Elliptical Curve Problem Solving with Hardware

Gregory Schumacher
University of Minnesota Morris
600 E. 4th St.
Morris, MN 56267
schum319@umn.edu

ABSTRACT

This paper discusses the feasibility of breaking elliptic curve cryptography (ECC) using specialized hardware. The elliptic curves are mathematical groups of points that are used in ECC. Although data security with ECC is an important subject in computer science, not many papers look into having hardware breaking ECC. This paper examined two proposed hardware designs for solving ECC and compares the costs of the two systems, as well as their speed.

Keywords

discrete logarithm problem, elliptic curves, collision points, distinguished points,finite fields, modular arithmetic, Pollard rho method, elliptic curve cryptography, elliptic curve discrete logarithm problem

Categories and Subject Descriptors

[**Data security, Hardware**]

General Terms

cryptography, group, finite field

1. INTRODUCTION

In the cybernetic world, messages sent from computer A to computer B are encrypted so that the message is not intercepted and read by other people. This science of encrypting messages is called cryptography. This word comes from the Greek words for "hidden" and "writing". Encryption systems are easy to encrypt and hard for third parties to decrypt (undo encryption). Third parties that want to read these messages will need to solve a very hard mathematical problem that would take years or perhaps decades to solve, such as the discrete logarithm problem(DLP). Elliptic Curve Cryptosystems (ECC)[2] are based on a mathematical problem called the elliptic curve discrete logarithum problem (ECDLP) with the same mathematical concepts as DLP. ECC leaves only generic and long tedious ways of solving it under desired conditions. The concept of hardware built to solve ECC is a relatively new topic in the field of data security. If a system can be built to break ECC in a reasonable time, then messages encrypted in this system will not be secure from third parties.

One method to solve the mathematical problem that ECC is based on and break the encryption is the Pollard rho method. Creating a hardware system that would effectively use the Pollard rho method would be a useful and powerful tool for breaking ECC encryption. Hardware like this would have an impact on the security of text encrypted by ECDLP and would drastically decrease the amount of time it takes to decode text that would otherwise be secure. This paper describes two developing hardware systems created to solve ECC. Hardware may play a more critical role in solving these problems if the cost of the hardware is low enough relative to software implementations. The cost of the hardware will affect its feasibility in the real world. In examining the two proposed hardware systems we look at the time it takes for the hardware to decrypt messages and the cost of the hardware used.

Our discussion will briefly explain the mathematical foundation on which DLP and ECC are formed.The complexity of DLP can be given and explained how it is useful in encryption. Understanding DLP will make ECC easier to understand. An example of an ECC encryption will be given. After ECC is explained, the Pollard rho method will be introduced, it will be shown how it solves DLP. Then it will be explained how the Pollard rho method can be used to break ECC. The hardware implementations created to break ECC from [1] and [2] will then be discussed. Results of these hardware systems will then be examined to show how well ECC stands against them.

In order to understand the hardware and the results it is important understand the mathematical ideas in the encryption processes presented in sections 2-5.

UMM CSci Senior Seminar Conference,04/28/12 Morris, MN.

2. MATHEMATICAL BASIS

The difficulty of decrypting ECC can be explained by looking at a similar problem of solving DLP. The DLP mentioned previously in the paper is a mathematical oddity that has useful properties for encryption. Because of its detailed study in mathematics, we can be reasonably sure that encryption methods similar to DLP will be difficult to solve.

2.1 Modular Arithmetic

The encryption algorithms discussed in this paper use fi-

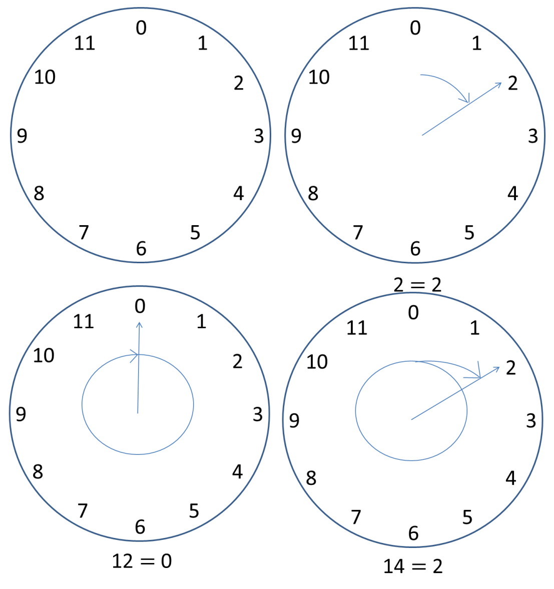

Figure 1: Clock $\mathbb{Z}_{12}$

nite fields in their scheme.

DEFINITION 1. *An abelian group is a set of elements G with a binary operations * is a set of elements that satisfy the following properties.*

- $\forall a, b \in G, a * b \in G$
- $\exists\ I \in G$ *(an identity value) such that* $a * I = a = I * a$
- $\forall a \in G\ \exists$ *an inverse* $a^{-1} \in G$ *such that* $a * a^{-1} = I = a^{-1} * a$
- $\forall a, b, c \in G, a * (b * c) = (a * b) * c$
- $\forall a, b \in G, a * b = b * a$

DEFINITION 2. *A finite field is a set of elements G with two binary operations +,* that form an abelian group under + and for non zero elements under * such that*

$$\forall a, b, c \in G, a * (c + b) = a * c + c * b$$

DEFINITION 3. *Modular arithmetic is arithmetic on integers in which the result of every operation is taken modulo a fixed positive integer. This shall be written as as* $j \equiv x(mod\ y)$.

DEFINITION 4. $\mathbb{Z}_k$ *is the set of integers under operation* $(mod\ k)$ *which are 0 to* $k - 1$.

DEFINITION 5. *The additive inverse of an integer a in* $a \in \mathbb{Z}_k$ *is b where* $a + b \equiv 0(mod\ k)$.

In $\mathbb{Z}_7$ we have additive inverses $[(1, 6), (2, 5)(3, 4)]$

A good example of modular arithmetic $\mathbb{Z}_{12}$ would be a clock.

Let $\mathbb{Z}_k$ represent a standard clock with the 12 replaced with a zero (this is the abelian group of integers form 0 to 11 set under addition operation). Let 5 hours pass from hour 0. Then we are at 5.

$$5 \equiv 5(mod\ 12)$$

If we are 15 hours past 0 then we loop around the entire clock and end up at $15 \equiv 3(mod\ 12)$. For each element $a \in \mathbb{Z}_k$ there exists another element b such that $(a + b) \equiv 0(mod\ k)$ were $b = k - a$. The element b is an additive inverse of element a. As an example in $\mathbb{Z}_{12}$ (1,11),(2,10),(3,9),(8,4),(7,5),(6,6) are pairs of elements and their additive inverses. A multiplicative inverse for element $a \in \mathbb{Z}_{12}$ would be element c that when multiplied with a create 1 in $\mathbb{Z}_k$.

$$[a \times c] \equiv 1(mod\ k)$$

However not all elements a in $\mathbb{Z}_{12}$ have multiplicative inverses c (an example would be 2). So $\mathbb{Z}_{12}$ is not a finite field. An example of a finite field would be set $\mathbb{Z}_7$ that has multiplicative inverses for all $y \in \mathbb{Z}_7$. If we use a prime number p then all (non zero) x in set $\mathbb{Z}_p$ have a multiplicative inverse. This is important, because later when messages are encrypted the inverse of the value used to encrypt will be used to decrypt the messages.

DEFINITION 6. *The order of a finite field is the number of elements in that field, denoted as* $|F|$ *for field F.*

All finite fields have an order. In the example above $\mathbb{Z}_7$ has an order of 7. In $\mathbb{Z}_k$ k being a positive integer the order is k.

DEFINITION 7. *The order of element* $g \in$ *group G under operation * (denoted as* $ord(g)$*) is the smallest positive integer a such that* $g * g * g .. * g$ *(a times) equal to multiplicative identity* I.

An example of this would be the multiplicative order of $4 \in \mathbb{Z}_7$. We compute $4^2 = 16 \equiv 2(mod\ 7)$, $4^3 = 64 \equiv 1(mod\ 7)$, so $ord(4) = 3$.

DEFINITION 8. $g \in$ *group G such that* $ord(g) \geq ord(f)$ *for all* $f \in G$*, because we will want the order as large as possible for encryption.*

This ideally means that when g is raised to the power of integers 1 to p-1 we get every (non zero) element in field $\mathbb{Z}_p$. An example of this would be 2 in $\mathbb{Z}_5$ $2 \equiv 2(mod\ 5)$, $2 \times 2 \equiv 4(mod\ 5)$, $2 \times 3 \equiv 1(mod\ 5)$, $2 \times 4 \equiv 3(mod\ 5)$. If we choose a different element , such as 0 in $\mathbb{Z}_k$, we would have only one possible outcome 0. So 0 is a poor choice and can lead to security problems.

3. DISCRETE LOGARITHM AND ECC

The DLP is a complicated problem. This is important, because encryption methods that use similar principals like ECC cannot be decrypted quickly and reliably.

3.1 Discrete Logarithm Problem

DEFINITION 9. *The discrete logarithm problem is solving for a in equation* $g^a = x(mod\ p)$ *were g and x are in finite field* $\mathbb{Z}_p$.

If we try to solve a in the set of all real numbers, then we could approximate for a, and work our way to the solution. An example is $5^a = 12$. We can say $a = log_5 12$. This would give us an approximation for a. This will not work in the finite field, because we need an exact answer. If one needs to find a for $3^a \equiv 4(mod\ 31)$ there is no quick and easy solution to find a which is $a = 18$.

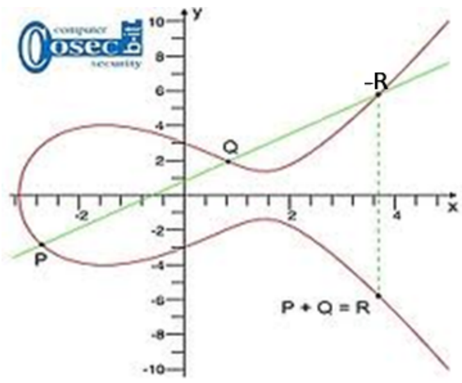

Figure 2: Point addition image [4]

3.2 Elliptical Curves

Definition 10. *An elliptical curve is the set of points that satisfy cubic equation of the form $E(x) = y^2 = x^3 + xq + e$, such that all roots of said equation are distinct [5].*

We can describe the addition of two points P,Q as the point P+Q=R geometrically: by drawing a line through points P,Q and finding a third point where this line intersects with the elliptical curve. This point is $-R$. If we reflect $-R$ over the x-axis we get R. [6] The visual representation of this addition is given in Figure 1.

The points of the elliptical curve form an abelian group under addition, however we want to represent the coordinates of points as integers or binary, so software or hardware can represent them accurately.

3.3 Elliptical Curve $E(F_P)$

To get an elliptical curve of integer points an elliptical curve is generated over field F_p. This curve $E(F_p)$ is the set of points that satisfy the equation $y^2 = x^3 + xq + e(mod\, p)$. Addition in $E(F_p)$ is similar to addition in $E(x)$, but the (X,Y) coordinates of the resulting point from addition are placed in function $x(mod\, p)$. This forms an abelian group because F_p is a field. [5] Examples of this will be presented in section 4.1.

3.4 Elliptical Curve $E(F_{2^m})$

Another type of the elliptic curve is over F_{2^m}, where m is an integer. This elliptical curve is similar to the curve under F_p, but the coordinates of the points are represented as polynomials under $\mathbb{Z}_2$. This is ideal for representing the coordinates as binary and easy for hardware to interpret. Addition in $E(F_{2^m})$ is similar to addition in $E(F_p)$ and forms an abelian group [5]. The elliptical curves create a large set of points in large fields, which makes them useful for encryption.

3.5 Elliptical Curve Discrete Logarithm Problem

The elliptical curve discrete problem (ECDLP) is a problem similar to DLP. The problem of solving ECDLP is defined as follows: given two points G,Q in $E(F_p)$ or $E(F_{2^m})$, find the integer l such that

$$l * G = Q$$

$l * G$ can be described as l additions of G on itself $(G + G + G.. + G)$ [1]. The elliptical curve problem is a complicated one that cannot be solved easily when elliptical curves are in large fields, because of the large set of points that can be generated by the elliptical curves.

4. ELLIPTIC CURVE ENCRYPTION

Since the elliptical curve problem is a complicated mathematical problem, it is useful for encryption. Suppose Alice wants to send message "m" to Bob. ECC uses public key cryptography with elliptic curve group operation.

Definition 11. *Public Key cryptography has the recipient with a set of matching keys (public and private). The recipient has given out public information (public key) to any sender of messages that is used to encrypt that message. After the message is encrypted the sender will send the message to the recipient. Once the recipient gets the encrypted message they use private information that only they have (private key) to decrypt the message. Suppose Alice wants to send a message "m" to Bob without it being read by anyone. These are the steps that Alice would take.*

Encrypted message exchange works as follows:

1. Create a curve, $E(x)$, over a finite feild, F_p. Choose a point G on curve $E(x)$. With point G generate a group $[G, 2G, .., nG = I]$ (I is the identity point) of order n. The private key integer l will be picked randomly from $[1, 2, .., n-1]$ and used to create public key lG. Message "m" will be mapped to a point M on the elliptic curve.
2. Alice creates a random integer k in $[1, 2, .., n - 1]$ and computes kG
3. Alice looks up Bob's public key lG and computes klG, $M + klG$
4. Alice sends Bob the pair of elements $(kG,\ M + klG)$
5. Knowing l Bob computes $(M + klG - lkG) = M$ and gets the message point M, which is mapped back to message "m".

The problem exists in finding l from G and lG. This is similar to trying solve for a in DLP $g^a = x(mod\, p)$. There are however methods that can through tedious work solve the problem. The most popular and reliable of these methods is the Pollard rho method.

Now here is an example of ECC encryption to show how it is used. The elliptical curve chosen for this encryption is:

$$E : y^2 = x^3 + x + 1(mod\, 23)$$

Our generating point G = (3,10), the private key l = 5, the public key lG = G+G+G+G+PG = (9,16). Points P,lG along with the elliptic curve equation $y^2 = x^3+x+1(mod\, 23)$ will be the public key. If Alice wants to send message "m" to Bob then Alice will need to do the following:

1. Turn message "m" into a point(s) $M = (7, 12)$ on the elliptical curve
2. Generate a random number k (which for this case is 3)
3. Create points $klG = (1, 16)$ and $kG = (19, 5)$
4. Send points $(kG, M + klG = (18, 3))$

The points are now set, so Bob has the encrypted message point(s) M. Now Bob will decrypt the message in the following steps.

1. Take point kG and use private key l to generate point $lkG = (1, 16) = (klG)$
2. Generate $-lkG = (1, -16) = (1, 7)$
3. Use cancellation to get point $M = M + klG + -lkG$
4. Now Bob can turn point(s) M into message "m"

5. POLLARD RHO METHOD

If there is a room full of people what are the odds that two people have the same birthday? If there are 367 people there is a $\frac{100}{100}$ chance that there will be a collision of two birthdays, but you only need 57 people to have a $\frac{99}{100}$ chance of a birthday collision and only 23 people for a $\frac{50}{100}$ chance of a birthday collision. This is known as the birthday paradox, because we need a small number of people to get a large probability of a a collision of two birthdays [1]. Pollard rho method uses a method to solve the following problems with collision of random points. We want to use these collisions to solve DLP or find the private key in ECC by solving ECDLP.

5.1 Pollard rho DLP

Consider a DLP problem $g^a = x(mod\ p)$ where we want to solve for a. This can be done by finding two points that form the following collision where $c, b, u, v \in F_p$:

$$g^c x^b = g^u x^v$$

$$a \equiv \frac{(c-u)}{(b-v)}(mod\ ord(g))$$

We have $(mod\ ord(g))$ in the equation above, because we want the smallest integer that a is equivalent to. The Pollard rho method is a collision based method to solve DLP. The end goal of this algorithm is to create two equal points that are generated differently. These points are called collision points if they are not generated with the same values of c, b, u, v. The collision points are found by first having a starting point that goes on a random leap (a random increment increase from previous value) from one point to the next point generated until a collision is found. We then create a set of jumping sizes were X is a function of integer i that increments b and c. So $X_i = g^{a_i} x^{b_i}$ We will create a large set of S of size k so we have more points for a collision. Finally we create a separate set of points. The algorithm is summarized below.

1. Generate point $X_k = g^{a_k} x^{b_k}$.
2. Look for a point X_i in S such that $X_i = X_k$.
3. If no such X_i exists go back to step 1.
4. If there exists a X_i such that $X_k = X_i$ check to make sure $b_i \neq b_k$. If $b_i = b_k$ go back to step 1, else go to step 5.
5. Calculate a with equation $a \equiv \frac{(c-u)}{(b-v)}(mod\ ord(g))$

The Pollard rho method takes on average $\sqrt{\pi ord(g)/2}$ steps to solve [3]. Now we discuss how a modified Pollard rho method can be used to find the private key in ECC by solving ECDLP.

5.2 Pollard rho elliptical curve

The process of solving the elliptical curve problem is similar to DLP. Just like with DLP, it is important to have a starting point and generate other random points from walks to find two points that equal each other to find integer l such that $l * G = Q$ (l is the private key in ECC).

Now we can modify the Phollard rho method to solve ECDLP. Create a random point $R_0 = q_0G + e_0Q$, where q_0, e_0 are numbers between 1 and $ord(G)$. Then create a random point $H_k = q_kG + e_kQ$ with q_k, e_k randomly chosen. To generate the next point we set $R_{i+1} = R_i + H_k$. When we find two points $R_i = R_j$ calculate l by $R_i = q_iG + e_iQ = q_jG + e_jQ = R_j$. So if we replace e_jQ with e_jlG then $q_iG + e_ilG = q_jG + e_jlG$, so $l \equiv (q_j - q_i)/(e_i - e_j)(mod\ ord(G))$.

DEFINITION 12. *A distinguished point (DP) is a point that satisfies a specific condition. In this case the condition is the x-coordinate of the point is made of bits consisting of a specific number of consecutive zeros. The number of consecutive zeros is set based on the size of the field the elliptical curve is on.*

The advantage of searching for distinguished points is the odds of collision between two distinguished points is greater than that of regular points (depending on the number of consecutive zeros).

6. HARDWARE USED TO SOLVE ECDLP

In this section we will review two proposed hardware systems presented in [1] and [2]. These hardware systems work over two different elliptical curves.The hardware used in [1] will solve ECDLP for elliptical curves in $E(F_p)$ and the hardware in [1] will solve ECDLP in $E(F_{2^m})$.

6.1 Hardware used in prime Elliptical Curve

The first hardware tries to solve for groups smaller or equal to 128 bits with COPACOBANA which is a parallel computer of 120 field programmable gate arrays (FPGAs). They are integrated circuits that process in parallel. The hardware used is Xilinx Spartan-3 XC3S100. "Due to the relatively low cost at low quantities and the reconfigurability, the choice of FPGAs seems optimal"[1]. Some properties of this hardware include low cost, high performace logic solution for high volume, and 1,872 Kbits of total RAM.

6.2 Normal Elliptical Curve Hardware Implementation

In this hardware system we will have W point processors (W is an arbitrary number) to generate DP to find a collision.

There is a central server to store the possible collision points from the processors. The points need to be stored so we can eventually find two points that match. The server will also have a communication controller for data exchange with W point processors.

There will also be a database for storing values (c,d,R) for point R = cP +dQ in a table. So if two points are equal we want to make sure they were not generated the same way or the Pollard rho method will not work.

The servers also needs a unit for validating distinguished points from a processor. This is necessary for defective processors which could undermine the result. Once we find two points that collide and are not generated the same way, we then use an arithmetic unit for computing Pollard rho method from detected collision points. Finally the server will need an elliptic curve generator for testing hardware. Finding a collision will allow us to compute the private key

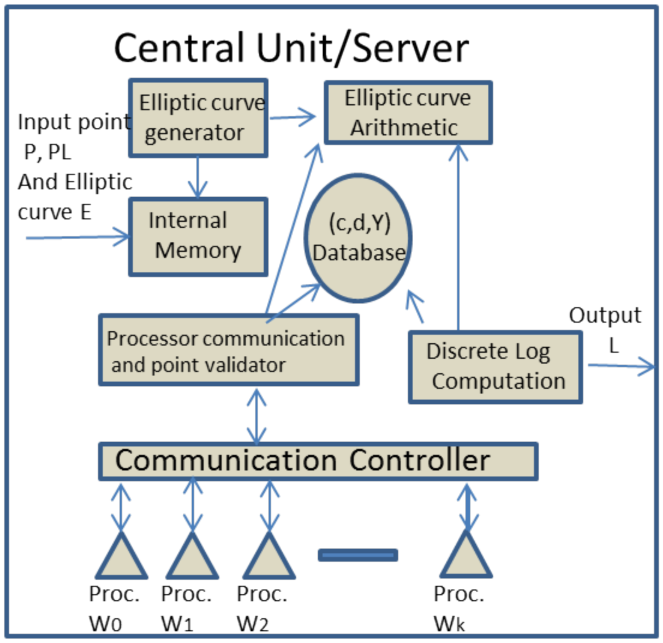

Figure 3: Hardware system used in hardware 1 [1]

used in ECC. The central server will have more modest computational requirements depending on the size of the subset used and the number of point processors. The central server will only process a collision point occasionally.

The point processors will compute the trial of each point in four steps.

The first step is distinguished point (DP) detection. We need to find if the current point R is a possible collision point. The hardware uses a DP property which is satisfied when the z most significant bits of point R = (x,y) are zero. If such a point is found, it is transmitted to the central sever. The advantage of this is immediate control of the number of point transfers between database server and point processors in hardware. If we choose a large z value it will cause a longer computation trials and less points to give central server. If z is too small we will send a large number of points, but they will be less likely to be a collision point and may waste the central servers time. If a collision is not found we will continue by partitioning (creating a new number with the sum of two other numbers). In order to select the next random point H_k we need to update the coefficients. This is done by creating random numbers c_r, d_r which are used to create partitions that will be added to the old coefficients to create coefficients $c_k = c_{k-1} + c_r, d_k = d_{k-1} + d_r$. $q_r, e_r = 1, 2...s - 1$ where s is an arbitrary number to limit the size of the partitions.

Now with our partitions we will need to update the corresponding coefficients c and d for point X with X $= cP + dQ$ by adding the random coefficients $c_i \equiv c + q_i(mod\ p)$ and $d_i \equiv d + e_i(mod\ (ord(P)))$ according to the selected partitions.

Finally we need to do point addition operation. In order to update the current point by calculating H_k we need to do some point addition $H_k = c_kP + d_kQ$.

6.3 Hardware used in 2^m Curve

The second elliptic curve uses Spartan 3E FPGAs. They make calculations for S3E1600-5, S3E1200-4 and S3E1600-5. This hardware has similar properties to the hardware listed above: low cost, and high-performance logic solution for high volume.

6.4 2^m Elliptical Curve Hardware Implementation

The second hardware is made of several subsystems, each designated a role in finding the private key. The connected point processors are designed to run on special-purpose hardware. The central server will have more modest computational requirements depending on the size of the subset used and the number of point processor. The central server will only process a collision point occasionally.

For the second hardware system we will have an arbitrary number of client systems. First the client hardware will generate the points with a starting point and will generate the new points with partitions until a collision is found. All the hardware clients will embed a First in First Out (FIFO) communication interface, a main controller, and some elliptic curve processors with a communication buffers for EC-uP (Elliptic curve update point) point generator, so client hardware generates appropriate points. We will also have a software server which will handle the less intensive job of dispatching the starting points for each hardware client, collecting DP (distinguished points), and checking for collisions. Each data point sent through communication will have a header. The header will be created from the x-y coordinates and the coefficients c_i,d_i components in point $R_i = c_iP + d_iQ$. The header will determine the address of EC-uP; depending if point is DP or SP (starting point).

In the initial phase, the data is sent from main controller to the first EC-uP, which collects one SP for every walking chain. When the length of the chain extends $(20 \times \theta)$ where θ is an arbitrary large number, use a new SP to start a new chain. SPs are fed to FIFOs to load point updates and coefficient updates (in most architectures coefficient update work on k-bit data to avoid a specific serialize circuit). When DP is found, the point and its coefficients are stored in other FIFOs to serialize and output data as soon as possible. EC-uP will compute point additions, update coefficients and track the chain length until the problem is solved.

7. RESULTS

From papers [1] and [2] the performance of these hardware systems can be estimated. Although the systems measure their results in different ways, the data collected at the end tells that hardware decrypts ECC by finding the private key where the key is of a relatively small bit size. However, time estimates show that ECC decryption would be impractical for a significantly large key size.

The estimated time for solving the elliptical curve problem was calculated for 3 different hardware systems. A Pentium M hardware as a more common hardware will serve as a control, XC3S1000, and ASIC. ASIC is believed to solve the elliptical curve problem much faster for larger k bit size for the private key. Pentium M hardware acts as a control to better compare the effectiveness of ASIC against other hardware.

The time it takes to find the private key of bit size 64 for the hardware in system 1 is just over 78 minutes. However the system drastically slows down for k sized of over 80.

k	Pentium M	XCS1000	ASIC
64	5.14 h	78.1 min	----
80	70.5 d	21.5 d	----
96	55.1 y	20.4 y	----
128	$4.86 * 10^6$ y	$4.86 * 10^6$ y	$2.05 * 10^6$ y
160	$3.78 * 10^{11}$ y	$3.10 * 10^{11}$ y	$2.48 * 10^9$ y

Figure 4: Time to solve ECC by private key size k [1]

k	C(k)	Total Pts / sec.
160	855	93,000
128	695	173,00
96	535	270,00
96	535	331,00
80	455	447,00
64	375	585,000
64	375	693,000

Figure 5: Complexity vs. Point generation [1]

This means that this hardware cannot solve ECDLP within a reasonable amount of time. Most ECC will have a k size of 128, so this hardware is not capable of breaking mainstream ECC.

In [1] the complexity of generating a point with private key of bit size k was calculated using the equation,

$$C(k) = 5k + 55$$

The complexity C(k), total number of points the hardware can generate per second (Total Pts/s) are given in Figure 5.

As you can see from the table the complexity of generating points increases with the size of k.

The estimated cost of this system with a market cost of XC3s1000 is 50 dollars a chip. The total cost of a system with 120 Xc3S1000 chips is approximately 10,000 dollars.

In paper [2] with FPGA hardware XC3S1200E-4FT256 at 21 dollars or XC3S1600E-5FG320 at 33 dollars and electricity at 0.1 dollars per KWh. The paper focused on the cost of attacking ECC F_{2^m}. It was found that ECC of F_{2^m} $m = 113$ and F_{2^m} $m = 131$ was not secure against attacks from the hardware. ECC in F_{2^m} $m = 163$ which is the standard security was beyond the hardware to decrypt. The estimated cost of the hardware is given below in Figure 6. The variable number of Copa for 1 year is the estimated number of hardware systems needed to break an encryption in 1 year, costs are is US dollars.

It is estimated to take 6 months to solve the elliptical curve problem on an eliptical curve in F_{2^m} for m=113. The estimated cost for a 150 dollar computer in electricity is 35 times greater than the hardware. For a computer with no dedicated F_{2^m} arithmetic logic init ALU the cost can be 500 times greater. So the cost and run time of this system makes ECC decryption by a third party with this hardware more of a threat than a system that utilized software. The

m	#Copa for 1 year	Cost of electricity	Total cost
113	2	2.1×10^3	$22.1 $\times 10^3$
131	1728	1.8×10^6	$19.1 $\times 10^6$
163	125×10^6	131\times 10^9$	$1.4 $\times 10^{12}$

Figure 6: Complexity vs. Point generation [2]

hardware for ECC F_{2^m} was compared to the hardware for F_p in [1]. The authors of F_{2^m} thought that the hardware would be more effective against ECC F_{2^m}. They found the hardware [1] to hardware [2] speed of solving an elliptical curve of security level k = 160 in F_p was a ratio of near 50 to 1. We can see a clear case of the improvement of hardware to solve ECC from these two papers [1] and [2].

8. CONCLUSION

In conclusion, the process of building a hardware system to solve DLP and ECDLP is a complex one. The mathematical principles that protect encrypted messages still (at this papers current date) protect most encrypted messages of large fields. The research done in the papers above may encourage other researchers to invest hardware to break encryption. Another aspect of hardware that may make it attractive is the decrease of cost in hardware over time. Take for example the cost of a Xc3S1000 chip that in 2007 was 50 dollars. The rising computing power of specialized hardware makes it an attractive system for solving the elliptical curve problem in the future. This paper has only examined two hardware systems, but more powerful hardware systems may exist that have not been described in research literature.

9. REFERENCES

[1] T. Güneysu, C. Paar, and J. Pelzl. Special-purpose hardware for solving the elliptic curve discrete logarithm problem. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, 1(2):8, 2008.

[2] G. Meurice de Dormale, P. Bulens, and J. Quisquater. Collision search for elliptic curve discrete logarithm over gf (2 m) with fpga. *Cryptographic Hardware and Embedded Systems-CHES 2007*, pages 378–393, 2007.

[3] R. Montenegro and P. Tetali. How long does it take to catch a wild kangaroo? In *Proceedings of the 41st annual ACM symposium on Theory of computing*, pages 553–560. ACM, 2009.

[4] C. Scheytt. Resource-efficient hardware-software-combinations for elliptic curve cryptography. Point addition, 2007.

[5] J. Silverman and J. Tate. *Rational points on elliptic curves.* Springer, 1992.