A 521-bit Dual-Field Elliptic Curve Cryptographic Processor with Power Analysis Resistance

Jen-Wei Lee, Yao-Lin Chen, Chih-Yeh Tseng, Hsie-Chia Chang, and Chen-Yi Lee
 Department of Electronics Engineering and Institute of Electronics,
 National Chiao Tung University, Hsinchu, Taiwan.
 Email: jenweilee@gmail.com

Abstract—Recently, several hardware implementations for elliptic curve cryptography have been proposed but few of them considered the dual-field functions, real-time requirement, hardware efficiency, and power analysis resistance as a whole. In this paper, a new unified division algorithm and a free pre-computation scheme are introduced to accelerate the $GF(p)/GF(2^n)$ elliptic curve arithmetic functions. The overall hardware is optimized by a very compact *Galois field arithmetic unit* with the fully pipelined technique. Moreover, a *key-blinded* technique with regular calculation is designed against the power analysis attacks without degrading clock speed. After fabricated in 90nm CMOS 1P9M process, our ECC processor occupied 0.55mm^2 can perform the scalar multiplication in 19.2ms over $GF(p_{521})$ and 8.2ms over $GF(2^{409})$, respectively.

I. INTRODUCTION

Nowadays, secure information exchange is an important issue for the communication network. Since the aging RSA algorithm has been challenged by the quick factoring attack, elliptic curve cryptography (ECC) that can provide the same level of security with shorter key size becomes more attractive. The ECC device is advantageous in terms of speed, cost and power consumption.

For the ECC schemes specified in IEEE P1363 [1], the most time-critical function is the elliptic curve scalar multiplication (ECSM), which computes a multiple point kP with an n -bit private key k and a point P on elliptic curves (ECs). This can be achieved by the basic point calculation such as point addition and point doubling, which involve the finite field operations either for $GF(p)$ or $GF(2^n)$. By representing the points in affine coordinates, the formulas are given in Table I.

To target at high speed, the carry-bypass adder [2] and fixed polynomials [3] are used for processing $GF(2^n)$ functions on specific ECs. However, the applications of IEEE P1363 including digital signature are approved for supporting dual-field functions on arbitrary ECs. Exploiting multiple function units is another common technique to accelerate the ECC functions, but this requires large hardware cost. In [4], an architecture of four dual-field Montgomery multipliers implemented with over hundred thousands gates can only support 160-bit operations. The short key size just can accommodate the security of low-end device. Besides, the real-time requirement is necessary to the data service. The well-known Montgomery algorithm [5] is usually adopted to avoid trivial modular reduction over $GF(p)$, while it requires pre-computed values for domain conversion. Because these values are different from users with distinct

TABLE I
 FORMULAS OF POINT CALCULATION IN AFFINE COORDINATES

	Point Addition ($P_3 = P_1 + P_2$)	Point Doubling ($P_3 = 2P_1$)
$GF(p)$	$\lambda = \frac{y_2 - y_1}{x_2 - x_1}$ $x_3 = \lambda^2 - x_1 - x_2$ $y_3 = \lambda(x_1 - x_3) - y_1$	$\lambda = \frac{3x_1^2 + a_p}{2y_1}$ $x_3 = \lambda^2 - 2x_1$ $y_3 = \lambda(x_1 - x_3) - y_1$
$GF(2^n)$	$\lambda = \frac{y_2 + y_1}{x_2 + x_1}$ $x_3 = \lambda^2 + \lambda + x_1 + x_2 + a_b$ $y_3 = \lambda(x_1 + x_3) + x_3 + y_1$	$\lambda = x_1 + \frac{y_1}{x_1}$ $x_3 = \lambda^2 + \lambda + a_b$ $y_3 = \lambda(x_1 + x_3) + x_3 + y_1$

keys and ECs, the pre-computations calculated from host CPU result in the system retard.

Also, the key of unprotected crypto-chip can be revealed by power measurement including both simple power analysis (SPA) and differential power analysis (DPA) [6]. For chip implementation, the countermeasure described in [7] uses the switched capacitor to prevent current running through the encryption core. But this approach results in halved frequency for replenishing charge every cycle.

In this paper, the proposed ECC processor aims at providing a hardware-efficient and real-time solution to support 521-bit $GF(p)/GF(2^n)$ functions on arbitrary ECs with power analysis resistance. Since the field division dominates the computation time of point calculation, a new unified division algorithm is also proposed. This is based on the Montgomery inversion [8] to extend the functionality by directly implementing both the Montgomery and modular division with less latency. To improve hardware efficiency further, a highly integrated and pipelined Galois field arithmetic unit (GFAU) with circular shift register is exploited. And the overall pre-computations for domain conversion can be eliminated by performing several Montgomery divisions and multiplications on-the-fly. To prevent the power analysis attacks without significant speed degradation, both the key-blinded technique and regular calculation with dummy operations are adopted. Compared with other related works, our ECC processor shows the advantages for hardware efficiency.

II. MONTGOMERY MODULAR ALGORITHM

For a given n -bit prime p , the Montgomery domain representation of an integer a is defined as $A \equiv a \cdot r \pmod{p}$ with the Montgomery constant $r = 2^n \pmod{p}$. The Montgomery multiplication of two Montgomery domain integers A and B is given in Algorithm 1. To achieve the inversion in Montgomery

domain, Kaliski [8] proposed an iterative algorithm shown in Algorithm 2. For the Montgomery division, it needs a Montgomery multiplication following the Montgomery inversion such that $\text{MonMul}(A, B^{-1} \cdot r^2) \equiv a \cdot b^{-1} \cdot r \pmod{p}$. Since the maximum iteration time at Step 2 and Step 4 in Algorithm 2 are $2n$ and n respectively, it requires $n \sim 3n$ iteration time and two Montgomery multiplications to perform one operation of the Montgomery division.

Algorithm 1: Montgomery multiplication
Input: $A \equiv a \cdot r \pmod{p}$, $B \equiv b \cdot r \pmod{p}$, p , n .
Output: $C = \text{MonMul}(A, B) \equiv A \cdot B \cdot r^{-1} \pmod{p} \equiv a \cdot b \cdot r \pmod{p}$.
1. Let $A = (A_{n-1}, A_{n-2}, \dots, A_1, A_0)_2$, $C = 0$.
2. For i from 0 to $n-1$ do { $T = (C + A_i \cdot B)$, $C = \frac{T+T_0 \cdot p}{2}$. }
3. If $C \geq p$, then $C = C - p$.
4. Return C .
Algorithm 2: Kaliski's Montgomery inversion
Input: $B \equiv b \cdot r \pmod{p}$, p , n , r^2 .
Output: $R = \text{MonInv}(B) \equiv B^{-1} \cdot r^2 \pmod{p} \equiv b^{-1} \cdot r \pmod{p}$.
1. Let $U = p$, $V = B$, $R = 0$, $S = 1$, $i = 0$.
2. While ($V > 0$) do {
If U is even, then { $U = \frac{U}{2}$, $S = 2S$. }
else if V is even, then { $V = \frac{V}{2}$, $R = 2R$. }
else if $U > V$, then { $U = \frac{U-V}{2}$, $R = R + S$, $S = 2S$. }
else { $V = \frac{V-U}{2}$, $R = 2R$, $S = R + S$. }
$i = i + 1$.
3. If $R \geq p$, then $R = R - p$.
4. For $l = 1$ to $i - n$ do $R = \frac{R+R_0 \cdot p}{2}$.
5. $R = p - R$.
6. $R = \text{MonMul}(R, r^2)$.
7. Return R .

Algorithm 3 shows our proposed *unified* division algorithm. From the observation in Algorithm 2, the derivation is based on the invariants at Step 2 as follows

$$\begin{cases} B \cdot R \equiv -U \cdot 2^i \pmod{p} \\ B \cdot S \equiv V \cdot 2^i \pmod{p}. \end{cases}$$

By setting the initial values of (U, V, R, S) to be $(p, B, 0, A)$ at Step 1, our approach modifies the invariants at Step 2 such that

$$\begin{cases} A^{-1} \cdot B \cdot R \equiv -U \cdot 2^i \pmod{p} \\ A^{-1} \cdot B \cdot S \equiv V \cdot 2^i \pmod{p}. \end{cases}$$

Before the last iteration, both U and V are 1 because the initial values of U and V are relatively prime. Then after finishing Step 2, the values of (U, V, R, S) are $(1, 0, -A \cdot B^{-1} \cdot 2^i \pmod{p}, 0)$. Note that the modular reductions are needed for every iteration to remain the values of both operands R and S in finite field set $\{0, 1, \dots, p-1\}$. The iterations at Step 3 are to set the 2^i of R to be 2^n or identity by dividing two. Before returning the final result, it is simple to output the positive result by using a subtraction $p - R$ but there is no need as performing for $GF(2^n)$. The output is an optional result which includes the residue of the Montgomery division $R \equiv a \cdot b^{-1} \cdot 2^n \pmod{p}$ and the residue of the modular division $R \equiv a \cdot b^{-1} \pmod{p}$. As a result, our proposed Montgomery division operation only takes $n \sim 3n$ iteration time without any Montgomery multiplication operation. Compared with the Kaliski's Montgomery inversion by using Algorithm 1 to implement one Montgomery multiplication with n iterations, our proposed Montgomery division has 62% cycle reduction in average. The modular division, which is essentially required for ECC schemes, can be achieved

Algorithm 3: Our unified division
Input: $A \equiv a \cdot r \pmod{p}$, $B \equiv b \cdot r \pmod{p}$, p , n .
Output: R , where $\begin{cases} R = \text{MonDiv}(A, B) \equiv a \cdot b^{-1} \cdot r \pmod{p}. \\ R = \text{ModDiv}(A, B) \equiv a \cdot b^{-1} \pmod{p}. \end{cases}$
1. Let $U = p$, $V = B$, $R = 0$, $S = A$, $i = 0$.
2. While ($V > 0$) do {
If U is even, then { $U = \frac{U}{2}$, $S = 2S$. }
else if V is even, then { $V = \frac{V}{2}$, $R = 2R$. }
else if $U > V$, then { $U = \frac{U-V}{2}$, $R = R + S$, $S = 2S$. }
else { $V = \frac{V-U}{2}$, $R = 2R$, $S = R + S$. }
$i = i + 1$.
If $R \geq p$, then $R = R - p$.
If $S \geq p$, then $S = S - p$.
3. For j from 1 to $\begin{cases} i - n & \text{do } R = \frac{R+R_0 \cdot p}{2}. \end{cases}$
4. $R = p - R$.
5. Return R .

TABLE II
EXAMPLES OF MONTGOMERY DIVISION OVER DUAL FIELDS

U	V	R	S	i	j	U	V	R	S	i	j
13	5	0	12	0	-	(10011)	(01101)	(00000)	(01100)	0	-
4	5	12	11	1	-	(01111)	(01101)	(01100)	(01011)	1	-
2	5	12	9	2	-	(01111)	(00001)	(01011)	(00111)	2	-
1	5	12	5	3	-	(00111)	(00001)	(01100)	(01110)	3	-
1	2	11	4	4	-	(00011)	(00001)	(00010)	(01111)	4	-
1	1	9	4	5	-	(00001)	(00001)	(01101)	(01101)	5	-
1	0	5	0	6	-	(00001)	(00000)	(01001)	(00000)	6	-
-	-	9	-	-	1	-	-	(01101)	-	-	1
-	-	11	-	-	2	-	-	(01111)	-	-	2
Return: $R = 13 - 11 = 2$						Return: $R = x^3 + x^2 + x^1 + 1$					

with additional n iterations by dividing two without extra circuit components. Table II gives the examples to demonstrate the Montgomery division $2 \equiv 12 \cdot 5^{-1} \cdot 2^4 \pmod{13}$ and $x^3 + x^2 + x^1 + 1 \equiv (x^3 + x^2) \cdot (x^3 + x^2 + 1)^{-1} \cdot x^4 \pmod{x^4 + x + 1}$.

III. OUR PROPOSED ECC PROCESSOR

Fig. 1 shows the overall ECC architecture with a standard AMBA AHB bus interface. The ECSM with modular operations over dual fields, required for the ECC schemes such as signature, authentication and key exchange, can be calculated by the *Galois field arithmetic unit* (GFAU). The inputs are the user public/private-key, EC coordinates, EC parameters and protocol instructions. To real-time perform these contents, the instruction decoder and pre-/post-process are combined in our processor. After the instruction decoding, the pre-process stage is to convert the EC coordinates and parameters into the Montgomery domain and blind the key value to avoid

Fig. 1. System architecture of our ECC processor.

power analysis attacks. Before returning the calculation results, the EC coordinates are converted into the integer domain at post-process stage. All the 521-bit data operands stored in registers and transmitted to the GFAU are manipulated by EC control. To reduce the multiplexer complexity for the long bit length of registers, the separated 32-bit circular shift registers is exploited.

A. Galois Field Arithmetic Unit

Considering a hardware-efficient design to support all Montgomery/modular operations, we use a bit-level architecture to combine the multiplier and divisor into the GFAU. To effectively perform multiple arithmetic operations, which are addition/subtraction and shifting in the iterative loop, the carry-saved adder (CSA) with a carry-propagation adder (CPA) at final stage is used. Another benefit is that it can achieve additive operation for $GF(2^n)$ by hardware sharing because the sum of CSA is equal to two bitwise XOR operators.

If these iterative operations are performed within one cycle, the critical path is to calculate the results of operands R or S in Algorithm 3 which consists of the UV comparative cases with modular addition. The time-critical comparative cases of U, V achieved by a subtraction is nearly equal to an addition delay. Since the data path of the operands U and V are irrelevant to the operands of R and S , a fully pipelined stage can be inserted between the UV data path and the RS data path to moderate the critical path. One additional cycle is needed after pipelining, but this can be neglected as the operation takes hundreds or thousands cycles. Fig. 2 shows the timing flow of the pipelined scheme. As the comparative case is determined in $(m-1)$ th cycle, then the m th cycle is to set the values of the operands R, S and simultaneously determine the next case until $V = 0$.

Fig. 2. Fully pipelined scheme for the unified division.

Fig. 3 shows the major arithmetic unit for our ECC processor. Registers and modular operations for operands U, S, R in Algorithm 3 can be shared with operands A, B, C in Algorithm 1 to save circuit area since they are not processing in the same time. The selective signal c_2 for RS data path is fed from c_1 , which is determined by the UV comparative case, with one cycle delay due to pipelining.

B. Free Pre-Computation Scheme

Because the primary inputs of base point coordinates (x, y) and EC parameters a_p or a_b are in the integer domain, the domain conversion can be performed by the proposed Montgomery division such that $\text{MonDiv}(a, 1) \equiv a \cdot 1 \cdot 1^{-1} \cdot 2^n \pmod{p} \equiv a \cdot 2^n \pmod{p}$. On the other hand, to return the point coordinates in the integer domain, the Montgomery multiplication can be used as $\text{MonMul}(a2^n, 1) \equiv a2^n \cdot 1 \cdot 2^{-n} \pmod{p}$.

Fig. 3. All dual-field Montgomery and modular operations are integrated into the pipelined GFAU with circuit sharing.

$p) \equiv a \pmod{p}$. For calculating one ECSM, the overheads of domain conversion are three Montgomery divisions and two Montgomery multiplications; both of them can be performed on-the-fly by the GFAU to avoid any pre-computation from host system.

C. Power Analysis Resistance

To hide the leaked information against both the SPA and DPA, the *double-and-add-always* and *key-blinded* techniques are exploited. The previous one ensures that the ECSM in the process is uncorrelated with the key value by continuously performing point doubling and then (dummy) point addition. Because the multiplier of EC point, which is also the key value, is periodic of the point order, the multiplier can be blinded by adding $r \cdot \#E$, where r is a 32-bit random integer and $\#E$ is the point order. The random integer is generated by using the linear feedback shift register with the irreducible polynomial $x^{17} + x^{15} + x^8 + x^5 + 1$. These approaches to prevent power analysis attacks are shown in Fig. 4.

Fig. 4. Blinding key with a random number and calculating the ECSM with dummy point addition.

IV. IMPLEMENTATION AND MEASUREMENT RESULTS

Our proposed 521-bit dual-field ECC processor is fabricated in 90nm 1P9M CMOS technology. After adopting the circular shift register, our approach can improve the hardware utilization from 60% to 88%, leading to 0.26mm² core area reduction. From the measurement results, it can perform the ECSM over $GF(p_{521})$ in 19.2ms with 58.5mW and $GF(2^{409})$ in 8.2ms with 86.4mW, respectively.

Fig. 5 and Fig. 6 show the unprotected/protected chip micrograph and the experimental condition of power measurement. Fig. 7(a) shows the SPA of the unprotected ECC chip. The 1 and 0 key values can be disclosed by checking the peak

differences of the power traces, where the positive and negative peak differences are 112mV and 57mV in average. Due to the regular calculation, the peak differences of our protected chip, shown in Fig. 7(b), are reduced to below the noise region 20mV. Fig. 8(a) illustrates the DPA of the unprotected chip, where the spikes of correlation coefficients can be scattered from 500 power traces. Note that the inferior spikes appear at distinct bits of key since the point doubling exists inherently. However, the spikes are not given in Fig. 8(b) even after collecting 120K power traces of our protected ECC chip.

Fig. 5. Chip micrograph of unprotected and protected ECC chip.

Fig. 6. Environment of power measurement.

Table III gives the chip summary and comparison. Compared with the previous 160-bit dual-field design [4], our proposed ECC processor can achieve 521-bit operations and real-time requirement with similar gates count. The $GF(2^n)$ arithmetic is usually faster than $GF(p)$ arithmetic due to carry-free addition [2]. By using our proposed unified division without dummy operations, the binary-field-only ECC design can be implemented to perform the $GF(2^{409})$ ECSM in 2.37ms with only 96Kgates according to post-layout simulations. In contrast to [7] that utilizes the switched capacitor to prevent power analysis attacks, our approach is advantageous in system speed and power consumption.

Fig. 7. The power consumption of (111) and (000) bits for (a) unprotected chip (b) protected chip over time with 200mV/div voltage resolution and 200μs/div time base.

V. CONCLUSION

This paper presents a hardware-efficient ECC processor which supports arbitrary 521-bit $GF(p)/GF(2^n)$ operations with power analysis resistance. Both speed and efficiency are improved by introducing a new unified division algorithm and a highly integrated GFAU with the fully pipelined technique.

Fig. 8. Correlation coefficients of (0010) key value for (a) unprotected chip (b) protected chip.

TABLE III
COMPARISON AMONG PREVIOUS WORKS

	This Work						[4](Post-Sim)		[2](Post-Sim)		
Technology	90nm						0.13μm		0.13μm		
Gates Count	170K						151K		332K		
Core Area	0.55mm ²						1.06mm ²		2.33mm ²		
Finite Field	Dual						Dual		Single		
Field Size	GF(p)			GF(2 ⁿ)			GF(p)		GF(2 ⁿ)		
	160	256	521	163	283	409	160	160	163	283	409
Exe. Time (ms)	1.62	4.4	19.2	1.15	3.33	8.2	0.34	0.16	0.44	1.25	2.26
f _{max} (MHz)	154	147	132	188	182	166	217	350	415		
Power (mW)	66.3	67.6	58.5	72.5	87.0	86.4	NA	NA	139	177	250
Pre-Compt.	No need						Need		No need		
Resistance	SPA and DPA						No		No		
Overhead of Protected Chip							This Work				
	Chip Size						+10%		+7.2%		
	f _{max}						-2.5% (1V)		-50% (1.2V)		
	Power						+6% (1V, 100MHz)		+33% (1.2V, 100MHz)		

By processing the domain conversion on-the-fly and blinding key with dummy operations, our ECC processor achieves real-time performance and protection against the power analysis attacks without a large degradation of system speed. The chip has been fabricated in 90nm CMOS technology and measured to perform the ECSM in 19.2ms over $GF(p)$ and 8.2ms over $GF(2^n)$ with 0.55mm² core area.

ACKNOWLEDGMENT

This work was supported by NSC, Taiwan, under grant NSC97-2221-E-009-166-MY3. The authors thank UMC for chip fabrication and CIC for providing measurement facilities.

REFERENCES

- [1] IEEE, IEEE 1363 Standard Specifications for Public-Key Cryptography, Jan. 2000. [Online]. Available: <http://grouper.ieee.org/groups/1363/>
- [2] J.-H. Chen, M.-D. Shieh, and W.-C. Lin, "A high-performance unified-field reconfigurable cryptographic processor," *IEEE Trans. VLSI Syst.*, 2009.
- [3] K. Sakiyama, L. Batina, B. Preneel, and I. Verbauwhede, "High-performance public-Key cryptoprocessor for wireless mobile applications," *Mobile Netw. Appl.*, 2007.
- [4] J.-Y. Lai, and C.-T. Huang, "Elixir: high-throughput cost-effective dual-field processors and the design framework for elliptic curve cryptography," *IEEE Trans. VLSI Syst.*, vol. 16, no. 11, pp. 1567–1580, 2008.
- [5] P. L. Montgomery, "Modular multiplication without trial division," *Math. Computing*, vol. 44, no. 170, pp. 519–521, Apr. 1985.
- [6] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," *Advances in Cryptology, LNCS1666*, pp. 388–397, 1999.
- [7] C. Tokunaga, and D. Blaauw, "Secure AES engine with a local switched-capacitor current equalizer," in *Proc. ISSCC*, pp. 64–65.65(a), 2008.
- [8] B. S. Kaliski, "The Montgomery inverse and its applications," *IEEE Trans. Comput.*, vol. 44, no. 8, pp. 1064–1065, Aug. 1995.