A DISCUSSION OF ELLIPTIC CURVE CRYPTOGRAPHY AND CONFIGURABLE ECC SYSTEM DESIGN WITH APPLICATION TO DISTRIBUTED SIMULATION

P.H. ROBERTS and R.N. ZOBEL

Department of Computer Science
University of Manchester
Oxford Road
Manchester M13 9PL

Abstract: Distributed simulation, outside of the military area, necessarily operates over the internet, which implies the risk of many forms of attack. Current security systems offer limited protection because of the cost and complexity of using sufficient key lengths in existing public key encryption schemes. The use of the Discrete Logarithm Problem over elliptic curves defined over finite fields as a basis for trap-door based public key encryption (ECC) appears to offer improved performance with lower cost in terms of processor speed, memory requirement and processing time. This paper provides an outline of ECC and the complexities of a practical implementation of the technology. Some issues regarding choice of EC parameters, security, interoperability and performance are discussed. A proposal is made for a configurable ECC system architecture and the high-level design of a toolkit to enable the development of ECC systems is discussed. ECC cryptographic systems may be considered particularly suitable for supporting distributed interactive simulation, with its stringent timing requirements and particular security problems, with additional reference to mobile systems.

Keywords: Cryptography, Elliptic Curve, Distributed Simulation, ECC, System Architecture, Authentication

1. INTRODUCTION

Increasingly, distributed simulation is required for a variety of reasons, such as models running under differing operating systems, clock speeds or simulation environments, or for reasons of commercial, government or military secrecy. For cost reasons, many of these systems now have to use the internet to carry their intercommunication traffic. Consequently, the risk of lost or modified data, loss of secrecy and interference with simulation studies, makes it imperative to use some form of encryption.

Further, it is necessary to provide adequate authentication facilities for all intended participants in a shared distributed simulation environment.

1.1 Attacks

There are many forms of attack. Passive attacks concern listening, observing and collecting information. Active attacks include masquerade, replay, modification, and denial of service. Of these the last is the most dangerous, since the others can be defended against by using authentication and encryption techniques.

Denial of service and worse still, distributed denial of service, can be targeted or undirected random attacks, but are less likely to be targeted if it is difficult to identify who are the participants in a distributed simulation exercise and where on the network they are.

1.2 Security services

There are several important aspects of security which directly impact on the use of authentication and encryption for distributed simulation. They relate to maintaining privacy, strictly protecting data and messages, and system integrity during the set-up, simulation and post simulation analysis.

Further, sophisticated authentication procedures ensure that only bona fide participants are permitted to join a federation, whether active in simulations or just as observers.

1.2.1 Confidentiality

This concerns providing protection against passive attacks, such as acquisition of data by copying and traffic analysis for operational analysis. Encryption is often used to provide protection against such attacks. However, this may not give a complete guarantee of protection in respect of some currently employed encryption schemes.

1.2.2 Integrity

Maintaining system integrity is, at a lower level, concerned with ensuring that the contents of a message have not been changed. Full integrity concerns the entire message stream making up a communication, ensuring that no messages or parts of messages have disappeared, have been replayed and that no additional messages or parts thereof have been inserted by active attacks.

1.2.3 Authentication

Masquerade attacks occur when unauthorized participants assume the identity of those who are authorized. Authentication concerns ensuring that, at all times, the authorized principals are in fact who they claim to be.

1.2.4 Non-repudiation

This is not necessarily of particular interest to simulationists, but ensures that once a party has sent a communication, they are not able to deny having sent it, and that the receiving party can prove that the original party did indeed send the message.

1.2.5 Symmetric/Asymmetric key encryption

Discussed in detail later, this concerns the use of symmetric, shared encryption/decryption key, schemes with their associated problems of key distribution and management, and asymmetric, public key, systems.

1.2.6 Current limitations and susceptibility

The use of authentication and encryption for distributed simulation, and other related real-time activities, is limited by the additional time delay imposed by message construction, interchange and message encryption/decryption at each end of each interaction pair. With the use of real equipment and personnel the issue of mobile simulation arises and brings up the general issue of the use of mobile computing devices. First there is the issue of the physical security of small portable digital devices such as mobile phones, PDAs and smart cards. They are easily lost or stolen. Then there are the limited resources of such devices which limit the complexity and degree of security currently possible. For example processor power and memory size constrain the complexity of encryption/decryption algorithms. Additionally, there is limited bandwidth available in wireless networks resulting in transmission speed restrictions. Finally, there is the basic problem of the insecure nature of the radio medium, since it is a broadcast medium which may be received, decrypted, decoded and interpreted by any person with a suitable receiver.

2. ENCRYPTION

Many methods have been used over the centuries with varying degrees of success. However, with the advent of computers, life has become easier for the attacker. Cryptanalysis has become a sophisticated topic, available to security professionals, criminals and hackers alike. Consequently, existing systems are under threat of serious attack. Unconditional security is not a reality. Realistic conditions for computational security might be that the cost of breaking the cipher is more than the value of the encrypted data, and the time to break the cipher is longer than the useful lifetime of the encrypted data [Stallings W. 1995]. The latter condition can, in principle, be obviated by the use of time stamps. The first condition is less easy to achieve, because of novel backdoor approaches and clever mathematicians.

2.1 Classical or Symmetric Encryption

This usually involves three players. Two are the communicants (the principals) and the third is the attacker. The communicants use a publicly known encryption scheme and share a secret key. The important concepts are that the same secret key is used for both encryption and decryption, and usually, the decryption algorithm is simply the inverse of the encryption algorithm. Although potentially very efficient, there are three main problems with such schemes. The first is key distribution, which must be in itself secure; the second is key management, where the number of keys required in a system with a large number of principals does not scale well, as can be clearly seen in figure 1.

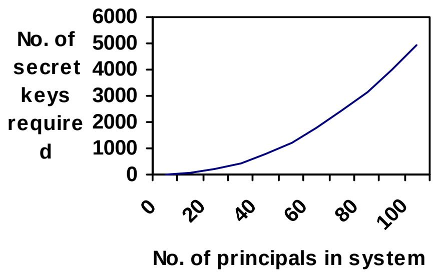


Figure 1: Number of keys required for symmetric encryption system

This can be a major problem for distributed simulation with a large number of players. Thirdly, symmetric key cryptographic systems suffer from a lack of provision for strong authentication, digital signatures and non-repudiation services. These problems can be solved through the use of Public Key encryption schemes, for a full discussion see [Stalling W. 1999].

2.2 Public Key or Asymmetric Encryption

Public Key Cryptosystems, first proposed theoretically by Diffie and Helman [Diffie et al, 1976], employ asymmetric key pairs, where each user has an individual key pair consisting of a publicly available encryption key, and a corresponding private decryption key. It should be computationally infeasible to compute the private key from knowledge of the corresponding public key and encryption algorithm, or from examples of encrypted and decrypted message pairs. The consequence of this is the need to find so called trap-door one-way functions (TOFs) which fulfill these criteria. A trap door is easy to open from the inside, but difficult from the outside without the key.

2.2.1 Public Key Cryptographic Services

Numerous different schemes, based on asymmetric key pairs, have been suggested to achieve the security services outlined in section 1 since Diffie and Helman's paper. There are three main classes of scheme,

- Digital signature schemes
- Encryption schemes
- Key Exchange schemes

The above three schemes are used as the building blocks of current network security systems.

2.3 The RSA Scheme

The RSA scheme [Rivest. et al, 1978], along with the Digital Signature Algorithm (DSA) and Diffie-Helman Key Exchange schemes are the most widely used and commercially accepted Public Key systems using the product of large prime integers as the basis of a family of TOFs. Their security is based on the difficulty of factoring large integers. However to provide adequate security it is currently necessary to use keys with at least 1024 bits, with encryption and decryption operations orders of magnitude slower than conventional symmetric cryptosystems. Thus their practical use in distributed simulation is limited. Additionally due to recent improvements in factoring algorithms the security per key bit provided by these schemes has been reduced. This means that as computer hardware power continues to increase, prohibitively large key sizes will become necessary to maintain current security levels.

3. ELLIPTIC CURVE CRYPTOGRAPHY (ECC)

3.1 Introduction and Motivation

Koblitz [Koblitz, N. 1987] and Miller [Miller, V. 1986] proposed the use of the discrete logarithm problem over a group of elliptic curve points, known as the ECDLP, as a basis for public key cryptographic schemes, as the difficulty of computing the ECDLP is much harder than factoring integers. Therefore the security offered per key bit by such schemes is greater and such schemes require smaller key sizes to provide comparable levels of security. Such schemes will also be more resilient to future improvements in computer hardware, assuming no significant improvements in the efficiency of algorithms for solving the ECDLP. Given that these schemes have now been under investigation for over 15 years such a dramatic breakthrough seems unlikely. A comparison of the key sizes required by ECC, Integer factorization (RSA) and traditional symmetric schemes is shown in table 1, adapted from [SEC 1, 2000].

Symmetric scheme	**ECDLP based scheme**	**Integer factorisation based scheme**
56	112	512
80	160	1024
112	224	2048
128	256	3078
192	384	7680
256	512	15360

Table 1: Comparison of key sizes

Smaller key sizes are beneficial in systems where keys must be transported across networks and stored on devices with limited memory. Additionally, smaller key sizes may result in faster execution timings for the schemes, which is beneficial to systems where real time performance is a critical fasctor.

3.2 Mathematical background

ECC schemes are based on the scalar multiplication of elliptic curve points and the computational intractability of the inverse operation, the Elliptic Curve Discrete Logarithm Problem (ECDLP).

3.2.1 Elliptic Curves

An elliptic curve **E(F)**, is the set of solutions, or points, which satisfy a Weierstrass equation, given by:

$$y^2z+a_1xyz+a_2yz^2 = x^3+a_3x^2z+a_4xz^2+a_5z^3$$

Where, a_i and the coordinates of each point are elements in a field **F**. For a full mathematical description see [Menezes, A. 1993].

3.2.2 Finite Fields

For a full introduction to the theory of fields and finite fields, see [Koblitz, N. 1987]. Two types of field are suitable for ECC, namely, characteristic two finite extension fields, where the Weierstrass equation can be simplified to,

$$\mathbf{y^2 + xy = x^3 + ax^2 + b}$$

and large prime characteristic finite fields, where the short Weierstrass form defines the curve,

$$\mathbf{y^2 = x^3 + ax + b}$$

In both cases the field chosen can be defined by its order, denoted **q**, and the chosen curve defined by the parameters **a** and **b** to the simplified Weierstrass equation.

3.2.3 Operations required by ECC

The scalar multiplication, or repeated addition, of EC points is the main operation required by ECC schemes, although other operations such as division may also be needed. For exact long integer word length mathematical arithmetic operations implied by such encryption/decryption systems are slow and possibly unique to this application area, since all rounding schemes are automatically excluded The addition of two EC points is illustrated in figure 2 below.

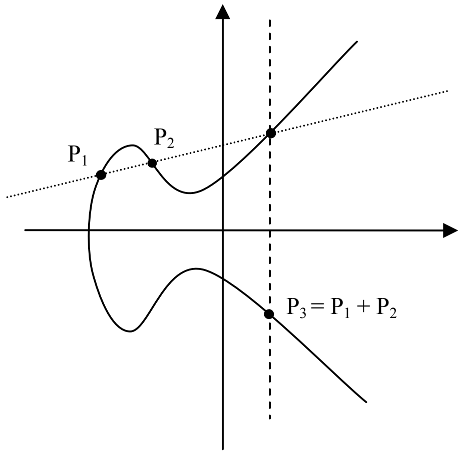


Figure 2: Illustration of adding two EC points.

As it is the scalar multiplication operation that dominates the actual execution timing of ECC schemes, its efficient implementation is crucial. The actual mathematics depends on the chosen curve and underlying field, see [Blake, I et al. 1999], however there is a clear hierarchy of underlying mathematical operations, as shown in figure 3 below.

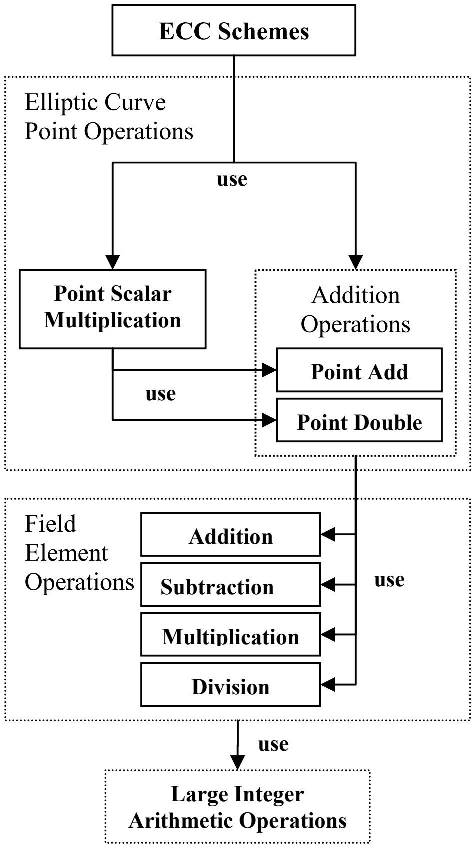


Figure 3: Hierarchy of required underlying mathematical operations

EC point compression techniques also require square root computation in the field. All the field operations require integer arithmetic, with operand lengths which may be considerably larger than typical computer word length. These low level integer arithmetic operations may be best implemented as hand coded assembler routines. Clearly, to efficiently implement ECC it is crucial to select the optimal algorithms at each level. These underlying algorithm choices depend first on the chosen curve and type of underlying finite field and then can often be optimised depending on the computing environment. Clearly in a distributed simulation environment the devices involved may have widely differing computing capabilities and architectures. Thus for optimal performance the underlying mathematical algorithms may need to be configured on a per device level.

3.3 ECC Standards

Recently a number of standards have been published each making various recommendations regarding the usage of ECC technology. Typically they include,

- Standardised versions of the common ECC schemes
- Pre-selected EC Parameter sets
- Standardised formats for the description of ECC keys, parameter sets and other cryptographic primitives.

There are currently 5 main standards, see [Groβschadl, J. 2002] for a recent survey, which essentially differ in how reserved they are in their recommendations. Figure 4 shows the compatibility between the standards, with the outer standards being, in general, less conservative and interoperable with the standards they encompass.

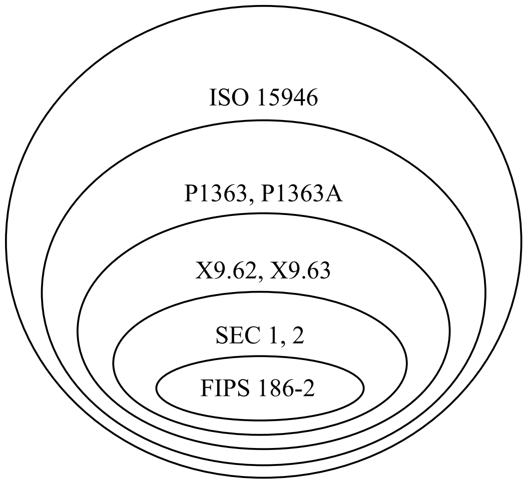


Figure 4: Interoperability between ECC Standards

4. ECC SYSTEM IMPLEMENTATION ISSUES

There are many issues to be discussed between the mathematical basis for an ECC scheme and a practical working ECC system. These are discussed in detail in the MSc thesis [Roberts P.H., 2004]. Below is not an exhaustive but indicative list of relevant issues. It will be seen from this that many decisions have to be made at several levels.

4.1 EC Parameters

ECC schemes operate on a sub-group of EC points on an elliptic curve. It should be appreciated that there are a very large number of possible curves and groups of points, which could be used. Some classes of curves contain mathematical structures, which lend themselves to more efficient implementation of the underlying mathematics, whilst others allow more efficient computation of the ECDLP and therefore are not cryptographically secure.

The issue of curve security will be discussed in more detail later, however in order to set up an ECC scheme it is necessary that **one** group of points is chosen. i.e. that all users use the **same** group of points on the same curve over the same finite field. This group of points can be specified by an EC parameter set, which can be defined as a 7-tuple,

(a, b, q, G, n, h, Fr)

where, **a** and **b** define the specific elliptic curve, **q** identifies the chosen underlying finite field, **G** is a base point which generates the sub-group of points on the chosen curve. These will be the actual points that will be used by the schemes, **n** gives the number of points in the specified sub-group and **h** gives the total number of points on the chosen curve.

Fr can be used to give an indication of the representation to use for the underlying field elements. This is important for characteristic two extension fields, where a number of different bases for field element representation are possible.

4.1.1 Parameter set selection

Parameter sets may be chosen from a public list of secure sets, however these may have been investigated by hackers. Alternatively, you may choose your own set of parameters. This involves selection of a finite field and generation of a curve, followed by selection of a suitable sub-group and optionally a field representation. The number of points on the curve can be checked as suitably secure using Schoof's algorithm [Schoof, 1985].

A further technique is to use a random number generator to select a curve, over a finite field, using an arbitrary seed string, which is also then included in the parameter set for later verification. This attempts to avoid the possibility of selecting a curve with a hidden cryptographic weakness, which would reduce the effective security level.

4.2 Security

The level of security offered by an ECC scheme is largely determined by the difficulty of solving the ECDLP over the group of EC points used with the scheme. This should be a prime-order cyclic sub-group of points on a suitably secure elliptic curve. It is essential that only groups where the ECDLP is computationally infeasible be used. Table 2 [ANSI X9.62, 2001], reproduced below, shows that reasonable sizes of n give very long time periods for solving the ECDLP, using the Rho algorithm [Pollard, 1987].

There are a number of special classes of curves where algorithms of sub-exponential complexity for solving the ECDLP are known, see [Roberts. P.H 2004]. These curves should be avoided, since they will exhibit much lower computing time estimates.

Bit size of n	√ (π / 4)	MIPS years
160	2^{80}	$8.5*10^{11}$
180	2^{93}	$7.0*10^{15}$
234	2^{117}	$1.2*10^{23}$
354	2^{177}	$1.3*10^{41}$
426	2^{213}	$9.2*10^{51}$

Table 2: Computing time estimates for solving the ECDLP for various values of n

It should be noted that it is the order of the base point of a parameter set **n**, which determines the difficulty of solving the ECDLP and the order of the underlying field **q** that determines the size of the keys generated. It should be noted that in a security system requiring more than one level of security, a separate parameter set is needed for each security level and that each user will require a separate key-pair for each parameter set currently in use in the system.

4.3 Interoperability

To achieve interoperability at any level, i.e. system wide or inter-system, it is necessary that all participating users of a scheme have key-pairs based on the same shared EC parameter set/s. In practice there are two additional requirements,

- Use of standardized descriptions of cryptographic primitives
- Use of standardized versions of the common ECC schemes and protocols

For performance and security reasons, the possibilities for interoperation may be considerably reduced.

4.4 Performance

Parameter set selection has a large bearing on performance, particularly with respect to the algorithms available for the implementation of the underlying mathematical operations. Optimization by efficient coding in assembler or with hardware implementation may well be necessary for real-time applications. Network performance depends on common standards for formatting and encoding, plus of bandwidth limitations. Ultimately, there will always be a difficult three sided compromise between interoperability, performance and security, which is application dependant, as illustrated in figure 5.

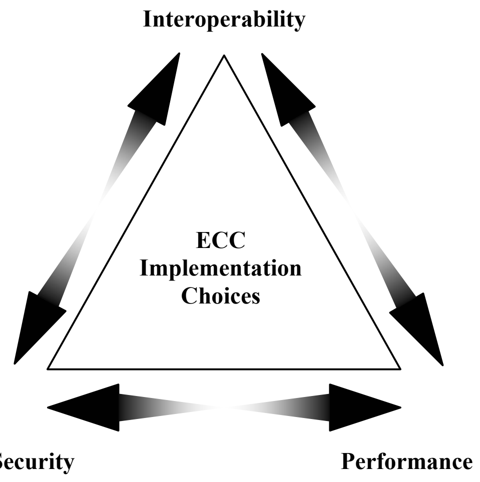


Figure 5: ECC implementation tradeoffs

5 IMPLEMENTATION APPROACHES

Optimal performance can be obtained by designing the complete ECC system tailored to the individual needs of a specific application, but this is usually expensive and requires expert knowledge. A second alternative is a two level tailored approach, encompassing first the perspective of the application and then second the perspective of the device/platform on which the ECC services will run:

5.1 Application Level Issues

These are most naturally considered from an application level perspective:

- Desired level of security and selection of appropriate EC parameter sets
- Conformance to standards
- Selection of cryptographic schemes
- Formats for network transfer of keys and other cryptographic primitives

5.2 Device Level Issues

Device level issues concern the selection and optimization of the available underlying mathematical algorithms, based on the parameter set/s chosen at the application level, to give the best performance for the computing resources available on the device, which involves:

- Selection of internal field element representation and associated mathematical algorithms.
- Selection of internal point representation and selection and optimization of point addition algorithms.

- Selection and optimization of a scalar multiplication algorithm taking side channel attacks into account.
- Utilization of any special purpose field or integer arithmetic hardware available on a specific device.

5.3 Toolkit Approach

Such tailored approaches still imply that a new ECC system must be built from scratch for each new application that wishes to use ECC. This is inevitably expensive in terms of application development. A third approach is therefore to provide a toolkit of re-usable ECC algorithm components, which can be plugged together in a framework to build a customized ECC solution.

5.4 ECC System Architecture Proposal

Shown in figure 6 is the high level architecture of such a two layered toolkit approach.

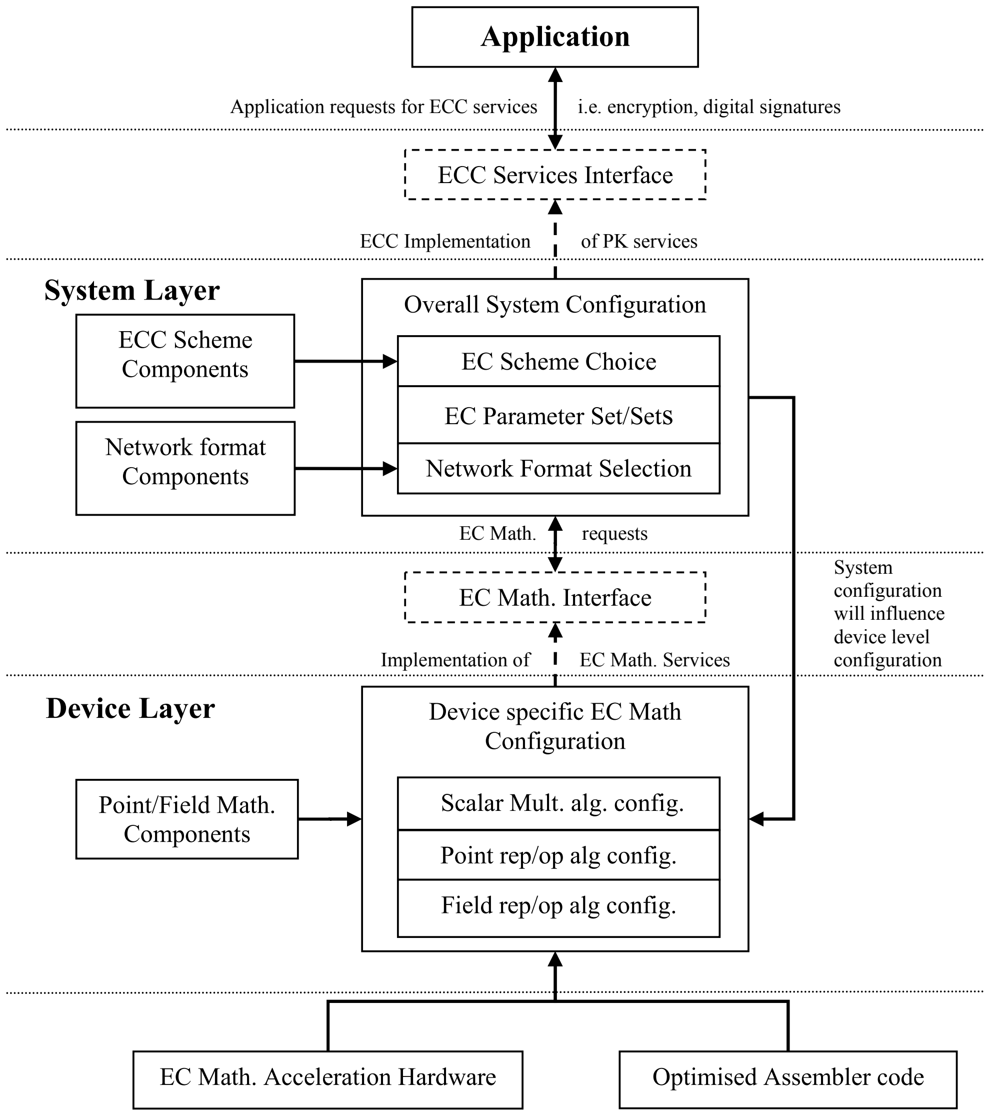


Figure 6: High level architecture of a two layered toolkit approach

This involves two important interfaces:

- An ECC Services Interface between the System Layer and the Application.
- An EC Mathematics Interface between the Device Layer and the System Layer.

The ECC Services interface separates the application specific system level implementation choices from the cryptographic services provided by the configured ECC system. The EC Mathematics interface separates the configured ECC system from the algorithm choices used to implement the underlying mathematical operations on a specific device. This architecture enables application level issues, which affect the entire system, to be resolved on a system wide basis, whilst enabling the dominant performance factor, namely, the implementation of the underlying mathematics, to be resolved on a per device basis. This separation makes for a valuable feature of the system, allowing the user to concentrate on the important issues.

6 JAVA ECC TOOLKIT DESIGN AND IMPLEMENTATION

In the next sections the design of a configurable ECC Toolkit is briefly discussed, for a full description see [Roberts, P.H. 2004]

6.1 Overall Toolkit Structure

A configured ECC system will have a number of selected ECC schemes. When an application requests a cryptographic service the system will need to know which ECC keys and EC parameter set to use. Additionally the schemes will need to use EC mathematics services specific to the EC parameter set being used. Figure 7 illustrates the high level structure of such a configured ECC System

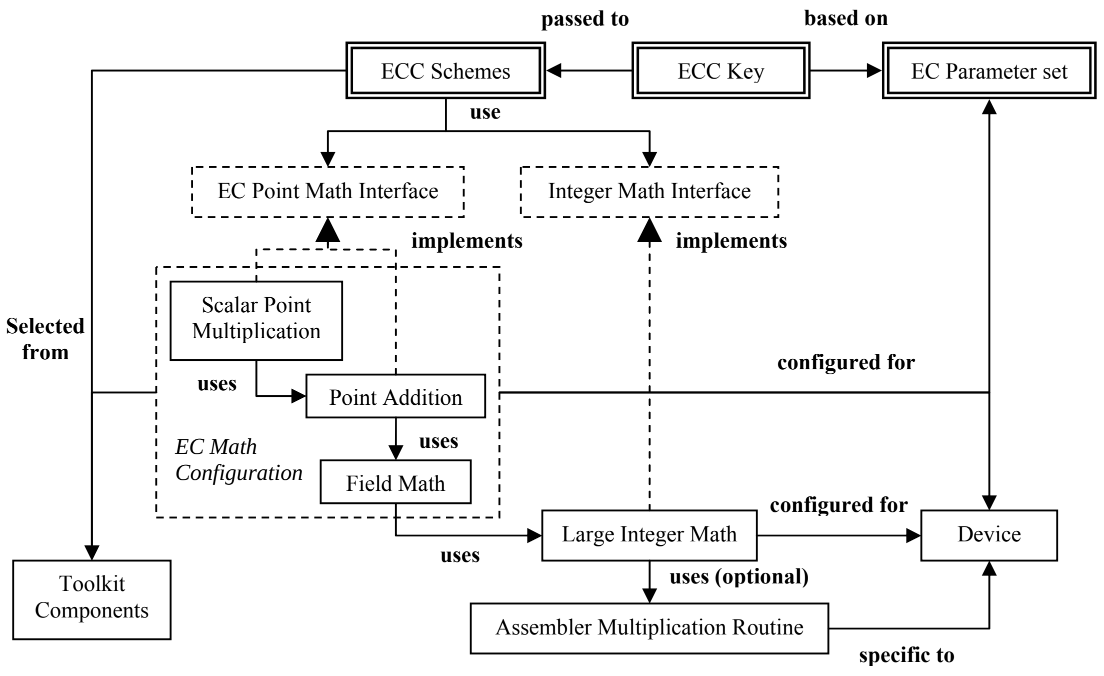


Figure 7: ECC System structure

6.2 ECC Keys and EC Parameter sets

An ECC key pair is generated from and based on a single EC parameter set. Therefore when an application specifies the use of a certain key, for that service request, the ECC scheme can imply the parameter set to use from the specified key. The EC parameter set can be encapsulated in the ECC key and therefore hidden from the application.

6.3 Underlying Math Implementation

As shown in figure 7, the implementation of the underlying mathematics services required by the ECC schemes is dependant on both the device the scheme is being run on and the EC parameter set specified in the cryptographic service request. If math implementations are linked to EC parameter sets, then, when an application requests a cryptographic service, the ECC scheme can imply the math implementation to use from the EC parameter set that the supplied ECC key is based on. This linkage of EC parameter sets to the appropriate mathematical implementations is best achieved when an EC parameter set is imported on a given device.

This has three main benefits:

- Linkage is done once for each EC parameter set
- Linkage is not necessary when a cryptographic service is requested
- Math implementation can be configured on a per device **and** an EC parameter set basis.

6.4 Additional Application Services

As well as requesting the cryptographic services of the ECC schemes, applications will also need to generate ECC keys from EC parameter sets and encode/decode these keys for network transport between system users.

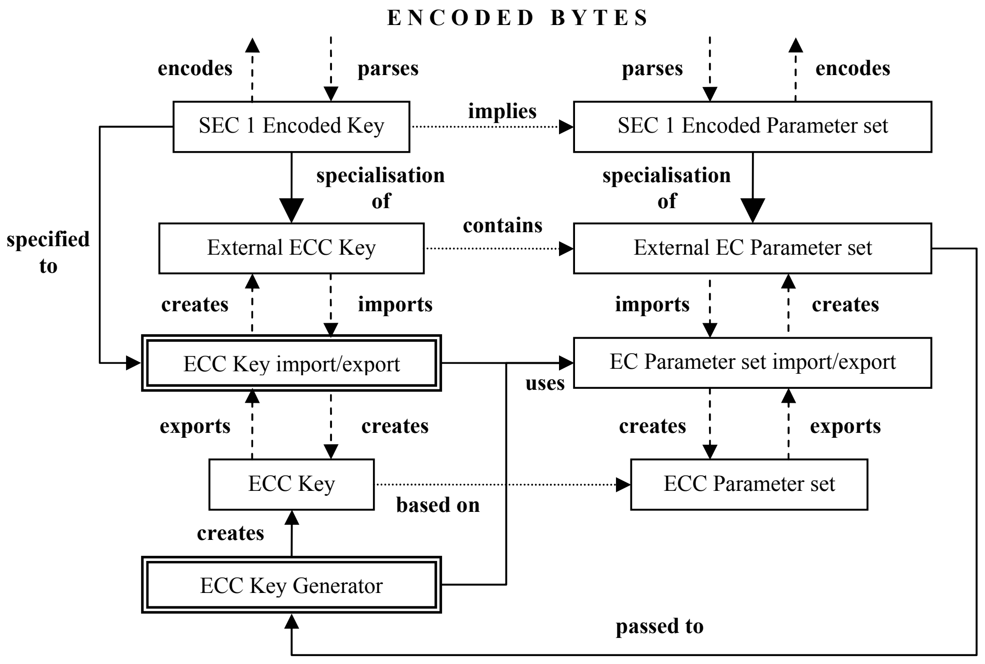


Figure 8: Generating, importing and exporting ECC keys and parameter sets

6.4.1 Importing/Exporting ECC Keys

As illustrated in figure 8, ECC key import/export can be achieved by an ECC key import/exporter object, which converts between internal and external ECC key descriptions. Internal ECC keys are linked to internal EC Parameter sets and therefore linked to appropriate math implementations. Only these internal keys can be passed by an application to the ECC schemes via a cryptographic service request.

The external key objects are math implementation independent and encapsulate a network encoding format. This allows numerous different network encoding formats to be used and allows different math implementations to be used on separate devices for the same keys and parameter sets.

6.4.2 Generating ECC Key Pairs

As also shown in figure 8, key pair generation can be achieved via a key generation object by specifying an external EC parameter set description. The key generator then imports the parameter set, links to a math implementation and having generated the key pair, links the generated keys to the EC parameter set they are based on and thus to the appropriate math implementation.

6.5 Validation

Transporting keys, parameter sets and cryptographic primitives across insecure networks introduces the risk of transmission errors and potential opportunities for hackers, therefore validation needs to be performed when these primitives are imported onto a device.

6.6 Generic Public Key Cryptography Services Interface

The ECC Services Interface of figure 8 needs to allow the application to request ECC versions of the three common public key cryptographic services,

- Computation of digital signatures
- Encryption/decryption of data
- Computation of secret key values through key exchange schemes.

In addition the interface needs to allow an application to generate and import/export keys, if and when they are needed. The requirements are in fact the same as if any public key cryptography system were being implemented. Figure 9 illustrates a generic public key cryptographic services interface.

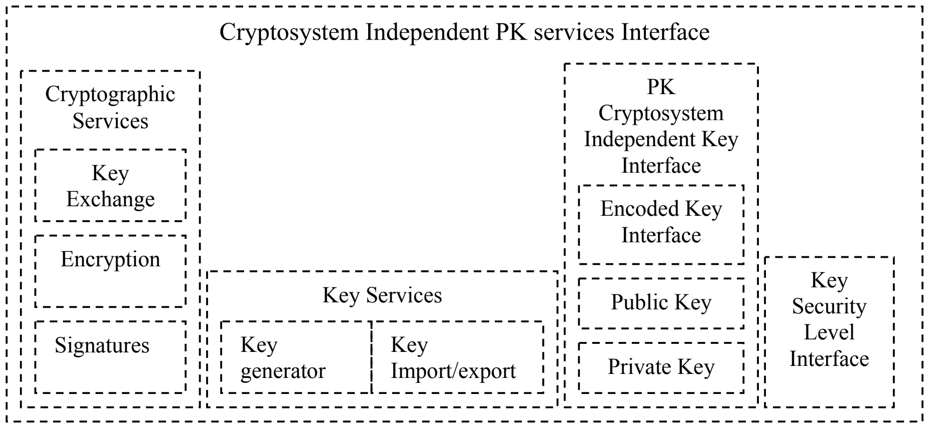


Figure 9: Generic Public Key Services interface

This interface is sufficiently abstract to hide which underlying public key cryptosystem is being used from the applications using the services it provides, thus allowing the underlying cryptosystem to be changed without needing to inform the application. Further, this perhaps provides some additional protection, since a direct lack of knowledge of the underlying cryptosystem makes the task of the attacker significantly more difficult.

If ECC is the underlying Public Key cryptosystem, the selected ECC schemes would implement the cryptographic services section of the interface with the ECC key generator and ECC key import/export objects implementing the key services section. The internal and external ECC key would implement the generic Public Key Interface section with the external EC Parameter objects implementing the Key Security Level interface.

7. CONCLUSIONS AND IMPLICATIONS FOR SIMULATION

It is clear that elliptic curve cryptography is complex and there are a number of practical issues to be resolved when integrating the technology into a security system. For successful use of distributed simulation, whether for discrete event, continuous or mixed, and whether for large numbers of participants (federates) or smaller federations, or whether for PCs or supercomputers, there is a real need for adequate protection against attack from any quarter. One must consider the performance aspects in respect of the time penalty for the use of authentication and encryption/decryption in real-time applications. This paper has provided a brief view into the complex topic of elliptic curve cryptography. It has enormous potential for industrial and commercial use both inside and outside of computer simulation, particularly where cost dictates the use of the internet, such as in the increasingly important area of mobile applications.

8. REFERENCES

ANSI X9.63, 2001. “Public Key Cryptography for the Financial Services Industry: Key Agreement and Key Transport Using Elliptic Curve Cryptography”.

Blake, I; Seroussi, G and Smart, N. 1999. Elliptic Curves in Cryptography. Cambridge University Press. ISBN: 0-521-65374-6.

Groβschadl, J; Kamendje, G; Oswald, E and Posch, R. 2002. “Elliptic curve Cryptography in Practice – The Concept of the Austrian Citizen Card for e-Government Applications”. SSGRR 2002

Koblitz, N. 1987. “A Course in Number Theory and Cryptography”. Springer-Verlag. ISBN: 0-387-96576-9

Menezes, A. 1993. Elliptic Curve Public Key Cryptography. Kluwer Academic Publishers. ISBN: 0-7923-9368-6

Pollard, J. 1978. “Monte Carlo Methods for Index Computation Mod p”. Mathematics of Computation, vol. 32, p918-924.

Rivest, R; Shamir, A and Adleman, L. 1978. “A Method for obtaining Digital Signatures and Public Key Cryptosystems”. Communications of the ACM, Feb.

Roberts, P.H. 2004. MSc Thesis, University of Manchester, U.K.

Schoof, R. 1985. “Elliptic Curves over Finite Fields and the Computation of Square Roots Mod p”. Mathematics of Computation, vol. 44, p483-494.

Stallings, W. 1995. Network and Internetwork Security Principals and Practice. Prentice Hall. ISBN 0-13-180050-7.

Stallings, W. 1999. Cryptography and Network Security Principals and Practise (2nd Ed.) Prentice Hall. ISBN: 0-13-869017-0.

Stinson, D. 2002. Cryptography Theory and Practice (2nd Ed). Chapman and Hall/CRC. ISBN 1-58488-206.

Diffie, W and Helman, M. 1976. New Directions in Cryptography. IEEE Transactions on Information Theory. 22.

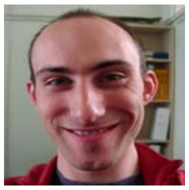
SEC 1. 2000. "Elliptic Curve Cryptography". Standards for Efficient Cryptography Group

SEC 2. 2000. "Recommended Elliptic Curve Domain Parameters". Standards for Efficient Cryptography Group.

ACKNOWLEDGEMENTS

The authors wish to thank the EPSRC and the Computer Science Department of the University of Manchester for their support of the work described in this paper, which was carried out during the 2002/2003 academic year. Richard Zobel retired from the University of Manchester in October 2003.

AUTHOR BIOGRAPHIES



PAUL-GEORGE ROBERTS has a BSc in Computer Science (2002) and has recently finished an MSc in Advanced Computer Science (2004), both at the University of Manchester, UK.

RICHARD ZOBEL graduated in Electrical Engineering from London University in 1963. His first experience of simulation was obtained during 1962-66 at Sperry Gyroscope whilst working on naval surface to air missiles, using mainly valve analog computers. His Ph.D., obtained in 1970 at Manchester University, concerned hybrid analog-digital computing. As Lecturer and Senior Lecturer he became involved in digital signal processing, instrumentation and design environments with special emphasis on the simulation aspects of real-time embedded systems. He is a Committee Member and former Chairman of the United Kingdom Simulation Society (UKSim), Former Secretary of the European Federation of Simulation Societies (EUROSIM), and was a European Director of SCSI, the Society for Computer Simulation International. His current research interests concern distributed simulation for non-military applications, model re-use, distributed simulation model databases, issues of verification and validation of re-useable simulation models and security for distributed simulation under commercial network protocols. He is now retired, but still very active.