International Journal of Scientific Research & Engineering Trends
Volume 6, Issue 6, Nov-Dec-2020, ISSN (Online): 2395-566X

Digital Signature Server Using Elliptic Curve Cryptograpic Algorithm

S.B.Sangeetha, Jasmine Sait Mohmmad, Praba.M
St Joseph's Institute of Technology, Chennai-119
sbs121296@gmail.com. prabait39@gmail.com. jasmiit107@gmail.com.

Abstract **- Application servers are heavily loaded by performing signature computation Instead of using Application server which act as the tenant to more dedicated proxy called Signature servers called GUESSThus to improve the performance of the GUESS comprehensive implementation of the EDSA is tested against network trafficDigital signatures are encrypted and decrypted using Elliptical curve cryptographic algorithm.**

Index Terms- **Data communication, Application, Cryptography**

I. INTRODUCTION

To implement Signature as a service (SIGaaS) in the enterprise private cloud Application servers are offloaded from signature computation (Signature generation and Signature verification) Security practitioners who out source signature computation from application server can use signature server for secure computation.

A. Signature as a Service

In this paper, following the business model of security as a service (SECaaS) [2], we focus on signature as a service(SIGaaS) as a case study, and present a tangible signatureserver called Guess. This is motivated by the observation that it has become a more and more common practice for security practitioners to outsource signature computations from application servers that are typically websites to dedicated proxies that are typically signature servers in the enterprise private cloud (EPC), so that the application servers as tenants can offload the computation costs of signature generation and/or verification to these specialized facilities.

Again take Alipay as an example, which was started by Alibaba Group as a simple e-payment system and is now a website in its own right; it is possible that Alibaba adopts SIGaaS in its EPC to facilitate Alipay and Alibaba's other subsidiaries (e.g., Taobao and Tmall). Besides the common (and widely-recognized) benefits offered by SECaaS (and cloud computing in general), SIGaaSprovides additional advantages in terms of security.First of all, application servers like Alipay have to be openly accessible from the Internet and thus are potentially exposed to various attacks, some of which, like the Heartbleed exploidisclosed in 2014, may reveal websites' private keys, i.e.,signing keys. With SIGaaS, the private keys are confinedto standalone signature servers instead of being held by thepublic application servers. The input/output interfaces of asignature server are fairly simple: it accepts a message digestfrom and returns a generated signature to a tenant employingits private key, and accepts a digest-signature pair from and returns a boolean verification result to a tenant employing aspecified public key. Therefore, the private key for signing(even the entire signature server) is beyond the "reach" ofInternet attackers. This significantly mitigates the risk of keycompromise.Second, by shifting signing operations to signature servers, tenants are far less vulnerable to "stealthy" threats like side channe attacks. For example, it would be very difficult for anInternet attacker to launch timing attacks to infer a tenant's privatekey, as the actual signing is shifted to a back-end signatureserver, typically located in the EPC serving multiple tenants.

In this paper, as our concrete implementation of SIGaaS, we present a GPU-accelerated universal elliptic-curve signature server, Guess for short. It is universal in that Guess supports various ECC (signature) schemes where scalar multiplication,also known as point multiplication (PM), is the chief cryptographic primitive. These includes ECDSA [6], EC-KCDSA [19, §4.4.2], the SM2 signature [20], and so on. EC-KCDSA is Korea's signature scheme, while SM2 is an ECC algorithm suite for digital signature, key agreement, and encryption,issued in late 2010 as the public key cryptographic standard in China.

In fact,Guess readily supports the entire SM2 suite. Among these, we choose ECDSA, the most widely standardized ECC signature scheme [19, §4.4.1], as the "target scheme" for Guess so that we can compare our work with a wide range of exGuess is designed to be a standalone signature serverwhose capability is delivered via high-speed network to tenantstypically located in a well-protected EPC (thus, in this paperwe do not address issues like authentication between tenantsand Guess, or transfer of private keys from tenants to Guess).

Due to budget constraints, our platform is merely a combinationof off-the-shelf commodity hardware and freely availablesoftware: it contains one 2.7GHz 12-core

Intel Xeon E5-2697v2 CPU [21], one NVIDIA GeForce GTX 780 Ti desktop GPUon a video card by EVGA [18], one 10Gb Ethernet controller,and 16GB DRAM; the OS is the 64-bit server edition ofUbuntu 12.04 and the compiler is from CUDA
.

II. ECDSA IMPLEMENTATION

This section details our comprehensive implementation ofECDSA. From a developer's perspective, our technique canbe decomposed into two levels (cf. Section I-D): theghlevelalgorithms translate mathematics into general softwaredescriptions, which are then supported by the lowlevel algorithmsconsisting of assembly-like instructions for the GPU.The high-level algorithms mainly address PMs, where wepursue solutions that are even more efficient than standard (ortermed as textbook) accelerations. The low-level algorithmsfocus on memory optimizations [10, 9].

1. Pre-Computing Table Aided Fixed Point Multiplication

PM dominates the execution time of ECC schemes includingECDSA. Clearly, a tenant cares most about the service latency,but is blind to the ways Guess implements ECDSA operations.Our implementation choices are not necessarily limitedto the many textbook approaches, but have to suit and wellharness the given platform. Particularly, our device, the EVGAvideo card [18], has affluent (3GB) global memory, only asmall part of which should be reserved for data exchangebetween the host and the device (cf. Section II-C). As a result,for fixed point multiplication, we choose to trade storage forcomputation, employing a precomputing table (PCT) builtoffline w.r.t. the fixed point G.Should Guess support different ECC signature schemes fordifferent tenants, a PCT would have to be built for each curve.For ECDSA with Curve P-256, only one PCT is needed, whichcan be generated using any approach and saved in a datasheet only once. Upon booting, Guess loads the entire PCTinto the device's global memory for readonly access later. Thelarger the PCT, the faster the fixed point multiplication, andour experiments show that a 64MB PCT is beneficial enoughfor acceptable latency. In theory, using an even larger PCT wemay boost Guess' performance forthrightly, but we find thatthe bottleneck of Guess then lies in service delivery ratherthan PM or other cryptographic processing (thus employingan even larger PCT does not help much in itself).

Now let us be more specific. Take kG for example.
Let k =kl−1|| . . . ||k1||k0 be its radix-2f form, i.e., k =Pl−1
i=0 2ifki
, where |ki| = f, fl & |k| = |n|.
Since kG =Pl−1
i=0 2ifkiG, thePCT needs to contain 2ifkiG for all 0 ki 2f−1, 0 i l−1, altogether 2f l items, with each item of size |(x, y)| = 2|n| bits.

Since |n| = 256,
we can select f = l = 16, resulting ina PCT of size 2|n|↠2f l bits, which is 64MB.

Now that each2ifkiG is immediately available via table lookup,
the fixedpoint multiplication kG =Pl−1i=0 2ifkiG is simply reduced tol−1 point additions.

Such reduction also prevents the noncek (and thus the private key d) from being leaked to a sidechannelattacker [24], [25] thanks to the table lookups.As mentioned earlier, we address the implementation ofECDSA on Guess from two levels. Next, we switch from thefixed point multiplication on the high level to point additionson the low level. We begin with the general case in SectionIII-B, and then consider a special case in Section III-C that isactually invoked when computing kG =Pl−1i=0 2ifkiG.

2. Optimized Jacobian Addition

To calculate point addition efficiently, we consider convertingthe chord-and-tangent rule [19, §3.1.2] into a lowlevelalgorithm in the Jacobian system (cf. Section II-A). This involves certain formula deduction. For page limits, we omit the mathematics and describe in Table I our intermediateresult, a "not-too-low-level" point addition working flow, thecorrectness of which can be easily checked against the chordand-tangent rule. Note that point addition and point doubling(to be addressed later) are distinct; they follow different laws.

Table I Lists The Variables That Need To Be Stored After Each Step And Shows That Our Working Flow Is Very Variableefficient:
TABLE I
POINT ADDITION IN THE JACOBIAN SYSTEM, WHERE Pi = (Xi : Yi : Zi)
2 F3
P. INPUT: P1, P2(6= ±P1). OUTPUT: P3 = P1 + P2.
Step (Calculation Over Fp) Variables That Need Saving
Load P1, P2 X1, Y1, Z1,X2, Y2, Z2
R1 = X1Z2
2 R1, Y1, Z1,X2, Y2, Z2
R2 = X2Z2
1 R1,R2, Y1, Z1, Y2, Z2
R3 = R2 − R1 R1,R3, Y1, Z1, Y2, Z2
R4 = Y1Z3
2 R1,R3,R4, Z1, Y2, Z2
R5 = Y2Z3
1 R1,R3,R4,R5, Z1, Z2
Z3 = Z1Z2R3 R1,R3,R4,R5, Z3
R6 = R5 − R4 R1,R3,R4,R6, Z3
R7 = R2
3R1 R3,R4,R6,R7, Z3
R8 = R3
3 R4,R6,R7,R8, Z3
X3 = R2
6 − 2R7 − R8 R4,R6,R7,R8,X3, Z3

Y3 = (R7 − X3)R6 − R4R8 X3, Y3, Z3
throughout the flow we never use more than 6 variables.

Thisreaches the lower bound: with less than 6 variables, one cannoteven load the two addends P1 and P2 at the very beginning.Note that at this stage we do not count the auxiliary variablesused within a certain step; we will consider them shortly.For GPU programming, memory optimizations are the mostimportant area for performance.

The general guideline is usingas much fast memory and as little slow memory as possible[10, §9], which explains why we rely so much on registers.Registers are accessed withPhotographs and grayscale figures should be prepared with 220 dpi resolution and saved with no compression, 8 bits per pixel (grayscale). To obtain a 3.45-in figure (onecolumn width) at 220 dpi, the figure should have a horizontal size of 759 pixels. Color figures should be prepared with 400 dpi resolution and saved with no compression, 8 bits per pixel (palette or 256 color). To obtain a 3.45-in figure (one column width) at 400 dpi, the figure should have a horizontal size of 1380 pixels. For more information on TIFF files, please go to http://www.ieee.org/organizations/pubs/transactions/infor ma tion.html and click on the link "Guidelines for Author Supplied Electronic Text and Graphics." 3) Somewhat Harder Way: If you do not have a scanner, you may create non-color PostScript figures by "printing" them to files.

First, download a PostScript printer driver from http://www.adobe.com/support/downloads/pdrvwin.htm (forWindows or from http://www.a.com/support /downloads/pdrvmac.htm (for Macintosh) and install the "Generic PostScript Printer" definition. In Word, paste your figure into a new document. Print to a file using the PostScript printer driver. File names should be of the form "fig5.ps." Use Adobe Type 1 fonts when creating your figures, if possible. 4) Other Ways: Experienced computer users can convert figures and tables from their original format to TIFF.

Some useful image converters are Adobe Photoshop, Corel Draw, and Microsoft Photo Editor, an application that is part of Microsoft Office 97 and Office 2000 (look for C:\Program Files\Common Files \Microsoft Shared\ PhotoEd\ PHOTOED.EXE. (You may have to custom-install Photo Editor from your original Office disk.) Here is a way to make TIFF image files of tables. First, create your table in Word. Use horizontal lines but no vertical lines. Hide gridlines (Table | Hide Gridlines). Spell check the table to remove any red underlines that indicate spelling errors. Adjust magnification (View | Zoom) such that you can view the entire table at maximum area when you select View | Full Screen. Move the cursor so that it is out of the way. Press "Print Screen" on your keyboard; this copies the screen image to the Windows clipboard. Open Microsoft Photo Editor and click Edit | Paste as New Image. Crop the table image (click Select button; select the part you want, then Image | Crop). Adjust the properties of the image (File | Properties) to monochrome (1 bit) and 600 pixels per inch. Resize the image (Image | Resize) to a width of 3.45 inches. Save the file (File | Save As) in TIFF with no compression (click "More" button). Most graphing programs allow you to save graphs in TIFF; however, you often have no control over compression or number of bits per pixel. You should open these image files in a program such as Microsoft Photo Editor and resave them using no compression, either 1 or 8 bits, and either 600 or 220 dpi resolution (File | Properties; Image | Resize). See Section II-D2 for an explanation of number of bits and resolution. If your graphing program cannot export to TIFF, you can use the same technique described for tables in the previous paragraph. A way to convert a figure from Windows Metafile (WMF) to TIFF is to paste it into Microsoft PowerPoint, save it in JPG format, open it with Microsoft Photo Editor or similar converter, and re-save it as TIFF. Microsoft Excel allows you to save spreadsheet charts in Graphics Interchange Format (GIF). To get good resolution, make the Excel charts very large. Then use the "Save as HTML" feature (see http://support.microsoft.com/support/ kb/articles/q158/0/79.asp). You can then convert from GIF

TABLE I
UNITS FOR MAGNETIC PROPERTIES

Symbol	Quantity	Conversion from Gaussian and CGS EMU to SI [a]
Φ	magnetic flux	1 Mx $\rightarrow 10^{-8}$ Wb = 10^{-8} V·s
B	magnetic flux density, magnetic induction	1 G $\rightarrow 10^{-4}$ T = 10^{-4} Wb/m^2
H	magnetic field strength	1 Oe $\rightarrow 10^3/(4\pi)$ A/m
m	magnetic moment	1 erg/G = 1 emu $\rightarrow 10^{-3}$ A·m^2 = 10^{-3} J/T
M	magnetization	1 erg/(G·cm^3) = 1 emu/cm^3 $\rightarrow 10^3$ A/m
$4\pi M$	magnetization	1 G $\rightarrow 10^3/(4\pi)$ A/m
σ	specific magnetization	1 erg/(G·g) = 1 emu/g $\rightarrow$ 1 A·m^2/kg
j	magnetic dipole moment	1 erg/G = 1 emu $\rightarrow 4\pi\times 10^{-10}$ Wb·m
J	magnetic polarization	1 erg/(G·cm^3) = 1 emu/cm^3 $\rightarrow 4\pi\times 10^{-4}$ T
χ, κ	susceptibility	$1 \rightarrow 4\pi$
χ_ρ	mass susceptibility	1 cm^3/g $\rightarrow 4\pi\times 10^{-3}$ m^3/kg
μ	permeability	$1 \rightarrow 4\pi\times 10^{-7}$ H/m $= 4\pi\times 10^{-7}$ Wb/(A·m)
μ_r	relative permeability	$\mu \rightarrow \mu_r$
w, W	energy density	1 erg/cm^3 $\rightarrow 10^{-1}$ J/m^3
N, D	demagnetizing factor	$1 \rightarrow 1/(4\pi)$

No vertical lines in table. Statements that serve as captions for the entire table do not need footnote letters.

[a]Gaussian units are the same as cgs emu for magnetostatics; Mx = maxwell, G = gauss, Oe = oersted; Wb = weber, V = volt, s = second, T = tesla, m = meter, A = ampere, J = joule, kg = kilogram, H = henry.

TIFF using Microsoft Photo Editor, for example. No matter how you convert your images, it is a good idea to print the TIFF files to make sure nothing was lost in the conversion. If you modify this document for use with other IEEE journals or conferences, you should save it as

type “Word 97-2000 & 6.0/95 - RTF (*.doc)” so that it can be opened by any version of Word.

III. CONCLUSION

In this paper, we have demonstrated our exhaustive experiencewith a concrete case study on SIGaaS. Specifically, wehave developed a tangible GPU-accelerated universal ellipticcurvesignature server called Guess for short.Our primary contribution is a novel, systematic, and inclusiveimplementation of ECDSA, turning cryptographic theoryinto productivity on off-the-shelf processors. For example, wehave proposed high-level algorithms for the PM primitives,which are the bottleneck of ECDSA and other ECC schemes.For another example, space-efficient lowlevel algorithms havebeen devised to bolster our one-threadper-task approach, sothat we can launch more GPU threads for higher throughput.Besides customizing and optimizing various algorithms forour platform to maximize its computing power, we have alsoleveraged efficient resource management to provide Guess’computing power to network tenants with minimum loss. Fieldexperiments have shown that Guess achieves significantlyhigher performance than existent prototypes and products.Guess is implemented with software on a general platformthat is fairly affordable and easily upgradable, implying thatfurther performance improvement and functionality extensionare feasible. This facilitates scalable and flexible cloud service

IV. REFERENCES

[1] “The numbers behind Alibaba’s Singles Day,” Nov. 2015. [Online]. Available: http://finance.yahoo.com/news/numbersbehindalibaba singles-day-202157979.html.

[2] V. Varadharajan and U. Tupakula, “Security as a service model for cloud environment,” IEEE Transactions on Network and Service Management, vol. 11, no. 1, pp. 60–75, Mar. 2014.

[3] J. W. Bos, “Low-latency elliptic curve scalar multiplication,” International Journal of Parallel Programming, vol. 40, no. 5, pp. 532–550, Oct. 2012.

[4] S. Cui, J. Großsch¨adl, Z. Liu, and Q. Xu, “High-speed elliptic curve cryptography on the NVIDIA GT200 graphics processing unit,” in proc. ISPEC’14, ser. LNCS, vol. 8434, May 2014, pp. 202–216.

[5] K. Jang, S. Han, S. Han, S. B. Moon, and K. Park, “SSLShader: Cheap SSL acceleration with commodity processors,” in proc. USENIX NSDI’11, Apr. 2011.

[6] NIST, “Digital Signature Standard (DSS),” FIPS 186-4, July 2013. [Online]. Available: http://csrc.nist.gov/publications/PubsFIPS.html

[7] “Recommendation for key management – Part 1: General (Revision 3),” SP 800-57 Part 1 - Rev. 3, July 2012. [Online]. Available: http://csrc.nist.gov/publications/PubsSPs.html

[8] Thales e-Security, “nShield Connect data sheet.” [Online]. Available: https://www.thales-esecurity.com/products-and-services/productsandservices/ hardware-security-modules/general-purpose-hsms/nshieldconnect/

[9] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone, and J. C. Phillips, “GPU computing,” Proceedings of the IEEE, vol. 96, no. 5, pp. 879–899, May 2008.

[10] NVIDIA, “CUDA C best practices guide.” [Online]. Available: http://docs.nvidia.com/cuda/cuda-c-best-practices-guide/

[11] “CUDA C programming guide.” [Online]. Available: http://docs.nvidia.com/cuda/cuda-c-programming-guide/

[12] S. Ant˜ao, J.-C. Bajard, and L. Sousa, “Elliptic curve point multiplication on GPUs,” in proc. IEEE ASAP’10, July 2010, pp. 192–199.

[13] F. Zheng, W. Pan, J. Lin, J. Jing, and Y. Zhao, “Exploiting the floatingpoint computing power of GPUs for RSA,” in proc. ISC’14, ser. LNCS, vol. 8783, Oct. 2014, pp. 198–215.

[14] “Exploiting the potential of GPUs for modular multiplication in ECC,” in proc. WISA’14, ser. LNCS, vol. 8909, Jan. 2015, pp. 295–306.

[15] M. Rivain, “Fast and regular algorithms for scalar multiplication over elliptic curves,” IACR Cryptology ePrint Archive 2011/338, 2011. [Online]. Available: http://eprint.iacr.org/2011/338.pdf

[16] J. W. Kim, J. Seo, J. Hong, K. Park, and S.-R. Kim, “Highspeed parallel implementations of the rainbow method in a heterogeneous system,” in proc. INDOCRYPT’12, ser. LNCS, vol. 7668, Dec. 2012, pp. 303– 316.

[17] P. Karpman, T. Peyrin, and M. Stevens, “Practical free-start collision attacks on 76-step SHA-1,” in proc. CRYPTO’15, ser. LNCS, vol. 9215, Aug. 2015, pp. 623–642.

[18] EVGA, “EVGA GeForce GTX 780 Ti Superclocked.” [Online]. Available: http://www.evga.com/articles/00795/#2883

[19] D. Hankerson, A. Menezes, and S. Vanstone, Guide to Elliptic Curve Cryptography. Springer Science & Business Media, 2004.

[20] “Public key cryptographic algorithm SM2 based on elliptic curves – Part 2: Digital signature algorithm,” Dec. 2010. [Online]. Available: http://.gov.cn/pFile /2010122214822692.p