Practical post-quantum key exchange from supersingular isogenies

Craig Costello

Invited talk at SPACE 2016
December 18, 2016
CRRao AIMSCS, Hyderabad, India

Microsoft® Research

Full version of Crypto'16 paper
(joint with P. Longa and M. Naehrig)
http://eprint.iacr.org/2016/413

Full version of compression paper
(joint with D. Jao, P. Longa, M. Naehrig, D. Urbanik, J. Renes)
http://eprint.iacr.org/2016/963

SIDH library (v2.0 coming soon)
https://www.microsoft.com/en-us/research/project/sidh-library/

Diffie-Hellman key exchange (circa 1976)

$$q = 1606938044258990275541962092341162602522202993782792835301301$$

$$g = 123456789$$

$g^a \bmod q = 7846737452942265357975459631985270257549969298008577794859 3$

$5600481042932181286674410213424831338026262713942994101287 98 = g^b \bmod q$

$a =$
685408003627063
761059275919665
781694368639459
527871881531452

$b =$
362059131912941
987637880257325
269696682836735
524942246807440

$g^{ab} \bmod q = 437452857085801785219961443000845969831329749878767465041215$

Diffie-Hellman key exchange (circa 2016)

$q =$

58096059953699580628595025333045743706869751763628952366614861522872037309971102257373360445331184072513261577549805174439905295945400471216628856721870324010321116397
06440498844049850989051627200244765807041812394729680540024104827976584369381522292361208779044769892743225751738076979568811309579125511333093243519553784816306381580
16186020024749256844815024251530444957718760413642873858099017255157393414625583036640591500086964373205321856683254529110790372283163413859958640669032595972518744716
90595408050123102096390117507487600170953607342349457574162729948560133086169585299583046776370191815940885283450612858638982717634572948835466388795543116154464463301
99254382340016292057090751175533888161918987295591531536698701292267685465517437915790823154844634780260102891718032495396075041899485513811126977307478969074857043710
716150121315922024556759241239013152919710956468406379442914941614357107914462567329693649

$g = 123456789$

$g^a \pmod q =$

197496648183227193286262018614250555971909799762533760654008147994875775445667054218578105133138217497206890599554928429450667899476
854668595594034093493637562451078938296960313488696178848142491351687253054602202966247046105770771577248321682117174246128321195678
537631520278649403464797353691996736993577092687178385602298873558954121056430522899619761453727082217823475746223803790014235051396
799049446508224661850168149957401474638456716624401906701394472447015052569417746372185093302535739383791980070572381421729029651639
304234361268764971707763484300668923972868709121665568669830978657804740157916611563508569886847487772676671207386096152947607114559
706340209059103703018182635521898738094546294558035569752596676346614699327742088471255741184755866117812209895514952436160199336532
6052422101474898256696660124195726100495725510022002932814218768060112310763455404567248761396399633344901857872119208518550803791724

411604662069593306683228525653441872410777999220572079993574397237156368762038378332742471939666544968793817819321495269833613169937
986164811320795616949957400518206385310292475529284550626247132930124027703140131220968771142788394846592816111078275196955258045178
705254016469773509936925361994895894163065551105161929613139219782198757542984826465893457768888915561514505048091856159412977576049
073563225572809880970058396501719665853110101308432647427786565525121328772587167842037624190143909787938665842005691911997396726455
110758448552553744288464337906540312125397571803103278271979007681841394534114315726120595749993896347981789310754194864577435905673
172970033596584445206671223874399576560291954856168126236657381519414592942037018351232440467191228145585909045861278091800166330876
4073238447199488070126873048860279221761629281961046255219584327714817248626243962413613075956770018017385724999495117779149416882188

$= g^b \pmod q$

$a =$

7147687166405; 9571879053605547396582
692405186145916522354912615715297097
100679170037904924330116019497881089
087696131592831386326210951294944584
4004974889298038584931918128447572321
023987160439062006177648318875457556
2337708539125052923646318332191217321
464134655845254917228378772756695589
845219962202945089226966507426526912
7802446416400\9025927104004338958261
1419862375878988193612187945591802864
062679\864839578139273043684955597764
13009721221824915810964579376354556\6
554629883777859568089157882151127357
4220422646379170599917677567\30420698
422392494816906777896174923072071297
603455802621072109220\54662739697748
553543758990879608882627763290293452
560094576029847\3913613887675543866
22479265299978059886472414530462194
52761811989\9746477252908878060493
17954195146382922889045577804592943
73052654\10485180264002079415193983
85114342508427311982036827478946058
7100\304977477069244278989689910572
12096357725203480402449913844583448

$b =$

655456209464694; 93360682685816031704
969423104727624468251177438749706128
879957701\93698826859762790479113062
308975863428283798589097017957365590
672\83571386389571224667609499300898
554802446403039544300748002507962036
386619315229886063541005322448463915
89798641210273772558373965\486539312
854838650709031919742048649235894391
90352993032676961005\088404319792729
916038927477470940948581926791161465
02863521484987\086232861934222391717
121545686125300672760188085915004248
49476686\706784051068715397706852664
532638332403983747338379697022624261
377163163204493828299206039808703403
575100467337085017748387148822224875
309641791879395483731754620034884930
540399950519191679471224\05558557093
219350747155777569598163700850920394
705281936392411084\43600686183528465
724969562186437214972625833222544865
996160464558\54629937016589470425264
445624157899586972652935647856967092
689604\42796501209877036845001246792
761563917639959736383038665362727158

$g^{ab} =$

330166919524192149323761733598426244691224199958894654036331526394350099088627302979833339501183059198113987880066739
419999231378970715307039317876258453876701124543849520979430233302777503265010724513551209279573183234934359636696506
968325769489511028943698821518689496597758218540767517885836464160289471651364552490713961456608536013301649753975875
6106596555755567474438180357958360226708742348175045563437075840969230826767034061119437657466993989389348289599600338
950372251336932673571743428823026014699232071116171392219599691096846714133643382745709376112500514300983651201961186
613464267685926563624589817259637248558104903657371981684417053993082671827345252841433337325420088380059232089174946
086536664984836041334031650438692639106287627157575758383128971053401037407031731509582807639509448704617983930135028
7596589383292751993079161318839043121329118930009948197899907586986108953591420279426874779423560221038468

ECDH key exchange (1999 – nowish)

$$p = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$$

$p = 115792089210356248762697446949407573530086143415290314195533631308867097853951$

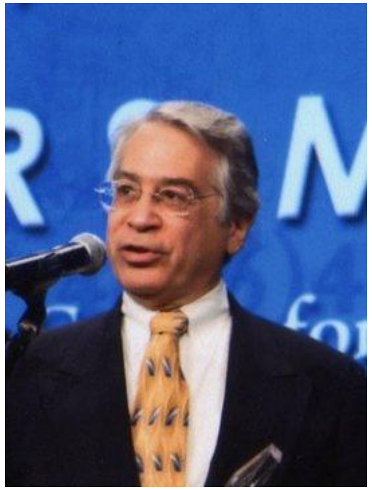
$$E/\mathbf{F}_p: y^2 = x^3 - 3x + b$$

$\#E = 115792089210356248762697446949407573529996955224135760342422259061068512044369$

$P = (48439561293906451759052585252797914202762949526041747995844080717082404635286,$
$36134250956749795798585127919587881956611106672985015071877198253568414405109)$

$[a]P = (84116208261315898167593067868200525612344221886333785331584793435449501658416,$
$102885655542185598026739250172885300109680266058548048621945393128043427650740)$

$[b]P = (101228882920057626679704131545407930245895491542090988999577542687271695288383,$
$77887418190304022994116595034556257760807185615679689372138134363978498341594)$



$a =$
$89130644591246033577639770641462855023145028492835255603183721922317324614395$

$[ab]P = (101228882920057626679704131545407930245895491542090988999577542687271695288383,$
$77887418190304022994116595034556257760807185615679689372138134363978498341594)$

$b =$
$10095557463932786418806938316190708032771910919058405391679781082193405190826$

Forthcoming post-quantum standards…

- Large-scale quantum computers break RSA, finite fields, elliptic curves
- Aug 2015: NSA announces plans to transition to quantum-resistant algorithms
- Yesterday: NIST published final call – Nov 30, 2017 deadline
  http://csrc.nist.gov/groups/ST/post-quantum-crypto/

Popular post-quantum public key primitives

- Lattice-based (e.g., NTRU'98, LWE'05)
- Code-based (e.g., McEliece'78)
- Hash-based (e.g., Merkle trees'79)
- Multivariate-based (e.g., HFE^{v-}'96)
- Isogeny-based (Jao and De Feo SIDH'11)

Current confidence may be smaller, but so are current key sizes!

SIDH: history

- **1999:** Couveignes gives talk "Hard homogenous spaces" (eprint.iacr.org/2006/291)
- **2006 (OIDH)**: Rostovsev and Stolbunov propose ordinary isogeny DH
- **2010 (OIDH break)**: Childs-Jao-Soukharev give quantum subexponential alg.
- **2011 (SIDH)**: Jao and De Feo fix by choosing supersingular curves

Crucial difference: supersingular (i.e., non-ordinary) endomorphism ring is not commutative (resists above attack)

WARNING

DO NOT BE DETERRED BY THE WORD SUPERSINGULAR

Elliptic Curves and j-invariants

- Recall that every elliptic curve E over a field K with $\mathbf{char}(K) > 3$ can be defined by
$$E : y^2 = x^3 + ax + b,$$
where $a, b \in K$, $4a^3 + 27b^2 \neq 0$

- For any extension K'/K, the set of K'-rational points forms a group with identity

- The j-invariant $j(E) = j(a, b) = 1728 \cdot \frac{4a^3}{4a^3+27b^2}$ determines isomorphism class over $\overline{K}$

- E.g., E': $y^2 = x^3 + au^2x + bu^3$ is isomorphic to E for all $u \in K^*$

- Recover a curve from j: e.g., set $a = -3c$ and $b = 2c$ with $c = j/(j - 1728)$

Isogenies

- **Isogeny:** morphism (rational map)

$$\phi : E_1 \to E_2$$

 that preserves identity, i.e. $\phi(\infty_1) = \infty_2$

- Degree of (separable) isogeny is number of elements in kernel, same as its degree as a rational map

- Given finite subgroup $G \in E_1$, there is a unique curve E_2 and isogeny $\phi : E_1 \to E_2$ (up to isomorphism) having kernel G. Write $E_2 = \phi(E_1) = E_1/\langle G \rangle$.

Torsion subgroups

- The multiplication-by-$\boldsymbol{n}$ map:

$$n : E \rightarrow E, \qquad P \mapsto [n]P$$

- The $\boldsymbol{n}$-torsion subgroup is the kernel of $[n]$

$$E[n] = \{P \in E(\overline{K}) : \ [n]P = \infty\}$$

- Found as the roots of the $\boldsymbol{n^{th}}$ division polynomial $\boldsymbol{\psi_n}$

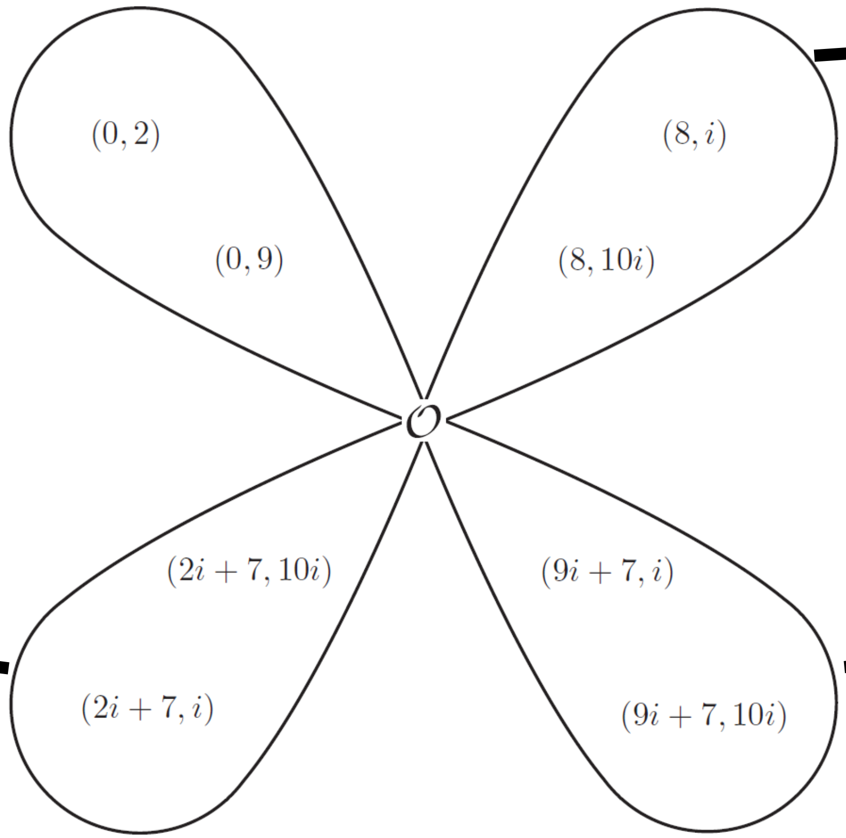
- If $\mathbf{char}(K)$ doesn't divide $\boldsymbol{n}$, then

$$E[n] \ \simeq \ \mathbb{Z}_n \times \mathbb{Z}_n$$

Recall example from tutorial: $E/\mathbb{F}_{11}: y^2 = x^3 + 4$

$E_1/\mathbb{F}_{11}: y^2 = x^3 + 2$ ← ϕ_1

ϕ_2 → $E_2/\mathbb{F}_{11}: y^2 = x^3 + 5x$



$E_3/\mathbb{F}_{11^2}: y^2 = x^3 + (7i + 3)x$ ← ϕ_3

ϕ_4 → $E_4/\mathbb{F}_{11^2}: y^2 = x^3 + (4i + 3)x$

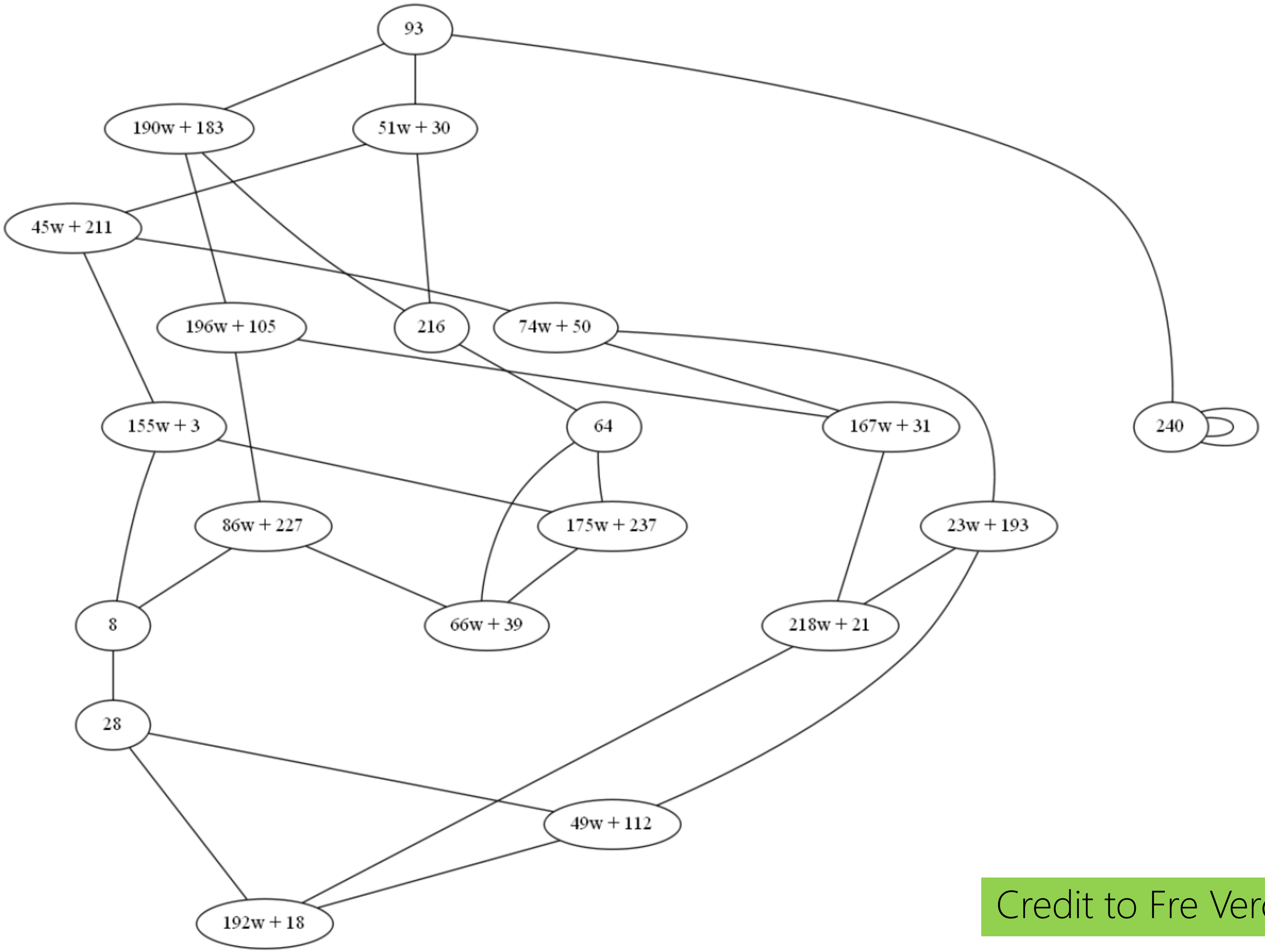
- Observe $E[3] \simeq \mathbb{Z}_3 \times \mathbb{Z}_3$, i.e., 4 cyclic subgroups of order 3 (2-dimensional)
- Velu's formulas: given E and subgroup $G \subset E$, outputs $E' = \phi(E)$ and $\phi(E)$

The supersingular isogeny graph

- SIDH works in set S_{p^2} of supersingular curves (up to $\cong$) over a fixed field
- Theorem: $\#S_{p^2} = \left\lfloor \frac{p}{12} \right\rfloor + b, \quad b \in \{0,1,2\}$
- Thm (Tate): E_1 and E_2 isogenous if and only if $\#E_1 = \#E_2$
- Thm (Mestre): all supersingular curves over $\mathbb{F}_{p^2}$ in same isogeny class
- Fact (see prev. e.g.): for every prime ℓ not dividing p, there exists $\ell + 1$ isogenies of degree ℓ originating from any supersingular curve

Upshot: immediately leads to $(\ell + 1)$ directed regular graph $X(S_{p^2}, \ell)$

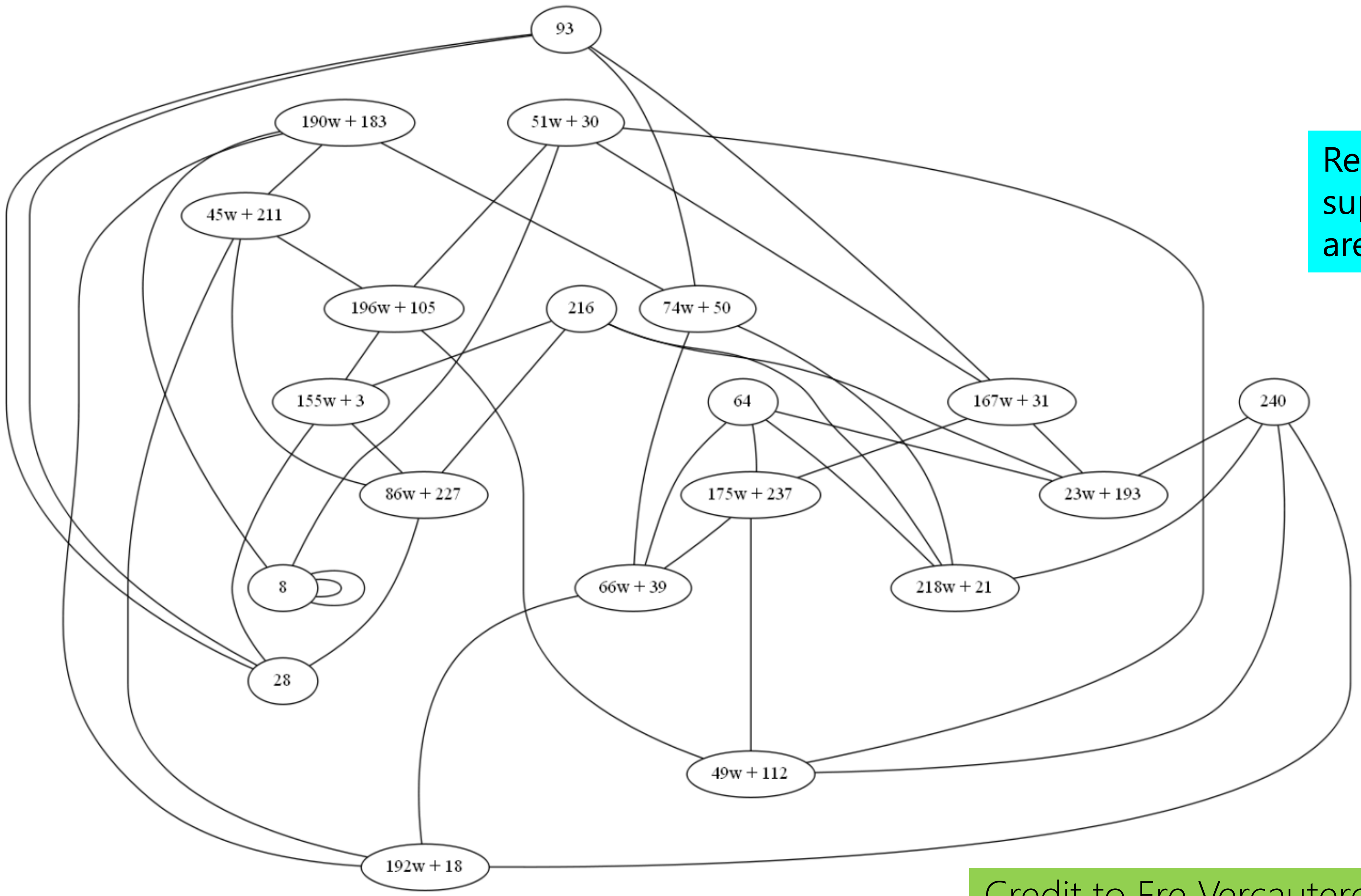
Supersingular isogeny graph for $\ell = 2$: $X(S_{241^2}, 2)$



Recall (from tutorials) that supersingular isogeny graphs are Ramanujan: **rapid mixing!**

Credit to Fre Vercauteren for example and picture…

Supersingular isogeny graph for $\ell = 3$: $X(S_{241^2}, 3)$



Recall (from tutorials) that supersingular isogeny graphs are Ramanujan: **rapid mixing!**

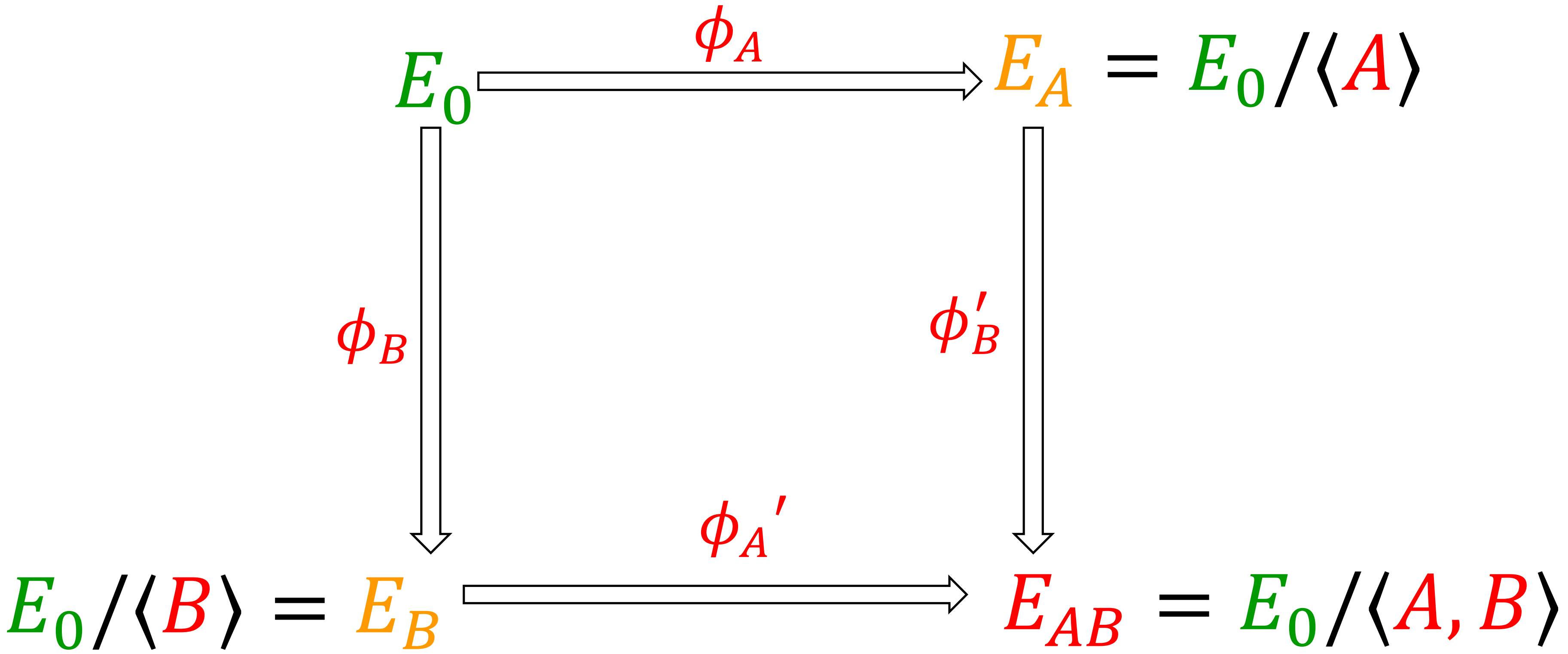
Credit to Fre Vercauteren for example and picture…

Analogues between Diffie-Hellman instantiations

	DH	ECDH	SIDH
elements	integers g modulo prime	points P in curve group	curves E in isogeny class
secrets	exponents x	scalars k	isogenies ϕ
computations	$g, x \mapsto g^x$	$k, P \mapsto [k]P$	$\phi, E \mapsto \phi(E)$
hard problem	given g, g^x find x	given $P, [k]P$ find k	given $E, \phi(E)$ find ϕ

SIDH in a nutshell:

params public private



e.g., Alice computes 2-isogenies, Bob computes 3-isogenies

SIDH in a nutshell:

params public private

Non-commutativity resolved by sending points in public keys

$$E_0 \xrightarrow{\phi_A} E_A = E_0/\langle P_A + [s_A]Q_A \rangle$$

$$(R_A, S_A) = (\phi_A(P_B), \phi_A(Q_B))$$

ϕ_B $\qquad$ ϕ_B'

$$E_0/\langle P_B + [s_B]Q_B \rangle = E_B \xrightarrow{\phi_A{}'} E_{AB} = E_0/\langle A, B \rangle$$

$$(\phi_B(P_A), \phi_B(Q_A)) = (R_B, S_B)$$

Jao & De Feo's key: Alice sends her isogeny evaluated at Bob's generators, vice versa

$$E_A/\langle R_A + [s_B]S_A \rangle \cong E_0/\langle P_A + [s_A]Q_A\, , P_B + [s_B]Q_B \rangle \cong E_B/\langle R_B + [s_A]S_B \rangle$$

SIDH shared secret is the j-invariant of E_{AB}

SIDH: security

- **Setting:** supersingular elliptic curves $E/\mathbb{F}_{p^2}$ where p is a large prime

- **Hard problem**: Given $P, Q \in E$ and $\phi(P), \phi(Q) \in \phi(E)$, compute ϕ
  (where ϕ has fixed, smooth, public degree)

- **Best (known) attacks:** classical $O(p^{1/4})$ and quantum $O(p^{1/6})$

- **Confidence:** above complexities are optimal for (above generic) claw attack

Motivation

Can we actually securely deploy SIDH?

Parameters

params public private

$$p = 2^{372}3^{239} - 1$$

$p \approx 2^{768}$ gives $\approx$ 192 bits classical and 128 bits quantum security against best known attacks

$$E_0 / \mathbb{F}_{p^2} : y^2 = x^3 + x$$

$$\#E_0 = (p+1)^2 = (2^{372}3^{239})^2$$

Easy ECDLP

$$P_A, P_B \in E_0(\mathbb{F}_p),\ Q_A = \tau(P_A),\ Q_B = \tau(P_B)$$

376 bytes

48 bytes

$$s_A, s_B \in \mathbb{Z}$$

$$\mathrm{PK} = [x(P), x(Q), x(Q-P)] \in (\mathbb{F}_{p^2})^3$$

564 bytes

188 bytes

$$j(E_{AB}) \in \mathbb{F}_{p^2}$$

Exploiting smooth degree isogenies

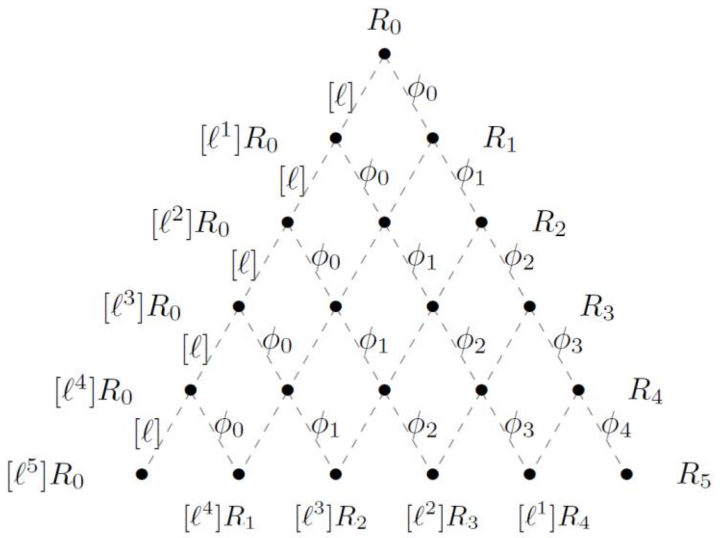
- Computing isogenies of prime degree ℓ at least $O(\ell)$
- We need exponential #secrets↔ #isogenies↔#kernel subgroups
- Upshot: isogenies must have exponential degree. Can't compute unless smooth!
- We will only use isogenies of degree ℓ^e for $\ell \in \{2,3\}$

Exploiting smooth degree isogenies

- Suppose secret point R_0 has order 2^{372}, we need $\phi : E \to E/\langle R_0 \rangle$

- Factor $\phi = \phi_{371} \ldots \phi_1 \phi_0$, with ϕ_i are 2-isogenies, and walk to $E/\langle R_0 \rangle$

$$\begin{aligned}
\phi_0 &= E_0 \to E_0/\langle [\ell^4] R_0 \rangle\,, & R_1 &= \phi_0(R_0); \\
\phi_1 &= E_1 \to E_1/\langle [\ell^3] R_1 \rangle\,, & R_2 &= \phi_1(R_1); \\
&\vdots & &\vdots \\
\phi_{370} &= E_{370} \to E_{370}/\langle [\ell^1] R_{370} \rangle\,, & R_{371} &= \phi_{370}(R_{370}) \\
\phi_{371} &= E_{371} \to E_{371}/\langle R_{371} \rangle\,.
\end{aligned}$$



- The above is naïve: there is a much faster way (see [DJP'14]).
- SIDH requires two types of arithmetic: $[m]P \in E$ and $\phi : E \to E'$

Our performance improvements

1. Projective isogenies $\rightarrow \mathbb{P}^1$ everywhere

2. Fast $\mathbb{F}_{p^2}$ arithmetic

3. Tight public parameters

(just 1 today...)

Point *and* isogeny arithmetic in $\mathbb{P}^1$

ECDH: move around different points on a fixed curve.

SIDH: move around different points and different curves

$$E_{\mathrm{a,b}}: \quad by^2 = x^3 + ax^2 + x$$

$(x, y) \leftrightarrow (X : Y : Z)$ $\quad\quad$ $(a, b) \leftrightarrow (A : B : C)$

$$E_{(\mathrm{A:B:C})}: \quad BY^2Z = CX^3 + AX^2Z + CXZ^2$$

The Montgomery B coefficient only fixes the quadratic twist. Can ignore it in SIDH since $j(E) = j(E')$

$\mathbb{P}^1$ point arithmetic (Montgomery): $(X : Z) \mapsto (X' : Z')$

$\mathbb{P}^1$ isogeny arithmetic (this work): $(A : C) \mapsto (A' : C')$

what was…

$$G\ :\ \frac{B}{2-A}y^2 = x^3 - 2\frac{A+6}{2-A}x^2 + x,$$
$$\psi : F \to G,$$
$$(x,y) \mapsto \left(\frac{1}{2-A}\frac{(x+4)(x+(A+2))}{x}, \frac{y}{2-A}\left(1-\frac{4(2+A)}{x^2}\right)\right)$$

```
void iso2_comp(iso2* iso, GF* iA, GF* iB, GF* iA24,
               const GF A, const GF B,
               const GF x, const GF z) {
  GF* tmp = x.parent->GFtmp;

  sub_GF(&tmp[0], x, z);
  sqr_GF(&tmp[1], tmp[0]);
  inv_GF(&tmp[0], tmp[1]);
  mul_GF(&tmp[1], tmp[0], z);
  mul_GF(iso, tmp[1], x); // iA2 = x z / (x-z)^2
  add_GF_ui(&tmp[0], A, 6);
  mul_GF(iB, B, *iso); // iB = B iA2
  mul_GF(iA, tmp[0], *iso); // iA = (A+6) iA2
  a24(iA24, *iA);
}
```

Division in $\mathbb{F}_p$

… is now (division-free):

$$(A' : C') = \left(2(2X_4^4 - Z_4^4) : Z_4^4\right),$$

$$(X' : Z') = \left(X\left(2X_4Z_4Z - X(X_4^2+Z_4^2)\right)(X_4X - Z_4Z)^2 :\right.$$
$$\left. Z\left(2X_4Z_4X - Z(X_4^2+Z_4^2)\right)(Z_4X - X_4Z)^2\right).$$

```
void get_4_isog(point_proj_t P, f2elm_t A, f2elm_t C, f2elm_t* coeff)
{ // Computes the corresponding 4-isogeny of a projective Montgomery point (X4:Z4) of order 4.
  // Input:  projective point of order four P = (X4:Z4).
  // Output: the 4-isogenous Montgomery curve with projective coefficient A/C and the 5 coefficients
  //         that are used to evaluate the isogeny at a point in eval_4_isog().

    fp2add751(P->X, P->Z, coeff[0]);                  // coeff[0] = X4+Z4
    fp2sqr751_mont(P->X, coeff[3]);                   // coeff[3] = X4^2
    fp2sqr751_mont(P->Z, coeff[4]);                   // coeff[4] = Z4^2
    fp2sqr751_mont(coeff[0], coeff[0]);               // coeff[0] = (X4+Z4)^2
    fp2add751(coeff[3], coeff[4], coeff[1]);          // coeff[1] = X4^2+Z4^2
    fp2sub751(coeff[3], coeff[4], coeff[2]);          // coeff[2] = X4^2-Z4^2
    fp2sqr751_mont(coeff[3], coeff[3]);               // coeff[3] = X4^4
    fp2sqr751_mont(coeff[4], coeff[4]);               // coeff[4] = Z4^4
    fp2add751(coeff[3], coeff[3], A);                 // A = 2*X4^4
    fp2sub751(coeff[0], coeff[1], coeff[0]);          // coeff[0] = 2*X4*Z4 = (X4+Z4)^2 - (X4^2+Z4^2)
    fp2sub751(A, coeff[4], A);                        // A = 2*X4^4-Z4^4
    fp2copy751(coeff[4], C);                          // C = Z4^4
    fp2add751(A, A, A);                               // A = 2(2*X4^4-Z4^4)
}
```

Performance benchmarks

SIDH operation	This work*	Prior work (AFJ'14)
Alice key generation	46	149
Bob key generation	52	152
Alice shared secret	44	118
Bob shared secret	50	122
Total	**192**	**540**

Table: millions of clock cycles for DH operations on 3.4GHz Intel Core i7-4770 (Haswell)

*includes full protection against timing and cache attacks

BigMont: a strong SIDH+ECDH hybrid

- No clear frontrunner for PQ key exchange
- Hybrid particularly good idea for (relatively young) SIDH
- Hybrid particularly easy for SIDH

There are exponentially many A such that $E_A\,/\mathbb{F}_{p^2}\colon y^2 = x^3 + Ax^2 + x$ is in the supersingular isogeny class. These are all unsuitable for ECDH.

There are also exponentially many A such that $E_A\,/\mathbb{F}_{p^2}\colon y^2 = x^3 + Ax^2 + x$ is suitable for ECDH, e.g. $A = 624450$.

SIDH vs. SIDH+ECDH hybrid

comparison		SIDH	SIDH+ECDH
bit security (hard problem)	classical	192 (SSDDH)	384 (ECDHP)
	quantum	128 (SSDDH)	128 (SSDDH)
public key size (bytes)		564	658
Speed (cc x 10^6)	Alice key gen.	46	52
	Bob key gen.	52	58
	Alice shared sec.	44	50
	Bob shared sec.	50	57

Colossal amount of classical security almost-for-free (≈ no more code)

SIDH vs. lattice “DH” primitives

Name	Primitive	Full DH (ms)	PK size (bytes)
Frodo	LWE	2.600	11,300
NewHope	R-LWE	0.310	1,792
NTRU	NTRU	2.429	1,024
SIDH	Supersingular Isogeny	900	564

Table: ms for full DH round (Alice + Bob) on 2.6GHz Intel Xeon i5 (Sandy Bridge)
See “Frodo” for benchmarking details.

All numbers above are for plain C implementations (e.g., SIDH w. assembly optimizations is 56ms)

Compression of public keys

- Azarderakhsh, Jao, Kalach, Koziel, Leonardi: instead of sending points with E, send scalars w.r.t. deterministic basis generating $E[n]$

- e.g., instead of sending $P \in E(\mathbb{F}_{p^2})[2^{372}]$, send $\alpha, \beta \in \mathbb{Z}_{2^{372}}$ such that $P = [\alpha]Q + [\beta]R$ for some "canonical" basis $\{Q, R\}$ of $E(\mathbb{F}_{p^2})[2^{372}]$ that Alice and Bob can compute from E alone

- Azarderakhsh et al. show that decomposing $P \mapsto \alpha, \beta$ costs roughly 10 times a full round of SIDH!!!

Efficient compression of public keys

- Three stages to SIDH public key compression $P \mapsto \alpha, \beta$

- Step 1: compute deterministic basis $Q, R \in E[n]$
- Step 2: compute pairings to transform discrete logarithms into μ_n^*
- Step 3: solve discrete logarithms using Pohlig-Hellman

(C-Jao-Longa-Naehrig-Renes-Urbanik: http://eprint.iacr.org/2016/963)

- Step 1: much faster bases computations using 2 & 3 descent 👍
- Step 2: much faster pairing computations using optimized Tate not Weil 👍
- Step 3: much faster PH using optimized windowing approach 👍

Performance benchmarks

Full round SIDH (Alice+Bob)	This work*	Prior work (AJKKL'16)
no compression	192	535
compression	510	15,395

Table: millions of clock cycles for DH operations (Haswell) scaled – see paper.

Compressed SIDH vs. lattice "DH" primitives

Name	Primitive	Full DH (ms)	PK size (bytes)
Frodo	LWE	2.600	11,300
NewHope	R-LWE	0.310	1,792
NTRU	NTRU	2.429	1,024
SIDH	Supersingular Isogeny	≈ **2390**	330

Compressed SIDH roughly 2-3 slower than uncompressed SIDH.

Validating public keys

- Issues regarding public key validation: Asiacrypt2016 paper by Galbraith-Petit-Shani-Ti
- NSA countermeasure: “Failure is not an option: standardization issues for PQ key agreement”
- Thus, library currently supports ephemeral DH only

Future work

- Cryptanalysis!
- Faster SIDH
- SIDH with static keys
- SI signatures

Thanks!

Full version of Crypto'16 paper
(joint with P. Longa and M. Naehrig)
http://eprint.iacr.org/2016/413

Full version of compression paper
(joint with D. Jao, P. Longa, M. Naehrig, D. Urbanik, J. Renes)
http://eprint.iacr.org/2016/963

SIDH library (v2.0 coming soon)
https://www.microsoft.com/en-us/research/project/sidh-library/