1 Introduction

The LSA cryptosystem is an asymmetric encryption algorithm which is based on both group and number theory that follows Kerckhoffs's principle, and relies on Gauss's Generalization of Wilson’s Theorem. Unlike prime factorization based algorithms, the eavesdropping cryptanalyst has no indication that he has successfully decrypted the ciphertext. For this reason, we aim to show that LSA is not only more secure than existing asymmetric algorithms, but has the potential to be significantly computationally faster.

2 Preliminaries

2.1 Groups

Any readers with prior knowledge of group theory (abstract algebra) may skip this section and start directly from section 2.2.

Since the LSA is operated from within mathematical groups, we will briefly explain the nature of a group and the properties used in this algorithm: A group is a set equipped with a binary operation. More specifically, it is a set of elements for which, when the elements are operated on with a specific binary operation (in our case modular multiplication) the following properties hold:

- Closure: If two elements are operated upon, the resulting element is an element of the group.

- Transitivity: If * is a binary operation and a, b, and c are elements of the group, then $a * (b * c) = (a * b) * c$.

- Identity: All groups have a unique identity element $e$ such that, for all elements $a$ of the group, $a * e = a$. 


- **Inverse:** Every element $a$ of the group has a unique inverse. The inverse is an element, $a^{-1}$, of the group such that $a \ast a^{-1} = e$, where $e$ is the aforementioned identity of the group.

The main group that is used for LSA is the *multiplicative group modulo* $n$. This is the set of all the integers coprime to, and less than, an integer $n$, equipped with modular multiplication. By convention, this group can be referred to symbolically as $U(n)$. For example, for $n = 12$ we have that $U(12) = \langle 1, 5, 7, 11 \rangle$ is a group if operated with multiplication modulo 12.

### 2.2 Theorem

A reader with prior knowledge of Gauss’s Generalization of Wilson’s Theorem can skip to section 2.3.

The core of the algorithm is based on a mathematical theorem called Gauss’s generalization of Wilson’s Theorem.

The theorem states that, if $U(n)$ is a group of all the integers relatively prime to $n$ and less then $n$, equipped with multiplication modulo $n$ we have that:

$$
\prod_{i=1}^{\phi(n)} a_i = \begin{cases} 
0 & n = 1 \\
-1 & n = 4, p^t, 2p^2 \\
1 & \text{otherwise}
\end{cases}
$$

where $p$ is a prime number, $t$ is a positive integer and the $a_i$’s are elements of the group $U(n)$, with $1 \leq i \leq \phi(n)$. Also recall that $\phi(n)$ is the Euler’s Totient Function which represent the number of integers coprime with an integer $n$ and less than $n$.

In other words, if $n = p^t$ or $2p^t$ we have that in $U(n) = \langle a_1, a_2, a_3, \ldots, a_{\phi(n)} \rangle$ it’s always true that

$$a_1 \cdot a_2 \cdot a_3 \cdots a_{\phi(n)} \equiv -1 \pmod{n}.$$

For the explanation of the procedure of the LSA we will use only the case where $n = p^t, 2p^t$. However the reader will easily understand how to use the algorithm in case $n \neq p^t, 2p^t$. 

3 Key Exchange

A connection through a key exchange is necessary in order to perform the LSA. We will not suggest any algorithm, if we name any, it is just to ensure the cleaner explanation.

Also, we now anticipate that when we refer to a key "k", the length of such integer is between 5 and 10 digit, which will ensure maximum security. Hence, if the algorithm starts by a connection through a Diffie-Hellman algorithm for example, the number k is not intended be the whole number generated by the algorithm, but only a choice of digits of it, between 5 and 10.

4 THE LSA

Suppose two parties want to secretly exchange information. This information should be considered symbolic by nature (e.g. numerically, alphabetically, etc.). It is customary in the field of cryptography, to assign names to the sender, receiver, and potential eavesdropper, as such we will we choose Alice, Bob and Eve respectively. They perform the following steps:

1. Alice and Bob begin by sharing a secret number k that is only known to Alice and Bob.
2. Next, k will be used as a reference by both parties to find the smallest positive integer n that satisfies the following properties:

   I) $k < n$.
   II) $n = p^t$ or $n = 2 \cdot p^t$ where p is an odd prime number and t a positive integer.

Note: The key exchange algorithms might generate massive numbers as keys. In that case, the k should be a derivation of a larger key K. It is very important to know that the length of k is efficient as long as $U(n)$ contains enough elements as there are symbols to share. For a merely alphabetical and digital messages 34 elements are enough. This means that a k of 4 digits is long enough to ensure security. It is not the scope of this paper to discuss the derivation of k from K and how long k must be; this will pertain the security that the user wants to provide. However, as an example, if the Diffie-Hellman algorithm generates a massive number K, then let k be the first 4 digits of K. Then derive n from k
3. Assuming that the above conditions have been met, then in $U(n)$ by Gauss’s Generalization of Wilson’s Theorem we have that

$$a_1 \cdot a_2 \cdot a_3 \cdots a_{\phi(n)} \equiv -1 \mod n.$$ 

At this point, Alice and Bob independently list the elements of $U(n)$ in ascending order as $U(n) = < \epsilon_1, \epsilon_2, \epsilon_3, ..., \epsilon_{\phi(n)} >$ where $\epsilon_i < \epsilon_j$ when $i < j$. Note that, since $k$ is only known to Alice and Bob, and $U(n)$ is chosen from the shared knowledge of $k$, then it must be the case that $U(n)$ is only known to Alice and Bob.

4. Alice chooses an element of the group to represent the plain text of her message, call it $\epsilon_h$.

5. To encrypt, Alice multiplies each of the elements of the group up to $\epsilon_h$ in the following way: $\epsilon_1 \cdot \epsilon_2 \cdots \epsilon_h \mod n \equiv c$ where $c$ is a component of the ciphertext.

6. Alice publicly sends $c$ to Bob.

7. Bob receives $c$ and multiplies $c$ with the other elements of the group in descending order and checks if $c \cdot \epsilon_{\phi(n)} \equiv -1 \mod n$, $c \cdot \epsilon_{\phi(n)} \cdot \epsilon_{\phi(n)-1} \equiv -1 \mod n$, all the way to $c \cdot \epsilon_{\phi(n)} \cdot \epsilon_{\phi(n)-1} \cdots \epsilon_{h+1}$ which will necessarily be congruent to $-1$ modulo $n$ by Gauss’s Generalization of Wilson’s Theorem, since all the elements of the group have been multiplied (and because $n$ in the desired form). At this point Bob knows that the next element of the group yet to be multiplied, $\epsilon_h$, is indeed the plaintext.

† Since $n - 1$ is an element of the group and it is indeed the element congruent to $-1$ modulo $n$, then the multiplication of the elements of the group might generate $n - 1$ even in some cases where not all the elements of $U(n)$ have been multiplied, unpredictably. Thus, before Alice sends the ciphertext $c$ she preforms $c \cdot \epsilon_{\phi(n)} \mod n$, $c \cdot \epsilon_{\phi(n)} \cdot \epsilon_{\phi(n)-1} \mod n$, all the way to $c \cdot \epsilon_{\phi(n)} \cdot \epsilon_{\phi(n)-1} \cdots \epsilon_{h+1}$ mod $n$ and records the number of additional times she generates elements congruent to $-1$ modulo $n$, and calls this number $\Sigma$. Then she publicly sends the tuple $C = (c, \Sigma)$, which becomes the ciphertext.

†† When Bob receives $C = (c, \Sigma)$ he will start multiplying the elements in descending order until he finds $\Sigma + 1$ elements congruent to $-1$ modulo $n$, revealing the plaintext.

8. Alice will send another symbol of the plaintext by running the LSA algorithm from the very beginning, including the sharing of the integer $k$. 

4
4.1 Example

Here is an example of the LSA operated within the group $U(n)$ equipped with multiplication modulo $n$.

For enhanced readability, we have chosen a group of small order, thus making each step more easily visualized.

Consider the case where Alice wants to secretly share the plaintext ‘7’ with Bob:

1. The key exchange algorithm generates $k = 53$.

2. Alice and Bob independently follow the LSA algorithm and conclude that the next useful integer is 54 because it satisfies the following properties:
   
   I) $53 < 54$
   II) $54 = 2 \cdot 3^3$.

3. Alice and Bob list the elements of the group in ascending order.
   
   $U(54) = \langle 1, 5, 7, 11, 13, 17, 19, 23, 25, 29, 31, 35, 37, 41, 43, 47, 49, 53 \rangle$.

4. Since Alice wishes to send ‘7’, she picks the 7th element of $U(54)$ in ascending order, which is 19.

5. Alice performs $1 \cdot 5 \cdot 7 \cdot 11 \cdot 13 \cdot 17 \cdot 19 = 1,616,615 \mod 54 \equiv 17 = c$.
   
   † Alice multiplies $c$ by each element in descending order to find the value of $\Sigma$ in the following way:

   \[
   17 \cdot 53 = 901 \mod 54 \equiv 37 \\
   37 \cdot 49 = 1,813 \mod 54 \equiv 31 \\
   31 \cdot 47 = 1,457 \mod 54 \equiv -1 \rightarrow \text{(First } \Sigma) \\
   53 \cdot 43 = 2,279 \mod 54 \equiv 11 \\
   11 \cdot 41 = 451 \mod 54 \equiv 19 \\
   19 \cdot 37 = 703 \mod 54 \equiv 1 \\
   1 \cdot 35 = 35 \mod 54 \equiv 35 \\
   35 \cdot 31 = 1,085 \mod 54 \equiv 5 \\
   5 \cdot 29 = 145 \mod 54 \equiv 37 \\
   37 \cdot 25 = 925 \mod 54 \equiv 7
   \]

   Since $c = 17$ and $\Sigma = 1$ (because Alice generated only one additional element congruent to $-1 \mod n$), then Stella publicly sends $C = (17, 1)$ to Bob.
6. †† Bob receives $C$ and starts to multiply $c$ by the elements of the group in descending order until he finds $\Sigma + 1$ (in this case $1 + 1$) elements congruent to $-1$ modulo 54 as follows:

\[
\begin{align*}
17 \cdot 53 &= 901 \mod 54 \equiv 37 \\
37 \cdot 49 &= 1,813 \mod 54 \equiv 31 \\
31 \cdot 47 &= 1,457 \mod 54 \equiv -1 \rightarrow \text{(First } \Sigma) \\
53 \cdot 43 &= 2,279 \mod 54 \equiv 11 \\
11 \cdot 41 &= 451 \mod 54 \equiv 19 \\
19 \cdot 37 &= 703 \mod 54 \equiv 1 \\
1 \cdot 35 &= 35 \mod 54 \equiv 35 \\
35 \cdot 31 &= 1,085 \mod 54 \equiv 5 \\
5 \cdot 29 &= 145 \mod 54 \equiv 37 \\
37 \cdot 25 &= 925 \mod 54 \equiv 7 \\
7 \cdot 23 &= 161 \mod 54 \equiv -1 \rightarrow (\Sigma + 1)
\end{align*}
\]

Since Bob found the second element that is congruent to $-1$ modulo 54, he knows that all of the elements of the group have been multiplied. This tells him that the next number in the sequence is the chosen number (19). Since 19 is the 7th group element, Bob has the plaintext ‘7’.

5 Extra Security

Note, if $p$ is a prime number, in $U(p)$, the product of all the integers in ascending order is an integer factorial, so recognizable. However, we can still use $U(p)$ groups just by starting the encryption by listing the elements of the group in descending order and use the normal procedure with the elements in backward order. So $c = \epsilon_{\phi(p)} \cdot \epsilon_{\phi(p-1)} \cdot \epsilon_{\phi(p-2)} \cdots \epsilon_h$, where $\epsilon_h$ represent the plaintext.

Note that, for the cyphertext $C = (c, \Sigma)$, the element $c$ is obviously an element of the group $U(n)$. Hence, more likely $c \pm 1$ is not. By previous agreement, Alice can send $C = (c + 1, \Sigma)$ or $C = (c - 1, \Sigma)$ to Bob anytime $c + 1$ or $c - 1$ are not in the group (as long as they previously agree on addition or subtraction). In this way, Bob will notice that the element sent by Alice is not listed in his group; In that case, Bob will consider the cyphertext $C = (c, \Sigma)$ after have subtracted (or added) 1 from $c+1$ ($c-1$).

Alice might send $(c + n, \Sigma)$, or $(c + n \pm 1, \Sigma)$, or $(c \cdot n, \Sigma)$.  

6
Alice might send a fake integer $d$ in $C = (d, \Sigma)$ as fake every once in a while after a certain signal. For example, if the previous element sent was a power of 2, then send $C = (d, \Sigma)$

### 6 The choice of the elements

In this section we will show one way to map a given group $U(n)$ to a set of meaningful symbols $\mathcal{P}$ that one might wish to encrypt. We will accomplish this via a surjective function $\psi : \mathcal{P} \mapsto U(n)$. The function $\psi$ must be surjective for each member of the codomain to have a corresponding element in the domain $\mathcal{P}$ that maps into it. This is required in order to make sure that each plaintext character is paired with a member of the group.

*Example:* Assume $\mathcal{P}$ is the set of symbols $\{e, f, g\}$. Then, in $U(9)$, $\psi$ maps the elements as follows:

1 $\rightarrow$ $e$
2 $\rightarrow$ $f$
4 $\rightarrow$ $g$
5 $\rightarrow$ $e$
7 $\rightarrow$ $f$
8 $\rightarrow$ $g$

Consider the case where Stella is operating with multiplication modulo 9, and she wants to send the symbol $f$. She can send this with either $1 \cdot 2 \equiv 2$, or $1 \cdot 2 \cdot 4 \cdot 5 \cdot 7 \equiv 1$. So she can send $f$ as either $c = 1$ or $c = 2$ as the first component of $C$.

### 7 One-Time-Pad

The LSA can be use as a One-Time-Pad when every symbol of the plaintext is sent through disjoint tuples, which is possible only by performing the algorithm from the very beginning for each symbol. This requires that Alice and Bob are equipped with as many integers $k$'s as there are symbols in the plaintext to share. Since running key exchange algorithms, which generate massive numbers, for each character of a message, is not a feasible task, we we show in this section how to generate a large number of secret $k$'s integers that will be used to start the LSA.
Assume that only one secret integer \( k \) is known by Alice and Bob. Also assume that with that \( k \) they start the LSA algorithm (see section 4). From \( k \) they derive \( n \). Alice sends an element of the group \( U(n) \) via the LSA. When Alice and Bob both agree on the element \( \epsilon_h \), they both perform the canonical multiplication of all the elements of the group up to \( \epsilon_h \):

\[
\epsilon_1 \cdot \epsilon_2 \cdot \epsilon_3 \cdots \epsilon_h = M.
\]

We will see in the example that for some wisely chosen \( n \) and \( \epsilon_h \), the integer \( M \) is large.

At this point Alice and Bob agree on a way to "cut the number in pieces" to generate many \( k \)'s.

### 7.1 Computational example

Assume Stella and Ben needs keys of lengths 4, 5 and 6. Then they will proceed as follow.

Assume the initial key exchange generated the integer 10000, then Stella and Ben List the elements of the cyclic group \( U(10082) \), which are 4970. Assume Stella chooses the 500th element of the ordered list. Ben runs the LSA and find the element. Then both multiply the first 500 elements together to generate \( M_1 \) and the next 4470 elements to generate \( M_2 \). Then they combine \( M_1 \) and \( M_2 \) and obtain

\[
M = 8 \, 606 \, 842 \, 941 \, 648 \, 980 \, 141 \, 500 \, 390 \, 929 \, 340 \, 333 \, 380 \, 821 \, 688 \, 776 \,
450 \, 913 \, 725 \, 462 \, 959 \, 925 \, 286 \, 301 \, 344 \, 474 \, 920 \, 318 \, 994 \, 753 \, 701 \, 941 \, 060 \, 148 \,
881 \, 573 \, 427 \, 303 \, 951 \, 969 \, 468 \, 229 \, 706 \, 399 \, 241 \, 390 \, 675 \, 879 \, 189 \, 573 \, 397 \, 083 \, 298 \, 234 \,
232 \, 002 \, 057 \, 590 \, 825 \, 541 \, 929 \, 196 \, 617 \, 798 \, 357 \, 864 \, 253 \, 415 \, 244 \, 900 \, 822 \, 598 \,
565 \, 148 \, 459 \, 722 \, 700 \, 786 \, 910 \, 964 \, 414 \, 143 \, 234 \, 585 \, 951 \, 686 \, 501 \, 951 \, 456 \, 012 \,
929 \, 058 \, 272 \, 386 \, 770 \, 627 \, 526 \, 673 \, 039 \, 661 \, 180 \, 847 \, 924 \, 218 \, 842 \, 419 \, 368 \, 978 \,
282 \, 454 \, 440 \, 860 \, 900 \, 662 \, 216 \, 576 \, 405 \, 681 \, 454 \, 036 \, 569 \, 302 \, 145 \, 779 \, 655 \, 295 \,
161 \, 024 \, 870 \, 390 \, 029 \, 412 \, 887 \, 939 \, 760 \, 085 \, 690 \, 088 \, 522 \, 580 \, 668 \, 382 \, 601 \, 362 \,
698 \, 591 \, 183 \, 043 \, 906 \, 726 \, 589 \, 771 \, 426 \, 474 \, 327 \, 106 \, 027 \, 446 \, 630 \, 110 \, 424 \, 304 \,
124 \, 378 \, 118 \, 465 \, 405 \, 405 \, 743 \, 664 \, 124 \, 241 \, 120 \, 045 \, 296 \, 239 \, 389 \, 519 \, 468 \, 482 \,
351 \, 202 \, 104 \, 266 \, 957 \, 141 \, 300 \, 679 \, 419 \, 724 \, 335 \, 696 \, 429 \, 700 \, 298 \, 358 \, 638 \, 780 \,
913 \, 615 \, 738 \, 551 \, 744 \, 801 \, 623 \, 131 \, 105 \, 809 \, 471 \, 972 \, 048 \, 150 \, 306 \, 016 \, 295 \, 179 \,
131 \, 035 \, 357 \, 607 \, 223 \, 358 \, 211 \, 005 \, 134 \, 001 \, 883 \, 185 \, 282 \, 488 \, 946 \, 730 \, 500 \, 335 \,
843 \, 269 \, 291 \, 572 \, 586 \, 763 \, 218 \, 825 \, 995 \, 118 \, 857 \, 751 \, 797 \, 398 \, 654 \, 362 \, 527 \, 314 \,
550 \, 465 \, 079 \, 665 \, 086 \, 098 \, 550 \, 853 \, 743 \, 078 \, 745 \, 496 \, 295 \, 016 \, 434 \, 317 \, 612 \, 040 \,
149 \, 655 \, 374 \, 590 \, 841 \, 804 \, 951 \, 466 \, 880 \, 939 \, 961 \, 763 \, 331 \, 490 \, 516 \, 798 \, 658 \, 974 \,
158 \, 057 \, 235 \, 405 \, 502 \, 012 \, 651 \, 675 \, 705 \, 822 \, 721 \, 872 \, 793 \, 215 \, 886 \, 863 \, 022 \, 494 \,
497 \, 302 \, 905 \, 269 \, 956 \, 369 \, 501 \, 159 \, 699 \, 301 \, 180 \, 259 \, 593 \, 349 \, 473 \, 092 \, 183 \, 890
\]
Since they need keys of length 4, 5, and 6 then we scroll the number \( M \) and see that the first 4 appears after the sequence 8 606 8. So Stella and Ben take 4 digits after the first 4 which are 2941. So, \( m_{1(4)} = 2941 \). Where the notation \( m_{s(t)} \) represent the \( s \)th key of length \( t \). The first 5 appears after the sequence 606 842 941 648 980 141 and it is followed by the digits 00 390 , so \( m_{1(5)} = 00390 \). If we continue we find \( m_{1(6)} = 684294 \), \( M_{2(4)} = 1648 \) and so on. It is easy to see that there is a large number of 4, 5 and 6’s in \( M \), so the number of keys is very large; in this case there are 5277 \( k \)’s.

Here is the full list of the keys of length 4:

2941, 1648, 8980, 1500, 333, 5091, 6295, 4474, 4749, 7492, 9203, 7537, 1060, 8881, 2730, 6822, 2320, 1929, 2534, 1524, 4900, 9008, 8459, 5972, 4141, 1414, 1432, 3234, 5859, 5601, 7924, 2188, 2419, 1936, 5444, 4408, 4086, 860, 568, 5403, 365, 5779, 8703, 1288, 3906, 2647, 7432, 3271, 4663, 6630, 2430, 3041, 1243, 3781, 6540, 540, 574, 3664, 1242, 2411, 1120, 5296, 6848, 8235, 2669, 1300, 1972, 3356, 2970, 4801, 8016, 7107, 8150, 18, 8894, 6730, 3269, 3625, 5504, 6507, 3078, 5496, 9629, 3431, 3176, 149, 9655, 5908, 1804, 9514, 6688, 9051, 1580, 550, 9449, 4973, 9730, 9473, 7309, 2031, 4605, 6051, 8555, 6443, 4390, 3901, 2444, 4441, 4147, 4173, 1738, 7242, 2876, 1573, 1874, 2866, 2876, 7722, 5858, 3936, 647, 7124, 6649, 9716, 6873, 7582, 7348, 8259, 8194, 269, 7021, 8437, 3751, 763, 3084, 6549, 9596, 9904, 1786, 7092, 4430, 4303, 3085, 7953, 25, 3876, 7154, 1431, 3126, 7039, 9255, 8572, 859, 8489, 8940, 304, 1065, 2639, 550, 7349, 9464, 6474, 7467, 6702, 711, 136, 3116, 7683, 3079, 1334, 4097, 974, 2860, 4808, 8089, 9799, 5068, 5511, 1817, 3179, 7055, 2175, 5451, 5152, 5051, 8046, 6170, 6221, 4473, 4734, 7346, 6763, 2805, 8536, 1276, 8892, 7017, 7302, 5098, 7496, 9677, 1955, 5810, 4904, 9040, 228, 8292, 4054, 543, 3057, 5980, 866, 2335, 2575, 3404, 439, 3901, 9956, 8529, 7285, 9640, 904, 1527, 7914, 3842, 2825, 6791, 3587, 633, 2052, 6997, 2844, 4844, 8427, 2578, 9536, 9353, 5726, 6070, 9708, 2117, 2791, 7958, 9324, 9900, 5725, 3533, 5570,
8 Conclusions

We hope to have left the reader with the understanding that the LSA algorithm can be used with different combinations of cyclic and non-cyclic groups. This means that this algorithm can be used in either symmetric or asymmetric modes of encryption. It is also of great benefit to the users of this algorithm that we can design systems of varying computational complexity and security by using different groups, encrypting $\Sigma$, and various other methods that make a group discovery by the cryptanalyst meaningless. In fact, we believe that exchanging keys between each character gives the LSA algorithm an equivalent time complexity to that of a one-time-pad.
In closing, the authors of this paper sincerely hope that if this bit of mathematics is found to be a useful tool, then humanity will use it to find ways to improve our standing in nature.
References


