Lucente Stabile Atkins (LSA) Cryptosystem

Francesco Lucente Stabile
marchesedp@gmail.com
(617) 785-7481

Carey Patrick Atkins
carey.atkins@gmail.com
(781) 771-2010

Abstract

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 a specific case of 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.

1 Introduction

The LSA algorithm is an original algorithm that serves the purpose of sharing secret information between two or more parties. Unlike most modern asymmetric encryption algorithms, LSA does not rely on the prime factorization problem and instead uses results brought upon by John Wilson and Carl Friedrich Gauss. That is, LSA's foundation stands solidly on the proof of Gauss's Generalization of Wilson's Theorem.

It is hoped by the authors of this paper that this algorithm has the potential to add to humanities cryptographic toolkit and maybe fix some forthcoming problems along the way. First and foremost, it is a growing concern that quantum computers will render encryption algorithms based on the prime factorization problem irrelevant. Algorithms such as RSA and ELGamal rely on harsh (and yet unproven) time complexities and

the computational in-feasibility of classical computers to guess the correct prime numbers p and q that make RSA's modulus n [3]. But in-feasible is not impossible because computers, classical or quantum, need only check prime numbers up to the square root of n.

As we will demonstrate, one of the main strengths in LSA is the fact that there are an infinite number of groups, both cyclic and not. Thus, unlike RSA, there is no n to find and no p and q to multiply together to find it.

Another strength of LSA is its ability to use smaller numbers than the ever increasing primes that are needed to keep existing algorithms secure. This will increase the efficiency of any organization that stores or computes large amounts of encrypted data by decreasing CPU cycles and reducing power consumption.

Above all, the strength of LSA resides in its simplicity.

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 an operation. More specifically, it is a set of elements for which, when the elements are operated with a specific operation (in our case modular multiplication or modular addition) the following properties hold:

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

- *Transitivity:* If $*$ is an 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 of the group a^{-1} such that $a * a^{-1} = e$, where e is the forementioned 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 (we assume that the reader is familiar with modular arithmetic). By convention, this group can be referred to symbolically as $U(n)$. For example, for $n = 12$ we have that $U(12) =< 1, 5, 7, 11 >$ is a group if operated with multiplication modulo 12. The proof that $U(n)$ is indeed a group [1] is beyond the scope of this paper, but is widely available for those interested.

The main question pertaining to any group that will be used for LSA encryption is whether or not the group is cyclic. A group is cyclic whenever it can be generated by operating an element with itself. Hence $U(n)$ is cyclic if $U(n) =< a^0, a^1, a^2...a^{\phi(n)-1} >$, where $\phi(n)$ (known in number theory as Euler's Tuotient function) represents the number of integers coprime with n, which is also the number of elements of the group $U(n)$.

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 specific case of 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, and $U(n)$ is cyclic, then by multiplying all the elements of the group together modulo n, we generate an element congruent to -1 modulo n.

So, if $U(n) =< a >=< 1, a, a^2, a^3, ..., a^{\phi(n)-1} >$, then

$$a^0 \cdot a^1 \cdot a^2 \cdot a^3 \cdots a^l \cdots a^{\phi(n)-1} \equiv -1 \mod n$$

A proof of the theorem is given in the Appendix of this paper.

If $U(n)$ is not cyclic, then

$$a^0 \cdot a^1 \cdot a^2 \cdot a^3 \cdots a^l \cdots a^{\phi(n)-1} \equiv 1 \mod n.$$

2.3 Key Exchange

Due to the nature of the IACR, we will assume that the reader is familiar with the concept of key exchanges.

3 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 Stella, Ben, and Joe respectively. They perform the following steps:

1. Stella and Ben begin by performing a key exchange that generates a shared key $k \in \mathbb{R}$ that is only known to Stella and Ben.

2. Next, k will be used as a reference by both parties to find an $n \in \mathbb{N}$ that satisfies the following properties:

   I) $k < n$.
   II) $n = p^t$, $n = 2 \cdot p^t$ where p is an odd prime number and $t \in \mathbb{N}$.
   III) For any $\beta \in \mathbb{Z}$ that satisfies I) and II) it must be the case that $n \leq \beta$.

3. Assuming that the above conditions have been met, it follows that $U(n)$ is a cyclic group of all of the integers relatively prime to n and less than n [2]. Since $U(n)$ is cyclic, then

$$U(n) =< a >=< a^0, a^1, a^2, ..., a^{\phi(n)-1} > .$$

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

4. Stella chooses an element of the group to represent the plain text of her message, ϵ_h.

5. To encrypt, Stella multiplies each of the elements up to ϵ_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. Stella publicly sends c to Ben.

7. Ben receives c and multiplies c with all the 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 $U(n)$ is cyclic). At this point Ben knows that the next

element of the group yet to be multiplied, ϵ_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.
Thus, before Stella 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 Σ. Then she publicly sends the tuple $C = (c, \Sigma)$, which becomes the ciphertext.

†† When Ben 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.

3.1 Example

Here is an example of the LSA operated within a cyclic 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 Stella wants to secretly share the plaintext '7' with Ben:

1. The key exchange algorithm generates $k = 53$.
2. Stella and Ben independently follow the LSA algorithm and conclude that the next useful integer is 54 because it satisfies the following properties:

   I) $53 < 54$
   II) $54 = 2 \cdot 3^3$.

Hence $U(54)$ is cyclic.

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

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

5. Stella performs $1 \cdot 5 \cdot 7 \cdot 11 \cdot 13 \cdot 17 \cdot 19 = 1,616,615 \mod 54 \equiv 17 = c$.

† Stella multiplies c by each element in descending order to find the value of Σ in the following way:

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

Since $c = 17$ and $\Sigma = 1$ (because Stella generated only one additional element congruent to $-1 \mod n$), then Stella publicly sends $C = (17, 1)$ to Ben.

6. †† Ben 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:

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

Since Ben 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, Ben has the plaintext '7'.

4 Use of Multiplicative Non-Cyclic Groups

Gauss's Generalization of Wilson's Theorem, for non-cyclic groups, states that if we multiply all the elements of $U(n)$, when $U(n)$ is not cyclic, we generate an element congruent to 1 modulo n (proof omitted).

This fact can be used in an analogous way as was used in cyclic groups. In fact, it will be sufficient for Stella and Ben to generate a non-cyclic $U(n)$ simply by choosing n such that $n \neq p^t$ and $n \neq 2 \cdot p^t$ where p is an odd prime number and $t \in \mathbb{N}$. Then, Stella and Ben perform the steps of the algorithm exactly the same with the exception of substituting 1 in the place of -1 modulo n.

5 Description for Use of Additive Groups

First, note that the group of all the integers less then n, equipped with addition modulo n is indeed a group [2].

The LSA can also be performed using additive groups in the following way:

Let the key k be used to generate $n \in \mathbb{N}$ where $\mathbb{Z}_n$ is the group of the positive integers less than n including 0, closed under addition modulo n. Then, Stella picks an element of the group, say l, that she wishes to encrypt and sends $c \equiv 0+1+2+...+l \mod n$ to Ben where c represents the cipher text that is sent in the open.

Ben then decrypts the message by performing the same steps as Stella, meaning that he performs $0+1+2+...+h$ until he finds that $0+1+2+...h \equiv c \mod n$. Here, he sees that $h = l$ and the plaintext is decrypted. As in the multiplicative method, since $\mathbb{Z}_n$ is a group, it is possible that the addition of the elements of the group will generate more than one element congruent to c modulo n. And, as before, all of these elements will be listed as Σ and communicated in a tuple such as $C = (c, \Sigma)$, where C will be public.

5.1 Example Using Additive Groups

Assume that from the shared key we generate the integer 5, thus Stella and Ben list the elements of $\mathbb{Z}_5$ in ascending order, namely $< 0, 1, 2, 3, 4 >$. Assume that Stella wants to send Ben the integer 3; then she performs $0+1+2+3 = 6 \equiv 1 \mod 5$, thus she find that $c = 1$. Next, she will check that

$$\begin{aligned} 0+1 = 1 \equiv 1 \mod 5 &\rightarrow (\text{This is one } \Sigma) \\ 0+1+2 = 3 \equiv 3 \mod 5. & \end{aligned}$$

At this point, she found the value of Σ and she sends $C = (1, 1)$ to Ben. From here, Ben performs the same steps until he finds the second element congruent to 1 modulo 5 in the following way:

$$\begin{aligned} 0 + 1 = 1 \equiv 1 \mod 5 &\rightarrow (\text{ one } \Sigma) \\ 0 + 1 + 2 = 3 \equiv 3 \mod 5 & \\ 0 + 1 + 2 + 3 = 6 \equiv 1 \mod 5. & \end{aligned}$$

Thus, Ben knows that $c = 3$, which decrypts the ciphertext.

6 Description for Use of Multiplicative Groups without the use of the Gauss's Generalization of Wilson's Theorem

Finally, the LSA can be used in multiplicative groups $U(n)$ without making use of Gauss's Generalization of Wilson's Theorem. To show this, let the shared key k be used to generate an integer n such that $U(n) =< \epsilon_1, \epsilon_2, \epsilon_3, ..., \epsilon_{\phi(n)} >$ and note that, as before, this is the group of all the integers less then n and coprime to n, where $\epsilon_i < \epsilon_j$ when $i < j$.

Next, Stella picks an element of the group, say ϵ_h, according to her needs, and sends the ciphertext c to Ben, such that $\epsilon_1 \cdot \epsilon_2 \cdots \epsilon_h \mod n \equiv c$.

From here, Ben decrypts the message by performing the same steps as Stella. This means that he begins multiplying $\epsilon_1 \cdot \epsilon_2 \cdot \epsilon_3 \cdot \cdot\cdot$ until finds the element ϵ_h. He will know when he finds it because $\epsilon_1 \cdot \epsilon_2 \cdots \epsilon_h \mod n \equiv c$.

As in the additive method, since $U(n)$ is a group, it is possible that the multiplication of the elements of the group will generate more than one element congruent to c modulo n, hence all of these elements will be listed

as Σ and communicated in a tuple such as $C = (c, \Sigma)$, where C will be public.

6.1 Example

Beginning with the key exchange algorithm, we generate the integer $n = 18$. Stella and Ben list the elements of $U(18)$ in ascending order, namely $< 1, 5, 7, 11, 13, 17 >$. Assume that Stella wants to send Ben the integer 13. Then, she performs $1 \cdot 5 \cdot 7 \cdot 11 \cdot 13 = 5005 \equiv 1 \mod 18$, finding that $c = 1$. Next, she checks

$$\begin{aligned} 1 &\equiv 1 \mod 18 \rightarrow (\text{this is one } \Sigma) \\ 1 \cdot 5 &\equiv 5 \mod 18 \\ 1 \cdot 5 \cdot 7 &\equiv 17 \mod 18 \\ 1 \cdot 5 \cdot 7 \cdot 11 &\equiv 7 \mod 18 \end{aligned}$$

to find the value of Σ and she sends $C = (1, 1)$ to Ben.

Ben then performs the same steps until he finds the second element congruent to 1 modulo 18, in the following way:

$$\begin{aligned} 1 &\equiv 1 \mod 18 \rightarrow (\text{one } \Sigma) \\ 1 \cdot 5 &\equiv 5 \mod 18 \\ 1 \cdot 5 \cdot 7 &\equiv 17 \mod 18 \\ 1 \cdot 5 \cdot 7 \cdot 11 &\equiv 7 \mod 18 \\ 1 \cdot 5 \cdot 7 \cdot 11 \cdot 13 &\equiv 1 \mod 18 \rightarrow (\Sigma + 1) \end{aligned}$$

At the conclusion, Ben finds that the plaintext is 13.

7 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 ψ 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 $U(9)$, ψ maps the elements as follows:

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

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 $1 \cdot 2 \mod 9 \equiv 2$, or $1 \cdot 2 \cdot 4 \cdot 5 \cdot 7 \mod 9 \equiv 1$. So she can send f as either $c = 1$ or $c = 2$ as the first component of C.

8 Enhancing Security (Using LSA as a 'one-time-pad')

We begin this section by revisiting the issue of using a one-time, pre-shared key, for each character of the message. The authors of this paper believe that this step will ensure that any kind of frequency analysis will fail the cryptanalyist attempt to learn the group used to encrypt. This, in-turn,

allows for the use of groups as small as the order of the character set that the sender wishes to encrypt.

Here we remind the reader that a **one-time-pad** is a method of encryption that uses a one-time, pre-shared key, that cannot be cracked other than to use the technique known in the field of cryptography as a "lucky guess".

Since the key exchange process might be a computationally heavy task, we suggest a modification to the LSA algorithm to generate variable length keys. Further research on this method called Lucente Stabile Atkins Leclerc Key Exchange Algorithm (LSAL), will soon be available for review. Here we will show that one iteration of the LSAL can be used to produce sets of keys of remarkable order. For instance, by choosing a single, arbitrary element, from a group of approximately 10,000 elements, LSAL can independently produce a key set with an approximate order of 20,000 keys of variable length. Note that different key sets, of similar order, can be generated by each element of the group.

The Lucente Stabile Atkins Algorithm can be implemented in several ways, that have varying degrees of security, that each come with different levels of computational complexity. The use of which will be determined by the needs of the sender/receiver.

Obviously, additional steps can be done to protect the information shared; for example, since the integer Σ is sent publicly, Stella may wish to hide its value. However, considering the nature of the audience, we will not suggest any security precautions, unless, to best of our knowledge, they are a novelty.

The following steps imply the use of mathematical groups, so we will present them:

1. In order to avoid revealing the size of the group that could be guessed

by the public sharing of Σ, the following procedure can be performed:
Let Ω be the total number of elements congruent to -1 modulo n that are generated by multiplying all the elements of the group in descending order. Then, Stella can send $(c, k\Omega + \Sigma)$ to Ben.
Since Ben knows all the elements of the group he can easily derive Ω by multiplying all the elements of the group one by one and checking their congruence modulo n each iteration, thus he can find Σ by performing $k\Omega + \Sigma \equiv \Sigma \mod k\Omega$.
This procedure allows the sender to publish a larger integer that substitutes Σ, even when working within a group of small order. Moreover, this step gets its strength due to the fact that n is hard to retrieve from k.

2. If $n = k$, any successful attack on n would result in the discovery of k, which in turn could be used to reveal Σ.

3. It is recommended that, when choosing a group $U(n)$, n not be prime or else the product of the group elements may be a factorial, and thus easily recognizable by a cryptanalyst. This can be easily ensured by setting $t > 1$.

4. Recall that for any integer k, $k + 1$ is coprime to k. In our case, for any non-prime n, by the definition $U(n)$, if k is an element of $U(n)$, then $k + 1$ is not. Now, observe the ciphertext $C = (c, \Sigma)$. Note that c, which is an element of $U(n)$ can reveal some clue about the nature of n. In fact, if c is even, and c is coprime to n, then n must be odd. Similarly if c is odd, then n is more likely to be even [1].
To avoid revealing such information it is enough for Stella to send $C = (c + 1, \Sigma)$, where $c + 1$ is not an element of $U(n)$.
When Ben receives the ciphertext, he will discover that the $c + 1$ is not in the list of its elements, hence he will simply need to subtract 1 to reveal c.

9 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 Σ, 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.

10 Appendix

10.1 Gauss's Generalization of Wilson's Theorem

The following is a proof of a specific case of Gauss's Generalization of Wilson's Theorem [2].

Proposition: If a group $U(n)$ is cyclic [2], then all of the elements of the group, multiplied together, will result in an element that is congruent to -1 modulo n, where $U(n)$ is the set of all the integers less then n, that are relatively prime to n, and equipped with multiplication modulo n.

Theorem 10.1 *Let $U(n) =< a >$ be a cyclic group of all of the integers relatively prime to n and less then n. Assume that $U(n)$ is generated by the element a, then $a \cdot a^2 \cdot a^3 \cdots a^{\phi(n)-1} \equiv -1 \pmod{n}$.*

Proof: Let $U(n) =< a >$ be a cyclic group of integers coprime to n and less then n, so that $U(n) =< a >=< 1, a, a^2, a^3, ..., a^{\phi(n)-1} >$, and that $gcd(a^j, n) = 1$ for all $j \in \mathbb{N}$.

First observe that $n - 1 \equiv -1 \pmod{n}$ and that $(n-1)^2 = n(n-2) + 1 \equiv 1 \pmod{n}$. It follows that $n-1$ is its own inverse. Also, since the inverse is unique, then $n-1$ has no other inverse than itself.

Moreover, $n-1$ is the only non-identity element of the group whose inverse is itself. In fact, let b be an arbitrary element of $U(n)$, such that $b^2 \equiv 1 \pmod{n}$, it follows that $b^2 - 1 \equiv 0 \pmod{n}$. This means that $n|(b-1)$ or $n|(b+1)$. Now, if $b \neq 1$, and $b \neq (n-1)$, then $3 \leq b+1 \leq n-1$, and $1 \leq b-1 \leq n-3$. Note that no integer in the group between 1 and $n-1$ is divisible by n, hence b does not exist.

Now, since $U(n)$ is a group, each element of $U(n)$ has a unique and distinct inverse that can be found within the elements of the group, except for $n-1$. Thus $(a^i)(a^i)^{-1} \equiv 1 \pmod{n}$, where a^i and $(a^i)^{-1}$ are distinct elements, for

all $i \neq 0$, except for an integer l for which $n - 1 = a^l$.

So consider

$$1 \cdot a \cdot a^2 \cdot a^3 \cdots a^l \cdots a^{\phi(n)-1}$$

We know that each element multiplied with its own unique and distinct inverse generates an element that is congruent to 1 modulo n, except for a^l.
So we have that

$$\begin{aligned} 1 \cdot a \cdot a^2 \cdots a^l \cdots a^{\phi(n)-1} &\equiv 1 \cdot aa^{-1} \cdot a^2 \left(a^2\right)^{-1} \cdots a^l \cdots a^{\phi(n)-1} \left(a^{\phi(n)-1}\right)^{-1} \\ &\equiv 1 \cdot 1 \cdot 1 \cdots -1 \cdots 1 \\ &\equiv -1 \,(\mathrm{mod}\, n). \end{aligned}$$

■

11 Prototype Source Code

Python prototype implementation written by

Colby Leclerc
colby@colby.io
(978) 708-1050

`<https://github.com/ColbyLeclerc/lsa>`

(Patent pending)

References

[1] David Burton, *M. Elementary Number Theory*, 7^{th} ed., McGraw-Hill Education (India) Private Limited, 2016.

[2] Joseph A. Gallian *Contemporary Abstract Algebra*, 8^{th} ed., Cengage Learning, 2016.

[3] Wade Trappe and Lawrence C. Washington, *Introduction to Cryptography: with Coding Theory*, Pearson Education, 2006.