Embedded Computing Systems:

Applications, Optimization, and Advanced Design

Mohamed Khalgui
Xidian University, China

Olfa Mosbahi
University of Carthage, Tunisia

Antonio Valentini
O3neida Europe, Belgium

Information Science
REFERENCE

Managing Director: Lindsay Johnston
Editorial Director: Joel Gamon
Production Manager: Jennifer Yoder
Publishing Systems Analyst: Adrienne Freeland
Assistant Acquisitions Editor: Kayla Wolfe
Typesetter: Erin O'Dea
Cover Design: Jason Mull

Published in the United States of America by
Information Science Reference (an imprint of IGI Global)
701 E. Chocolate Avenue
Hershey PA 17033
Tel: 717-533-8845
Fax: 717-533-8661
E-mail: cust@igi-global.com
Web site: http://www.igi-global.com

Library of Congress Cataloging-in-Publication Data

Embedded computing systems : applications, optimization, and advanced design / Mohamed Khalgui, Olfa Mosbahi and Antonio Valentini, editors.
pages cm
Summary: "This book brings together theoretical and technical concepts of intelligent embedded control systems and their use in hardware and software architectures by highlighting formal modeling, execution models, and optimal implementations"--Provided by publisher.
Includes bibliographical references and index.
ISBN 978-1-4666-3922-5 (hardcover) -- ISBN 978-1-4666-3923-2 (ebook) -- ISBN 978-1-4666-3924-9 (print & perpetual access) 1. Embedded computer systems. I. Khalgui, Mohamed. II. Mosbahi, Olfa, 1976- III. Valentini, Antonio, 1980-
TK7895.E42E564 2013
006.2'2--dc23
2012051534

British Cataloguing in Publication Data
A Cataloguing in Publication record for this book is available from the British Library.

Chapter 20
Securing Embedded Computing Systems through Elliptic Curve Cryptography

Elisavet Konstantinou
University of the Aegean, Greece

Yannis C. Stamatiou
University of Patras, Greece

Panayotis E. Nastou
University of the Aegean, Greece

Christos Zaroliagis
University of Patras, Greece

ABSTRACT

Embedded computing devices dominate our everyday activities, from cell phones to wireless sensors that collect and process data for various applications. Although desktop and high-end server security seems to be under control by the use of current security technology, securing the low-end embedded computing systems is a difficult long-term problem. This is mainly due to the fact that the embedded systems are constrained by their operational environment and the limited resources they are equipped with. Recent research activities focus on the deployment of lightweight cryptographic algorithms and security protocols that are well suited to the limited resources of low-end embedded systems. Elliptic Curve Cryptography (ECC) offers an interesting alternative to the classical public key cryptography for embedded systems (e.g., RSA and ElGamal), since it uses smaller key sizes for achieving the same security level, thus making ECC an attractive and efficient alternative for deployment in embedded systems. In this chapter, the processing requirements and architectures for secure network access, communication functions, storage, and high availability of embedded devices are discussed. In addition, ECC-based state-of-the-art lightweight cryptographic primitives for the deployment of security protocols in embedded systems that fulfill the requirements are presented.

DOI: 10.4018/978-1-4666-3922-5.ch020

INTRODUCTION

Today we are witnessing a proliferation of all kinds of inexpensive, portable computing and communication devices with complex versatile wireless connection capabilities. This has as a consequence that Internet is accessible from everywhere by anyone who carries such devices. Internet services proliferate, accordingly, offering services of increasing sophistication and coverage of user needs. However, this ubiquitous existence of devices and services, which exchange volumes of, possibly, sensitive user data and information has given rise to an unprecedented demand for security measures capable of protecting users and service providers alike.

Despite the enhancements in memory and speed capabilities of devices, which came through the technological advances in chip manufacturing processes, most of the portable wireless devices in the market today (Smart phones, VoIP phones, portable computers etc.) do not have sufficient resources for the execution of computationally expensive, multi-step cryptographic protocols essential for the security of the users. In view of the resource limitations of wireless devices modern mobile network protocols involve the heavy use of lightweight private key data encryption algorithms as well as *Elliptic Curve based* public key protocols.

The main objective of this chapter is to discuss the basic principles of the cryptographic primitives and protocols employed for the security of *resource limited devices*, which may be generally seen as belonging to the general class of *embedded systems*. A central theme of our discussion is the mathematical construct of Elliptic Curves and its applications to cryptography. Elliptic Curve Cryptography, or ECC for short, offers an attractive alternative to the classical public key cryptography protocols such as RSA (Rivest, Shamir & Adleman, 1978) and ElGamal (ElGamal, 1985). One of the main advantages of ECC is that ECC-based protocols use smaller key sizes than traditional cryptosystems for achieving the same security levels. For instance, an ECC system with a key size of 160 bits is roughly equivalent, in terms of security, to an RSA system with a key size of 1024 bits. As the key size is much smaller, the requirements in space and memory are also small, rendering ECC an excellent candidate for implementation in embedded devices.

The chapter is organized in three parts. The first part presents some of the most frequently employed cryptographic primitives and protocols, which include block and stream ciphers, private and public key ciphers, digital signatures and key exchange. The second part is focused on Elliptic Curves and ECC and presents the basic definitions and primitives. The third part builds on the first and second parts and presents real world security protocols for embedded devices that employ the primitives discussed in these parts.

Given the space constraints, our aim is not to provide an in depth coverage on all issues pertaining to embedded systems security, but to raise awareness in a (possibly) non-expert audience to security solutions and provide pointers for more extensive information.

A BRIEF INTRODUCTION TO CRYPTOGRAPHIC PRIMITIVES

We briefly review in this part the basic cryptographic primitives and protocols. For more in-depth information on the concepts discussed in this chapter, the reader may consult the excellent book (Stallings, 1999).

A message to be subjected to encryption is called the *plaintext* or *cleartext*. *Encryption* is the process that transforms the message into a form so that it cannot be understood by parties who do not possess a special *key*. The transformed message is called *ciphertext*. *Decryption* is the process of transforming back the ciphertext into its original (plaintext) form. The science of keeping messages secure from being understood by unauthorised

persons is *cryptography*. *Cryptanalysis* is the process of discovering the plaintext from the ciphertext without initial knowledge of the key.

A *cipher (cryptographic algorithm)* is a process for the encryption and decryption of messages. *Symmetric-key* algorithms use the same, secret key for encryption and decryption while *public-key* algorithms are designed in such a way that the key used for encryption is publicly available and different from the secret key used for decryption which is private. Moreover, the decryption key cannot be calculated in any reasonable amount of time from the encryption key (this is an essential requirement since the encryption key is publicised). Using the public key, one may send a message to the owner of the key while the decryption of the message is only possible by the owner of the public key, through the use of the private (decryption) key. It is assumed that the attackers have total access to the communications channel between the sender and the receiver.

In what follows, we present the basic encryption and decryption methods and algorithms, both for public and private key encryption systems, along with a comparative study of their relative advantages and disadvantages.

Block and Stream Ciphers

Symmetric-key block ciphers are the most prominent and important elements in many cryptographic systems. Individually, they provide confidentiality. As a fundamental building block, their versatility allows construction of pseudorandom number generators, stream ciphers, MAC (Message Authentication Code) and hash functions. They may, furthermore, serve as a central component in message authentication techniques, data integrity mechanisms, authentication protocols and symmetric-key digital signature schemes.

With few exceptions the best measure of security for practical ciphers is the complexity of the best currently known attack. Various aspects of such complexity are *data complexity*, *storage complexity* and *processing complexity*. The attack processing complexity is the dominant of these. When parallelization is possible, processing complexity may be divided across many processors reducing attack time. Given a data complexity of 2^n, an attack is always possible, since these many different n-bit blocks completely characterize the encryption function for a fixed n-bit key. Similarly, given a processing complexity of 2^k an attack is possible by exhaustive key search. Thus as a minimum, the effective key size should be sufficiently large to preclude exhaustive key search, and the block size sufficiently large to preclude exhaustive data analysis. A block cipher is considered computationally secure if these conditions hold and no known attack has both data and processing security significantly less than 2^n and 2^k respectively.

For symmetric-key block ciphers, data complexity is beyond the control of the adversary and is passive complexity. Processing complexity is active complexity that typically benefits from increased resources. A cipher provides *perfect secrecy* (unconditional security) if the ciphertext and plaintext blocks are statistically independent. There are a number of criteria for evaluating block ciphers and modes of operations that include the following: *Estimated security level*, *Key size*, *Throughput*, *Block size*, *Complexity of cryptographic mapping*, *Data expansion*. It is often desirable, and often mandatory, that encryption does not increase the size of plaintext data. Homophonic substitution and randomized encryption techniques result in data expansion, and *Error propagation*.

A *block cipher* encrypts plaintext in fixed-size n-bit blocks (often n=64). For messages exceeding n bits, the simplest approach is to partition the message into n-bit blocks and encrypt each separately. This electronic-codebook (ECB) mode has disadvantages in most applications, motivating other methods of employing block ciphers (*modes of operation*) on larger messages. The four most common modes are ECB, CBC, CFB and OFB.

A *product cipher* combines two or more transformations in a manner intending that the resulting cipher is more secure than the individual components. A *substitution-permutation* (SP) *network* is a product cipher composed of a number of stages involving substitutions and permutations. An *iterated block cipher* involves the sequential repetition of an internal function called *round function*. Parameters include the number of rounds *r*, the block bit size *n*, and the input key *K* from which *r subkeys* K_i (round keys) are derived. For invertibility (allowing unique decryption), for each value K_i the round function is a bijection on the round's input. A *Feistel cipher* is an iterated cipher mapping a 2*t*-bit plaintext (L_0, R_0) to a ciphertext (R_r, L_r), through an *r*-round process where $r \geq 1$. For $1 \leq i \leq r$, round *i* maps $(L_{i-1}, R_{i-1}) \rightarrow (L_i, R_i)$ as follows: $L_i = R_{i-1}$, $R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$, where each *subkey* is derived from the cipher key *K*. The function *f* of the Feistel Cipher may be a product cipher, though *f* itself needs not to be invertible to allow inversion of the Feistel cipher. Typically in a Feistel cipher $r \geq 3$ and often *r* is even. The Feistel structure specifically orders the ciphertext output as (R_r, L_r) rather than (L_r, R_r); the blocks are exchanged from their usual order after the last round. Decryption is thereby achieved using the same *r*-round process but with subkeys used in reverse order, K_r through K_1. For example, the last round is undone by simply repeating it. A notable example of a Feistel Cipher is the historic DES (DES, 1977).

Today, many modern cryptographic protocols employ the A*dvanced Encryption Standard* or AES (Daemen & Rijmen, 2002) which replaced DES. This block cipher is composed of a substitution-permutation network (unlike DES which is Feistel cipher) and it is considered as one of the fastest ciphers for both hardware and software implementations. AES has a 128-bit block size and a key of variable of 128, 192, or 256 bits. The AES cipher operates on a 4×4 column-major order byte matrix, which is termed the *state* and the majority of AES computations are performed within a specially chosen finite number field (Galois field). The computations are performed in a series of iterations (whose number depends on the key length) composed of certain non-linear transformations taking into account the input as well as the key bits. Decryption is obtained by applying the same iterations in reverse order using the same key with which data was encrypted.

Stream ciphers are another important class of encryption algorithms. They encrypt individual characters (usually in binary digits) of a plaintext message one at a time, using an encryption transformation which varies with time. By contrast, block ciphers, as we have seen above, tend to simultaneously encrypts groups of characters of a plaintext message using a fixed encryption transformation. In real world, block ciphers seem to be more general and stream ciphers seem to be easier to analyze mathematically. There is a large body of theoretical work on the analysis and design of stream ciphers as they have been used by the world's militaries since the invention of electronics. This, however, gradually changed since the last two decades numerous theoretical papers have been written on block cipher design principles. Otherwise the differences between stream ciphers and block ciphers are in implementation. Stream ciphers that only encrypt and decrypt data one bit at a time are not really suitable for software implementation. Block ciphers can be easier to implement in software, because they often avoid time-consuming bit manipulation and they operate on data in computer-sized blocks. On the other hand, stream ciphers can be more suitable for hardware implementation because they can be implemented very efficiently on silicon.

Public Key Encryption Algorithms

Public key encryption algorithms dictate the use of two keys: a private and a public one. Public key encryption schemes can be defined over a suitably chosen group. RSA (Rivest, Shamir & Adleman, 1978) has been one of the most popular

public-key algorithms for encryption and digital signature applications due to its simplicity and speed of operation. The security of RSA relies on the presumed computational intractability of the integer factoring problem on a suitably defined multiplicative group. Elliptic Curve based encryption schemes, on the other hand, rely on suitably defined additive groups of points on the two dimensional integer grid. In this section we focus on conventional public key schemes while Elliptic Curve based ones are discussed in a separate section.

For the RSA scheme, to generate the two required keys (private and public) we first choose two large random prime numbers p and q of about equal number of digits and compute their product $n=pq$. We then choose at random the encryption key e, making sure that e and $(p-1)(q-1)$ are relatively prime, i.e., their greatest common divisor is equal to 1. Finally, we compute the decryption key d so that e and d are *modular inverses*, i.e., $d=e^{-1} \bmod ((p-1)(q-1))$. Note that d and n are relatively prime. Then e and n comprise the public key and d is the private key. Now p and q may be discarded but *not* revealed since their knowledge leads to the recovery of the decryption key d.

To encrypt a message m, we first divide it into blocks smaller than n. For example, if both p and q are 100-digit primes, then n will have at most 200 digits. Thus, if our message is longer than n, it will be divided into blocks with less than 200 digits. If we must encrypt a fixed number of blocks, we can pad them with few zeros on the left to ensure that they will always be less than n. The encrypted message c, will contain roughly equally sized message blocks c_i. The encryption formula is simply $c_i=m_i^e \bmod n$. Note that modular exponentiation can be performed fast by repeated squaring techniques. Now to decrypt a message, we take each encrypted block c_i and compute $m_i=c_i^d \bmod n$. Since $c_i^d=m_i^{ed}=m_i m_i^{k(p-1)(q-1)}=m_i\ 1=m_i$ all mod n, the original message can be recovered. Note that the original message might, as well, have been encrypted with d and decrypted with e and this observation has many applications in electronic signature applications.

DSA (Kravitz, 1993) is a public-key digital signature algorithm suitable for digital signature applications and it is part of the Digital Signature Standard (DSS). DSA uses the following parameters and operations:

1. An L-bit prime number p, where $512 \le L \le 1024$ and $L=64k$, $k=1, 2, \ldots$,
2. a 160-bit prime number q, which is a factor of p-1,
3. a number g, given by $g=h^{(p-1)/q} \bmod p$ where $h<(p-1)$ such that $h^{(p-1)/q} \bmod p > 1$,
4. a number x such that $x<p$, and
5. a number y, given by $y=g^x \bmod p$.

The DSA scheme also uses a *hash function* H(m). A hash function is a cryptographic construct that takes as input a block of data of arbitrary size (also called the «message») and outputs a fixed-size bit string which is the hash value (also called the «digest») of the input. The main property of hash functions is that any change of the input value results in (possibly) many changes in the hash value.

The first three parameters p, q and g of DSA are publicly available and can be shared among many users of a network, while x and y denote the private and public keys respectively. Let m be a message. In what follows we will describe how one person (say Alice) can sign m and how another person (say Bob) may verify her signature.

First, a random number k is generated by Alice such that $k<q$. Alice's signature is composed by the parameters r and s defined by $r=(g^k \bmod p) \bmod q$ and $s=(k^{-1}(H(m)+xr)) \bmod q$. Bob, in order to verify Alice's signature, computes the following numbers: $w=s^{-1} \bmod q$, $a=(H(m)*w) \bmod q$, $b=(rw) \bmod q$ and $v=(g^a*y^b) \bmod q$. If $v=r$ then Alice's signature is verified by Bob.

Discrete Logarithm based signature schemes are digital schemes relying on the presumed intractability of the Discrete Logarithm Problem.

One of the most popular such schemes, is the DSA-like scheme we describe next. Two large prime numbers p and q (a 160-bit prime factor of $p-1$) and either $p-1$ or a large prime factor of $p-1$ are chosen. Then a number g such that $1 \leq g \leq p$ and $g^p = 1 \bmod p$ is chosen. These numbers are publicly known and can be shared among users of some group. Let also x be the private key with $x<q$ and the public key y given by $y=g^x \bmod q$. To sign a message m, a random number k less than and relatively prime to q must be chosen. Also if q is a prime, any k less than q can be chosen instead. We first compute $r=g^k \bmod p$. Then, the *generalised signature equation* is $ak \equiv b+cx \bmod q$, with a, b and c suitably chosen numbers.

To verify the signature, the recipient checks the *verification equation* $r^a = g^b y^c \bmod p$. Note that by using RSA for digital signatures a nice feature called *message recovery* can be exploited. After verifying an RSA signature, m is computed. Then m is compared to the message and it is examined whether the signature is valid for that message. With the previous DSA-like scheme, m cannot be recovered when the signature is computed and some candidate m must be used in the verification equation. However, a message recovery variant is possible. In order to sign a message m, we first compute $r=mg^k \bmod p$ and then we replace m by 1 in the signature. Then the verification equation can be reconstructed so that m can be computed directly.

Finally, we state the key-exchange protocol of Diffie and Hellman (Diffie and Hellman, 1976). Suppose two communicating parties, Alice and Bob, want to agree on a shared secret key. Then they may execute the following steps. They, first, agree on a finite cyclic group G as well as one of its generators g (it is not necessary to keep g secret). Alice initiates the protocol by generating a secret random positive integer a. Bob also generates a secret random positive integer b. Then Alice's public value is computed as $g^a \bmod p$ while Bob's public value as $g^b \bmod p$. Next Alice and Bob exchange their public values, Alice computes $g^{ab} = (g^b)^a \bmod p$, and Bob computes $g^{ba} = (g^a)^b \bmod p$. It is easy to check that $g^{ab} = g^{ba} = k$, which is taken by Alice and Bob as their secret shared key which they can, subsequently, use to communicate by means of a shared key block cipher.

ELLIPTIC CURVE CRYPTOGRAPHY

Elliptic curve based cryptosystems were introduced independently by Koblitz (1987) and Miller (1985) in 1985 as an alternative to conventional public key cryptosystems such as RSA (Rivest, Shamir & Adleman, 1978) and DSA (Kravitz, 1993). In particular, the cryptographic schemes that are based on elliptic curves are analogues of the corresponding ElGamal schemes (ElGamal, 1985) in which the group F_p is replaced by the group of points on an elliptic curve defined over a finite field. Since 1985, elliptic curves have become a major subject of research in cryptography and in number theory in general. In this part, we will define the parameters of an elliptic curve cryptosystem and present the most well-known encryption and signature schemes.

Elliptic Curve Properties and Generation Methods

There are many important decisions that one should make before starting the implementation of an elliptic curve cryptosystem (ECC). These include the type of the underlying finite field, the algorithms for implementing the basic algebraic operations, the type of the elliptic curve to be used as well as its generation, and finally the elliptic curve protocols. The fields usually used are either prime or binary fields. For simplicity, we shall restrict this section to elliptic curves over prime fields F_p, where p is a prime greater than 3.

An *elliptic curve* (EC) over the prime field F_p, denoted by $E(F_p)$, is the set of points $(x, y) \in F_p^2$ (represented by affine coordinates) which satisfy the equation

$$y^2 = x^3 + ax + b \quad (1)$$

where $4a^3+27b^2 \neq 0$ (this condition guarantees that Equation (1) does not have multiple roots in F_p), along with a special point denoted by O, called the *point at infinity*. An addition operation + is defined over $E(F_p)$ such that $(E(F_p), +)$ defines an Abelian group, called the *EC group*, with O acting as its identity.

A fundamental operation of cryptographic protocols based on EC is the *scalar (or point) multiplication*, i.e., the multiplication of a point P by an integer k (an operation analogous to the exponentiation in multiplicative groups), that produces another point $Q = kP$ (the point resulting by adding P to itself for k times). Several algorithms exist for the fast and efficient implementation of the scalar multiplication operation. Most of them are based on the binary representation of the integer k.

The *order m* of an elliptic curve is the number of points in $E(F_p)$. Hasse's theorem (see e.g., (Blake, Seroussi & Smart, 1999; Schoof, 1995)) gives upper and lower bounds for m that are based on the order p of F_p:

$$p + 1 - 2p^{1/2} \leq m \leq p + 1 + 2p^{1/2} \quad (2)$$

The *order of a point P* is the smallest positive integer n for which $nP = O$. Application of Lagrange's theorem (stating that the exponentiation of any group element to the power of the group's order gives the identity element) on $E(F_p)$, gives that $mP = O$ for any point $P \in E(F_p)$, which in turn implies that the order of a point cannot exceed the order of the elliptic curve.

Two important quantities associated with $E(F_p)$ are the *curve discriminant* Δ and the *j-invariant*, defined by

$$\Delta = -16(4\alpha^3 + 27b^2) \quad (3)$$

and

$$j = -1728(4a)^3/\Delta \quad (4)$$

Given a j-invariant $j_0 \in F_p$ ($j_0 \neq 0, 1728$), two elliptic curves can be easily constructed. The first EC is of the form defined by Equation (1) and can be constructed by setting $a = 3k \bmod p$, $b = 2k \bmod p$, where $k = j_0/(1728\text{-}j_0) \bmod p$. The second EC, called the *twist* of the first, is defined as

$$y^2 = x^3 + ac^2x + bc^3 \quad (5)$$

where c is any quadratic non-residue in F_p. If m_1 is the order of an EC and m_2 is the order of its twist, then $m_1 + m_2 = 2p + 2$, i.e., if one curve has order $p + 1 - t$, then its twist has order $p + 1 + t$, or vice versa (Blake et al., 1999, Lemma VIII.3).

Many of the security properties of elliptic curve cryptosystems depend on the order of the EC group and this is determined by the generated EC. If this order is *suitable*, i.e., it obeys some specific good properties, then there is a guarantee for a high level of security. The order m of an EC is called *suitable*, if the following conditions are satisfied:

1. m must have a sufficiently large prime factor (greater than 2^{160}).
2. $m \neq p$.
3. For all $1 \leq k \leq 20$, it should hold that $p^k \neq 1 \bmod m$.

The above conditions ensure the robustness of cryptosystems based on the *discrete logarithm problem for EC groups* (ECDLP), since it is very difficult for all known attacks to solve this problem efficiently, if m obeys the above properties. ECDLP asks for determining the value of t when two points P, Q in $E(F_p)$ are given such that P is of order n and $Q = tP$, where $0 \leq t \leq n - 1$. The best algorithm for attacking this problem takes time exponential in the size of the EC group, while for (conventional, non-EC) groups generated by prime numbers (e.g., the discrete logarithm problem – DLP) there are algorithms that take

subexponential time. This implies that one may use smaller parameters for the EC cryptosystems than the parameters used in RSA or DSA, obtaining the same level of security. A typical example is that a 160-bit key of an EC cryptosystem is equivalent to RSA and DSA with a 1024-bit modulus. As a consequence, smaller keys result in faster implementations, less storage space, as well as reduced processing and bandwidth requirements. This advantage is really crucial when we are interested in implementations on constrained devices, such as smart cards, pagers or sensors.

There are three methods for generating the parameters of an EC: the point counting method (see Schoof, 1995), the method based on the constructive Weil descent (Galbraith, 2001), and the Complex Multiplication (CM) method (see Blake et al., 1999). The point counting method does not necessarily construct an EC of suitable order, but it may achieve this by repeated applications of the method. The other two methods construct ECs of a suitable order. In (Galbraith, 2001) it was shown that the method based on the constructive Weil descent suffers from a major drawback that is not easy to handle: it samples from a very small subset of the set of possible elliptic curves. For this reason, in almost all applications the other two methods are used. The point counting method was presented in 1985 by Schoof (1995) and it was the first polynomial time algorithm for computing the order of an ordinary EC. The algorithm computes the values m mod l for small prime numbers l and then determines the order m using the Chinese Remainder Theorem. In practice, the method is inefficient for values of p that are used in practical cryptosystems, but since 1985 it has been improved by many researchers – more details on the improvements of Schoof's method can be found in (Blake et al., 1999).

The theory of complex multiplication (CM) of elliptic curves over the rationals can be used to generate elliptic curves of a suitable order m, resulting in the so-called *CM method*. The CM method computes j-invariants from which it is then easy to construct the EC. The method is based on the following idea (for more details see for example [Blake et al., 1999]). Hasse's theorem implies that $Z = 4p - (p + 1 - m)^2$ is positive. This in turn implies that there is a unique factorization $Z = Dv^2$, where D is a square free positive integer. Consequently,

$$4p = u^2 + Dv^2 \tag{6}$$

for some integer u satisfying

$$m = p + 1 \pm u \tag{7}$$

D is called a *CM discriminant for the prime* p and the elliptic curve has a *CM by D*. The CM method uses D in order to determine the j-invariant and constructs an EC of order $p + 1 - u$ or $p + 1 + u$.

The method starts with a prime p and then chooses the smallest D along with an integer u to satisfy Equation (6). Then, it checks whether $p + 1 - u$ and/or $p + 1 + u$ is suitable. If neither is suitable, the process is repeated. Otherwise, the so-called *Hilbert polynomials* have to be constructed (based on D) and their roots have to be found. A root of the Hilbert polynomial is the j-invariant we are seeking. The EC and its twist are then constructed as explained previously. Since only one of the ECs has the required suitable order, the particular one can be found using Lagrange's theorem by picking random points P in each EC until a point is found in some curve for which $mP \neq O$. Then, the other curve is the one we are seeking.

Concluding, the CM method is much faster than the best algorithms known for counting the points of randomly selected ECs over prime fields.

For ECs over binary fields, variations of Satoh's algorithm (Satoh, 2000) for point counting can find cryptographically secure ECs in few seconds.

Elliptic Curve Based Protocols

The security of cryptographic protocols based on ECs, is guaranteed by the intractability of ECDLP (Elliptic Curve Discrete Logarithm Problem). In particular, in EC-based protocols, each user selects his pair of public and private keys as follows: suppose that the base point P has prime order n. Then each user selects an integer k uniformly at random in the interval $[1, n-1]$ and computes the point $Q = kP$. His private key is k and his public key is the point Q. It is clear that the security of the private key is based on the difficulty of solving ECDLP.

Because of the straightforward correspondence of ECDLP and DLP, all cryptographic protocols that are based on DLP have their analogues in elliptic curve cryptosystems. For example, the Diffie-Hellman key agreement protocol (Diffie & Hellman, 1976), the ElGamal cryptosystem (ElGamal, 1985) and Digital Signature Algorithm (DSA) can be easily transformed to their elliptic curve analogues. In the Elliptic Curve Diffie-Hellman (ECDH) key agreement protocol, the two users A and B agree on the same elliptic curve parameters and compute the points $Q_A = k_A P$ and $Q_B = k_B P$ correspondingly. The point P is the base point, while the values k_A, k_B are randomly chosen integers in $[1, n-1]$. Then, the final shared key is calculated by the user A as $K = k_A Q_B = k_A k_B P$ and by user B as $K = k_B Q_A = k_A k_B P$.

A simple analogue of ElGamal encryption can similarly be created:

1. The plaintext m is first represented as a point M in $E(F_p)$. Then, an integer k is randomly selected such that $0 < k < n-1$, where n is the largest prime factor of the EC order m.
2. The point $C_1 = kP$ is computed, where P is the base point.
3. A second point $C_2 = M + k\,Q_B$ is computed. Q_B is the public key of the intended recipient.
4. The ciphertext is the pair (C_1, C_2).

The decryption procedure is the following:

1. Using his private key k_B the recipient of the ciphertext computes the point $k_B\,C_1$.
2. The point M is retrieved by $C_2 - k_B\,C_1$. Finally, the plaintext m is extracted from M.

A variant of the ElGamal encryption scheme is also the Elliptic Curve Integrated Encryption Scheme (ECIES) which was proposed by Bellare and Rogaway in 1997 (Bellare & Rogaway, 1997). The algorithm has been standardized in ANSI X9.63 and ISO/IEC 15946-3 standards, and is the most widely used elliptic curve based encryption scheme.

The Elliptic Curve Digital Signature Algorithm (ECDSA) (Johnson & Menezes, 1999) is the elliptic curve analogue of the Digital Signature Algorithm (DSA). It is the most well-known elliptic curve based signature scheme and it has been standardized in ANSI X9.62, FIPS 186-2 and several other standards. In the following, H(.) denotes a cryptographic hash function whose outputs have bit length smaller than the bit length of n (the largest prime factor of the EC order m). The steps of the signature generation are:

1. Choose a random integer k with $1 \leq k \leq n-1$.
2. Compute the point $kP = (x_1, y_1)$ where P is the base point. Next, compute $r = x_1 \bmod n$. If $r = 0$ return to the first step.
3. Compute the value $k^{-1} \bmod n$.
4. Use a hash function H to compute $e = H(M)$, where M is the message to be signed.

5. Compute $s = k^{-1}(e + dr) \bmod n$, where d is the signer's private key. If $s = 0$ then return to the first step.
6. The pair (r, s) is the signature.

The steps of the signature verification are the following:

1. Compute $e = H(M)$, where M is the signed message.
2. Compute $w = s^{-1} \bmod n$.
3. Compute the values $u_1 = ew \bmod n$ and $u_2 = rw \bmod n$.
4. Calculate the point $X = u_1P + u_2Q$, where Q is the signer's public key. If X is equal to the point at infinity, then reject the signature. Otherwise, compute $u = x_1 \bmod n$ where x_1 is the x-coordinate of the point X.
5. The signature is accepted only if $u = r$.

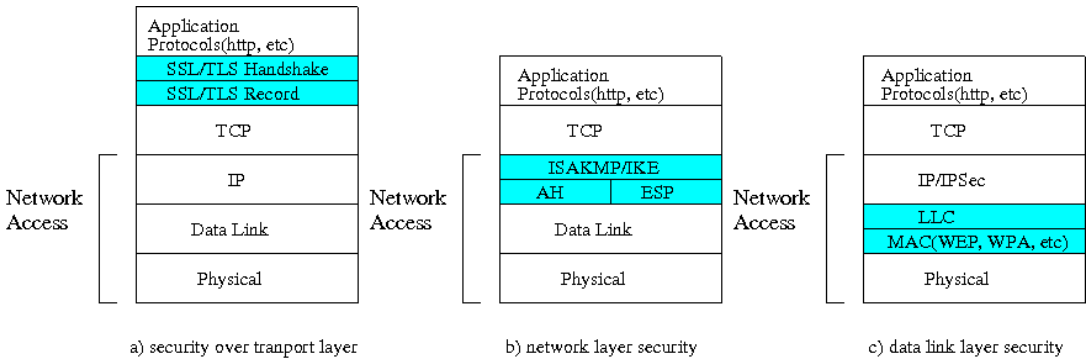
ECC BASED EXTENSIONS OF SECURITY PROTOCOLS AND THEIR APPLICATIONS ON EMBEDDED SYSTEMS

In this section, we present the main characteristics of some of the most widely employed protocols within the embedded systems domain and we show how such protocols can benefit from ECC. Furthermore, we show how a specially tailored port of a general purpose library, ECC-LIB (Konstantinou, Stamatiou & Zaroliagis, 2002) can be used in an ECC based extension of the Wireless Protected Access Protocol (Papaioannou, Nastou, Stamatiou, & Zaroliagis, 2009).

Generally, the client-server model is the standard model for most network applications. The majority of the embedded devices are equipped to support electronic applications for online banking, trading and voting. Peer applications on network devices can communicate over an insecure network through standard communication channels named sockets. The security of the e-commerce applications are based mainly on the Secure Sockets Layer protocol (SSL) originally developed by Netscape Communications Corporation and which evolved as a Web security standard since most Internet Browser developers adopted it for Web security. Later, the Transport Layer Security (TLS) working group was formed in the Internet Engineering Task Force (IETF) to develop a common standard and the TLS Internet standard for Web security was ratified (Dierks & Rescola, 2008).

Both protocols implement security above the Transport Control Protocol (TCP), as it is shown in Figure 1a. There is a number of specific applications that embeds the SSL protocol in order to provide application specific security services. SSL/TLS is considered as an inexpensive way to establish a secure connection between pairs of hosts when the demand for communication is occasional and it requires little user participation.

Figure 1. Levels of network security



Instead of adding security facilities to Web applications, there is the alternative of providing end-to-end security at the network layer. The initial version of the IP protocol (IPv4) did not support security mechanisms (Stallings, 1999). Thus, IETF formed a working group named IPSec that developed a protocol suite for providing security services in the IP protocol. In some sense, the IPSec group provided extensions to the IP protocol. In the IPv6, the IPSec security mechanisms have been adopted as mandatory. Actually, IPSec provides network layer security (Figure 1b) where any application can exploit it while transferring data through the network. A mobile device can be connected to a network device of a private or public network through an insecure network by using IPSec (Shneyderman & Casati, 2003).

However, for wireless mobile devices where the medium is considered absolutely insecure, data link layer security is important in addition to the network layer security (Raza, Voigt, & Jutvik, 2012). IPSec that provides network security is an end-to-end protocol which means that any attack can be detected at the end nodes. A data link layer security protocol (Figure 1c) controls the access to the insecure wireless medium. It is obvious that providing security at the link level any attack can be detected earlier. An example of this situation is the IEEE 802.11 standard for Wireless LANs (WLANs) where the members of the 802.11i Task Group have paid particular attention to provide WLAN users with a powerful security protocol at MAC layer.

Next, the architecture of the SSL/TLS, IPSec and WPA security protocols along with the involvement of the Elliptic Curve Cryptography in these protocols are presented.

The SSL/TLS Protocol

The idea is to establish a peer-to-peer connection over TCP/IP between the client and the server of an application. So, if an embedded device is to run more than one application concurrently, it must establish one SSL/TLS connection per application. SSL/TLS architecture is composed of the Handshake Protocol and the Record Protocol (Figure 1a). The Handshake protocol is used by the server and the client in order to establish a secure connection by authenticating each other, negotiating cipher suites and agreeing on cryptographic keys. The established secure connection is used by the Record Protocol to provide secure communication to the application required the secure connection.

A list of supported public-key and symmetric cryptographic algorithms, signature schemes, MAC schemes and hash functions forms the cipher suite. The generation of the keys that will be used by the symmetric cryptographic algorithms and the keyed hashed functions is based on a master key agreed by the server and the client through the Handshake Protocol. Thus, the Handshake protocol provides to the Record protocol the negotiated cipher suite and the agreed master key. The specification in (Dierks & Rescorla, 2008), recommends some cipher suites, e.g. TLS_DH_RSA_WITH_AES_128_CBC_SHA indicates the use of RSA Diffie Hellman key exchange mechanism, the AES with a key of 128-bit in CBC mode of operation for the encryption and decryption in the record protocol and SHA for hashing. Blake-Wilson et al. (2006) presented new ECC based key exchange algorithms for the TLS protocol. As for example, the EC cipher suite TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA indicates the use of Elliptic Curve Diffie-Hellman (ECDH) key agreement mechanism, the use of Elliptic Curve Digital Signature Algorithm (ECDSA) as a new authentication mechanism, the AES with a key of 128-bit in CBC mode of operation for the encryption and decryption in the record protocol and SHA for hashing.

The SSL/TSL handshake scheme is shown in Figure 2. As it is also pointed out in Figure 2, the server and the client initially negotiate the cipher suite that will be used for master key agreement and by the record protocol and they exchange two random numbers N_c and N_s through SSL/TLS Hello messages. The server sends to the client its

certificate which contains its public key signed by the issuing certification authority. If the negotiated cipher suites are ECDSA_fixed_ECDH for the client and ECDH_ECDSA_WITH_AES_128_CBC_SHA for the server, the server sends to the client its certificate that contains its ECDH-capable public key signed with ECDSA.

The server requests the client's certificate by sending a CertificateRequest in order to authenticate the client. The client responds by sending its certificate which contains its ECDH-capable public key on the same elliptic curve as the server's ECDH key (this scheme is computationally more efficient). The server and the client authenticate each other by verifying the signature of the received ECDH public keys. At the end of authentication, the client sends an empty KeyExchange message (this happens only when the TLS_ECDSA_fixed_ECDH cipher suite for the client has been negotiated) and both the server and the client generates the master key (MK) as it is presented in Figure 2. Finally, the client and the server verify that they possess the same master key by exchanging the hash value of their MK through the *Finished* message of SSL/TLS protocol.

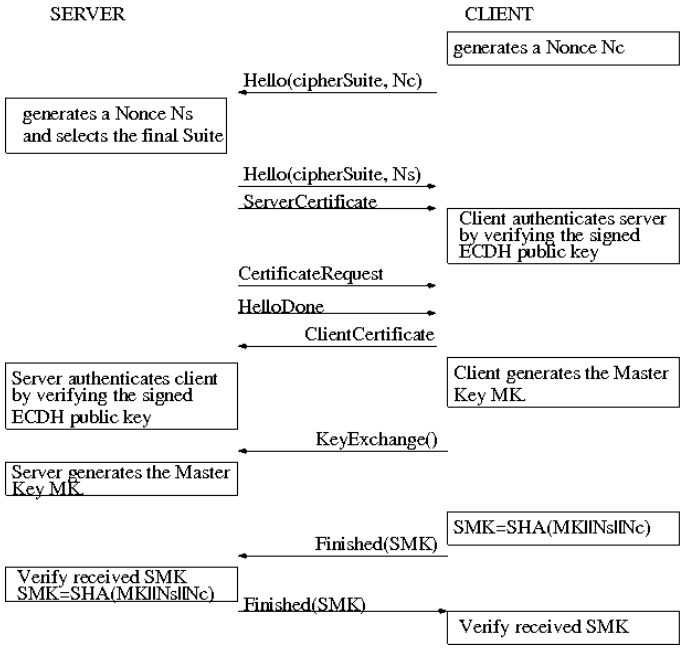
Gupta et al. (2005) presented Sizzle (Slim SSL), the world's smallest secure Web server on Mica2 motes implemented by Sun microsystems. Sizzle does not request client's certificate by CertificateRequest which eliminates the transmission of the certificate. The processor of Mica2 motes was the 8-bit Atmel ATmega 128L with 128KB flash, 4KB EEPROM, 4KB of RAM.

The IPSec Protocol Suite

The IPSec protocol suite provides security services through the Authentication Header protocol (AH), the Encapsulating Security Payload Protocol (ESP), the Internet Security Association and Key Management Protocol (ISAKMP) and the Internet Key Exchange protocol (IKE) (Figure 1b).

The AH protocol provides data origin authentication and data integrity through the AH header which is added either between the IP header and

Figure 2. SSL/TLS handshake scheme



the transport layer payload in the transport mode or between the new IP header and the initial IP header of the IP packet in the tunnel mode. Similarly, the ESP protocol provides authentication and confidentiality through the ESP header added either between the IP header and the transport layer payload in the transport mode or between the new IP header and the initial IP header of the IP packet in the tunnel mode (Yuan & Strayer, 2001). The transport mode protects an upper-layer protocol while in the tunnel mode the original IP packet, which either encrypted or subjected to other security measures, is encapsulated into another IP packet by forming a new IP header that steers the packet in the network. Moreover, the ESP protocol appends an ESP trailer and authentication data in both tunnel and transport mode. The ESP protocol is used, either in tunnel or in transport mode, in the majority of Virtual Private Network (VPN) approaches whereas in Mobile IP the AH protocol in transport mode is used (Shnyderman & Casati, 2003).

Each IPSec device should support the Security Policy Database (SPD) and the Security Association Database (SAD). The SPD contains an ordered list of certain rules that should be applied to IP packets e.g. which encryption algorithm should be used for encryption. These rules are determined by the IPSec device administrator. A source device in order to send data to a destination device negotiates with the destination the security parameters that should be used to protect the IP traffic based on the rules of the SPD of the destination. The agreed security services form a Security Association (SA) which is recorded in the SAD of the source and the destination device. An IPSec SA is uniquely determined by the IP address of the destination device, the IPSec protocol identifier (AH or ESP) and the Security Parameter Index (SPI). If a bidirectional communication is needed the two devices should form a second IPSec SA for the other direction. A device establishes an IPSec SA with its peer dynamically by ISAKMP (Harkins & Carrel, 1998). ISAKMP determines the procedures for securing the communication channel between two communicating devices.

Version 2 of IKE – IKEv2 (Harkins & Carrel, 1998; Kaufman, 2005) – is a protocol that uses the ISAKMP framework for mutual authentication and key exchange. Initially, IKE establishes an IKE Security Association (IKE_SA) that includes the shared secret information for establishing IPSec SAs between communicating devices and the cryptographic algorithms that should be used to protect traffic while establishing an IPSec SA. Actually, an IKE establishes a secure channel between two communicating devices through which the devices establish its IPSec secure channel for data transfer. An IKE_SA is established after the exchange of two pairs of messages, the IKE_SA_INIT and IKE_AUTH, between communicating devices. The IKE_SA_INIT pair of messages negotiate the cryptographic algorithms, exchange nonces and the public information in DH key exchange, (i.e., the large prime p that defines the modular exponentiation group, a generator g of this group and two powers of g or the EC parameters a, b and m, a point B on EC and the public key $P_e = K_d B$) and establish the key of IKE_SA which is the master key, since it is used as a seed for the construction of keys used for encryption and integrity protection of the IKE_AUTH pair of messages. By IKE_AUTH messages the recipients verify that they hold the right master key and negotiate the cryptographic algorithms that should be used by IPSec SA established in the next phase of IKE.

Upon the completion of IKE_SA, the two communicating parties in order to establish the IPSec SA exchange the CREATE_CHILD_SA pair of messages. One of the two communicating devices sends to its peer the details of SA, a nonce, a DH value for the IPSec SA key generation and the traffic selectors. The peer responds by sending a message that contains its corresponding data to the initiator. Both messages are encrypted and authenticated using the cryptalgorithms negotiated in the IKE_SA establishment and the keys generated by the IKE_SA key. The result of this

exchange is the IPSec SA and its master key which is considered as the user master key since it is used by the IPSec SA for the protection of the data traffic.

Fu and Solinas (2010) described for use in IKEv2 new elliptic curve groups which provide efficiency advantages in hardware applications. Although, the RFCs recommend the use of EC groups, the current IKE implementations use the RSA in DH for key exchange. Raza, Voigt and Jutvik (2012) suggested an implementation of ECC in DH based on standardized ECC algorithms and NIST recommended EC and prime numbers using the uIP stack and Contiki OS, which runs on networked embedded systems and wireless sensor networks.

The Wireless Protected Access Protocol

In 802.11i specification, the operation of a Robust Security Network (RSN) is defined in an Extended Service Set (ESS) or an Independent Basic Service Set (IBSS), which is the ad-hoc case. The operation of an RSN is based on the establishment of RSN Associations between Stations which can be based on Pre-Shared Key (PSK) or on IEEE 802.1X AKM (Authentication and Key Management). In an ESS, the Access Point (AP) is the Authenticator, and associated devices are the Supplicants.

Upon completion of an RSN association between Authenticator and Supplicant, the Authenticator initiates the 4-way Handshake protocol (802.11i). This is a Key Management protocol through which the existence of the PMK is confirmed, as well as that it is current, then a unique Pairwise Transient Key (PTK) from the PMK is derived, and the unicast encryption and message integrity keys are generated. Those keys are used in the 802.11 MAC Layer symmetric block ciphers (mainly AES) in various modes of operation. Initially, the Authenticator sends a nonce-value to the Supplicant (ANonce), and the Supplicant constructs the PTK. The Supplicant responds by sending its own nonce-value to the Authenticator along with a Message Integrity Code (MIC) and the Authenticator generates the PTK and if it is needed the Group Temporal Key (GTK). Finally, the Authenticator sends to Supplicant the GTK and a sequence number along with MIC and the Supplicant sends a confirmation to Authenticator and establishes the PTK and GTK (if it was transmitted by Authenticator).

The software library ECC-LIB, presented in (Konstantinou, Stamatiou & Zaroliagis, 2002), was applied in (Papaioannou, Nastou, Stamatiou & Zaroliagis, 2009) for the implementation of an EC Diffie-Hellman Key Exchange Protocol that imitates the operation of the 802.11 4-way handshake protocol. This library is suitable for embedded systems since it contains implementations of new algorithms for the Complex Multiplication (CM) method, which are based on the use of the lightweight Weber polynomials instead of the computationally heavy Hilbert polynomials, thus resulting in faster and space-efficient methods compared to implementations with Hilbert polynomials for generating secure ECs over a prime field F_p, i.e., ECs of a suitable order. The details of these new algorithms and their experimental evaluation are reported in (Konstantinou, Stamatiou & Zaroliagis, 2007) and (Konstantinou, Kontogiorgis, Stamatiou & Zaroliagis, 2010). The ECC-LIB code has been successfully ported to Windows CE (one of the most commonly used operating systems for PDAs) and details can be found in (Argyroudis, 2004). For the implementation of an EC Diffie-Hellman Key Exchange Protocol that imitates the operation of the 802.11 4-way handshake protocol, only the code related to CM method variant has been ported to μcLinux which is a widely used operating system for embedded network devices (for example WLAN Access Points). Similar ports of ECC-LIB can be produced for other limited resource devices as well as embedded systems by simply compiling it with the right compilation switches.

The idea presented in (Papaioannou et. al, 2009) was simply to construct a protocol based on Elliptic Curve cryptography that will create PTK in a more secure way. A simple client-server application was written in standard ANSI C in (Papaioannou et al., 2009). The server part of the application runs on the AT76C520 device, an ATMEL's 802.11 WLAN Access Point, while the client part runs on a notebook. Since the EC generation and the construction of a public and private key pair using the ECC-LIB requires much time, a scenario was considered where the authenticator creates an EC with certain characteristics and a pair of private K_a and public P_a keys during its initialization instead of generating them in real time. The PMK now is the discriminant D and the size of prime p that the authenticator needs for the EC generation using the CM method.

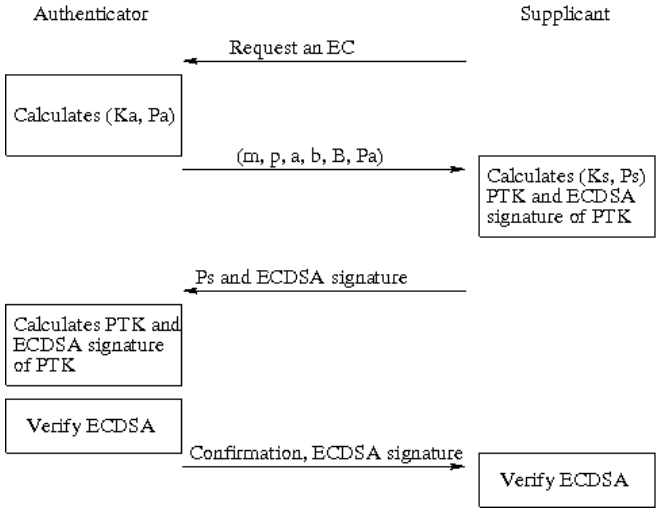
After a successful association between the authenticator and the supplicant, the latter requests from an 802.11 authenticator an elliptic curve as it is shown in Figure 3.

The authenticator sends to the supplicant the EC parameters *a, b* and *m*, the prime number *p*, the coordinates of the base point *B* and its public key $P_a = K_a B$. The supplicant then creates its own set of private and public keys (K_s, P_s) where $P_s = K_s B$ based on the received EC and transmits P_s back to the authenticator to complete the Diffie-Hellman protocol. The shared key of the two devices is then calculated on these devices: $PTK_a = K_a P_s$ and $PTK_s = K_s P_a$. In order to assure both the authenticator and the supplicant that $PTK_s = PTK_a$, the supplicant sends to the authenticator along with the public key, an ECDSA signature using PTK_s. The authenticator verifies the received public key, installs the constructed PTK and sends to the supplicant its public key ECDSA signature using PTK_a. Finally, the supplicant verifies that the received signature is the correct one and installs the constructed PTK. The above protocol was tested (Papaioannou et. al, 2009) using an Ethernet connection between a notebook (supplicant) and the AT76C520 embedded device.

CONCLUSION AND FUTURE RESEARCH

In this chapter, we presented an overview of cryptographic primitives and protocols that can be implemented on embedded systems. We focused on the promising cryptographic constructs based on Elliptic Curves that can lead to performance

Figure 3. An EC Diffie-Hellman key management protocol in 802.11



improvements when properly implemented on hardware, since their security can be assured with key sizes smaller than the ones required by conventional cryptographic schemes such as ElGamal or RSA. Our aim was to demonstrate the potential of ECC cryptography as well as the range of applications that can benefit from it and provide pointers that lead to more in depth information. To this end, we described the cryptographic primitives of ECC and how they are used in building security protocols. Then we showed how standard security protocols can be enhanced to use ECC and what improvements are obtained. Finally, we discussed open source software libraries for ECC that can be tailored to the needs of practical security problems in embedded systems using as an example the port of ECC-LIB to certain operating systems. As future work, we plan to explore the use of ECC in other types of embedded platforms and devices such as TPMs (Trusted Platform Modules) and eIdentity smart cards. Our goal is to contribute to the development and performance evaluation of a single, comprehensive library that can be tailored to the needs of a wide range of devices based on the settings of compilation switches.

REFERENCES

Argyroudis, P. (2004). *NTRG ECC-LIB WINCE – A WinCE port of ECC-LIB*. Retrieved from http://ntrg.cs.tcd.ie/~argp/software/ntrg-ecc-lib-wince.html

Bellare, M., & Rogaway, P. (1997). Minimizing the use of random oracles in authenticated encryption schemes. In *Proceedings of Information and Communications Security '97 (LNCS)* (*Vol. 1334*, pp. 1–16). Berlin: Springer. doi:10.1007/BFb0028457.

Blake, I., Seroussi, G., & Smart, N. (1999). *Elliptic curves in cryptography*. London: Cambridge University Press.

Blake-Wilson, S., Bolyard, N., Gupta, V., Hawk, C., & Moeller, B. (2006). *Elliptic curve cryptography (ECC) cipher suites for transport layer security (TLS). RFC 4492*. IETF.

Daemen, J., & Rijmen, V. (2002). *The design of Rijndael: AES - The advanced encryption standard*. Berlin: Springer. doi:10.1007/978-3-662-04722-4.

DES. (1977). *Data encryption standard*. Washington, DC: Federal Information Processing Standards Publication.

Dierks, T., & Rescorla, E. (2008). *Transport layer protocol version 1.2 (TLS). RFC 5246*. IETF.

Diffie, W., & Hellman, M. E. (1976). New directions in cryptography. *IEEE Transactions on Information Theory*, *22*, 644–654. doi:10.1109/TIT.1976.1055638.

ElGamal, T. (1985). A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, *31*, 469–472. doi:10.1109/TIT.1985.1057074.

Fu, D., & Solinas, J. (2010). *Elliptic curve groups modulo a prime for IKE and IKEv2. RFC 5903*. IETF.

Galbraith, S. (2001). Limitations of constructive Weil descent. In Public-Key Cryptography and Computational Number Theory.

Gupta, V., Wurm, M., Zhu, Y., Millard, M., Fung, S., & Gura, N. et al. (2005). Sizzle: A standards-based end-to-end security architecture for the embedded internet. *Pervasive and Mobile Computing*, 425–445. doi:10.1016/j.pmcj.2005.08.005.

Harkins, D., & Carrel, D. (1998). *The internet key exchange (IKE). RFC2409*. IETF.

IEEE. (n.d.). 802.11i, IEEE medium access control (MAC) security enhancements. IEEE Task Group I P802.11i.

Johnson, D., & Menezes, A. (1999). *The elliptic curve digital signature algorithm (ECDSA) (Technical report CORR 99-06)*. Waterloo, Canada: University of Waterloo.

Kaufman, C. (2005). *Internet key exchange (IKEv2) protocol. RFC 4306*. IETF.

Koblitz, N. (1987). Elliptic curve cryptosystems. *Mathematics of Computation*, *48*, 203–209. doi:10.1090/S0025-5718-1987-0866109-5.

Konstantinou, E., Kontogiorgis, A., Stamatiou, Y. C., & Zaroliagis, C. (2010). On the efficient generation of prime order elliptic curves. *Journal of Cryptology*, *23*(3), 477–503. doi:10.1007/s00145-009-9037-2.

Konstantinou, E., Stamatiou, Y. C., & Zaroliagis, C. (2002). *On the efficient generation of elliptic curves over prime fields*. Berlin: Springer-Verlag.

Konstantinou, E., Stamatiou, Y. C., & Zaroliagis, C. (2007). Efficient generation of secure elliptic curvers. *International Journal of Information Security*, *6*(1), 47–63. doi:10.1007/s10207-006-0009-3.

Kravitz, D. W. (1993). *Digital signature algorithm*. U.S. Patent #5,231,668. Washington, DC: US Patent Office.

Miller, V. (1985). Uses of elliptic curves in cryptography. In *Proceedings of Advances in Cryptology – Crypto '85 (LNCS)* (*Vol. 218*, pp. 417–426). Berlin: Springer. doi:10.1007/3-540-39799-X_31.

Papaioannou, P., Nastou, P., Stamatiou, Y., & Zaroliagis, C. (2009). Secure elliptic curve generation and key establishment on a 802.11 WLAN embedded device. In *Proceedings of the 9th IEEE International Symposium on Autonomous Decentralized Systems*, (pp. 41-48). IEEE.

Raza, S., Voigt, T., & Jutvik, V. (2012). Lightweight IKEv2: A key management solution for both the compressed IPSec and the IEEE 802.15.4 security. In *Proceedings of the IETF International Workshop on Smart Object Security*. IETF.

Rivest, R., Shamir, A., & Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, *21*, 120–126. doi:10.1145/359340.359342.

Satoh, T. (2000). The canonical lift of an ordinary elliptic curve over a prime field and its point counting. *Journal of the Ramanujan Mathematical Society*, *15*, 247–270.

Schoof, R. (1995). Counting points on elliptic curves over finite fields. *Journal Theorie des Nombres de Bordeaux*, *7*, 219–254. doi:10.5802/jtnb.142.

Shneyderman, A., & Casati, A. (2003). *Mobile VPN: Delivering advanced services in next generation wireless systems*. Indianapolis, IN: Wiley.

Silverman, J. H. (1986). *The arithmetic of elliptic curves*. Berlin: Springer. doi:10.1007/978-1-4757-1920-8.

Stallings, W. (1999). *Cryptography and network security: Principles and practice*. Upper Saddle River, NJ: Prentice Hall.

Yuan, R., & Strayer, T. (2001). *Virtual private networks: Technologies and solutions*. Boston, MA: Addison Wesley.

KEY TERMS AND DEFINITIONS

Cipher: A cryptographic algorithm for the encryption and decryption of messages.

Ciphertext: An encrypted plaintext.

Decryption: A process that transforms back the ciphertext into its original (plaintext) form.

Embedded System: A networked computing system characterized by limited computing and storage resources.

Encryption: A process that transforms a message into a form so that it cannot be understood by parties who do not possess a special key.

Plaintext: A message to be subjected to encryption.

Protocol: A set of rules that two or more communicating parties should follow in order to establish communication channels and to exchange data messages.

Security Protocol: A protocol for the establishment of secure channels and the secure exchange of messages between communicating parties.