Developing an automatic generation tool for cryptographic pairing functions

Luis J. Dominguez Perez

Cinvestav. Departamento de Computación.
ldominguez.computing.dcu.ie.

"Seminario de Computación"
Mexico City.

February 28^{th} 2011.

CONACYT
Consejo Nacional de Ciencia y Tecnología

Claude Shannon Institute
Discrete Mathematics, Coding, Cryptography and Information Security
www.shannoninstitute.ie

DCU

Cinvestav

Outline

Cinvestav

Group

A group $< G, \circ >$ is a non-empty set G together with a binary operation $\circ$ such that:

- *It is closed*
- *it is associative*
- *has an identity and inverse element*

A group G is **Abelian** (or **commutative**) if $a \circ b = b \circ a$, $\forall a, b \in G$.

A group is **finite** if G has a finite number of elements. This is called the **order** of G and denoted as $|G|$.

Cinvestav

Finite field

A field F is a group with $+$, $\times$ operations as $(F, +)$ and $(F \backslash \{0\}, \times)$ which also satisfies:

- *Additive identity and inverse*
- *Multiplicative identity and inverse*
- *Commutative*

i.e. the set of integers modulo p-prime, also denoted as $\mathbb{F}_p$, is a finite field.

Cinvestav

Elliptic curves over finite fields

Let p-prime > 3. The elliptic curve

$$y^2 = x^3 + ax + b, \qquad \text{over } \mathbb{F}_p$$

denoted by $E(\mathbb{F}_p)$, is the set of solutions $x, y \in \mathbb{F}_p$ satisfying

$$y^2 \equiv x^3 + ax + b \bmod p$$

where $a, b \in \mathbb{F}_p$ and

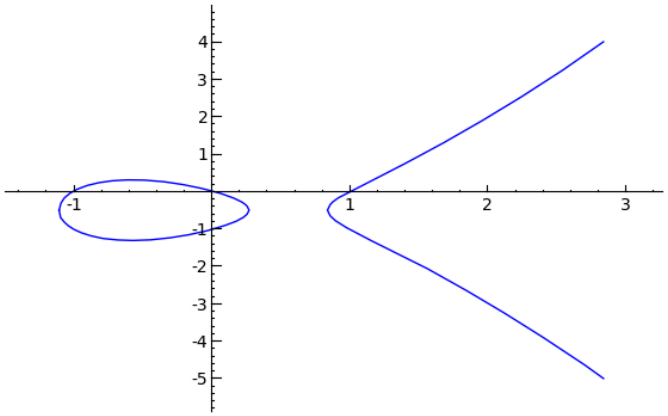
$$4a^3 + 27b^2 \neq 0 \bmod p$$

together with the point $\mathcal{O}$

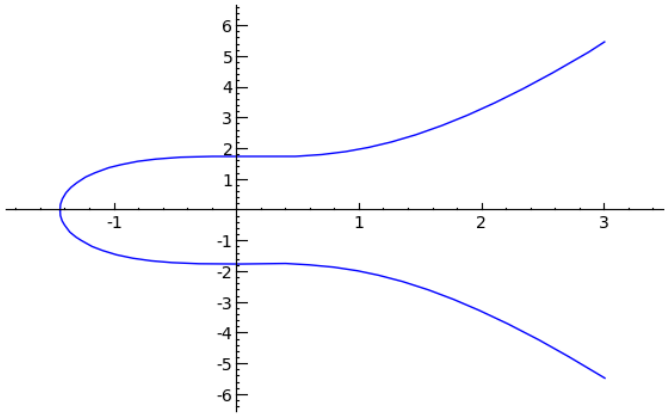
The **order**, or number of points on $E(\mathbb{F}_p)$ is denoted as $\#E(\mathbb{F}_p) = p + 1 \pm t$ and $t \leq 2\sqrt{p}$.

Cinvestav

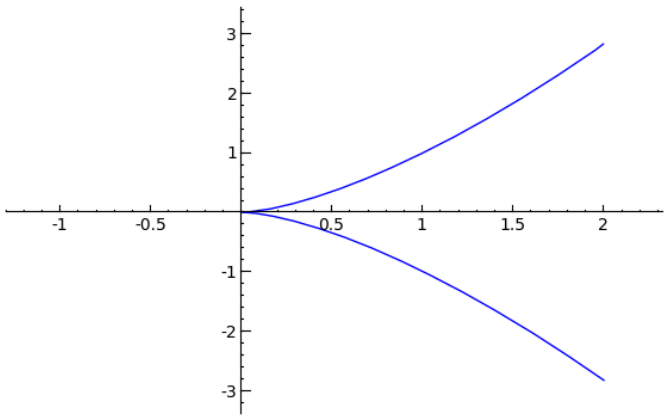
Types of elliptic curves (over $\mathbb{C}$)



With 3 distinct real roots



With 1 real and 2 complex roots

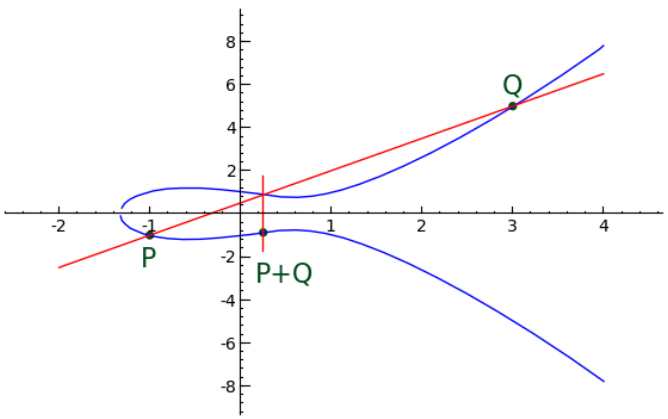


With a triple real root

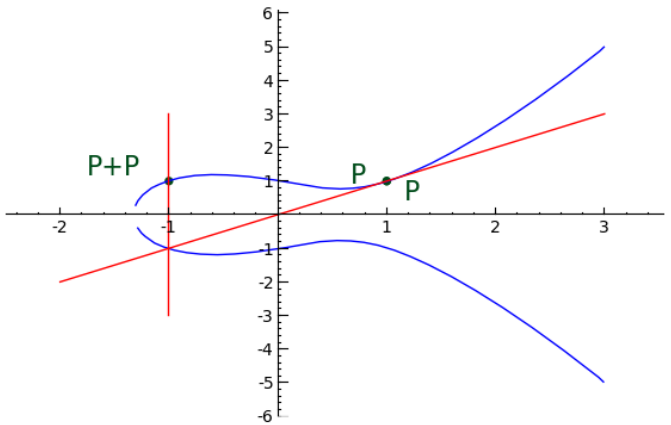
Cinvestav

The group law on EC

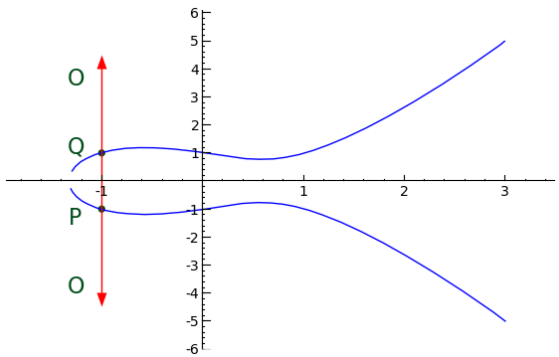
Suppose $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ with $P, Q \in E(\mathbb{F}_p)$ $P + Q = (x_3, y_3)$, where $x_3 = \lambda^2 - x_1 - x_2, y_3 = \lambda(x_1 - x_3) - y_1$



$$\lambda = \frac{y_2 - y_1}{x_2 - x_1}$$



$$\lambda = \frac{3x_1^2 + a}{2y_1}$$



$$P + Q = \mathcal{O} \text{ or } Q = -P$$

Families of elliptic curves

Let the subgroup size be the large prime number $r \mid \#E$. Let k be the **embedding degree**, $r \mid (p^k - 1)$.

A pairing-friendly elliptic curve has a **small embedding degree** and a **large subgroup size**[1].

Random curves cannot meet these requirement. A family of pairing-friendly elliptic curves use polynomials as parameters and suit a **required security level**.

[1] $k < 50$, r 160-bits

KSS curves, k=18

Kachisa et al. [2008] presented a new method for constructing pairing-friendly elliptic curves.

The parameters of these types of curves are:

- $t(x) = (x^4 + 16x + 7)/7$
- $p(x) = (x^8 + 5x^7 + 7x^6 + 37x^5 + 188x^4 + 259x^3 + 343x^2 + 1763x + 2401)/21$
- $r(x) = (x^6 + 37x^3 + 343)/343$
- Let $\rho \approx \frac{\deg(p(x))}{\deg(r(x))} = 4/3$.

$p(x)$ and $r(x)$ represent primes and $t(x)$ represents integers when $x \equiv 14 \bmod 42$

Cinvestav

More curves

Other families of pairing-friendly elliptic curves with different embedding degrees:

- MNT curves
- Freeman curves
- BN curves
- KSS curves

For an extended description of these and other families of pairing-friendly elliptic curves, refer to Freeman et al. [2006]

Cinvestav

Discrete logarithm problem

Let $P = (x, y) \in E(\mathbb{F}_p)$ and $n \in \mathbb{Z}$, $n \geq 0$. Define $[n]P = P + P + \cdots + P$. The **order of the point** P is the smallest n such that $[n]P = \mathcal{O}$.

Denote $< P >$ the group generated by P. In other words,

$$< P >= \{\mathcal{O}, P, P + P, P + P + P, ...\}$$

Let $Q \in < P >$. Given Q, find n such that $Q = [n]P$. This is known as the **Elliptic Curve Discrete Logarithm Problem (ECDLP)**.

Known attacks affect some anomalous curves, P with a small prime order and some weak combinations of parameters.

Cinvestav

Security level

The security levels in ECC are defined to match the Advanced Encryption Standard (AES), which are as follows. Freeman et al. [2006]

Security level (in bits)	Subgroup size r (in bits)	Extension field size p^k (in bits)	Embedding degree k	
			$\rho \approx 1$	$\rho \approx 2$
80	160	960–1280	6–8	2*,3–4
112	224	2200–3600	10–16	5–8
128	256	3000–5000	12–20	6–10
192	384	8000–10000	20–26	10–13
256	512	14000–18000	28–36	14–18

Table: Bitsize comparison between levels of security

Cinvestav

What is a Pairing?[2]

A **Pairing** is a *bilinear map* on an *Abelian group* M taking values in some other Abelian group R,

$$\langle \cdot, \cdot \rangle : M \times M \longrightarrow R$$

There are many Abelian groups we might consider

- $\mathbb{Z}$, or more generally $\mathbb{Z}^d$
- $\mathbb{Z}/m\mathbb{Z}$, a cyclic group of order m, or generally $(\mathbb{Z}/m\mathbb{Z})^d$
- $\mathbb{F}_p$ with addition as the group law, or generally $\mathbb{F}_p^d$
- $\mathbb{F}_p^*$ with multiplication as the group law
- $E(\mathbb{F}_p)$**, the group of** $\mathbb{F}_p$**-points on an elliptic curve**
- μ_m**, the group of** m^{th}**-roots of unity**
- etc.

[2] this slide is taken from J. H. Silverman's talk in Pairing 2010

Cinvestav

Definition

Here, we define a **pairing** as a map: $G_1 \times G_2 \longrightarrow G_T$.

These groups are finite and cyclic. G_1 and G_2 are additively-written and at least one is of prime order r, $G_{1,2} \subseteq E(\mathbb{F}_{p^d})$.

G_T, is multiplicati vely-written and of order r, $G_T \subseteq \mu_r$ or just $E(\mathbb{F}^*_{p^k})$

Properties:

- Bilinearity
  $$e(P + P', Q) = e(P, Q) \times e(P', Q)$$
  $$e(P, Q + Q') = e(P, Q) \times e(P, Q')$$
- Non-degeneracy
  $$\forall P \in G_1,\ P \neq \mathcal{O}\text{: } \exists Q \in G_2 \text{ s.t. } e(P, Q) \neq 1$$
  $$\forall Q \in G_2,\ Q \neq \mathcal{O}\text{: } \exists P \in G_1 \text{ s.t. } e(P, Q) \neq 1$$
- Efficiently computable

Cinvestav

Twist of a curve

To **speed-up** the pairing computation and to save **bandwidth**[3], use of the twist of a curve.

Let $E/\mathbb{F}_{p^k}$ and $E'/\mathbb{F}_{p^e}$, $e = k/d$, $d \in [2, 3, 4, 6]$ and e **minimal**. Then E' is said to be a twist of degree d if there exists an isomorphism $\psi : E' \longrightarrow E$.

Let $\chi \in \mathbb{F}^*_{p^k}$, then the short Weierstraß formulae for the twists of a curve, corresponding to χ mod $(\mathbb{F}^*_{p^k})$ are given by:

$$d = 2 : y^2 = x^3 + \frac{a}{\chi^2}x + \frac{b}{\chi^3}$$

$$d = 4 : y^2 = x^3 + \frac{a}{\chi}x$$

$$d = 3, 6 : y^2 = x^3 + b/\chi.$$

[3] bandwidth means power consumption in hardware

Twist of a curve... cont

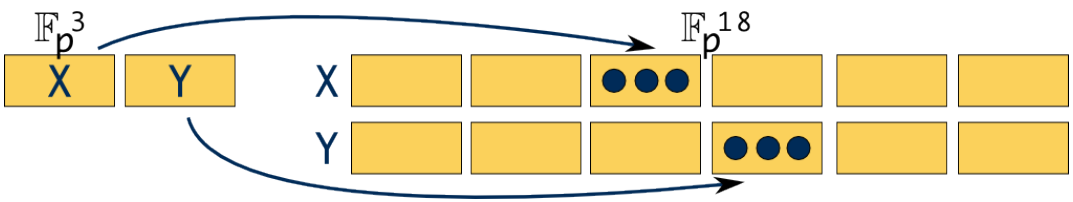
To use the twist of a curve:

- Define the point $Q \in G_2$ over the twist curve $E'(\mathbb{F}_{p^e})$
- Choose $\chi \in \mathbb{F}_{p^e}$ such that $W^d - \chi$ is irreducible over $\mathbb{F}_{p^e}[W]$
- If $\delta \in \mathbb{F}_{p^k}$ is a root of $W^d - \chi$, then there exists a homomorphism from the twist to the original curve.

For example, KSS: $k = 18$ $(d = 6)$

$\psi : E'(F_{p^3}) \rightarrow E(\mathbb{F}_{p^{18}})$ defined by: $(x, y) \rightarrow (x.\delta^{\frac{1}{3}}, y.\delta^{\frac{1}{2}})$

with an isomorphism given by $[.] : \mu_d \rightarrow Aut(E) : \delta \mapsto [\delta]$ with $[\delta](x, y) = (x.\delta^2, y.\delta^3)$



Cinvestav

The Tate pairing

Miller [1986] discovered a function to compute cryptographic pairings. The **Tate pairing** requires one application of the Miller loop.

Apply $\lfloor Log_2(r) \rfloor - 1$ times the double-and-add, line-and- tangent algorithm.

The Tate pairing is a map $E(\mathbb{F}_p)[r] \times E(\mathbb{F}_{p^k})/rE(\mathbb{F}_{p^k}) \rightarrow \mathbb{F}^*_{p^k}/(\mathbb{F}^*_{p^k})^r$ defined as:

Definition

$$e_r : (P, Q) \mapsto \langle P, Q \rangle_r = f_{r,P}(Q)^{(p^k-1)/r}$$

Cinvestav

The ate pairing

The **ate pairing** is particularly suitable for pairing-friendly curves with small values of t. Hess et al. [2006]

Apply $\lfloor Log_2(t-1) \rfloor - 1$ times the double-and-add, line-and-tangent algorithm.

Let $T = t - 1$, $Q \in G_2[r]$ and $P \in G_1[r]$, the ate pairing is defined as:

Definition

$$e_T : (Q, P) \mapsto \langle Q, P \rangle_T = f_{T,Q}(P)^{(p^k-1)/r}$$

Cinvestav

The R-ate pairing

The **R-ate pairing** introduced by Lee, et al. [2007] takes three (two) short Miller loops to calculate a shorter pairing than the ate.

Let $A = aB + b$ as in Zhao [2007], $m_{1,2} = max/min\{a, b\}$.

Apply upto $\lfloor Log_2(m_2) \rfloor + \lfloor Log_2([\frac{m_1}{m_2}]) \rfloor - 2$ times the double-and-add, line-and-tangent algorithm.

The R-ate pairing with $P \in G_2[r]$ and $Q \in G_1[r]$ is as follows:

Definition

$$e_{A,B} : (P, Q) \mapsto \langle P, Q \rangle_{A,B} = f_{a,BP}(Q) \times f_{b,P}(Q) \times G_{aBP,bP}(Q)$$

Needs a careful choice of non-trivial pairs A and B.

Cinvestav

The Pairing Lattice

The pairing lattice introduced by Hess, is a uniquely defined monic function of low degree.

Given an integer s, and a vector $h(s) = \sum_{i=0}^{d} h_i x^i \in \mathbb{Z}[x]$, with $h(s) \equiv 0 \bmod r$, and $d = \varphi(k)$, then the pairing lattice is defined as follows:

Definition

$$e_{s,h} : (Q, P) \mapsto f_{s,h,Q}(P)^{(p^k-1)/r}$$

where $\varphi(.)$ is the Euler totient function.

Cinvestav

Constructing s and $h(s)$

From the definition, we sets $= r$, the subgroup size, but we prefer to take $s = T = t - 1$, the trace of the Frobenius.

To construct $h(s)$, we take a matrix $m \times m$, with $m = \varphi(k)$:

$$M = \begin{pmatrix} r & 0 & \cdots & 0 \\ -T & 1 & 0 & \cdots & 0 \\ -T^2 & 0 & 1 & \cdots & 0 \\ \vdots & & & \ddots & \vdots \\ -T^{m-1} & 0 & & \cdots & 1 \end{pmatrix}$$

Let $w = (w_0, w_1, \ldots, w_{m-1})$ be the shortest $\mathbb{Z}$-linear combination of the rows of M, then we can construct $h = \sum_{i=0}^{m-1} w_i x^i$.

Cinvestav

Reducing $h(s)$

To reduce the matrix M, which contrains the vector $h(s)$, we transform it into a Weak Popov form (polynomial-echelon form). The shortest vector should be of low degree.

Then, we can construct the pairing lattice as:

$$(Q, P) \mapsto \left(\prod_{i=0}^{l} f_{c_i, Q}^{p^i}(P) \cdot \prod_{i=0}^{l} G_{[s_i+1]Q, [c_i q^i]Q}(P) \right)^{(p^k-1)/r}$$

... and removing any repeated (unneccesary) operations whenever possible.

Cinvestav

The Optimal Pairing

A pairing function $e(\cdot,\cdot)$ is called Optimal Pairing if it can be computed in $\log_2 r/\varphi(k) + \varepsilon(k)$ basic Miller iterations, with $\varepsilon(k) \leq \log_2 k$.

For some families of elliptic curves, the optimal pairing can be the pairing lattice, the R-ate pairing, or even the ate pairing (for families with small trace of the Frobenius, such is the case of the BLS curves).

Cinvestav

Implementing the pairing function

The Tate and ate pairing are straight forward to implement.

The R-ate pairing is a generalisation of the ate and ate$_i$ pairing, we form a set with all of the $T_i = a.T_j + b$, where $T_i \equiv p^i \bmod r$, and we choose the best.

For the pairing lattice, we construct a matrix $m = \varphi(k) \times \varphi(k)$ with $m_i \equiv 0 \bmod T$, and we transform it into a Weak Popov form (polynomial-echelon form), to get a vector of low degree.

Cinvestav

The pairings.

Let $Q \in G_2[r]$ and $P \in G_1$.

The pairing functions are defined as:

Tate	e_r:	$(Q,P) \longmapsto f_{r,Q}(P)^{(p^k-1)/r}$
ate	e_T:	$(Q,P) \longmapsto f_{T,Q}(P)^{(p^k-1)/r}$
R-ate	$e_{A,B}$:	$(Q,P) \longmapsto (f_{a,BQ}(P) \times f_{b,Q}(P) \times G_{aBQ,bQ}(P))^{(p^k-1)/r}$
Lattice	$e_{s,h(s)}$:	$(Q,P) \longmapsto \left(\prod_{i=0}^{l} f_{h_i,Q}^{p^i}(P) \cdot \prod_{i=0}^{l} G_{[s_i+1]Q,[h_i p^i]Q}(P)\right)^{(p^k-1)/r}$

Cinvestav

Comparison

The total **Miller loop length** can be used for a comparison between the Tate, ate and R-ate pairings functions. Using a KSS elliptic curve with $k = 18$,for $\approx$ AES-192, one has:

Miller-length in iterations		
Tate	$e_r(P, Q)$	376
ate	$e_t(P, Q)$	253
R-ate	$e_{A,B}(P, Q)$	61
P.L.	$e_{A,B}(P, Q)$	62

Table: Comparison of the Miller loop length

Clearly the R-ate pairing, in this case, presents a shorter Miller loop length, but it is not necessary the fastest to compute.

Cinvestav

Automatic generation of pairing functions' code

Motivation:

- Pairing based cryptography is more complex to understand and to implement than, for example, the RSA.
- If a higher security level is desired, a different pairing friendly curve should be selected. For optimisation, it will need a different implementation.
- There are efforts to implement pairings, but only a selected set of the interesting curves and parameters is actively researched.

Cinvestav

Automatic generation of pairing functions' code

... Motivation:

- The pairing function has two main parts: the Miller loop and the final exponentiation:
  - Depending on the pairing function, one or more Miller loops are constructed. Some extra operations are required to link them.
  - The final exponentiation to be efficiently computed need extra work.

- ... but there are a few more things around the pairing function itself that needs to be implemented efficiently.

Cinvestav

Automatic generation of pairing functions' code

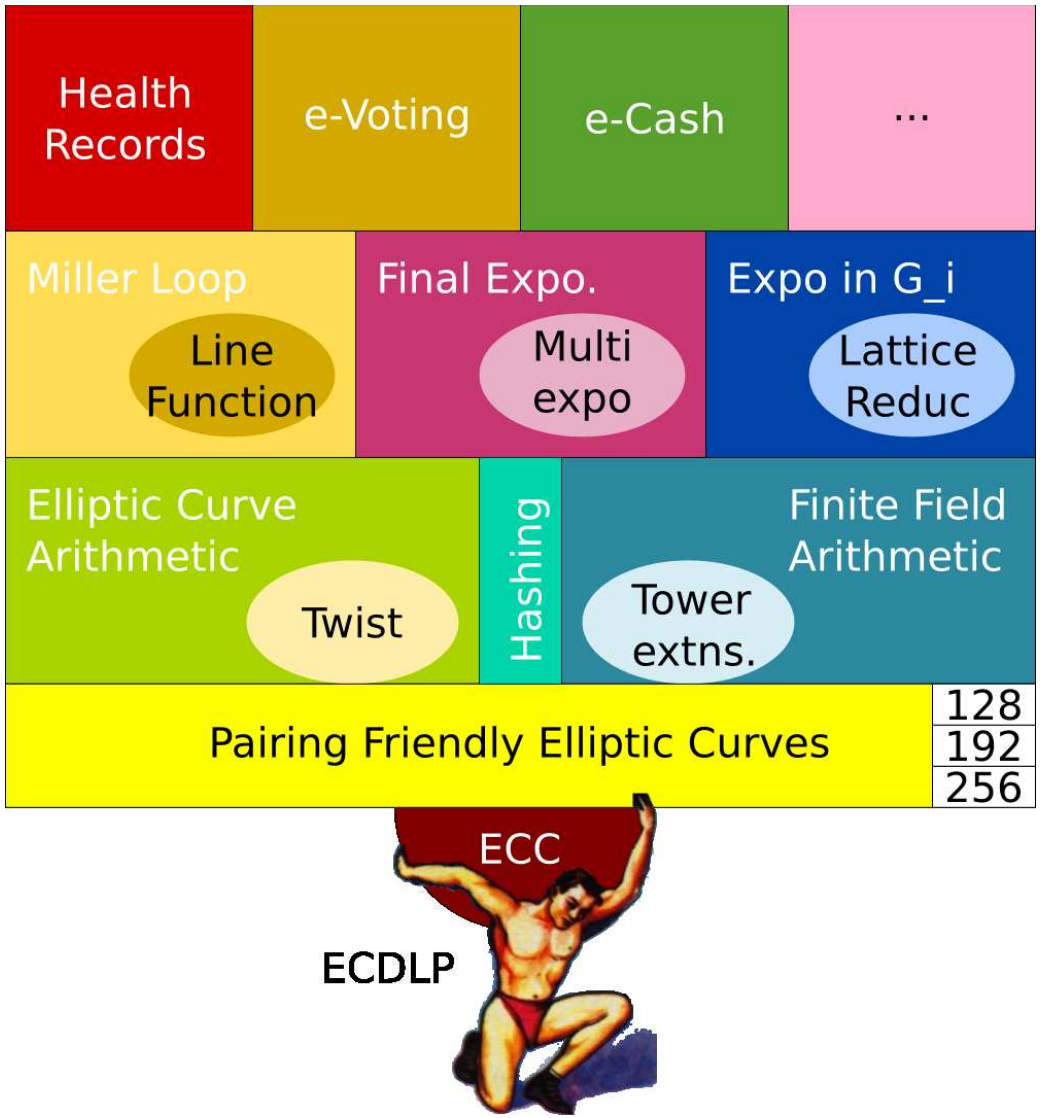
... motivation:

- The draft of the IEEE 1363 is now suggesting families of elliptic curves:
  - 80-bits security level: supersingulars and MNT curves
  - 128-bits security level: BN curves.
  - 192-bits security level: KSS k=18 curves.
  - 256-bits security level: KSS k=36 and BLS k=24 curves.

... but nobody has implemented the top-levels, and some other curves should be of interest too.

Cinvestav

Pairing-Based Cryptography



Cinvestav

The addition-chain

Definition: addition chain

An *addition chain* for a given number e is a sequence $U = (u_0, u_1, u_2, \ldots, u_\ell)$ such that $u_0 = 1, u_\ell = e$ and $u_k = u_i + u_j$ for some i, j with $0 \leq i \leq j < k \leq \ell$.

Deciding if we have found the shortest addition chain for a given positive integer e is an NP-complete problem.

Definition: addition sequence

Given a list of integers $\Gamma = \{v_1, .., v_\ell\}$ where $v_\ell > v_i$ $\forall i = 1, .., \ell - 1$, an addition sequence for Γ is an addition chain for v_ℓ containing all elements of Γ.

Cinvestav

Addition sequence

Definition: multi-addition chain

Given a list of integers $\Lambda = \{s_1, .., s_\ell\}$ where $s_\ell > s_i$ $\forall i = 1, .., \ell - 1$, a multi addition chain for Λ is a set containing multiple addition chains in which some s_i are common.

A multi addition-chain, also known as addition sequence, is a sequence containing several addition chains sharing elements.

Cinvestav

Solving an addition-sequence

To construct a multi-addition-chain we modify the Cruz-Cortéz et al. method, which is designed to generate a simple addition-chain for the RSA method using "Artificial Immune Systems".

The method uses randomization to get a smaller sequence than the supplied. If we are lucky, with the time the sequence will get shorter.

Our code can generate better addition chains than the binary method, however, some complex sequences take a lot of time to get an optimized solution.

Cinvestav

The final exponentiation

One of the most expensive operations in the pairing computation is the final exponentiation by $(p^k - 1)/r$ in the extension field $\mathbb{F}^*_{p^k}$. This is required in the computation of the Tate family of pairing functions on ordinary elliptic curves.

Usually one separates the exponent into 3 pieces: $(p^k - 1)/r \Rightarrow (p^{\frac{k}{2}} + 1) \cdot (p^{\frac{k}{2}} - 1)/\Phi_k(p) \cdot (\Phi_k(p))/r$. The first 2 parts can be easily computed using the Frobenius exponentiation. The remaining, using the Scott et al.[1] method.

where $\Phi(.)$ is the cyclotomic polynomial.

Cinvestav

The final exponentiation, II

At the end of the Scott et al.[1] method, one uses the multi addition-chain method and then generates the code using vector chains with the Olivos method.

Exponentiation in RSA using addition chains usually requires the shortest possible addition chain.

For the final exponentiation method, one may prefer a longer chain if it contains a greater number of doublings and lower number of additions, as it will generate code with more squarings and less multiplications.

Cinvestav

Hashing to $G_2[r]$.

Implementing identity-based protocols using ordinary pairing-friendly elliptic curves requires two groups, at least one of which is to be of order r.

If there is a requirement to hash to a point in G_2 of order r, then the operation becomes much more complex. Thanks to the use of a twisted curve, the operation cost is not exorbitant, but still need to be addressed.

Cinvestav

Hashing to $G_2[r]$, II.

The Scott et al.[2] method for fast hashing to G_2 requires the use of our addition chains method. In this case, the code will generate a doubling or addition depending on the chain.

In the final exponentiation case, the operations are squarings and multiplications in $\mathbb{F}^*_{p^k}$. Here the operations are doublings or additions of a point in a large subgroup of points on a curve. A proper chain selection may lead to shorter computation times.

Cinvestav

Prefix, suffix, inffix...

To add flexibility on the way the code is constructed by different programming languages or libraries, we added a definition for the position of operators.

For an *example* operation there exists an *example*POS key in our definition file:

- 0, prefix: -P
- 1, suffix: P^(-1)
- 2, inffix: P+=Q
- 3, circumfix: inverse(P)
- etc.

Cinvestav

General code construction

To construct a scalar-point multiplication, we use the following instruction:

```
GenericOperation("m1P", "c", "m2P",
language[targetRATE]["scalar-pointPOS"],
"scalar-pointMAP", "scalar-pointMAP2", targetRATE,
language);
```

- The MIRACL library, has the following language specification:
  - `A["scalar-pointPOS"]:=4;`
  - `A["scalar-pointMAP"]:="=";`
  - `A["scalar-pointMAP2"]:="*";`

  We can generate code as:
  - $m1P = c * m2P$

Cinvestav

Code construction RELIC

- RELIC tool-kit has this language specification:
  - `A["scalar-pointPOS"]:=6;`
  - `A["scalar-pointMAP"]:="epD_mul(";`
  - `A["scalar-pointMAP2"]:=")";`

  The output code is:
  - $epD\_mul(m1P, c, m2P)$

This library is unstable and some of its definitions are changing but it is very promising. For example, an earlier version of the library requires the 2nd and 3rd parameter to be swapped.

Cinvestav

Final Exponentiation/Hashing to G_2 code construction

This part of the code generator follow a code sequence construction.

This sequence is a matrix with three columns: *left-hand* operand, and two *right-hand* operands. The operator is implied by the set of instructions used as a parameter.

There is one exemption, if there is a "set" flag on the first column. This flag indicates the code generator to assign a variable with a value from the context. This assignment is done just-in-time to save memory.

Cinvestav

... FE/G_2 specific code construction

For example, a matrix-code sequence:

```
[   set   xB   x1   ]
[   t0    t1   xB   ]
[   set   xB   x0   ]
[   t1    t1   xB   ]
```

will instruct to $xB \leftarrow x1$, then $t0 \leftarrow t1 \circ xB$, and so on.

Cinvestav

More code construction

Any other code construction is done by rewriting the function names: functions whose name is specific to the finite field they are designed, or a construction using reversed polish notation.

At the end, the code is packed in a compressed directory. If the user decided to compile it with the prescibed set of instruction, then it is decompressed over the target directory: typically the one containing the library, or the code for the arithmetic, and it is compiled using an example testbed.

Cinvestav

Sample code constructions

We will demonstrate the code for the following curves:

- BN ($k = 12$) using the pairing lattice, at 128-bit security level
- BN ($k = 12$) using the R-ate pairing, at 128-bit security level
- KSS: $k = 18$ using the pairing lattice, at 192-bit security level
- KSS: $k = 18$ using the R-ate pairing, at 192-bit security level
- KSS: $k = 36$ using the R-ate pairing, (partially). at 256-bit security level.

Cinvestav

Main outcomes (joint work)

- Faster final exponentiation
- Faster hashing to G_2
- Automated generation of optimal pairing code
- A pairing implementation over higher security levels.

Cinvestav

Final thoughts

We have adapted and created several construction methods for automatically generate cryptographic pairing functions' code to ease the job of the protocol implementer.

Our tool shows that automatically generated code is nearly as fast as carefully hand-crafted code. Extra tuning by hand can be done to close the gap against the speed record.

For more information, please consult: (includes videos)
`http://www.computing.dcu.ie/~ldominguez/phdproject.html`

Cinvestav