Conference and Workshop on Ambient Intelligence and Embedded Systems

Portable Elliptic Curve Cryptography For Medium-Sized Embedded Systems

Johan Dams (jd@puv.fi)

Overview

"The protection provided by encryption is based on the fact that most people would rather eat liver than do mathematics,"

Bill Neugent

- Elliptic Curve Cryptography (ECC)
  - Math
- Implementing ECC
  - Signatures
  - Encryption
- Practical Implementation and Algorithms
  - Cryptographic Hash (TIGER)
  - Advanced Encryption Standard (AES)
  - Elliptic Curve Crypto Module

ECC - Why?

- Compare ECC to RSA keylength

```
RSA (1024 bit):
B52264FB7B9154350F1BE765F2979A13091E539B40167BC8FE58F5AB5C4DF3C8B0CC
06A68BF6BBBA30D777345A48F81AC60F2397EDE31E6BCCDF78A584D0E913EC10F07C
A55D368B44ADBB3B82E3606310083DF41318872196852E5B20FA1C6AB1B44C943E21

ECC (192 bit):
FFDF1C7C598311CC1287836B540FB29AF8A35393797D11C8
```

- These two offer the same level of security

...Size Does Matter!

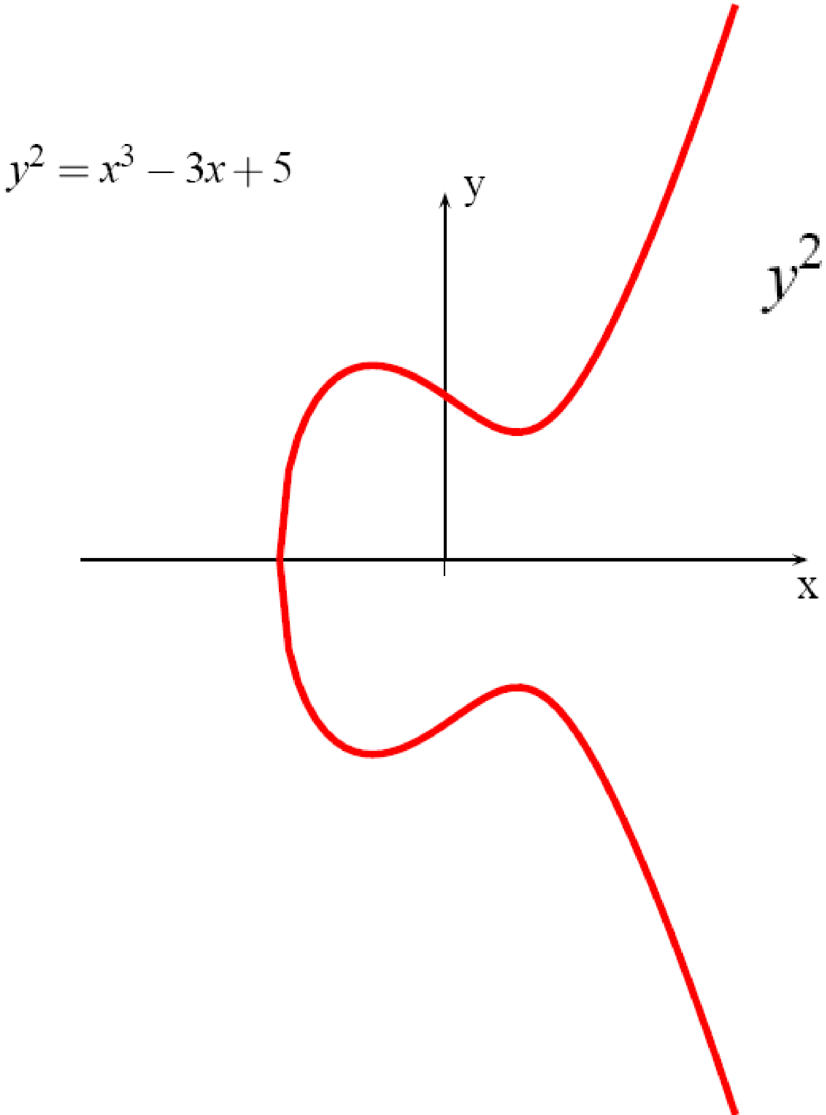
ECC key size (bits)	RSA key size (bits)	Key size ratio
163	1024	1/6
256	3072	1/12
384	7680	1/20
512	15360	1/30

Table 1: ECC and RSA Equivalent Key Sizes

Bits of Security	Symmetric Algorithm	RSA	ECC
80	2TDEA	$k = 1024$	$f = 160 - 223$
112	3TDEA	$k = 2048$	$f = 224 - 255$
128	AES-128	$k = 3072$	$f = 256 - 383$
192	AES-192	$k = 7680$	$f = 384 - 511$
256	AES-256	$k = 15360$	$f = 512+$

Table 2: Equivalent Key Sizes for Symmetric and Asymmetric Cryptography

The Elliptic Curve



Weierstrass Form:

$$y^2 + a_1 xy + a_3 y = x^3 + a_2 x^2 + a_4 x + a_6$$

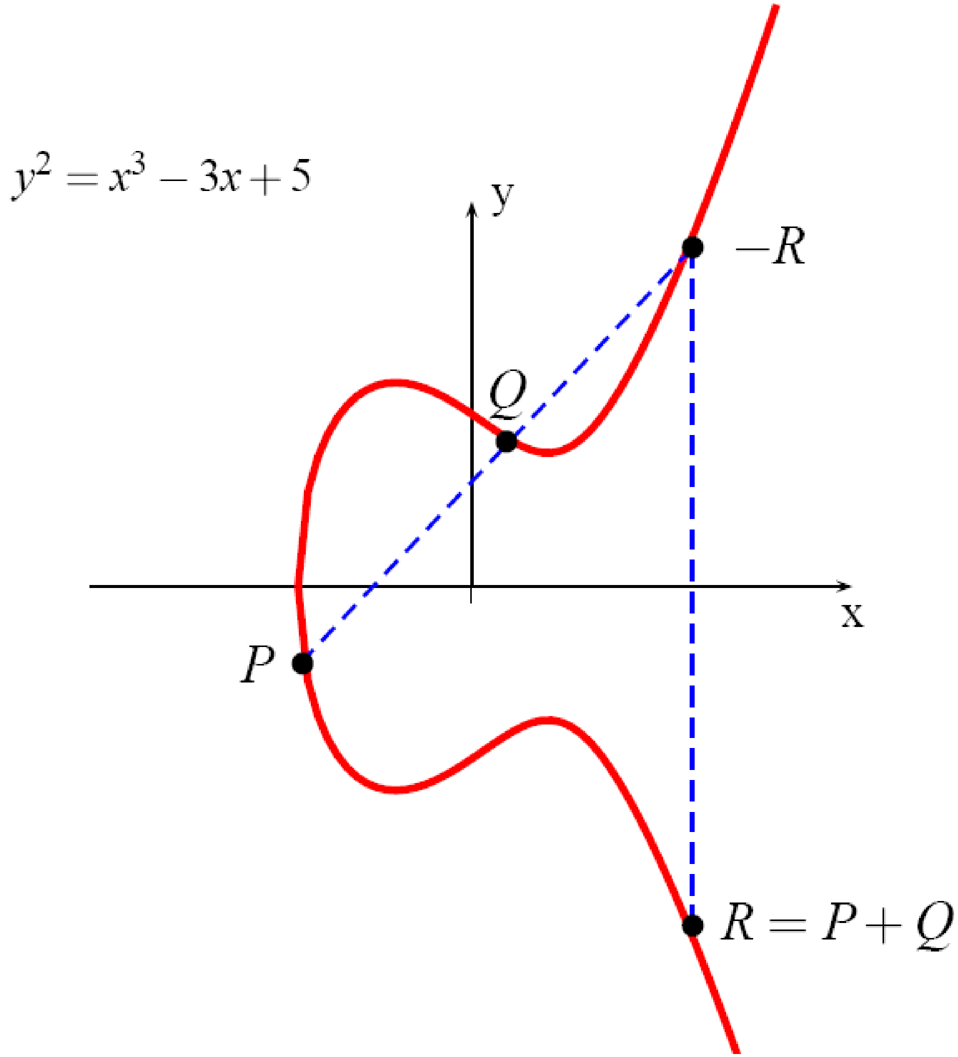
Elliptic Curve Discrete Logarithm Problem

- let P and Q be two points on an elliptic curve such that
  - $kP = Q$
- k is a scalar
- Given P and Q, it is computationally infeasible to obtain k, if k is sufficiently large
- k is the discrete logarithm of Q to P

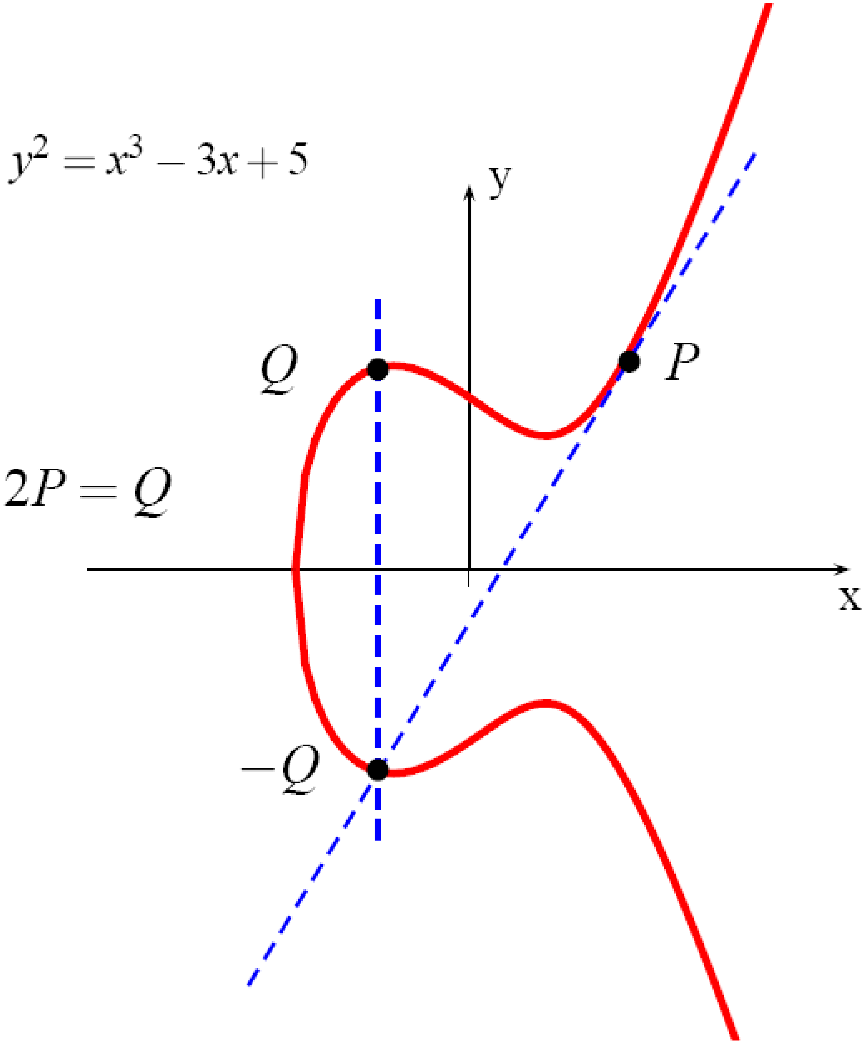
kP = Q

- Main operation involved in ECC: *point multiplication*
- Specially designated point *G* (base point)
  - large fraction of the elliptic curve points are multiples of it
- To generate a key pair:
  - random integer *k* which serves as the private key
  - computes kG which serves as the corresponding public key

Point Addition



Point Doubling



Finite Fields

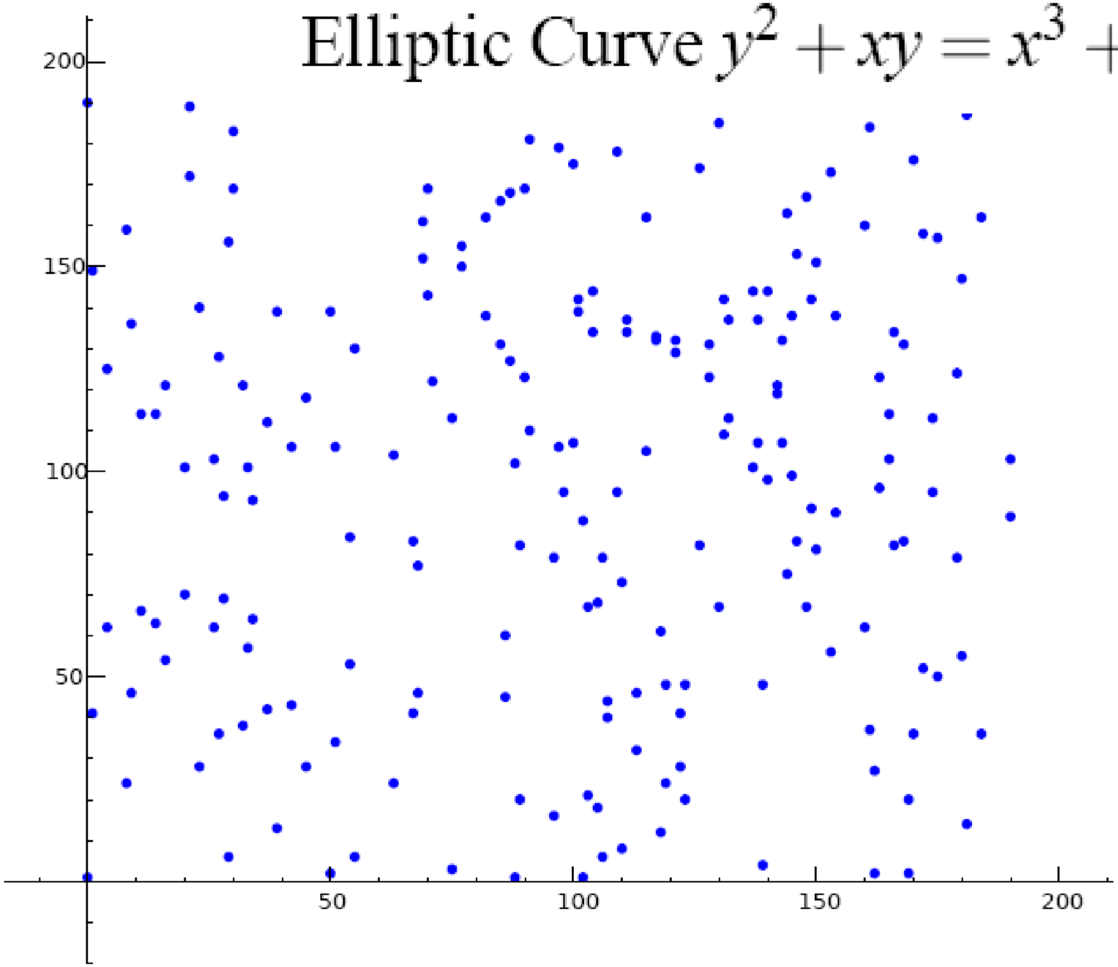
- Operations in the previous section were on real numbers
  - Slow and inaccurate due to rounding
  - Cryptographic operations: fast and accurate

Prime field $GF(p)$

Binary field $GF(2^m)$

Finite Fields

Elliptic Curve $y^2 + xy = x^3 + x^2 + 1$ over $GF(191)$



Finite Fields

- Which Field?
  - Curves over prime field are more efficient to implement in software
  - We can optimise systems using the binary field too
  - Chosen option: binary field
    - Lack of implementations available
    - Always up for a challenge :-)

Practical Implementation

- Requirements:
  - Portable
  - Fast
  - Efficient
- Needs to provide:
  - Signatures and Verification
  - Encryption and Decryption of data

Practical Implementation

- Mathematics in ECC high overhead
  - Goes for public key crypto in general
- Do not use ECC to encrypt the message...
  - Use a combination of symmetric and asymmetric crypto
  - Use ECC to encrypt a symmetric key
  - Use symmetric crypto (in our case AES) to encrypt and decrypt the data – much faster!

Practical Implementation

- Recipient of data generates public/private key pair
- Public key *P* is generated such that *P=dQ*, *d* being the hashed secret (in essence the private key)
- The sender of the message suggests a common secret key *K* so that *K=kQ, k* is a random multiplier
- *k* can then be communicated to the receiver in the form of message *M=kP*
- the receiver can recover the symmetric key *K* using its private key *d*

$$e = d^{-1} mod\, p$$

$$Me = (kP)e = (kdQ)e = (kQ)de = K$$

Choices

- The asymmetric module, based on an elliptic curve over (2^191)
- The symmetric key module, based on AES (Rijndael) with a 192 bits key
- 192 bit TIGER hash function

- For signature generation, the Elliptic Curve Digital Signature Algorithm is implemented

Algorithms to Speed Things Up

- Karatsuba Multiplication
- Itoh-Tsujii Inversion
- de Rooij Point Multiplication

Karatsuba Multiplication

- The traditional method for multiplying A(x) and B(x) would require the following steps
  - $D0 = a0b0$
  - $D1 = a0b1$
  - $D2 = a1b0$
  - $D3 = a1b1$
- $C(x) = A(x).B(x)$ is calculated:
  - $C(x) = D3x^2+(D2+D1)x+D0$

Karatsuba Multiplication

- calculate the following products:
  - $E0 = a0b0$
  - $E1 = a1b1$
  - $E2 = (a0+a1)(b0+b1)$
- $C(x) = A(x).B(x)$ is then calculated as follows:
  - $C(x) = E1x^2+(E2-E1-E0)x+E0$

Karatsuba Multiplication

- The traditional method requires four multiplications and one addition
- Karatsuba method requires three multiplications and four additions
- We thus exchanged one multiplication for three additions.
  - In GF(2^m), addition is especially easy, since addition and subtraction modulo 2 are the same thing and can be done using a basic XOR operation.

Itoh-Tsujii Inversion

$$A^{-1} = (A^r)^{-1} A^{r-1} \text{ where } r = \frac{p^{m-1}}{p-1}$$

Algorithm 1: Itoh-Tsujii Inversion

Input: $A \in GF(p^m)$
Output: A^{-1}

$r \leftarrow (p^m - 1)/(p-1)$
compute A^{r-1} in $GF(p^m)$
compute $A^r = A^{r-1}A$
compute $(A^r)^{-1}$ in $GF(p)$
compute $A^{-1} = (A^r)^{-1}A^{r-1}$
return A^{-1}

Itoh-Tsujii Inversion

- This algorithm is fast because steps 3 and 5 both involve operations in the sub-field GF(p)
- Similarly, if a small value of p is used, a look-up table can be used for inversion in step 4
- The majority of time spent in this algorithm is in step 2, the first exponentiation.
  - However, since r is known ahead of time, an efficient addition chain for the exponentiation in step 2 can be precomputed and hard-coded into the algorithm.

de Rooij Point Multiplication

- Remember point multiplication
  - Most occuring operation in ECC
- Some methods used for ordinary integer exponentiation can be adapted to improve these operations.
  - The (binary)-double-and-add algorithm is perhaps the most well-known algorithms in this regard.
- Using de Rooij, we are able to reduce the number of group operations necessary by a factor of four over the binary double-and-add algorithm.

de Rooij Point Multiplication

Algorithm 2: de Rooij Fixed Point Multiplication using Pre-Computation and Vector Addition Chains

Require: $\{b^0A, b^1A, \ldots, b^tA\}, A \in E(GF(p^m))$, *and* $s = \sum_{i=0}^{t} s_i b^i$

Ensure: $C = sA, C \in E(GF(p^m))$

Define $M \in [0,t]$ such that $z_M \geq z_i$ for all $0 \leq i \leq t$
Define $N \in [0,t], N \neq M$ such that $z_N \geq z_i$ for all $0 \leq i \leq t, i = M$
for $i \leftarrow 0$ *to* t **do**
  $A_i \leftarrow b^i A$
  $z_i s_i$
end
Determine M and N for $\{z_0, z_1, \ldots, z_t\}$
while $z_N \geq 0$ **do**
  $q \leftarrow \lfloor Z_M / Z_N \rfloor$
  $A_N \leftarrow qA_M + A_N$ (Here we apply binary-double-and-add)
  $z_M \leftarrow z_M \bmod z_N$
  Determine M and N for $\{z_0, z_1, \ldots, z_t\}$
end
$C \leftarrow z_M A_M$

Conclusion and Future Work

- Results
  - Nokia N800, TI OMAP 2420 clocked at 330MHz (GNU Libc)
  - Freescale MPC5200B clocked at 400MHz (GNU Libc)
  - Renesas SH7203 clocked at 200MHz (uClibc)
  - Freescale ColdFire MCF54455 clocked at 266MHz (GNU Libc)
  - Freescale ColdFire MCF52277 clocked at 160MHz (uClibc)

Conclusion and Future Work

	multiplication	squaring	quad-solving	inversion
MPC5200B	0.0014 s	0.0000 s	0.0500 s	0.0500 s
Nokia N800	0.0043 s	0.0000 s	0.1100 s	0.0300 s
SH7203	0.0057 s	0.0100 s	0.2700 s	0.1300 s
MCF54455	0.0214 s	0.0050 s	0.4800 s	0.3500 s
MCF52277	0.0486 s	0.0250 s	1.1200 s	0.7800 s

Table 3: Field Operations

Conclusion and Future Work

	scalar multiplication
Nokia N800	0.035 s
MPC5200B	0.037 s
SH7203	0.088 s
MCF54455	0.297 s
MCF52277	0.737 s

Table 4: Scalar Multiplication

	Encrypt-Decrypt	Sign/Verify	Key generation
Nokia N800	0.122 s/cycle	0.087 s/cycle	0.036 s/key
MPC5200B	0.130 s/cycle	0.100 s/cycle	0.030 s/key
SH7203	0.303 s/cycle	0.223 s/cycle	0.086 s/key
MCF54455	1.021 s/cycle	0.763 s/cycle	0.300 s/key
MCF52277	2.559 s/cycle	1.928 s/cycle	0.761 s/key

Table 5: Cryptographic Operations

Conclusion and Future Work

- The focus of future work will be on the development of an efficient Public Key Infrastructure (PKI) with implementations for sensor networks and other applications
- Management of a large number of keys especially while certifying every key is a major obstacle that needs to be tackled before easy and large scale deployment becomes feasible.

Demo

- Demo is running on a Renesas SH7203
- Decrypts images and display on screen
- Encrypt a data file

Questions?

- Thank You for Your Attention!