Evaluation of the Security of ECDSA

Johannes Buchmann

January 11, 2001

Contents

Chapter 1

Introduction

The goal of this document is to evaluate the security of the digital signature scheme ECDSA.

The security of ECDSA depends on the intractability of the discrete logarithm problem in point groups of elliptic curves over finite fields (ECDL) and on the security of the used hash function and pseudorandom number generator. In order to evaluate the security of ECDSA, it is, therefore, necessary to evaluate the difficulty of ECDL and the security of the used hash function and pseudorandom number generator. But this is not sufficient. Even if the underlying number theoretic problem is hard and the hash function and random number generator is secure, ECDSA may still be insecure. A famous example for such a situation is the discovery of the possibility of an attack against the RSA encryption standard PKCS #1 (see [2]). In this standard, an insecure padding scheme was used, which compromised the security of the whole scheme.

Because of this situation, it is desirable to find a security proof for ECDSA. Unfortunately, no provably hard computational problem in number theory is known, which could serve as the basis of a secure signature scheme. Also, no provably secure cryptographic hash function and pseudorandom number generator are known. Therefore, given current knowledge, there are no provably secure digital signature schemes. But it is possible to say more about the security of a digital signature scheme than just arguing that the underlying computational problems are intractable. Modern security proofs for digital signature schemes reduce their security to the difficulty of basic computational problems in mathematics. This means that the difficulty of those basic problems is not only necessary but also sufficient for the security of the digital signature scheme which relies on their security.

The questions that I answer in this report are the following. Which are the basic computational problems, on which the security of ECDSA is based, and to what extent can this security be reduced to the intractability of those problems? How difficult are those basic mathematical problems and how difficult are the instances which arise from the specific applications in ECDSA?

This report is organized as follows. Chapter 2 gives an overview over ECDSA. Chapter 3 describes the models and techniques that are used in the security proofs of digital signature schemes and explains to what extent the security of ECDSA can be proved secure in those models. The security of ECDSA relies on the intractability of computing discrete logarithms in elliptic curve point groups, finding collisions of hash functions and guessing the output of pseudorandom number generators. Chapter 4 describes the current knowledge concerning the intractability of those basic problems. Chapter 5 describes the specific instances of the basic computational problems on which the security of ECDSA rely and their difficulty. I conclude this report by summarizing the security of ECDSA in Chapter 6.

Chapter 2

Description of ECDSA

2.1 The setup

I describe the structure of a digital signature scheme. A digital signature scheme has three parts.

Key generation The signer generates a private key and the corresponding public key. He keeps the private key secret and publishes the public key. The authenticity of the public key is certified by a certification authority (CA). The certification authority guarantees with its signature that the verifiers obtain the valid public keys of the signers. It is also possible that the certification authority generates the key pair and gives the secret key to the user.

Signature generation In this step the signer produces the digital signature of a document d. The document d is a bit string of arbitrary length. To generate its digital signature, the signer uses his private key.

Signature verification The verifier uses the public key of the signer to verify the digital signature. The signature is convincing if only the signer, knowing his private key, is able to produce that valid signature. The existence of that valid signature then implies that the signer must have produced it, thereby agreeing to the content of the document.

2.2 ECDSA

I give a summary of ECDSA. Technical details can be found in the submission.

ECDSA is similar to the ElGamal signature scheme (see [3]). The ECDSA description can be found in [15]. There are two variants of ECDSA. The first one works over finite prime fields. The second one works over finite fields of characteristic 2.

For ECDSA over finite prime fields an n-bit prime p is chosen such that

$$n \in \{112, 128, 160, 192, 224, 256, 384, 521\}.$$

Then elements $a, b \in \mathbb{F}_p$ with $4a^3 + 27b^3 \neq 0$ and a point G on the elliptic curve

$$E(p, a, b) = \{(x, y) \in \mathbb{F}_p^2 : y^2 = x^3 + ax + b\} \cup \mathcal{O}$$

of prime order l are chosen which satisfy the following conditions.

1. $\sharp E(p, a, b) \neq p$,
2. $p^B \not\equiv 1 \bmod l$ for $1 \leq B < 20$, and
3. $\sharp E(p, a, b)/l \leq 4$.

For ECDSA over finite fields of characteristic 2 an integer

$$m \in \{113, 131, 163, 193, 233, 239, 283, 409, 571\}$$

is chosen. Then m is a prime number. Then elements $a, b \in \mathbb{F}_{2^m}$ with $b \neq 0$ and a point G on the elliptic curve

$$E(2^m, a, b) = \{(x, y) \in \mathbb{F}_{2^m}^2 : y^2 + xy = x^3 + ax^2 + b\} \cup \mathcal{O}$$

of prime order l are chosen which satisfy the following conditions:

1. $\sharp E(2^m, a, b) \neq 2^m$,
2. $2^{mB} \not\equiv 1 \bmod l$ for $1 \leq B < 20$, and
3. $\sharp E(2^m, a, b)/l \leq 4$.

In ECDSA, it is possible to chose random curves with those properties. However, for compatibility reasons with existing standards such as P1363 [9], the choice of special curves from [16] is recommended.

In ECDSA the SHA-1 hash function h is used (see [19]) which maps strings of at most 2^{61} octets to 160-bit strings.

2.2.1 Key generation

A finite field, an elliptic curve over that field and a point G on that curve of prime order l are selected as described above. Those data are the *domain parameters*. The key generation consists of the following steps.

1. A random $v \in \{1, \ldots, l-1\}$ is chosen.
2. The point $Y = vG$ is computed.

The private key is v. The public key is Y.

2.2.2 Signature generation

The document d is signed. It is a bit string of at most 2^{61} octets. The secret ECDSA key from the previous section is used. The following operations are performed.

1. Select a random $k \in \{1, \ldots, l-1\}$.
2. Compute the point $(x, y) = kG$ on E.
3. Set $r = x \bmod l$
4. Compute the hash value $h = h(d)$.
5. Compute $s = k^{-1}(h + rv) \bmod l$
6. The signature is (r, s).

2.2.3 Signature verification

The verification of the signature of the previous section works as follows.

1. Verify that r, s are both in $\{1, \ldots, l-1\}$.
2. Compute the hash value $h = h(d)$.
3. Compute $(x, y) = hs^{-1}G + rs^{-1}Y$.
4. If $x \equiv r \bmod l$ accept. Otherwise, reject.

Chapter 3

Security proofs

In this chapter I explain what a security proof for a digital signature system is and which security proofs for ECDSA are given in the self evaluation of ECDSA.

3.1 Security of signature schemes

3.1.1 Security proofs are reductions

The security of all known digital signature schemes depends on the intractability of certain computational problems in mathematics, specifically in number theory. Examples are the integer factoring problem and the discrete logarithm problem in an appropriate group. However, no provably hard computational problems are known which can serve as the security basis of a digital signature scheme. Therefore, no rigorous security proofs for signature schemes are known and there is little hope that such proofs will be found in the future.

Todays security proofs are reductions. The goal of such a reduction is to show that the ability of an attacker to mount a successful attack on a signature scheme implies his ability of solving a basic computational problem in mathematics. This is supposed to increase the trust in the security of a digital signature system. The idea of this approach is the following.

When analyzing the security of a signature scheme it is hard to predict which attacks are possible since the system may be very complex and may depend on numerous parameters. Even, if the underlying basic computational problems are intractable, some part of the signature scheme might be implemented in such a way that an attack is possible. A famous example for such a situation is the discovery of the possibility of an attack against the RSA encryption standard PKCS # 1 (see [2]). In this standard, an insecure padding scheme was used, which compromised the security of the whole scheme, even though the RSA encryption scheme is based on the intractable integer factoring problem. However, if the security of the digital signature scheme can be reduced to the difficulty of a well defined computational problem in mathematics, then, in order to evaluate the security of the digital signature scheme, it is sufficient to study the difficulty of the underlying problem. The difficulty of the underlying mathematical problem can be studied thoroughly and, therefore, the level of security of the signature scheme is easier to estimate.

Such a security reduction also solves another problem. It is possible that a weakness of a digital signature scheme is discovered, for example, by a government agency of some country. That agency may then try to keep this weakness secret and take advantage of it. However, if the weakness of the signature scheme implies that a basic computational problem is no longer intractable, then keeping this weakness secret may be more difficult, since the solution of important scientific problems can be expected to happen at the same time in different places. Hence, reduction proofs make it less likely that a security hole can be abused.

It is an important question what the computational problems are to which the security of digital signature schemes should be reduced in order for the scheme to be considered more secure. Clearly, breaking a digital signature scheme in one of the ways explained below, can be considered to be a computational problem. In this sense, the security of any digital signature scheme can be trivially reduced to the intractability of a computational problem, namely to the problem of breaking itself. However, evaluating the computational difficulty of this problem is very difficult, since it is very complex and has many parameters. This is even more true since breaking a digital signature scheme is a so called *interactive problem*, that is, in that problem several parties are involved: a signer, who knows his secret key, an attacker who does not know that key but wants to generate valid signatures of the signer, and perhaps an honest verifier. In the process of forging signatures the attacker can try to use the help of the signer and the verifier (see Section 3.1.5).

To make the security level of a digital signature scheme easy to evaluate, it is desirable to reduce its security to easy to specify *non-interactive* computational problems. An example is the factoring problem for RSA-modules: Given an integer n which is the product of two large primes p and q, find those factors p and q. It would be optimal to reduce the security of a digital signature scheme to problems which are of mathematical interest independently of their cryptographic applications. Then, the difficulty of those problems would be studied also outside the crypto community and would therefore be easier to evaluate. However, digital signature schemes whose security can be reduced to such problems seem not to be known. In the known reductions, the computational problems depend to a certain extent on the specific digital signature scheme whose security is reduced to them. This is also true in our context and I will discuss this below. In my opinion, the less the computational problems depend on the digital signature schemes the stronger the security proof by reduction is.

3.1.2 Security of the secret key

A minimum requirement for secure digital signature schemes is the security of the secret key. An attacker has access to the public key of the signer. In a secure digital signature scheme, the determination of the secret key from the public key must be infeasible. In the digital signature scheme under review the security of the secret key can be reduced to well studied intractable problems. This will be explained below.

3.1.3 Existential forgery

Suppose that the problem of computing the secret key from the public key is intractable. This does not necessarily mean that the digital signature scheme is secure. It may still be possible that an attacker is able to generate valid signatures without the knowledge of the secret key.

In an *existential forgery* the attacker produces such a signature. In such a forgery, the attacker is not required to have control over the document which is signed. The only requirement is, that the result of an existential forgery a new signature of some document which has been produced without the knowledge of the secret key.

3.1.4 No message attacks

A *no message attack* or a *passive attack* is an existential forgery in which the attacker only knows the public key of the signer and has no access to further information such as valid signatures of other documents. A digital signature scheme is considered to be secure against no message attacks, if the possibility of such an attack implies the ability of solving a computational problem which is considered to be intractable.

3.1.5 Adaptive chosen message attacks

I explain the strongest security notion known for digital signature schemes: the security against existential forgery using an *adaptive chosen message attack*.

In an *adaptive chosen message attack* the adversary knows the public signature key of the signer and obtains valid signatures of a sequence of messages of his choice. The messages in the sequence may depend on signatures of previous messages. The goal of the adversary is an *existential forgery*, i.e. he wants to produce a new signature which has not been generated by the legitimate signer. In particular, the signature is not in the sequence of messages whose signatures the attacker has obtained. But the newly signed message is not necessarily a message of the attackers choice.

One practical application of this notion is as follows. Suppose that a signature scheme is used in a challenge response identification, for example in the ESIGN identification. Then the verifier generates challenges which the prover is supposed to sign, thereby proving his identity. Those challenges can be generated as a sequence of adaptive chosen messages. If an adaptive chosen message attack makes existential forgery possible, then the verifier is able to forge valid signatures without knowing the secret key. The use of signature schemes in challenge response identification is quite common.

I explain a method for proving security against chosen message attacks more precisely. In a chosen message attack the attacker can generate a sequence of pairs (message, signature). A message in that sequence may depend on the previous pairs. The signature generation algorithm is probabilistic. Therefore, the signatures are generated according to some probability distribution. The signature scheme is considered secure against a chosen message attack if it is secure against no message attacks and if without using the secret key it is possible to generate a sequence which is algorithmically indistinguishable (see [1]) from the sequence which is generated using the signature algorithm. The idea of this concept is the following. If an existential forgery is possible using an adaptively chosen sequence of pairs (message, signature), then an existential forgery is possible using the algorithmically indistinguishable simulation of such a sequence. This latter existential forgery is a no message attack since the signing algorithm is not used. However, the digital signature scheme is known to be secure against no message attacks. Therefore, an adaptive chosen message attack is impossible.

It is common belief that security against adaptive chosen message attacks is the strongest possible security notion for digital signature schemes. In other words, no attack against a digital signature scheme is known which cannot be modeled as an adaptive chosen message attack. The role of this security notion is somewhat similar to the role of the model of a Turing machine in the theory of computation. No computing device is known which cannot be modeled as a Turing machine. However, no proof is known that no stronger computing model exists. Likewise, no proof is known that the security against adaptive chosen message attacks is the strongest possible security notion.

In my opinion, if a signature scheme is proven secure against adaptive chosen message attacks, then it can be considered secure in the strongest sense. However, there are no such proofs but only reductions (see Section 3.1.1).

3.1.6 Random oracle model

Security proofs for digital signature schemes are difficult since a digital signature scheme consists of many components and their interaction may be complicated. An important ingredient of most signature schemes are cryptographically secure hash functions. The hash functions map very long messages to short strings of fixed length. The security of hash functions is discussed in Section 4.4. There I explain that no provably secure cryptographic hash functions are known.

If the security of a signature scheme is analyzed in the random oracle model (see [13], [4]), then the concrete hash function which is used in the digital signature scheme, is replaced by a so called *random oracle*. A random oracle can be viewed as a black box which contains a random function which maps long strings to short strings of fixed length. Nothing is known about this function, but it can be evaluated by making an explicit query. A typical proof of security against passive attacks in the random oracle model works as follows. If it is possible to come up with a forged signature for a document using one random oracle then such a forgery is also possible with another random oracle, resulting in another falsified signature (forking lemma, see [13]). The two valid signatures of the same document can then be used to solve an underlying mathematical problem.

Does a security proof in the random oracle imply the security of the real digital signature scheme in which a concrete hash function is used? Such an implication cannot be proved today. However, assuming that the concrete hash function behaves like a random oracle, a security proof in the random oracle model makes the security of the real scheme more plausible. On the other hand, there exist insecure signature schemes that can be proved secure in the random oracle model (see [4]). Those schemes look fairly artificial. Nevertheless, their existence raises the question what security proofs in the random oracle model really prove.

In my opinion, security proofs in the random oracle cannot prove the security of digital signature schemes but they make their security more plausible.

3.2 ECDSA

3.2.1 Security of the secret key

Computing the secret ECDSA key from the public ECDSA key is equivalent to solving the discrete logarithm problem in the elliptic curve point group specified in the ECDSA variants.

3.2.2 Security reductions

No reduction of the security of ECDSA to a basic mathematical problem such as the discrete logarithm problem in the group of points on an elliptic curve over a finite field is known. However, a modification as described in [13] would allow a security proof for an ECDSA variant in the random oracle model. Apparently, the ECDSA designers want to be compliant with current standards. This requires the choice of the present ECDSA variant which admits no security proofs.

Chapter 4

Basic computational problems

In this chapter I describe the basic computational problems which, given current knowledge, have to be solved in order for ECDSA to be insecure and I evaluate the difficulty of solving those problems.

4.1 Basics

In this section I explain the terminology which is used in this chapter.

4.1.1 Asymptotic complexity

To estimate the running time and storage requirement of the algorithms that solve the basic problems the function

$$L_x[u, v] = e^{v(\log x)^u (\log \log x)^{1-u}}$$

is used, where x, u, v are positive real numbers. I explain the meaning of this function. We have

$$L_x[0, v] = e^{v(\log x)^0 (\log \log x)^1} = (\log x)^v \tag{4.1}$$

and

$$L_x[1, v] = e^{v(\log x^1 (\log \log x)^0} = e^{v \log x}. \tag{4.2}$$

Let x be a positive integer which is the input for an algorithm. In the context of this evaluation, x is the cardinality of the finite field over which an elliptic curve is considered. The binary length of x is $\lfloor \log_2 x \rfloor + 1$.

If an algorithm has running time $L_x[0, v]$, then by (4.1) it is a polynomial time algorithm. Its complexity is bounded by a polynomial in the size of the input. The algorithm is considered efficient, although its real efficiency depends on the degree v of the polynomial.

If the algorithm has running time $L_x[1, v]$, then by (4.2) it is exponential. Its complexity is bounded by an exponential function in the length of the input. The algorithm is considered inefficient.

If the algorithm has running time $L_x[u, v]$ with $0 < u < 1$, then it is *subexponential*. The algorithm is slower than polynomial but faster than exponential. So the function $L_x[u, v]$ can be viewed as a linear interpolation between polynomial time and exponential time.

4.1.2 Practical run times

As usual, the experimental run times of the algorithms are given in *MIPS Years*. One MIPS Year is defined as the amount of computation that can be performed in one year on a single DEC VAX 11/780. Using this terminology, one year of computing on an n-MHz PC is comparable to n MIPS Years. However, this is only a rough estimate of the computing power used, since the computation may have space intensive parts, such as the solution of large linear systems, which cannot be executed on a PC.

4.1.3 Quantum computers

In the early 1980s, Richard Feynman (among others) introduced the idea of a new computing device which is based on the laws of quantum mechanics. Peter Shor [17] was able to prove that on such a quantum computer the integer factoring problem (IFP) and the discrete logarithm problem in finite fields have polynomial time solutions. This means that all cryptosystems under consideration here and, more generally, all public-key cryptosystems which are currently being used in practice, are insecure, if quantum computers become practical. There are first experiments with quantum computers, for example at Los Alamos. However, it is unclear whether quantum computers will ever be practical. For the time being, quantum attacks are not feasible. However, it is necessary to watch the development in the area of quantum computing. Also, it appears to be necessary to develop new digital signature schemes which remain secure even if quantum computers become practical.

4.2 Discrete logarithms on elliptic curves

The security of ECDSA depends on the intractability of the discrete logarithm problem on elliptic curves over finite fields (ECDL) which I discuss in this section.

4.2.1 The problem

Cryptography with elliptic curves over finite fields (used in ECDSA and My-Ellty) uses the following setting. F is a finite field of cardinality q. E is an elliptic curve over F. G is a point on E of prime order l. The *domain parameters* F, E, G, and l are publicly known.

The elliptic curve discrete logarithm problem (ECDL) is the following: Given a point Y in the subgroup generated by G, find an integer $v \in \{0, \ldots, l-1\}$ such that $Y = vG$.

By a theorem of Hasse, the order of the group of points on E over F is $q + 1 - t$ with $|t| \leq 2\sqrt{q}$. The number t is called the *trace* of the curve (see [11]).

4.2.2 Square root attacks

The only known general purpose algorithms for ECDL are *generic DL-algorithms.* They work in any cyclic group as long as it is known, how the group elements are multiplied, inverted and how the equality of group elements can be decided.

If the group order including its prime factorization is known, then the Pohlig-Hellman algorithm reduces the DL problem in the full group in polynomial time to discrete logarithm problems in groups whose orders are the prime divisors of the group order (see [3]). Since the order of the elliptic curve point group generated by P is a prime number, the Pohlig-Hellman algorithm gives no advantage.

The fastest generic discrete logarithm algorithm in a cyclic group of prime order is the parallel Pollard ρ-algorithm (see [20], [5]). Its running time is proportional to $\sqrt{l}/r = L_l[1, 1/2]/r$ where l is the group order and r is the number of processors used. Using this algorithm it was possible to compute discrete logarithms in an elliptic curve point group over a finite prime field where the group order is a 97-bit prime using $2 * 10^{14}$ group operations (see [5]). Also, it was possible to compute the discrete logarithm in an elliptic curve point group with a 108-bit order over a finite field of characteristic 2 using $2.3 * 10^{15}$ group operations (see [5]).

Using those data points and the complexity of the parallel Pollard ρ-algorithm, Lenstra and Verheul [12] estimate that for the next 20 years ECDL is intractable in an elliptic curve point group of prime order $l > 2^{161}$. If algorithmic progress is taken into account then $l > 2^{188}$ is recommended.

4.2.3 ECDL algorithms for special curves

There are a number of algorithms which solve ECDL for specific classes of curves.

Frey-Rück attack From [7] we obtain the following: Let

$$k = \min\{i \in \mathbb{Z}_{>0} : q^i \equiv 1 \bmod l\}. \tag{4.3}$$

where l, q are as in Section 4.2.1. In other words, k is the order of q in the group $\mathbb{F}_l^*$. Then there is a polynomial time reduction of the discrete logarithm problem on the elliptic curve to the discrete logarithm problem in the multiplicative group of the finite field $\mathbb{F}_{q^k}$. For fixed k the discrete logarithm problem in $\mathbb{F}_{q^k}$ can be solved in time $L_{q^k}[1/3, c]$ for some constant c (see [14]). For small k this subexponential attack is much faster than all known general purpose ECDL-algorithms (see Section 4.2.2). To prevent the Frey-Rück attack, it is necessary to choose k at least $\lceil 2000/\log_2(q) \rceil$. This condition is based on the assumption that the discrete logarithm problem in a finite field, whose cardinality is a 2000-bit number, is intractable. For a

160-bit prime p this means that $k \geq 13$. I remark that the German Information Security Agency requires $k \geq 10^4$ since some researchers feal that more efficient attacks are possible. But no such attacks are known.

Anomalous curve attack If the trace of the curve E is 1, then ECDL can be reduced to a discrete logarithm problem in the additive group $\mathbb{Z}/q\mathbb{Z}$ (see [18]) which can be solved by the extended euclidean algorithm in polynomial time. Therefore, curves of trace 1 must be avoided in cryptographic applications.

Weil descent attacks If $q = 2^{nm}$ with small n such as $n = 4$ then ECDL can be reduced to a DL problem on a hyperelliptic curve which by an algorithm of Gaudry can be solved faster then the general ECDL problem (see [8]). Therefore, those field sizes must be avoided.

4.2.4 Quantum attacks

Using the methods from [17] it is possible to show that ECDL can be solved in polynomial time on a quantum computer. It is not yet clear whether quantum computers become ever practical. Currently, quantum attacks do not threaten ECDL.

4.3 Random and pseudorandom number generation

The key and signature generation in ECDSA require the generation of random numbers, specifically random primes. They are generated as sequences of random bits.

Such a sequence is generated as follows. A random bit generator (RBG) is used to generate a short sequence of true random bits. Since a RBG is too inefficient, the short true random sequence is expanded by a pseudorandom bit generator (PRBG) into a sequence of the necessary length.

The RBGs used by the scheme under review are not described in the submission. I can therefore not discuss their security here. However, in real applications it is necessary that a cryptographically secure RBG is used.

A PRBG receives as input a random bit sequence and outputs a longer pseudorandom bit sequence. A PRBG is cryptographically secure if an attacker is not able to distinguish its output from a true random sequence in polynomial time.

No provably secure PRBG is known. Several PRBGs are known whose security can be reduced to the intractability of certain number theoretic problems such as the discrete logarithm problem in finite fields (see [3]). However, those PRBGs are not sufficiently efficient.

The PRBG used in practice survive a broad class of statistical tests specified, for example, in [6].

4.4 Hash functions

In signature schemes, hash functions are used to map long documents to shorzt bit strings of a fixed length, which are actually signed. A *collision* of a hash function is a pair of different documents which are mapped to the same hash value. A hash function is called *collision resistant* if finding a collision of that hash function is intractable.

In signature schemes, which sign hash values, the used hash functions must be collision resistant. Otherwise, if a collision (d, d') is found then the two documents d and d' have the same signature. If an attacker is able to obtain a valid signature of d, then he has also a signature for d'. For example, if the signer signs d in a challenge-response authentication, then he has also signed the other document d', possibly without knowing it. Collision resistant hash function are one way functions. This means, that computing an inverse image for a given image is intractable. Therefore, in many cases, the use of collision resistant hash functions prevents existential forgeries since even if it is possible to generate a valid signature for a hash value it is impossible to find a document with that hash value.

Using the birthday paradox (see [3]) a collision for a hash function whose image has n elements can be found with probability $> 1/2$ by computing approximately $\sqrt{n}$ hash values. Therefore, the image of the hash function should at least contain 2^{160} elements.

No hash function is known for which the birthday attack is provably the only possible attack. In the past, hash functions such as MD4 have been shown not to be collision resistant. Today, the hash functions SHA-1 [19] and RipeMD-160 [10] are used in practice. Given current knowledge, they are collision resistant.

The birthday attack can be prevented if a *keyed hash function* is used. This is a function which maps a bit string and a key from a predefined key space to a hash value of fixed length. In the signature process, a random key is generated. The signature algorithm signs the hash value of a document that is generated by the hash function which is parameterized by the chosen key. The key is part of the signature. It is also used in the verification process. In order for digital signature algorithm to be secure, the keyed hash function must be a *universal one-way hash function* (UOWF). This means that given a hash value and a key it is intractable to find a document such that the value of the hash function parameterized by the given key is the given hash value. No provably universal one-way hash function is known. However, there are constructions that use compression functions such as SHA-1 (see [19]) and are assumed to have the universal one-way property.

Chapter 5

Basic cryptographic problems in ECDSA

In this chapter I discuss the hardness of the non-interactive computational problems which are the basis of the security of ECDSA.

5.1 ECDSA-$\mathbb{F}_p$

I use the notation from Section 2.2.

If ECDL can be solved in the subgroup generated by G then ECDSA is insecure because the secret ECDSA key can be obtained from the public ECDSA key.

The choice of the parameters in ECDSA-$\mathbb{F}_p$ prevents the use of the Frey-Rück attack and the anomalous curve attack. In fact, the integer k from (4.3) is at least 20 and I have explained in Section 4.2.3 that $k \geq 13$ is sufficient. The Weil descent attack does not work over prime fields. For $n \geq 160$ square root attacks are infeasible for the next 20 years (see Section 4.2.2 and [12]). Therefore, given the current algorithmic knowledge, ECDL in the subgroup generated by G will remain intractable for the next 20 years. This is also true for the special curves which for compatibility reasons are suggested in [16]) since no attacks are known which take advantage of the special form of the curves.

5.2 ECDSA-$\mathbb{F}_{2^n}$

If ECDL can be solved in the subgroup generated by G then ECDSA is insecure because the secret ECDSA key can be obtained from the public ECDSA key.

The choice of the parameters in ECDSA-$\mathbb{F}_{2^n}$ prevents the use of the Frey-Rück attack and the anomalous curve attack. In fact, the integer k from (4.3) is at least 20 and I have explained in Section 4.2.3 that $k \geq 13$ is sufficient. The Weil descent attack does not work over fields of characteristic 2^m where m is a prime number. For $m \geq 160$ square root attacks are infeasible for the next 20 years (see Section 4.2.2 and [12]). Therefore, given the current algorithmic knowledge, ECDL in the subgroup generated by G will remain intractable for the next 20 years. This is also true for the special curves which for compatibility reasons are suggested in [16]) since no attacks are known which take advantage of the special form of the curves.

5.3 Hash function

The hash function used in ECDSA is SHA-1 (see [19]). Given the current algorithmic knowledge, SHA-1 will be collision resistant for the next 20 years.

5.3.1 Random numbers

No specific random number or pseudorandom number generator is suggested.

Chapter 6

Final evaluation

No security reductions are given for ECDSA. But ECDSA is a well studied standard which has been reviewed by many cryptanalysts.

With respect to the underlying number theoretic problem, ECDSA using random curves (see Sections 5.1 and 5.2) appears to be a very strong system. The fastest algorithm for the general ECDL, on which is the security of ECDSA is based, is of exponential complexity (see Section 4.2.2). Also, if the curves are randomly chosen, then special attacks are very unlikely to be successful.

In practice, ECDSA with the specific curves from the standards is used (see [16]) since those curves have an easy encoding and are compatible with existing and implemented standards. Given present knowledge, the curves from the standards are as secure as random curves. They are constructed in such a way that the known attacks are impossible.

The hash function used in ECDSA is plain SHA-1. The only known attack for that hash function is the birthday attack which is expected to be infeasible for the next 20 years (see Section 4.4).

Nothing is said in ECDSA submission about the generation of random numbers or pseudorandom numbers. If the corresponding standards are used then the random number generation in ECDSA can be considered secure.

In my opinion, ECDSA with the parameters recommended in this evaluation, is a secure digital signature scheme, although nor formal security proofs exist. I recommend that a future version of ECDSA should be one which admits a security reduction.

I feal that unexpected mathematical breakthroughs in all areas are the most serious threat for ECDSA. It is therefore crucial, that ECDSA is implemented in such a way that it can be easily replaced if it turns out to be insecure.

Bibliography

[1] BELLARE, M., AND GOLDWASSER, S. Lecture notes on cryptography. `www-cse.ucsd.edu/usres/mihir`.

[2] BLEICHENBACHER, D. Chosen ciphertext attacks against protocols based on the rsa encryption standard pkcs # 1. In *Advances in Cryptology - Crypto '98* (1998), pp. 1–12.

[3] BUCHMANN, J. *Introduction to Cryptography.* Springer-Verlag, New York, 2000.

[4] CANETTI, R., GOLDREICH, O., AND HALEVI, S. The random oracle methodology revisited. In *30th ACM Symp. on Theory of Computing (STOC)* (1998), pp. 209–218.

[5] ESCOTT, A. E., SAGER, J. C., AND SELKIRK, A. P. L. Attacking elliptic curve cryptosystems using the parallel pollard rho method. *Cryptobytes 4* (1999), 15–19.

[6] FIPS 140-1, security requirements for cryptographic modules. Federal Information Processind Standards Publication 140-1, U.S. Department of Commerce/N.I.S.T., National Technical Information Service, Springfield, Virginia, 1994.

[7] FREY, G., AND RÜCK, H. A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves. *Math. Comp. 62* (1991), 865–874.

[8] GAUDRY, P., HESS, F., AND SMART, N. Constructive and destructive facets of weil decent on elliptic curves. *J. Cryptology* (to appear).

[9] IEEE P1363 Draft. `http://grouper.ieee.org/groups/1363/P1363/`.

[10] ISO/IEC 10118-3, information technology - security techniques - hash-functions - part 3: Dedicated hash-functions. draft (CD), 1996.

[11] Koblitz, N. *A Course in Number Theory and Cryptography.* Springer-Verlag, 1994.

[12] Lenstra, A., and E.R.Verheul. Selecting cryptographic key sizes, October 1999.

[13] Pointcheval, D., and Stern, J. Security arguments for digital signatures and blind signatures. *J. Cryptology 13* (2000), 361–396.

[14] Schirokauer, O., Weber, D., and Denny, T. Discrete logarithms: the effectiveness of the index calculus method. In *ANTS II* (Berlin, 1996), H. Cohen, Ed., vol. 1122 of *Lecture Notes in Computer Science*, Springer-Verlag.

[15] SEC 1 : Elliptic curve cryptography. Certicom Research, September 20 2000.

[16] SEC 2 : Elliptic curve cryptography. Certicom Research, September 20 2000.

[17] Shor, P. W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Computing 26* (1997), 1484–1509.

[18] Smart, N. The discrete logarithm on elliptic curves of trace one. *J. Cryptology 12* (1999), 193–196.

[19] standard, s. h. National Institute of Standars and Technology (NIST), FIPS Publication 180-1, April 1995.

[20] Van Oorschot, P., and Wiener, M. Parallel collision search with cryptanalytic applications. *J. Cryptology 12* (1999), 1–28.

Index