Tampere University

BATCH BINARY WEIERSTRASS

LatinCrypt 2019, Santiago de Chile

04 October 2019

Billy Bob Brumley Sohaib ul Hassan Alex Shaindlin
Nicola Tuveri Kide Vuojärvi
Network and Information Security Group (NISEC)
Tampere University, Tampere, FINLAND

Bitslicing: extreme SIMD

SSE2 SIMD

You could do 32-bit ops in 4-way parallel ($w = 128$).

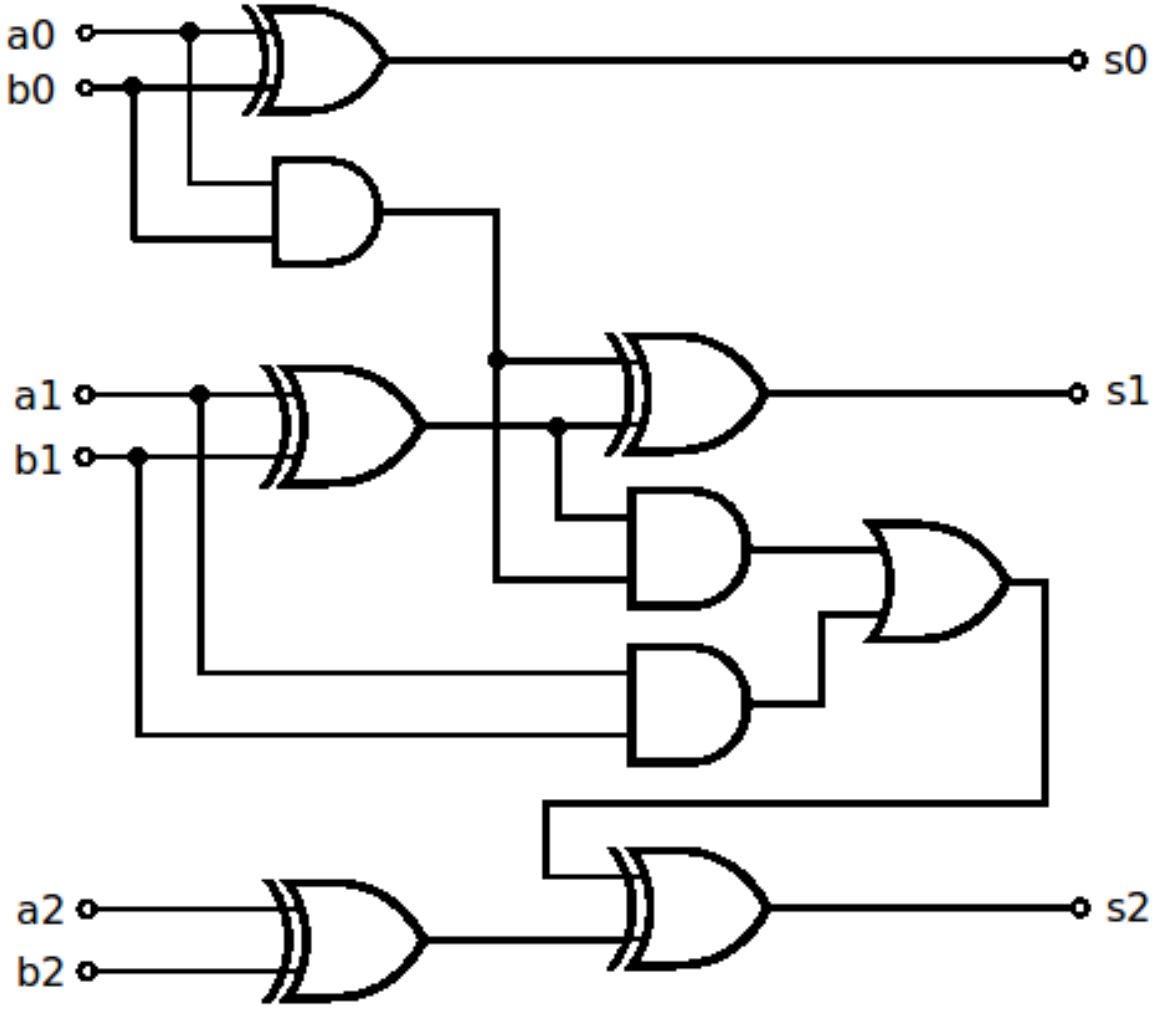
```
lane:       3       2       1       0
      +------+------+------+------+
      |7F..60|5F..40|3F..20|1F..00|
      +------+------+------+------+
```

Bitslicing

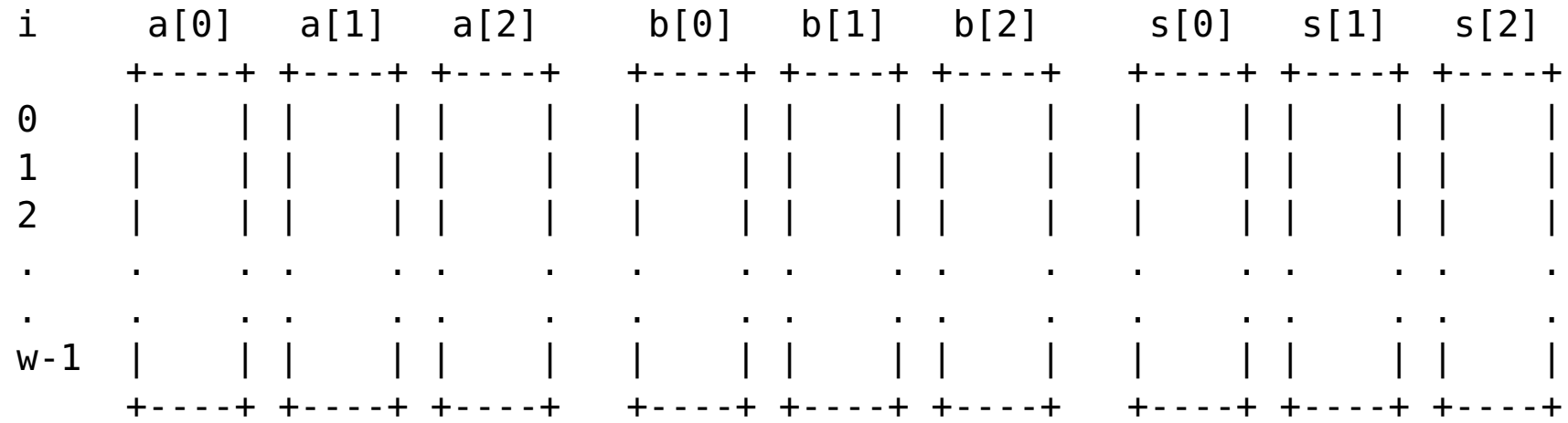
You could do 1-bit ops in w-way parallel.

```
lane:  7F 7E .. 02 01 00
      +--+--+--+--+--+--+
      |7F|7E|..|02|01|00|
      +--+--+--+--+--+--+
```

3-bit ripple carry adder: addition modulo 8



Parallel bitsliced implementation (1)



Parallel bitsliced implementation (2)

```
void add(reg_t *s, const reg_t *a, const reg_t *b) {
  reg_t t0, cc, s0, s1, s2,
        a0 = a[0], a1 = a[1], a2 = a[2],
        b0 = b[0], b1 = b[1], b2 = b[2];
  s0 = XOR(a0, b0);
  cc = AND(a0, b0);
  t0 = XOR(a1, b1);
  s1 = XOR(cc, t0);
  cc = AND(cc, t0);
  t0 = AND(a1, b1);
  cc =  OR(cc, t0);
  t0 = XOR(a2, b2);
  s2 = XOR(cc, t0);
  s[0] = s0; s[1] = s1; s[2] = s2;
}
```

This computes (in *w*-way parallel) 3-bit integer additions: `s[0],s[1],s[2]` holds the 3-bit results, result for instance *i* aligned at position *i* in *s*.

Bitslicing in the wild

Bitsliced symmetric crypto

- Primitive designs (Serpent, SHA3, Ascon, . . .)
- OpenSSL (AES, Käsper & Schwabe CHES 2009)
- JohnTheRipper (DES)

Bitsliced PKC

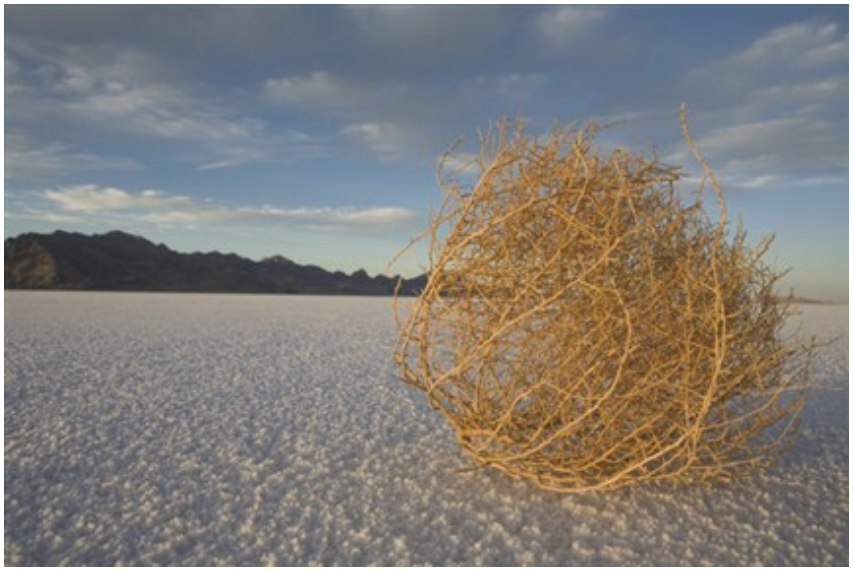


Figure: https://commons.wikimedia.org/wiki/File:PlantaRodadora.jpg

PKC bitslicing: binary Edwards curves

CRYPTO 2009

Batch Binary Edwards

Daniel J. Bernstein*

Department of Computer Science (MC 152)
The University of Illinois at Chicago
Chicago, IL 60607–7053
djb@cr.yp.to

Abstract. This paper sets new software speed records for high-security Diffie-Hellman computations, specifically 251-bit elliptic-curve variable-base-point scalar multiplication. In one second of computation on a $200 Core 2 Quad Q6600 CPU, this paper's software performs 30000 251-bit scalar multiplications on the binary Edwards curve $d(x+x^2+y+y^2) = (x+x^2)(y+y^2)$ over the field $\mathbf{F}_2[t]/(t^{251}+t^7+t^4+t^2+1)$ where $d = t^{57}+t^{54}+t^{44}+1$. The paper's field-arithmetic techniques can be applied in much more generality but have a particularly efficient interaction with the completeness of addition formulas for binary Edwards curves.

Good gate counts! But straightline isn't always practical when bitslicing. Why?

GF layer: splitting strategies

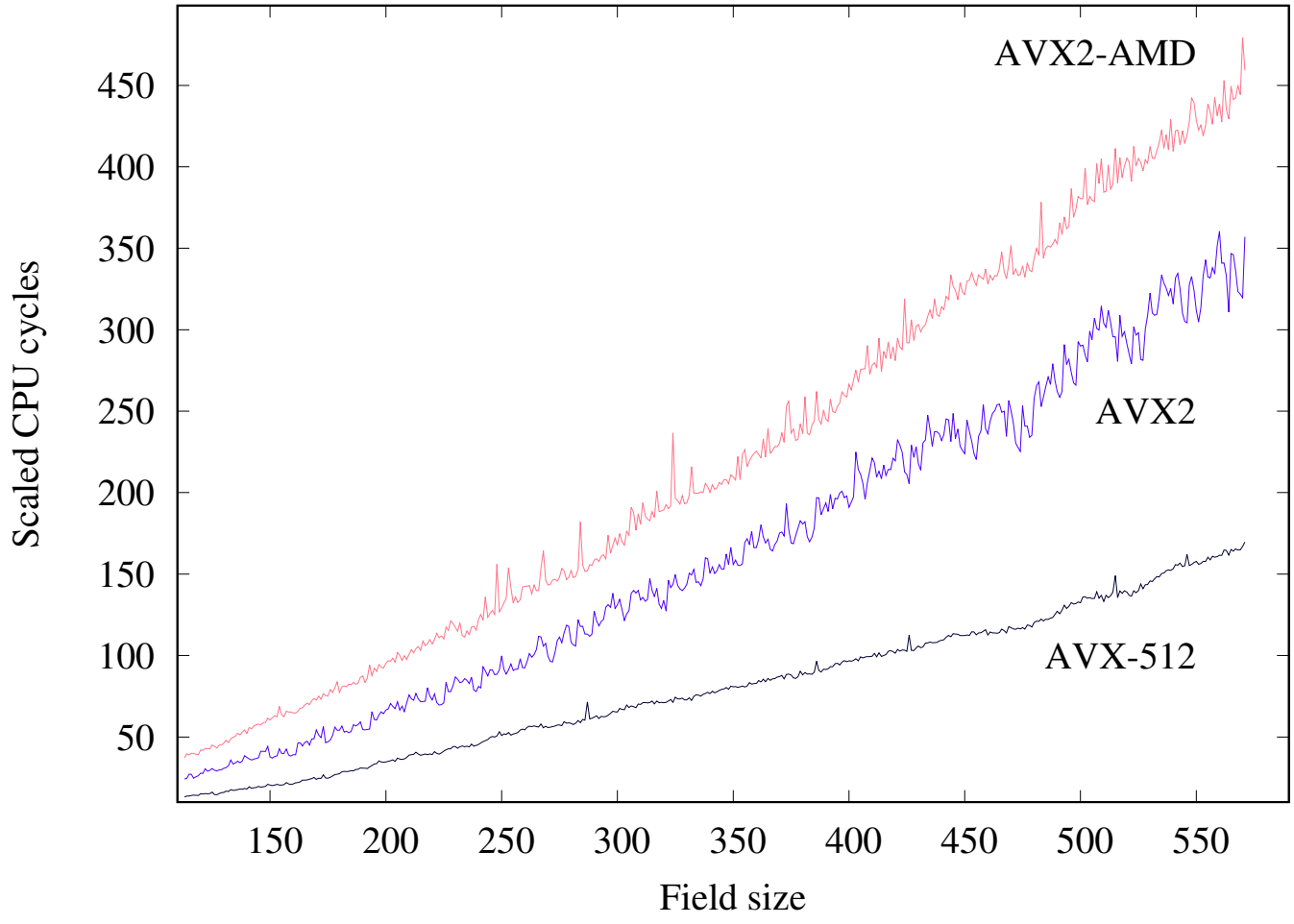
```
/* original BBE251 split */
/* (32K251, 31K85, 30K84, 32K83,
    42K30, 41K29, 40K28, 43K27, G8, G7, G6) */
WAY43(27, gf2_mul_6, gf2_mul_7) /* 9M */
WAY40(28, gf2_mul_7)
WAY41(29, gf2_mul_7)
WAY42(30, gf2_mul_7, gf2_mul_8)
WAY32(83, karatmult27, karatmult28, karatmult29)
WAY30(84, karatmult28, karatmult30) /* 5M 2E 1I */
WAY31(85, karatmult27, karatmult29, karatmult30)
WAY32(251, karatmult83, karatmult84, karatmult85)

/* another possible split -- better or worse? */
/* (43K251, 43K63, 42K62, G16, G15) */
WAY42(62, gf2_mul_15, gf2_mul_16)
WAY43(63, gf2_mul_15, gf2_mul_16)
WAY43(251, karatmult62, karatmult63)
```

GF layer: overview

Automated tool description

- Started as just $m = 251$. Then exploded.
- Supports (potentially) any field size and/or field polynomial.
- Enumerate possible field configurations. Benchmark them.
- Sometimes there's a lot. Apply a few (sane) heuristics.
- Supports several architectures: NEON, $w = 128$; AVX2, $w = 256$; AVX-512, $w = 512$

GF layer: benchmarks



m	113	131	163	191	193	233	239	**251**	283	359	409	431	571
AVX-512	13	16	22	30	31	44	44	51	59	82	99	104	170
AVX2	24	30	43	54	55	84	77	91	121	166	216	214	351
AVX2-AMD	37	47	67	83	89	114	116	132	153	224	274	300	459
NEON	228	290	425	523	534	708	724	805	928	1340	1659	1819	2953

EC layer: overview

$$E(F_{2^m}) : y^2 + xy = x^3 + ax^2 + b$$

- Started with potential changes to OpenSSL's EC module. The binary curve part.
- Why legacy curves?
- Targets key generation, i.e. kG. Allows relaxing several implementation details.
- Covered use cases: keygen, ECDSA sign, half of ECDH
- Stock ladder, stock ladder step formulae
- Curve coefficient b not guaranteed to have special form
- We tried linear maps for constant muls. But ...

EC layer: benchmarks

Curve	AVX-512	AVX2	AVX2-AMD	NEON
sect113r1	9547	18074	27470	153944
sect113r2	9540	17962	27487	153948
sect131r1	13684	26821	40478	227765
sect131r2	13639	26856	40466	228168
sect163r1	22849	45231	70046	427274
sect163r2	23175	46826	70413	448744
c2pnb163v1	23005	45017	74888	435651
c2pnb163v2	22881	45490	74413	426473
c2pnb163v3	22805	45380	74422	429631
c2tnb191v1	36094	66799	102005	632907
c2tnb191v2	36262	64963	101235	617958
c2tnb191v3	35680	64454	101325	629464
sect193r1	37899	67325	109376	639270
sect193r2	37936	67730	109434	640406
sect233r1	65491	125804	167761	1013458
c2tnb239v1	67144	119490	175914	1079930
c2tnb239v2	67105	117703	174388	1085560
c2tnb239v3	67181	120304	175676	1063116
curve2251	57756	106391	146031	870376
sect283r1	105304	218130	272544	1595423
c2tnb359v1	186680	362665	504219	2961857
sect409r1	260619	546690	697021	4229741
c2tnb431r1	283319	567608	780886	4812995
sect571r1	627668	1303759	1629335	10676160

OpenSSL layer: engine

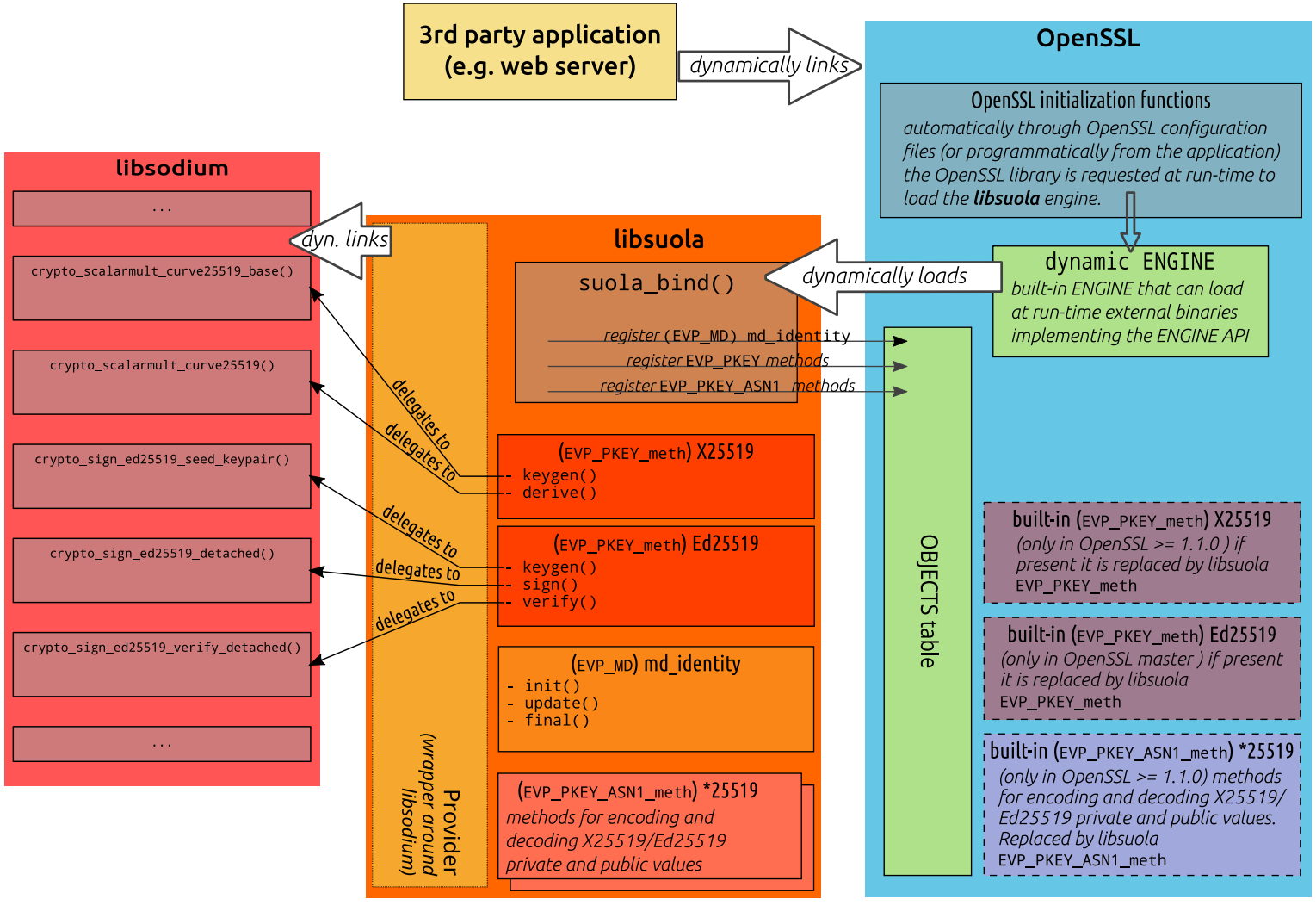


Figure: Start your ENGINES: dynamically loadable contemporary crypto (IEEE SecDev 2019, N. Tuveri, B. B. Brumley)

OpenSSL layer: AVX2 benchmarks

Curve	Average CPU cycles per operation			
	Key generation		Signature generation	
	default	`libbecc`	default	`libbecc`
`sect113r1`	309998	26758 (11.6x)	323163	25049 (12.9x)
`sect113r2`	309892	26810 (11.6x)	323586	25043 (12.9x)
`sect131r1`	515314	37617 (13.7x)	533484	35950 (14.8x)
`sect131r2`	519635	37293 (13.9x)	540221	35711 (15.1x)
`c2pnb163v1`	699094	54717 (12.8x)	718671	52785 (13.6x)
`c2pnb163v2`	690930	54603 (12.7x)	710865	52653 (13.5x)
`c2pnb163v3`	700345	54432 (12.9x)	722117	52709 (13.7x)
`sect163r1`	690258	53996 (12.8x)	719847	52548 (13.7x)
`sect163r2`	697992	54875 (12.7x)	725706	52635 (13.8x)
`c2tnb191v1`	673839	74407 (9.1x)	694368	72989 (9.5x)
`c2tnb191v2`	668479	74308 (9.0x)	695895	72811 (9.6x)
`c2tnb191v3`	669240	73836 (9.1x)	697687	73037 (9.6x)
`sect193r1`	762628	77378 (9.9x)	803603	76853 (10.5x)
`sect193r2`	758937	77109 (9.8x)	800200	76908 (10.4x)
`sect233r1`	940852	133436 (7.1x)	985741	133941 (7.4x)
`c2tnb239v1`	966659	127149 (7.6x)	1008116	126843 (7.9x)
`c2tnb239v2`	960048	126222 (7.6x)	1004913	126053 (8.0x)
`c2tnb239v3`	961976	125478 (7.7x)	1008270	125681 (8.0x)
`curve2251`	1139439	118368 (9.6x)	1198634	118661 (10.1x)
`ED25519`	130295	130254 (1.0x)	131174	129597 (1.0x)
`secp256r1`	35805	36741 (1.0x)	69228	68921 (1.0x)
`sect283r1`	1631437	226075 (7.2x)	1700907	225973 (7.5x)
`c2tnb359v1`	2016295	377226 (5.3x)	2126677	374781 (5.7x)
`sect409r1`	2731705	552476 (4.9x)	2880190	551485 (5.2x)
`c2tnb431r1`	2968140	587026 (5.1x)	3125245	579913 (5.4x)
`ED448`	960595	957772 (1.0x)	969825	969300 (1.0x)
`sect571r1`	6283098	1359731 (4.6x)	6624862	1311432 (5.1x)

Conclusion

- Over time, bitslicing is scaling well with register size
- Solid baseline for serial implementations to target
- Parallel PKC *can* be practical
- Who uses binary curves?

Future work

- OpenSSL engine: traditional SIMD (P-256, . . .)
- GitLab URLs for the libraries will be live soon (really . . .)

Marketing

- Help us help you get your stuff into OpenSSL
- We are hiring (security, applied crypto, SCA, privacy preserving protocols)