International Journal of Multidisciplinary and Current Educational Research (IJMCER)
ISSN: 2581-7027 ||Volume|| 1 ||Issue|| 2 ||Pages|| 12-18||2019||

A Simple Cryptographic Algorithm for Source Data Verification

[1,]Penn P. Wu, [2,]Michael Kern
[1,2,]*DeVry University*

ABSTRACT : *In this paper the authors propose a simple algorithm to generate cryptographic key pairs. Each pair consists of two keys: one contains three floating-point values (denoted as x1, y2, and D), the other contains two floating-point values (denoted as x2, y1). The five values are obtained by randomly select two points, P1(x1, y1) and P2(x2, y2), on a given line (represented by ax+by+c=0) as well as the distance D between them. The key pair is generated by repositioning the five variables in two separated "containers," (x1, y2, D) and (x2, y1, K), with K being the series number of the key pair. While the (x1, y2, D) key is kept by a "confirmor" device, the (x2, y1, K) key is assigned to a "confirmee." The verification happens only when the "confirmee" key can precisely and mathematically match with the "confirmor" key. In practice, the proposed algorithm provides low-resource hardware implementation, which is proper to ubiquitous computing device such as a sensor in a Radio-Frequency Identification (RFID) tag. The proposed cryptographic key is easy for implementation to be light, but also has as much security as decent encryption algorithm.*

KEYWORDS:*Cryptographic Key, Key-Based Verification, Cryptographic Verification, Source Data Verification.*

I. INTRODUCTION

Cryptographic algorithms have been providing security applications for years, and much of the time adding security can mean taking away efficiency (McDonald, n.d.). Computer systems have been shrinking and the security industry is getting to the point where they are designing ubiquitous systems that are almost out of sight and out of mind (Markoff, 2015; Kai, 2010). For example, radio frequency identification (RFID) systems are used a lot in physical distribution, identifying and tracking of documents, and even entry control points (Ahsan, Shah, and Kingston, 2010). The communication facilitated between the RFID tag and its reader is a wireless data transmission system, this brings in concerns about the confidentiality and overall security of its design and application it has in real-world scenarios (Duc et al., 2009).In terms of Geometry, every point along a line on a two-dimensional surface plane has a unique set of x and y coordinates, and is commonly denoted as (x, y). A point on a line in a given plane is denoted as $P_n(x_n, y_n)$ where n is an integer starting as 1. An "line segment" is said to be the collection of $P_1(x_1, y_1)$, $P_2(x_2, y_2)$, $P_3(x_3, y_3)$, …., $P_n(x_n, y_n)$.

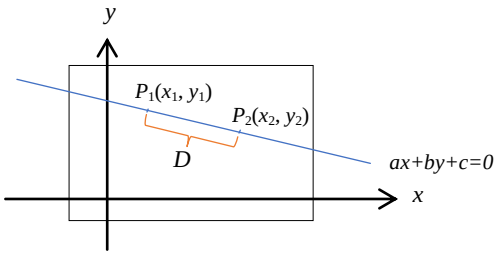

Figure 1. 2D coordinates

In this paper, the authors propose a simple cryptographic key pair consisting of five values, x1, x2, y1, y2, and D. They are represented by five variables of double type and are said to be "64-bit" in length. The cryptographic algorithm is based on calculating the distance, D, between two randomly selected points, $P_1(x_1, y_1)$ and $P_2(x_2, y_2)$. Both P_1 and P_2 must reside on a same line (represented by $ax+by+c=0$) of a two-dimensional surface plane, which is constructed by an x axis and a y axis as illustrated by Figure 1. The next sections will discuss the proposed cryptographic algorithm in detail.

Line Selection: The cryptographic algorithm starts with selecting a line in a two-dimensional plane. In Geometry, one generic form of line equation is the so-called linear equation, as shown below, which uses two variable x and y with two non-zero coefficient a and b as well as a variable c.

$ax + by + c = 0$ with $a \neq 0$ and $b \neq 0$

Let a=567, b=-3579, and c=12345, for example, the equation would be expressed as:

$567x + (-3579)y + 12345 = 0$

Generate Candidate Points : After defining the equation, the algorithm will ask the user to provide: (a) the number of key pairs needed, (b) the minimum number of x-coordinate where to start generating key pairs from, and (c) the increment. The term "increment" specifies how to increase value by a fixed scale. Figure 2 illustrates how a series of values increment by 0.06.

-8942365.25 [0.06] -8942365.19 [0.06] -8942365.13 [0.06] -8942365.07 [0.06] -8942365.01

Figure 2. Increment

The following is a sample of user inputs.

Min: -8942365.25
Amount needed: 1008
Increment: 0.06

With the given minimum number (-8942365.25) of x and the total number (1008) of key pairs needed, the maximum number (-8942304.71) of x can be yield by the following statement. It is necessary to note that giving the user the ability to dictate these variables enhances the security and increases the applications the algorithm can be used for.

```
max = min + (increment * (totalNumberNeeded+1));
```

The following is a sample code that can create a set the points, denoted as $P = \{P_1, P_2, P_3, \ldots, P_n, P_{n+1}\}$, from the selected line ($ax+by+c=0$), where n is total number of key pairs needed.

```
String^ getXY(float a, float b, float c, double x)
{
 y = -(a*x + c) / b;

 return (x + "," + y);
}
.....
for (x = min; x <= max; x += increment)
{
Console: WriteLine( getXY(a, b, c, x) );
}
```

Each element in the P set is said to be a "candidate point" because they all have an equal probability to be picked as the "primary" point. The next section will describe how to shuffle the elements in the P set, select the "primary" point, and then use the "primary" point to pair with other points to generate n number of key pairs.

Randomization : The term "to shuffle" refers to an action that will reposition all elements of the P set to an unpredictable order. Similar to the act of shuffling a deck of new playing cards, the purpose of shuffling the P set is to scramble the sequence of the index 1, 2, 3, …., and n. The following is an example of the P set after shuffling. By the way, P_{541} is said to be the "primary" point because it is the first element of the shuffled P set. The rest points (P_{28}, P_{935}, …, P_{94}) are said to be "partner" points.

$P = \{P_{541}, P_{28}, P_{935}, P_{409}, \ldots.., P_{37}, P_{162}, P_{609}, \ldots., P_{1002}, P_{435}, P_{94}\}$

The following is a sample code that illustrates how the algorithm shuffle the points in the P set. The x coordinate of all points are being kept in an array, and the array is shuffled using a *for* loop. Ultimately, after the complete

shuffling is done, the algorithm will produce a scrambled set of points. Shuffling the sequence will make the sequence unpredictable and further increase the security of the algorithm.

```
void shuffle(array<Int32>^ x, int totalNumberNeeded)
{
 Random^ rn = gcnew Random;
 int temp;
 int n;

 for (int i=0; i<totalNumberNeeded; i++)
 {
  n = rn->Next(totalNumberNeeded);

  temp = x[i];
  x[i] = x[n];
  x[n] = temp;
 }
}
```

Key Pair Generation: After shuffling the *P* set and identifying the "primary" point, the algorithm will calculate the distance from the "primary" point to each of the "partner" points using their coordinate sets. These distances are denoted as D_1, D_2, …., D_n. Table 1 illustrates how the pairing concept works.

Table 1.

Key No.	Primary	Partner	Coordinate Sets	Distance (Primary to Partner
1	P_{541}	P_{28}	(x_{541}, y_{541}) and (x_{28}, y_{28})	D_1
2		P_{935}	(x_{541}, y_{541}) and (x_{935}, y_{935})	D_2
3		P_{409}	(x_{541}, y_{541}) and (x_{409}, y_{409})	D_3
			.....	
			……..	
1006		P_{1002}	(x_{541}, y_{541}) and (x_{1002}, y_{1002})	D_{1006}
1007		P_{435}	(x_{541}, y_{541}) and (x_{435}, y_{435})	D_{1007}
1008		P_{94}	(x_{541}, y_{541}) and (x_{94}, y_{94})	D_{1008}

The algorithm uses the distance equation as shown below. In Geometry, the distance (*D*) between two points, (x_1, y_1) and (x_2, y_2), can be calculated using the distance equation.

$$D = \sqrt{(x1 - x2)^2 + (y1 - y2)^2}$$

The distance equation itself is actually derived from the Pythagorean Theorem and some simply call it the Pythagorean Theorem in disguise. The Pythagorean Theorem is a formula used to calculate the distance of a particular side of a right triangle. This equation is a useful tool that has been used to calculate the distance between two different points on a line.

To generate key pairs that will be used for the cryptographic implementation, the algorithm will swap the *y* coordinates of the "primary" and "partner" points, as shown in Figure 3. Notice the *y* coordinates were switched out with each other giving the reference to a completely different equation. After swapping, the coordinate sets are said to be "value sets." Every "value set" is assigned a unique key number, denoted as K_i. Then, add the distance between the primary point and the associated "partner" point, D_i, to the value set of "primary" point, where *i* is an integer in the range from 1 to *n*. Optionally, the key number can be added to the value set of "partner" key as reference. Figure 4 illustrates the components of a key pair.

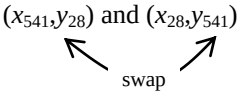

Figure 3. Swap *y* coordinates to make "value sets"

Key 1: (x_{541},y_{28}, D_1)
Key 2: (x_{28},y_{541}, K_1)

Figure 4. Components of a key pair

Table 3 is a sample lists of generated key pairs. It is necessary to note that each key pair in the Table 2 has a unique key number (which is denoted as K_i).

Table 3. Sample Key Pairs

Key No.	Component	Key Pair
1	Key 1: (x_{541}, y_{541}, D_1) Key 2: (x_{28}, y_{28}, K_1)	Key 1: (-8942332.85, -1439164.956, 31.17596204) Key 2: (-8942363.63, -1439169.909, 1)
2	Key 1: (x_{541}, y_{541}, D_2) Key 2: (x_{935}, y_{935}, K_1)	Key 1: (-8942332.85, -1439164.956, 23.94426079) Key 2: (-8942309.21, -1439161.151, 2)
3	Key 1: (x_{541}, y_{541}, D_3) Key 2: (x_{409}, y_{409}, K_1)	Key 1: (-8942332.85, -1439164.956, 8.021812513) Key 2: (-8942340.77, -1439166.23, 3)
	.....	
	……..	
1006	Key 1: (x_{541}, y_{541}, D_{1006}) Key 2: (x_{1002}, y_{1002}, K_1)	Key 1: (-8942332.85, -1439164.956, 28.01599372) Key 2: (-8942305.19, -1439160.504, 1006)
1007	Key 1: (x_{541}, y_{541}, D_{1007}) Key 2: (x_{435}, y_{435}, K_1)	Key 1: (-8942332.85, -1439164.956, 6.44174891) Key 2: (-8942339.21, -1439165.979, 1007)
1008	(x_{541}, y_{541}, D_{1008}) Key 2: (x_{94}, y_{94}, K_1)	Key 1: (-8942332.85, -1439164.956, 27.16505579) Key 2: (-8942359.67, -1439169.272, 1008)

Implementation : As shown in Table 3, the proposed algorithm would produce a list of paired keys: Key 1 and Key 2. The authors, hereinafter, choose to use (x_1, y_1) and (x_2, y_2) to represent the "primary" and "partner" point accordingly. The distance between each point is represented by D, and their unique key number is K.

Table 4. Simplified Notation

Key	Key 1	Key 2
Component	(x_1, y_2, D)	(x_2, y_1, K)
Role	Confirmor	Confirmee
Holder	Verification device	Value submitter

Key 1 will play the role of "confirmor" and would be saved to the server for verifying against Key 2 when submitted. Key 2 will play the role of "confirmee" and would be the key that is assigned to an RFID, IoT, or ticket. Value set of Key 2 will be submitted to or scanned by the "verification device" and compare against Key 1 for validation.

Verification: The verification is simply a reversed operation that verifies the "confirmee" key (Key 2) by cross-referencing the value sets with the key saved on the server (Key 1). This would be done by switching out the y coordinates between Key 1 and Key 2. The following represents a value set submitted to a designated "verification device" with v_1, v_2, and v_3 being three values.

(v_1, v_2, v_3)

The algorithm will check if (v_1, v_2, v_3) precisely equals to (x_2, y_1, K). It will begin with using the value of v_3 as the substitute of K, and then treating v_3 as reference to quickly search for the specified Key 1 stored on the server. If a matching Key 1 is found, as shown below, it should contain the value of D as the expected distance between the two points, (x_1, v_2) and (v_1, y_2), because (v_1, v_2) is assumed to be (x_2, y_1).

(x_1, y_2, D)

Based on the above logic, the algorithm will perform the following repositions, and then calculate their distance, denoted as d, using the distance equation.

- First set of x and y coordinates: (x_1, v_2)
- Second set of x and y coordinates: (v_1, y_2)

The following illustrates how the value of *d* is obtained. If the calculated value of *d* is precisely equal to the value of *D*, then (v_1, v_2, v_3) is proven to be (x_2, y_1, *K*); otherwise, the submitted value set (v_1, v_2, v_3) is an invalid set and should be rejected.

$$d = \sqrt{(x1 - v1)^2 + (v2 - y2)^2}$$

II. SCENARIO EXAMPLE

An event (such as concert) needs 2000 tickets, with each ticket having a unique pair of keys: one is kept by the server for verification; the other is printed (or burned) on the ticket to be verified against the server key. The following generates 2001 points (with one of the point to be picked as "primary" point and the rest 2000 points as candidates of "partner" points).

Enter the total number needed: 2000
Enter the min: -8942365.25
Enter the increment: 0.06
Enter the value of 'a': 567
Enter the value of 'b': -3579
Enter the value of 'c': 12345

Table 4 is a sample list of the 2000 key pairs whose Key 2 will be printed on the concert tickets. While P_{1542} is the "primary" point, P_{621}, P_{401}, P_{713}, …, P_{1441} are "partner" points in a randomized sequence.

Table 4:

Key No.	Primary	Partner	Key Pair
1	P_{1542}	P_{621}	Key 1: (-8942272.85, -1439169.909, 91.94814028) Key 2: (-8942363.63, -1439155.299, 1)
2	P_{1542}	P_{401}	Key 1: (-8942272.85, -1439161.151, 36.82791746) Key 2: (-8942309.21, -1439155.299, 2)
3	P_{1542}	P_{713}	Key 1: (-8942272.85, -1439166.23, 68.79399073) Key 2: (-8942340.77, -1439155.299, 3)
2000	P_{1542}	P_{1441}	Key 1: (-8942272.85, -1439160.504, 32.75618453) Key 2: (-8942305.19, -1439155.299, 2000)

The first tickets sold for the concert would contains a value set (-8942363.63, -1439155.299, 1), as shown in Figure 5.

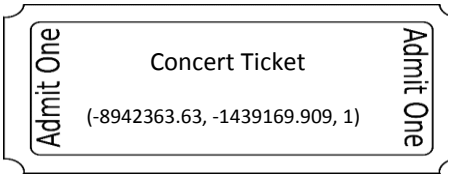

Figure 5. A Concert Ticket

Whenever the ticket is processed, the algorithm will check if the submitted value set (-8942363.63, -1439155.299, 1) is precisely the Key 2, (x_2, y_1, *K*), of the key pair whose key number is 1. By assuming $K = 1$, the algorithm will use 1 as reference to obtain the associated Key 1 set (-8942272.85, -1439169.909, 91.94814028). Then, swap -1439155.299 and -1439169.909 to produce two points, (-8942272.85, -1439155.299) and (-8942363.63, -1439169.909). Finally, the algorithm will perform the following calculation and found the result matches with expected value of *D*. The ticket has in fact a valid key pair and decision is to allow access.

$$\sqrt{(-8942272.85 - (-8942363.63))^2 + (-1439169.909 - (-1439155.299))^2} = 91.94814028$$

III. DESIGN PRINCIPLES

In this section, the authors list brief descriptions of design principles of the cryptographic key.

- The implementation of this cryptographic key is not overly complicated, so it does not hinder performance or efficiency, providing ease of use for a wide range of users and applications.
- The cipher is based off of a line, since there is no end to a line, the amount of available data points is unlimited. Therefore, increasing the amount of applications the cryptographic key is suited for.
- The computational power needed to perform the equations is low enough to be used by Ubiquitous Sensor Networks, (USN)
- Implementation of this cryptographic key is light weight enough to be used on ubiquitous systems or IoT devices.
- Programmed with variables being assigned with a double operator to implement 64-bit key length needed for low resource implementation.

IV. SECURITY ANALYSIS

A few things need to happen for proper security with this algorithm. First, and foremost, the points on the given line need to be shuffled. Without a shuffled sequence, simply collecting two tickets could give anyone enough information to extrapolate the rest of the points. This would be done by simply measuring the distance between the two points assigned to the ticket, finding the distance, and applying that to other tickets. Secondly, both the x and y coordinates need to be used. Using 0 every time for either the x or y points will result in a straight line along the x or y coordinate. Having perfectly vertical or horizontal line, on the graph, dramatically decreases the cryptographic key's strength. Removing one of the x or y points and using a zero would be removing half of the data, creating simple equations that can ultimately be solved quickly. For example, when the algorithm multiplies x_1 and y_2 if one was a zero the answer would be guaranteed to be a zero, this is something predictable, something that cannot be present in a cryptographic key like this. Finally, after the usage of the cryptographic key, the user needs to generate a new one for the next use. Using the same sets of points, from the same line, will result in a vulnerability within the cryptographic key. Using the concert example, if the same points were used for two different concerts, one ticket could give you access to both shows. Fortunately, with this type of cryptographic key, it is very easy to generate a completely new line and set of points.

V. CONCLUSION

The authors proposed a new cryptographic key with variable length and "64-bit" key length. It provides low-resource hardware implementation, which is proper to ubiquitous computing devices, IoT devices, and many others. With the emergence of the IoT, and the ever-growing amount of "connected" devices bring more security vulnerabilities. Taking preventative actions like adding cryptographic algorithms similar to what the authors proposed in this paper is the first step to being able to have a secure device on a network of ever-changing threats. Most crime including digital, comes from opportunity, having no encryption on many devices is an opportunity for malicious users, very similar to having an unlocked door for anyone to come in and snoop around. Manufactures need to understand that having a low resource device does not mean there is not room for encryption. People need to understand that in order to have proper encryption does not mean you always need "256-bit" encryption for every application. From a security standpoint, the authors have concluded this cryptographic key has suitable security for today's cryptography practices. The algorithm is made for ease of use and does not hinder efficiency and has the ability to accommodate an unlimited number of users and applications. The "64-bit" key length is an appropriate size for low resource applications and will not hinder the performance or effectiveness of the cryptographic key.

REFERENCES

[1] Ahsan, K., Shah, H., & Kingston, P. (2010). RFID applications: An introductory and exploratory study. IJCSI International Journal of Computer Science Issues, 7(1), p. 1-7.

[2] Duc, D., Lee, H., Konidala, D., & Kim, D. (2009). Open issues in RFID security. Proceedings of 2009 International Conference for Internet Technology and Secured Transactions, (ICITST). DOI: 10.1109/ICITST.2009.5402510
https://pubweb.eng.utah.edu/~nmcdonal/Tutorials/EncryptionResearchReview.pdf

[3] Kai, C. (2010). Design Principle for Ubiquitous Computing. Proceedings of Master Seminar of University of Fribourg, Switzerland. Retrieved from

https://diuf.unifr.ch/main/pai/sites/diuf.unifr.ch.main.pai/files/education_seminar_design_principle_of_ubiquitous_computing_kchen.pdf

[4] Markoff, J. (2015). Smaller, faster, cheaper, over: The future of computer chips. Retrieved from Technology Site of The New York Times: https://www.nytimes.com/2015/09/27/technology/smaller-faster-cheaper-over-the-future-of-computer-chips.html

[5] McDonald, N. (n.d.). Past, present, and future methods of cryptography and data encryption. Department of Electrical and Computer Engineering, University of Utah. Retrieved from