

# Fast Point Decompression for Standard Elliptic Curves

Billy Brumley & Kimmo Järvinen

Department of Information and Computer Science  
Helsinki University of Technology

June 16–17, 2008



HELSINKI UNIVERSITY OF TECHNOLOGY  
Department of Information and Computer Science

# Outline

Elliptic Curves

Addition Chains

Fast Point Decompression

Implementations



## Definition (Elliptic Curve)

With  $p > 3$ , all the  $(x, y)$  solutions to the curve

$$E(\mathbb{F}_p) : y^2 = x^3 + ax + b$$

along with the identity  $\mathcal{O}$  form an abelian group.

This group is written additively. Diffie-Hellman in  $(G, +)$ :

- $A \rightarrow B : aG$
- $B \rightarrow A : bG$
- Shared key:  $a(bG) = b(aG) = abG$



## Point Compression (Miller 86)

Given  $P = (x, y) \in E$  (maybe a public key), it suffices to send only  $x$ ; the recipient recovers  $y = \sqrt{x^3 + ax + b}$ . A sign bit can be used to distinguish between the two  $y$  solutions (if needed). This cuts the storage requirement in half, but requires a square root in  $\mathbb{F}_p$ .

By the group law  $-P=(x,-y)$  so  $[kP]_x = [k(-P)]_x$  and sometimes we can drop the sign bit.

## Cost?

Point decompression is done online, the speed is important. How fast can we make the square root operation?



## Definition (Euler's Criterion)

For  $\beta \in \mathbb{QR}_p$ ,  $\beta^{(p-1)/2} \equiv 1 \pmod{p}$ .

## Square Roots Modulo $p \equiv 3 \pmod{4}$

To solve  $\alpha^2 = \beta \pmod{p}$  for  $\alpha$ , we see

$$\alpha^2 \equiv \beta \beta^{(p-1)/2} \equiv \beta^{(p+1)/2} \pmod{p} \text{ and}$$

$$\alpha \equiv \pm \beta^{(p+1)/4} \pmod{p}.$$

## Cost?

We need this exponentiation to be fast!



## Curves in Practice

OpenSSL has 27 named curves over  $\mathbb{F}_p$ ; 24 of them satisfy  $p \equiv 3 \pmod{4}$ . Mersenne-like  $p$  is often chosen to make reduction faster. Examples from various standards:

- WTLS7:  

$$p = 2^{160} - 2^{32} - 2^{14} - 2^{12} - 2^9 - 2^8 - 2^7 - 2^3 - 2^2 - 1$$
- WTLS9:  $p = 2^{160} - 229233$
- secp192k1:  $p = 2^{192} - 2^{32} - 2^{12} - 2^8 - 2^7 - 2^6 - 2^3 - 1$
- P-192:  $p = 2^{192} - 2^{64} - 1$
- P-384:  $p = 2^{384} - 2^{128} - 2^{96} + 2^{32} - 1$
- ...and so on.



## Definition (Addition Chain)

*An addition chain for an integer  $a_l$  is a sequence  $\gamma$  of  $l$  pairs  $((j(1), k(1)), \dots, (j(l), k(l)))$  of integers with  $0 \leq k(i) \leq j(i) < i$  for all  $1 \leq i \leq l$ , and the semantics  $\mathcal{S}(\gamma) = \{a_0, \dots, a_l\}$  is the set of integers such that  $a_0 = 1$  and  $a_i = a_{j(i)} + a_{k(i)}$  for  $1 \leq i \leq l$ . An addition chain can be viewed as a directed graph consisting of nodes  $\mathcal{S}(\gamma)$  and edges from  $a_{j(i)}$  and  $a_{k(i)}$  to  $a_i$ .*

## Definition (Addition Sequence)

*An addition sequence for a set of integers  $\mathcal{W}$  is an addition chain  $\gamma$  such that  $\mathcal{W} \subset \mathcal{S}(\gamma)$ .*

## Optimal Chains

Finding chains with minimal length ( $l$ ) is NP-hard!

Binary Chain for 15 ( $l = 6$ ):  $a^{15} = ((a^2a)^2a)^2a$

$$\gamma = ((0, 0), (1, 0), (2, 2), (3, 0), (4, 4), (5, 0))$$

$$\mathcal{S}(\gamma) = \{1, 2, 3, 6, 7, 14, 15\}$$

$$1 \Rightarrow 2 \Rightarrow 3 \Rightarrow 6 \Rightarrow 7 \Rightarrow 14 \Rightarrow 15$$

Optimal Chain for 15 ( $l = 5$ ):  $a^{15} = ((a^2a)^2)^2a^2a$

$$\gamma = ((0, 0), (1, 0), (2, 2), (3, 3), (4, 2))$$

$$\mathcal{S}(\gamma) = \{1, 2, 3, 6, 12, 15\}$$

$$1 \Rightarrow 2 \Rightarrow 3 \Rightarrow 6 \Rightarrow 12 \Rightarrow 15$$

## Definition (Repunit)

A base-2 repunit is a number with all 1's in its binary expansion, e.g. of the form  $2^n - 1$ .

## Addition Chains for Repunits (Brauer 39)

Denoting  $w_j = 2^j - 1$  it follows

$$w_{a+b} = \sum_{0 \leq i < (a+b)} 2^i = \left( \sum_{0 \leq i < a} 2^i \right) 2^b + \sum_{0 \leq i < b} 2^i = w_a 2^b + w_b$$

This problem is much easier: find an addition chain for  $n$

An addition chain for  $w_{a+b}$  can be constructed using  $b$  squarings and two smaller addition chains for  $w_a$  and  $w_b$ .

## Addition chain for $w_{15} = 2^{15} - 1$

$$w_1 \xrightarrow{\cdot 2} w_2 \xrightarrow{\cdot 2} w_3 \xrightarrow{\cdot 2^3} w_6 \xrightarrow{\cdot 2} w_7 \xrightarrow{\cdot 2^7} w_{14} \xrightarrow{\cdot 2} w_{15}$$



## Do Offline

- $e = (p + 1)/4$  consists of  $k$  repunit windows (split on zeros).
- Search for the shortest addition sequence containing the length of all these windows.

## Do Online

- Precompute all repunit windows using one iteration of the previous equation.
- Put everything together using a sliding window approach:
  - Start squaring from the LSB of the first window (the one in the MSB) to the right.
  - At the end of each window, multiply by one of the precomputed values.

## Computational Cost

$\lfloor \log_2 e \rfloor - n_s + n_b$  squarings and  $k - 1 + l$  multiplications in  $\mathbb{F}_p$ .

$$P-192: e = (2^{128} - 1)2^{62}$$

$$w_1 \xrightarrow{\cdot 2} w_2 \xrightarrow{\cdot 2^2} w_4 \xrightarrow{\cdot 2^4} w_8 \xrightarrow{\cdot 2^8} w_{16} \xrightarrow{\cdot 2^{16}} w_{32} \xrightarrow{\cdot 2^{32}} w_{64} \xrightarrow{\cdot 2^{64}} w_{128}$$

Cost: 7M + 127S and 62S.

$$P-384: e = \sum_{i=127}^{381} 2^i + \sum_{i=94}^{125} 2^i + 2^{30}$$

$$w_1 \xrightarrow{\cdot 2} w_2 \xrightarrow{\cdot 2^2} w_4 \xrightarrow{\cdot 2^4} w_8 \xrightarrow{\cdot 2^8} w_{16} \xrightarrow{\cdot 2^{16}} w_{32} \xrightarrow{\cdot 2^{32}} w_{64} \xrightarrow{\cdot 2^{16}} w_{80} \xrightarrow{\cdot 2^4} w_{84} \xrightarrow{\cdot 2} w_{85} \xrightarrow{\cdot 2^{85}} w_{170} \xrightarrow{\cdot 2^{85}} w_{255}$$

Cost: 254S + 11M,  $k - 1 = 2M$ , 127S.



## Software results, OpenSSL point decompression ( $\mu\text{s}$ )

p	OpenSSL Sliding Window	Binary	New Method	Savings (%)
wtls8	159.9	214.3	123.1	23.0 42.6
secp128r1	159.3	152.6	129.8	18.5 14.9
secp160r1	278.1	365.3	224.4	19.3 38.6
secp160k1	300.7	393.9	234.4	22.0 40.5
wtls9	288.8	385.2	231.3	19.9 40.0
secp192r1	358.6	299.6	190.8	46.8 36.3
secp192k1	386.7	547.0	305.4	21.0 44.2
p239v1	507.9	616.6	412.8	18.7 33.1
secp256r1	516.6	310.8	280.5	45.7 9.7
secp256k1	597.5	804.3	431.4	27.8 46.4
secp384r1	1574.7	1386.3	789.9	49.8 43.0



## Hardware, implementation summary

p	Multiplier size	Reduction method	Memory size (elements)	M	S	S/M
secp128r1	64x64-bit	Mersenne-like	256x128-bit (256)	9	8	0.89
secp160r1	80x80-bit	Mersenne-like	256x160-bit (256)	9	8	0.89
secp160k1	80x80-bit	Crandall	256x160-bit (256)	9	8	0.89
wtls9	80x80-bit	Crandall	256x160-bit (256)	10	9	0.90
secp192r1	96x96-bit	NIST	256x192-bit (256)	9	8	0.89
secp384r1	96x96-bit	NIST	256x192-bit (128)	29	23	0.79

## Hardware timings ( $\mu$ s) (Altera Stratix II EP2S180C3 FPGA)

p	Binary		New Method		Speedup	Savings (%)
	Latency	Time ( $\mu$ s)	Latency	Time ( $\mu$ s)		
secp128r1	1,426	17.78	1,205	15.03	8.63	15.4
secp160r1	2,408	29.00	1,328	15.99	14.03	44.9
secp160k1	2,913	64.32	1,562	34.49	6.80	46.4
wtls9	3,154	47.39	1,713	25.74	8.98	45.7
secp192r1	2,971	40.25	1,771	23.99	7.95	40.4
secp384r1	17,755	448.93	9,535	241.09	3.28	46.3



## Conclusion

- We can save lots of field multiplications in point decomposition by using simple addition chains.
- Typical use is point decomposition, then scalar multiplication: your mileage may vary.
- Questions? Comments?

