CS ^{2}, Conference of PhD Students in Computer Science
4th July, 2002.

Efficiency Analysis and Comparison of Public Key Algorithms

Csilla Endrodi
BUTE DMIS, Ph.D. student
csilla@mit.bme.hu
S E A R C H Laboratory

Contents

- **Public Key Algorithms**
  - Existing Algorithms
  - Differences
  - RSA
  - ECC
- **Analysis and Comparison**
  - The Measured Data
  - The Correspondence for the Comparison
  - Software Implementation and Measuring Environment
- **Results**
  - Time of Execution
  - Size of Data Files
- **Summary**

Public Key Algorithms

There **exist** several public key algorithms.

Mathematical Hard Problems:

- Integer Factorisation Problem (IFP)
- Discrete Logarithm Problem (DLP)
- Elliptic Curve Discrete Logarithm Problem (ECDLP)
- Lattice Reduction – or Closest Vector Problem (CVP)
- others…

Data Security Functions:

- Key Agreement
- Encryption
- Digital Signature
- Anonymity Protocols
- etc.

Public Key Cryptography Algorithms

*Algorithms Based On **DLP**:*

- Diffie-Hellmann Key Agreement (DH)
- Diffie-Hellmann Key Agreement 2 (DH2)
- El-Gamal Encryption Scheme
- Menezes-Qu-Vanstone (MQV)
- Efficient Compact Subgroup Trace Representation (XTR)
- Digital Signature Algorithm (DSA)
- BlumGoldwasser
- Zheng-Seberry
- Ballare-Rogaway
- Nyberg-Rueppel Signature Scheme
- Schnorr

*Algorithms based on **IFP**:*

- RSA
- Rabin-Williams

Public Key Cryptography Algorithms

*Algorithms based on **ECDLP**:*

- Elliptic Curve Diffie-Hellmann Key Agreement (ECDH)
- Elliptic Curve Menezes-Qu-Vanstone (ECMQV)
- Elliptic Curve Integrated Encryption Scheme (ECIEC)
- Elliptic Curve Digital Signature Algorithm (ECDSA)
- Elliptic Curve Nyberg-Rueppel (ECNR)

Other:

- LUCDIF
- LUCELG
- LUCRSA
- Merkle-Hellmann
- Chor-Rivest
- NTRU
- McEliece

Public Key Cryptography Algorithms

*Algorithms based on **ECDLP**:*

- Elliptic Curve Diffie-Hellmann Key Agreement (ECDH)
- Elliptic Curve Menezes-Qu-Vanstone (ECMQV)
- Elliptic Curve Integrated Encryption Scheme (ECIEC)
- Elliptic Curve Digital Signature Algorithm (ECDSA)
- Elliptic Curve Nyberg-Rueppel (ECNR)

Other:

- LUCDIF
- LUCELG
- LUCRSA
- Merkle-Hellmann
- Chor-Rivest
- NTRU
- McEliece

Currently most of the applications uses RSA.

Public Key Cryptography Algorithms

Algorithms based on ***ECDLP****:*

- Elliptic Curve Diffie-Hellmann Key Agreement (ECDH)
- Elliptic Curve Menezes-Qu-Vanstone (ECMQV)
- Elliptic Curve Integrated Encryption Scheme (ECIEC)
- Elliptic Curve Digital Signature Algorithm (ECDSA)
- Elliptic Curve Nyberg-Rueppel (ECNR)

Other:

- LUCDIF
- LUCELG
- LUCRSA
- Merkle-Hellmann
- Chor-Rivest
- NTRU
- McEliece

Currently most of the applications uses RSA.

A new promising alternative is ECC.

Public Key Cryptography Algorithms

Differences

- They accomplish the same data security functions,
- BUT they are even not absolutely interchangeable.

Differences:

- Different mathematical background
  - Incompatible parameters (publ., sec. keypairs, common param.)
  - Different limitations
  - Different efficiency feature during their usage
- The best known, general breaking algorithms are different
  - Different requirements for key size belonging to a defined security level

Public Key Cryptography Algorithms

Differences

- ➢ They accomplish the same data security functions,
- ➢ BUT they are even not absolutely interchangeable.

Differences:

- Different mathematical background
  - – Incompatible parameters (publ., sec. keypairs, common param.)
  - – Different limitations
  - – Different efficiency feature during their usage
- The best known, general breaking algorithms are different
  - – Different requirements for key size belonging to a defined security level

Which to chose?

Security level: Must be clearly defined as an *assumption*

– Their limitations of the operation

– Their efficiency characteristics *(not only the key size!)*

Public Key Cryptography Algorithms

Differences

- They accomplish the same data security functions,
- BUT they are even not absolutely interchangeable.

Differences:

- Different mathematical background
  - Incompatible parameters (publ., sec. keypairs, common param.)
  - Different limitations
  - Different efficiency feature during their usage
- The best known, general breaking algorithms are different
  - Different requirements for key size belonging to a defined security level

Which to chose?

Security level: Must be clearly defined as an *assumption*
- Their limitations of the operation
- Their efficiency characteristics *(not only the key size!)*

How to compare them?

The incompatibility of the algorithms becomes their comparison difficult.

RSA

RSA algorithm (1978)

Key Generation:	p, q primes; $m = p*q$
	$\Phi(m) = (p-1) * (q-1)$
	e optional: $1 \leq e \leq \Phi(m)$; $(e, \Phi(m)) = 1$
Public Key:	(e, m)
Secret Key:	(d, m)
Encryption Function:	$E(x) = x^e \bmod m$; ! $x < m$;
Decryption Function:	$D(y) = y^d \bmod m$

- It has not been broken for a quarter of century
- Its security is not proved
- Simple, fine mathematical background: modulo arithmetic
- It had been the patent of the RSA Security Inc. up to 20 Sept. 2000.
- The most wide-spread public key algorithm

- Typical parameters: 512-4096-bit modulus; recommended: 1024 bit
- Speed-up: choose for ‘e’ a small value (e = 65537 = 10001h)
- The maximal size of data that can be encrypted in one step is determined by the modulus

ECC algorithm (1985-)

Common parameters:	E(K) elliptic curve, **B** base point
Secret Key:	k; random number; k < order of the group
Public Key:	**P** (= **B**⊗**B**⊗ … ⊗**B** = k⊙**B**)
Encryption Function :	E(**M**) = ($\mathbf{Y_1}$, $\mathbf{Y_2}$);
	$\mathbf{Y_1}$ = r⊙**B**; $\mathbf{Y_2}$ = **M** ⊗ r⊙(k⊙**B**); r random integer
Decryption Function :	D($\mathbf{Y_1}$, $\mathbf{Y_2}$) = $\mathbf{Y_2}$ ⊗ (-) k⊙ $\mathbf{Y_1}$

- Quite young
- Difficult mathematical background: group over the points of EC
- Breaking methods are also slower

To reach a given security level
a much smaller key size is needed in case of ECC than in case of RSA,
that is the „security-per-key-bit" rate is higher.

- Parameters:
  - Size of Key: 110 - 570-bit key; recommended: 160-bit.
  - Common parameters: recommended (ANSI, Certicom)
- Speed-up: Precomputed tables for **B** and **P**.

Public Key Cryptography Algorithms

ECC

ECC algorithm (1985-)

Common parameters:	E(K) elliptic curve, **B** base point
Secret Key:	k; random number; k < order of the group
Public Key:	$\mathbf{P}$ (= $\mathbf{B} \otimes \mathbf{B} \otimes \ldots \otimes \mathbf{B} = k \odot \mathbf{B}$)
Encryption Function :	$E(\mathbf{M}) = (\mathbf{Y_1}, \mathbf{Y_2})$;
	$\mathbf{Y_1} = r \odot \mathbf{B}$; $\mathbf{Y_2} = \mathbf{M} \otimes r \odot (k \odot \mathbf{B})$; r random integer
Decryption Function :	$D(\mathbf{Y_1}, \mathbf{Y_2}) = \mathbf{Y_2} \otimes (-)\, k \odot \mathbf{Y_1}$

- Quite young
- Difficult mathematical background: group over the points of EC
- Breaking methods are also slower

**To reach a given security level
a much smaller key size is needed in case of ECC than in case of RSA,
that is the „security-per-key-bit” rate is higher.**

- Parameters:
  - Size of Key: 110 - 570-bit key; recommended: 160-bit.
  - Common parameters: recommended (ANSI, Certicom)
- Speed-up: Precomputed tables for **B** and **P**.

Analysis and Comparison

Aims:

- **Analysis:** Mapping the dependencies between the parameter setting and the efficiency features.
- **Comparison:** Compare the chosen algorithms and try to specify the better one.

Analysis and Comparison

Aims:

- **Analysis:** Mapping the dependencies between the parameter setting and the efficiency features.
- **Comparison:** Compare the chosen algorithms and try to specify the better one.

Which operations?

Tests were made with ECC and with RSA during the

- Key Generation
- Establishment Common Parameters
- Encryption
- Decryption
- Signing
- Signature Verification

Introduction of the Measurement Process

What data?

The measured datas were
– the time of execution
– and the size of data, namely:
- size of the Common Parameter Files,
- size of the Public and Secret Key Files,
- size of the Encrypted Data Files,
- size of the Signature Files

Depending on what?

The parameters of the operations are:
- the size of the applied key
- the size and content of the input data

Other facts having effect for the efficiency behaviour:
- In RSA the value of the exponent (little or great)
- In ECC that precomputed tables were used or not
- In ECC the type of the group ($GF(2^n)$ or $GF(p)$); in case of $GF(2^n)$ the type of the generator polynomial (trinomial or pentanomial): *ECP, EC2NT and EC2NP*

Analysis and Comparison

Analysis

High number of the test cases:

- Many sub-types of the algorithms
- The changeable parameters and many kind of their possible values

with ECC ab. 4000,
with RSA more, than 2000 measures.

Analysis and Comparison

Analysis

High number of the test cases:

- Many sub-types of the algorithms
- The changeable parameters and many kind of their possible values

with ECC ab. 4000,
with RSA more, than 2000 measures.

What datas to compare?

Comparison

Incompatible algorithms:

- Different security „meaning” of the key sizes
- Algorithm sub-types

It had to be made correspondence with each other somehow.

Analysis and Comparison

How to make correspondence?

(1) Size of Keys:

The starting assumption: ***The Security Level***

Key Size ~ Level of Security

⇒ Comparison by **Identical Security Level Providing Key Sizes**

Adequate pairs are determined on the ground of international research results.

RSA (bit)	ECC (bit)
512	113
768	136
1024	**160**
2048	282
4096	409

(2) Type of Algorithms:

Can not be ordered together at all.

⇒ Handled as further versions of the algorithms.

RSA: 3 relevant sub-cases depending on the public exponent:
$e = 3$; $e = 65537$; e = random.

ECC: 3 relevant sub-cases depending on the type of the applied group:
$GF(p)$; $GF(2^n)$ trinomial; $GF(2^n)$ pentanomial.

Analysis and Comparison

The selected software implementation: **Crypto++**

- Open source
- Implements the required ECC and RSA algorithms
- Developed since 1995
- Used by many programmers
- Includes several implementations of cryptography algorithms according to the established international recommendations (IEEE P1363, X.509, SEC1, SEC2 etc.).

Analysis and Comparison

The selected software implementation: **Crypto++**

- Open source
- Implements the required ECC and RSA algorithms
- Developed since 1995
- Used by many programmers
- Includes several implementations of cryptography algorithms according to the established international recommendations (IEEE P1363, X.509, SEC1, SEC2 etc.).

Software framework:

- Microsoft Visual C++ 6.0; Crypto++ version 4.1
- Command line usage, parameters from file
- Measurement results to text file → analysed with Microsoft Excel

Hardware and software environment:

PC: Intel Celeron 450 MHz processor, 128 KB cache, 192 MB memory, 75 MHz memory bus speed, 75 MHz FSB

OS: Windows 98 (version 4.10.1998) operating system

Results

I. Time of Execution

Common Parameters and Key Generation

RSA

- The *generation of the prime numbers* is a crucial subprocess.
  Applied method: Generating random number and testing it. Acceptance is on the ground of probability. The times of execution are not always the same, occasionally can be very long.
- The times of the key generations show exponential distribution.
- Depends on the key size, but does not depend on the value of the exponent.
- Measured values: 0,2 s - 14 min. Typical: 2,8 s (1024-bit key).

ECC

- Generation of new common parameters is difficult. (Recommendations)
- Using precomputed tables: needs time for preparation and disk space for storage.
- Depends on the key size, the type of ECC and the usage of precomputed tables.
- Measured values: 0,054 s - 1,4 min. Typical: 0,09 s (ECP, 161-bit key, precomputed table).

I. Time of Execution

Encryption

RSA

- Depends on the key size and the value of the public exponent, but does not depend on the size and content of the data to be encrypted.
- Measured values: 0,02 s - 6,7 s. Typical: 0,025 s (1024-bit key, e=65537).

ECC

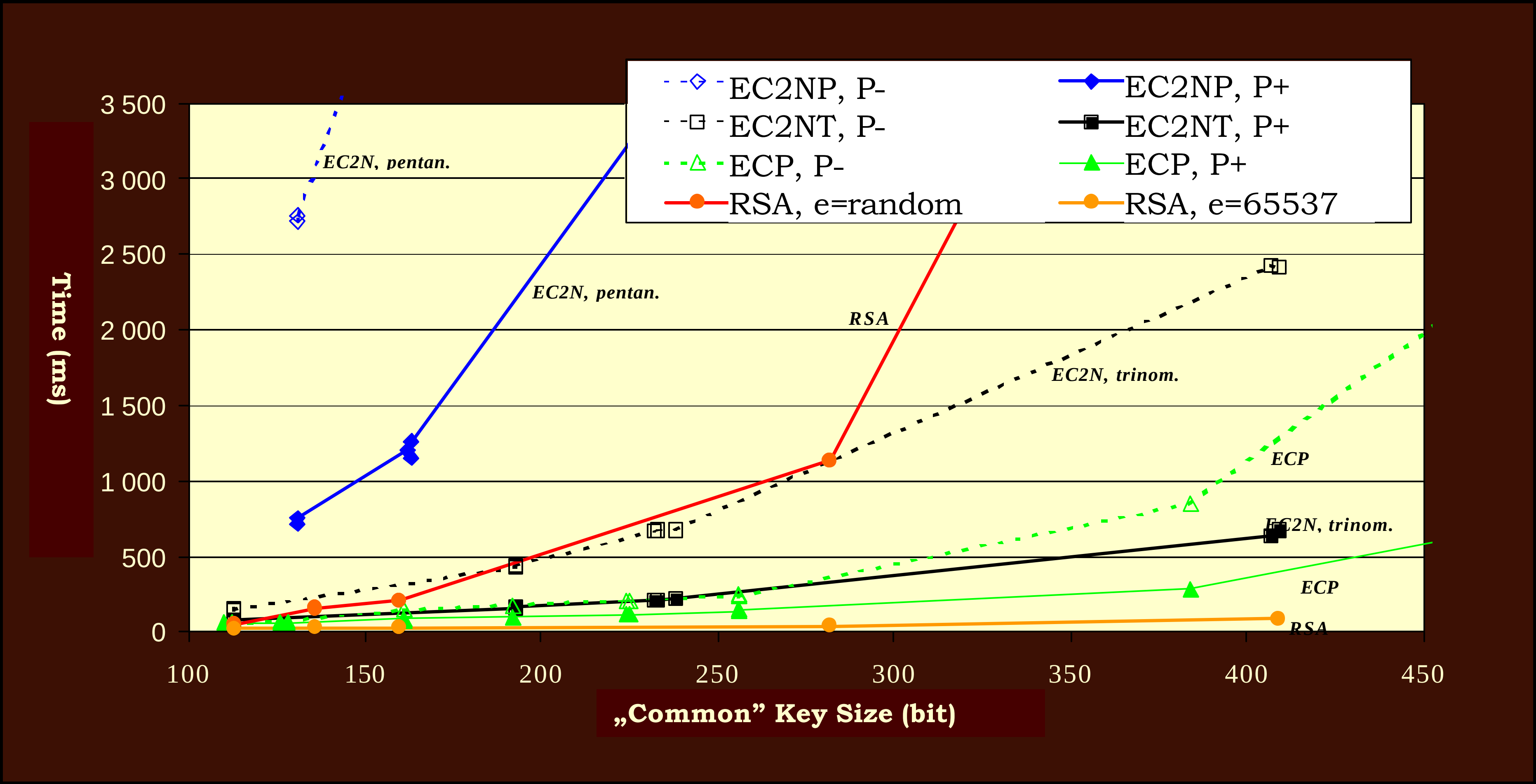
- Depends on the key size, the algorithm type, the size of the data to be encrypted and the usage of precomputed tables.
- Using precomputed tables speeds up the operation more than twice.
- Measured values: 0,05 s - 2,8 min. Typical: 0,12 min (ECP, 163-bit key, 2048 byte data, precomputed table).

Comparison

The fastest encryption is the RSA with small exponent (even e=65537). The measured values show ab. 4-5 times speed.

I. Time of Execution

Encryption



I. Time of Execution

Decryption

RSA

- Depends on the key size, but does not depends on the value of the public exponent and the size and content of the input data.
- Measured values: 0,03 s - 4,45 s. Typical: 0,13 s (1024-bit key).

ECC

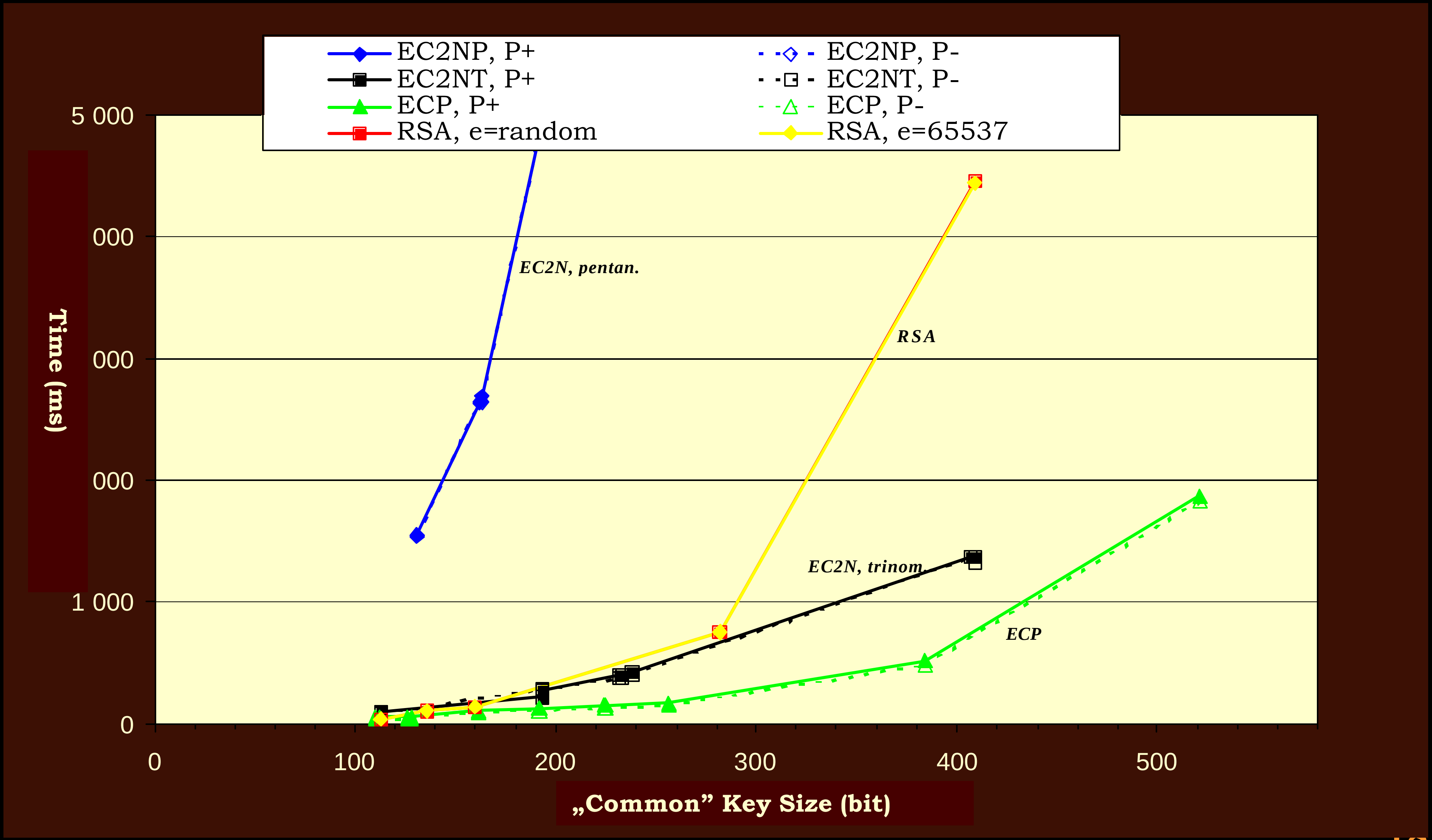
- Depends on the key size, the algorithm type, the size of the input data, but does not depends on the usage of precomputed tables.
- Measured values: 0,03 s - 1,55 min. Typical: 0,136 min (ECP, 163-bit key, 4096 byte data); 0,05 s (ECP, 163-bit key, 22 byte data)

Comparison

The fastest decryption is ECP, according to the tests it means in average double speed compared to the RSA.

I. Time of Execution

Decryption



Signing and Signature Verification

Signing = hash + operation with the secret key

Signature verification = hash + operation with the public key + comparison

The relation of times needed for the signing / signature verification are very similar to the relation of times needed for the decryption / encryption.

Time of Execution

	RSA	ECC
Common Params	—	Difficult ! *Recommended params.*
Key Gen.	! *Nondeterministic* **K** 0,2 s - 14 min 1024-bit key: 2,8 s	0,054 s - 1,4 min ECP, P+, 161-bit key: 0,09 s **K T P**
Encryption	0,02 s - 6,7 s **K E** 1024-bit key, e=65537: 0,025 s	0,05 s - 2,8 min ECP, 163-bit key, 2048 byte data: 0,12 min **K T D P**
Decryption	0,03 s - 4,45 s **K** 1024-bit key: 0,13 s	0,03 s - 1,55 min ECP, 163-bit key, 22 byte data: 0,05 s **K T D**
Signing	~ *Decryption* **K E**	~ *Decryption* **K T D P**
Sign Verification	~ *Encryption* **K**	~ *Encryption* **K T D**

Time of Execution

	RSA	ECC
Common Params	—	Difficult ! *Recommended params.*
Key Gen.	! *Nondeterministic* **K**	**ECC 3 ×** **K T P**
Encryption	0,02 s - 6,7 s 1024-bit key, e=65537: 0,025 s **K E**	0,05 s - 2,8 min ECP, 163-bit key, 2048 byte data: 0,12 min **K T D P**
Decryption	0,03 s - 4,45 s 1024-bit key: 0,13 s **K**	0,03 s - 1,55 min ECP, 163-bit key, 22 byte data: 0,05 s **K T D**
Signing	~ *Decryption* **K E**	~ *Decryption* **K T D P**
Sign Verification	~ *Encryption* **K**	~ *Encryption* **K T D**

Time of Execution

	RSA	ECC
Common Params	—	Difficult ! *Recommended params.*
Key Gen.	! *Nondeterministic* K	**ECC 3 ×** K T P
Encryption	**RSA** with small exponent **4 - 5 ×** K E	K T D P
Decryption	0,03 s - 4,45 s 1024-bit key: 0,13 s K	0,03 s - 1,55 min ECP, 163-bit key, 22 byte data: 0,05 s K T D
Signing	~ *Decryption* K E	~ *Decryption* K T D P
Sign Verification	~ *Encryption* K	~ *Encryption* K T D

Time of Execution

	RSA	ECC
Common Params	—	Difficult ! *Recommended params.*
Key Gen.	K ! *Nondeterministic*	K T P **ECC 3 ×**
Encryption	K E **RSA** with small exponent **4 - 5 ×**	K T D P
Decryption	K	K T D **ECP** with precomp. **2 ×**
Signing	K E ~ *Decryption*	K T D P ~ *Decryption*
Sign Verification	K ~ *Encryption*	K T D ~ *Encryption*

Time of Execution

	RSA	ECC
Common Params	—	Difficult ! *Recommended params.*
Key Gen.	**K** ! *Nondeterministic*	**K T P** **ECC 3 ×**
Encryption	**K E** **RSA** with small exponent **4 - 5 ×**	**K T D P**
Decryption	**K**	**K T D** **ECP** with precomp. **2 ×**
Signing	**K E** **RSA** with small exponent	**P**
Sign Verification	**K** ~ *Encryption*	**K T D** ~ *Encryption*

Time of Execution

	RSA	ECC
Common Params	—	Difficult ! *Recommended params.*
Key Gen.	! *Nondeterministic* K	**ECC 3 ×** K T P
Encryption	**RSA** with small exponent **4 - 5 ×** K E	K T D P
Decryption	K	**ECP** with precomp. **2 ×** K T D
Signing	**RSA** with small exponent K E	P
Sign Verification	K	**ECP** with precomp. K T D

II. Size of Data Files

Common Parameter and Key Files

RSA

- Depends on the key size and the value of the exponent.

ECC

- Depends on the key size and the algorithm type.

Comparison (using Identical Security Level Providing Key Sizes)

Common Key Size (bit)	RSA sum. (byte)	ECC sum. (byte)	ECC sum. with PP (byte)
ECC 113 = RSA 512	872-998	1452	3608
ECC 131 = RSA 768	1224-1422	1632	4044
ECC 160 = RSA 1024	**1584-1844**	**1890**	**4686**
ECC 283 = RSA 2048	3016-3532	3158	8006
ECC 409 = RSA 4096	5848-6870	4428	11328

Encrypted Data, Maximum Size of Encryptable Data

RSA

- The size of the encrypted data depends on the size of the key and the size of data. The encrypted size is equal to the size of the modulus.
- *Demand*: The value of data to be encrypted must be smaller than the modulus.

ECC

- The size of the encrypted data depends on the key size and the input data size. With a given key size the encrypted data's size is always larger than input size with a constant value (e.g. with 160-bit key the enlargement is always 61 byte.)
- The maximal size of the data that can be encrypted in one step depends on the key size. With a 160-bit key this limitation is 4035 byte.

Comparison

Encrypted data made by ECC is smaller. Besides using ECC the size of data that can be encrypted in one step is much larger than in case of RSA, where this restriction is quite strong.

II. Size of Data Files

Encrypted Data

Common Key Size (bit)	**Size of Data to be encrypted (byte)**	**Size of Enc. Data with RSA (byte)**	**Size of Enc. Data with ECC (byte)**
ECC 113 = RSA 512	22	64	73
ECC 131 = RSA 768	22	96	77
	54	96	**109**
ECC 160 =RSA 1024	22	128	**83**
	54	128	**115**
	86	128	147
ECC 283 =RSA 2048	22	256	**115**
	54	256	**147**
	86	256	**179**
	214	256	307
ECC 409 =RSA 4096	22	512	**147**
	54	512	**179**
	86	512	**211**
	214	512	**339**
	470	512	595

II. Size of Data Files

Maximum Size of Encryptable Data

Common Key Size (bit)	RSA Max. Enc. Data (byte)	RSA Size of Encrypted Data (byte)	ECC Max. Enc. Data (byte)	ECC Size of Encrypted Data (byte)
ECC 113 = RSA 512	22	64	4045	4096
ECC 131 = RSA 768	54	96	4041	4096
ECC 160 = RSA 1024	**86**	**128**	**4035**	**4096**
ECC 283 = RSA 2048	214	256	4003	4096
ECC 409 = RSA 4096	470	512	3971	4096

Signature

Both

- Does not depends on the input data (because of the hashing), but only the key size and the applied hash function (e.g. MD5 or SHA).

Comparison: Signature with ECC are quite smaller.

Common Key Size (bit)	RSA Signature Size (byte)	ECC Signature Size (byte)
ECC 113 = RSA 512	64	30
ECC 131 = RSA 768	96	34
ECC 160 = RSA 1024	128	42
ECC 283 = RSA 2048	256	72
ECC 409 = RSA 4096	512	102

Size of Data Files

	RSA		ECC	
Common Params & Key Files	872 byte - 6870 byte 1024-bit key, e=65537: 1584 byte	**KE**	1452 byte - 11328 byte 160-bit key, P-: 1890 byte 160-bit key, P+: 4684 byte	**KP**
Encrypted Data Files	64 byte - 512 byte 1024-bit key, 22 byte: 128 byte	**K**	73 byte - 595 byte 160-bit key, 22 byte: 83 byte	**KD**
Maximal. Size of Encrypted Data Files	! *Strong limitation* 22 byte - 470 byte 1024-bit key: 86 byte	**K**	3971 byte - 4045 byte 160-bit key: 4035 byte	**K**
Signature	64 byte - 512 byte 1024-bit key: 128 byte	**K**	30 byte - 102 byte 160-bit key: 42 byte	**KP**

Size of Data Files

	RSA	ECC
Common Params & Key Files	872 byte - 6870 byte **K E** 1024-bit key, e=65537: 1584 byte	1452 byte - 11328 byte **K P** 160-bit key, P-: 1890 byte 160-bit key, P+: 4684 byte
Encrypted Data Files	64 byte - 512 byte **K** 1024-bit key, 22 byte: 128 byte	73 byte - 595 byte **K D** 160-bit key, 22 byte: 83 byte
Maximal. Size of Encrypted Data Files	! *Strong limitation* **K** 22 byte - 470 byte 1024-bit key: 86 byte	3971 byte - 4045 byte **K** 160-bit key: 4035 byte
Signature	64 byte - 512 byte **K** 1024-bit key: 128 byte	30 byte - 102 byte **K P** 160-bit key: 42 byte

(*) Encryption of large data (it means even 100 byte data depending on the key size) by RSA must be done in several steps, while by ECC only one step can be enough. Thus the encryption by ECC can be *10 times faster* than by RSA (despite of that the encryption executed once by low exponent RSA is faster than by ECC).

Size of Data Files

	RSA	ECC
Common Params & Key Files	**RSA 3 ×** **K E**	**K P**
Encrypted Data Files	64 byte - 512 byte **K** 1024-bit key, 22 byte: 128 byte	73 byte - 595 byte **K D** 160-bit key, 22 byte: 83 byte
Maximal. Size of Encrypted Data Files	! *Strong limitation* **K** 22 byte - 470 byte 1024-bit key: 86 byte	3971 byte - 4045 byte **K** 160-bit key: 4035 byte
Signature	64 byte - 512 byte **K** 1024-bit key: 128 byte	30 byte - 102 byte **K P** 160-bit key: 42 byte

(*) Encryption of large data (it means even 100 byte data depending on the key size) by RSA must be done in several steps, while by ECC only one step can be enough. Thus the encryption by ECC can be *10 times faster* than by RSA (despite of that the encryption executed once by low exponent RSA is faster than by ECC).

Size of Data Files

	RSA	ECC
Common Params & Key Files	**RSA 3 ×** K E	K P
Encrypted Data Files	K	**~ ECC** K D
Maximal. Size of Encrypted Data Files	! *Strong limitation* 22 byte - 470 byte 1024-bit key: 86 byte K	3971 byte - 4045 byte 160-bit key: 4035 byte K
Signature	64 byte - 512 byte 1024-bit key: 128 byte K	30 byte - 102 byte 160-bit key: 42 byte K P

(*) Encryption of large data (it means even 100 byte data depending on the key size) by RSA must be done in several steps, while by ECC only one step can be enough. Thus the encryption by ECC can be *10 times faster* than by RSA (despite of that the encryption executed once by low exponent RSA is faster than by ECC).

Size of Data Files

	RSA	ECC
Common Params & Key Files	RSA 3 × K E	K P
Encrypted Data Files	K	~ ECC K D
Maximal. Size of Encrypted Data Files	! *Strong limitation* (*) K	ECC 50 × K
Signature	64 byte - 512 byte 1024-bit key: 128 byte K	30 byte - 102 byte 160-bit key: 42 byte K P

(*) Encryption of large data (it means even 100 byte data depending on the key size) by RSA must be done in several steps, while by ECC only one step can be enough. Thus the encryption by ECC can be *10 times faster* than by RSA (despite of that the encryption executed once by low exponent RSA is faster than by ECC).

Size of Data Files

	RSA	ECC
Common Params & Key Files	RSA 3 × K E	K P
Encrypted Data Files	K	~ ECC K D
Maximal. Size of Encrypted Data Files	! *Strong limitation* (*) K	ECC 50 × K
Signature	K	ECC 3 × K

(*) Encryption of large data (it means even 100 byte data depending on the key size) by RSA must be done in several steps, while by ECC only one step can be enough. Thus the encryption by ECC can be *10 times faster* than by RSA (despite of that the encryption executed once by low exponent RSA is faster than by ECC).

Summary

Weak Points:

RSA:

- Key Generation
- Maximal Encryptable Data Size

ECC:

- Generating New Common Parameters
- Need for Usage of Precomputed Tables

Strong Side:

RSA:

- Speed of Encryption
- Speed of Sign Verification
- Size of Parameter Files

ECC:

- Speed of Decryption
- Speed of Signing
- Size of Encrypted Data
- Size of Signature

Σ:

Summary

Weak Points:

RSA:

- Key Generation
- Maximal Encryptable Data Size

ECC:

- Generating New Common Parameters
- Need for Usage of Precomputed Tables

Strong Side:

RSA:

- Speed of Encryption
- Speed of Sign Verification
- Size of Parameter Files

ECC:

- Speed of Decryption
- Speed of Signing
- Size of Encrypted Data
- Size of Signature

Σ: Which algorithm is the better...?

Summary

Weak Points:

RSA:

- Key Generation
- Maximal Encryptable Data Size

ECC:

- Generating New Common Parameters
- Need for Usage of Precomputed Tables

Strong Side:

RSA:

- Speed of Encryption
- Speed of Sign Verification
- Size of Parameter Files

ECC:

- Speed of Decryption
- Speed of Signing
- Size of Encrypted Data
- Size of Signature

Σ: **In general there is no clear winner.**

Summary

Weak Points:

RSA:

- Key Generation
- Maximal Encryptable Data Size

ECC:

- Generating New Common Parameters
- Need for Usage of Precomputed Tables

Strong Side:

RSA:

- Speed of Encryption
- Speed of Sign Verification
- Size of Parameter Files

ECC:

- Speed of Decryption
- Speed of Signing
- Size of Encrypted Data
- Size of Signature

$\sum$:

- **In general there is no clear winner.**
- ***BUT***

Summary

Weak Points:

RSA:

- Key Generation
- Maximal Encryptable Data Size

ECC:

- Generating New Common Parameters
- Need for Usage of Precomputed Tables

Strong Side:

RSA:

- Speed of Encryption
- Speed of Sign Verification
- Size of Parameter Files

ECC:

- Speed of Decryption
- Speed of Signing
- Size of Encrypted Data
- Size of Signature

Σ:

- **In general there is no clear winner.**
- ***BUT*** **for a known application it can be decided.**

Thank You for Your Attention!

Csilla Endrodi

<csilla@mit.bme.hu>

References

- **RSA**

RFC 2437, PKCS #1: RSA Cryptography Specifications

http://www.rsa.com, http://www.ietf.org

- **ECC**

Sandards for Efficient Cryptography Group (SECG)

SEC1: *Elliptic Curve Cryptography*

SEC2: *Recommended Elliptic Curve Cryptography Domain Parameters*

- **Crypto++ "*a C++ Class Library of Cryptographic Primitives*"**

http://www.cryptopp.com

- ***XTR (Efficient Compact Subgroup Trace Representation)***

http://www.ecstr.com/

- ***NTRU***

http://www.ntru.com/

Some Points of Interest

- New Competitors
- The Elliptic Curves

New Competitors

XTR (<u>E</u>fficient <u>C</u>ompact <u>S</u>ubgroup <u>T</u>race <u>R</u>epresentation)
http://www.ecstr.com/

- Created by *Arjen Lenstra, Eric Verheul*
- First introduced at Crypto2000 Conference
- XTR's security based on DLP over finite fields

- *„XTR is therefore at least as secure as the RSA system"*
- *"XTR's parameter and key generation is up to 80 times faster than RSA's."*
- *„XTR is as compact as Elliptic Curve Crypto systems."*
- *"XTR is faster than Elliptic Curve Crypto systems."*
- *"XTR is the best of two worlds, i.e. RSAs and ECCs."*
- *"XTR is an alternative to RSA\ECC in SSL & WAP."*

New Competitors

NTRU

http://www.ntru.com/

- Created by: *Jeffrey Hoffstein, Joseph Silverman, Jill Pipher, Daniel Lieman*
- 1996: NTRU Cryptosystems Inc.
- The algorithm and security analysis were first published in the proceedings of the Algorithmic Number Theory Symposium (ANTS III, Portland 1998)
- 2000, July: USA patent
- Security of NTRU is based on the lattice reduction (or closest vector problem), which has been scrutinized for more than 100 years
- NTRU Challenge (none of them were solved up to now)
- The core NTRU algorithms are currently being standardized in the IEEE P1363 Working Group for Standards In Public Key Cryptography.
- The algorithm and the implementation were scrutinised by well-known experts.

New Competitors

NTRU

- *“NTRU operates as much as 2,000 times faster than its nearest competitor “*
- *„NTRUEncrypt is 100 times faster than any competitor.”*
- *„Key generation of NTRUEncrypt tops 300 times speed advantage compared to RSA”*
- *„The structure of NTRU is even more simply than the structure of RSA”*
- *„The size of foot-print can be 50 times smaller.”*
  (Encryption can be implemented with a 20-line, decryption with a 52-line program in C.)
- *„NTRU key sizes are similar to the currently used.”*
  - Secret key: max. 80 bit
  - Sum total: max. 2000 bit
- *Corresponding key sizes with RSA:*
  NTRU 167 » RSA 512;
  NTRU 263 » RSA 1024;
  NTRU 503 » RSA 2048;
- *Time for breaking:*
  NTRU 167: 550 year;
  NTRU 503: 5,4 10^3 year

The Elliptic Curves

The general form of the elliptic curves:

$$y^2 + axy + by = x^3 + cx^2 + dx + e \qquad x, y, a, b, c, d, e \in \mathbf{F}$$

In case of **F** = Real Number Field

$$y^2 = x^3 + ax + b \qquad x, y, a, b \in \mathbf{R}$$

The (x,y) points, which satisfy the equation form the elliptic curve **E(F)**.

Points of the elliptic curve forms a *group* with a suitably chosen operation.

The **0** element is also a member of the group.

The **0** element is the neutral element of the group.

In cryptography the **finite field** based elliptic curves can be applied.

The Elliptic Curves

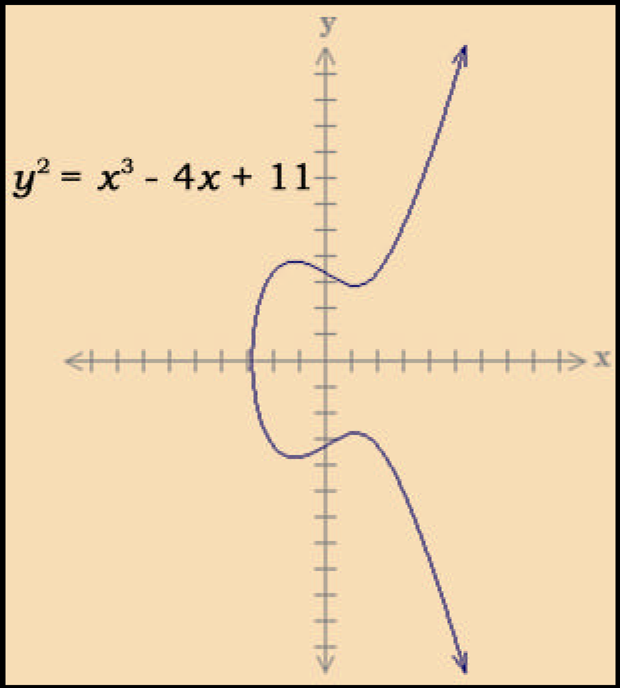
The general form of the elliptic curves:

$$y^2 + axy + by = x^3 + cx^2 + dx + e \qquad x, y, a, b, c, d, e \in \mathbf{F}$$

In case of **F** = Real Number Field

$$y^2 = x^3 + ax + b \qquad x, y, a, b \in \mathbf{R}$$

The (x,y) points, which satisfy the equation form the elliptic curve **E(F)**.



Points of the elliptic curve forms a *group* with a suitably chosen operation.

The **0** element is also a member of the group.

The **0** element is the neutral element of the group.

In cryptography the **finite field** based elliptic curves can be applied.

The Elliptic Curves

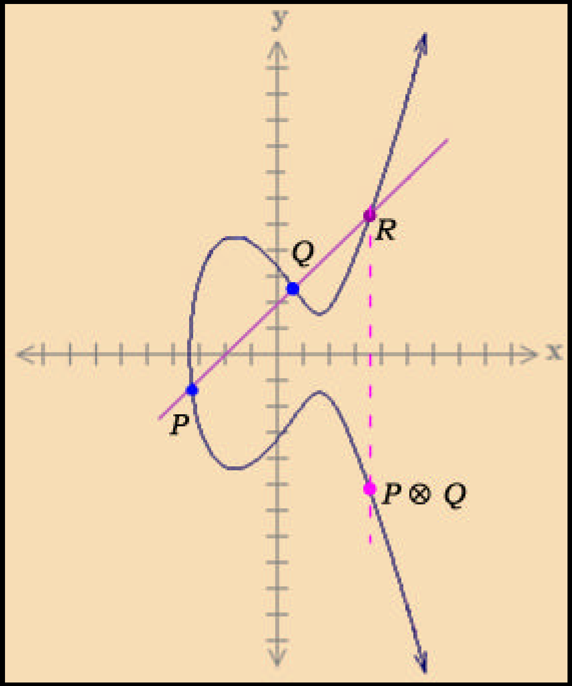
The general form of the elliptic curves:

$$y^2 + axy + by = x^3 + cx^2 + dx + e \qquad x, y, a, b, c, d, e \in \mathbf{F}$$

In case of **F** = Real Number Field

$$y^2 = x^3 + ax + b \qquad x, y, a, b \in \mathbf{R}$$

The (x,y) points, which satisfy the equation form the elliptic curve **E(F)**.



Points of the elliptic curve forms a *group* with a suitably chosen operation.

The **0** element is also a member of the group.

The **0** element is the neutral element of the group.

In cryptography the **finite field** based elliptic curves can be applied.

The Elliptic Curves

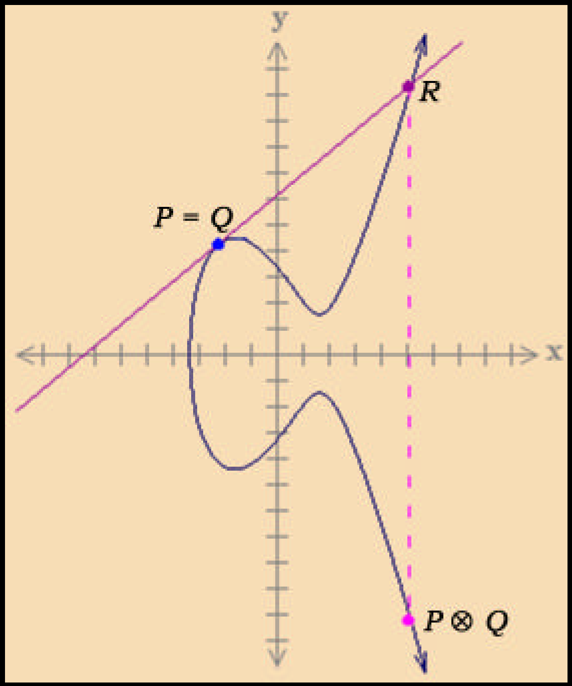
The general form of the elliptic curves:

$$y^2 + axy + by = x^3 + cx^2 + dx + e \qquad x, y, a, b, c, d, e \in \mathbf{F}$$

In case of **F** = Real Number Field

$$y^2 = x^3 + ax + b \qquad x, y, a, b \in \mathbf{R}$$

The (x,y) points, which satisfy the equation form the elliptic curve **E(F)**.



Points of the elliptic curve forms a *group* with a suitably chosen operation.

The **0** element is also a member of the group.

The **0** element is the neutral element of the group.

In cryptography the **finite field** based elliptic curves can be applied.

The Elliptic Curves

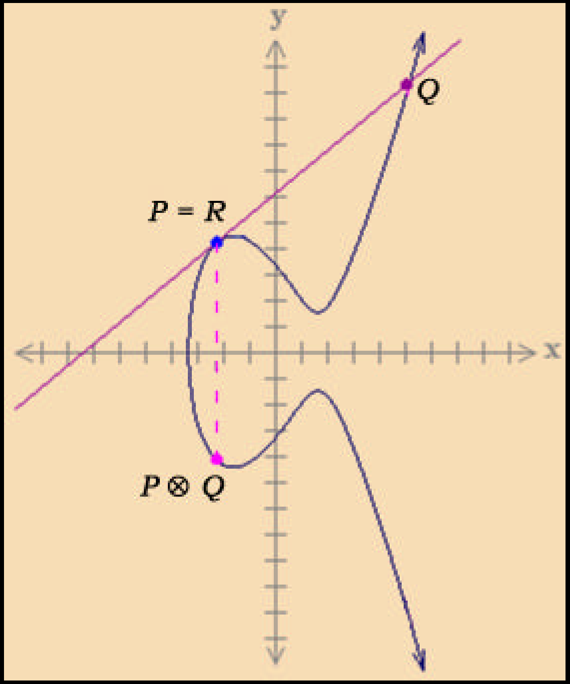
The general form of the elliptic curves:

$$y^2 + axy + by = x^3 + cx^2 + dx + e \qquad x, y, a, b, c, d, e \in \mathbf{F}$$

In case of **F** = Real Number Field

$$y^2 = x^3 + ax + b \qquad x, y, a, b \in \mathbf{R}$$

The (x,y) points, which satisfy the equation form the elliptic curve **E(F)**.



Points of the elliptic curve forms a *group* with a suitably chosen operation.

The **0** element is also a member of the group.

The **0** element is the neutral element of the group.

In cryptography the **finite field** based elliptic curves can be applied.

The Elliptic Curves

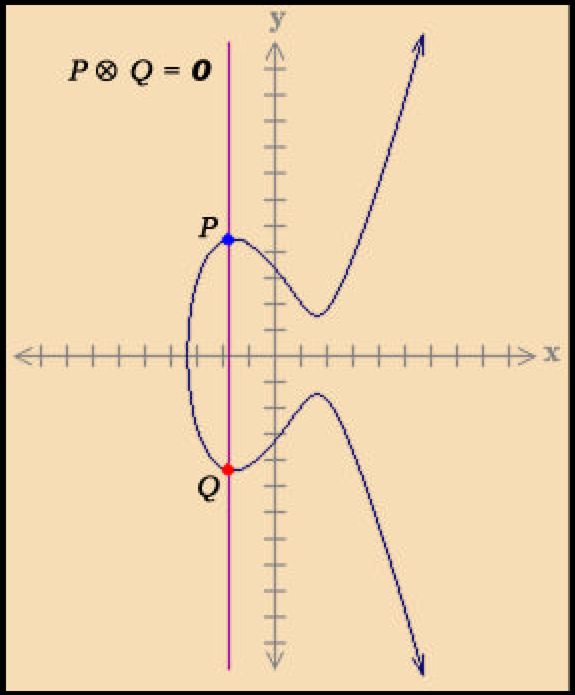
The general form of the elliptic curves:

$$y^2 + axy + by = x^3 + cx^2 + dx + e \qquad x, y, a, b, c, d, e \in \mathbf{F}$$

In case of **F** = Real Number Field

$$y^2 = x^3 + ax + b \qquad x, y, a, b \in \mathbf{R}$$

The (x,y) points, which satisfy the equation form the elliptic curve **E(F)**.



Points of the elliptic curve forms a *group* with a suitably chosen operation.

The **0** element is also a member of the group.

The **0** element is the neutral element of the group.

In cryptography the **finite field** based elliptic curves can be applied.

The Elliptic Curves

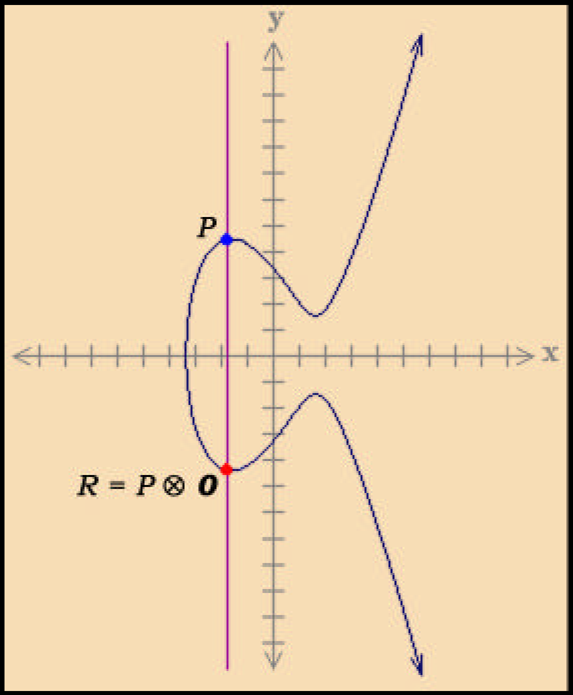
The general form of the elliptic curves:

$$y^2 + axy + by = x^3 + cx^2 + dx + e \qquad x, y, a, b, c, d, e \in \mathbf{F}$$

In case of **F** = Real Number Field

$$y^2 = x^3 + ax + b \qquad x, y, a, b \in \mathbf{R}$$

The (x,y) points, which satisfy the equation form the elliptic curve **E(F)**.



Points of the elliptic curve forms a *group* with a suitably chosen operation.

The **0** element is also a member of the group.

The **0** element is the neutral element of the group.

In cryptography the **finite field** based elliptic curves can be applied.

The Elliptic Curves

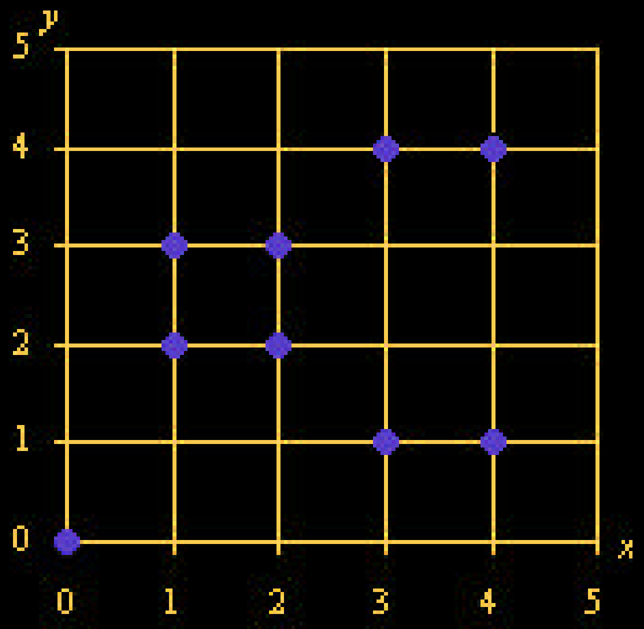
Elliptic curve over finite field:

$$y^2 + \mathbf{a}xy + \mathbf{b}y = x^3 + \mathbf{c}x^2 + \mathbf{d}x + \mathbf{e}$$

$$x, y, \mathrm{a}, \mathrm{b}, \mathrm{c}, \mathrm{d}, \mathrm{e} \in \mathbf{GF(q)}$$

In practice the elliptic curves over **GF(2n)** and **GF(p)** are of importance.

Example for an elliptic curve over finite field: $y^2 = x^3 - 2x^2 \; (mod\ 5)$



Modulo 5 Plan has 5*5 = 25 points.

The Elliptic curve has 10 points:
(0,0), (1,2), (1,3), (2,2), (2,3), (3,1), (3,4), (4,1), (4,4), **0**