Elliptic Curves a Hardware Perspective

Joppe W. Bos

NIST Workshop on Elliptic Curve Cryptography Standards

June 11- June 12 2015, Gaithersburg, MD, USA

NXP | SECURE CONNECTIONS FOR A SMARTER WORLD

Motivation

We all want **fast**, **high security, affordable and easy-to-use** elliptic curves for cryptography.

- ❑ How to choose them? (Does a truly rigid curve selection even exist?)
- ❑ Do we need different curves for different applications due to different security models?

This talk: **A** hardware perspective on selecting cryptographic elliptic curves

Motivation

We all want **fast**, **high security, affordable and easy-to-use** elliptic curves for cryptography.

- ❑ How to choose them? (Does a truly rigid curve selection even exist?)
- ❑ Do we need different curves for different applications due to different security models?

This talk: **A** hardware perspective on selecting cryptographic elliptic curves

I try to comment on these often heard phrases about hardware implementations

- Why should we care about hardware considerations?
- Hardware implementations just communicate to each other in a closed environment.
- There is much more usage of ECC in software than hardware,
  so software requirements are much more important!
- If the new curves are fast in software they are also fast in hardware, right?

Elliptic Curves in Cryptography

1985-1987
- Koblitz and Miller: **elliptic curves in cryptography**

2000
- Certicom: First curve standard **Standards for Efficient Cryptography**
- NIST: FIPS 186-2 **Digital Signature Standard**

2005
- ECC Brainpool: **Standard Curves and Curve Generation**

2006
- D. J. Bernstein: **Curve25519 (128-bit security only)**

2013
- New York Times (related to Dual EC-DRBG):
  "*the National Security Agency had written the standard and could break it*"

Elliptic Curves and Hardware

We see an increase in support for ECC in software, for example

- 2013 scan observed: “about 1 in 10 systems support ECC across the TLS and SSH protocols”
- Around **5 million hosts** support ECC in TLS / SSH
- Many TLS servers prefer ciphersuites with ECDHE

Elliptic Curves and Hardware

We see an increase in support for ECC in software, for example

- 2013 scan observed: “about 1 in 10 systems support ECC across the TLS and SSH protocols”
- Around **5 million hosts** support ECC in TLS / SSH
- Many TLS servers prefer ciphersuites with ECDHE

Hardware ECC

- ✓ Currently, ECC coprocessors are used
  - ✓ in **billions** of smart cards securing ID cards, passports and banking
  - ✓ for 15 years in devices supporting the Digital Transmission Content Protection system

(Short-term) future: Internet-of-Things, prediction

- ✓ 5 billion things at the end of 2015
- ✓ 25 billion things around 2020

- For asymmetric crypto, ECC is the logical choice: small keys, fast on embedded platforms, etc
- Many “things” need to communicate securely with user-apps and possibly the world wide web
- Hardware and software implementation will start to talk to each other (more frequently)!

Software and Hardware Perspective

In both environments we want efficient and secure implementations!

However, the settings are quite different:

- **Implementation strategy / algorithm selection**
  - Software optimizations mainly focus on improved performance (performance, performance, performance!)
  - In hardware: size matters (area size, number of registers, memory requirement)

Software and Hardware Perspective

In both environments we want efficient and secure implementations!

However, the settings are quite different:

- **Implementation strategy / algorithm selection**
  - Software optimizations mainly focus on improved performance (performance, performance, performance!)
  - In hardware: size matters (area size, number of registers, memory requirement)
- **Maintainability**
  - Patching / upgrading deployed software is relatively cheap and easy (but still a pain!)
  - Patching / upgrading deployed hardware is expensive in terms of effort and money

Software and Hardware Perspective

In both environments we want efficient and secure implementations!

However, the settings are quite different:

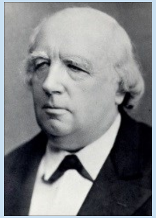
- **Implementation strategy / algorithm selection**
  - Software optimizations mainly focus on improved performance (performance, performance, performance!)
  - In hardware: size matters (area size, number of registers, memory requirement)
- **Maintainability**
  - Patching / upgrading deployed software is relatively cheap and easy (but still a pain!)
  - Patching / upgrading deployed hardware is expensive in terms of effort and money
- **Security model**
  - Software security model: susceptible to mainly *timing attacks and cache attacks*
  - Hardware security model: susceptible to *fault injections, simple power analysis, differential power analysis, correlation power analysis, template attacks, higher-order correlation attacks, mutual information analysis, linear regression analysis, horizontal analysis, vertical analysis etc.*

Elliptic Curves

Weierstrass curves

$$y^2 = x^3 + ax + b$$

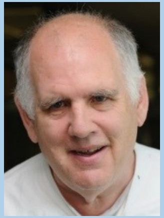
- Most general form
- [+] Prime order possible
- [-] Exceptions in group law
- NIST and Brainpool curves



Montgomery curves

$$By^2 = x^3 + Ax^2 + x$$

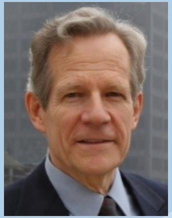
- Subset of curves
- [-] Not prime order
- [+] Montgomery ladder



Twisted Edwards curves

$$ax^2 + y^2 = 1 + dx^2y^2$$

- Subset of curves
- [-] Not prime order
- [+] Fastest arithmetic
- [+] Some have complete group law



Backwards compatibility

Implementing arithmetic on (short) Weierstrass curves makes a lot of sense.
Given a curve in another curve model one can always translate this to an equivalent Weierstrass curve

"One curve model to rule them all"

- Implement group law, counter measures etc. once.
- If new curves are proposed no need to change implementation.

Backwards compatibility

Implementing arithmetic on (short) Weierstrass curves makes a lot of sense.
Given a curve in another curve model one can always translate this to an equivalent Weierstrass curve

"One curve model to rule them all"

- Implement group law, counter measures etc. once.
- If new curves are proposed no need to change implementation.

Existing hardware / software implementations might assume

- prime order `[ almost always assumed ]`
- short Weierstrass curves `[ always assumed ]`
- with curve parameter $a = -3$ `[ not widely assumed? ]`

Backwards compatibility

Implementing arithmetic on (short) Weierstrass curves makes a lot of sense.
Given a curve in another curve model one can always translate this to an equivalent Weierstrass curve

"One curve model to rule them all"

- Implement group law, counter measures etc. once.
- If new curves are proposed no need to change implementation.

Existing hardware / software implementations might assume

- prime order [almost always assumed]
- short Weierstrass curves [always assumed]
- with curve parameter $a = -3$ [not widely assumed?]

Historically this makes sense:
Standard curves $E(\mathbf{F}_p)$ with p>3 prime have these three properties
For instance see:

- NIST, FIPS 186-4, App. D: Recommended Elliptic Curves for Government Use
- SEC 2: Recommended Elliptic Curve Domain Parameters*

(* Except the three Koblitz curves secp192k1, secp224k1, secp256k1, where $a = 0$)

Backwards compatibility

Existing hardware / software implementations might assume

- prime order [almost always assumed]
  - This rules out (twisted) Edwards / Montgomery curves
  - Need additional code to avoid small-subgroup attacks
- short Weierstrass curves [always assumed]
  One curve model to rule them all: not a problem
- with curve parameter $a = -3$ [not widely assumed?]

Backwards compatibility

Existing hardware / software implementations might assume

- prime order   [almost always assumed]
  - ❖ This rules out (twisted) Edwards / Montgomery curves
  - ❖ Need additional code to avoid small-subgroup attacks

- short Weierstrass curves   [always assumed]

  One curve model to rule them all: not a problem

- with curve parameter $a = -3$ [not widely assumed?]

One can transform

$$y^2 = x^3 + ax + b \qquad \text{to an isomorphic} \qquad y^2 = x^3 - 3x + b'$$

if and only if there exists $u \in \mathbf{F}_p^*$ such that $u^4 = a/-3$ and $u^6 = b/b'$

Backwards compatibility

Existing hardware / software implementations might assume

- prime order [almost always assumed]
  - This rules out (twisted) Edwards / Montgomery curves
  - Need additional code to avoid small-subgroup attacks
- short Weierstrass curves [always assumed]
  One curve model to rule them all: not a problem
- with curve parameter $a = -3$ [not widely assumed?]

One can transform

$$y^2 = x^3 + ax + b \quad \text{to an isomorphic} \quad y^2 = x^3 - 3x + b'$$

if and only if there exists $u \in \mathbf{F}_p^*$ such that $u^4 = a/{-3}$ and $u^6 = b/b'$

Example: Such a u does not exist for curve25519 → no isomorphic $a = -3$ short Weierstrass curve.

Have to use isogenies instead:

- more complexity
- what is the degree of this isogeny?

Backwards compatibility

Existing hardware / software implementations might assume

- prime order `[ almost always assumed ]`
  - ❖ This rules out (twisted) Edwards / Montgomery curves
  - ❖ Need additional code to avoid small-subgroup attacks
- short Weierstrass curves `[ always assumed ]`
  One curve model to rule them all: not a problem
- with curve parameter $a = -3$ `[ not widely assumed? ]`

Example: Such a u does not exist for curve25519 → no isomorphic $a = -3$ short Weierstrass curve.

Have to use isogenies instead:

- more complexity
- what is the degree of this isogeny?

One can transform

$$y^2 = x^3 + ax + b \quad \text{to an isomorphic} \quad y^2 = x^3 - 3x + b'$$

if and only if there exists $u \in \mathbf{F}_p^*$ such that $u^4 = a/{-3}$ and $u^6 = b/b'$

Preference: prime-order curves

Side Channel Attacks I

Assumption
When executing a cryptographic operation on a particular hardware device, the power consumption at a certain state depends on the (secret) data involved and some random noise

Simple power analysis: deduce the secret key by visual examination of the graph of the current over time (a large family of software timing attacks can be seen as SPA)

Correlation power analysis: correlate the power consumption to the bits of the secret key

Side Channel Attacks I

Assumption
When executing a cryptographic operation on a particular hardware device, the power consumption at a certain state depends on the (secret) data involved and some random noise

Simple power analysis: deduce the secret key by visual examination of the graph of the current over time (a large family of software timing attacks can be seen as SPA)

Correlation power analysis: correlate the power consumption to the bits of the secret key

Setting ECDH, well-known countermeasure: randomize input point

1) **Use isomorphic curve**

$$y^2 = x^3 + ax + b \quad \rightarrow \quad y^2 = x^3 + au^4x + bu^6$$
$$(x, y) \quad \rightarrow \quad (u^2x, u^3y)$$

2) **Use projective coordinates**

For example, Jacobian coordinates, use non-zero r such that

$$(X: Y: Z) \rightarrow (r^2X: r^3Y: rZ)$$

Side Channel Attacks II

However, Goubin's attack (zero-coordinate) + [Akishita, Takagi]'s attack (zero-value) apply

Idea, focus on points with a zero coordinate

Weierstrass	Twisted Edwards
$(x, 0)$, point of order 2	$(0,1)$, 1-torsion
$(0, \pm\sqrt{b})$	$(0, -1)$, 2-torsion
	$(\pm\sqrt{a^{-1}}, 0)$, 4-torsion

Side Channel Attacks II

However, Goubin's attack (zero-coordinate) + [Akishita, Takagi]'s attack (zero-value) apply

Idea, focus on points with a zero coordinate

Weierstrass	Twisted Edwards
$(x, 0)$, point of order 2	$(0,1)$, 1-torsion
$(0, \pm\sqrt{b})$	$(0, -1)$, 2-torsion
	$(\pm\sqrt{a^{-1}}, 0)$, 4-torsion

Preference

When proposing new curves take already known side-channel attacks and weaknesses into consideration

Side Channel Attacks II

However, Goubin's attack (zero-coordinate) + [Akishita, Takagi]'s attack (zero-value) apply

Idea, focus on points with a zero coordinate

Weierstrass	Twisted Edwards
$(x, 0)$, point of order 2	$(0,1)$, 1-torsion
$(0, \pm\sqrt{b})$	$(0, -1)$, 2-torsion
	$(\pm\sqrt{a^{-1}}, 0)$, 4-torsion

Preference

When proposing new curves take already known side-channel attacks and weaknesses into consideration

Example: curve25519 can be written as the Weierstrass curve

y^2 = x^3 - 236839902241/3 x + 230521961007359098/27

$(0, \pm\sqrt{230521961007359098/27})$ is a valid point and has full order

Side Channel Attacks: Special Primes

These attack ideas carry over to the modular multiplication level as well.

- Typical hardware approach:
  generic hardware multiplier + generic modular reduction

- Typical software approach:
  specialized reduction routine tailored for a specific “special” prime (performance)

Example of special primes:

$$p255 = 2^{255} - 19$$
$$p256 = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$$
$$p521 = 2^{521} - 1$$

❖ Specialized hardware reduction routines → more gates

❖ Not uncommon to have special hardware for multiplication (re-usage for other components)
→ integer multiplication-only hardware routines amplify zero-value attacks on the finite-field layer.

Side Channel Attacks: Special Primes

Other popular countermeasure: *additive scalar blinding*

Idea

Add a *small* random multiple of the group order n to the scalar d

$$d' = d + r \cdot n$$

- Problematic with such special primes since the Hasse bound states that

$$|\#E(\mathbf{F}_p) - (p+1)| \leq 2\sqrt{p}$$

Side Channel Attacks: Special Primes

Other popular countermeasure: *additive scalar blinding*

Idea
Add a *small* random multiple of the group order n to the scalar d

$$d' = d + r \cdot n$$

- Problematic with such special primes since the Hasse bound states that

$$| \#E(\mathbf{F}_p) - (p+1) | \leq 2\sqrt{p}$$

Example: Curve25519
Prime subgroup order $n = 2^{252} + c$ such that $\mathbf{2^{252} + 2^{124} < n < 2^{252} + 2^{125}}$.
If $r < 2^{32}$ then the least significant $125 + 32 = 157$ bits are blinded,
the 95 most significant bits of d can be directly extracted from d'

Side Channel Attacks: Special Primes

Other popular countermeasure: *additive scalar blinding*

Idea
Add a *small* random multiple of the group order n to the scalar d

$$d' = d + r \cdot n$$

- Problematic with such special primes since the Hasse bound states that

$$|\#E(\mathbf{F}_p) - (p+1)| \leq 2\sqrt{p}$$

Example: Curve25519
Prime subgroup order $n = 2^{252} + c$ such that $\mathbf{2^{252} + 2^{124} < n < 2^{252} + 2^{125}}$.
If $r < 2^{32}$ then the least significant $125 + 32 = 157$ bits are blinded,
the 95 most significant bits of d can be directly extracted from d'

- Usage of special primes <u>reduces</u> the number of available countermeasure techniques

Preference: <u>Use randomly generated primes</u>

Conclusions

Some current curve proposals suggest to use

- sometimes the Montgomery curve for ECDH
- twisted Edwards for ECDSA

➢ Adding HW support in the near future for Montgomery and (twisted) Edwards curves is not realistic

➢ Supporting non-prime order curves (Montgomery / (twisted) Edwards) in their Weierstrass form requires adding code complexity to avoid small-subgroup attacks

✓ There are billions of HW devices and in the future billions of more “things” that will support ECC

✓ These implementations (will) interact with software implementations, user-apps and the world-wide-web

✓ We should select curves which make it easier to be secure in this security model

Conclusions

Both in software and in hardware we want efficient and secure implementations!

Our preferences when selecting new elliptic curves for cryptography
(from a HW perspective)

1) prime-order curves
2) take already known side-channel attacks and weaknesses into consideration
3) use randomly generated primes (but how to generate these primes?[1])
4) twist security (nice feature to have)

[1] *A. K. Lenstra and B. Wesolowski:* A random zoo: sloth, unicorn, and trx. **Cryptology ePrint Archive: Report 2015/366**

SECURE CONNECTIONS
FOR A SMARTER WORLD