Signed Binary Representations Revisited

Katsuyuki Okeya, Hitachi
Katja Schmidt-Samoa, Christian Spahn,
Tsuyoshi Takagi, TU Darmstadt

http://www.informatik.tu-darmstadt.de/KP/

Content

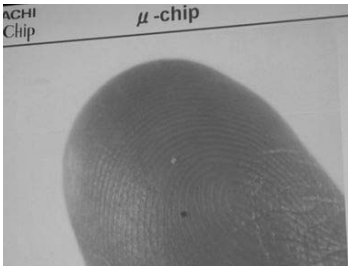
- Motivation
- Non-Adjacent Form
- Proposed Scheme – MOF
- Application to ECC
- Conclusion

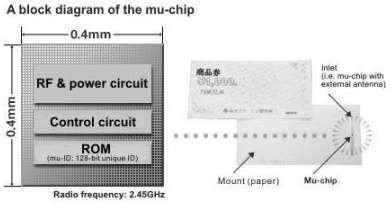
Efficiency is Important

Smart Cards
- tamper-resistant
- mobility

⇒ We need efficient cryptographic algorithms

μ-chip (Hitachi)





Contact-less chip card, 0.4x0.4 mm^2, Radio Frequency 2.45 GHz, 128-bit ROM.

⇒ RFID (Radio Frequency Identification)
Ubiquitous Computing, Ad hoc Network

Elliptic Curve Cryptosystem

RSA (1024 bits)

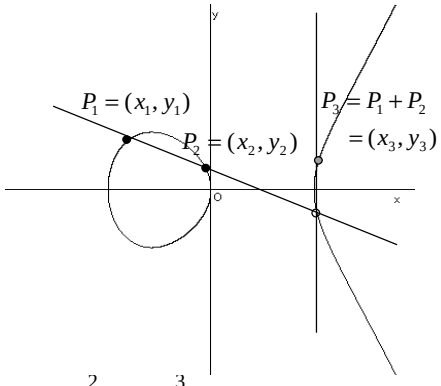
e = 10001
n = 826ed558a0f0cba7ae09485abf80c544837efeb7116153f5d6479d5945fdb6c61f50c984445d601d85eceb6b
ad9f700b90ae28984dd590f5ca3e6ed968a3ca32a5cf584992d92590ae9ed4f81b70d008a9e4a16905925dbb
79d82b67dc6b70869a83f037c147d298c0e2eea5f858f3881ad1071c5c221ecb795d78b68bae7863
d = 21b67db4237d72766beea667b95143c0a22f4f07b4f25d1b75e400397b45b7c45e108addc4f03a9000d0fb5
c76da4480fef42651830090682b1a0bfadeb92dee047626b1417651aa832469b59792e2fc8688d187201d6d
0c7de9301144e003473ecbf859ababa15311adea452d160f11b5b5fe2338b00e57728b4b691f43fc1

ECC (160 bits)

p = ffffffff ffffffff ffffffff ffffffff 7fffffff
a = ffffffff ffffffff ffffffff ffffffff 7ffffffC
b = 1c97befc 54bd7a8b 65acf89f 81d4d4ad c565fa45
x = 4a96b568 8ef57328 46646989 68c38bb9 13cbfc82
y = 23a62855 3168947d 59dcc912 04235137 7ac5fb32
n = 01 00000000 00000000 0001f4c8 f927aed3 ca752257
h = 01
s = 203370bf 41c7ca08 22e2ccd8 f4d4a011 91977373

Standard Addition Formula

$E / GF(p): y^2 = x^3 + ax + b$ (Weierstraß-form of an elliptic curve)



$y^2 = x^3 - x$

[Standard Formula]

$$x_3 = \lambda^2 - x_1 - x_2$$
$$y_3 = \lambda(x_1 - x_3) - y_1$$
$$\lambda = \begin{cases} \dfrac{y_1 - y_2}{x_1 - x_2} & \text{ECADD} \\ \dfrac{3x_1^2 + a}{2y_1} & \text{ECDBL} \end{cases}$$

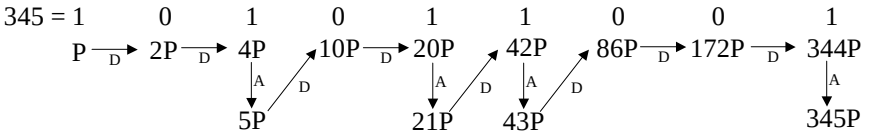
Scalar Multiplication

$$d \times P = \underbrace{P + P + \ldots + P}_{d \text{ times}}, \quad d = d[n-1]2^{n-1}+d[n-2]2^{n-2}+\ldots+d[1]2^{1}+d[0]2^{0}.$$

Square & Multiply Method

```
Q = P
for i=n-2 down to 0
  Q = ECDBL(Q)
  if d[i] = 1, then Q = ECADD(Q,P)
return(Q)
```

Example d = 345



Non-Adjacent Form (NAF)

NAF is $d = d[n-1]2^{n-1}+d[n-2]2^{n-2}+\ldots+d[1]2^{1}+d[0]2^{0}$, where $d[i]$ in $\{-1,0,1\}$ and $d[i]*d[i-1]=0$ for $i=1,2,..,n-1$.

345 =	1	0	1	0	1	1	0	0	1	(Binary representation)
	1	0	-1	0	-1	0	-1	0	0	1 (NAF representation)

Scalar Multiplication using NAF

```
Q = P
for i=n-2 down to 0
  Q = ECDBL(Q)
  if d[i] = 1,  then Q = ECADD(Q,P)
  if d[i] = -1, then Q = ECADD(Q,-P)
return(Q[0])
```

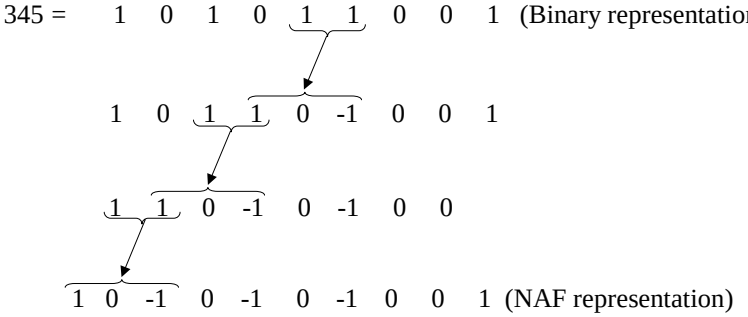
-P = (x,-y) for P = (x,y), virtually for free

The average density of non-zero bits of NAF is asymptotically 1/3.

⇒ NAF can achieve faster scalar multiplication.

How to generate NAF?

The crucial conversion is 10-1 ← 11, where 1 is a carry.



NAF can not be generated in left-to-right due to carry 10-1 ← 11.

Left-to-right is more efficient

Left-to-right method

```
Q[0] = P
for i=n-2 down to 0
  Q[0] = ECDBL(Q[0])
  Q[0] = ECADD(Q[0],d[i]P)
return(Q[0])
```

P is represented by P = (X:Y:1), which is fixed during the scalar multiplication.

ECADD with Z=1 requires only 11 multiplications

Right-to-left method

```
Q[0] = O, Q[1] = P
for i=0 down to n-1
  Q[0] = ECADD(Q[0],d[i]Q[1])
  Q[1] = ECDBL(Q[1])
return(Q[0])
```

Q[1] is NOT represented by Q[1] = (X:Y:1).

ECADD with Z≠1 requires 16 multiplications

⇨ Is there any efficient left-to-right exponent recording?

Related Works

- Joye, Yen: IEEE Trans., 2000.
- Joye, Tymen: PKC 2001.

Very recently,

- Muir, Stinson: TR of CACR, 2004.
- Avanzi: SAC 2004.
- Heuberger, Katti, Prodinger, Ruan: Preprint.

Mutual Opposite Form (MOF)

MOF is $d = d[n-1]2^{n-1}+d[n-2]2^{n-2}+\ldots+d[1]2^{1}+d[0]2^{0}$, *where*
(1) $d[i]$ *in* $\{-1,0,1\}$
(2) The signs of adjacent non-zero bits (ignoring 0 bits) are opposite
(3) The most and least non-zero bits are 1 and -1, respectively.

345 =	1	0	1	0	1	1	0	0	1	(Binary representation)
	1	-1	1	-1	1	0	-1	0	1	-1 (MOF representation)

We can prove the following facts:

- Every n-bit integer is uniquely represented by (n+1)-bit MOF.
- The average density of non-zero bits is asymptotically 1/2.

How to generate MOF?

We can prove that bit-wise subtraction *2d – d* yields the MOF of d.

2d	=	1	0	1	0	1	1	0	0	1
- d	=		1	0	1	0	1	1	0	0	1
		1	-1	1	-1	1	0	-1	0	1	-1

⇒ This conversion algorithm has no carry.

Interestingly,
the MOF representation of integer d is essentially equal to classical Booth encoding of multiplier A and multiplicand B:
(1) No operation, if $(a_i,a_{i-1}) = (0,0)$ or (1,1)
(2) Subtract multiplicand B from the partial product, if $(a_i,a_{i-1}) = (1,0)$
(3) Add multiplicand B to the partial product, if $(a_i,a_{i-1}) = (0,1)$

MOF & NAF

Surprisingly, we can prove that
if we apply right-to-left sliding window (without carry) conversions
01 ← 1-1 and 0-1 ← -11 to MOF of d, then the NAF of d is obtained.

345 =	1	0	1	0	1	1	0	0	1	(Binary representation)
	1	-1	1	-1	1	0	-1	0	1	-1	(MOF representation)
	1	-1	1	-1	1	0	-1	0	0	1	
	1	-1	1	0	-1	0	-1	0	0	1	
	1	0	-1	0	-1	0	-1	0	0	1	(NAF representation)

⇒ We can generate NAF from MOF without carry.

Left-to-right conversion to MOF

How about applying the sliding window conversions 1-1 → 01, -11 → 0-1?

345 = 1 0 1 0 1 1 0 0 1 (Binary representation)

1 -1 1 -1 1 0 -1 0 1 -1 (MOF representation)

0 1 1 -1 1 0 -1 0 1 -1

0 1 0 1 1 0 -1 0 1 -1

0 1 0 1 1 0 -1 0 0 1 (New Chain: 2MOF)

Happily, we can prove that
the average non-zero density of this new chain is asymptotically 1/3.

Application to ECC

Algorithm 7 Left-to-Right Scalar Multiplication Algorithm (On the Fly), $w = 2$

Input: a point P, a non-zero n-bit binary string $d = d_{n-1}|d_{n-2}|...|d_1|d_0$
Output: product dP

$d_{-1} \leftarrow 0$; $d_n = 0$
$i \leftarrow c + 1$ for the largest c with $d_c \neq 0$
if $d_{i-2} = 0$ **then**
  $Q \leftarrow P$; $i \leftarrow i - 2$
else $\{d_{i-2} = 1\}$
  $Q \leftarrow$ ECDBL(P); $i \leftarrow i - 2$
while $i \geq 1$ **do**
  if $d_{i-1} = d_i$ **then**
    $Q \leftarrow$ ECDBL(Q); $i \leftarrow i - 1$
  else $\{d_{i-1} \neq d_i\}$
    $Q \leftarrow$ ECDBL(Q)
    if $(d_i, d_{i-2}) = (1, 1)$ **then**
      $Q \leftarrow$ ECDBL(Q); $Q \leftarrow$ ECADD$(Q, -P)$
    else if $(d_i, d_{i-2}) = (1, 0)$ **then**
      $Q \leftarrow$ ECADD$(Q, -P)$; $Q \leftarrow$ ECDBL(Q)
    else if $(d_i, d_{i-2}) = (0, 1)$ **then**
      $Q \leftarrow$ ECADD(Q, P); $Q \leftarrow$ ECDBL(Q)
    else if $(d_i, d_{i-2}) = (0, 0)$ **then**
      $Q \leftarrow$ ECDBL(Q); $Q \leftarrow$ ECADD(Q, P)
    $i \leftarrow i - 2$
if $i = 0$ **then**
  $Q \leftarrow$ ECDBL(Q); $Q \leftarrow$ ECADD$(Q, -d_0 P)$
return Q.

wNAF

wNAF is $d = d[n-1]2^{n-1}+d[n-2]2^{n-2}+\ldots+d[1]2^{1}+d[0]2^{0}$, *where*
(1) $d[i]$ *in* $\{0, \pm1, \pm3, \ldots, \pm(2^{w-1}-1)\}$
(2) Among any w consecutive digits, at most one is non-zero.
(3) The most significant non-zero bit is positive.

345 =	1	0	1	0	1	1	0	0	1	(Binary representation)
	1	0	0	-3	0	0	3	0	0	1 (3NAF representation)

We can prove the following facts:

- Every integer is uniquely represented by wNAF.
- The average non-zero density is asymptotically 1/(w+1).

How to generate wNAF

We can extend NAF to its width-w version, called wNAF.
For example, crucial conversion for w=3 is
100-1 ← 111, 100-3 ← 101, and 003 ← 011

345 =	1	0	1	0	1	1	0	0	1	(Binary representation)
	1	0	1	0	1	1	0	0	1	
	1	0	1	0	0	3	0	0		
	1	0	0	-3	0	0	3	0	0	1 (3NAF representation)

wNAF can not be generated in left-to-right due to carry.

MOF & wNAF

If we apply width-w sliding window (without carry) conversion to MOF of d, then the wNAF of d is obtained.

For example, for w=3, 001 ← 01-1 , 00-1 ← 0-11,
003 ← 1-11 or 10-1, and 00-3 ← -11-1 or -101.

345 = 1 0 1 0 1 1 0 0 1 (Binary representation)
1 -1 1 -1 1 0 -1 0 1 -1 (MOF representation)

1 -1 1 -1 1 0 -1 0 0 1

1 -1 1 -1 0 0 3 0 0 1

1 0 0 -3 0 0 3 0 0 1 (3NAF representation)

We can generate NAF from MOF without carry.

wMOF

Similarity, we can construct width-w version of MOF, called wMOF.
For example, crucial conversion for w=3 is
1-11,10-1 → 003, -11-1,-101 → 00-3, 1-10 → 010, and -110 → 0-10.

345 = 1 0 1 0 1 1 0 0 1 (Binary representation)
1 -1 1 -1 1 0 -1 0 1 -1 (MOF representation)

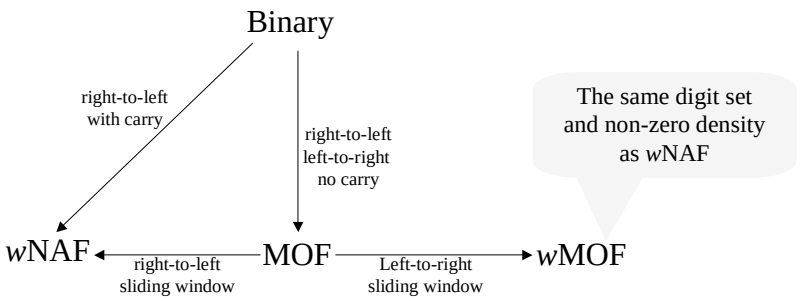
0 0 3 -1 1 0 -1 0 1 -1

0 1 0 0 -1 0 -1 0 1 -1

0 1 0 0 -1 0 0 0 -3 -1 (3MOF representation)

We can prove that
the average density of non-zero digits are asymptotically 1/(w+1).

Conclusion



Thank you!

The full version is available from IACR ePrint.
http://eprint.iacr.org/2004/195/

The home of MOF is the following URL.
http://www.informatik.tu-darmstadt.de/KP/MOF/