ELLIPTIC CURVE CRYPTOGRAPHY: PRE AND POST QUANTUM

JEREMY WOHLWEND

Abstract. Public-key cryptography has been at the center of online communication and information transfer for decades. With computing power growing at an exponential rate, some of the most widely used encryption schemes are starting to show their limits. The RSA algorithm, which is still widely used around the world, now requires very large keys to ensure security. Since these systems may appear on low computing power devices such as mobile phones, or chips, it has become essential to create protocols for which we can reach the same level of security without spending considerable computing power setting up the system in the first place. Elliptic curve cryptography (ECC) provides an exciting alternative to RSA, and has shown to be a lot more efficient in terms of key size. In this paper, we provide a description of how elliptic curves are used in modern cryptography, as well as their current limitations and future prospects. Because quantum computers pose a serious threat to the currently in use public-key systems, we also describe the recent progress on super singular elliptic curves isogenies, which may offer a quantum resistant cryptosystem and a viable alternative for the future of elliptic curve based cryptography.

1. Introduction and History

Up until the 1970's, all the encryption in use around the world was based on *symmetric* ciphers, which means that in order for two parties to communicate securely, they must have had to previously meet and agree on a shared secret. While such methods provided all the security needed at the start of the 20^{th} century, the second World War and the rise of the internet and online information, motivated the idea of a cryptographic protocol where two parties could create a secure communication channel without any kind of previous communication.

This kind of cryptography is known as *asymmetric*, or *public-key cryptography*. The first to demonstrate the existence of such methods were Diffie, Hellman and Merkle in 1976 [9]. While their protocol is usually referred to as the Diffie-Helllman Key Exchange, in 2002, Hellman suggested that it should be called the Diffie-Helllman-Merkle Key Exchange, including Merkle in recognition of his contribution to the invention of public key cryptography [14]. Because asymmetric protocols tend to require relatively large amounts of computation, it is common to use them to send a small key which can then be used to establish a secure symmetric channel.

Key words and phrases. Elliptic curve cryptography, quantum computing, super singular elliptic curve isogeny.

Definition 1.1. We denote the discrete logarithm problem on the multiplicative group of the integers modulo p, $\mathbb{Z}_{/\mathbb{Z}_p}$, as follows. Given $g, a \in \mathbb{Z}_{/\mathbb{Z}_p}$, where a is a member of the cyclic subgroup generated by g, find an integer k such that:

$$g^k \equiv a \ mod \ p \tag{1.1}$$

The security of the Diffie-Hellman-Merkle protocol relies on the assumption that while g^a can be computed easily using repeated squaring, its counterpart, the discrete logarithm problem, is computationally hard. While there is no formal proof of the hardness of the problem, this assumption is widely conjectured to be true. In particular, there are groups under which the logarithm is easy but also groups where it is believed to be particularly hard. Elliptic curves are believed to be part of the latter, as is discussed see later in this paper, and to be considerably more resistant than Diffie-Hellman-Merkle and prime groups, for which efficient methods in practice exist, for instance, index calculus [4].

In 1977, Ron Rivest, Adi Shamir, and Leonard Adleman proposed another asymmetric encryption scheme, known as RSA [24]. The security of RSA is based on an similar problem to Diffie-Hellman-Merkle, namely, the discrete factoring problem.

Definition 1.2. We denote the discrete factoring problem as follows. Given a number N, the factor of two large primes p and q, find p and q.

RSA was the first public key algorithm to go in wide use, and is still adopted in many modern protocols. It improves on Diffie-Hellman-Merkle by providing signatures, and long term private keys. The factoring problem however, can be solved in sub-exponential time [22]. With the increase in computational power, the key sizes in the RSA protocol have had to grown at a similar rate, which raises questions about the scalability of the protocol on low power devices [13].

Elliptic curve cryptography was introduced in 1985 by Victor Miller and Neal Koblitz who both independently developed the idea of using elliptic curves as the basis of a group for the discrete logarithm problem. [16, 20]. Believed to provide more security than other groups and offering much smaller key sizes, elliptic curves quickly gained interest. In the early 2000's, the NSA made Elliptic curve its standard suite B algorithm for both encryption and signature and its use on low power devices has been shown to be much more scalable [13]. Elliptic curves are the main focus of this paper, which offers a primer to modern elliptic curve cryptography and discusses the future prospects of the protocol, including a recently published method of public key encryption based on elliptic curve isogenies, which is believed to offer a quantum resistant scheme.

This is of particular interests since Peter Shor showed, in 1994, that using a Quantum Computer, it would be possible to solve the integer factoring problem in polynomial time, something which is believed to be impossible on classical computers [28]. In his paper, Shor presented two similar algorithms, one for integer factoring, and the other one for the discrete logarithm problem. It was then shown that Shor's algorithm could be extended to the discrete logarithm problem on any abelian group [11]. While quantum computers remain mostly theoretical, the need for post quantum cryptography is crucial, and has already attracted many in the literature. Recently, in April 2016, numbers slightly larger than 200000 were factored using D-wave Quantum processors [10]. In August 2015, the NSA made a announced that they would soon move to a quantum-resistant algorithms suite to

replace ECC. Neal Koblitz, one of the founders of ECC, and Alfred J.Menezes recently published a paper discussing the NSA's decision [17].

As of 2016, it doesn't seem as though quantum computers will be able to break symmetric ciphers more efficiently than classical computers. In fact, most research currently supports the limitations of quantum computers [3]. The only known attack is based on Grover's search algorithm where the speedup is small enough that it can be compensated by doubling the key size of symmetric based cryptosystems such as AES or most hash functions. For these reasons we omit in this paper the discussion of symmetric cryptosystems, and focus on a public key method, ECC, which is based on an interesting mathematical problem and will require more attention from the cryptography community as Quantum Computers become a practical reality. In particular, the next section of the paper covers some of the core mathematical concepts needed to fully understand the elliptic curve based cryptography. The third section covers the concepts behind modern elliptic curve cryptography (ECC), including the Elliptic curve Diffie Hellman Key Exchange (ECDH) protocol and the Elliptic curve Digital signature algorithm (ECDSA). The fourth section summarizes some of the main classical attacks on ECC, and Shor's algorithm. Finally, the fifth section briefly describes a recently published method of public key encryption based on Elliptic curve isogenies, and which is believed to offer a quantum resistant scheme.

2. Preliminaries

Before diving into Elliptic Curve Cryptography, we make a quick review of field theory. Any familiar reader should feel comfortable moving to section 3.

Definition 2.1. A *group* is an algebraic structure composed of:

- A set of elements G
- A closed operation on the set G, which is associative. That is $(a \cdot b) \cdot c = a \cdot (b \cdot c)$, for $a, b, c \in G$
- An identity element
- The existence of inverses under the set operation

A group where the set operation is also commutative (i.e $a \cdot b = b \cdot a$) is known as an *abelian group*.

For instance, one might use + as the set operation and 0 as the identity element. Using these and the set of integers $\mathbb{Z}$, we get a valid group, where we use integers of opposite signs as pairs of inverses. On the contrary, the set of natural numbers $\mathbb{N}$ does not form a group as one cannot define inverses.

Definition 2.2. Let a be an element a group G with identity 1 and $\cdot$ as group operation. The *order* of a, is the smallest integer n such that:

$$\underbrace{a \cdot a \cdot a \cdot \ ... \ \cdot a}_{n} \ = \ 1 \tag{2.1}$$

The set $\{a, a^2, a^3, a^4, ..., a^n\}$ forms a cyclic subgroup of G of order n, where a is called a *generator* for that subgroup.

Definition 2.3. A *field* is an algebraic structure composed of:

- A set of elements G closed under multiplication and addition
- Addition and multiplication are both associative under the set G.
- Addition and multiplication are both commutative under the set G.
- The existence of inverses under both addition and multiplication
- Multiplication is distributive over addition: $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$.
- Distinct identities for multiplication and addition.

We call the *characteristic* of a field F the smallest number n such that summing n copies of 1 is equal to 0. For instance, if the characteristic of F, $char(F)$, is 2 and the identity in F is 1, then $1 + 1 = 0$. If $char(F) = 3$, then $1 + 1 + 1 = 0$.

Galois fields are fields that consist of a finite number of elements. One example of such filed is the integers modulo a prime number p, $\mathbb{Z}_{/\mathbb{Z}_p}$. With p elements, 0 to $p - 1$, we denote this set as $GF(p)$. Note that this field is the combination of two abelian groups, one under addition with identity 0, and one with multiplication and identity 1, where in the latter case 0 is not taken to be part of the group set to preserve the existence of inverses.

For the rest of this paper, we will be mostly interested in prime fields $GF(p)$, and more generally $GF(q)$ where $q = p^n$ is the power of a prime.

3. Elliptic Curve Cryptography (ECC)

While the idea of using elliptic curves in cryptography protocols was first introduced in the 1980's, it took about 20 years to see them become widely adopted. Cryptosystems based on elliptic curves follow a very similar construction to other protocols based on abelian groups, such as Diffie-Hellman-Merkle. As was discussed earlier, the discrete logarithm problem has an analog in elliptic curves groups on finite fields. This problem lies at the heart of elliptic curve cryptography where it is conjectured to be harder to solve than on the multiplicative group $\mathbb{Z}_p$.

3.1. Basics.

We define an elliptic curve $E(F)$ as a set of points in a filed F, satisfying an equation of the form:

$$y^2 + a_1xy + a_2y = x^3 + a_3x^2 + a_4x + a_5 \tag{3.1}$$

Where $a_1, a_2, ..a_5 \in F$. If we assume that the characteristic of the field is different than 2, then this equation can be simplified to:

$$y^2 = x^3 + a_3x^2 + a_4x + a_5 \tag{3.2}$$

This equation can be simplified further if the characteristic of the field is also different than 3, thus giving the more familiar equation, known as the Weierstrass normal form:

$$y^2 = x^3 + ax + b \mid_{a=a_4, b=a_5} \tag{3.3}$$

The following figure shows an elliptic curve in $\mathbb{R}^2$.

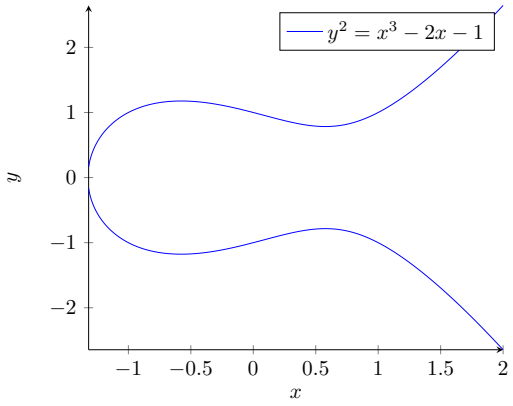

FIGURE 1. An elliptic curve E in $\mathbb{R}^2$

We can define a group G over the set of solutions to the curve equation as follows. The set of points on the curve E are the elements of the group G. We add another point, the "point at infinity" (∞, ∞), as the identity, and call it O.

Note that for each point $P = (x, y)$ on the curve, the point $P' = (x, -y)$ must also be on the curve. We let P and P' be inverses in the group G. In particular, $P + P' = O$ if '+' is the group law.

All we need now is a closed operation '+' over the group G. Given points P and Q on the curve E and members of G, let $R = P + Q$, be the point such that $P+Q-R = O$. In the field $\mathbb{R}^2$, there is an easy geometric construction for R which we illustrate below.

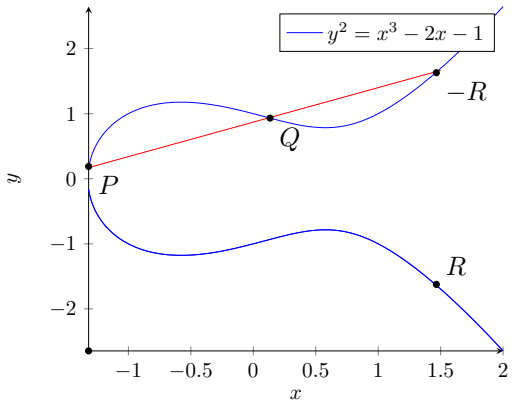

FIGURE 2. The group law of G in $\mathbb{R}^2$, : Given points P and Q on the curve E, we draw a line between P and Q. The line crosses the curve E in a third point, $-R$. Taking the inverse (i.e the symmetric across the x-axis), we get the point $R = P + Q$.

Note that such construction is always possible unless $P = Q$ or $Q = -P$ (i.e the line through P and Q is vertical). In the first case, one simply takes the tangent to the curve at the point P and use the second intersection with the curve as the point $-R$. In the second case, we let $P + -P = 0$, and similarly $P + O = P$.

If $P(x_1, y_1)$ and $Q(x_2, y_2)$ are distinct points of E, then we can express the coordinates of the point $R(x_3, y_3)$ by solving for the intersection between the line going through P and Q and the curve E:

$$\begin{cases} y = \frac{(y_2 - y_1) \cdot x + y_1 x_2 - y_2 x_1}{x_2 - x_1} \\ y^2 = x^3 + ax + b \end{cases}$$

Then the coordinates of $R(x_3, -y_3)$ are given by:

$$x_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_2 - x_1 \tag{3.4}$$

$$y_3 = -\left(y_2 + \frac{y_2 - y_1}{x_2 - x_1}(x_3 - x_2)\right) \tag{3.5}$$

If $P = Q$, then we take the intersection of the tangent line and the curve E, which gives a simpler expression for $R(x_2, y_2)$:

$$x_2 = \frac{3x_1^2 + a}{2y_1} - 2x_1 \tag{3.6}$$

$$y_2 = -\left(y_1 + \frac{3x_1^2 + a}{2y_1}(x_2 - x_1)\right) \tag{3.7}$$

Note that this expression includes the parameter a of the curve and not b as we take the derivative of the curve to get the slope of the tangent line, and b vanishes. Using this operation, we define *scalar multiplication* on the group G, by repeatedly adding a point to itself: $nP = \underbrace{P + P + ... + P}_{n}$. One can compute nP efficiently using a variation of repeated squaring, the double and add method:

Algorithm 1 Double and Add

Input: n, P
Output: out = nP
1: out = 0
2: m = $log_2(n) + 1$
3: **for** $i = m$ to 0 **do**
4: out = 2 out
5: **if** *the* i^{th} *bit of* n *is 1* **then**
6: out = out + P
7: **end if**
8: **end for**
9: **return** *out*

With this algorithm, we can compute nP in $O(m)$ or equivalently, $O(log(n))$, which is exponentially better than the naive linear algorithm, which consists of adding P to itself n times.

Using scalar multiplication, one can create subgroups of G, where the elements of the subgroup are the multiples of a point P on $E(F)$. Note that $nP + mP =$

$(P+P+P..+P)+(P+...+P) = (n+m)P$. This implies that adding multiples of P yields another multiple of P, hence the subgroup of G formed by the multiples of a point P is closed under the group law. Furthermore, the multiples of P form a cyclic subgroup with some order k. Here P is a generator for the subgroup.

We would like to know the order of the subgroup generated by a point P. Lagrange's theorem states that the order of the subgroup generated by P must be a divisor of the order of the group G. This implies that, knowing the order of G, one could generate all its divisors and test each of them in increasing order and choose the first n such that $nP = 0$.

Luckily, Schoof's algorithm allows us to compute the order of the set of points on the curve $E(F)$, where F is a finite field, in polynomial time. This algorithm, presented in 1985, improved on the previously exponential algorithms for counting the number of points on en elliptic curve over a discrete field [26, 27]. From there, Largrange's theorem implies that if the order of the curve is prime, then the subgroup order is either 1 or p. If the order is p, then all the points on $E(F)$ are multiples of each other, or put simply, all the points are generators for that subgroup. Therefore, given a curve E, one can check efficiently that the order of E is a prime number q and that therefore all subgroups of G on $E(F)$, have order q.

3.2. **Discrete Logarithm and Cryptography.** While computing nP can be done efficiently using the double and add algorithm, the opposite problem, namely finding n given P and nP can be made a particularly hard problem when the elliptic curve is taken over a finite field. This is known as the elliptic curve discrete logarithm problem for which no efficient algorithm is known. Furthermore, while the discrete factoring problem, for instance, can be solve in sub exponential time, the best algorithms known to solve the elliptic curve discrete logarithm are purely exponential [5].

Definition 3.1. Let E be an elliptic curve on some finite field F, and G the group containing the points of E, and with group law + defined as above. Given a point P in the group G, and a multiple of P, call it $Q = kP$ where k is a scalar, the discrete logarithm problem on elliptic curve is stated as follows:

Given P and Q, find k, the discrete logarithm of Q in base P. In other words, find the smallest integer k such that $kP = Q$.

Consider the field $\mathbb{F}_p$ of integers modulo p, where p is prime. This field, because it is finite, is sometimes referred to as $GF(p)$. In this field we can define division by taking the multiplicative inverse and using the multiplicative law, such that $x/y = x \cdot y^{-1}$. Computing multiplicative inverses can be done in $O(log(p))$ using the extended Euclidean algorithm. Now a point $P(x, y), x, y \in \mathbb{F}_p^2$, is on the curve if and only if, for $a, b \in: \mathbb{F}_p^2$, parameters of the curve, the following relationship is verified:

$$y^2 \equiv x^3 + ax + b \quad mod\ p \tag{3.8}$$

We now define the group $E(\mathbb{F}_p^2)$, as the set of points in $\mathbb{F}_p^2$ which are on the curve E. $E(\mathbb{F}_p^2)$ forms an abelian group in the finite field $\mathbb{F}_p^2$, and the formulas given in section 2 still apply, though one needs to do all the computation mod p.

For the discrete logarithm problem to be difficult on elliptic curves, the order of the cyclic subgroup needs to be large. All this requires is that the order of the group G be a non smooth integer (i.e it is divisible by a large prime), and to find an appropriate base point P. Let h the cofactor of the group G be a positive integer such that $h \cdot k = |E(F)|$, where k is the order of the subgroup generated by P. In cryptography protocols, we usually try to have $h \leq 4$.

We now have a complete method for constructing the domain parameters required to set up the Elliptic Curve protocol, namely (p, a, b, G, k, h).

Algorithm 2 Constructing domain parameters

Input: p
Output: $out = (p, a, b, P, k, h)$
1: Select a random curve E on the field $\mathbb{F}_p$, that is find random a and b.
2: Define the group G on $E(F)$, and use Schoof's algorithm to find the order of G.
3: Compute the divisors of $|G|$, and ensure that one of them is a large prime k. In other words, find $hk = |G|$, such that the cofactor h of the group G is small, say ≤ 4.
4: **if** *the order of G is smooth* **then**
5: Select a new curve, back to step 1
6: **end if**
7: Select a point P in G at random and compute nP using the double and add algorithm.
8: **if** *nP != 0* **then**
9: n is not the order of the subgroup generated by P, go back to step 7.
10: **else**
11: k = n
12: **end if**
13: **return** $out = (p, a, b, P, k, h)$

Note that this procedure is computationally expensive. Curves are thus usually pre-computed and form a families with particular properties. The main families of curves used in ECC are presented next.

3.3. **Special curves.** There are special curves which are used in cartographic protocols to optimize certain computational steps. We give below the main families of such curves:

- *Koblitz curves over binary fields*: these are of the form $y^2+xy = x^3+ax^2+1$ with $a \in \{0, 1\}$ over $GF(2^m)$ for m prime. These allow particularly fast addition and scalar multiplication.
- *Binary curves*: these have the form $y^2 + xy = x^3 + x^2 + b$, where b is a random number, and also allow efficient addition and scalar multiplication.
- *Edward curves*: these are of the form $x^2 + y^2 = 1 + ax^2y^2$ with $a \in \{0, 1\}$. These also fast addition and scalar multiplication but also a unique group law operation which doesn't change if $P = Q, P \neq Q,$ *or* $P = -Q$. These are useful in avoiding side channel attacks which happen when an adversary tries to learn information from the time the algorithm takes to compute.

- *Weierstrass curves over prime fields*: These are the curves which were discussed so far in this paper. They are believed to offer more security than the Koblitz or Binary curves, although not as efficient of a computation.

In particular, NIST provides the following set of curves which can be used for ECC protocols. Specifically, FIPS 186-3 recommends 10 finite fields [1]:

- Five prime fields $\mathbb{F}_p$ for certain primes p of sizes 192, 224, 256, 384, and 521 bits. Each prime is associated with a curve, for more or less security.
- Five binary fields $\mathbb{F}_{2^m}$ for m equal 163, 233, 283, 409, and 571. For each of the binary fields, one elliptic curve and one Koblitz curve is recommended.

These curves all offer efficient computation and can be used in practice, assuming of course that NIST did not find a way to make people use weak curves. There exists, indeed, curves which are considered "weak", and should be avoided. We discuss these in section 4 of this paper. Before this, the two protocols currently suggested by the NSA in its suite-B algorithms are presented: the Elliptic Curve Diffie-Hellman key exchange, and the Elliptic Curve Digital Signature Algorithm.

3.4. **Elliptic curve Diffie-Hellman (ECDH).** Alice and Bob would like to communicate over a secured channel using elliptic curve cryptography, avoiding the man-in-the-middle attack. First Alice and Bob agree on a set of domain parameters, as defined in the previous section. In the context of this protocol, there are two types of keys:

- The **public key** in this cryptosystem is a point $Q = nP$
- The corresponding **private key** is the scalar n which takes P to Q.

ECDH is a key exchange protocol. We omit here the details of encryption using the keys, which can be done in many ways, and focus instead on the establishment of a shared secret between Alice and Bob. The protocol runs as follows:

(1) Alice and Bob each generate a public key and a private key pair $(Q = nP, n)$. Note that Alice and Bob both use the point P as a generator. We now have two pairs (Q_A, n_A) and (Q_B, n_B).
(2) Alice and Bob exchange their public key over a potentially insecure channel. An eavesdropper may learn Q_A or Q_B but won't be able to compute n_A or n_B, thanks to the difficulty of the discrete logarithm problem.
(3) Alice computes $n_A \cdot Q_B$, and Bob computes $n_B \cdot Q_A$. Alice and Bob now share the point $n_A n_B P$. They can then use the x or y coordinate of the shared point to establish a secured symmetric channel for communication.

3.5. **Elliptic Curve Digital Signature Algorithm (ECDSA).** Now, let's say that Alice wants to sign a message m so that Bob knows without a doubt that the message came from Alice. This is the ECDSA protocol, which works as follows:

(1) For some message m, Alice takes a hash of m and truncates it so that it has bit-length k, where k is the order of the subgroup generated by P in G. Let z be the resulting truncated hash of m.
(2) Alice chooses an integer n such that $1 \leq n \leq k$
(3) Alice computes $Q = nP$ and r, the x-coordinate of the point Q, modulo k. If $r = 0$, then go back to step 1.
(4) Alice then calculates $s = n^{-1}(z + rn_A)$, where n_A is Alice's private key and n^{-1} is the multiplicative inverse of n mod k. If $s = 0$, go back to step 1.

(5) The pair (r, s) forms the signature.

Note that step 4 requires computing the multiplicative inverse of n mod k. However this is only possible when the order of the subgroup of G is itself prime. Curves with prime order are therefore essential in the use of ECDSA. Now to verify Alice's signature, Bob uses Alice's public key A, and proceeds as follows:

(1) Bob computes $u_1 = s^{-1}z$ mod z
(2) Bob computes $u_2 = s^{-1}r$ mod z
(3) Finally, Bob computes the point $Q = u_1P + u_2A$ and verifies $r = x_Q \ mod \ n$

A proof of correctness is presented below:

Proof. We start with Q as described above. That is $Q = u_1P + u_2A$, where A is Alice's public key, which is equal to n_AG, for some number n_A, which is Alice's private key. Then:

$$Q = u_1P + u_2d_AP \tag{3.9}$$
$$= (u_1 + u_2d_A)P \tag{3.10}$$

Now let s be such that $s = n^{-1}(z + rn_A)$. Then:

$$Q = (s^{-1}z + s^{-1}rd_A)P \tag{3.11}$$
$$= s^{-1}(z + rd_A)P \tag{3.12}$$
$$= nP \tag{3.13}$$

So in both cases we found the point Q, which would have only been possible if the message was signed by Alice's private key. This completes the proof. □

Elliptic curve cryptography takes an edge over RSA because of the shortness of its keys for virtually the same level of security, which has made it popular in the past decade [13]. One of the main disadvantage of ECC, however, is that the curves usually need to be pre-computed in advance, as the domain parameters are expensive to generate. Thus, there is the need to do a thorough analysis of the curves used, when given by a third-party, like NIST. Furthermore, just like RSA, Elliptic curve cryptography is subject to some classical attacks and most importantly the same quantum attack, which promises to break RSA: Shor's algorithm. Until then however, ECC, which has shown great promises in the past decade, will likely continue to be used for the years to come.

4. Classical and Quantum attacks on ECC

Elliptic curve cryptography, in the format presented in this paper, is vulnerable to attacks by classical and quantum computers. In the classical case, the most efficient algorithms have purely exponential running time. In the quantum case, however, there exists a variant of Shor's algorithm which can solve the elliptic curve discrete logarithm problem in polynomial time. Presented below are some of the weak curves to avoid in ECC protocols, and the reasons behind Sony's attack a few years ago. Then, the Pollard's ρ algorithm is presented, as well as a brief overview of Shor's algorithm.

4.1. **Weak curves and Sony's problem.** Recall the domain parameters described above : (p, a, b, P, k, h), where p is the prime number of for the field $\mathbb{F}_p$, a and b are the coefficients in Weierstrass normal form, P is the point of origin, k is the order of the subgroup of G generated by P, and the group law.

Nigel P. Smart showed that when the curve's group order is equal to the order of the finite field, then the discrete logarithm can be computed in polynomial time. Such curves should thus be avoided. More about Smart's attack can be found here [29]. Another set of curves which was shown to be vulnerable are what are called super singular elliptic curves. For these curves, the elliptic curve discrete logarithm can be reduced to a regular discrete logarithm in $\mathbb{Z}_p$, for which there exists sub-exponential algorithms [18]. Super singular curves turn out to be useful in a different cryptographic scheme which is presented in section 5 of this paper.

Sony a few years ago, suffered an attack on its ECDSA protocol. The issue with Sony's procedure was that they were using a fixed number n instead of a random one in the ECDSA protocol. In this case, the following proposition holds:

Proposition 4.1. Let n the number chosen between 1 and k, the order of the subgroup, for the signature algorithm ECDSA. If n is fixed, then we can recover the private key n_A from Alice's signature.

Proof. Let (r_1, s_1) and (r_2, s_2) be two signatures Alice made on two different messages, using the same "random" number n. Since $r = x_Q$, where $Q = nP$, then since n is fixed, we have $r_1 = r_2$. We also know that $s_1 - s_2 = n^{-1}(z_1 - z_2)$ mod n, where z_1 and z_2 are the hashed truncated versions of the respective messages. Thus we obtain $n = (z_1 - z_2)(s_1 - s_2)^{-1}$ mod n. The private key can thus be recovered:

$$n_A \equiv {r_1}^{-1}(s_1 n - z_1) \ mod \ n \tag{4.1}$$

□

If n is not fixed, but somehow predictable, similar techniques can be used. This is what happened to Sony in 2011 [15]. Sony was using a vulnerable pseudo-random number generator for the seed n in the ECDSA signature scheme, which lead to a similar attack to what is described above.

Another issue which is raised earlier in this paper is that the NIST curves may be have been built with a backdoor, compromising their security. However, it is possible to check that a curve was generated randomly, which suggests that it would have been difficult to make it weak on purpose.

Proposition 4.2. Curve parameters generated by a hash function are verifiably random

Proof. Given a random seed S, it is possible to generate parameters a and b using a hash of s, $H(s)$. Then an arbitrary function can generate a and b from $H(s)$. The idea is that given the parameters and the seed, one can check that the parameters came form that seed. However, specifying a certain seed for desired parameters requires to inverse the hash function, which is computationally very difficult. □

This should give some degree of confidence that NIST did not arrange the curve parameters in a way that would make the curve vulnerable, but that these were randomly generated instead.

4.2. **Pollard's ρ attacks.** Let an instance of the discrete logarithm problem on elliptic curves be as follows. Given $Q = nP$, we would like to find n. Let k be the order of the subgroup generated by the point P. Pollard's ρ is the best known algorithm for breaking elliptic curve cryptosystems on classical computers. It runs in time roughly $O(\sqrt{k})$, which is exponential in the number of bits of k, the order of the subgroup of P. However, a rigorous analysis of Pollard's ρ complexity remains an open question [12]. Another method, the baby step, giant step algorithm achieves a similar bound for time complexity but happens to require a large amount of space, which makes it unpractical. Note that the running of Pollards' ρ is purely exponential, as opposed to the general number sieve which can also be used to solve the general discrete logarithm problem on the multiplicative group $\mathbb{Z}_p$, and which has sub-exponential running time.

The first step in Pollard's ρ is to find integers a, b, c, d such that $aP + bQ = cP + dQ$, where the pairs (a, b) and (c, d) are distinct. If we just try all possible pairs randomly we have an $O(k^2)$ running time. However, there is a more efficient method, which is at the heart of Pollard's ρ: Floyd's cycle finding algorithm.

Algorithm 3 Floyd's cycle finding algorithm

Input: P, Q
Output: $out = (a, b, c, d)$
1: Choose a random sequence S of (a, b) pairs
2: Use Pointers $p_1 = 0$ and $p_2 = 0$ to walk S.
3: **while** $aP + bQ \neq cP + dQ$ **do**
4: $\quad p_1 = p_1 + 1$
5: $\quad p_2 = p_2 + 2$
6: $\quad (a, b) = S[p_1]$
7: $\quad (c, f) = S[p_2]$
8: **end while**
9: **return** (a, b, c, d)

The correctness of the algorithm is immediate, as it only outputs when it found a pair. The complexity of the algorithm is the part that is hard to analyze in Pollard's ρ but a probabilistic proof can show a bound of roughly $O(\sqrt{k})$, as stated before. We can compute the group law operations using the double and add method for efficiency. Note also that applying addition and scalar multiplication keeps the resulting point in the cyclic subgroup. That is the subgroup is closed under the group law. As for the termination of the algorithm, since the order of the subgroup is finite and cyclic, it eventually finds a pair that works with 100% probability. Now that we have the (a, b) and (c, d) pairs, Pollard's ρ is a simple computation.

Algorithm 4 Pollard's ρ

Input: P, Q
Output: $out = n$
1: Use Floyd's cycle finding algorithm to find two pairs (a, b) and (c, d) such that $aP + bQ = cP + dQ$
2: Then $n = (a - c)(d - b)^{-}1 \mod k$, where k is the order of the subgroup of generated by P, which includes Q or any linear combination of the two
3: **return** n

A proof of correctness is given below.

Proof. Given two pairs (a, b) and (c, d) distinct such that $aP + bQ = cP + dQ$, we can compute n, such that $Q = nP$ as follows.

$$aP + bQ = cP + dQ \tag{4.2}$$
$$aP + bnP = cP + dnP \tag{4.3}$$
$$(a - c)P = (d - b)nP \tag{4.4}$$
$$(a - c)P \equiv (d - b)nP \mod k \tag{4.5}$$
$$(a - c) \equiv (d - b)n \mod k \tag{4.6}$$
$$n \equiv (a - c)(d - b)^{-1} \mod k \tag{4.7}$$

□

In practice, it takes months for Pollar's ρ to be able to break a standard elliptic curve protocol. There is reason to believe that elliptic curves will remain at the center of public key cryptography for many years before we really need to worry about quantum computers. The next section covers a brief overview of Shor's algorithm for the discrete logarithm problem, which can be applied to elliptic curves, and any abelian group in general, to solve the hidden subgroup problem. Should be be able to build quantum computers, major changes to the current public-key cryptosystems will have to be made. All thanks to Peter Shor, who in 1994, showed that a quantum computer could factor large numbers or take the discrete logarithm over a finite filed in polynomial time. The following section provides a brief overview of the intuition behind the algorithm.

4.3. **Shor's Algorithm for The Elliptic Curve Discrete Logarithm Problem.** Shor's algorithm involves heavy quantum computation, which is beyond the scope of this paper. However, we give intuition for the algorithm using the discrete logarithm on finite prime fields. For a more formal description of the algorithm, see [28].

Quantum computers have an interesting ability. Instead of computing over bits, which are either 0 and 1, they compute on what are called *qubits* and represent a probability to be either a 0 or a 1. We say that the quibit is in a *superposition* of the 0 and 1 state. This fact is important. It is a common misconception that quantum computers can perform their computation in parallel, which is a bad representation of what is actually happening. A quantum computer works on probability distributions, which in a sense allows it to perform operations which can impact many states at a time by changing a quibit's probability distribution, but the resulting state is limited. While we may be able to produce a computation over superposition of states, when measuring the state of a quibit, we get a single random value . We say that the wave function collapses when measuring the quibit forcing it into a definite state (i.e not a superposition).

Suppose we have $g^r \equiv x \mod p$. We wish to find r. Let $f : \mathbb{Z}_p \times \mathbb{Z}_p \to \mathbb{Z}_p$ be the following function:

$$f(a, b) = g^a x^{-b} \mod p \tag{4.8}$$

Shor's algorithm uses a particular procedure on quantum computers, called the quantum Fourier transform, QFT. Here quantum computing offers an exponential speedup in comparison to the classical FFT, which is really the key to the polynomial running time of Shor's algorithm. We don't present the QFT in this paper but assume we have some black box which can perform the QFT. The QFT allows to find the periods of the function f over a and b. These periods have the interesting property that for a a pair (c, d), output of the QFT, we have the relation $d + cr = 0 \mod (p-1)$. Now, as long as we find c such that $gcd(c, p-1) = 1$, then we can recover r by taking $d + cr$ modulo $p - 1$. A similar trick can be used with elliptic curves where we use the additive group law and scalar multiplication instead of multiplication and exponentiation. The math behind the choice of function f is a little elaborate and is omitted for simplicity. For a detailed implementation of a polynomial time quantum algorithm for the elliptic curve discrete logarithm problem see [23].

5. Super singular elliptic curve isognenies and Post-Quantum Cryptography

Shor's algorithm poses a serous threat to current public key cryptographic system. While quantum computers remain mostly theoretical, large organizations such as the NSA have already made moves toward the use of quantum resistant cryptography, in prevention of these future possible attacks [2]. Sadly, the day that quantum computers can work with a practical number of qubits will mark the end of Elliptic Curve Cryptography as we know it. As was noted previously, the discrete logarithm on abelian groups can be broken in polynomial time using Shor's algorithm on a quantum computer. In order to create quantum resistant schemes, one can for instance look at non abelian groups, where Shor's algorithm does not apply. For this section of the survey, we focus on the development of a new encryption method, also based on elliptic curves, which has given hopes of being quantum resistant. This method is based on isogenies of super singular elliptic curves and was suggested by De Feo and al. in 2014 [8]. Because of its recent discovery, this encryption scheme will require more attention and research from the cryptography community in order to affirm its security. As a side note, the first Diffie-Hellman-Merkle key exchange equivalent on elliptic curve isogenies was first suggest in 2006 [25]. However, the protocol used ordinary curve, for which a subexponential quantum algorithm was found in 2014, which motivated the use of super-singular curves [7]. We define the necessary terms below and look at the Diffie-Hellman-Merkle key exchange equivalent for supersingular curve isogenies.

5.1. Basics.

The key exchange uses super singular curves over $\mathbb{F}_{p^2}$ for some prime p. Supersingular cuvres can be defined in many ways.

Definition 5.1. For this particular field, we define a *supersingular curve* as having no points order p, and in particular the curve as $p \pm 1$ points. These types of curve have unusually large endomorphism rings.

Definition 5.2. An *isogeny* is a rational map from a curve E to a another curve E' such that $|E| = |E'|$. That is the number of points on the two curves is the same.

Definition 5.3. The j-invarient of a curve E, namely $j(E)$, is a function which is fixed over sets of isomophic curves and which can be computed from a given curve's parameters. In particular, for a curve with equation $y^2 = x^3 + ax + b$, the j-invarient is given by :

$$j(E) = 1728\frac{4a^3}{4a^3 + 27b^2}$$

The following protocol is shown for intuition and global understanding of the encryption scheme. For a formal proof of correctness and complexity, see [8]

5.2. **Key exchange protocol.** Before starting the protocol we need the publicly available domain parameters:

- A prime p of the form $p = a^e b^d \pm 1$, were a and b are small primes and b, d are arbitrary elements in $\mathbb{F}_{p^2}$.
- A super singular curve E over $\mathbb{F}_{p^2}$
- Points P_A, Q_A, P_B, Q_B on the curve, such taht the order of P_A, Q_A is a^e and the order of P_B, Q_B is b^d

We now detail the protocol:

(1) Alice and Bob each generate two random integers$m_a, n_a < a^e$ and $m_b, n_b < b^d$.
(2) Alice and Bob use their randomly generated integers to computer a linear combination of $P_A Q_A$, and P_B, Q_B. We get $R_A = m_a P_A + n_a Q_A$
(3) Alice uses R_A and Bob R_B to create isogenies ϕ_A and ϕ_B
(4) Using their respective isogenies, Alice and Bob create points $\phi_A(P_A), \phi_A(Q_A)$ and $\phi_B(P_B), \phi_B(Q_B)$ on the curves E_A and E_B, images of E by their respective isogenies.
(5) Alice sends $E_A, \phi_A(P_A), \phi_A(Q_A)$ to Bob, and Bob does the same with $E_B, \phi_B(P_B), \phi_B(Q_B)$
(6) Now Alice and Bob can compute S_A and S_B such that:

$$S_A = m_a\phi_B(P_B) + n_a\phi_B(Q_B)$$
$$S_B = m_b\phi_A(P_A) + n_b\phi_A(Q_A)$$

(7) Using S_A and S_B, Alice and Bob each create a new isogeny ψ_A and ψ_B. These isogenies take E to the isogenous curves $E_A B$ and $E_B A$, respectively.
(8) The j-invarient of the curves $E_A B$ and $E_B A$ is the same, and forms the shared secret k between Alice and Bob. The actual key is a function of k.

Security and Quantum. The reasons for which the j-invarient is the same for two curves involves an elaborate background, which is also beyond the scope of this paper. However, one can see how the Diffie-Hellman-Merkle protocol is easily applicable to other mathematical structures. Elliptic curves in particular offer both a classical encryption using ordinary curves and a potentially quantum resistant scheme using super singular curves.

The main reason that the scheme is believed to be quantum resistant is that it is based on a non-abelian group: the set of isogenies of for a given elliptic curve and the operation of composition. This is fundamentally different from the other protocols discussed in this paper, which are based on abelian groups, and thus, are candidate to Shor's attack. It is currently an open question in quantum computing of whether the same kind of polynomial time algorithm exist for non-abelian groups.

For its small keys, the isogeny based protocol has strong arguments to replace ECC. More research will be however necessary before their security has been supported to the point of practical use. There are also many other classical public key methods which are also based on non-abelian groups and are believed to be quantum resistant. To name a few: lattice based cryptography is the one which has been the most researched to this day [19], multivariate equations, error codes, and hash based cryptography are also potential candidates. For a comprehensive overview of these different methods see [6, 21].

6. Conclusion

Elliptic curve cryptography, since the beginning of its wide adoption in the early 2000's, has brought considerable improvements to its predecessors by dramatically reducing the key size required for the same amount of security. ECC, however, like other cryptographic methods based on abelian group arithmetic, is threatened by the rise of quantum computing and will most likely be obsolete once these computers become a reality. This should however not discourage elliptic curve enthusiasts. First, elliptic curve will be around for a long time before quantum computers are able to break the protocol. Second, elliptic curves are fascinating mathematical structures which hold many interesting properties. As we discuss in the second part of this paper, another possible use of elliptic curves resides in isogeny based cryptography, which may prove to be quantum resistant and thus a natural successor to ECC.

References

1. *Nist, digital signature standard, fips publication 186-3*, (2009).
2. *National security agency, cryptography today*, (August 2015).
3. Scott Aaronson, *The limits of quantum computers*, Scientific American **298** (2008), no. 3, 62–69.
4. Leonard Adleman, *A subexponential algorithm for the discrete logarithm problem with applications to cryptography*, Proceedings of the 20th Annual Symposium on Foundations of Computer Science (Washington, DC, USA), SFCS '79, IEEE Computer Society, 1979, pp. 55–60.
5. Ramachandran Balasubramanian and Neal Koblitz, *The improbability that an elliptic curve has subexponential discrete log problem under the menezesokamotovanstone algorithm*, Journal of cryptology **11** (1998), no. 2, 141–145.
6. Daniel J Bernstein, Johannes Buchmann, and Erik Dahmen, *Post-quantum cryptography*, Springer Science & Business Media, 2009.
7. Andrew Childs, David Jao, and Vladimir Soukharev, *Constructing elliptic curve isogenies in quantum subexponential time*, Journal of Mathematical Cryptology **8** (2014), no. 1, 1–29.
8. Luca De Feo, David Jao, and Jérôme Plût, *Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies*, Journal of Mathematical Cryptology **8** (2014), no. 3, 209–247.
9. W. Diffie and M. Hellman, *New directions in cryptography*, IEEE Trans. Inf. Theor. **22** (2006), no. 6, 644–654.
10. Raouf Dridi and Hedayat Alghassi, *Prime factorization using quantum annealing and computational algebraic geometry*, arXiv preprint arXiv:1604.05796 (2016).
11. Mark Ettinger, Peter Høyer, and Emanuel Knill, *The quantum query complexity of the hidden subgroup problem is polynomial*, Information Processing Letters **91** (2004), no. 1, 43–48.
12. Steven D Galbraith, *Mathematics of public key cryptography*, Cambridge University Press, 2012.

13. Nils Gura, Arun Patel, Arvinderpal Wander, Hans Eberle, and Sheueling Chang Shantz, *Comparing elliptic curve cryptography and rsa on 8-bit cpus*, Cryptographic hardware and embedded systems-CHES 2004, Springer, 2004, pp. 119–132.
14. Martin E Hellman, *An overview of public key cryptography*, IEEE Communications Magazine **40** (2002), no. 5, 42–49.
15. BBC News Jonathan Fildes, *iphone hacker publishes secret sony playstation 3 key*, (2011).
16. Neal Koblitz, *Elliptic curve cryptosystems*, Mathematics of computation **48** (1987), no. 177, 203–209.
17. Neal Koblitz and Alfred Menezes, *A riddle wrapped in an enigma*, Tech. report, IACR Cryptology ePrint Archive, Report 2015/1018, 2015.
18. Alfred J Menezes, Tatsuaki Okamoto, and Scott A Vanstone, *Reducing elliptic curve logarithms to logarithms in a finite field*, Information Theory, IEEE Transactions on **39** (1993), no. 5, 1639–1646.
19. Daniele Micciancio and Oded Regev, *Lattice-based cryptography*, Post-quantum cryptography, Springer, 2009, pp. 147–191.
20. Victor S Miller, *Use of elliptic curves in cryptography*, Advances in CryptologyCRYPTO85 Proceedings, Springer, 1985, pp. 417–426.
21. Ray A Perlner and David A Cooper, *Quantum resistant public key cryptography: a survey*, Proceedings of the 8th Symposium on Identity and Trust on the Internet, ACM, 2009, pp. 85–93.
22. Carl Pomerance, *Fast, rigorous factorization and discrete logarithm algorithms*, Discrete algorithms and complexity (1987), 119–143.
23. John Proos and Christof Zalka, *Shor's discrete logarithm quantum algorithm for elliptic curves*, arXiv preprint quant-ph/0301141 (2003).
24. R. L. Rivest, A. Shamir, and L. Adleman, *A method for obtaining digital signatures and public-key cryptosystems*, Commun. ACM **21** (1978), no. 2, 120–126.
25. Alexander Rostovtsev and Anton Stolbunov, *Public-key cryptosystem based on isogenies.*, IACR Cryptology ePrint Archive **2006** (2006), 145.
26. René Schoof, *Elliptic curves over finite fields and the computation of square roots mod* , Mathematics of computation **44** (1985), no. 170, 483–494.
27. ———, *Counting points on elliptic curves over finite fields*, Journal de théorie des nombres de Bordeaux **7** (1995), no. 1, 219–254.
28. Peter W Shor, *Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer*, SIAM review **41** (1999), no. 2, 303–332.
29. Nigel P Smart, *The discrete logarithm problem on elliptic curves of trace one*, Journal of cryptology **12** (1999), no. 3, 193–196.

E-mail address: jw2016@mit.edu