3D cryptography

João Carlos Leandro da Silva

Via Medole 22, 46043 Castiglione delle Stiviere (MN), Italy

We propose a new method to encrypt, transmit and decrypt data in three-dimensions between two smartphones. This technique consists of three distinct algorithms here called encryption, transmission and decryption. The first algorithm transforms any alphanumeric string into a multicolored three-dimensional figure. The second algorithm converts the given string into a sequence of binary bits. The third algorithm transforms the respective binary sequence into the original string. We assume that the transmitter first types the message and then sees a three-dimensional representation of it, while the receiver first sees a three-dimensional figure and then reads the original message. Practical implementation of this system was done using the program *Mathematica* and the size of the code is small. Since all three algorithms can be executed relatively fast, we present an outline for a client-server architecture and an application.

I. INTRODUCTION

Cryptography can be defined as the art and science of transmitting information in a secure manner. There are many types of cryptography such as public-key cryptography, visual cryptography [1], quantum cryptography, physical cryptography among others. Some of these are well established technologies while others are simply interesting theoretical ideas that may never be transformed into a tangible service and/or product.

The believe that abstract concepts such as numbers could be efficiently translated into real objects has endured for a long time but was only achieved recently [2]. A valid coding scheme was created for every positive integer whereby each number has a unique three-dimensional representation. For example, five is represented by an orange cube followed by an empty space and a green pillar. Using the standard ASCII character code where uppercase letters A, B,…,Z are assigned the numbers 65, 66,…,90 and the lowercase letters a, b,…,z have the numbers 97, 98, …,122 we can proceed to encrypt any text message. In order to decrypt a given three-dimensional figure, we must first translate it into an equivalent sequence of binary digits and then apply the previously mentioned coding scheme to get the respective text.

II. ENCRYPTION ALGORITHM

The encryption algorithm is implemented as a user-defined *Mathematica* function of one variable. We have split the function into five sections in order to explain it more clearly. The first section checks if the input is indeed a valid string and, if so, converts it into its ASCII character code. Furthermore, the resulting ASCII character code is converted into a positive integer. The second section checks the length of the input string. If the string is only one character long then the resulting three-dimensional figure will always be the same. We assume that text messages consisting of a single character are meaningless and therefore do not require encryption. Please note that if you type in any character plus a single space (before or after the character) such text message has length two, showing that it is possible to disguise the number of characters within a message. If the string is composed of two or more characters then the resulting three-dimensional figure will always be different, every time the function is called. At the end of this section, the previous positive integer has been transformed into another number. The

third section takes such number and converts it into a binary sequence. It repeats such operation with both the previous and next integers. For example, if our number is 345 then its binary sequence is 101011001. The next integer is 346 and its binary sequence is 101011010. Likewise, the previous integer is 344 and its binary sequence is 101011000. These three binary sequences are set up vertically with the most significant bit on top and the least significant bit on the bottom. As a result, a matrix is created composed of three columns (previous, number, next) and, in this case, nine rows. It is obvious that the entries of such matrix are zeros and ones. The fourth section uses the given matrix to generate a new positive integer. The fifth section takes this integer and builds the three-dimensional representation associated with the input string.

III. IMPLEMENTATION OF ENCRYPTION (section 1)

```
encryption[s_]:=(If[StringQ[s]&& StringLength[s]>0 ,code=ToCharacterCode[s],Print["Error! Invalid Input."]];

max=Length[code];

char=Map[IntegerString,code];

insert[u_String]:=StringInsert[u,"0",1];

r=1;While[r<=max,If[StringLength[char[[r]]]==3,char[[r]]=char[[r]],char[[r]]=insert[char[[r]]]];r++]; char;

n=FromDigits[Apply[StringJoin,char]];
```

Please note that the *Mathematica* code above is only the first section of the user-defined function **encryption**. In order to make use of this function it is necessary to join the code of all five sections. The variable "n" represents the positive integer as described previously.

III. IMPLEMENTATION OF ENCRYPTION (section 2)

```
If[StringLength[s]==1,p=NextPrime[n^2+1,1],p=RandomPrime[{10^5,10^6}]];

m=Mod[FromDigits[IntegerDigits[n,2]],p];

While[m==0|| m==1,m=Mod[FromDigits[IntegerDigits[n,2]],p=NextPrime[3p+1]]];

m;
```

The variable "m" stands for the new positive integer as described previously.

III. IMPLEMENTATION OF ENCRYPTION (section 3)

```
number=IntegerDigits[m,2];

plus=IntegerDigits[m+1,2];

minus=IntegerDigits[m-1,2];

center=Append[PadLeft[number,First[Sort[Map[Length,{number,plus,minus}],Greater]]],0];

right=Append[PadLeft[plus,First[Sort[Map[Length,{number,plus,minus}],Greater]]],0];

left=Append[PadLeft[minus,First[Sort[Map[Length,{number,plus,minus}],Greater]]],0];
```

The previously mentioned matrix consists of three columns (previous, number, next) which are associated with the above variables ("minus", "number", "plus") and ("left", "center", "right"), respectively.

III. IMPLEMENTATION OF ENCRYPTION (section 4)

```
j=1;

adj=0;

seq={};

While[j<=Length[center]-1,

If[j==1&&First[Take[Transpose[{left,center,right}][[j]],{2,2}]]==1,

adj=Count[Transpose[{left,center,right}][[j]],1]+Count[Transpose[{left,center,right}][[j+1]],1]-1];

If[First[Take[Transpose[{left,center,right}][[j]],{2,2}]]==1,adj=Count[Transpose[{left,center,right}][[j-
1]],1]+Count[Transpose[{left,center,right}][[j]],1]+Count[Transpose[{left,center,right}][[j+1]],1]-1,adj=0];

seq=Append[seq,adj];

j++];

If[Length[seq]>1&&First[seq]==0,seq=Drop[seq,1],seq=seq];
```

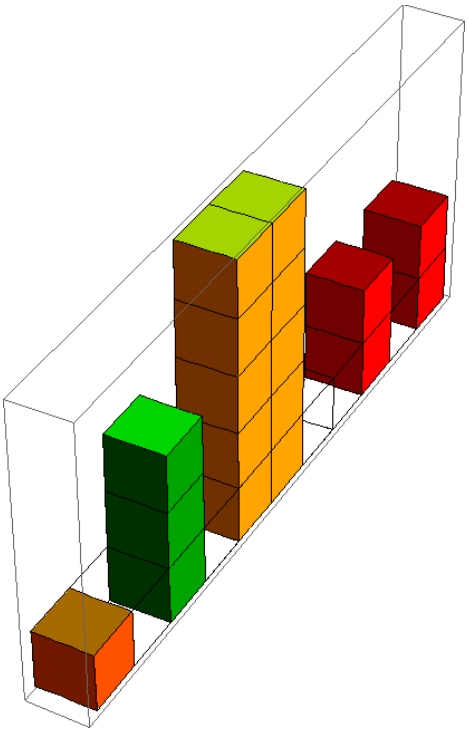
The above variable "seq" represents the previously mentioned new positive integer.

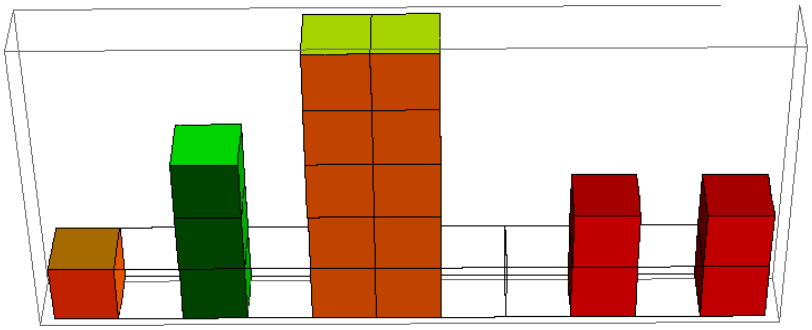
III. IMPLEMENTATION OF ENCRYPTION (section 5)

```
k=1;

rep={};

While[k<=Length[seq],

If[seq[[k]]==0,col={Opacity[0.0],Cuboid[{0,Length[seq]-k,0}]},

Which[seq[[k]]==1,col={Orange,Cuboid[{0,0,0}]},seq[[k]]==2,col={Red,Cuboid[{0,Length[seq]-
k,0}],Red,Cuboid[{0,Length[seq]-k,1}]},seq[[k]]==3,col={Green,Cuboid[{0,Length[seq]-
k,0}],Green,Cuboid[{0,Length[seq]-k,1}],Green,Cuboid[{0,Length[seq]-
k,2}]},seq[[k]]==4,col={Lighter[Purple],Cuboid[{0,Length[seq]-k,0}],Lighter[Purple],Cuboid[{0,Length[seq]-
k,1}],Lighter[Purple],Cuboid[{0,Length[seq]-k,2}],Lighter[Purple],Cuboid[{0,Length[seq]-k,3}]},

seq[[k]]==5,col={Yellow,Cuboid[{0,Length[seq]-k,0}],Yellow,Cuboid[{0,Length[seq]-
k,1}],Yellow,Cuboid[{0,Length[seq]-k,2}],Yellow,Cuboid[{0,Length[seq]-
k,3}],Yellow,Cuboid[{0,Length[seq]-k,4}]},seq[[k]]==6,col={Darker[Green],Cuboid[{0,Length[seq]-
k,0}],Darker[Green],Cuboid[{0,Length[seq]-k,1}],Darker[Green],Cuboid[{0,Length[seq]-
k,2}],Darker[Green],Cuboid[{0,Length[seq]-k,3}],Darker[Green],Cuboid[{0,Length[seq]-
k,4}],Darker[Green],Cuboid[{0,Length[seq]-k,5}]},seq[[k]]==7,

col={Black,Cuboid[{0,Length[seq]-k,0}],Black,Cuboid[{0,Length[seq]-k,1}],Black,Cuboid[{0,Length[seq]-
k,2}],Black,Cuboid[{0,Length[seq]-k,3}],Black,Cuboid[{0,Length[seq]-k,4}],Black,Cuboid[{0,Length[seq]-
k,5}],Black,Cuboid[{0,Length[seq]-k,6}]},seq[[k]]==8,col={Brown,Cuboid[{0,Length[seq]-
k,0}],Brown,Cuboid[{0,Length[seq]-k,1}],Brown,Cuboid[{0,Length[seq]-k,2}],Brown,Cuboid[{0,Length[seq]-
k,3}],Brown,Cuboid[{0,Length[seq]-k,4}],Brown,Cuboid[{0,Length[seq]-k,5}],Brown,Cuboid[{0,Length[seq]-
k,6}],Brown,Cuboid[{0,Length[seq]-k,7}]}]];

rep=Append[rep,col];

k++];

Graphics3D[rep])
```

The *Mathematica* code above is the last section of the user-defined function **encryption**. In order to make use of this function, it is necessary to join the code of all five sections. The variable "rep" represents the three-dimensional representation associated with the input as mentioned previously.

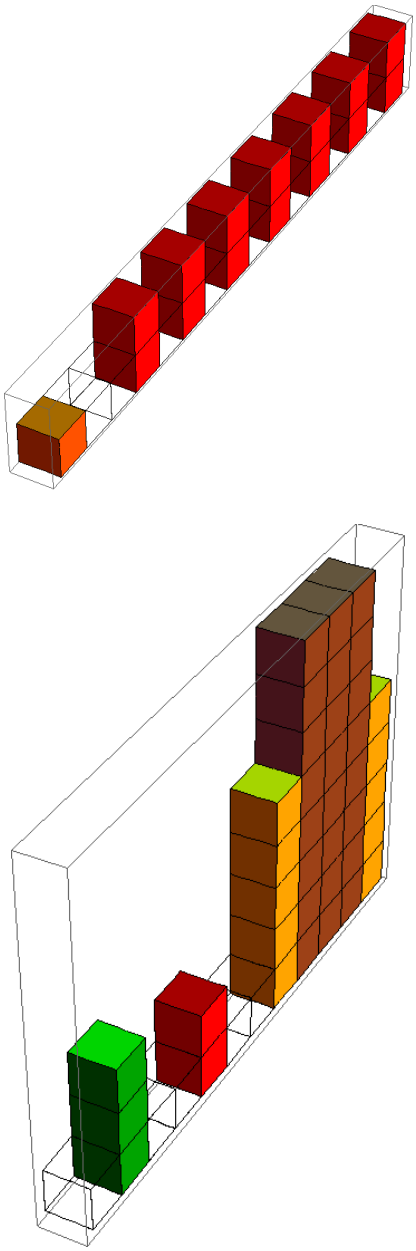
IV. ENCRYPTION EXAMPLES





Both of the above three-dimensional representations are the output of calling the user-defined function **encryption** using as input the capital letter "L". Please note that both figures are identical. We are just viewing the output from a different perspective. In fact, since "L" is a single character, every time the function **encryption** is called with this input, the resulting three-dimensional figure will always be the same.

IV. ENCRYPTION EXAMPLES (continuation)



The top figure is the output of calling the user-defined function **encryption** using as input the word "Love". Once again, the bottom figure is the output of calling the user-defined function **encryption** using as input the word "Love". In fact, since "Love" is a word, every time the function **encryption** is called with this input, the resulting three-dimensional figure will always be different.

V. TRANSMISSION ALGORITHM

The transmission algorithm is implemented as a user-defined *Mathematica* function of one variable. First, the algorithm checks if the input is indeed a string. If not, an error message is printed. Second, it transforms each character of the input string into its corresponding ASCII character code. Next, they are all joined into a single positive integer. Third, such integer is changed into its binary equivalent.

V. IMPLEMENTATION OF TRANSMISSION

```
transmission[s_]:=(If[StringQ[s],code=ToCharacterCode[s],Print["Error! Please type your text between quotation marks."]];

max=Length[code];

char=Map[IntegerString,code];

insert[u_String]:=StringInsert[u,"0",1];

r=1;While[r<=max,

If[StringLength[char[[r]]]==3,char[[r]]=char[[r]],char[[r]]=insert[char[[r]]]];

r++];char;

n=FromDigits[Apply[StringJoin,char]];

binarynumber=IntegerDigits[n,2])
```

The *Mathematica* code above pertains to the user-defined function **transmission**. The variable "binary number" represents the binary representation of the number mentioned previously.

VI. DECRYPTION ALGORITHM

The decryption algorithm is implemented as a user-defined *Mathematica* function of one variable. The input is a sequence of binary bits. First, the algorithm converts the input into a decimal number. Next, such number is transformed into a string. Second, the string is divided into equal segments. Each segment is converted into an integer. Third, every such integer is an ASCII code, so the corresponding character is saved. Finally, all characters are joined to form the original text.

VI. IMPLEMENTATION OF DECRYPTION

```
decryption[cn_List]:=(binary=cn;
word=IntegerString[FromDigits[binary,2]];
add[v_String]:=StringInsert[v,"0",1];
If[Mod[StringLength[word],3]==0,word=word,word=add[word]];
h=3;bag={};
While[StringLength[word]>0,
bag=Append[bag,StringTake[word,h]];
word=StringDrop[word,h]];
delete[f_String]:=StringDrop[f,1];
t=1;While[t<=Length[bag],
If[SameQ[StringTake[bag[[t]],1],"0"],bag[[t]]=delete[bag[[t]]],bag[[t]]=bag[[t]];
t++]];
FromCharacterCode[Map[FromDigits,bag]])
```

The *Mathematica* code above pertains to the user-defined function **decryption**. The variable "binary" represents the sequence of binary bits mentioned previously.

VII. CLIENT-SERVER ARCHITECTURE

The transmitter must accomplish two tasks. First, it needs to convert the input text into a three-dimensional representation so the user can see it. Second, it must be able to send a binary sequence to the server when the user decides to send his message to the receiver. At this point, it is easy to realize that in order to design such transmitter we need both the encryption and transmission algorithms. The encryption algorithm will perform the first task while the transmission algorithm will do the second. The server will manage all communications (encrypted with standard technology) between both transmitter and receiver. Also the server will perform all the required authentications. The receiver must convert the binary sequence from the server into the original text and transform such text into a three-dimensional representation. In order to design this receiver, we need both the decryption and encryption algorithms. Given that a smartphone is both a transmitter and a receiver it is clear that any cryptographic application must include all three algorithms. But where can we apply such technology?

VIII. MOBILE APPLICATION

As an innovative application for sending and receiving SMS (short message service) on smartphones. Let's go back in time and think about public-key cryptography. The typical scenario is two users (here called X and Y) where X wants to send a secret message to Y and vice-versa. They need to communicate over an insecure channel such as the Internet but the endpoints are assumed to be safe. Endpoints usually refer to the physical location of the devices that generate the messages to be transmitted. When public-key cryptography was invented, computers existed as devices that were physically located in a permanent position such as a mainframe in an office building. Since access to such facility was protected, the mainframe or endpoint was considered to be safe.

Today, computers exist as mobile devices and, consequently, both users and devices are constantly changing their physical locations which brings about a completely different landscape. In fact, the present situation is the reverse of the previously described scenario since the endpoints are insecure but the communication channel is safe because we now know how to secure it. For example, user X is the owner of smartphone1, while user Y owns smartphone2. We assume that X is a male and Y is a female and both users are far away from each other. Consider user X alone inside his house with all doors and windows locked. If X picks smartphone1 from the kitchen table and writes a message, we can state that X is in a secure zone because nobody can see his message nor get hold of his phone. So what do we mean by insecure zones? Now, consider user Y together with other people outside a restaurant. If Y sends a message from smartphone2, we can easily state that Y pertains to an insecure zone because someone can see her message or take possession of her phone. It is obvious that both X and Y can freely move from a secure to an insecure zone and vice-versa. So how does 3D cryptography assure that SMS privacy is guaranteed between any two mobile users? Suppose that you are sending a message to a friend and both of you have this application installed on your smartphones. You type your message, then press "Encrypt" and your text is transformed into a three-dimensional representation. You press on "Send" and your message is sent to your friend. On the other side, your friend notices that he has received a new message. He presses on "Show" and immediately visualizes (on the screen of his smartphone) another three-dimensional representation, but does not know what it means. He then presses on "Decrypt" and the application converts the given three-dimensional representation into the original text. Remember it is not possible to uncover the message from the three-dimensional representation because every time you press on either "Encrypt" or

"Show" the three-dimensional representation changes, even if the message is always the same. Furthermore, the application has two types of operation mode: normal and secure. The normal mode is used whenever you are alone and you do not have to worry about someone trying to read your SMS. The secure mode should be used whenever you are in the presence of others. In order to activate the secure mode, all you need to do is create your own personalized secret code which can be as short as a single character but as long as you want. Once done, the only person that can read and send any SMS is you. In case, someone gets hold of your mobile, he or she will only have three possibilities to guess your personalized secret code, after which the application will automatically close. In order to open the application, you must type the PIN (Personal Identification Number) of your smartphone.

IX. CONCLUSIONS

In this paper we proposed a new method to encrypt, transmit and decrypt data in three-dimensions between two smartphones. Since the size of all three algorithms is small and their execution speeds are relatively fast such technology seems suitable for mobile devices. Likewise, basic requirements for security and authentication have been outlined. Given that, an application to protect SMS privacy between smartphones has been considered. As a final remark, it would be interesting to search for additional applications of the present technique.

REFERENCES

[1] Moni Naor and Adi Shamir, *Visual Cryptography*, EUROCRYPT (1994), 1-21

[2] João Carlos Leandro da Silva, *The Rainbow of Primes*, Freund Publishing House, Tel-Aviv, 2009, 77-89