SPA-resistant Scalar Multiplication on Hyperellipitc Curve Cryptosystems Combining Divisor Decomposition Technique and Joint Regular Form

T. Akishita, M. Katagi, and I. Kitamura

Sony Corporation

In my talk …

- Hyperelliptic Curve Cryptosystems
  - Genus2, F_{2^n}

- New countermeasure against Simple Power Analysis
  - Divisor Decomposition Technique (DDT)
  - Joint Regular Form (JRF)

Agenda

- Introduction (1) : Simple Power Analysis
- Introduction (2) : Theta Divisors on HECC
- Proposed Method
  - Divisor Decomposition Technique (DDT)
  - Joint Regular Form (JRF)
  - Marriage of DDT + JRF
- Concluding Remarks

Simple Power Analysis

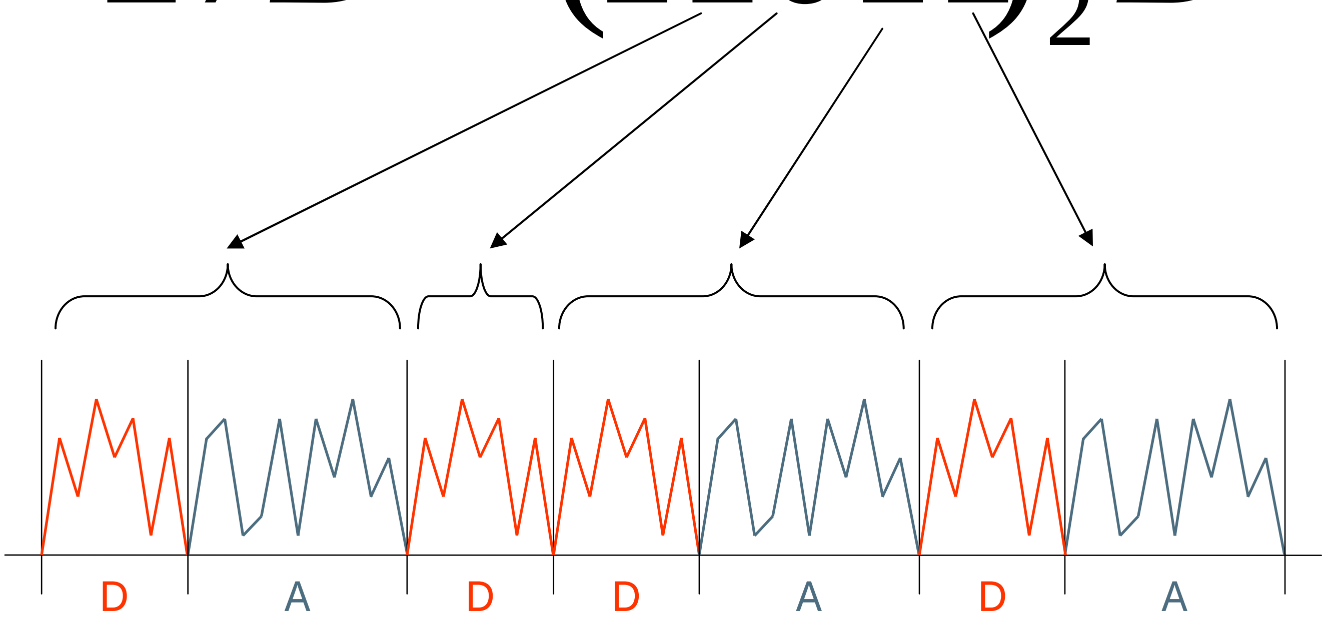
Simple Power Analysis

- Simple Power Analysis (SPA)
  - Single observation of power consumption trace
  - Extract some secret information

- Elliptic curve / Hyperelliptic curve cryptosystems
  - dD : Scalar Multiplication
    - d: Secret information, D: point / divisor

Binary method

$$27D = (11011)_2 D$$



Double-and-add always method

Double-and-add always Method

Input: $D, \quad d = (d_{m-1} \cdots d_0)_2$

Output: dD

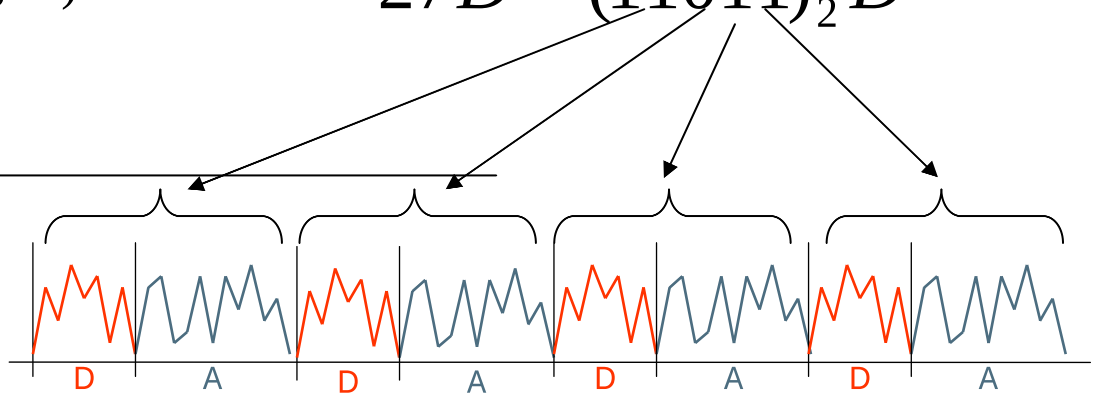
1. $Q[0] \leftarrow D$
2. for $i = m-2$ downto 0

   $Q[0] \leftarrow \textbf{DBL}\,(Q[0])$

   $Q[1] \leftarrow \textbf{ADD}\,(Q[0], D)$

   $Q[0] \leftarrow Q[d_i]$
3. return $Q[0]$

$$27D = (11011)_2 D$$



SPA resistance

Theta Divisors on HECC

Hyperelliptic Curve

- Hyperelliptic Curve

$$y^2 + h(x)y = f(x)$$

$f(x)$ monic polynomial, $\deg f = 2g+1$

$h(x)$ Polynomial, $\deg h \le g$

- genus, g
  - Curves are characterized by genus
    - genus 1

      Elliptic curves
    - genus 2

      $$f(x) = x^5 + f_4x^4 + f_3x^3 + f_2x^2 + f_1x + f_0$$
      $$h(x) = h_2x^2 + h_1x + h_0$$
    - genus 3 ….

Divisor

- Divisor
  - Points on hyperelliptic curve do not form a group

- Representation of divisor

$$D = (u(x), v(x)) \in J(F_{2^m}) \Leftrightarrow u(x), v(x) \in F_{2^m}[x].$$

  - Genus 2
    - $u(x) = x^2 + u_1 x + u_0$, $v(x) = v_1 x + v_0$

- Weight w(D)
  - Degree of polynomial u(x)
  - Weight 2 divisor

General Divisor

Special Divisor: theta divisor

- the weight of D is smaller than genus
- "Special" means low probability
- Genus 2 case

general divisor

$D=(x^2+u_1x+u_0,\ v_1x+v_0)$ // w(D)=2

theta divisor

$D=(x+x_0, y_0)$ // w(D)= 1

Group op. with theta divisor is fast!

Group operations	Cost	
DBL	1I + 22M + 5S	“general”
ADD	1I + 22M + 3S	“general”+”general”
TDBL	1I + 5M + 2S	“theta”
TADD	1I + 10M + 1S	“genral” + “theta”

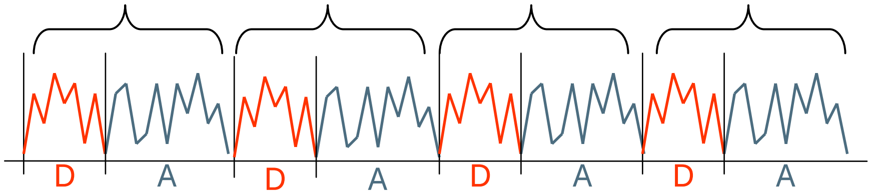
I:inversion, M:multiplication, S:Squaring

DAA_TD : speed up the DAA_GD

- DAA_GD
  - Double-and-add-always method using General Divisors
- DAA_TD
  - Double-and-add-always method using Theta Divisors

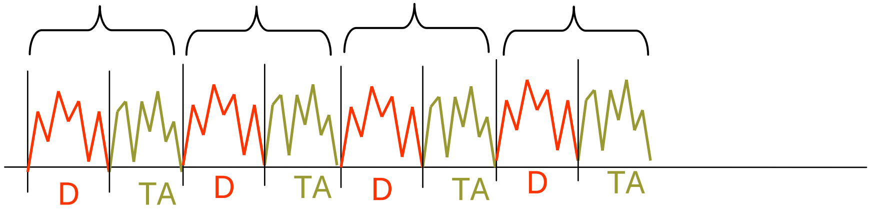
$$27D = (11011)D$$

general divisor



$$27D_0 = (11011)D_0$$

theta divisor



Improvement is 20%! [9]

DAA_TD : Motivation

- DAA_TD
  - is much faster than DAA_GD.
  - But, application is limited
    - the base point (fixed point)
    - theta divisor is chosen

*How to apply **group operations with theta divisor** to speed up scalar multiplication using **a general divisor** ?*

Our Idea

$$kD = kD_1 + kD_2$$

Decomposing into Two Theta Divisors

(1)Divisor Decomposition Technique (DDT)

$$= \underbrace{kD_1 + (k+1)D_2}_{} - D_2$$

Simultaneous scalar multiplication with JRF

(2)Joint Regular Form (JRF)

Divisor Decomposition Technique (DDT)

Divisor Decomposition Technique

- DDT
  - A general divisor D → theta divisors $D_1 + D_2$
    - $D=(u(x),v(x))=(x^2+u_1x+u_0, v_1x+v_0)$
      - $u_i,v_i \in \mathbf{F}_2{}^m$
    - $D_1=(x+x_1,y_1), D_2=(x+x_2,y_2)$
      - $u_i,v_i \in \mathbf{F}_2{}^m$

DDT condition

General divisor

$$D=(\underline{x^2+u_1x+u_0}, v_1x+v_0)$$

$\mathrm{Tr}(u_0/u_1^2)$

Check the reducibility of $x^2+u_1x+u_0$ over $\mathbf{F}_{2^n}$

0 → D1+D2

1 → Fail to decompose

DDT is efficient ?

- $kD : kD_1+kD_2$
- Direct computation of kD_1+kD_2
  - Slower than kD
    - 2 times TADD < ADD
- Any other ideas?

$$kD_1+kD_2 = kD_1 + (k+1)D_2 - D_2$$

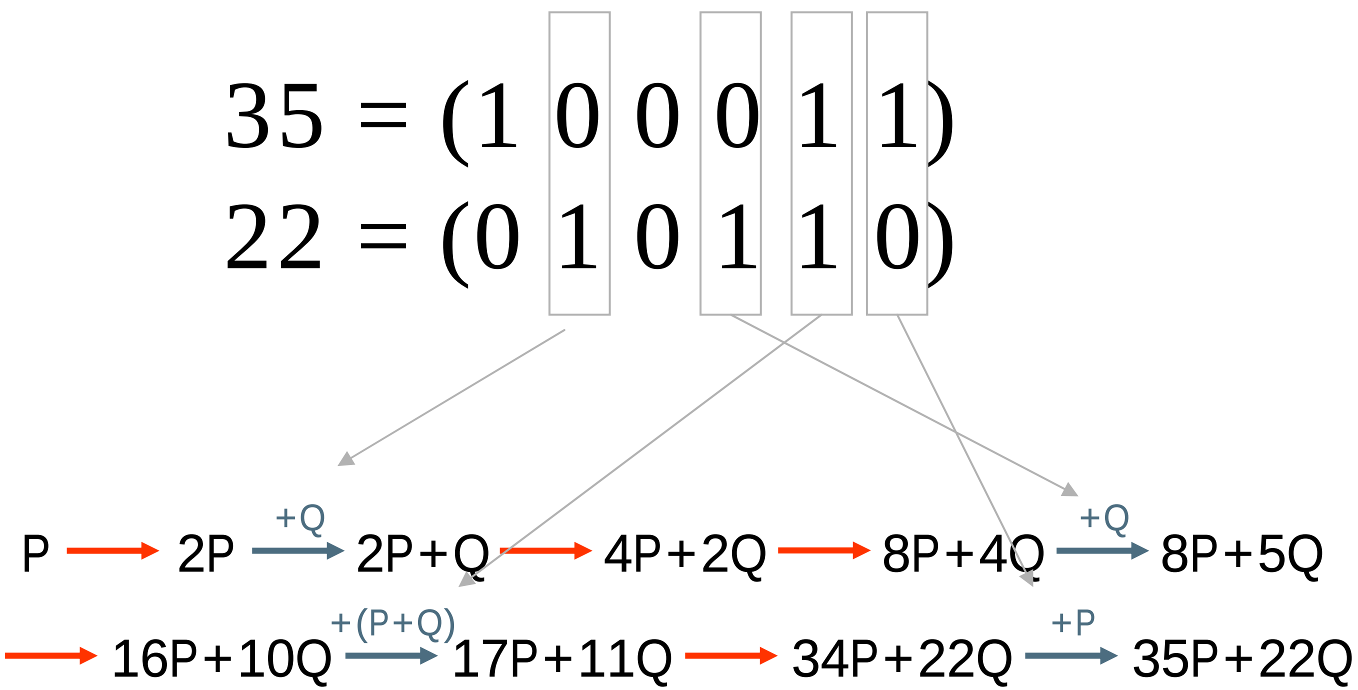
Simultaneous multiplication of $kD_1+(k+1)D_2$

We need good representation of (k,k+1)

Joint Regular Form (JRF)

Simultaneous Scalar Multiplication

- Simultaneous scalar multiplication of kP+lQ
- Shamir's trick
- Ex. 35P+22Q = $(100011)_2$P+$(010110)_2$Q



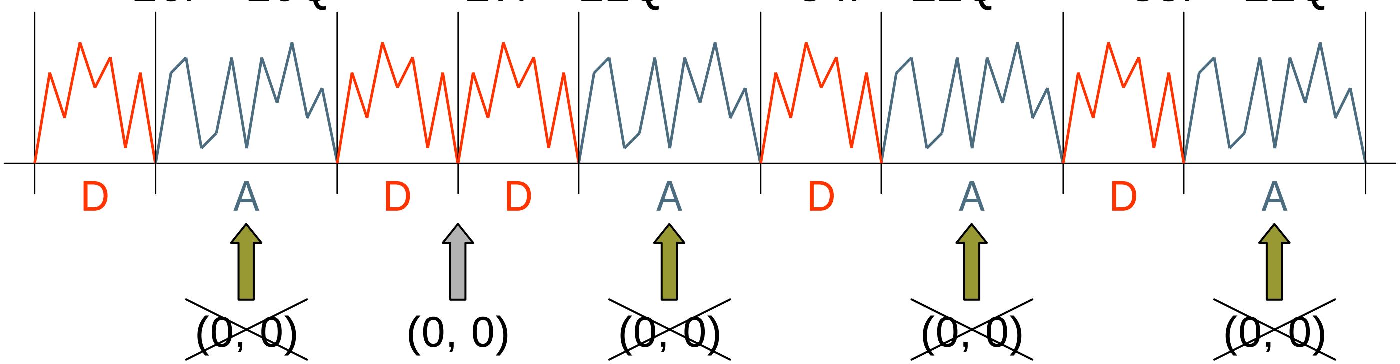
Power Analysis to Simultaneous Scalar Multiplication

- Shamir's method

  Ex. $35P+22Q = (100011)_2P+(010110)_2Q$

  P → 2P →(+Q) 2P+Q → 4P+2Q → 8P+4Q →(+Q) 8P+5Q

  → 16P+10Q →(+(P+Q)) 17P+11Q → 34P+22Q →(+P) 35P+22Q



Vulnerable to SPA

- Inserting dummy operation can prevent SPA in exchange for efficiency

Joint Regular Form (JRF)

- (k, l) is (even, odd) or (odd, even)

- Joint Regular Form (JRF) of (k, l):
  - $\langle k_{n-1} \ldots k_0 \rangle$, $\langle l_{n-1} \ldots l_0 \rangle$

1. $k_i + l_i = \pm 1$, i.e. $(k_i, l_i) = (0, \pm 1)$ or $(\pm 1, 0)$

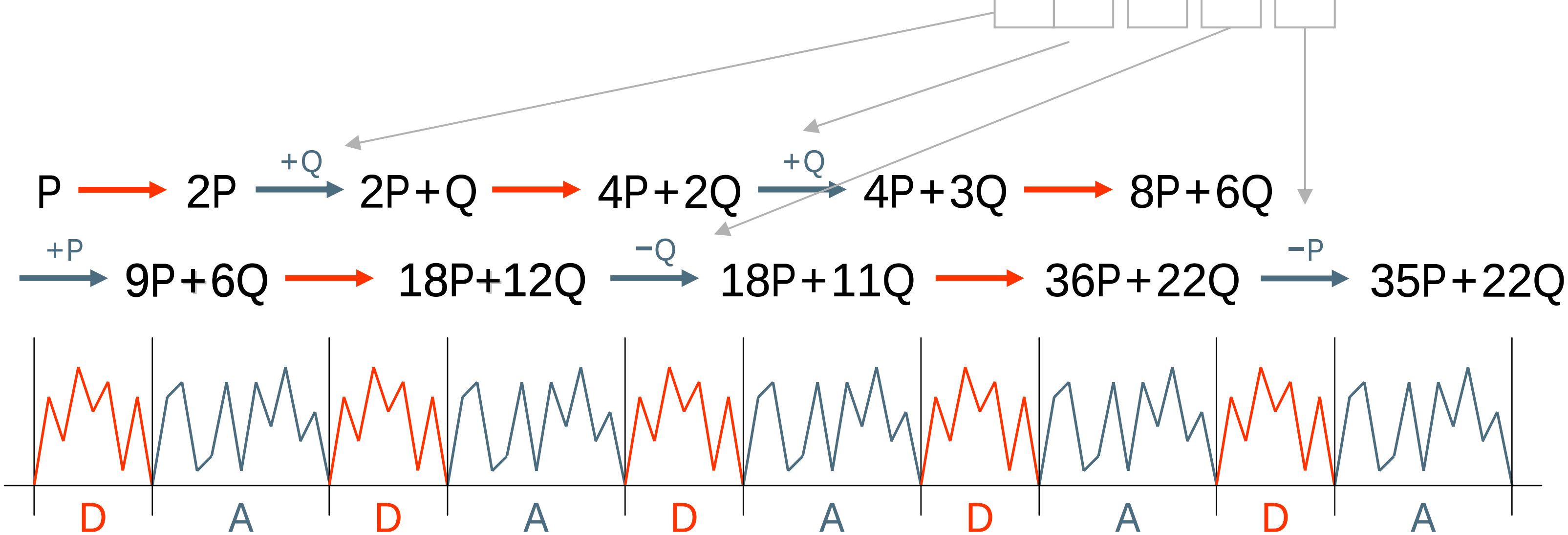
- Always repeat doubling and addition when computing kP+lQ
  - SPA–resistance without dummy operation because of regularity

Simultaneous Scalar Multiplication with JRF

Ex. 35P+22Q $\bar{1} = -1$

JRF of (35, 22):

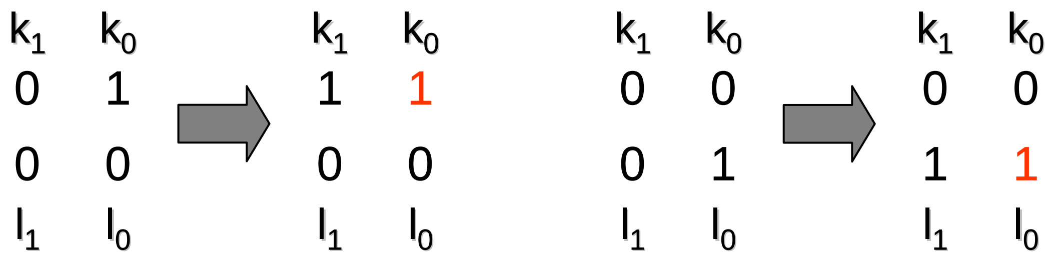
$$35 = \langle 1\ 0\ 0\ 1\ 0\ \bar{1} \rangle$$
$$22 = \langle 0\ 1\ 1\ 0\ \bar{1}\ 0 \rangle$$



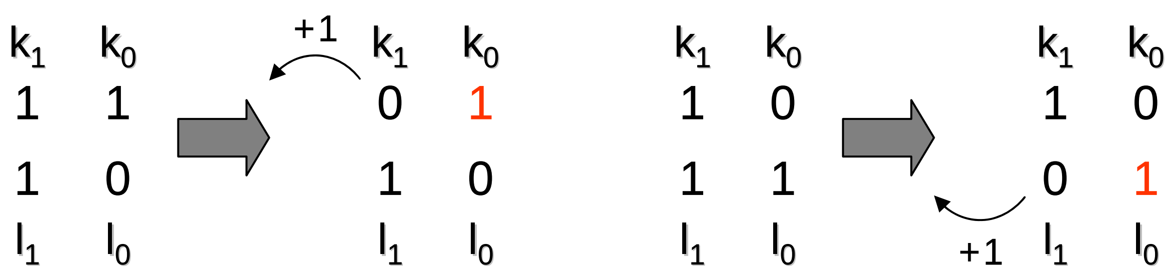
Need not P+Q !

How to Construct JRF: General Case

- Transform binary representation $k = (k_{n-1} \ldots k_0)_2$, $l = (l_{n-1} \ldots l_0)_2$ to JRF $\langle k_{n-1} \ldots k_0 \rangle$, $\langle l_{n-1} \ldots l_0 \rangle$ from LSB
  - $(k_0, l_0) = (0, 1)$ or $(1,0)$
  - If $(k_1, l_1) = (0, 1)$ or $(1,0)$, no transformation is needed
  - If $(k_1, l_1) = (0, 0)$, one of following transformation is done



  - If $(k_1, l_1) = (1,1)$, one of following transformation is done and carry over +1 to k_2 or l_2



How to construct JRF : (d, d+1)

- $dD = dD_1 + (d+1)D_2 - D_2$

- JRF

$27 = (1\ 1\ 0\ 1\ 1)_2$

$28 = (1\ 1\ 1\ 0\ 0)_2$

→

$27 = <\ \ 1\ 1\ 0\ 1\ 1 >_2$

$28 = < 1\ 0\ 0\ \underline{1}\ 0\ 0 >$

- $d+1 = 2^n + \Sigma_i^{n-1}(d_i - 1)2^i$

$27 = \ \ 1\ 1\ 0\ 1\ 1$

↓ Flip 0/1

$0\ 0\ 1\ 0\ 0$

↓ Change sign

$0\ 0\ \underline{1}\ 0\ 0$

↓ Append “1” in MSB

$28 = 1\ 0\ 0\ \underline{1}\ 0\ 0$

Marriage of DDT and JRF

Proposed Method: DDT + SimJRF

(1) Decomposing into Two Theta Divisors

$$kD = kD_1 + kD_2$$

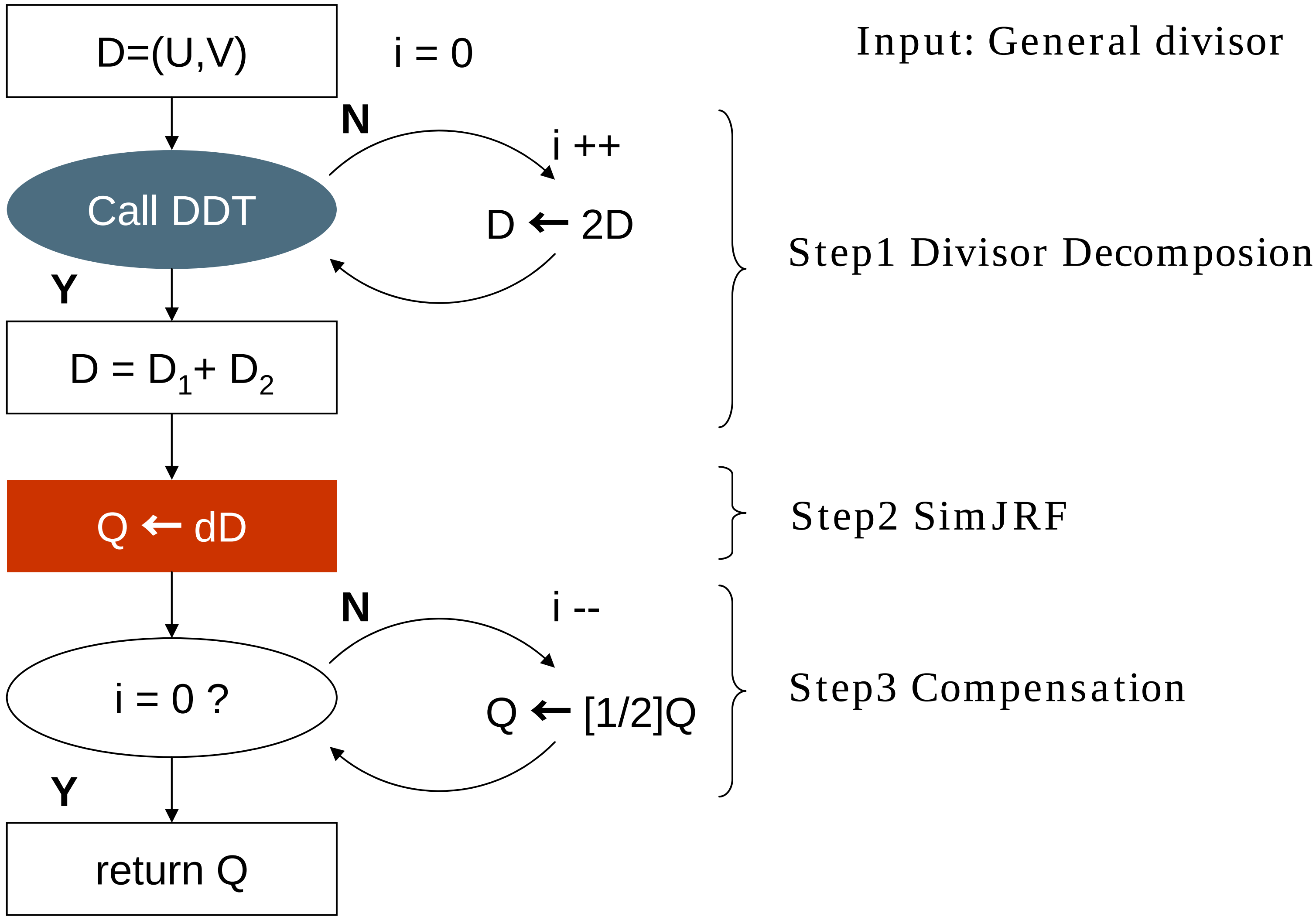
$$= \underbrace{kD_1 + (k+1)D_2}_{} - D_2$$

(2)Simultaneous scalar multiplication with JRF

Any Genral Divisor cannot be decomposed …

- General Divisor D=(u(x),v(x))
  - DDT condition
    - u(x) is reducible over F_{2^m}
    - u(x) is irreducible over F_{2^m} Not decomposed
- In order to apply DDT to Any general divisors,
  - Use inverse map of divisor
    - $dD = d((1/2)^i 2^i D) = (1/2)^i d(2^i D)$
    - Repeat "doubling" until DDT returns success!
    - Correct the value using "halving"[10]

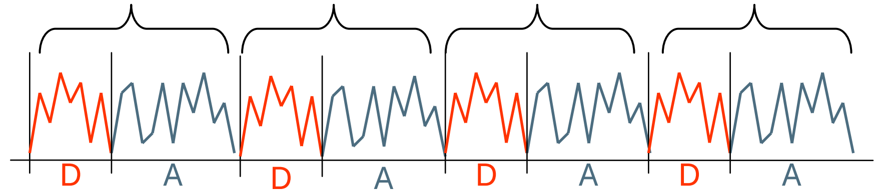
Complete Procedure of the proposed method



DDT+SimJRF

DAA_GD

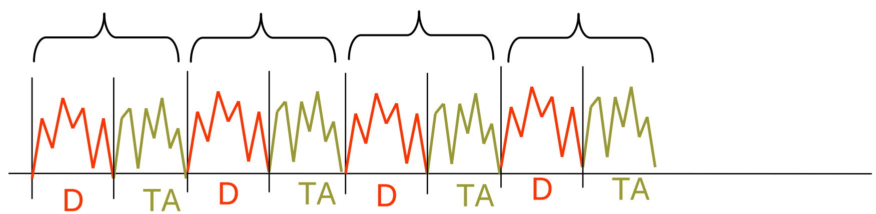
$27D = (11011)D$



DDT+SimJRF

Always add +/- D_1 or +/- D_2

$27D = \langle 11011 \rangle D_1$
$+ \langle 100\underline{1}00 \rangle D_2$



Comparison of scalar multiplication

- DAA_GD, DAA_TD, DDT+SimJRF

Table 2. Comparison of scalar multiplication (160bit)

Method	Divisor	Dummy	Cost
DAA_GD	general	use	$318I + 6996M + 1272S \quad (9667.2M)$
DAA_TD	theta	use	$318I + 5084M + 951S \quad (7723.1M)$
DDT+SimJRF	general	NOT use	$325I + 5160.5M + 967S$ $+2.5SR + 3H + 4T \quad (7860.3M)$

DDT+SimJRF is 18.7% faster than DAA_GD

1.8% increase compared to DAA_TD in spite of extra cost (DDT)

Concluding Remark

- DDT+JRF
  - Genus 2 HECC over binary field
  - Efficient SPA-resistant scalar multiplication
    - DDT
    - JRF
  - 18.7% faster than DAA_GD

- JRF
  - New Signed Representation for Two integers
  - Application to HECC (this talk)
  - Have nice applications for ECC
    - Lim-Lee method, GLV method, BRIP, ….