

1-Mar-04 (1)

# CEG5010: A Microcoded Elliptic Curve Cryptographic Processor

1-Mar-04 (2)

## Overview

- Motivation
- Introduction to cryptography
- Mathematical Background
- Introduction to elliptic curves
- Previous Works
- Elliptic curve cryptographic processor (ECP)
- Result
- Conclusion

1-Mar-04 (3)

# Motivation

- Motivation
  - Cryptography is important to the Internet and E-commerce systems
  - ECC is better than other public key cryptosystems
    - Lower memory and computational requirements
    - Higher security
  - Reconfigurable hardware provides high flexibility, low cost, short turnaround time and possibility of upgrading in the future

1-Mar-04 (4)

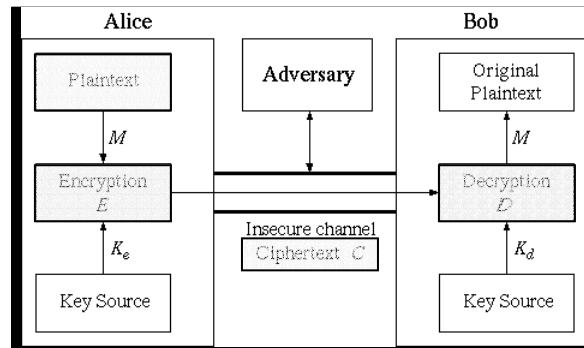
# Overview

- Motivation
- Introduction to cryptography
- Mathematical Background
- Introduction to elliptic curves
- Previous Works
- Elliptic curve cryptographic processor (ECP)
- Result
- Conclusion

1-Mar-04 (5)

# Cryptography

- Keeps a message secret during transmission through untrusted and insecure channel



1-Mar-04 (6)

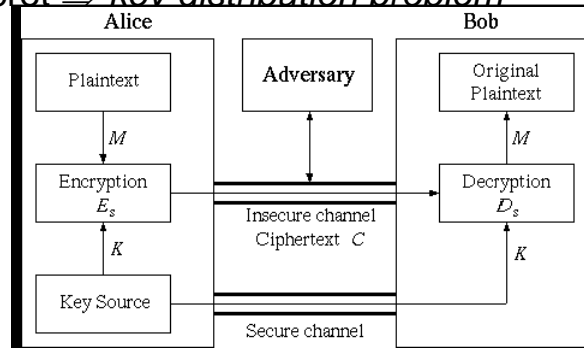
# Cryptography

- Secret key cryptography
  - The encryption key and decryption key are the same
  - Example:
    - Data encryption standard (DES) has been widely used for over 20 years

1-Mar-04 (7)

# Cryptography

- Secret key cryptography
  - The receiver's keys must be distributed in secret  $\Rightarrow$  *key distribution problem*



1-Mar-04 (8)

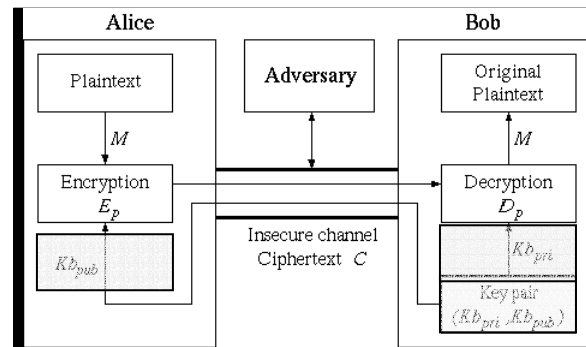
# Cryptography

- Public key cryptography
  - Different key for encryption and decryption
    - Public-key and private-key
  - Key distribution problem is solved
  - Mainly used for key exchange & digital signatures

1-Mar-04 (9)

# Cryptography

- Public key cryptography



1-Mar-04 (10)

# Cryptography

- One-way function
  - Basis of public key cryptosystems
  - Easier to compute a function in one direction than in the reverse direction
  - “Trap door” can make the inverse computation easy  
 $\Rightarrow$  private key
  - Example:
    - Factorization of a integer  $n$  which is the product of two large prime numbers
    - If one of the prime is known, the factorization is easy (One-way function of RSA)

1-Mar-04 (11)

# Cryptography

- Discrete Logarithm Problem (DLP)
  - One-way function of difficult to find a logarithm in a group
  - Definition:
    - For  $(g, y \in G)$ , easy to calculate  $g^x = y$  (any finite cyclic group  $G$  can be used)
    - DLP: compute  $x$  given  $g$  and  $y$  (difficult)

1-Mar-04 (12)

# Cryptography

- RSA
  - Most commonly used public key cryptosystem
  - Based on the problem of factorization  $n$  which is the product of two large primes

1-Mar-04 (13)

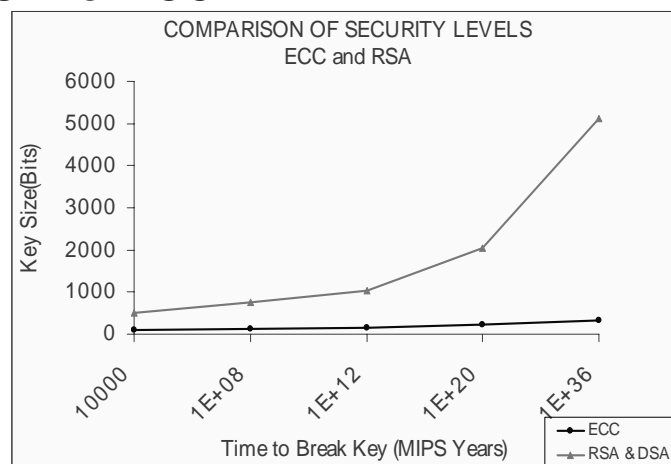
# Cryptography

- Advantages of ECC over RSA
  - Smaller key size for equivalent security
    - Higher security per bit
    - Leads to faster implementations i.e. higher security for the same amount of computation
  - Thought to be more secure (largest ECC and RSA systems broken to date are 108b & 512b respectively)  
Largest effort ever expended in a public key cryptography challenge for solving 108b ECC.  
Amount of work required was about 50 times of 512b RSA.

1-Mar-04 (14)

# Cryptography

## • RSA vs. ECC

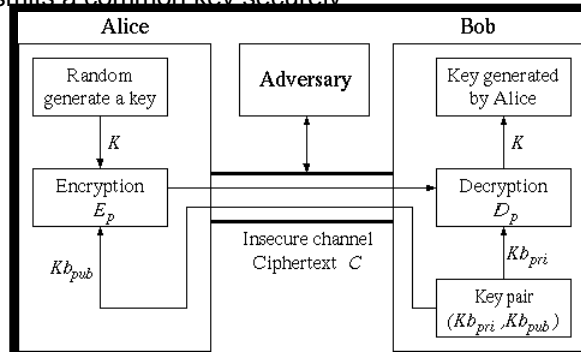


From Certicom Corp.

1-Mar-04 (15)

# Cryptography

- Key exchange protocol
  - Public key cryptosystem solves the key distribution problem
  - Transmits a common key securely



1-Mar-04 (16)

# Cryptography

- Diffie-Hellman key exchange protocol
  - Suppose Alice and Bob want to agree on a common key  $k$ 
    - Public parameters:  $p$  and  $g$
    - Alice generates a secret random integer  $a$  and sends the point  $(g^a \text{ mod } p)$  to Bob
    - Bob generates a secret random integer  $b$  and sends the point  $(g^b \text{ mod } p)$  to Alice
    - Alice and Bob can both compute the key,  $k$
    - $$k = (g^a)^b = (g^b)^a = g^{ab}$$
  - An adversary only knows  $p, g, g^a, g^b$ 
    - To determine  $k$  she would have to solve the discrete logarithm problem

1-Mar-04 (17)

# Cryptography

- Digital signature
  - Digital form of traditional signature
  - Definition:
    - Signer encrypts the document with his private key
    - Decrypts the signed document with signer's public key to verify the signature
  - Main functions:
    - Signer cannot deny the signature
    - The signature is not reusable

1-Mar-04 (18)

# Cryptography

- Secret key vs. Public key cryptography

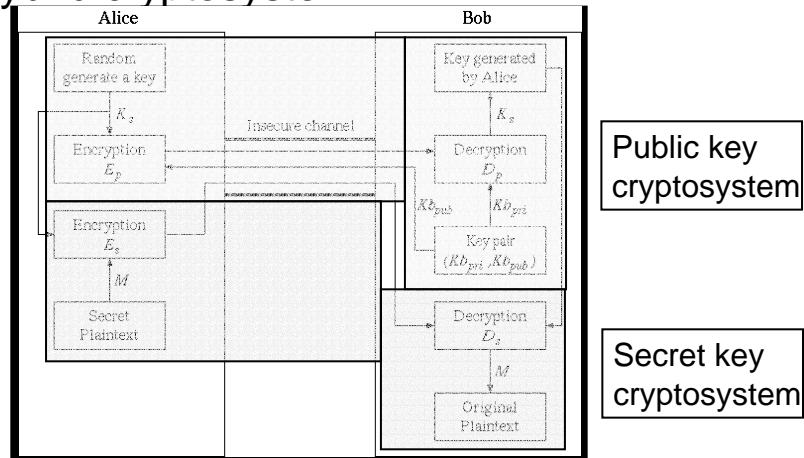
	Secret key	Public key
Speed	Fast	Slow
Key distribution problem	Yes	No

- In practice, a hybrid method is used
  - Combined of secret and public key cryptosystems
  - Used public key cryptosystem to perform key exchange to make a common key which acts as the key of secret key cryptosystem afterwards

1-Mar-04 (19)

# Cryptography

- Hybrid cryptosystem



1-Mar-04 (20)

## Overview

- Motivation
- Introduction to cryptography
- Mathematical Background
- Introduction to elliptic curves
- Previous Works
- Elliptic curve cryptographic processor (ECP)
- Result
- Conclusion

1-Mar-04 (21)

## Mathematical Background

- Operations of ECC are based on finite fields
- Field
  - Consists of a set of elements  $F$  together with two binary operations
    - + (addition), and
    - $\times$  (multiplication)
  - Finite number of elements ( $\#F$ ) in  $F$ 
    - $\Rightarrow$  finite field

1-Mar-04 (22)

## Mathematical Background

- Finite field  $F_{2^n}$ 
  - $\#F = 2^n$
  - Elements can be represented as  $n$ -bit binary number and are in term of a basis
  - Two kinds of bases
    - Polynomial basis
    - Normal basis
  - The elliptic curve processor is in optimal normal basis
    - Normal basis is more suitable for hardware implementation
    - Optimal normal basis gives a minimum complexity

1-Mar-04 (23)

## Mathematical Background

- Normal basis
  - Any element,  $E$  in the  $F_{2^n}$  can be written in terms of a normal basis

$$A = \sum_{i=0}^{n-1} a_i \beta^{2^i}$$

where  $a_i \in F_2, \beta \in F_{2^n}$

- It is a  $n$ -bit binary number
- Efficient for hardware implementation

1-Mar-04 (24)

## Normal Basis Operations

- Addition

$$A + B = \left( \sum_{i=0}^{n-1} a_i \beta^{2^i} \right) + \left( \sum_{j=0}^{n-1} b_j \beta^{2^j} \right) = \sum_{i=0}^{n-1} (a_i + b_i) \beta^{2^i}$$

- Bit-wise exclusive-OR (XOR)

- Squaring

$$A^2 = \left( \sum_{i=0}^{n-1} a_i \beta^{2^i} \right)^2 = \sum_{i=0}^{n-1} a_i (\beta^{2^i})^2 = \sum_{i=0}^{n-1} a_i \beta^{2^{i+1}} = \sum_{i=0}^{n-1} a_{i-1} \beta^{2^i}$$

$$(\beta^{2^{n-1}})^2 = \beta^{2^n} = \beta$$

- Rotate left (ROTL) among the coefficients  $a_i$  of  $\beta$

## Normal Basis Operations

- Multiplication

$$C = A \times B = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} a_i b_j \beta^{2^i} \beta^{2^j} = \sum_{i=0}^{n-1} c_i \beta^{2^i}$$

$$c_k = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \lambda_{ijk} a_i b_j$$

- Refer to thesis for the derivation of  $\lambda_{ijk}$
- The multiplication is defined as cyclic shift to  $\lambda_{ij0}$ , since

$$c_k = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \lambda_{i-k, j-k, 0} a_i b_j = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \lambda_{ij0} a_{i+k} b_{j+k}$$

## Normal Basis Operations

- Multiplication

- Minimum number of non-zero terms  $(2n-1)$  in  $\lambda$  matrix
- ⇒ Optimal Normal Basis (ONB)

– For  $n=5$

$$c_k = a_k b_{1+k} + a_{1+k} b_k + a_{1+k} b_{3+k} + a_{2+k} b_{3+k} + a_{2+k} b_{4+k} + a_{3+k} b_{1+k} + a_{3+k} b_{2+k} + a_{4+k} b_{2+k} + a_{4+k} b_{4+k}$$

- Since it is ONB, there are 9 non-zero terms in  $\lambda$  matrix, thus 9 product terms

1-Mar-04 (27)

## Normal Basis Operations

- Inversion
  - Definition of inversion of  $a$ :

$$aa^{-1} \equiv 1 \pmod{n}$$

- From *Fermat's Little Theorem*

$$a^{2^n-1} = 1 \Rightarrow a^{-1} = a^{2^n-2} = (a^{2^{n-1}-1})^2$$

$$a^{2^{n-1}-1} = \begin{cases} a^{(2^{(n-1)/2}-1)(2^{(n-1)/2}+1)} & n \text{ is odd} \\ a^{2^{(n-2)/2}-1)(2^{(n-2)/2}+1)+1} & n \text{ is even} \end{cases}$$

- By decompose the power of  $a$  continuously, inversion becomes a series of squarings and multiplications
- Total number of multiplications is

$$\lfloor \log_2(n-1) \rfloor + (\# \text{ of bits set in } n+1) - 1$$

1-Mar-04 (28)

## Overview

- Motivation
- Introduction to cryptography
- Mathematical Background
- Introduction to elliptic curves
- Previous Works
- Elliptic curve cryptographic processor (ECP)
- Result
- Conclusion

1-Mar-04 (29)

## Elliptic Curves

- Elliptic Curve over real number ( $\mathfrak{R}$ )

$$y^2 = x^3 + a_4x + a_6$$

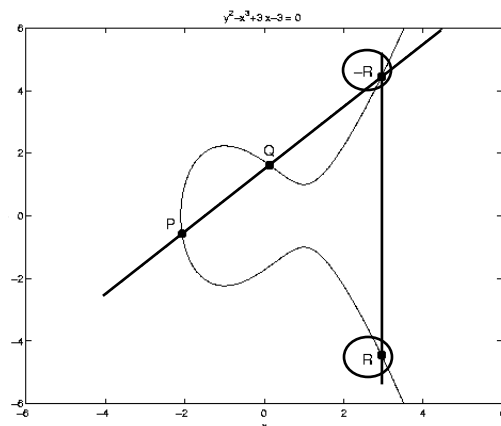
with  $x, y, a_4, a_6 \in \mathfrak{R}$

- Point addition and doubling on elliptic curve in *affine* coordinates are defined geometrically (also known as curve addition and curve doubling in the thesis)

1-Mar-04 (30)

## Elliptic Curves over $\mathfrak{R}$

- Curve addition (ESUM)

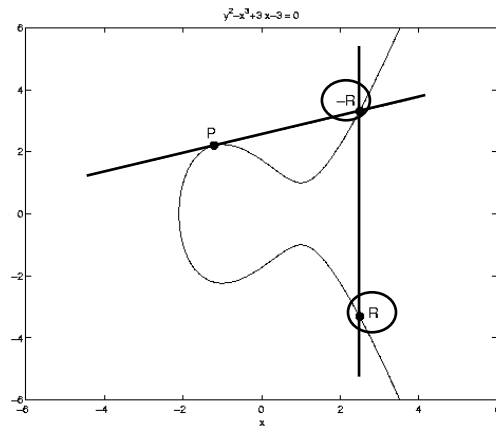


- $R = P + Q$ ,  $P \neq Q$
- line connecting  $P$  and  $Q$  intersects the curve at exactly 1 point  $-R$

1-Mar-04 (31)

## Elliptic Curves over $\mathcal{R}$

- Curve doubling (EDBL)



- $R = P + Q = 2P, \therefore P = Q$
- the tangent to the curve at  $P$  is used

1-Mar-04 (32)

## Elliptic Curves over $\mathcal{R}$

- Curve addition and doubling in affine coordinates

$$x_3 = \begin{cases} \left( \frac{y_2 - y_1}{x_2 - x_1} \right)^2 - x_1 - x_2 & P \neq Q \\ \left( \frac{3x_1^2 + a_4}{2y_1} \right)^2 - x_1 - x_2 & P = Q \end{cases}$$

$$y_3 = \begin{cases} (x_1 + x_3) \left( \frac{y_2 - y_1}{x_2 - x_1} \right) - y_1 & P \neq Q \\ (x_1 + x_3) \left( \frac{3x_1^2 + a_4}{2y_1} \right) - y_1 & P = Q \end{cases}$$

1-Mar-04 (33)

## Elliptic Curves over $F_{2^n}$

- A non-supersingular elliptic curves over finite fields  $F_{2^n}$

with  $a_6 \neq 0$  and  $y^2 + xy = x^3 + a_2x^2 + a_6$

- ECs over  $F_{2^n}$  is practical
  - Binary natural of  $F_{2^n}$
  - Less hardware
    - $F_{2^{155}}$  ECC processor 11,000 gates
    - 512b RSA 50,000 gates
 (these processors have roughly the same level of security)
- Operations of ECs over  $F_{2^n}$  are defined as same as ECs over  $\mathfrak{R}$

1-Mar-04 (34)

## Elliptic Curves over $F_{2^n}$

- In affine coordinates

$$x_3 = \begin{cases} \left( \frac{y_1 + y_2}{x_1 + x_2} \right)^2 + \left( \frac{y_1 + y_2}{x_1 + x_2} \right) + x_1 + x_2 + a_2 & \text{ESUM} \\ \left( x_1 + \frac{y_1}{x_1} \right)^2 + \left( x_1 + \frac{y_1}{x_1} \right) + a_2 & \text{EDBL} \end{cases}$$

$$y_3 = \begin{cases} (x_1 + x_3) \left( \frac{y_1 + y_2}{x_1 + x_2} \right) + x_3 + y_1 & \text{ESUM} \\ (x_1 + x_3) \left( x_1 + \frac{y_1}{x_1} \right) + x_3 + y_1 & \text{EDBL} \end{cases}$$

There are one inversion and two multiplications in both ESUM and EDBL

1-Mar-04 (35)

## Elliptic Curves over $F_{2^n}$

- In projective coordinates
  - A non-supersingular curve E can be equivalently viewed as the set of all points in the projective plane which satisfy

$$y^2z + xyz = x^3 + a_2x^2z^2 + a_6z^3$$

- By using projective coordinates, the inversion operations can be eliminated
- Any point  $(a,b)$  on ECs over  $F_{2^n}$  in affine can be viewed as a 3-tuple  $(a,b,1)$  on ECs in projective
- In projective coordinates,  $(tx,ty,tz) \equiv (x,y,z)$ , with  $t \neq 0$

1-Mar-04 (36)

## Elliptic Curves over $F_{2^n}$

- Conversion between affine and projective
  - From affine to projective

$$A(x, y) = A'(x, y, 1)$$

- From projective to affine

$$P'(a, b, c) = P'\left(\frac{a}{c}, \frac{b}{c}, 1\right) = P\left(\frac{a}{c}, \frac{b}{c}\right) \quad \text{for } c \neq 0$$

1-Mar-04 (37)

## Elliptic Curves over $F_{2^n}$

- Operations in projective coordinates

- ESUM

$$x_3 = AD$$

$$y_3 = CD + A^2(Bx_1 + Ay_1)$$

$$z_3 = A^3z_1$$

$$A = (x_2z_1 + x_1), B = (y_2z_1 + y_1)$$

$$C = A + B, D = A^2(A + a_2z_1) + z_1BC$$

13 multiplications but no inversion
-------------------------------------

- EDBL

$$x_3 = AB$$

$$y_3 = x_1^4A + B(x_1^2 + y_1z_1 + A)$$

$$z_3 = A^3$$

$$A = x_1z_1, B = a_6z_1^4 + x_1^4$$

7 multiplications but no inversion
------------------------------------

1-Mar-04 (38)

## Elliptic Curves over $F_{2^n}$

- Affine vs. Projective

Operation	Affine		Projective	
	ESUM	EDBL	ESUM	EDBL
Field Multiplication	2	2	13	7
Field Inversion	1	1	0	0

1-Mar-04 (39)

## Elliptic Curves over $F_{2^n}$

- Curve multiplication (EMUL)

- Definition:  $Q = cP = \underbrace{P + P + \dots + P}_{c \text{ times}}$  where  $P, Q$  are on ECs

- The “doubling and add” algorithm

- Affine coordinates

INPUT:  $P$  on EC and  $c \in F_{2^n}$   
 OUTPUT:  $Q = cP$

1.  $c = \sum_{i=0}^{n-1} b_i 2^i, b_i \in \{0,1\}$  and  $b_k$  is the MSB set ( $k < n$ )
2. Set  $Q \leftarrow P$
3. For  $i$  from  $k-1$  downto 0
  - 3.1 Set  $Q \leftarrow Q + Q$  (Affine EDBL)
  - 3.2 If  $b_i = 1$  then
    - 3.2.1 Set  $Q \leftarrow Q + P$  (Affine ESUM)
4. Return  $Q$

Inversion operations occur

1-Mar-04 (40)

## Elliptic Curves over $F_{2^n}$

- Curve multiplication (EMUL)

- The “doubling and add” algorithm

- Projective coordinates

INPUT:  $P$  on EC and  $c \in F_{2^n}$   
 OUTPUT:  $Q = cP$

1.  $c = \sum_{i=0}^{n-1} b_i 2^i, b_i \in \{0,1\}$  and  $b_{k-1}$  is the MSB set ( $k \leq n$ )
2. Convert  $P$  to projective  $P'$
3. Set  $Q' \leftarrow P'$
4. For  $i$  from  $k-2$  downto 0
  - 3.1 Set  $Q' \leftarrow Q' + Q'$  (Projective EDBL)
  - 3.2 If  $b_i = 1$  then
    - 3.2.1 Set  $Q' \leftarrow Q' + P'$  (Projective ESUM)
5. Convert  $Q'$  to affine  $Q$
6. Return  $Q$

No inversion operation

1 inversion operation

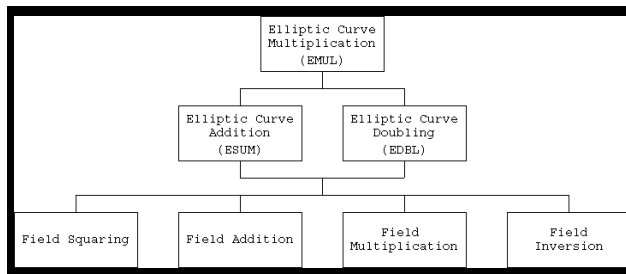
$k + (\# \text{ of bits set in } c) - 2$

- Number of point additions/doublings required is

1-Mar-04 (41)

## Elliptic Curve Operations

- Field Operations
  - Addition
  - Squaring
  - Multiplication
  - Inversion
- Curve Operations
  - Curve addition (ESUM)
  - Curve doubling (EDBL)
  - Curve multiplication (EMUL)



Basic operation of an EC processor is EMUL

1-Mar-04 (42)

## EC Discrete Logarithm Problem

- Elliptic curve discrete logarithm Problem (ECDLP)
  - Definition:
    - For  $(Y \in G)$ , easy to calculate  $Q = xY$  ( $G$  is elliptic curve group)
    - ECDLP: compute  $x$  given  $Q$  and  $Y$  (difficult)
  - ECDLP is particularly difficult if the underlying group is based on points on an elliptic curve
    - No sub-exponential time algorithm known which can solve the ECDLP
    - Index calculus methods offer sub-exponential time algorithms for some other groups but not elliptic curve group

1-Mar-04 (43)

# Elliptic Curve Cryptography

- e.g. Alice wants to send a message  $M$  to Bob secretly

Public parameters: elliptic curve  $E$ , a point  $Q$  on  $E$   
 Public-key of Bob:  $(P, Q)$  where  $P = xQ$ ,  $P, Q$  on  $E$   
 Private-key of Bob:  $x$   
 Alice: generates random integer  $z$   
 $A(x_a, y_a) = zQ$  and  $T(x_t, y_t) = zP$   
 $B(x_b, y_b) = (x_t, x_m, y_t, y_m)$  where  $M$  is embedded to  $E$  giving  $(x_m, y_m)$   
 ciphertext  $(A, B)$  is sent to Bob

Bob: receives  $(U, V)$ ,  $U = (x_u, y_u)$  and  $V = (x_v, y_v)$   
 $xU = xzQ = zP = T(x_t, y_t)$

$$\Rightarrow x_m = \frac{x_v}{x_t}, y_m = \frac{y_v}{y_t}$$

1-Mar-04 (44)

# EC Key Exchange Protocol

- Elliptic curve Diffie-Hellman key exchange (ECDH)
  - Suppose Alice and Bob wish to agree on a common key  $k$

Public parameters: elliptic curve  $E$  and point  $P$  on  $E$   
 Alice generates a secret random integer  $c_A$  and sends the point  $(c_A P)$  to Bob  
 Bob generates a secret random integer  $c_B$  and sends the point  $(c_B P)$  to Alice

Alice and Bob can both compute the key,  $k$

$$k = c_A (c_B P) = c_B (c_A P)$$

1-Mar-04 (45)

## Overview

- Motivation
- Introduction to cryptography
- Mathematical Background
- Introduction to elliptic curves
- Previous Work
- Elliptic curve cryptographic processor (ECP)
- Result
- Conclusion

1-Mar-04 (46)

## Previous Designs

- Hardware implementation
  - Field processor
    - A field processor over  $F_{2^{155}}$  [AMV93]
      - 11,000 gates, 40 MHz
    - Improved multipliers are reported in [OP99,Gro01]
  - Curve processor
    - A FPGA-based EC processor over  $F_{(2^n)^m}$  was developed [Ros98b]
    - A scalable, FPGA-based EC processor over  $F_p$  was reported in [OP01]
- Software implementation
  - Full ECC package with many algorithmic optimizations [Ros98a]
    - Used to evaluate the performance and correctness of the design

High bandwidth interface is required to supply the processor with its data and the field operations are restricted to field of fix  $n$

1-Mar-04 (47)

## Overview

- Motivation
- Introduction to cryptography
- Mathematical Background
- Introduction to elliptic curves
- Previous Works
- Elliptic curve cryptographic processor (ECP)
- Result
- Conclusion

1-Mar-04 (48)

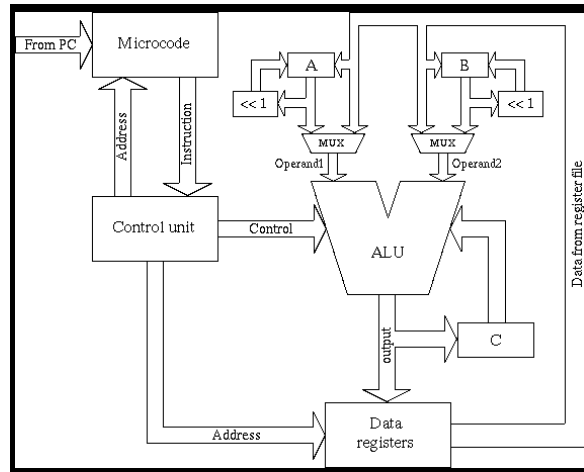
## Hardware Platform

- Design tested on Annapolis Micro Systems Wildstar Board
  - PCI interface
  - Consists of
    - 3 Processing elements, PEs (Xilinx Virtex FPGA XCV1000-6)
    - Each PE has 128-kbits of BlockRAM and 6144 CLBs (12288 slices) which is equivalent to more than one million system gates

1-Mar-04 (49)

# Elliptic Curve Processor

- Architecture
  - ALU
  - Register file
  - Control
  - Microcode register

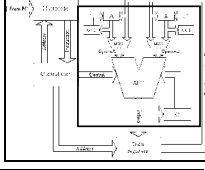


1-Mar-04 (50)

# Elliptic Curve Processor

- Datapath
  - Microcode Sequencer
    - Fetch instruction from microcode register
    - Microcode controls datapath (register file, ALU etc)
    - Update PC
  - Register file
    - $16 \times n$ -bit dual ported registers implemented using Xilinx Virtex distributed memory
  - ALU
    - Field addition (XOR)
    - Field squaring (rotate left)
    - Field multiplication

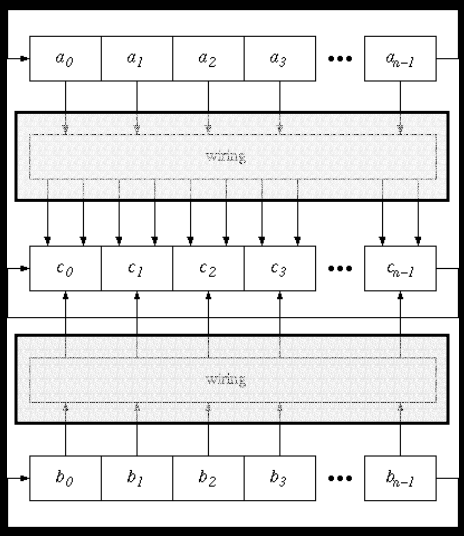
# ALU



- Multiplication implementation
 
$$c_k = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \lambda_{ij0} a_{i+k} b_{j+k}$$

$$\Rightarrow c_k = \sum_{j=0}^{n-1} b_{j+k} \sum_{i=0}^{n-1} \lambda_{ij0} a_{i+k}$$

$$\Rightarrow F(k) = b_{2k} \sum_{i=0}^{n-1} \lambda_{ik0} a_{i+k}$$
  - It defines the connection between register A,B,C
  - Property of an ONB:
    - $a_i$  has a max. fanout of 4
      - High speed



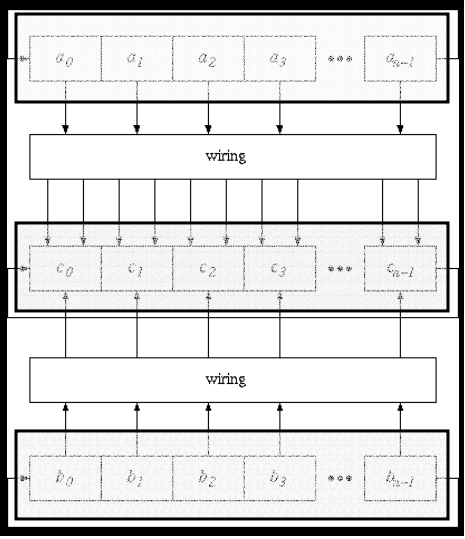
1-Mar-04 (52)

# ALU

- Multiplication Implementation
 
$$c_k = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \lambda_{ij0} a_{i+k} b_{j+k}$$

$$\Rightarrow c_k = \sum_{j=0}^{n-1} b_{j+k} \sum_{i=0}^{n-1} \lambda_{ij0} a_{i+k}$$

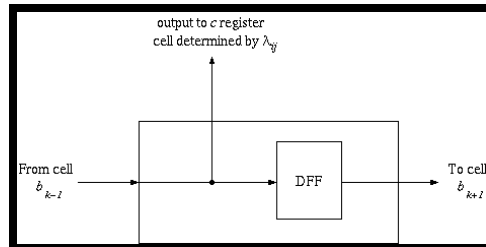
$$\Rightarrow F(k) = b_{2k} \sum_{i=0}^{n-1} \lambda_{ik0} a_{i+k}$$
  - Cyclic shift on A,B,C in each cycle



1-Mar-04 (53)

# ALU

- Multiplication Implementation
  - Structure of  $b_k$

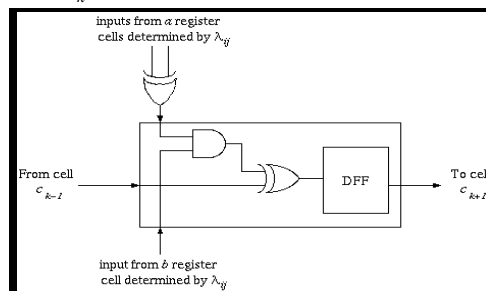


- D-type flip flop
- Cyclic shift on register A and B

1-Mar-04 (54)

# ALU

- Multiplication Implementation
  - Structure of  $c_k$



- Each cell calculate: 
$$b_{j+k} \sum_{i=0}^{n-1} \lambda_{ij} a_{i+k}$$
- After  $n$  cycles: 
$$c_k = \sum_{j=0}^{n-1} b_{j+k} \sum_{i=0}^{n-1} \lambda_{ij} a_{i+k}$$

# ALU

- Multiplication implementation

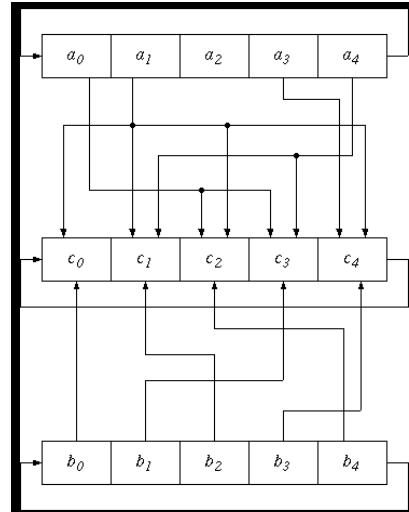
- E.g  $n=5$

- From multiplication table  $\lambda_{ijk}$   
and 
$$F(k) = b_{2k} \sum_{i=0}^{n-1} \lambda_{ik0} a_{i+k}$$

$$\begin{aligned} F(0) &= b_0(a_1), \\ F(1) &= b_2(a_1+a_4), \\ F(2) &= b_4(a_0+a_1), \\ F(3) &= b_1(a_4+a_0), \\ F(4) &= b_3(a_1+a_3) \end{aligned}$$

$$\begin{aligned} c_k &= b_k(a_{k+1}) + b_{k+1}(a_k + a_{k+3}) + b_{k+2}(a_{k+3} + a_{k+4}) \\ &\quad + b_{k+3}(a_{k+1} + a_{k+2}) + b_{k+4}(a_{k+2} + a_{k+4}) \end{aligned}$$

which is the same as the equation shown in previous slide



# Normal Basis Operations

- Multiplication

- Minimum number of non-zero terms  $(2n-1)$  in  $\lambda$  matrix

$\Rightarrow$  Optimal Normal Basis (ONB)

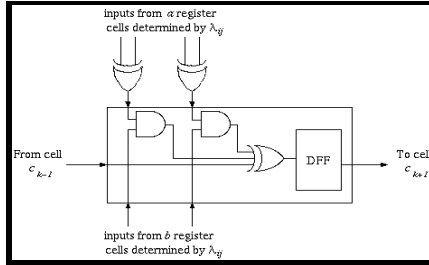
- For  $n=5$ ,

$$\begin{aligned} c_k &= a_k b_{1+k} + a_{1+k} b_k + a_{1+k} b_{3+k} + a_{2+k} b_{3+k} + a_{2+k} b_{4+k} + \\ &\quad a_{3+k} b_{1+k} + a_{3+k} b_{2+k} + a_{4+k} b_{2+k} + a_{4+k} b_{4+k} \end{aligned}$$

- Since it is ONB, there are 9 non-zero terms in  $\lambda$  matrix, thus 9 product terms

# ALU

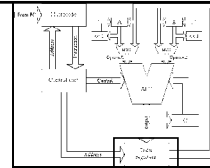
- Parallel multiplier
  - Field multiplication process can be paralleled easily
  - Instead of calculating  $c_k$  by one set of inputs from register A and B, we determine  $c_k$  by  $p$  sets of inputs



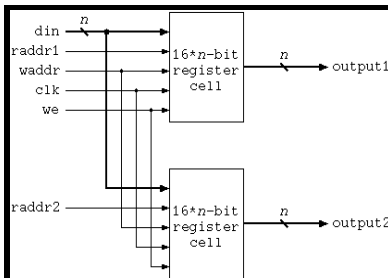
$$\lfloor n/p \rfloor + 2$$

- Increase the parallelism by a factor  $p$ , the multiplier logic can be duplicated  $p$  times and reduce the number of cycle to

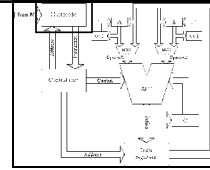
# Register File



- Stores the parameters used in ECC
  - A  $16 \times n$ -bit dual ported registers implemented using Xilinx Virtex distributed RAM
  - Updates at the same time
  - Outputs two  $n$ -bit operands simultaneously

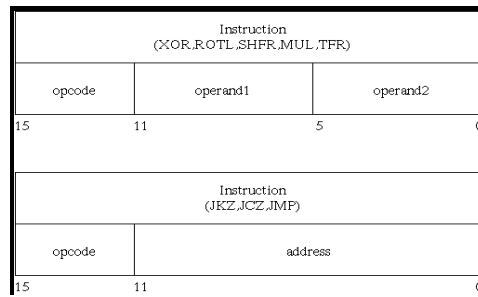


# Microcode



- For performing high level elliptic curve multiplication
  - Stored in Virtex BlockRAMs so control logic does not use FPGA logic resources
  - Can be changed without re-compilation of the processor
  - Algorithmic optimizations can be performed entirely in microcode
  - 16-bit wide microcode

Operation	Action
NOP	No operation
XOR r1,r2	$r0 \leftarrow r1 \oplus r2$
ROTL r1,r2	$r2 \leftarrow r1$ rotate left
SHFR r1,r2	$r2 \leftarrow r1$ right shift
MUL r1,r2	$r0 \leftarrow r1 \times r2$
TFR r1,r2	$r2 \leftarrow r1$
JKZ,JCZ,JMP	$PC \leftarrow$ new address



1-Mar-04 (60)

# Development Environment

- Parameterized Module Generator
  - ECP with different size
    - Generates the codes of elliptic curve processors for any  $n$  (key size) with ONB
    - Processors with different strength of security can be generated to fit different requirements
  - Parallelized multiplier
    - Generates processor with different level of parallelism of multiplier
    - High resource utilization
    - Different speed/area tradeoffs can be made

1-Mar-04 (61)

## Development Environment

- Microcode Toolkit
  - Facilitates the microcode development
  - Simulator and debugger
    - Supports all microcode instructions
    - Supports setting breakpoints
    - Supports single stepping
  - Assembler
    - Two pass symbolic assembler
    - Converts symbolic input to binary microcode

1-Mar-04 (62)

## Development Environment

- Bitstream reconfiguration
  - Modifies the bitstream to change the contents of the register file
  - Initializes the register file
  - Removes the circuitry to download the parameters from host PC to FPGA

1-Mar-04 (63)

## Results

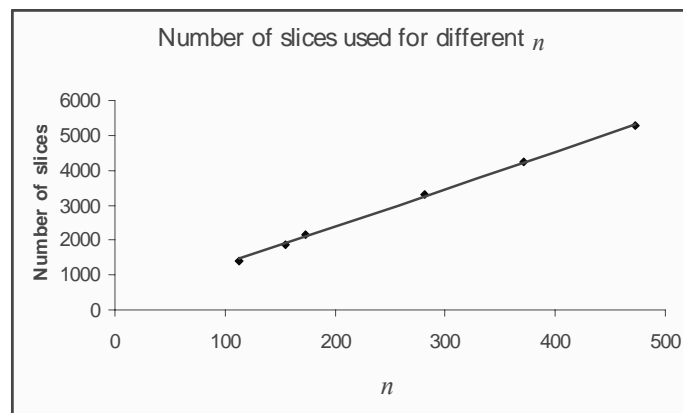
- ECP was generated using parameterized module generator
- Synthesize and implemented by Synopsys FPGA Express 3.4 and Xilinx Foundation 3.2i respectively
- ECP with serial multiplier (XCV1000 has 12288 slices)

$n$	# of slices	Reported Max. frequency (MHz)
113	1466	31
155	1868	30
173	2148	28
281	3315	26
371	4247	22
473	5264	18

1-Mar-04 (64)

## Results

- Resource requirements are linear with  $n$



1-Mar-04 (65)

## Results

- Compared with software

(SUN Enterprise E4500 with UltraSPARC-II 400 MHz with 8 GBytes of RAM)

$n$	SW time( $ms$ )	HW time( $ms$ )	Speed-up
113	27.6	4.3	<b>6</b>
155	63.2	8.3	<b>8</b>
173	86.6	11.1	<b>8</b>

- Projective vs. Affine coordinates

$n$	# of cycles (affine)	# of cycles (projective)	HW time affine( $ms$ )	HW time projective( $ms$ )	Projective: Affine
113	148581	134484	4.8	4.3	<b>0.9</b>
155	324717	249879	10.8	8.3	<b>0.77</b>
173	402926	310043	14.4	11.1	<b>0.77</b>

1-Mar-04 (66)

## Results

- Dynamic instruction counts and frequencies

$n$	Projective			Affine		
	113	155	173	113	155	173
NOP	291	391	444	291	391	444
XOR	616	847	946	784	1078	1204
MUL	128820 (95.79%)	242112 (96.89%)	301368 (97.2%)	127680 (85.93%)	288288 (88.78%)	359136 (89.13%)
ROTL	673	925	1033	12825	24102	30015
TFR	3625	4972	5548	5432	7931	8858
JKZ	113	155	173	113	155	173
JCZ	111	153	171	111	153	171
JMP	111	153	171	111	153	171
SHFR	123	170	188	1233	2465	2753
Total	134484	249879	310043	148581	324717	402926

# Results

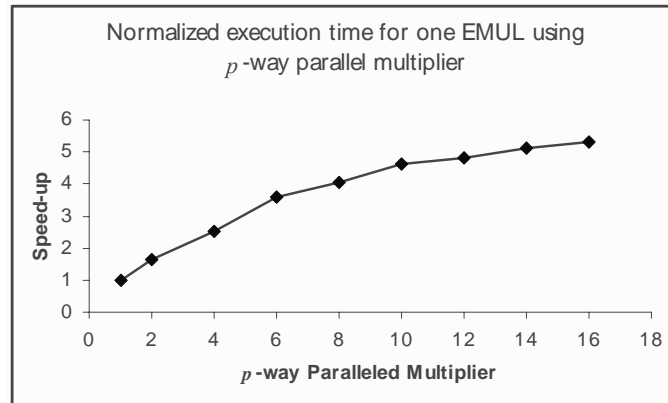
- ECP with parallel multiplier
  - $n = 113$

$p$ -way paralleled	slices	cycles	Time(ms)
1	1410	134484	<b>4.3</b>
2	1860	71204	<b>2.6</b>
4	1970	39564	<b>1.7</b>
6	2076	28264	<b>1.2</b>
8	2182	23744	<b>1.06</b>
10	2300	20354	<b>0.93</b>
12	2434	18094	<b>0.89</b>
14	2515	16964	<b>0.84</b>
16	2614	15833	<b>0.81</b>

# Results

- ECP with parallel multiplier
  - $n = 113$

$$\text{Speedup} = \frac{\text{Time}(p=1)}{\text{Time}(p>1)}$$



1-Mar-04 (69)

## Conclusion

- An elliptic curve processor over  $F_{2^n}$  using an optimal normal basis was developed
  - To perform a scalar multiplication on elliptic curve which is the central operation of ECC
  - Curve multiplication over  $F_{2^n}$  with  $n=113,155$  and  $173$  was successfully tested on the hardware with reported frequencies 31,30 and 28 MHz respectively
  - Curve multiplication in projective coordinates has 10% improvement over affine ( $n=113$ ) and 23% for ( $n=155,173$ )
  - Projective implementation with serial multiplier gave 6-8 times improvement over optimized software implementation
  - 16-way parallel multiplier ( $n=113$ ) gave 5 times improvement over serial multiplier and linear improvement when degree of parallelism  $\leq 6$
  - More flexible since the design used reconfigurable nature of FPGA

1-Mar-04 (70)

## Conclusion

- A parameterized module generator can generate field processors using an ONB for arbitrary  $n$  using multiplier with different speed/area tradeoffs
- Microcode approach can implement curve operation on top of field operations
- High I/O requirement interface is not needed
- Algorithmic optimizations can be done in microcode

## Publications

- Paper

P. H. W. Leong, and K. H. Leung, “A Microcoded Elliptic Curve Processor using FPGA

Technology,” IEEE Transactions on Very Large Scale Integration Systems (to appear). Available from

[http://www.cse.cuhk.edu.hk/~phwl/papers/ecc\\_tvls\\_i02.pdf](http://www.cse.cuhk.edu.hk/~phwl/papers/ecc_tvls_i02.pdf)

- Thesis

<http://www.cse.cuhk.edu.hk/~phwl/students/khleu>