلی پیکسل XOR می شوند.

$$\begin{cases} C_{-1} = K_d \\ C_i = V_i \oplus q[f(C_{i-1}), L] \end{cases}$$
$$f(C_i) = \mu * C_{i-1} * (1 - C_{i-1})$$
$$q(X, L) = \lfloor 2^L * X \rfloor$$

(۲)

در معادله ۲، V_i مقدار پیکسل iام در تصویر می‌باشد و C_i ارزش پیکسل iام در تصویر رمزنگاری شده می‌باشد. تابع f تابع لجستیک می‌باشد و تابع q، ۱۶ رقم بعد از اعشار را به روش مشخص شده در معادله ۲ جدا می‌کند. K_d کلید رمزنگاری و L مقدار ثابت ۵۶ است.

در مرحله جایگزینی، ارزش پیکسل‌ها با استفاده از یک روش ترتیبی تغییر می کنند و این به معنای اجرای ترتیبی نگاشت Logistic است. برای حل مشکل ترتیبی اجرا شدن و ایجاد توازی جهت افزایش سرعت اجرای الگوریتم ، تصویر را به بلاک‌های ۸*۸ تقسیم می کنیم و الگوریتم ترتیبی نگاشت Logistic را برای هر بلاک به صورت مستقل از بلاک‌های دیگر اجرا می‌کنیم، با استفاده از این تقسیم‌بندی اجرای نگاشت Logistic برای هر بلاک را با یک ریسمان مجزا انجام خواهیم داد.

بازدهی برنامه‌های پیاده سازی شده توسط CUDA، بدلیل نبود مدیریت کد در کامپایلر آن، بشدت به نوع الگوریتم استفاده شده و استراتژی استفاده از حافظه وابسته است. به همین دلیل برای پیاده‌سازی الگوریتم فوق در مرحله اول نیاز به مناسب سازی آن، جهت پیاده‌سازی توسط CUDA است. در مرحله بعد بهترین استراتژی جهت استفاده از حافظه انتخاب می‌شود. الگوریتم فوق را می توان به دو قسمت، یک قسمت جایگشت پیکسل‌ها و قسمت دیگر جایگزینی پیکسل‌ها تقسیم کرد. دلیل جداسازی این دو قسمت، درصد توازی آنهاست. قسمت جایگشت پیکسل‌ها را می‌توان به صورت کاملا موازی اجرا کرد در حالیکه در قسمت جایگزینی همان‌طور که پیشتر ذکر شد، تصویر به بلاک هایی تقسیم بندی می‌شود و بلاک ها به طور موازی اجرا شوند.

در ابتدا تصویر توسط پردازنده مرکزی به یک ماتریس بلاک بندی شده تبدیل می گردد که هر درایه آن حاوی سه مقدار P، M و n_s است که P مقدار عددی پیکسل، M شماره بلاکی که پیکسل در آن قرار دارد و n_s نیز مقدار ذکر شده در معادله ۱ است.

این ماتریس از حافظه اصلی پردازنده به حافظه Global پردازنده گرافیکی منتقل می شود. سرعت ارتباط حافظه Global با پردازنده گرافیکی حداقل ۲۰ برابر سریعتر از سرعت ارتباط حافظه اصلی با پردازنده مرکزی است.

در قسمت جایگشت پیکسل ها، به ازای هر پیکسل یک ریسمان ایجاد می شود. ریسمان مورد نظر، داده های درایه مورد نظر پیکسل مربوطه را از حافظه Global پردازنده‌گرافیکی به حافظه Shared پردازنده‌گرافیکی، که جزء سریع ترین حافظه های موجود در پردازنده گرافیکی است و سرعت دسترسی آن برابر سرعت پردازنده‌گرافیکی است، منتقل می کند. سپس محاسبات مربوطه جهت جایگزینی پیکسل، با سرعت بسیار زیاد اجرا می‌گردد. قبل از انتقال پیکسل‌ها به مختصات جدید از تابع Syncthread جهت جلوگیری از ازدست رفتن پیکسل‌ها استفاده می شود. هنگامی که یک ریسمان به تابع Syncthread می‌رسد، اجرای آن متوقف می‌شود تا زمانی که تمامی ریسمان‌ها به اجرای Syncthread برسند و پس از آن اجرای تمامی ریسمان‌ها ادامه می‌یابد. پس از پایان کار تمام ریسمان ها، کنترل اجرا به پردازنده مرکزی باز می گردد تا تنظیمات جدید برای پردازنده گرافیکی جهت اجرای قسمت جایگزینی انجام گردد.

در این مرحله به ازای هر بلاک از تصویر یک ریسمان ایجاد می گردد. هر ریسمان اطلاعات مورد نیاز خود را از حافظه Global پردازنده‌گرافیکی فراخوانی کرده و با استفاده از حافظه Shared پردازنده‌گرافیکی به رمزگذاری بلاک مورد نظر خود می پردازد.

پس از پایان کار تمامی ریسمان ها، کنترل بار دیگر به پردازنده مرکزی باز می گردد و ماتریس داده ها نیز از حافظه Global پردازنده‌گرافیکی به حافظه اصلی منتقل می شود.

۴- نتایج تجربی

در این قسمت با پیاده سازی الگوریتم پیشنهادی بر روی عکس Lena، تفاوت سرعت اجرا بر روی پردازنده‌مرکزی و پردازنده گرافیکی نشان داده شده‌است. در این آزمایش، عکس خاکستری Lena با ابعاد ۲۵۶*۲۵۶ پیکسل که در شکل۲ دیده می شود، به عنوان تصویر اصلی در نظر گرفته شده‌است.

کلید در الگوریتم رمزنگاری تصویر، حالت اولیه K_d و پارامتر μ می باشد که به صورت اعشاری و با دقت ۵۶ بیت در نظر گرفته شده است.

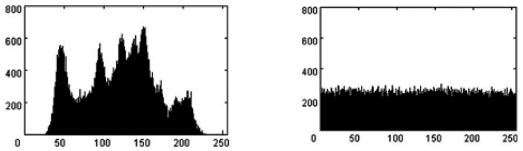
شکل ۲: تصویر Lena و تصویر رمز گزاری شده (از چپ به راست)

پس از اتمام رمزنگاری تصویر، تضمین امنیت و سرعت دو نکته اساسی می باشند. امنیت الگوریتم پیشنهادی از لحاظ تحلیل آماری، تحلیل حساسیت این روش نسبت به تغییرات کلید و تحلیل فضای کلید، مورد بررسی قرار گرفته است، که نتایج تجربی آن در جدول ۱ آورده شده است.

ضرایب همبستگی	افقی	-۰.۰۰۳۱
	عمودی	-۰.۰۲۵۰
	قطری	-۰.۰۰۰۷
حساسیت به کلید	NPCR	۰.۹۹۵۴
	UACI	۰.۳۳۱۳
آنتروپی اطلاعات	۷.۹۸۹۴	
فضای کلید	$۲^{۲۰۹}*۲^{۱۱۲}$	

جدول ۱: تحلیل الگوریتم پیشنهادی از لحاظ امنیتی

همچنین هیستوگرام تصویر رمزنگاری شده، هیستوگرام یکنواختی می باشد که در شکل ۳ نشان داده شده است.



شکل ۳: هیستوگرام تصویر Lena و هیستوگرام تصویر رمز شده (از چپ به راست)

الگوریتم رمزنگاری آشوبی، جهت رمزنگاری تصویر پیشنهاد شده، بر روی پردازنده گرافیکی GeForce 9800 GTX پیاده سازی شده است. الگوریتم پیشنهادی بر روی سیستم عامل windows 7 و محیط برنامه نویسی CUDA نسخه‌ی ۲.۳ اجرا شده است. جدول ۲ میانگین زمان اجرای الگوریتم پیشنهادی بر روی پردازنده گرافیکی GeForce 9800 GTX در مقابل اجرا بر روی پردازنده‌مرکزی را نشان می دهد. این افزایش سرعت چشم‌گیر (۱۰۰۰۰ برابر) به دلیل توان پردازنده‌گرافیکی در اجرای همزمان هزاران ریسمان می‌باشد.

CPU dual core E5200	GeForce 9800 GTX
2.11ms	0.2µs

جدول ۲: تفاوت سرعت اجرا الگوریتم پیشنهادی بر روی CPU و GPU

۵- نتیجه گیری

در این مقاله یک روش رمزنگاری تصویر با استفاده از توابع آشوب که قابلیت اجرا بر روی سکوی موازی داشته باشد ارائه شد. در این روش، از نگاشت Kolmogorov برای جایگشت پیکسل های تصویر و تابع لجستیک برای جایگزینی پیکسل های تصویر استفاده شده است. سرعت بخشیدن به جریان پردازش با بکارگیری قدرت پردازش موازی که توسط پردازنده‌های گرافیکی حاصل شده است باعث افزایش سرعت اجرای الگوریتم‌های پیشنهادی شده است. با پیاده سازی الگوریتم فوق بر روی پردازنده دوهسته ای و پردازنده گرافیکی نشان داده شد که نه تنها الگوریتم فوق از امنیت لازم برخوردار است بلکه پیاده سازی آن بر روی پردازنده گرافیکی و استفاده از ریسمان ها می تواند سرعت اجرا را تا حدود ۱۰۰۰۰ برابر نسبت به پردازنده دو هسته ای افزایش دهد.

مـر اجع

[1] Gao H, Zhang Y, Liang S, Li D, “A new chaotic algorithm for image encryption”. Chaos, solution & Fractals 2006;29(2)393-9.
[2] Pareek NK, Patidar V, Sud KK, “Image encryption using chaotic logstic map”. Image Vision Comput 2006;24(9):926-34.
[3] Chen D, Jifang L, Lifeng X, Jie H, Ran J,”Improvement of an image encryption algorithm based on combined multidimensional chaotic system”. Lecture Notes in Computer Science, 2007, Volume 4693/2007, 209-216.
[4] Qing Z, Kwok-wo W, Xiaofeng L, Tao X, Yue H,” Parallel image encryption algorithm based on discretized chaotic map”. Chaos, Solitons & Fractals Volume 38, Issue 4, November 2008, Pages 1081-1092
[5] Kwok-wo W, Bernie S, Wing-Shing,”A fast image encryption scheme based on chaotic standard map”. Physics letters A 372, 2008, pp. 2645-2652
[6] Yong W, Kwok-wo W, Xiaofeng L, Guanrong C,”A new chaos-based fast image encryption”. Applied soft computing 11, 2011,pp. 514-522.
[7] W. Liu, B. Schmidt, G. Voss, W. Mullerwitting, “ streaming algorithms for biological sequence alignment on GPUs”. IEEE Trans. Parallel and distributed systems, 2007, pp. 1270-1281
[8] NVIDIA Company, (2007): NVIDIA CUDA compute unified device architecture programming guide version 1.1.
[9] N. Leischner, Osipov, P. Sanders,”GPU Sample sort”. IEEE Trans. Parallel & distributed processing,2010.
[10] V.Simek, R. Asn,” GPU Acceleration of 2D-DWT Image Compression in MATLAB with CUDA”. IEEE Trans. Computer Modeling and Simulation, 2008, pp 284.
[11] L.Caiwei, Z. Lei, Y. Jiwen,” A CUDA Based Implementation of an Image Authentication Algorithm ”. IEEE Trans. Information Engineering and Computer Science (ICIECS), 2010 ,Pp.2156-7379.