Register Transfer Level Concurrent Error Detection in Elliptic Curve Crypto Implementations

Richard Stern, Nikhil Joshi, Kaijie Wu and Ramesh Karri

L-3 Communication – Systems East, Citibank, NA, University of Illinois, Chicago, Polytechnic University, Brooklyn

{rstern01, njoshi01, kwu01, rkarri}@utopia.poly.edu

History of ECC

- Proposed independently in 1985 by Neal Koblitz from University of Washington and Victor Miller at IBM.
- Elliptical Curve Cryptography (ECC) was proposed as an alternative to traditional public key cryptosystems such as RSA.
- Public key cryptography method based on Elliptic Curve Theory.

Elliptic Curve Overview

- An elliptical curve with the underlying field F_2m is defined as the set of points (x,y) that satisfy the equation, $y^2 + xy = x^3 + a_2x^2 + a_6$
- There are finitely many points on such an elliptical curve.

Elliptic Curve Operations

- Two geometrically defined operations over elliptical curve groups are point addition and point doubling.
- By selecting a point in an elliptical curve group, one can double it to obtain the point 2P. After that, one can add the point P to the point 2P to obtain the point 3P.
- The determination of a point nP in this manner is referred to as Scalar Multiplication of a point.

ECC – How it works

- K=Secret Integer, acts as Private Key.
- P=Point on the Elliptic Curve.
- Q=kP is the Public Key.
- Given Q and P it is difficult to find K. (ECC Discrete Logarithmic Problem)

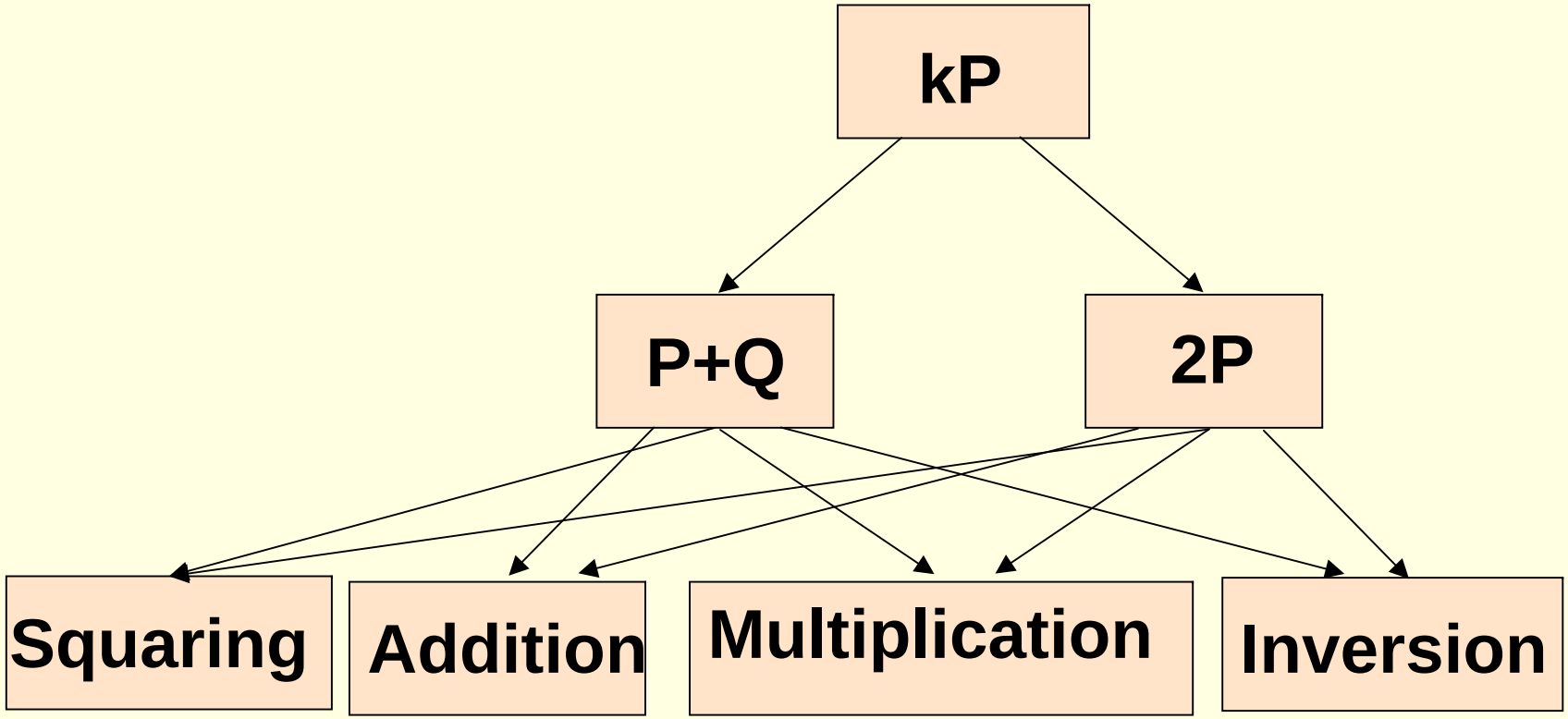
ECC – Discrete Logarithmic Problem

- The elliptical curve discrete logarithmic problem is: given points P and Q in the group, find a number, k, such that Pk=Q.
- One way we might suggest in finding k is to compute multiples of P until Q is found.
- However, in real applications, k would be large enough such that it would be impractical to determine k in this manner.

ECDH – Elliptic Curve Diffie-Hellman

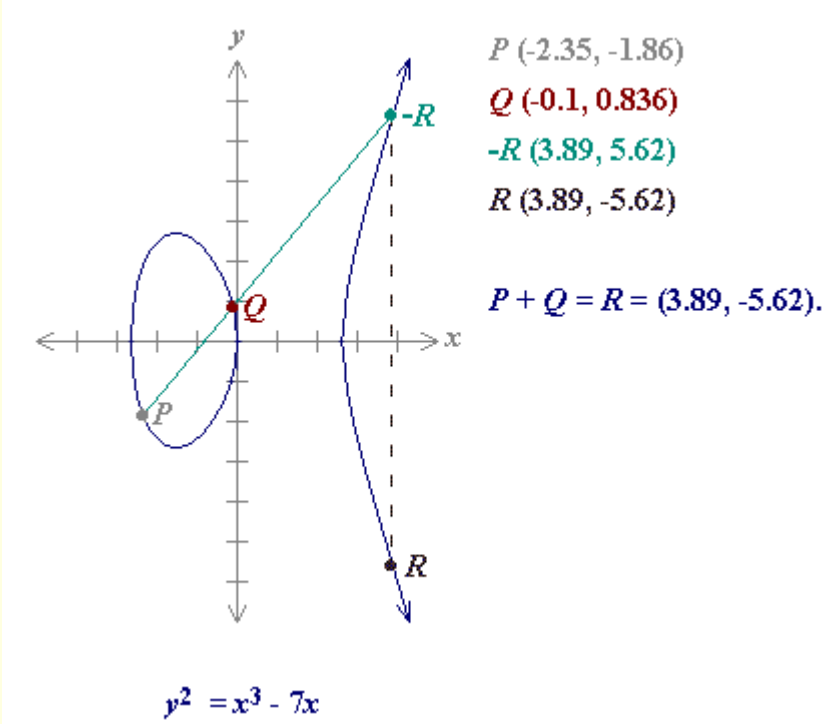
- Elliptical Curve Diffie-Hellman Protocol is a way for two parties to generate a private key over an insecure network using ECC.
- This key can later be used for communication by both parties who wish to engage in secure communication using a symmetric block cipher (e.g. RC5, AES, DES).
- The protocol for generation of the shared secret key using ECC is described below.
  1. Alice takes a point Q and generates a random number k_a .
  2. Alice computes the point $P_1 = k_a$*Q and sends it to Bob
  3. Bob generates a random number k_b, computes $P_2 = k_b$*Q and sends it to Alice
  4. Alice computes $P_{s1} = k_a * P_2$, and Bob computes $P_{s2} = k_b * P_1$
  5. $P_{s1} = P_{s2} = k_a k_b$ Q, This is used as the shared secret key
- The only keys that are available to the public are P_1 , P_2 and Q. Due to the Elliptical curve property, it is not practical to calculate k given P and Q. Thus, this method for negotiating the secret key is secure.

ECC Hierarchy



Point Addition (Geometric Approach)

- Suppose that P and Q are two distinct points on an elliptic curve.
- To add the points P and Q, a line is drawn through the two points. This line will intersect the elliptic curve in exactly one more point, call -R.
- The point -R is reflected in the x-axis to the point R.
- The law for addition in an elliptic curve group is P + Q = R



Point Addition (Algebraic Approach)

- The addition of two points on an elliptic curve is shown below.

Input: $P = (x_p, y_p)$

$Q = (x_q, y_q)$

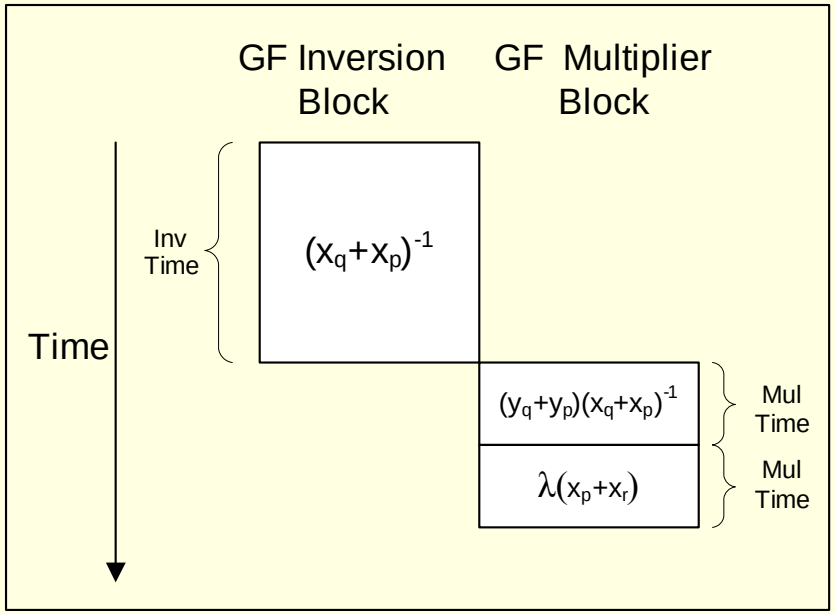
a_2

Output: $R = (x_R, y_R)$

$$\lambda = (y_q + y_p) \cdot (x_q + x_p)^{-1}$$

$$x_R = \lambda^2 + \lambda + x_p + x_q + a_2$$

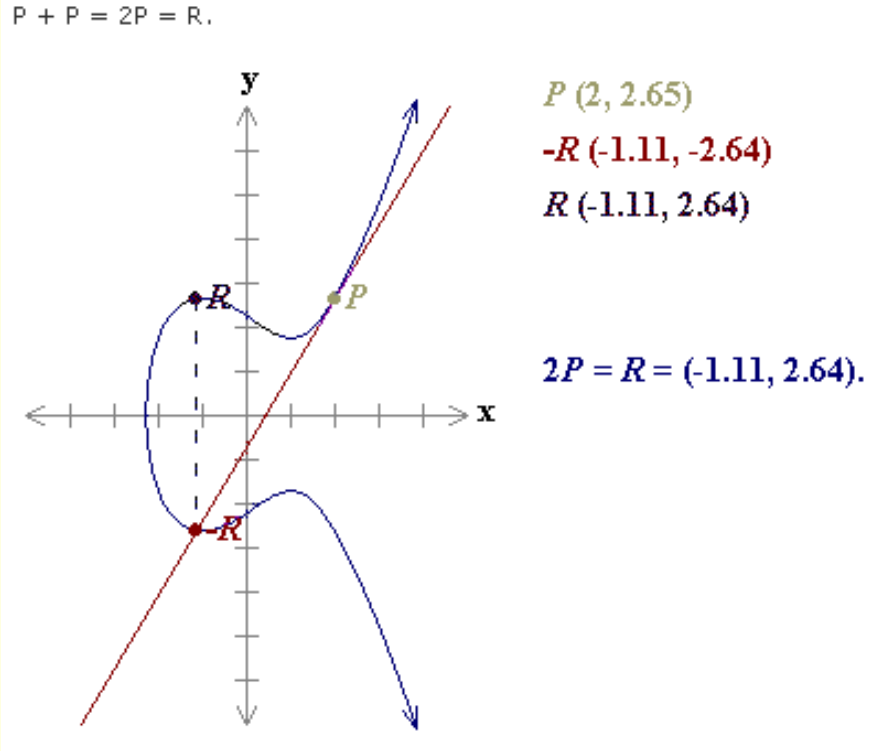
$$y_R = \lambda \cdot (x_p + x_R) + x_R + y_P$$



Point Addition

Point Doubling (Geometric Approach)

- To add a point P to itself, a tangent line to the curve is drawn at the point P.
- The tangent line intersects the elliptic curve at exactly one other point, -R. -R is reflected in the x-axis to R.



Point Doubling (Algebraic Approach)

- The doubling of a point on the elliptical curve is shown below.

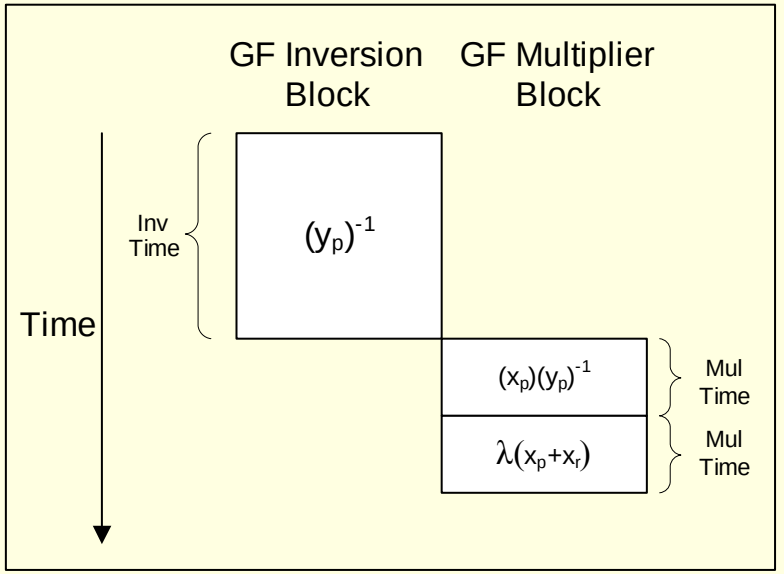
Input: $P = (x_p, y_p)$

a_6

Output: $R = (x_R, y_R)$

$$x_R = a_6 (x_p^{-1})^2 + x_p^2$$

$$y_R = x_p^2 + (x_p + y_p \cdot x_p^{-1}) \cdot x_R + x_R$$



Point Doubling

Multiplication

- The shift-and-add algorithm for multiplication is given below

Input: A(x), B(x) $\in F_2 m$
P(x) irreducible polynomial of degree 163
Output: $C(x) = A(x) \cdot B(x) \bmod P(x)$

1. C(x) <= 0
2. for(i=162 downto 0)
3. C(x) <= (C(x) << 1) xor A(x)b_i xor $C_m P(x)$
4. end for

Multiplication with interleaved modular reduction

Inversion

- The inversion algorithm is shown below

Input: A(x) $\in F_2m$, $A(x) \neq 0$

P(x) irreducible polynomial of degree 163

Output: $C(x) = A(x)^{-1} \bmod P(x)$

1. Y(x) <= A(x), D(x) <= P(x), B(x) <= 0, X(x) <= 1
2. loop
3. while ($Y_0 = 0$)
4. Y(x) <= Y(x)>>1
5. X(x)<=(X(x) xor X_0 P(x)) >> 1
6. end while
7. if (Y(x) = 1) then
8. return X(x)
9. end if
10. if (Y(x) < D(x)) then
11. Y(x)⇔D(x)
12. B(x)⇔X(x)
13. end if
14. Y(x) <= Y(x) xor D(x)
15. X(x) <= X(x) xor B(x)
16. end loop

Strength of ECC

- ECC devices require small key sizes.
- Hardware implementations of ECC require:
  - Less storage space
  - Less processing power
  - Results in faster computations compared to conventional public key systems.
- This is especially useful in area critical and constrained environments such as smart cards, wireless devices, and handheld computers.

NIST guidelines for public key sizes for AES

ECC KEY SIZE (Bits)	RSA KEY SIZE (Bits)	KEY SIZE RATIO	AES KEY SIZE (Bits)
163	1024	1 : 6	
256	3072	1 : 12	128
384	7680	1 : 20	192
512	15 360	1 : 30	256

Concurrent Error Detection Introduction

- Faults in VLSI chips
  - Transient faults - die away after sometime.
  - Permanent faults - do not die away with time.
- Concurrent Error Detection (CED)
  - Hardware Redundancy – Duplicate hardware
  - Timing Redundancy – Re-compute using same hardware

Problems with current CED methods

- Hardware redundancy
  - Two copies of hardware
  - Detects transient and permanent faults
- Timing Redundancy
  - No hardware overhead
  - Detects only transient faults

CED for ECC based systems

- Due to its designed use in area critical devices, Hardware redundancy is impractical.
- We propose a timing redundancy based technique which can detect both transient and permanent faults.
  - This is done by changing the timing redundancy technique such that different data is operated on during the CED re-computation.
    - Exploits the involution property of the inversion block
    - Exploits the multiplicative inverse property of the multiplication block.

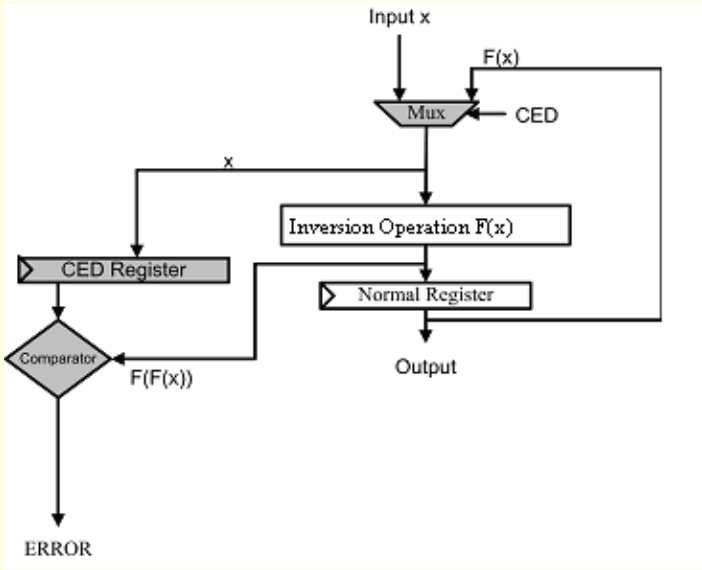
CED for ECC based systems

- Other ECC CED implementations describe CED techniques for the multiplication over GF(2m). In [32],they describe a robust Montgomery multiplier array with concurrent error detection.
  - The multiplier is a small subset for the ECC design
  - This focuses on a specific multiplier algorithm
- We describe a CED technique for the entire ECC implementation
  - We do not focus on just one primitive of the design.
  - Our technique is independent of the algorithm for the primitive.

32. Chiou, C., Lee, C., Deng, A., and Lin, J. 2006. Concurrent Error Detection in Montgomery Multiplication over GF(2m). *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* E89-A, 2 (Feb. 2006), 566-574.

Involution property

- The involution property of the inversion block states that inv(inv(x))=x.
  - This is true regardless of the algorithm used to implement inversion.



CED scheme for the inversion block [21].

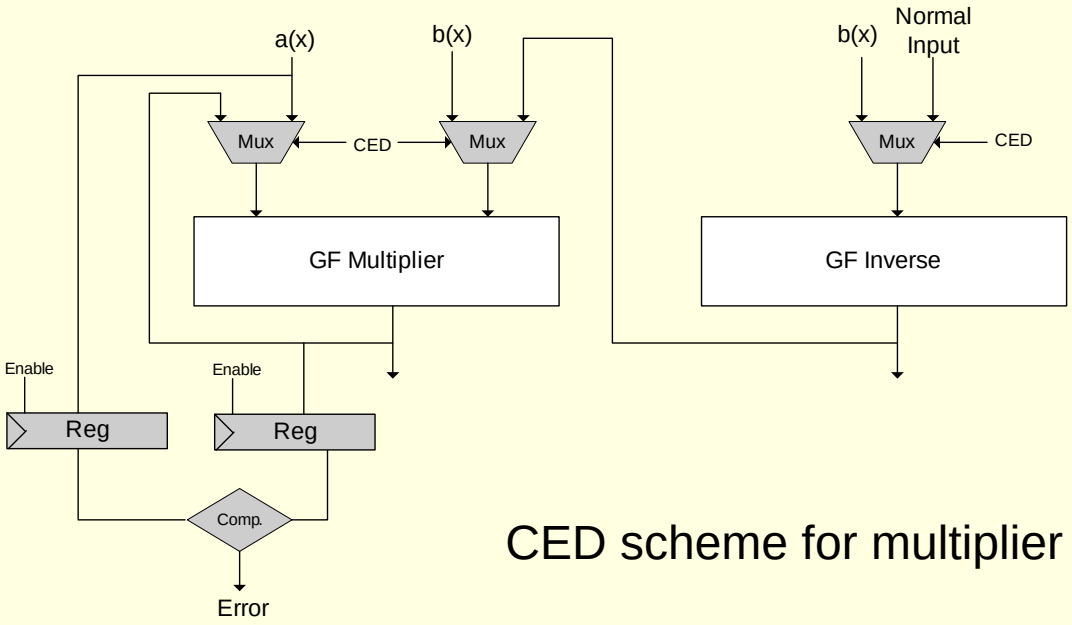
21. Nikhil Joshi, KaijieW u, Jayachandran Sundararajan, and Ramesh Karri. "Concurrent Error Detection for Involutional Functions With Applications in Fault-Tolerant Cryptographic Hardware Design". IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, VOL. 25, NO. 6, JUNE 2006.

Multiplicative inverse property

- The multiplicative inverse property of the inversion block states that inv(x)*x = 1.
- The CED scheme performed for the multiplication block is:

$$(a(x)*b(x))*\ b^{-1}(x) = a(x)*(b(x)*\ b^{-1}(x)) = a(x)$$

- If either the multiplication or the inversion block has a fault and produces the wrong result, the redundant multiplication will not result in *a(x)*.
- We could have implemented this in such a way that it just checks for a constant value of inv(x)*x = 1.
  - We have chosen against this as not to have a single point of failure. That is, stuck at one faults would render the implementation useless.
- Our implementation takes the input data, a(x), to compare against.
  - In normal operation, the input data will be changing and thus the comparator will always have different data to compare against.

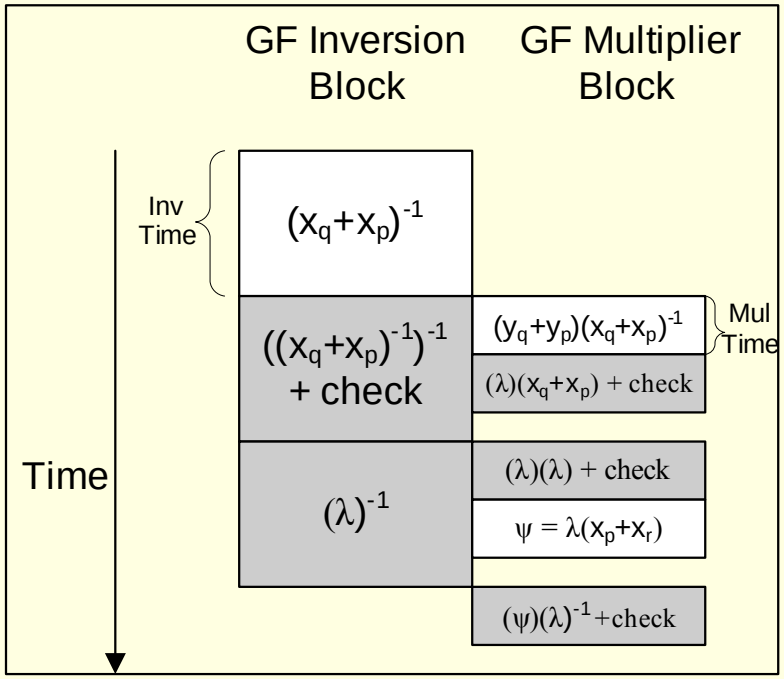


CED scheme for multiplier

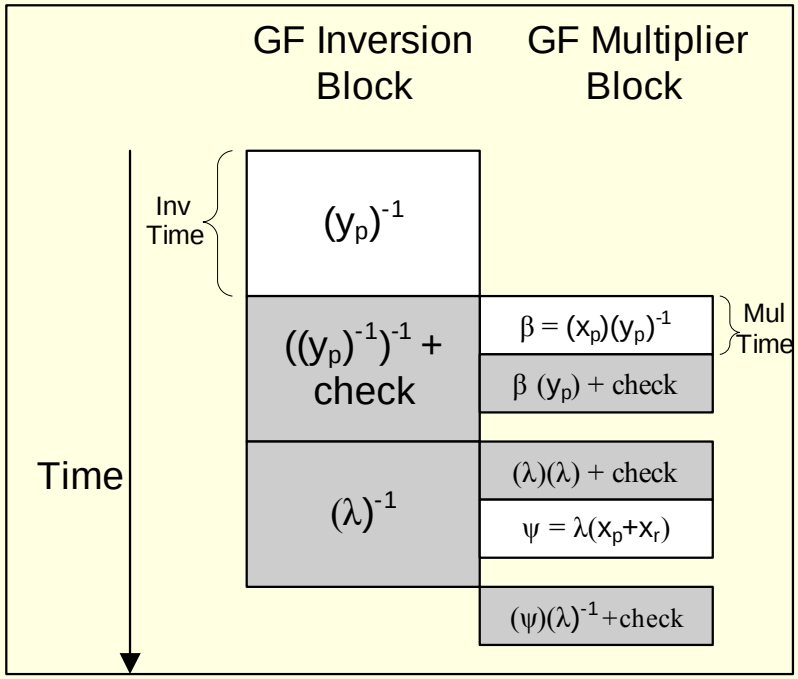
CED architecture for point addition

- The inversion operation is performed on (x_q+x_p) to obtain $(x_q+x_p)^{-1}$.
- $(x_q+x_p)^{-1}$ and (y_q+y_p) are sent to the GF multiplication block to obtain λ .
- Concurrent to this operation, $(x_q+x_p)^{-1}$ is sent to the GF inversion block to perform a CED redundant computation to get $((x_q+x_p)^{-1})^{-1}$ and check if this is equal to the original input (x_q+x_p). Futhermore, (x_q+x_p) is multiplied with λ to check if it will be equal to (y_q+y_p).
- The second normal multiplication operation is performed on λ and (x_p+x_r). To check this operation for errors, λ^{-1} is required and is thus fed into the inversion block.
- Concurrent to this operation, we can check for errors in the square and reduction block. by feeding λ as both the inputs to the multiplier and checking if the result is the same as that obtained using the square and reduce module, failing which an error is reported.
- Finally, the result of the final multiplication and λ^{-1} is fed to the multiplier and checked if the result obtained is (x_p+x_r).

CED Architecture



Point Addition



Point Doubling

Fault Coverage Analysis

- VHDL code was synthesized into a netlist
- Single and Multiple bit faults were injected
- Fault detected if:
  - Output incorrect
  - Error signal asserted (1)
- Fault not detected:
  - Output incorrect
  - Error signal de-asserted (0)
- Results:
  - ~100% Fault Coverage
  - These are our results that we obtained by randomly injecting faults into the design.
  - While we did strive for testing as many locations as possible it did not test all signals of the netlist.
  - There is a small and negligible probability for multiple bit faults to affect the CED hardware in such a way that it goes undetected.

Results

- The CED architecture:
  - Takes three inversion times plus one multiplication time
- The non-CED architecture:
  - Takes one inversion time plus two multiplication time.
- This implies a worst case 90% time overhead, however, we detect all transient and permanent faults.

Timing Results in Normal Implementation

Operation	Time required
Addition	2.248 ns
Squaring and Reduction	0.0051704 µ s
Multiplication	0.375416 µ s
Inversion	0.98912 µ s
Point Addition	1.72711 µ s
Point Doubling	1.75861 µ s
Elliptic Curve Scalar Multiplication (162 Point Doubles, 81 Point Adds)	424.7907 µ s

Area and Time overhead in CED Implementation

	Normal Implementation	CED Implementation	Overhead
Area (gates)	23801	30921	29.91%
Clock Period (ns)	2.248	2.588	15.12%
Point Addition (µs)	1.72711	3.8502	122%
Point Doubling (µs)	1.75861	3.8398	118%
Scalar Multiplication(µs)	424.7907	933.9138	119%

References (1/2)

1. D. Boneh, R. DeMillo and R. Lipton, “On the importance of checking cryptographic protocols for faults”, Proceedings of Eurocrypt, Lecture Notes in Computer Science vol 1233, Springer-Verlag, pp. 37-51, 1997.
2. E. Biham and A. Shamir, “Differential Fault Analysis of Secret Key Cryptosystems”, Proc. of Crypto, 1997.
3. J. Bloemer and J.-P. Seifert, “Fault based cryptanalysis of AES www.iacr.org/eprint/2002/075.pdf.
4. C. Giraud, “Differential Fault Analysis on AES”, http://eprint.iacr.org/2003/008.ps
5. D. Pradhan, On-Line Testing for VLSI, Kluwer Academic Publishers, April 1998
6. I. Biehl, B. Meyer and V. Muller, “Differential Fault Attacks on Elliptic Curve Cryptosystems”, Proc. of Crypto, LNCS 1880, pp.131-146, 2000.
7. M. Ciet and M. Joye, “Elliptic Curve Cryptosystems in the Presence of Permanent and Transient Faults”, Cryptology ePrint Archive: Report 2003/028, available at http://eprint.iacr.org/2003/028.pdf
8. J. Bloemer and J.-P. Seifert, “Sign Change Fault Attacks On Elliptic Curve Cryptosystems,” Cryptology ePrint Archive: Report 2004/227, available at http://eprint.iacr.org/2004/227.pdf
9. N. Joshi, K. Wu and R. Karri, “Concurrent Error Detection Schemes For Involution Ciphers” Cryptographic Hardware and Embedded Systems - CHES 2004 LNCS 3156 Springer 2004
10. R Karri, K. Wu, P. Mishra and Y. Kim, “Concurrent Error Detection of Fault Based Side-Channel Cryptanalysis of 128-Bit Symmetric Block Ciphers,” IEEE Transactions on CAD, Dec 2002
11. G. Bertoni, L. Breveglieri, I. Koren, and V. Piuri, “Error Analysis and Detection Procedures for a Hardware Implementation of the Advanced Encryption Standard,” IEEE Transactions on Comp., vol. 52, No. 4, pp. 492-505, Apr 2003.
12. V.S. Miller, Use of elliptic curves in cryptography, Advances in Cryptology - Crypto '85, Springer-Verlag (1986), 417-426
13. N. Koblitz, Elliptic curve cryptosystems, Mathematics of Computation 48 (1997), 203-209.
14. R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining public key cryptosystems. CACM, 21:120-126, 1978.
15. J. Lopez, R. Dahab, “An Overview of Elliptic Curve Cryptography”, Tech Report, State University of Campinas, 2000, http://citeseer.ist.psu.edu/lop00overview.html
16. A.J. Menezes, “Elliptic Curve Public Key Cryptosystems”, Kluwer Academic Publishers
17. I. Blake, G. Seroussi, and N. Smart, “Elliptic Curves in Cryptography, Cambridge University Press”
18. D. Hankerson, J. L. Hernandez and A. Menezes, “Software Implementation of Elliptic Curve Cryptography over Binary Fields”, Proc. CHES 2000, LNCS 1965, 2000, pp. 1-24.
19. D. V. Bailey and C. Paar, “Efficient arithmetic in finite field extensions with application in elliptic curve cryptography”, Journal of Cryptology, 14(3):153-176, 2001
20. R. Schroeppel, H. Orman, S. O’Malley, “Fast Key Exchange with Elliptic Curve Systems”, Advances in Cryptology- Proceedings of Crypto’95, LNCS, vol.963, pp.43-56, Springer Verlag.

References (2/2)

21. N. P. Smart, “A comparison of different finite fields for use in Elliptic Curve Cryptosystems”, Computers and Mathematics with Applications, vol. 42: 91--100, Oct 2001
22. Y. Choi, H. Kim, M. Kim, Y. Park, and K. Chung, “Design of Elliptic Curve Cryptographic Coprocessor over Binary Fields over GF(2163) for ECC protocols,” ITC-CSCC, 2001.
23. J.H. Patel, L.Y. Fung, “Concurrent Error Detection in ALUs by Recomputing with Shifted Operands,” IEEE Transaction on Comp., Vol. C.31, No.7, pp. 589 – 595. 1982.
24. Int'l Workshop on Fault Diagnosis and Tolerance in Cryptography, 2004, 2005, 2006.
25. IEEE Transactions on Computers, SPECIAL SECTION ON FAULT DIAGNOSIS AND TOLERANCE IN CRYPTOGRAPHY to appear in June 2006
26. E. Normand, “Single event upset at ground level”, IEEE transactions on Nuclear Science, Vol. 43, No.6, pp. 2742–2750, Dec. 1996.
27. S. Buchner, M. Baze, D. Brown, D. McMorrow, J. Melinger, “Comparison of error rates in combinational and sequential logic”, IEEE transactions on Nuclear Science. Vol. 44, No.6, pp. 2209-2216, Dec. 1997
28. A.H. Johnston, “Radiation effects in advanced microelectronics technologies”, IEEE transactions on Nuclear Science, Vol. 45, No. 3, pp. 1339-1354, Jun. 1998.
29. Nikhil Joshi, Jayachandran Sundararajan, Kaijie Wu, Bo Yang, and Ramesh Karri. "Tamper Proofing by Design Using Generalized Involution-Based Concurrent Error Detection for Involutional Substitution Permutation and Feistel Networks". IEEE TRANSACTIONS ON COMPUTERS, VOL. 55, NO. 10, OCTOBER 2006.
30. Nikhil Joshi, KaijieW u, Jayachandran Sundararajan, and Ramesh Karri. "Concurrent Error Detection for Involutional Functions With Applications in Fault-Tolerant Cryptographic Hardware Design". IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, VOL. 25, NO. 6, JUNE 2006.
31. Willi Geiselmann, Adi Shamir, Rainer Steinwandt, Eran Tromer, Scalable hardware for sparse systems of linear equations, with applications to integer factorization, proc. CHES 2005, LNCS 3659, 131--146, Springer, 2005
32. Chiou, C., Lee, C., Deng, A., and Lin, J. 2006. Concurrent Error Detection in Montgomery Multiplication over GF(2m). *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* E89-A, 2 (Feb. 2006), 566-574.