Engineering and Applied Science Letters

PSR Press

Article

Rethinking the halting problem-angles trisectability cryptographic analogy

Kimuya M. Alex[1,*] and Munyambu C. June[2]

[1] Department of Physical Sciences (Physics), Meru University of Science and Technology (MUST), Kenya.
[2] Department of Education Science (Physics/Mathematics), Meru University of Science and Technology (MUST), Kenya.
[*] Correspondence: alexkimuya23@gmail.com

Received: 1 January 2022; Accepted: 21 May 2022; Published: 30 June 2022.

Abstract: The "angle trisection-halting problem" impossibility analogy is fundamentally based on the obscure perception that; the classical geometric notion of constructability in Euclidean plane geometry corresponds to the modern theory of computability. Specifically, the difficulty of empirical trisectability of any angle has been viewed as analogous to the impossibility of solving the halting problem. The primary goal of this paper is to establish the inherent incompatibility between the geometric trisectioning of angles and the halting problem. The exposed proof concern the genetic solutions methodic characterization of the inconsistencies between the angle trisection problem and the halting problem. We show that regarding their attempts at solutions, the genetic trisectability of an arbitrary angle leads to solving the halting problem in geometric cryptographic schemes. It is upon the characteristic inconsistencies that we establish a provable refute of the validity of considering the practical applications of geometric cryptography as a solid source for cryptographic principles.

Keywords: Geometric cryptography; Halting problem; Angle trisection; Computability; Classical geometry; Turing machine; Elliptic curve cryptography.

1. Introduction

Encryption is the technique of encoding messages (information) to prevent eavesdroppers or hackers from decrypting them but allowing authorized parties to do so. The message or information (commonly referred to as plaintext) is encrypted using an encryption technique, resulting in the unreadable ciphertext. Several cryptography techniques exist; including the geometric cryptographic schemes [1,2].

Based on the symmetric cryptography algorithm used to encrypt input data, this paper establishes the incompatibility between the geometric trisectioning of angles and the halting problem in recursion theory. We will in specific, focus on a form of a cryptographic technique called symmetric encryption. In this method, a similar key is utilized for both encryption and decryption procedures, and it is shared between the two communicating parties. It is a technique in which the information source encodes the plaintext and sends the ciphertext to the recipient using the key (or some set of restrictions). To decode the message and recover the plaintext, the receiver uses the same key (a set of predetermined rules). As a result, it is a type of cryptography in which both the information source and the recipient must know the key [3]. We will then employ the developed workflow for generalized geometric cryptographic schemes.

The "angles trisection-halting problem impossibility" analogy has been employed as a handy genetic example of symmetric encryption systems. The "angles trisection impossibility-halting problem impossibility" analogy is a geometric encryption method in which communications and ciphertexts are geometrically represented by angles or intervals of other geometric quantities (that may represent angles such as curves or straight-line segments). Computations are genetically based on straightedge and compass construction rules in this method [4]. The difficulty and impossibility assertions associated with solving some geometric problems, such as the trisection of an arbitrary angle using a straightedge and compass operations, serve as the foundation for the many geometric cryptography schemes [1,5]. Across the cryptographic historical frameworks, the first geometric cryptographic scheme was presented by [3], and the field has seen consistent growth in scientific research for decades [6]. Even though cryptography methods based on the plane Euclidean geometry system have practically limited physical applications (the geometric cryptographic schemes do not concern the classical straightedge and compass constructions), they are mostly utilized as pedagogic machinery for explaining more advanced cryptographic protocols [3,6].

The focus of this paper is to show that although the "angles trisectability-halting problem" impossibility analogy has frequently been utilized in the design of cryptographic techniques, its fundamental building blocks are geometrically misconstructed and it provides a weak framework for the application in cryptographic pedagogies and practices. We investigate the genetic characteristics of the two problems, the trisection of an arbitrary angle and the halting problem (based on the symmetric encryption method), and show the characteristic contrasts between the two problems in demonstrating their inherent incompatibility. Furthermore, we believe that for one to break the "angle trisectioning-halting problem" impossibility scheme; any such assertions should objectively be based on either, having sought the trisection of an arbitrary angle, or, by solving the halting problem or solving both of the problems. In this case, we use the "AG-Algorithm" [7] approach, which, when combined with the genetic definition of the two problems, invalidates the geometric cryptography analogy.

The rest of the paper's workflow is organized as follows: §2 deals with halting problem characterization, §3 concerns the angle trisection problem characterization, §4 deals with the characterization of the "angle trisection impossibility-halting problem" analogy and its limitations, §5 provides detailed discussions of the developed issues though the paper, and, §6 concludes the workflow.

2. Characterization of the Halting Problem

The halting problem is a decision problem in which you must determine if a computer program will eventually halt when run with the provided input given a computer program and an input to the program. The halting problem is a conceptual model in which there is no memory or time constraint on program execution, and so it can take an arbitrary amount of time and use a large amount of system storage space before halting.

In pseudo-code, for example, a program may run indefinitely in an infinite loop or may come to a halt relatively quickly. As a result, the halting problem has been determined to be algorithmically unsolvable [3,8]. This means that no method can be applied randomly to a program as input to determine if it will stop when run with a specific input. A generic technique to solve the halting problem for all potential program-input sets cannot exist, according to the Turing machine [3]. Turing machines are abstract computational devices that have been presented objectively to aid in examining the breadth and limitations of computational capabilities. Turing machines are the most powerful computational machines, and they form the theoretical basis for modern computers [8,9]. The next §2.1 presents a common understanding of the halting problem's operational principles.

2.1. A characteristic proof of the Halting impossibility

We set a straightforward interpretation of the problem as, given a random Turing machine M over the expression $\Sigma = \{x, y\}$, and a random string w over Σ, does M halt when it is given w as an input? The goal of this section is to characterize and show that the halting problem is not decidable, hence unsolvable.

Claim 1. The halting problem is undecidable and hence, unsolvable.

A Contradiction Assertion [10]: We establish an elementary proof by contradiction. We suppose that the halting problem is decidable, implying that there is a Turing machine operational named T_M that solves the halting problem. That is, given a description of a Turing machine M (expressible over Σ) and a string w, T_M writes "True" if M halts on w and "False" if M does not halt on w, and then T_M halts. Consider Figure 1;

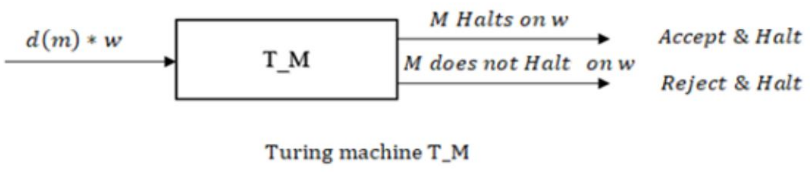

Figure 1. Basic Working Mechanism of Turing Machine

Let us now build the new Turing machine T_{M_a} shown in Figure 2. First, we create a Turing machine T_{M_m} by adjusting the T_M depicted in Figure 1 so that if T_M accepts a string and halts, then T_{M_m} goes into an infinite loop (T_{M_m} halts if the original T_M rejects a string and halts).

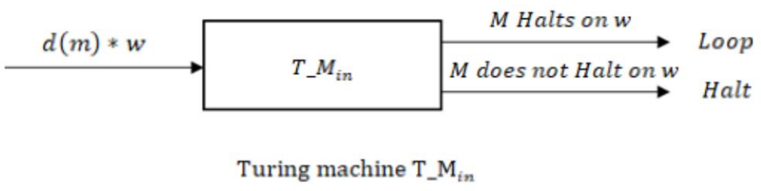

Figure 2. Basic Working Mechanism of Turing Machine

Then following (Figure 2 (T_{M_m})) we create another Turing machine T_{M_a} scheme shown in Figure 3 as: recall that a T_{M_a} takes as input a description of a Turing machine T_M expressed by $d(M)$, copies it to acquire the string $d(M) * d(M)$, where $*$ is a symbol that separates the two copies of $d(M)$ and then supplies $d(M) * d(M)$ to the Turing machine T_{M_m}.

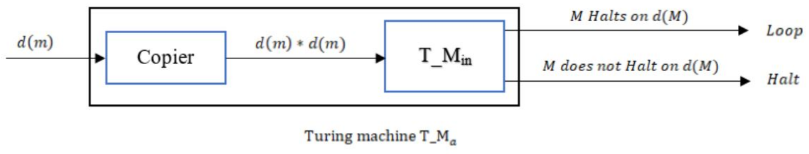

Figure 3. Finite State Machine as a Turing Machine

We examine how T_{M_m} operates when provided with a string describing T_{M_m} itself is given as input. Considerably, when T_{M_m} gets the input $d(T_{M_m})$, it makes a copy, constructs the string $d(T_{M_m}) * d(T_{M_m})$ and gives it to the revised T_M. Thus the modified T_M is described as the Turing machine T_{M_m} and the string $d(T_{M_m})$ as depicted in Figure 4.

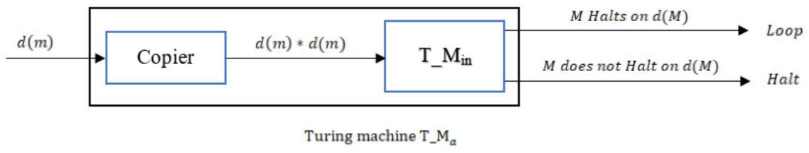

Figure 4. Genetic Characteristics of Turing Machine

From Figure 2, the modified T_M would execute an infinite loop if T_{M_a} halts on $d(T_{M_a})$, and on the opposite, it halts if T_{M_a} does not halt on $d(T_{M_a})$. In other terms, T_{M_a} goes into an infinite loop if T_{M_a} halts on $d(T_{M_a})$ and it halts if T_{M_a} does not halt on $d(T_{M_a})$. This is a contradiction. The contradiction has been assumed from the hypothesis that there is a Turing machine that solves the halting problem. Hence such a supposition must be wrong. The following pseudo-proof is aimed at clarifying the described hypothesis.

Elaborate Proof: The aim here is to establish an operational model fleshing out the exposed halting impossibility characteristic proof. We start by assuming that the halting problem is solvable, which implies that a solution algorithm exists. [6] states that a program X can be written as input to any program P with data D, and the program will determine whether P initiated on D ultimately halts. To make a new program Y, we append some commands (strictly, mathematical expressions) to X. We let Y modify X_s so that if X halts due to a decision that P started on D halts, Y goes into an infinite loop.

If X comes to a halt with the decision that P initiated on D, then Y comes to a halt as well. Finally, we make a new program called Z with the input P. Z is defined so that it raises Y on program P with an input corresponding to P (that is, the input data for Z is a program). There are two probable prospects for explaining what happens when we run Z on Z.

1. [6] Z started on input Z halts: If Z run on Z halts, then Y run on Z with input Z halts. If Y executed on Z with input Z halts, then X decided that Z started on Z never halts! Therefore, Z started on input Z halts implying that Z executed on input Z does not halt. This is a contradiction.
2. Z started on input Z does not halt: If Z executed on Z does not halt, then Y run on Z with input Z does not halt. If Y run on Z with input Z does not halt, then X decided that Z started on Z halts! Therefore, Z run on input Z never halts implying that Z started on input Z halts. This leads to a contradiction again.

Either alternative yields a contradiction, so the assumption that the halting problem is solvable must be incorrect.

2.1.1. A genetic empirical characterization of the Halting problem

This section presents both the genetic and the physical characteristics of the halting problem by examining the relationship between Turing Machines and modern-day general-purpose computers. In specific, we will be focusing on the machinery functioning frameworks. A Turing Machine is similar to a general-purpose computer in the following ways:

a) A Turing machine is made up of a "tape" of cells and a single active cell called the "head". The tape's cells can be any color, and the head can be in any condition. A rule specifies what the head should do at each step for each Turing machine. The rule considers the state of the head as well as the color of the cell in which the head is located. Then it describes what the head's new state should be, the color it should "write" into the tape, and the direction it should move in (either to the left or to the right). The initial color arrangement of cells on the tape corresponds to the computer's input. Both "program" and "data" can be included in this input. The Turing machine's steps correlate to the computer's operation.
b) The Turing machine's rules are analogous to computer machine-code instructions. Each section of the rule indicates what "operation" the computer should execute in response to a specific input. Turing Machines are distinguished from modern general-purpose computers by their storage capacities. However, because Turing machines have unlimited storage, we investigate the technical notion that the general-purpose computer is closer to a "finite state machine (FSM)" than a Turing machine, based on the proven characteristic similarities between the two machines. The presence of micro-transistors, which comprise the computer's major functional components, including computer memory, is important to the computer as an FSM.

2.1.2. Functional characterization of computer Halting operations

As mentioned in the previous section, general-purpose computers are fundamentally binary systems whose functioning principle is based on the working mechanisms of the constituent micro-transistors. All information in a computer is conveniently binary coded (both numerical data and text data are binary-coded).

The general-purpose computers use the micro-transistors states of either "High (True)" or "Low (False)" which genetically correspond to the binaries "1" or "0" respectively, in controlling the events between the background of the computer and the physical world. For instance;

i) in a color image, every pixel is represented by three binary sequences that correspond to the three-pixel primary colors (green, red, and blue). Each sequence encodes an expression (could be mainly numerical valued expressions) that represents the intensity of the corresponding particular color in the pixel. These sequences are then transmitted through a video driver program to millions of liquid crystals on the computer screen, and translated into the hue that the eye observes.

ii) Sound signals are also stored in binary form with help of a technique called pulse code modulation in which continuous sound waves are digitized by taking snapshots of their respective amplitudes every few milliseconds. The amplitudes are recorded as binary numbers which when read by the computer audio firmware, respective numbers determine how quickly the coils in the speakers should vibrate thus creating sounds of different frequencies.

From examples (i) and (ii), we deduce that physical activity such as recording and playing audio signals from a computer is genetically based on the binary functional characteristics of the computer. So such an activity is typically inherent, and not predetermined. This feature implies that the amplitudes of the signals correspond to a sequence of 1_s and 0_s if the signal is not continuum, and otherwise if continuum. We consider that for a program or a signal to be continuum, the computer operations must only be based on a single characteristic invariant binary-based state (either 1_s or 0_s) and not both states. Computer programs are naturally designed to operate with respect to binary characteristics. Considerably, the computer programs are distinct from an inherent mathematical function in that they cannot be characterized as total mathematical functions from an initial state to a final state without considerations of a time element. So inherently, computer programs have been characterized as functions or relations with a time component. The characterization based on the time element is responsible for non-termination in a computer program augmented as either a total function or as a partial function. The time element may be as simple as a Boolean variable that distinguishes finite from infinite time [5]. However, for physical interpretation, it can be a numeric variable extended with an infinite value to account for the non-termination [1]. We proceed by exploring the working principle of a (FSM) based on the binary system. We consider that given an arbitrary program and an input, for the input to decide with success if the given program halts, both programs must have a similar initial state (true or false). Otherwise, the input decides a non-terminating program. This conclusion is based on the genetic understanding that the states 1_s or 0_s cannot represent similar computation in a single event. For this reason, the halting problem would be impossible.

3. Establishing Euclidean geometric problems

The Euclidean geometric system (here we refer to the model established in the first six books of Euclid's Elements [11]) accentuates formalizing geometry using axioms, which appeal directly to basic concepts of geometry such as points and lines as opposed to a background for objects such as Cartesian spaces. For this paper, we in specific focus on the trisectability of angles posed by the ancient Greeks. The problem of "angles trisection" asks that: given an arbitrary angle (specifically a plane angle [12]), conceive a straightedge-compass scheme for sectioning the angle into three "equal" fractions. The inherent angles trisectability statement asks for a fixed straightedge and compass scheme that works for all plane angles. Further, the problem is restricted to some fundamental geometric conditions which most people inadvertently violate in their attempts to trisect angles (we suppose that the illustrated incompatibility schemes show the faults in employing analytic techniques as authoritative means for proving the non-trisectability of angles. It is upon the non-trisectability proofs where the geometric cryptographic protocols are established). It is expected that to resolve the trisection of angles one must use straightedge compass techniques, the solution (proof) must not involve measurements, an attempt at such a solution should not involve arithmetic, and that the geometric scheme must have a finite construction step. All these restrictions are inherently in agreement with the arbitrary nature of Euclidean geometric constructions. We consider that in solving the trisection of an angle, the size of the given angle should not be known [7].

This observation rules out the validity of applying real numbers as geometric magnitudes (a common practice characterizing the non-Euclidean geometric proofs). For centuries, mathematicians, philosophers, and science scholars have wrestled with the trisection of angles, till 1836 when first, the problem was vaguely stated as an impossible straightedgecompass problem; using ideas from the Galoi's fields [13]. However, we retrofit that the stated nontrisectability proof for any angle is flawed and it does not exactly show the trisection problem as an impossible problem. This work will employ a trisection scheme developed by [7]; to show the incompatibility between the angle trisection problem and the halting problem. The following §3.1 involves the genetic characterization of the angle trisection problem.

3.1. A characteristic proof of the angle trisection problem

A handy example to consider in this case is the geometric proof for the Pythagorean theorem (illustrated by [7]). The genetic proof for the Pythagorean theorem typically involves establishing the geometric relationship between magnitudes of similar kinds. In the proof, the is no application of real numbers as a substitute for geometrically constructed magnitudes. We think the reason for this was to preserve the purity of the Euclidean geometric system (in which real numbers play significant roles that which is not a representation of geometric magnitudes). A characteristic proof of trisecting any given angle must equally, follow the genetic rigor for establishing Euclidean geometric proofs. We consider that, any proof based on the constructability of only those magnitudes expressible as relative primes of the form a/b suggest an inherently non-Euclidean geometric generic property that given straightedge and compass, any constructible magnitude should be expressible as a rational fraction. Throughout §2, attempts have been made to characterize the existing geometric cryptographic principles. We assert that neither of the available geometric cryptographic models suits the inherent requirements for a Euclidean geometric proof. This is considered due to the nature of the proof for the halting problem. In both theory and practice, the proof for the halting problem is shy off the rigor required for typical straightedge and compass operations proofs. We thus carry on with the analytic examination of the angles trisection scheme established in [7]. The focus here is to establish the characteristic relations between the halting problem and the analytic nature of the angles nontrisectability proofs.

3.1.1. Characteristic analysis of the "AG-algorithm" solution

The **"AG-Algorithm"** provides a fixed system that works for any given angle. The scheme presents a "mothoddiversity" framework in which results at any desired accuracy are possible. The principle goal of this section is to provide a review characteristic of the **"AG-Algorithm"** algorithm using purely algebraic approaches.

3.1.2. Algebraic equivalence of the "AG-algorithm" solution

We consider Figure 5 adapted from [7] obtained following application of the **"AG-Algorithm"** on an arbitrarily small-angle $\angle JAB$ constructed in a unit circle with; $A = (0,0), B = (1,0)$, and, $J = (\cos\theta, \sin\theta)$.

Let the points O and N trisect the chord $\overline{JB}$ such that $|BO| = |ON| = |NJ|$. We set points P and Q on the unit circle defined by the following equations: P = Intersection of the line $\overline{D'N}$ and the arc $\overline{JB}$. Q = Intersection of the line $\overline{OA}$ and the arc $\overline{JB}$.

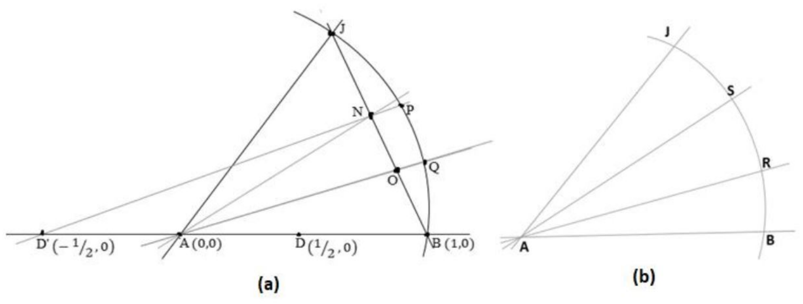

Figure 5. AG-Algorithm Results Analysis (Figure adapted from [7])

Appendix (2(a)) [7] we assume a point of convergence upon which the trisection error the of chord $\overline{JB}$ (although for unlike in appendix 1 [7] which is constructed upon a more complex geometric scheme appendix (2(a)) offers a simple analytic scheme for examining a specified angle which is not inherently obtained through complex operations as established in appendix 1 [7]) corresponds to the trisection error. Therefore, we reduce the trisection scheme depicted in Figure 5 (b)) to the assumption based on Figure 5 (a)) on the trisectability of the curve $\widehat{JB}$ so that $\angle PAQ = \angle JAB/3$, implying $\overline{PQ} = \overline{BR} = \overline{RS} = \overline{SJ}$ and $\widehat{PQ} = \widehat{BR} = \widehat{RS} = \widehat{SJ}$. Thorough analysis is done with the aid of a MATLAB code to show that as in the case of n-sectioning of a straight line segment, the **"AG-Algorithm"** is an exact trisection [7]. The subsequent argument concerns investigating if

$\angle PAQ = \theta/3$, where $\theta = \angle JAB$. We initiate the investigation with the trisection of chord $\overline{JB}$, which aids us in determining the positions O and N.

Claim 2. Chord $\overline{JB}$ is exactly trisected such that $\overline{BO}$ = $\overline{ON}$ = $\overline{NJ}$.

Proof of Claim 2. Applying vector notation based analysis, we set:

$$\overline{JB} = (1,0) + t(\cos\theta - 1, \sin\theta) \ \text{ with } \ 0 \leq t. \tag{1}$$

From Eq. (1), when $t = 0$ we get $B = (1,0)$, and when $t = 1$ we get $J = (\cos\theta, \sin\theta)$. We preserve the positions B and J for later use. Assuming that chord $\overline{JB}$ is exactly trisected, we now set $t = \frac{2}{3}$ and solve for point N as:

$$\begin{aligned} N &= (1,0) + 2/3(\cos\theta - 1, \sin\theta)\,, \\ N &= (1 + 2/3\cos\theta - 2/3, 2/3\sin\theta)\,, \\ N &= (1/3 + 2/3\cos\theta, 2/3\sin\theta)\,. \end{aligned}$$

Again from Eq. (1), we put $t = 1/3$ and solve for point O as follows:

$$\begin{aligned} O &= (1,0) + 1/3(\cos\theta - 1, \sin\theta)\,, \\ O &= (1 + 1/3\cos\theta - 1/3, 1/3\sin\theta)\,, \\ O &= (2/3 + 1/3\cos\theta, 1/3\sin\theta)\,. \end{aligned}$$

This proof assumes that since $\overline{D'N}$ and $\overline{OA}$ are parallel as established by [7] and that since points O and N were obtained using similar plane geometric configurations, chord $\overline{JB}$ is exactly trisected. Now to compute the angle trisection errors, we have to determine the points $\overline{D'N}$ and $\overline{OA}$ on the circumference of a unit circle. We proceed using points O and N in writing the line equations $\overline{D'N}$ and $\overline{OA}$ as follows:

$$\text{Line } \ \overline{D'N} \rightarrow y - 0 = \left(\frac{(2/3\cos\theta - 0)}{(1/3 + 2/3\sin\theta + 1/2)}\right)(x + 1/2)\,. \tag{2}$$

Eq. (2) further implies that

$$y = a(x + 1/2)\,, \tag{3}$$

$$\text{Line } \ \overline{OA} \rightarrow y - 0 = \left(\frac{(1/3\cos\theta - 0)}{(1/3 + 2/3\sin\theta + 1/2)}\right)\,. \tag{4}$$

Eq. (4) further implies that

$$y = b(x)\,. \tag{5}$$

To find the coordinates of the points P and Q we determine the points of intersections between lines $\overline{D'N}$ and $\overline{OA}$ with the unit circle defined as:

$$x^2 + y^2 = 1. \tag{6}$$

Using the equation for $\overline{D'N}$ in (6) we get;

$$x^2 + x\,(x + 1/2) = 1\,. \tag{7}$$

Simplifying (7) we obtain Eqs (8) and (9) as follows;

$$\begin{aligned} &x^2 + a^2\left(x^2 + x + 1/4\right) = 1\,, \\ &(1 + a^2)x^2 + a^2x + a^2/4 - 1 = 0\,, \\ &x = \frac{-a^2 + \sqrt{a^4 - 4(1 + a^2)\left(\frac{a^2}{4} - 1\right)}}{2(1 + a^2)}\,, \end{aligned}$$

$$x = \frac{-a^2 + \sqrt{a^4 - 4\left(\frac{a^2}{4} + \frac{a^4}{4} - 1 - a^2\right)}}{2(1+a^2)},$$

$$x = \frac{\sqrt{3a^4+4} - a^2}{2(1+a^2)}, \tag{8}$$

$$y = a\left(\frac{\sqrt{3a^4+4} - a^2}{2(1+a^2)} + \frac{1+a^2}{2(1+a^2)}\right) = a\left(\frac{\sqrt{3a^4+4}+1}{2(1+a^2)}\right). \tag{9}$$

Consequently, Eqs (8) and (9) give us the point P, Eq. (10)

$$P = \left(\frac{\sqrt{3a^4+4} - a^2}{2(1+a^2)}, \frac{\sqrt{3a^4+4}+1}{2(1+a^2)}\right). \tag{10}$$

Now we find the point of intersection between the line $\overline{OA}$ and the unit circle circumference as follows:

$$x^2 + (bx)^2 = 1 = (1+b^2)x^2 = 1 = x = \left(\frac{1}{\sqrt{1+b^2}}\right). \tag{11}$$

From Eq. (11) we make:

$$y = b(x) = \left(\frac{b}{\sqrt{1+b^2}}\right). \tag{12}$$

So that the point Q becomes:

$$Q = \left(\frac{1}{\sqrt{1+b^2}}, \frac{b}{\sqrt{1+b^2}}\right). \tag{13}$$

which lies on the unit circle.

From Figure 5, we apply the dot product in determining the angle between AQ and AP as follows:

$$\underline{u}.\underline{v} = |\underline{u}|.|\underline{v}|\cos(u, v). \tag{14}$$

Since P and P are on the unit circle we get:

$$\cos(\angle PAQ) = \overline{AP}.\overline{AQ}. \tag{15}$$

To determine the dot product, we consider the trigonometric functions (cosine and sine) for an angle of some magnitude (say $\angle PAQ = 0.75^\circ$) at 15 decimals accuracy and the slopes a and b as follows;

$$PAQ = (\cos(0.75^\circ), \sin(0.75^\circ)) = (\cos(\pi/240), \sin(\pi/240)), \tag{16}$$

$$a = \left(\frac{(2/3\sin\theta)}{(1/3 + 2/3\sin\theta + 1/2)}\right) = \left(\frac{(2/3\cos\theta)}{((1+2(\cos\theta)+3/2)/3)}\right) = (4(\sin\theta)/(5+4(\cos\theta))), \tag{17}$$

$$b = \left(\frac{1/3(\sin\theta)}{2/3 + 1/3(\cos\theta)}\right) = (\sin\theta/2 + (\cos\theta)), \tag{18}$$

$$c = (\cos\theta/2 + (\cos\theta)). \tag{19}$$

Using Eqs (17) and (19) in Eqs (10) and (13), we obtain the coordinates of points P and Q respectively. We then use in the cosine triple angle formula $\cos(3\theta) = 4\cos^3(\theta) - 3\cos(\theta)$; the variables a and b and the points P ad Q in determining the resulting angle trisection error. For simplistic analysis, we employ the MATLAB script shown in (appendix 2 (a) [7]) which is based on the condition:

$$pi/(\theta \times 1000000000) with 0 > \theta = pi/240.$$

Under this condition, the **"AG-Algorithm"** projects the most accurate to exact results depending on the required accuracies limits. □

4. The angle trisection-Halting problem analogy

The "angle trisection-halting problem" analogy of geometric cryptography is based on the notion that it is easy to triple a given angle than it is to trisect such an angle. The geometric cryptography common operations in the realm of symmetric cryptography functions assign the triple of a magnitude (say angle) to a given angle. Hence, symmetric cryptography can be thought of as a one-way function in which the only constructions permissible are those mimicking the ruler and compass constructions workflows. This section aims at constructing a one-way geometric identification scheme that is then used to characterize the "angle trisection-halting problem" impossibility analogy.

4.1. One-way geometric cryptography identification scheme

This section adopts the identification scheme suggested by [6,10]. The identification model is probabilistic based on the tossing of a coin. Depending on the number the coin is flipped, the outcome is "true" if and only if all the chances are "heads" and otherwise, if "tails" appear. We assume two parties; Alex and June where Alex wishes to establish a means of proving his identity later to June.

4.1.1. Initialization of the identification scheme

Alex publishes a copy of an angle Y_A which is constructed by Alex as the triple of an angle X_A he has constructed at random. Because trisecting an angle is impossible Alex is confident that he is the only one who knows X_A.

4.1.2. Working of the identification scheme

Alex gives June a copy of an angle R which he has constructed as the triple of an angle K that he has selected at random. June flips a coin and tells Alex the result. If June says "heads" Alex gives June a copy of the angle K and June checks that $3 * K = R$. If June says "tails" Alex gives June a copy of the angle $L = K + X_A$ and June checks that $3 * L = R + Y_A$. The four steps are repeated t times independently. June accepts Alex's proof of identity only if all t checks are successful. This protocol is an interactive proof of knowledge of the angle X_A (the identity of Alex) with error 2^{-t}.

4.2. Characteristic limitations of the "Angle trisection-Halting impossibility" analogy

Through §4.1 we note that although the "angle trisection-halting problem" analogy is typically basic, this paper asserts that the concept is built upon a misconception. Consider the following statements which expose the limitations exhibited by the "angle trisection-halting problem" analogy.

a The "angle trisection-halting problem" analogy assumes that the trisectability of an arbitrary angle using compass and straightedge is inherently probabilistic. This perspective is incorrect as it is built on the assumption of prior knowledge of the size of the resulting angles that trisect a given angle (as well, a known angle) thus it is either "true" or the binary 1 when the trisection is achieved, and "false" or the binary 0 otherwise. In contrast, the problem of angles trisection requires no predetermined results, neither is the size of the trisectioned angle necessary in the construction (established in §3)

b As illustrated using the (FSM), the halting problem is inherently characterized by the two binary states (1_s and 0_s) which determines the nature of a program (either halting or continuum) and so the problem can never be solved since arbitrarily, computing events based on either (1_s or 0_s) cannot be inherently similar as $1 \neq 0$. From a geometric perspective, the use of the states (1_s and 0_s) assumes that binary 1 is geometrically reducible to binary 0. This is an inconsistent cognitive meaning of the analogy which implies that a quantity of magnitude 1 is reducible to another quantity of magnitude 0 via compass and straightedge, thus deciding the halting problem. The Euclidean geometric system forbids the application of real numbers as geometric magnitudes, and thus sensible, being a practical model, the number 0 is not allowed in straightedge and compass schemes.

c From §4.1.2, the error expression 2^{-t} suffer two main limitations: (i) it assumes that an angle could be trisected if the error is infinitely small because the more iterations there is, the small the error (that is, the error decrease with increase in t values); (ii) it also assumes that the exact trisection of an angle corresponds to infinitely repeated bisection steps in the interface between the alternatives of the states

(1_s and 0_s). Contrary, it is possible to have a small angle exactly trisected at $t = 1$ (following the **"AG-Algorithm"** [7]), and this angle multiplied to the desired fraction that leads to the trisection of a larger angle (this is an attribute of the **"AGAlgorithm"**). These limitations are further illustrated using the MATLAB code provided in appendix (2 (a) [7]), where the trisection of the given angle is exact for the values of t with respect to the sizes of the angles.

d Further, we consider from §4.1.2, the error expression 2^{-t}. The expression implies that the geometric trisection of an angle corresponds to repeated bisections in the sense that if we apply the t values sequentially in according to the number of iterations such that $t = 1,2,3,4,5,6,7$; we obtain the corresponding decreasing error results as; $t = 0.50.250.1250.06250.031250.0156250.0078125$ (the resulting angle is half of the previous angle). This is the misconception exposed by relating the trisection of an angle and solving the halting problem. The problem of angles trisection does not ask for approximate solutions regarding repeated bisections operations.

5. Discussion

Regarding the physical significance of the problems; the halting problem is characterized by a time component which may be based on a Boolean variable that distinguishes finite from infinite time [10], or it can be based on a numeric variable (such as integers) extended with an infinite value to account for non-termination [1]. On the other hand, the trisection of an arbitrary angle problem requires typically, a plane geometric scheme that sections any given angle into three equal fractions. It has been established that the Euclidean geometric system allows only those operations mimicking straightedge and compass rigor of constructions. Although in cryptographic schemes the ordinary practical straightedge and compass constructions are not exhibited, here we concern ourselves with the principles behind the "angles trisectability-halting problem" analogy. We begin by asserting that strictly, Euclidean geometric problems are not time-bound, and so is the trisectability of angles. The notion that the trisection of an angle in general requires infinite construction steps is a misconstruction of the inherent characteristics of the problem. Indeed, the notion suggests approximate solutions for angle trisection based on multiple bisections of a given angle. This view completely violates the standard notion of Euclidean geometric exactness as provided by [14].

Further, the characteristic non–Euclidean proofs (as modeled in cryptographic applications) introduce time and resource requirements for the problem of angles trisection. Indeed, it is one of the fundamental reasons that the practical trisectability of an angle via straightedge and compass operations correspond to the infinite geometric bisection of a plane angle. This notion is an evident characteristic feature of the elliptic curve cryptographic (ECC) applications [1,15]. For instance, in §4.1.2 the iteration error expression 2^{-t} is signify that no single instance the trisection of an angle could be numerically exact, or that the trisection accuracy of an angle increase with the increase in the number of iterations via repeated geometric operations of bisecting a magnitude. Nonetheless, we treat such reasoning as elusive and an invalid notion about the trisectability of angles. Further, the anticipation that a trisection is achievable if all results are "heads" impose a genetic Euclidean restriction which Euclid himself has not demonstrated through the Elements. §4.1.1 has outlined some of the possible limitations of the "angle trisection-halting problem" impossibility analogy which makes it an invalid concept for application of cryptographic principles in information processing. Further, the pseudo-codes in appendix (1 [7]), appendix (2 (a) [7]), and appendix (2 (b) [7]) expose that the trisection of any angle is possible given the treatment suggested is using the probabilistic error 2^{-t} for t iterations of a program; in which the error 2^{-t} corresponds to some degree of accuracy. To this effect, we observe that the genetic differences between the two problems ("angle trisection-halting problem") make them incompatible and thus invalidating the geometric cryptography concepts.

6. Conclusion

Throughout the provided review, we note that there is no genetic and physical relationship between the geometric trisection of an arbitrary angle and the halting problem. Thus, the focus of the paper has been achieved; to create an inherent demarcation between the theory of Euclidean plane geometry and the computability theory by breaking the ("angle trisection-halting problem") impossibility analogy. Since the trisection of any angle or any curve that subtends a given angle is reasonably, geometrically possible [7], the ("angle trisection-halting problem") analogy is fully invalidated. We, therefore, assert that the concept of geometric cryptography is genetically faulty and thus it should not be used as an important subject for cryptographic techniques as it provides weak geometric frameworks that can be easily broken during information transmissions between entities.

Author Contributions: All authors contributed equally to the writing of this paper. All authors read and approved the final manuscript.

Conflicts of Interest: "The authors declare no conflict of interest."

References

[1] Gaur, P., & Singh, P. (2013). A review of geometry based symmetric key encryption using ellipse. *International Journal of Computer Science and Mobile Computing*, 2(6), 1-6.

[2] Chen, H., & Cramer, R. (2006, August). Algebraic geometric secret sharing schemes and secure multi-party computations over small fields. In *Annual International Cryptology Conference* (pp. 521-536). Springer, Berlin, Heidelberg.

[3] Church, A. (1937). AM Turing. On computable numbers, with an application to the Entscheidungs problcm. Proceedings of the London Mathematical Society, 2 s. vol. 42 (1936-1937), pp. 230-265. *The Journal of Symbolic Logic*, 2(1), 42-43.

[4] "Halting problem," Wikipedia. Dec. 04, 2020. Accessed: Dec. 21, 2020. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Halting-problem&oldid=992374115.

[5] PGaur, P., & Singh, D. P. (2013). Geometry based symmetric key cryptography using ellipse. *International Journal of Application or Innovation in Engineering & Management*, 2(6), 37–44.

[6] Burmester, M., Rivest, R., & Shamir, A. (1997). Geometric cryptography: Identification by angle trisection. https://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=D0128CE747411B22666180F3395AB94F?doi=10.1.1.55.5432&rep=rep1&type=pdf.

[7] Alex, K. M. The independence of euclidean geometric system: A proof on the constructability of geometric magnitudes.

[8] Lin, S., & Rado, T. (1965). Computer studies of Turing machine problems. *Journal of the ACM*, *12*(2), 196-212.

[9] Minsky, M. L. (1961). Recursive unsolvability of Post's problem of "tag" and other topics in theory of Turing machines. *Annals of Mathematics*, *74*(3), 437-455.

[10] Rompel, J. (1990, April). One-way functions are necessary and sufficient for secure signatures. In *Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing* (pp. 387-394).

[11] Euclid. Euclid's Elements (Joyce, David E., Tran.). http://aleph0.clarku.edu/~djoyce/java/elements/elements.html.

[12] "Euclid's Elements, Book I, Definition 8." https://mathcs.clarku.edu/~djoyce/elements/bookI/defI8.html. (accessed Nov. 26, 2021).

[13] Wantzel, P. L. (1837). Recherches sur les moyens de reconnaître si un problème de géométrie peut se résoudre avec la règle et le compas. *éditeur Inconnu*, 366-372.

[14] Panza, M. (2012). Rethinking geometrical exactness. *Rethinking Geometrical Exactness*, 249-282.

[15] Kapoor, V., Abraham, V. S., & Singh, R. (2008). Elliptic curve cryptography. *Ubiquity*, *2008*(May), 1-8.