# Fast Scalar Multiplication on Elliptic Curves fo Sensor Nodes

**Youssou FAYE**
**Hervé GUYENNET**
**Yanbo SHOU**
**Université de Franche-Comté**

Besançon le 24 octobre 2013

# TABLE OF CONTENTS

# Sensor Node

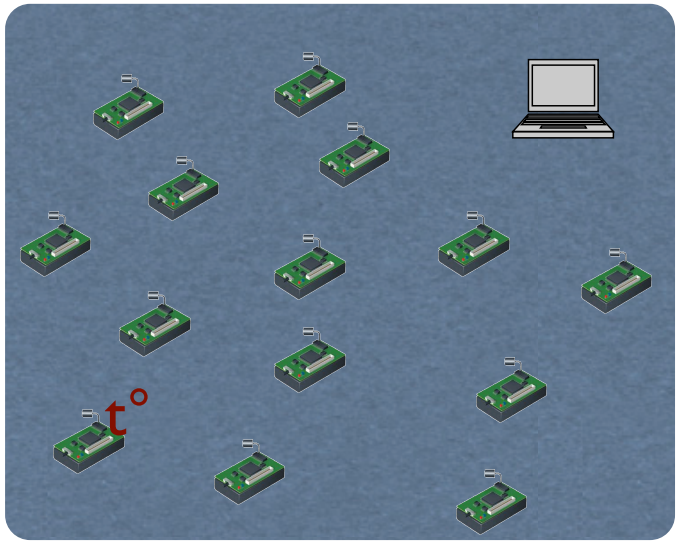
- Limited Computation Power (Microcontroller)
- Sensor
- Low Power Transmission
- Limited Memory (RAM, ROM)
- Low Energy (batteries)
- Example: Telosb

- Processor MSP430 8MHz
- RAM 10 Ko, ROM 48Ko
- 802.15.4/ZigBee RF 2.4 to 2.4835GHz
- Two AA batteries

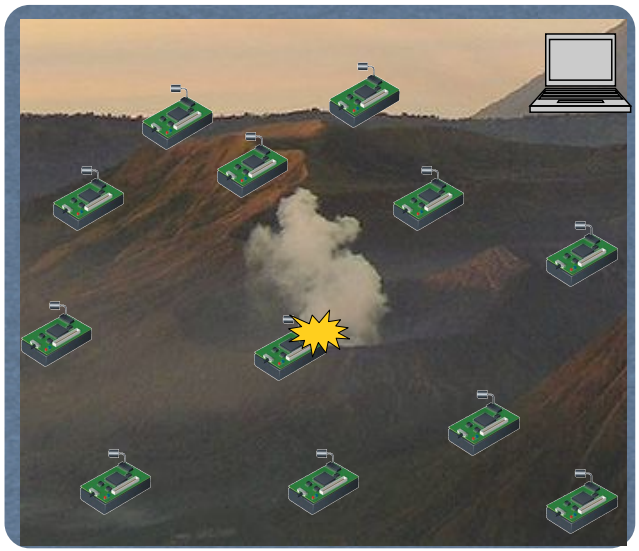
# Wireless Sensor Networks

- Large Number of Sensor Nodes
- Wireless Communication
- Applications: Data Collection, Monitoring
- Example of Applications:

Time-driven Application




Event-driven Application



# Security Challenge in WSNs

- **Minimization of:**
  - Communication
  - Memory Storage
  - Computation

**Energy**

- **Approaches**
  - Symmetric key cryptography
  - Public-key cryptography feasible with ECC for sensor

# Elliptic CurveDiscrete Logarithm

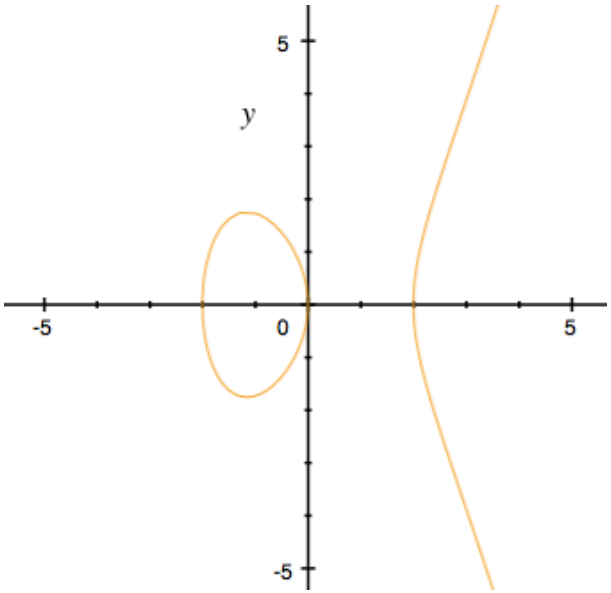
- **Integer Factorisation: N=pq ( p, q, 2 primes)**
  - RSA (1977)
  - Key Generation, Encrytion/Decryption, Signature
- **Discrete Logarithm: $y=g^x$ (y,g 2, integers)**
  - Key Generation (Diffie-Hellman 1976),
  - Encrytion/Decryption (ElGamal 1984),
  - Signature (DSA 1991 (NIST*))
- **Elliptic Curve Discrete Logarithm: Q=dP (Q,P, 2points)**
  - Key Generation (Diffie-Hellman),
  - Encrytion/Decryption (ElGamal),
  - Signature (ECDSA 1992 Scott Vanstone NIST*)

# ECC Introduction (1)

- **Elliptic Curves: 1980 by Miller and Koblitz**
  - Fast Computation
  - Short Key
  - Same Security Level than RSA
- **ECC vs RSA**

| | Security level | Signature | Key Generation |
|---|---|---|---|
| **RSA (IF)** | 1024bits | 315,9ms | 319,4ms |
| **ECC(EDL)** | 160bits | 67,91ms | 44,6ms |
| **SKIPJACK** | 80bits | | |

# ECC Introduction (2)




$y^2=x^3-4x$

## General Weierstrass Equation

E: $y^2=a_1xy+a_3y=x^3+a_2x^2+a_4x+a_6$

## Weierstrass Equation: cryptography

E: $y^2 = x^3+ax+b$

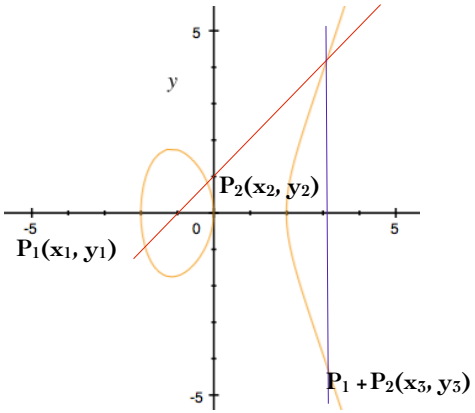
- Defined in finit field $F_q$;
- $q= p^m$, with p prime;
- If m=1, p≠ 2 ou 3, $F_q$ is a prime field;
- If $q=2^m$, Fq is a binary field

# ECC Introduction (3)

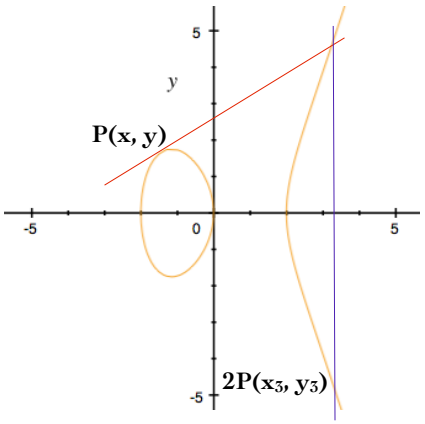
- **ECC on finit prime field $F_p$**
  - E: $y^2 = x^3+ax+b \pmod p$, avec $\Delta=4a^3+27b^2 \neq 0$
- **Abelian group ($E(F_p)$, +)**
  - $(E(F_p), +)=\{(x,y) \in F_p \times F_p : y^2 = x^3+ax+b \pmod p\} \cup \{\infty\}$
  - Addition: $P_1+P_2= P_3$, $P_3 \in E(F_p)$
  - Identity: $\{\infty\}$, $P+ \infty= P$
  - Inverse: $P+(-P)= \infty$

# ECC Introduction (4)

- ECC Arithmetic Level




Addition




Doubling

## Finit Field Arithmetic

- Addition/subtraction, multiplication, squaring inversion in $F_P$ (on coordonates)

## Points Arithmetic

- Addition
  - $x_3=\lambda^2-x_1-x_2$, $y_3=\lambda(x_1-x_3)-y_1$
  - $\lambda= (y_2-y_1)/(x_2-x_1)$
- Doubling (Tripling, quadrupling etc...)
  - $x_3=\lambda^2-2x_1$, $y_3=\lambda(x_1-x_3)-y_1$
  - $\lambda= 3x_1^2+a/2y_1$

## Scalar Arithmetic

- kP, with k an integer number
- $kP=\underbrace{P+P+.....+P}_{k}$

# Scalar Multiplication

- Q=kP, k large integer( minimum 160bits)
- The dominant Operation in ECC
  - Key Generation
  - Encryption / Decryption
- Costly Operation, for Embedded Devices

# Fast Scalar Multiplication Methods

## Classic Binary Algorithm is the Widely Used Techniques

- Double-and-Add Algorithm
- Non-Adjacent Form (NAF, wNAF)
- Slinding Windows Method
- etc..

## Example:Double-and-Add

Input $k = (k_{l-1}, k_{l-2}, \ldots\ldots, k_1, k_0)_2$,
$P \in E(F_P)$
Output $kP$
**Begin**
$Q \leftarrow \infty$;
**For** i from 0 to l-1 **do**
IF $k_i=1$ **Then**
$Q \leftarrow Q + P$;
**EndIF**
$P \leftarrow 2P$;
**EndFor**
**Return** (Q);
**End**

# Presentation (1)

## Based on Point Order and Point Inverse

- P Generator Point with Order n (order of P= #P=n)
- $\lfloor\ \rfloor$ = Integer part function

1) If $k > n$, $kP = dP$ where $d=(k-\lfloor k/2 \rfloor.n)$

2) If $k \in ]\lfloor n/2 \rfloor, n-1]$, $\mathbf{k}P=\mathbf{d}P$ where $\mathbf{d}=(k-n)$

3) If $k \in ]0, \lfloor n/2 \rfloor]$, $\mathbf{k}P = \mathbf{d}P$ where $\mathbf{d} = k$

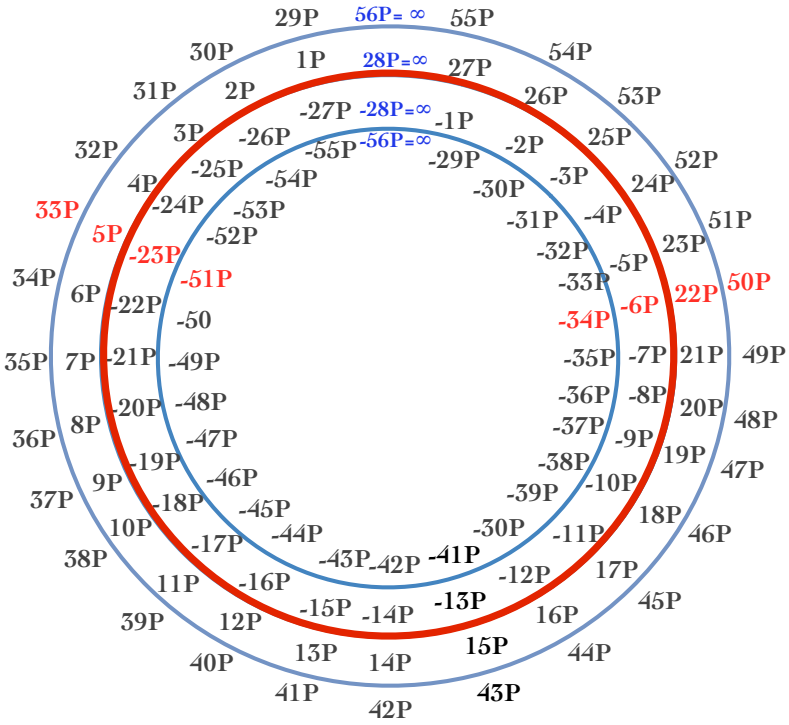
4) If $k=n$ or $-n$, $kP = \infty$

5) If $k \in ]-(n-1), -\lfloor n/2 \rfloor[$, $kP=\mathbf{d}P$ where $\mathbf{d}=(n+k)$

6) If $k \in ]-\lfloor n/2 \rfloor, 0[$, $\mathbf{k}P = \mathbf{d}P$ where $\mathbf{d} = k$,

7) If $k < -n$, $kP = \mathbf{d}P$ where $\mathbf{d}=k+n.\lfloor |k|/2 \rfloor$

## Example: #P(0,1)=28
## $E(F_{23})$: $y^2 = x^3+x+1$

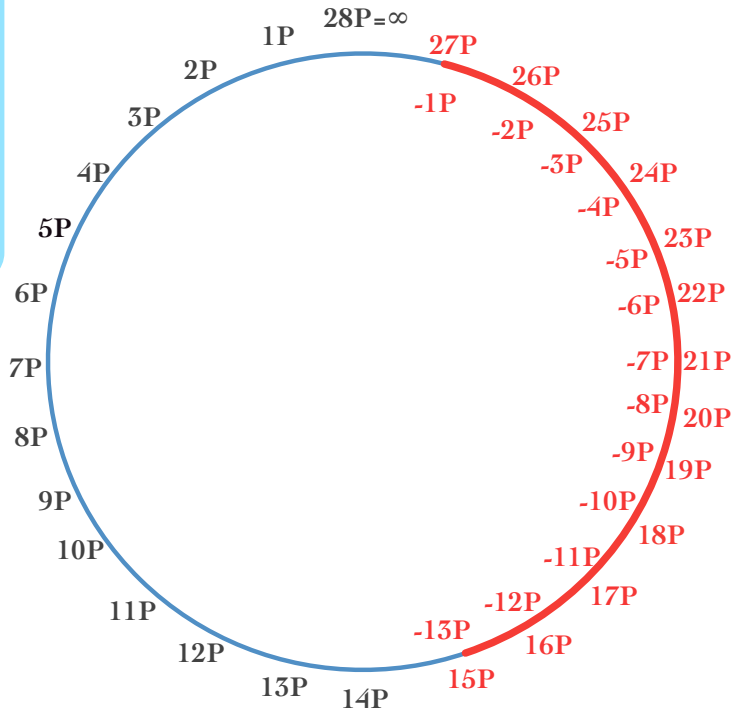


# Presentation (2)

- P Point with Order $n$ (#P=$n$)
- 2) If k ∈ ]⌊n/2⌋, $n$-1], **k**P=**d**P where **d**=(k−n)
- 3) If k ∈ ]0, ⌊n/2⌋], **k**P = **d**P where **d** = k

**Example: E($F_{23}$): $y^2 = x^3+x+1$**

- P(0,1) Generator point, with #P=28
- 26P(6,4) <=> 2P(6,-4)
- 27P(0,-1) <=> P(0,1) (almost free)



# Analytical Evaluation (1)

(1) $$\sum_{k=1}^{n-1} kP = \sum_{k=1}^{\lfloor n/2 \rfloor - 1} kP + \lfloor \frac{n}{2} \rfloor P + \sum_{k=\lfloor n/2 \rfloor + 1}^{n-1} kP$$

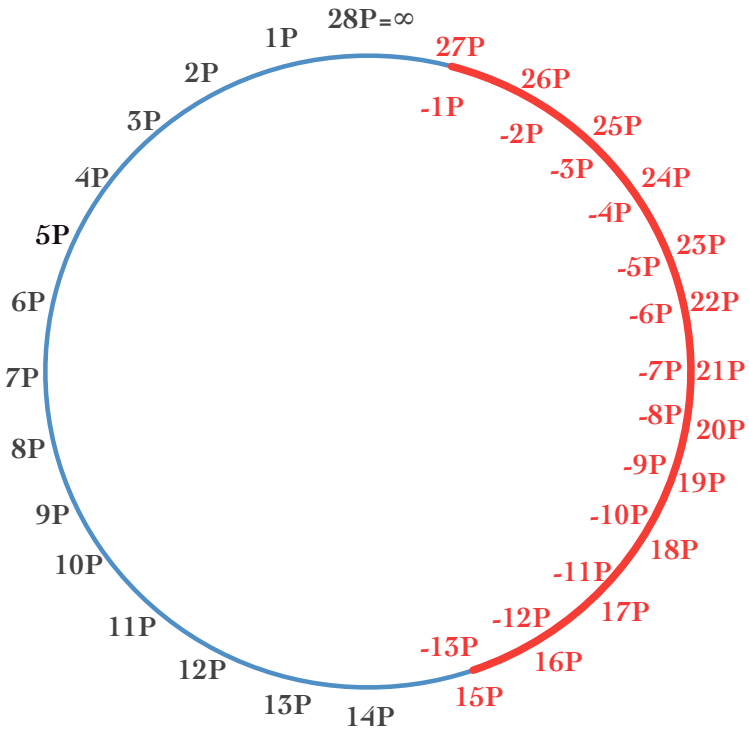
Where $$\sum_{k=\lfloor n/2 \rfloor + 1}^{n-1} kP = \sum_{k=1}^{\lfloor n/2 \rfloor - 1} kP + 2 \sum_{k=1}^{\lfloor n/2 \rfloor - 1} kP$$

(2) $$\sum_{k=1}^{n-1} kP = 2 \sum_{k=1}^{\lfloor n/2 \rfloor - 1} kP + \lfloor \frac{n}{2} \rfloor P + 2 \sum_{k=1}^{\lfloor n/2 \rfloor - 1} kP$$

Replaced $$\sum_{k=\lfloor n/2 \rfloor + 1}^{n-1} kP$$ by $$\sum_{k=1}^{\lfloor n/2 \rfloor - 1} |k|P$$

(3) $$\sum_{k=1}^{n-1} kP = \sum_{k=1}^{\lfloor n/2 \rfloor - 1} kP + \sum_{k=1}^{\lfloor n/2 \rfloor - 1} |k|P + \lfloor \frac{n}{2} \rfloor P$$

(4) Speed-up = $$2 \sum_{k=1}^{\lfloor n/2 \rfloor - 1} kP$$



# Analytical Evaluation (2)

- Speed-up for all k= $2\sum_{k=1}^{\lfloor n/2 \rfloor - 1} kP$

Speed-up for a given k=2(k-(n/2))

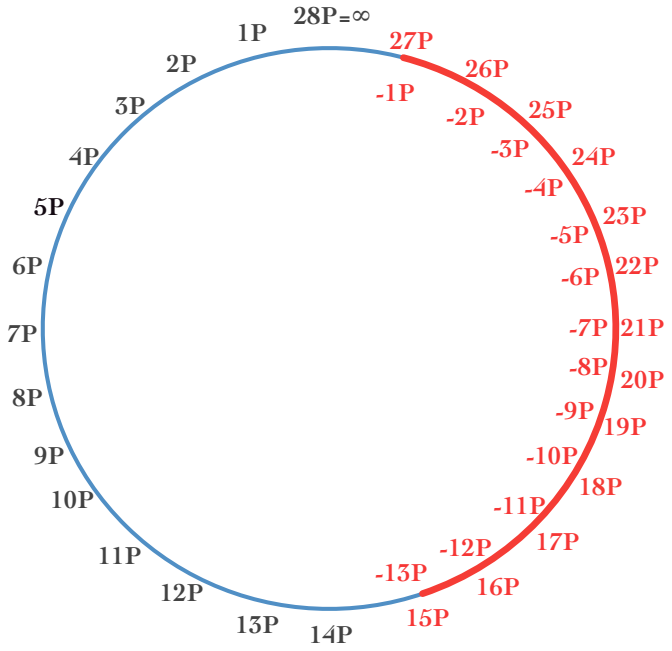
Example: 22P=6P=>2(22-(28/2))=16P

Because 6P+16P=22P

- Determined by the length bit of k
- $\log_2(k)$ if $k=2^x$ , x integer
- Or $\lfloor \log_2(k) \rfloor + 1$

$$log_2(k - 2(k - \frac{n}{2})) = log_2(k) + log_2(k + \frac{n - 2k}{k})$$

$$|log_2(k + \frac{n - 2k}{k})| = |log_2(\frac{|d|}{k})| \qquad log_2(k + \frac{n - 2k}{k}) < 0$$



# Analytical Evaluation (3)

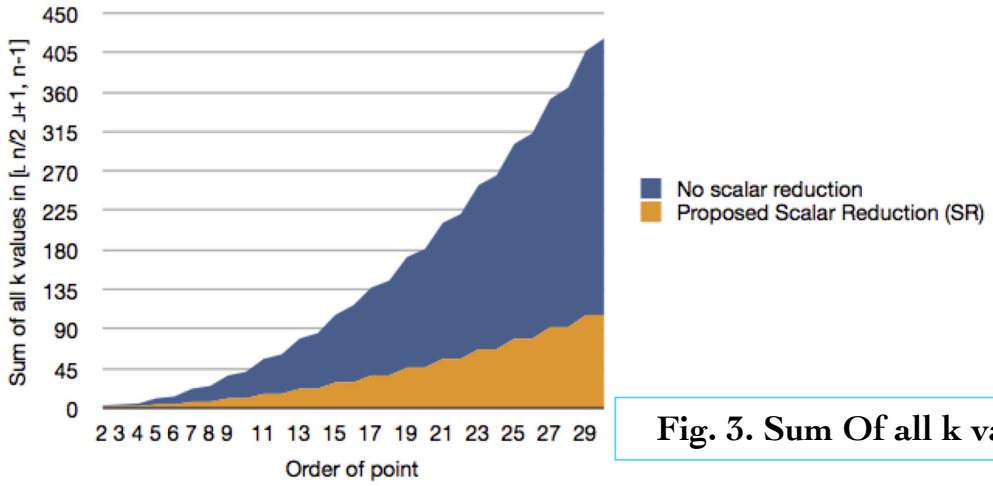



Fig. 3. Sum Of all k values function of order $n$ in [⌊n/2⌋+1, n-1]

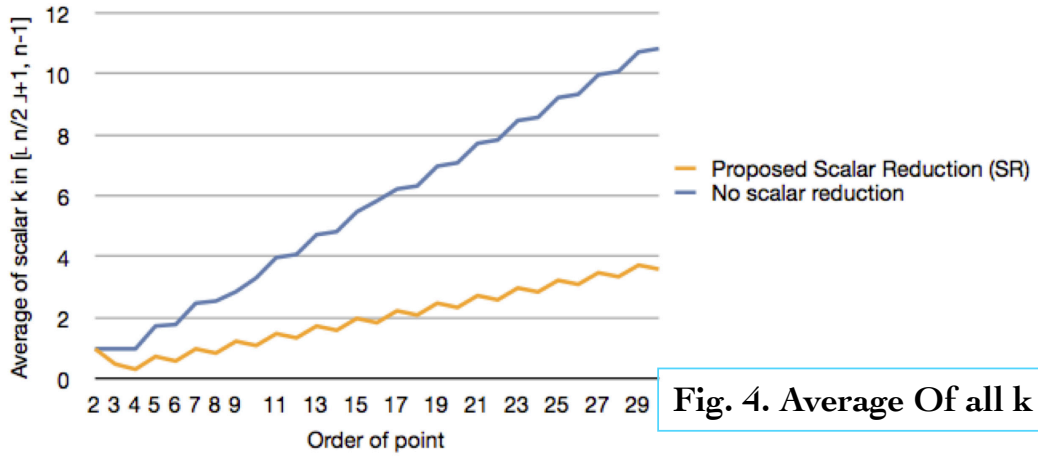



Fig. 4. Average Of all k values function of order $n$ in [⌊n/2⌋+1, n-1]

# Efficiency Analysis (1)

- If $\log_2(n)=x$, with x integer, speed-up in $[\lfloor n/2 \rfloor+1, n-1]$
- If $\log_2(n)=x$, with x not integer, speed-up in $[2^{\lfloor \log_2 \frac{n}{2} \rfloor+1}, n-1]$
- If k=n-1, is the maximum speed-up is $\log_2(n-1)$

Table 1. Speed-up S for some values of k for x integer

| Values of k | $\lfloor n/2 \rfloor + 1$ | $>=(\lfloor n/2 \rfloor + 1)$ | (n-1) |
|---|---|---|---|
| Speed-up(bits) | 1 | $1 < S < log_2(k)$ | $log_2(k)$ |

Table 1. Speed-up S for some values of k for x not integer

| Values of k | $2^{\lfloor \log_2 \frac{n}{2} \rfloor+1}$ | $>= 2^{\lfloor \log_2 \frac{n}{2} \rfloor+1}$ | (n-1) |
|---|---|---|---|
| Speed-up(bits) | 1 | $1 < S < log_2(k)$ | $log_2(k)$ |

# Efficiency Analysis (2) (by used even or odd order)

If the order n >2 is an even number: $\sum_{k=\lfloor n/2 \rfloor+1}^{n-1} kP = 3\sum_{k=1}^{\lfloor n/2 \rfloor-1} kP$

If the order n ≥3 is an odd number: $3\sum_{k=1}^{\lfloor n/2 \rfloor-1} kP > \sum_{k=\lfloor n/2 \rfloor+1}^{n-1} kP \geq 2\sum_{k=1}^{\lfloor n/2 \rfloor-1} kP$

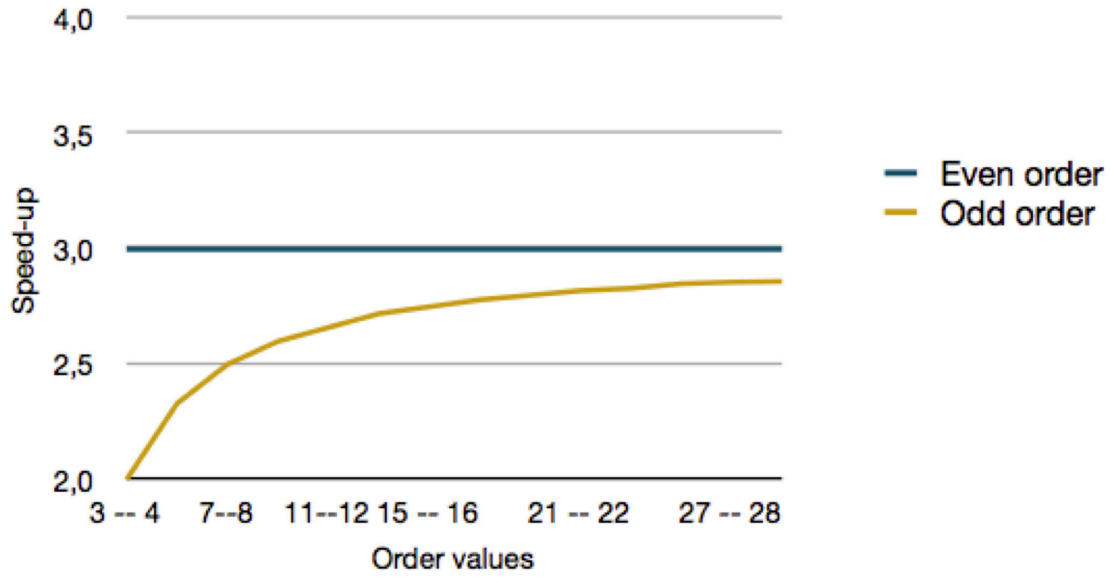



Fig. 5. Speed-up rate between even and odd order in [⌊n/2⌋+1, n-1]

# Performance Evaluation

- Using NIST-192 recommended parameters
- Java simulateur on an Intel Core i5-2520 processor, taking into account the computing power difference between this processor and a MSP 430 MCU

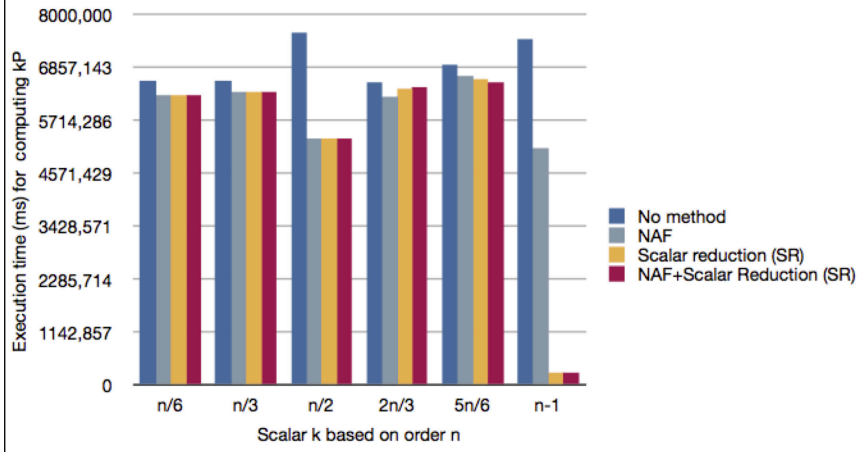



Fig.6. Running times (ms) using affine coordinates

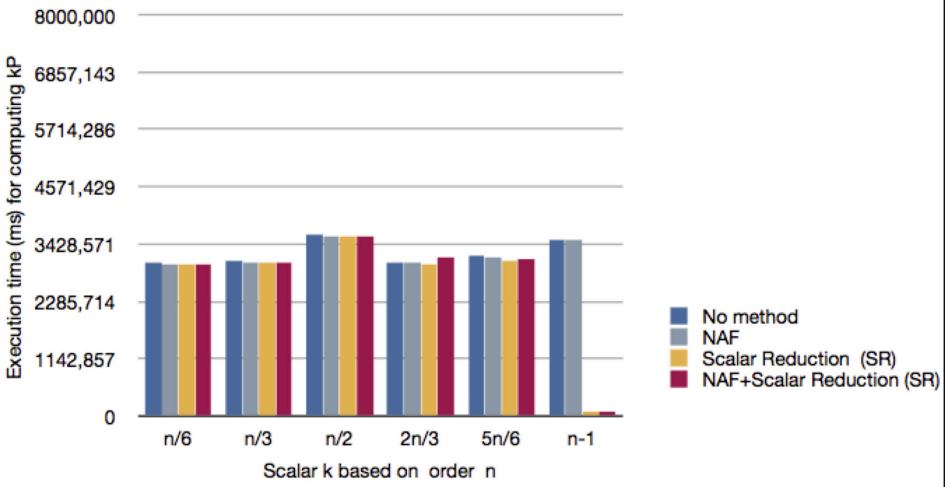



Fig.7. Running times (ms) using Jacobian coordinates

# Conclusion

## Conclusion

- The proposed mechanism significantly the computation time in the interval [⌊ n/2⌋+1, n-1 ].
- We show that the usage of even order is more efficient than odd order.
- It can be easily applied to almost all existing fast scalar multiplication methods

## Perspectives

- Experimenting our current technique on real sensor nodes with elliptic curves over finite prime fields.

# Thank You For Your Attention

Questions?