An improved compression technique for signatures based on learning with errors

Shi Bai and Steven D. Galbraith

Department of Mathematics,
University of Auckland.

CT-RSA 2014

Outline

Introduction

We describe a new approach to the compression technique of Lyubashevsky et al. [8, 5] for lattice-based signatures based on the learning with errors (LWE) and short integer solution (SIS) problems. Our main focus is to reduce the size of signatures.

The security of the signature, in the random oracle model, is based on worst-case general lattice assumptions.

The signature size, to the best of our knowledge, is smaller than any previous proposal for provably-secure signatures based on standard lattice problems: at the 128-bit level we improve the signature size from more than 16500 bits [8] to around 9000 bits.

Lattice-based cryptosystem

Lattice-based cryptosystems are often built upon the average-case hardness of the short integer solution problem (SIS) and learning with errors (LWE) problems.

- SIS: Given a matrix $\mathbf{B} \in \mathbb{Z}_q^{n \times m}$ where $n < m$, find a "short" vector $\mathbf{v}$ such that $\mathbf{Bv} \equiv 0 \pmod{q}$
- LWE: Let χ and ϕ be distributions on $\mathbb{Z}$. Given $\mathbf{s} \leftarrow \chi^n$ and $e \leftarrow \phi^m$, let $\mathbf{A}$ be uniform over $\mathbb{Z}_q^{m \times n}$. The LWE problem is to compute the pair $(\mathbf{s}, \mathbf{e})$ given $\mathbf{A}, \mathbf{b}$ where $\mathbf{b} = \mathbf{As} + \mathbf{e} \pmod{q}$. LWE is well-defined if parameters are chosen properly.

The security of our signature schemes will be based on them (and their variants).

Related work

- A series of works by Lyubashevsky [7, 8] have developed efficient schemes using the Fiat-Shamir paradigm in the random oracle model. At the 100-bit security level, the signatures are about 16500 bits.
- Güneysu, Lyubashevsky and Pöppelmann [5] described a compression technique which further reduces the signature size. The security depends on the Ring-SIS and DCK (an NTRU-like variant of Ring-LWE with small parameters) assumptions. At around the 100-bit security level, the signatures are about 9000 bits.
- Another approach is to use the trapdoor functions and the hash-and-sign methodology (see Gentry, Peikert and Vaikuntanathan [4], Stehlé and Steinfeld [9]).

Lyubashevsky's signature scheme [8]

Algorithm Key generation (LWE)

1: $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times n}$, $\mathbf{S} \leftarrow \sigma_S^{n\times k}$, $\mathbf{E} \leftarrow \sigma_E^{n\times k}$
2: $\mathbf{T} \equiv \mathbf{AS} + \mathbf{E} \pmod q$
3: **return** $\mathbf{A}, \mathbf{T}$

Algorithm Signing

INPUT: $\mu, \mathbf{A}, \mathbf{T}, \mathbf{S}, \mathbf{E}, \mathrm{H}, B, D$
OUTPUT: $\mathbf{z}_1, \mathbf{z}_2, \mathbf{c}$

1: $\mathbf{y}_1, \mathbf{y}_2 \leftarrow [-D, D]^n$
2: $\mathbf{v} \equiv \mathbf{A}\mathbf{y}_1 + \mathbf{y}_2 \pmod q$
3: $\mathbf{c} = \mathrm{H}(\mathbf{v}, \mu)$
4: $\mathbf{z}_1 = \mathbf{y}_1 + \mathbf{Sc}$, $\mathbf{z}_2 = \mathbf{y}_2 + \mathbf{Ec}$
5: **return** $(\mathbf{z}_1, \mathbf{z}_2, \mathbf{c})$ when $\|\mathbf{z}_i\|_\infty \leq B$

Algorithm Verifying

INPUT: $\mu, \mathbf{z}_1, \mathbf{z}_2, \mathbf{c}, \mathbf{A}, \mathbf{T}, B, \mathrm{H}$
OUTPUT: Accept or Reject

1: $\mathbf{c}' = \mathrm{H}(\mathbf{A}\mathbf{z}_1 + \mathbf{z}_2 - \mathbf{Tc}, \mu)$
2: **if** $\mathbf{c}' = \mathbf{c}$ and $\|\mathbf{z}_i\|_\infty \leq B$ **then**
3: **return** "Accept"
4: **else**
5: **return** "Reject"
6: **end if**

Security

Theorem (Lyubashevsky [8])

If there is a polynomial-time forger $\mathcal{F}$, who makes s queries to the signing oracle and h queries to the random oracle H, who breaks the signature with non-trivial probability, then there exists a polynomial-time algorithm who can solve the the search-SIS problem (for some bound β) with non-trivial probability.

Some considerations on the security:

1. SIS: The forgery SIS problem is hard.
2. LWE: Given $(\mathbf{A}, \mathbf{T})$, keys $\mathbf{S}$ and $\mathbf{E}$ need to be secure.
3. Rejection sampling: Adversaries may attempt a statistical analysis of the values $(\mathbf{z}_1, \mathbf{z}_2, \mathbf{c})$. We need to make sure $\mathbf{y}_i$ are large enough so $(\mathbf{z}_1, \mathbf{z}_2, \mathbf{c})$ is independent of the secrets.
4. Hash function H.

GLP's compression technique [5]

Instead of sending $(\mathbf{z}_1, \mathbf{z}_2, \mathbf{c})$, GLP [5] uses a compression technique to send $(\mathbf{z}_1, \mathbf{z}_2', \mathbf{c})$ where $\mathbf{z}_2'$ is much smaller than $\mathbf{z}_2$ such that $R(\mathbf{A}\mathbf{z}_1 + \mathbf{z}_2 - \mathbf{T}\mathbf{c}) = R(\mathbf{A}\mathbf{z}_1 + \mathbf{z}_2' - \mathbf{T}\mathbf{c})$ where R is some truncating (rounding) function.

Algorithm GLP Signing

INPUT:

$\mu, \mathbf{A}, \mathbf{T}, \mathbf{S}, \mathbf{E}, \mathrm{H}, B, D, R, C$

OUTPUT: $\mathbf{z}_1, \mathbf{z}_2, \mathbf{c}$

1: $\mathbf{y}_1, \mathbf{y}_2 \leftarrow [-D, D]^n$
2: $\mathbf{v} \equiv R(\mathbf{A}\mathbf{y}_1 + \mathbf{y}_2) \pmod q$
3: $\mathbf{c} = \mathrm{H}(\mathbf{v}, \mu)$
4: $\mathbf{z}_1 = \mathbf{y}_1 + \mathbf{S}\mathbf{c},\ \mathbf{z}_2 = \mathbf{y}_2 + \mathbf{E}\mathbf{c}$
5: $\mathbf{z}_2' = C(\mathbf{A}\mathbf{z}_1 - \mathbf{T}\mathbf{c}, \mathbf{z}_2, B)$
6: **return** $(\mathbf{z}_1, \mathbf{z}_2', \mathbf{c})$ when $\|\mathbf{z}_i\|_\infty \leq B$

Algorithm GLP Verifying

INPUT: $\mu, \mathbf{z}_1, \mathbf{z}_2', \mathbf{c}, \mathbf{A}, \mathbf{T}, \mathrm{H}, B, R$

OUTPUT: Accept or Reject

1: $\mathbf{c}' = \mathrm{H}(R(\mathbf{A}\mathbf{z}_1 + \mathbf{z}_2' - \mathbf{T}\mathbf{c}), \mu)$
2: **if** $\mathbf{c}' = \mathbf{c}$ and $\|\mathbf{z}_1\|_\infty, \|\mathbf{z}_2'\|_\infty \leq B$ **then**
3: **return** "Accept"
4: **else**
5: **return** "Reject"
6: **end if**

Improved compression technique

At a high level, Lyubashevsky and GLP's schemes behave like a proof of knowledge of the pair $(\mathbf{s}, \mathbf{e})$. Our scheme proves knowledge of only $\mathbf{s}$. The proof of knowledge of $\mathbf{e}$ becomes implicit in the verification: so no longer need to send any information about $\mathbf{e}$.

- Lyubashevsky's scheme:
  - Generating $\mathbf{y}_1, \mathbf{y}_2$;
  - Hashing $\mathrm{H}(\mathbf{A}\mathbf{y}_1 + \mathbf{y}_2)$;
  - Signing $\mathbf{z}_1 = \mathbf{y}_1 + \mathbf{S}\mathbf{c},\ \mathbf{z}_2 = \mathbf{y}_2 + \mathbf{E}\mathbf{c}$;
  - Verifying $\mathbf{A}\mathbf{z}_1 + \mathbf{z}_2 - \mathbf{T}\mathbf{c} \equiv \mathbf{A}\mathbf{y}_1 + \mathbf{y}_2 \pmod{q}$;
- New signature:
  - Generating $\mathbf{y}_1$;
  - Hashing $\mathrm{H}(R(\mathbf{A}\mathbf{y}_1))$;
  - Signing $\mathbf{z}_1 = \mathbf{y}_1 + \mathbf{S}\mathbf{c}$;
  - Verifying $R(\mathbf{A}\mathbf{z}_1 - \mathbf{T}\mathbf{c}) \equiv R(\mathbf{A}\mathbf{y}_1 - \mathbf{E}\mathbf{c}) \equiv R(\mathbf{A}\mathbf{y}_1) \pmod{q}$;

Improved compression technique and signature

- The Lyubashevsky signature needs to send is $(\mathbf{z}_1, \mathbf{z}_2, \mathbf{c})$ where $\mathbf{z}_i$ are length n vectors.
  - Signature size $\approx 2n \log 2B$.
- The GLP technique compresses $\mathbf{z}_2$ (by dropping bits in $\mathbf{A}\mathbf{y}_1 + \mathbf{y}_2$). The signature to send is $(\mathbf{z}_1, \mathbf{z}_2', c)$ for a $\mathbf{z}_2'$ with very small entries.
  - Signature size $\approx n \log 2B + 2n$.
- We describe an improved "compression" technique which drops $\mathbf{z}_2$ completely.
  - Signature size $\approx n \log 2B$.

Signature

Key generation is the same as before.

Algorithm Key generation (LWE)

INPUT: $n, m, k, q, \sigma_S = \sigma_E$
OUTPUT: $\mathbf{A}, \mathbf{T}$
1: $\mathbf{A} \leftarrow \mathbb{Z}_q^{m \times n}$, $\mathbf{S} \leftarrow \sigma_S^{n \times k}$, $\mathbf{E} \leftarrow \sigma_E^{m \times k}$
2: $\mathbf{T} \equiv \mathbf{AS} + \mathbf{E} \pmod{q}$
3: **return** $\mathbf{A}, \mathbf{T}$

For $a \in \mathbb{Z}$ and $d \in \mathbb{N}$, let $[a]_{2^d} \in (-2^{d-1}, 2^{d-1}]$ be such that $[a]_{2^d} \equiv a \pmod{2^d}$. Define $\lfloor a \rceil_d = (a - [a]_{2^d})/2^d$ (dropping the d-least significant bits).

Let H be a hash function to binary strings c of fixed length κ, and F be an encoding function that maps c to $\mathbf{c}$ (length $k = n$ vectors of weight w in $\{-1, 0, 1\}$).

Signature (cont.)

Algorithm Signing

INPUT:
$\mu, \mathbf{A}, \mathbf{T}, \mathbf{S}, d, w, \sigma_E, \mathrm{H}, \mathrm{F}, B, D$
OUTPUT: $(\mathbf{z}, c)$

$\mathbf{y} \leftarrow [-D, D]^n$
$\mathbf{v} \equiv \mathbf{A}\mathbf{y} \pmod{q}$
$c = \mathrm{H}(\lfloor \mathbf{v} \rceil_d, \mu)$
$\mathbf{c} = \mathrm{F}(c)$
$\mathbf{z} = \mathbf{y} + \mathbf{S}\mathbf{c}$
$\mathbf{w} \equiv \mathbf{A}\mathbf{z} - \mathbf{T}\mathbf{c} \pmod{q}$
if $|[\mathbf{w}_i]_{2^d}| > 2^{d-1} - 7w\sigma_E$ **then**
    Restart
end if
return $(\mathbf{z}, c)$ if $\|\mathbf{z}\|_\infty \leq B$

Algorithm Verifying

INPUT: $\mu, \mathbf{z}, c, \mathbf{A}, \mathbf{T}, B, d, \mathrm{H}, \mathrm{F}$
OUTPUT: Accept or Reject

$\mathbf{c} = \mathrm{F}(c)$
$\mathbf{w} \equiv \mathbf{A}\mathbf{z} - \mathbf{T}\mathbf{c} \pmod{q}$
$c' = \mathrm{H}(\lfloor \mathbf{w} \rceil_d, \mu)$
if $c' = c$ and $\|\mathbf{z}\|_\infty \leq B$ **then**
    return "Accept"
else
    return "Reject"
end if

Security

Theorem

Let parameters n, m, d, κ, B, q be such that

$$(2B)^n q^{m-n} \geq (2^{d+1})^m 2^\kappa. \tag{1}$$

Let A be a forger against the signature scheme. Then A can be turned into either of the following two algorithms:

1. *an algorithm that solves some decisional-LWE problem.*
2. *an algorithm that solves some search SIS problem: Given an $m \times (n+m)$ matrix $\boldsymbol{A}'$ to find a length n vector $\boldsymbol{y}_1$ and a length m vector $\boldsymbol{y}_2$ such that $\|\boldsymbol{y}_1\|_\infty, \|\boldsymbol{y}_2\|_\infty \leq \max(2B, 2^{d-1}) + 2E'w$ and $\boldsymbol{A}'\left(\frac{\boldsymbol{y}_1}{\boldsymbol{y}_2}\right) \equiv 0 \pmod{q}$ where E' satisfies*

$$(2E')^{m+n} \geq q^m 2^\kappa. \tag{2}$$

Sketch of proof

Our proof follows the logic of Lyubashevsky's framework in [8].

- First, one simulates the signing algorithm in the random oracle model (using the rejection sampling lemma). Note we consider an adaptive security setting and hence need to handle sign queries for the adversaries.
- Second, one uses the forking lemma (by rewinding the forger) to get two signatures on the same message.

Sketch of proof: simulation of signing

Algorithm Simulation of signing

INPUT: $\mu, \mathbf{A}, \mathbf{T}, D, B, d, w, \sigma_E, \mathrm{H}, \mathrm{F}$

OUTPUT: $(\mathbf{z}, c)$

1: choose uniformly a κ-bit binary string c
2: $\mathbf{c} = \mathrm{F}(c)$
3: $\mathbf{z} \leftarrow [-D, D]^n$
4: $\mathbf{w} \equiv \mathbf{Az} - \mathbf{Tc} \pmod{q}$
5: **if** $|[\mathbf{w}_i]_{2^d}| > 2^{d-1} - 7w\sigma_E$ **then**
6: Restart
7: **end if**
8: **if** H has already been defined on $(\lfloor \mathbf{w} \rceil_d, \mu)$ **then**
9: Abort game
10: **else**
11: Program $\mathrm{H}(\lfloor \mathbf{w} \rceil_d, \mu) = c$
12: **end if**
13: **return** $(\mathbf{z}, c)$ if $\|\mathbf{z}\|_\infty \leq B$

Sketch of proof: simulation of signing (cont.)

The simulation and true signing algorithm are computationally indistinguishable.

Notes:

- Line 8, the occurrence of "H has already been defined on $(\lfloor \mathbf{w} \rceil_d, \mu)$" is negligible: if Equation (1) holds, then the probability that two values $\mathbf{y}_1, \mathbf{y}_2$ sampled uniformly from $[-B, B]^n$ give the same $\lfloor \mathbf{A}\mathbf{y} \pmod q \rceil_d$ value is at most $1/2^\kappa$;
- Line 13 uses a variant of rejection sampling lemma;

□

Sketch of proof: alternative keys

Equation (2):

$$(2E')^{m+n} \geq q^m 2^\kappa.$$

is required since the reduction (to SIS) requires existence of alternative keys. There needs to exist at least two independent pairs $(\mathbf{S}, \mathbf{E}), (\tilde{\mathbf{S}}, \tilde{\mathbf{E}})$ such that $\mathbf{T} \equiv \mathbf{AS} + \mathbf{E} \equiv \mathbf{A}\tilde{\mathbf{S}} + \tilde{\mathbf{E}} \pmod{q}$.

Algorithm Alternative key generation in the simulation

INPUT: $n, m, k, q, \sigma'_S (= \sigma'_E) \gg \sigma_S = \sigma_E$

OUTPUT: $\mathbf{A}, \mathbf{T}$

1: $\mathbf{A} \leftarrow \mathbb{Z}_q^{m \times n}$, $\mathbf{S} \leftarrow \sigma_{S'}^{n \times k}$, $\mathbf{E} \leftarrow \sigma_{E'}^{m \times k}$

2: $\mathbf{T} \equiv \mathbf{AS} + \mathbf{E} \pmod{q}$

3: **return** $\mathbf{A}, \mathbf{T}$

The forger can not tell (computationally) which algorithm generates $\mathbf{A}, \mathbf{T}$ (decisional-LWE problem).

Parameters

Table: Parameters for LWE Signatures using Uniform Distributions (more than 128 bit security).

		I	II	III	IV
n		576	512	512	400
m		969	945	1014	790
w	$2^w \cdot \binom{k}{w} \geq 2^{128}$	18	19	19	20
Approx. $\log_2(q)$		33.10	30.84	32.66	28.71
κ		132	132	132	132
σ_E		68	66	224	70
σ_S		68	66	224	70
$\log_2(B)$	$14\sigma_{\mathbf{Sc}}(n-1)$	21.15	20.97	22.74	20.74
2^d		2^{24}	2^{24}	2^{26}	2^{24}
Prob. accept. in line 7 of Alg .	$\left(1 - 14\sigma_{\mathbf{E}} w/2^d\right)^m$	0.371	0.372	0.406	0.397
Signature (bits)	$n\lceil \log_2(2B) \rceil + \kappa$	13380	11396	12420	8932
Public key (Mb)	$2mn\log_2(q)$	4.4	3.6	4.0	2.2
Signing key (Mb)	$2mn\log_2(4\sigma_S)$	1.0	0.9	1.2	0.6

Practical security of parameters

To evaluate the security of the parameters against practical lattice attacks, we consider the LWE problem for the secret key and the SIS problem in the forgery (using Chen and Nguyen's BKZ 2.0 [2] estimate).

- Solving the LWE instances in $(\mathbf{A}, \mathbf{b} \equiv \mathbf{As} + \mathbf{e} \pmod{q})$:
  - CVP problem in the lattice $\{\mathbf{v} \in \mathbb{Z}^m : \mathbf{v} \equiv \mathbf{As} \pmod{q}\}$.
  - Inhomogeneous SIS problem $\mathbf{b} = (\mathbf{A}|\mathbf{I}_m)(\mathbf{s}^T, \mathbf{e}^T)^T \pmod{q}$.
  - Liu and Nguyen's enumeration [6]: choose σ_E to be large.
- The forgery security depends on a SIS problem where the short vector has entries bounded by $\max(2B, 2^{d-1}) + 2E'w$. We can evaluate the security using the BKZ 2.0.

Conclusion

We have described a new approach for compressing the lattice-based signatures. The new signature scheme, together with the compression, is based on the standard average-case hardness of LWE and SIS (and hence worse-case hardness of SVP/SIVP) in general lattices.

Some further considerations:

- LWE with small secrets and/or non-standard LWE;
- Sampling vectors $\mathbf{y}$ from Gaussian (or bimodal Gaussian [3]);
- Signatures based on ring-variants.

Reference

[1] S. Bai and S. D. Galbraith, Lattice Decoding Attacks on Binary LWE, IACR Cryptology ePrint Archive 2013: 839 (2013).

[2] Y. Chen and P. Q. Nguyen, BKZ 2.0: Better Lattice Security Estimates, in D. H. Lee and X. Wang (eds.), ASIACRYPT 2011, Springer LNCS 7073 (2011) 1–20.

[3] L. Ducas, A. Durmus, T. Lepoint and V. Lyubashevsky, Lattice Signatures and Bimodal Gaussians, in R. Canetti and J. A. Garay (eds.), CRYPTO 2013, Springer LNCS 8042 (2013) 40–56.

[4] C. Gentry, C. Peikert and V. Vaikuntanathan, Trapdoors for Hard Lattices and New Cryptographic Constructions, in C. Dwork (ed.), STOC 2008, ACM (2008) 197-206.

[5] T. Güneysu, V. Lyubashevsky and T. Pöppelmann, Practical Lattice-Based Cryptography: A Signature Scheme for Embedded Systems, in E. Prouff and P. Schaumont (eds.), CHES 2012, Springer LNCS 7428 (2012) 530–547.

[6] M. Liu and P. Q. Nguyen, Solving BDD by Enumeration, An Update, in E. Dawson (ed.), CT-RSA 2013, Springer LNCS 7779 (2013) 293–309.

[7] V. Lyubashevsky, Fiat-Shamir with Aborts: Applications to Lattice and Factoring-Based Signatures, in M. Matsui (ed.), ASIACRYPT 2009, Springer LNCS 5912 (2009) 598–616.

[8] V. Lyubashevsky, Lattice Signatures without Trapdoors, in D. Pointcheval and T. Johansson (eds.), EUROCRYPT 2012, Springer LNCS 7237 (2012) 738–755.

[9] D. Stehlé and R. Steinfeld, Making NTRUEncrypt and NTRUSign as Secure as Standard Worst-Case Problems over Ideal Lattices, Cryptology ePrint Archive: Report 2013/004.

Thank you for your attention.

Questions and comments?

RSA CONFERENCE 2014
FEBRUARY 24 – 28 | MOSCONE CENTER | SAN FRANCISCO

Share.
Learn.
Secure.
Capitalizing on
Collective Intelligence

Efficient and Secure Algorithms for GLV-Based Scalar Multiplication and Their Implementation on GLV-GLS Curves

SESSION ID: CRYP-T07

Patrick Longa

Microsoft Research
http://research.microsoft.com/en-us/people/plonga/
@PatrickLonga

Joint work with: Armando Faz-Hernández (UNICAMP, Brazil)
Ana H. Sánchez (CINVESTAV-IPN, México)

Basics on ECC Scalar Multiplication

Basics on Elliptic Curve Scalar Multiplication

Let an elliptic curve $E\colon\ y^2 = x^3 + ax + b$ be defined over the prime field $\mathbb{F}_p$, such that $\#E = h.r$ with small co-factor h and large prime order r.

- The central operation in ECC, known as scalar multiplication, consists on computing the multiple $[k]P$ of a point $P \in E(\mathbb{F}_p)$, given an integer $k \in [1, r)$.
- Naïvely, $[k]P = P + P + \cdots + P$ (k times).

Basics on Elliptic Curve Scalar Multiplication

Assume that the point P is unknown before the computation.

- $[k]P$ can be computed using a (signed) binary representation, e.g., non-adjacent form (NAF):

$$k = (k_l, \dots, k_0)_{\mathrm{NAF}} \text{ , where } l = \lfloor \log_2(k) \rfloor \text{ and } k_i \in \{0, \pm 1\}.$$

- Then, one applies a double-and-add algorithm.

Basics on Elliptic Curve Scalar Multiplication

Given $k = (k_l, \ldots, k_0)_{\text{NAF}}$ and point $P \in E(\mathbb{F}_p)$	
1.	$Q = \mathsf{D}$
2.	for $i = l$ downto 0 do
3.	$Q = [2]Q$
5.	if $k_i \neq 0$, then $Q = Q + s_i P$ $\{s_i$ is the sign of $k_i\}$
7.	end for
8.	return (Q)

- The cost is given by $(l + 1)$ point doublings and, in average, $(l + 1)/3$ point additions.
- Extending the use of windowing reduces the number of additions to $\left(\frac{l+1}{w+1}\right)$ with $w \geq 2$.
- BUT, the conditional execution makes it vulnerable to timing attacks (and others).

Constant-time Elliptic Curve Scalar Multiplication

Given $k = (k_t, \dots, k_0)_{fixed-w}$ and point table $P[j] = \{1,3, \dots, 2^{w-1} - 1\}P$		
1.	$Q = P[(\lvert k_t \rvert - 1)/2)]$	
2.	for $i = (t-1)$ downto 0 do	
3.	$\quad Q = [2^{w-1}]Q$	
5.	$\quad Q = Q + s_i P[(\lvert k_i \rvert - 1)/2)]$	$\{s_i$ is the sign of $k_i\}$
7.	end for	
8.	return (Q)	

Using the fixed-window method [Okeya and Takagi, CT-RSA 2003]:

- Represent odd scalar k with a fixed length representation $(k_t, \dots, k_0)_{fixed-w}$, where $t = \left\lceil \frac{\log_2(r)}{w-1} \right\rceil$ and $k_i \in \{\pm 1, \pm 3, \dots, \pm(2^{w-1} - 1)\}$.
- The cost is given by $t \cdot (w-1)$ point doublings and t point additions (plus precomputation).

GLV-Based Scalar Multiplication

GLV-Based Elliptic Curve Scalar Multiplication

Given a point $P \in E(\mathbb{F}_p)$, an integer $k \in [1, r)$ and an efficiently computable endomorphism φ, the Gallant-Lambert-Vanstone (GLV) method computes

$$[k]P = [k_0]P + [k_1]\varphi(P),$$

where $\max(|k_0|, |k_1|) = \mathrm{O}(\sqrt{r})$.

- Using simultaneous multi-scalar multiplication, the number of doublings is cut to half. E.g., it costs roughly $(l+1)/2$ point doublings and $(l+1)/3$ point additions when using NAF.

GLV-Based Elliptic Curve Scalar Multiplication

- φ is a nontrivial endomorphism defined over $\mathbb{F}_p$ with characteristic polynomial $X^2 + uX + v$, where $\Delta = u^2 - 4v < 0$.
- $\varphi(P) = \&P$, where $\& \in [1, r-1]$ is a root of the char polynomial of φ modulo r.
- By solving a closest vector problem in a lattice, one can get values k_0, k_1 such that $k = k_0 + k_1 \& \pmod r$, or equivalently, $[k]P = [k_0]P + [k_1]\varphi(P)$.
- Recent advances extend GLV from *two* dimensions to *four* when working over a quadratic extension field $\mathbb{F}_{p^2}$ (this is discussed later).

Constant-time GLV Scalar Multiplication (first attempt)

(m-dimension GLV) Given m scalars $k_i = (k_{i,t}, \dots, k_{i,0})_{fixed-w}$ and point table $P[i][j] = \{1,3, \dots, 2^{w-1}-1\}P[i]$, for m base points $P[i]$ and $j \in \{0,1, \dots, 2^{w-2}-1\}$

1. $Q = \sum_i P[i][(|k_{i,t}|-1)/2)]$
2. for $j = (t-1)$ downto 0 do
3. $\quad Q = [2^{w-1}]Q$
4. $\quad Q = Q + \sum_i s_{i,j} P[i][(|k_{i,j}|-1)/2)]$ $\quad \{s_{i,j}$ is the sign of $k_{i,j}\}$
5. end for
6. return (Q)

Using the fixed-window method:

- Represent odd scalars k_i with a fixed length representation $(k_{i,t}, \dots, k_{i,0})_{fixed-w}$, where $t = \left\lceil \frac{\log_2(r)}{m \cdot (w-1)} \right\rceil$ and $k_{i,j} \in \{\pm 1, \pm 3, \dots, \pm(2^{w-1}-1)\}$.
- The cost is given by $t \cdot (w-1)$ point doublings and t point additions.
- Computing the m tables $P[i][j]$ costs m doublings and $m \cdot (2^{w-2}-1)$ additions.

GLV-SAC Representation

Least Significant Bit - Set (LSB-set) representation

Feng, Zhu, Xu and Li, 2005:

- Partition an odd scalar k in w consecutive parts of $d = \left\lceil \frac{\log_2(r)}{w} \right\rceil$ bits each, padding with $(dw - t)$ zeroes to the left.
- Recode first d bits to signed nonzero digits b_i using $1 \rightarrow 1\bar{1}\bar{1}\ldots\bar{1}$ $(\bar{1} = -1)$.
- Recode remaining bits b_i such that $b_i \in \{0, b_{i \bmod d}\}$.
- Feng et al. exploits this representation for computing $[k]P$ with P fixed using comb methods.

Adapting LSB-set to the GLV setting: GLV Signed-Aligned Column (GLV-SAC) representation

Given m scalars k_j for m-GLV scalar multiplication and $l = \left\lceil \frac{\log_2(r)}{m} \right\rceil + 1$:

- Pad each k_j with zeroes to the left such that each one has bit-length l.
- Take one $k_J \subset k_j$, convert it to odd and recode it to signed nonzero digits b_i using $1 \rightarrow 1\bar{1}\bar{1}\ldots\bar{1}$:

$$k_J = (b_{l-1}^J, \ldots, b_0^J), \text{ where } b_i^J \in \{\pm 1\}.$$

- Recode remaining scalars k_j such that $b_i^j \in \{0, b_i^J\}$.

GLV-Based Scalar Multiplication using GLV-SAC

(m-dimension GLV) Given m scalars such that: k_0 is recoded as $(b_{l-1}^0, \ldots, b_0^0)$, where $b_i^0 \in \{\pm 1\}$ and remaining k_j are recoded as $(b_{l-1}^j, \ldots, b_0^j)$, where $b_i^j \in \{0, b_i^0\}$.

1. Precompute $P[u] = P_0 + u_0 P_1 + \cdots + u_{m-2} P_{m-1}$ for all $0 \le u < 2^{m-1}$, where $u = (u_{m-2}, \ldots, u_0)_2$
2. $Q = s_{l-1} P[K_{l-1}]$ $\{K_i = b_i^1 + b_i^2 \cdot 2 + \cdots + b_i^{m-1} \cdot 2^{m-2}\}$
3. for $i = (l-2)$ downto 0 do
4. $\quad Q = [2]Q$
5. $\quad Q = Q + s_i P[K_i]$ $\{s_i$ is the sign of $b_i^0\}$
6. end for
7. return (Q)

- The main loop costs $(l-1) = \left\lceil \frac{\log_2(r)}{m} \right\rceil$ point doublings and $(l-1)$ point additions.
- Computing the table $P[u]$ costs $(2^{m-1} - 1)$ additions.

GLV-Based Scalar Multiplication using GLV-SAC

- Example: let $m = 3$, $\log_2(r) = 9$ and $[k]P = 11P_0 + 2P_1 + 5P_2$. Then $l = \left\lceil \frac{9}{3} \right\rceil + 1 = 4$, and the GLV-SAC representation is given by:

$$\begin{bmatrix} k_0 \\ k_1 \\ k_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & \bar{1} & 1 \\ 0 & 1 & \bar{1} & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

- Precomputed values are: $P[0] = P_0,\ P[1] = P_0 + P_1,\ P[2] = P_0 + P_2,\ P[3] = P_0 + P_1 + P_2$.
- Computation: $2P_0 + (P_0 + P_1 + P_2) \rightarrow 2(3P_0 + P_1 + P_2) - (P_0 + P_1) \rightarrow 2(5P_0 + P_1 + 2P_2) + (P_0 + P_2) = 11P_0 + 2P_1 + 5P_2$.

GLV-Based Scalar Multiplication using GLV-SAC

- Total cost using fixed-window: $(l + m - 1)$ point doublings and $m \cdot \frac{l-1}{w-1} + (2^m - 1) + m \cdot (2^{w-2} - 1)$ point additions, using $m \cdot (2^{w-2} + 1)$ points.
- Total cost of the new method: $(l - 1)$ point doublings and $(l + 2^{m-1} - 1)$ point additions, using 2^{m-1} points.
- E.g., $r = 256, m = 4, w = 5$ (typical parameters for 128-bit security) :
  - Fixed-window: 68 doublings and 99 additions using 36 points
  - New method: 64 doublings and 72 additions using 8 points

20% speedup using only ~1/5 of storage (assuming one addition = 1.3 doubling)

Implementation on GLV-GLS Curves

Selected Curve

Longa and Sica, ASIACRYPT 2012:

- GLV-GLS curve in Twisted Edwards form $E'/\mathbb{F}_{p^2}: -x^2 + y^2 = 1 + dx^2y^2$, where $p = 2^{127} - 5997$, $\#E'(\mathbb{F}_p) = 8r$, where r is a 251-bit prime, with $d = 170141183460469231731687303715884099728 + 116829086847165810221872975542241037773i$.
- This curve supports a 4-GLV decomposition:

$$[k]P = [k_0]P + [k_1]\Phi(P) + [k_2]\Psi(P) + [k_3]\Psi\Phi(P),$$

where $\max_i(|k_i|) < 179\, n^{1/4}$.

Efficient Implementation on ARM: Interleaving ARM and NEON instructions over GF(p^2)

- Strategy: interleave independent NEON-based and ARM-based integer operations and reductions to exploit instruction level parallelism (ILP).
- An example with multiplication over $\mathbb{F}_{p^2}$: $C = (a_0 + ia_1) \times (b_0 + ib_1)$

$$C_0 = a_0 \times b_0 - a_1 \times b_1,\ C_1 = (a_0 + a_1)(b_0 + b_1) - a_0 \times b_0 - a_1 \times b_1.$$

➔ Independent integer multiplies $a_0 \times b_0$, $a_1 \times b_1$ and $(a_0 + a_1)(b_0 + b_1)$ can be computed in “parallel”.

Efficient Implementation on ARM: Interleaving ARM and NEON instructions over GF(p^2)

- Over $\mathbb{F}_{p^2}$ we designed:
  - A double integer multiply: one NEON-based, one ARM-based
  - A triple integer multiply: two NEON-based, one ARM-based
  - A double reduction: one NEON-based, one ARM-based

Efficient Implementation on ARM: Interleaving ARM and NEON instructions over GF(p^2)

- Triple 128-bit integer multiplication with ARM/NEON interleaving:

$a = \{a_i\},\ b = \{b_i\},\ c = \{c_i\},\ d = \{d_i\},\ e = \{e_i\},\ f = \{f_i\}$, for $i \in \{0,1,2,3\}$	
1. $(F, G, H) = (0, 0, 0)$	
2. for $i = 0$ downto 3 do	
3. $(C_0, C_1, C_2) = (0, 0, 0)$	
4. for $j = 0$ downto 3 do	
5. $(C_0, F_{i+j}, C_1, G_{i+j}) = (F_{i+j} + a_j b_i + C_0, G_{i+j} + c_j d_i + C_1)$	{done by NEON}
6. for $j = 0$ downto 3 do	
7. $(C_2, H_{i+j}) = H_{i+j} + e_j f_i + C_2$	{done by ARM}
8. $(F_{i+4}, G_{i+4}, H_{i+4}) = (C_0, C_1, C_2)$	
9. return $(F, G, H) = (a \times b, c \times d, e \times f)$	

Experimental Results

Comparison of Constant-Time Implementations

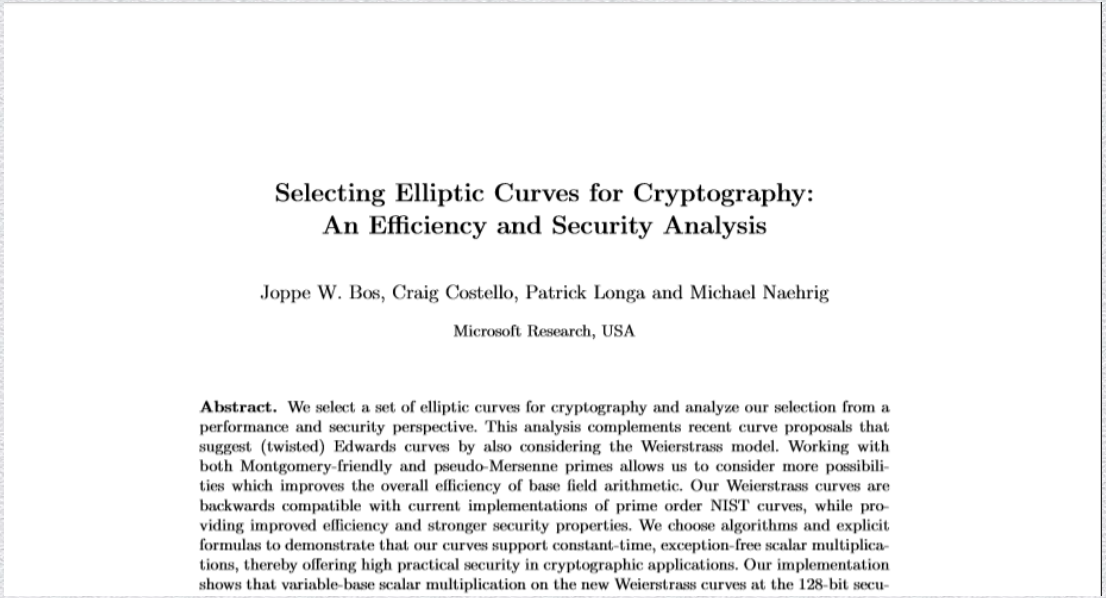
Curve	ARM Cortex-A9	ARM Cortex-A15	Intel Sandy Bridge	Intel Ivy Bridge
TEdwards ($\mathbb{F}_{p^2}$), 4-GLV (this work)	417,000cc	244,000cc	96,000cc	92,000cc
TEdwards ($\mathbb{F}_{p^2}$), 4-GLV, Longa-Sica 2012	-	-	137,000cc	-
Binary GLS ($\mathbb{F}_{2^{254}}$), Olivera *et al.* 2013	-	-	115,000cc	113,000cc
Genus 2 Kummer ($\mathbb{F}_p$), Bos *et al.* 2013	-	-	126,000cc	117,000cc
Curve25519 ($\mathbb{F}_p$), Bernstein *et al.* 2011	-	-	194,000cc	183,000cc
Curve25519 ($\mathbb{F}_p$), Bernstein *et al.* 2012	568,000cc	-	-	-
Montgomery ($\mathbb{F}_p$), Hamburg 2012	616,000cc	-	153,000cc	-

Related work I

- Extended paper version: http://eprint.iacr.org/2013/158
  - Covers (side-channel protected) fixed-base scalar multiplication and double scalar multiplication (for signature verification)

Related work II

- New elliptic curves for cryptography, including rigorous analysis from an efficiency and security perspective: http://eprint.iacr.org/2014/130

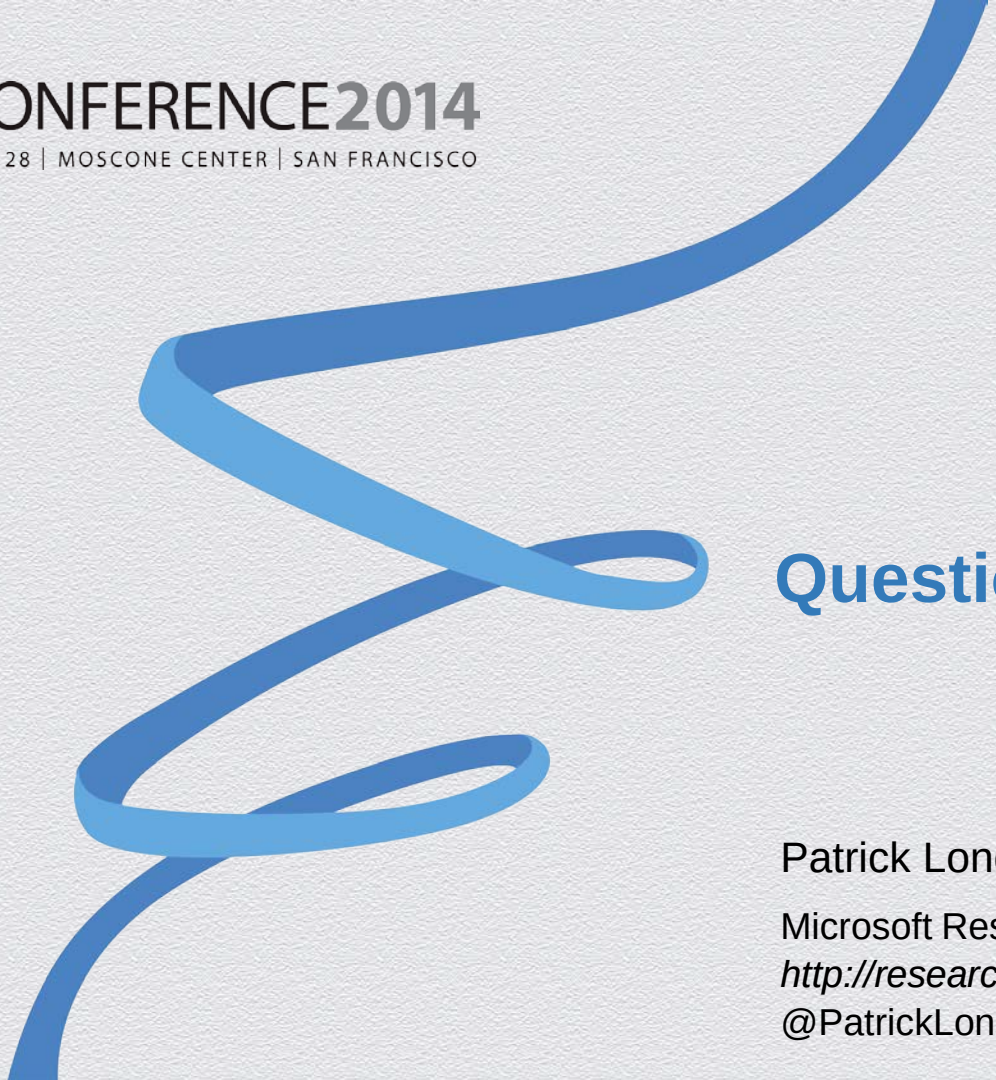


Selecting Elliptic Curves for Cryptography: An Efficiency and Security Analysis

Joppe W. Bos, Craig Costello, Patrick Longa and Michael Naehrig

Microsoft Research, USA

Abstract. We select a set of elliptic curves for cryptography and analyze our selection from a performance and security perspective. This analysis complements recent curve proposals that suggest (twisted) Edwards curves by also considering the Weierstrass model. Working with both Montgomery-friendly and pseudo-Mersenne primes allows us to consider more possibilities which improves the overall efficiency of base field arithmetic. Our Weierstrass curves are backwards compatible with current implementations of prime order NIST curves, while providing improved efficiency and stronger security properties. We choose algorithms and explicit formulas to demonstrate that our curves support constant-time, exception-free scalar multiplications, thereby offering high practical security in cryptographic applications. Our implementation shows that variable-base scalar multiplication on the new Weierstrass curves at the 128-bit secu-



Questions?

Patrick Longa

Microsoft Research
http://research.microsoft.com/en-us/people/plonga/
@PatrickLonga