# CRIPTOSISTEMA SOBRE UNIDADES EN ANILLOS DE GRUPO

YADIRA BELEN QUITIAQUEZ GOYES

FACULTAD DE CIENCIAS EXACTAS Y NATURALES
DEPARTAMENTO DE MATEMÁTICAS Y ESTADÍSTICA
UNIVERSIDAD DE NARIÑO
SAN JUAN DE PASTO
2018

# CRIPTOSISTEMA SOBRE UNIDADES EN ANILLOS DE GRUPO

YADIRA BELEN QUITIAQUEZ GOYES

Trabajo presentado como requisito parcial para optar al título de
Licenciado en Matemáticas

Asesor
John Hermes Castillo Gómez
Doctor en Matemáticas

FACULTAD DE CIENCIAS EXACTAS Y NATURALES
DEPARTAMENTO DE MATEMÁTICAS Y ESTADÍSTICA
UNIVERSIDAD DE NARIÑO
SAN JUAN DE PASTO
2018

# Nota de Responsabilidad

Todas las ideas y conclusiones aportadas en el siguiente trabajo son responsabilidad exclusiva de los autores.

Artículo $1^{ro}$ del Acuerdo No. 324 de octubre 11 de 1966 emanado por el Honorable Consejo Directivo de la Universidad de Nariño.

Nota de Aceptación

John Hermes Castillo Gómez

**Director de Tesis**

Alexander Holguín Villa

**Jurado**

Fernando Andrés Benavides Agredo

**Jurado**

San Juan de Pasto, Mayo 2 de 2018

*Este trabajo está dedicado a:*
*Mis padres Angela Goyes y Nelson Quitiaquez como reconocimiento de su amor, valiosos consejos y apoyo incondicional.*
*Yadira*

# Agradecimientos

La vida se encuentra llena de sueños y a pesar que cada uno se convierte en un reto, con el debido esfuerzo será superado para finalmente alcanzar dicha meta, en esta ocasión he cumplido con terminar la redacción del trabajo de grado para optar al título de Licenciada en Matemáticas, el cual me enseñó en el trascurso de esta etapa que más allá de ser un reto o un sueño, se convirtió en una base, no solo para la comprensión de las matemáticas y la implicación del significado de ser docente, sino para lo que concierne en la vida misma, el ser persona y en mi futuro. Le agradezco infinitamente a mi familia en especial a mi padre y madre, a quienes les debo cada uno de sus esfuerzos y quienes con su comprensión y apoyo incondicional me han permitido que alcance cada logro. Además agradezco a mi asesor Dr. John Castillo por la orientación, el seguimiento y la supervisión continua de este trabajo, pero sobre todo por la dedicación, la motivación y apoyo a lo largo de este tiempo. Finalmente le agradezco a mis amigas Monica Díaz y Melissa Rojas por caminar junto a mi en todo el proceso de esta carrera universitaria.

Yadira Belen Quitiaquez Goyes.

# Resumen

Dado un anillo $R$ con identidad y un grupo $G$ se puede construir el anillo de grupo $RG$ donde sus elementos son combinaciones lineales formales de los elementos de $G$ con coeficientes en al anillo $R$. Un elemento $u \in RG$ se denomina unidad si existe un elemento $v \in RG$ tal que $uv = vu = 1$. En las últimas décadas los anillos de grupo son estructuras algebraicas a las cuales se les han encontrado diferentes aplicaciones, por ejemplo en teoría de códigos y en *criptografía*. La criptografía es indispensable en la seguridad electrónica, esta se encarga, precisamente, de cifrar o descifrar mensajes para evitar que su contenido pueda ser leído por un tercero no autorizado; es decir, la generación de códigos y algoritmos de cifrado buscan proteger la información real. El criptosistema RSA es uno de los criptosistema más conocidos y fáciles de emplear. Recientemente, primero T. Hurley y B. Hurley y luego T. Hanoymak y Ö. Küsmüş realizaron investigaciones encaminadas a la construcción de un sistema criptográfico basado en RSA utilizando unidades en anillos de grupo. En este trabajo se estudia la implementación de los criptosistemas RSA, criptosistema con unidades en anillos de grupo y el criptosistema RSA con unidades en anillos de grupo, esto se hace mediante el software computacional `SAGE`. Finalmente se presenta los algoritmos de estos criptosistemas implementados en `SAGE`.

# Abstract

Let $R$ be a ring with unity and $G$ a group. and $G$ a group. It is possible to construct the group ring $RG$ where its elements are formal linear combinations of elements in $G$ whit coefficients in $R$. An element $u \in RG$ is called a unit in $RG$ if there exists $v \in RG$ such that $uv = vu = 1$. Over the last decades, the group rings are algebraic structures to which different applications have been found, for example in coding theory and *cryptography*. Cryptography is responsible essentially to encrypt and decrypt messages to prevent its content from being read by an unauthorized third person; that is, the generation of codes and encryption algorithms seek to protect the real information. The RSA cryptosystem is one of the best-known and easiest tools in cryptography. Recently, first T. Hurley and B. Hurley and next T. Hanoymak and Ö. Küsmüş conducted research aimed at the construction of a cryptographic system based on RSA using units in group rings. In this work we study the implementation of RSA cryptosystems, cryptosystem with units in group rings and the RSA cryptosystem with units in group rings, this is done by means of the free open-source mathematics software system `SAGE`. Finally, the algorithms of these cryptosystems implemented in `SAGE` are presented.

# Índice general

**Introducción** IX

**1. Preliminares** 1
- 1.1. Teoría de congruencias . . . 1
- 1.2. Teoría de grupos, anillos y cuerpos . . . 2
- 1.3. Módulos y álgebras . . . 4
- 1.4. Anillo de grupo . . . 5
- 1.5. Exponenciación modular binaria . . . 11
- 1.6. Criptografía . . . 12
  - 1.6.1. Criptosistema . . . 12
  - 1.6.2. Criptografía simétrica . . . 13
  - 1.6.3. Criptografía asimétrica . . . 14
  - 1.6.4. Firma digital . . . 14
- 1.7. Problema de factorización entera . . . 14
- 1.8. Problema de logaritmo discreto . . . 15
- 1.9. Código ASCII . . . 15

**2. Unidades en anillos de grupo** 17
- 2.1. Introducción . . . 17
- 2.2. Unidades triviales . . . 18
- 2.3. Unidades unipotentes . . . 18
- 2.4. Unidades bicíclicas . . . 19
- 2.5. Unidades cíclicas de Bass . . . 19
- 2.6. Métodos para construir unidades . . . 20

**3. Criptosistema RSA** 24
- 3.1. Introducción . . . 24
- 3.2. Criptosistema RSA . . . 25
  - 3.2.1. Cifrado RSA en SAGE . . . 25
  - 3.2.2. Descifrado RSA en SAGE . . . 27

**4. Criptosistema y unidades en anillos de grupo** 28
- 4.1. Método . . . 28
  - 4.1.1. Cifrado de bloque de Hurley . . . 31
  - 4.1.2. Una extensión al cifrado de bloque de Hurley . . . 33

4.2. Criptosistema RSA y unidades de anillos de grupo . . . . . 35
4.3. Criptosistema alternativo RSA con unidades en $\mathbb{Z}_nG$ . . . . . 37
4.4. Ejemplos implementados en SAGE. . . . . 40
4.4.1. Cifrado con unidades en $RG$ . . . . . 40
4.4.2. Cifrado y descifrado RSA con unidades en $RG$ . . . . . 42
4.4.3. Cifrado y descifrado alternativo RSA con unidades en $\mathbb{Z}_nG$ . . . . . 42

**5. Conclusiones** **44**

**A. Algoritmos en SAGE** **45**

**Referencias** **55**

# Introducción

La criptografía es primordial en los sistemas de seguridad electrónica. Es posible ver que hay muchos conceptos tales como criptografía de clave pública y clave privada, estandartes de cifrado de datos, protocolos de intercambio de claves, firmas digitales entre otros ver [1, 2]. En términos de seguridad, los sistemas de clave privada o también llamados cifrado de clave simétrica son inadecuados, puesto que surge el problema de comunicarse la clave de manera secreta y segura. Probablemente uno de los sistemas criptográficos más conocidos de clave pública es el sistema RSA, nombrado así por las iniciales de sus creadores Ronald Rivest, Adi Shamir y Leonard Adleman. Este se basa principalmente en la estructura de grupos Abelianos finitos y trabaja en $\mathcal{U}(\mathbb{Z}_n)$, las unidades de los enteros módulo $n$, donde $n$ es el producto de dos primos grandes $p$ y $q$. La seguridad de RSA depende de la dificultad de factorizar $n$.

El método criptográfico de unidades en anillos de grupo, $\mathcal{U}(RG)$, consiste en utilizar las unidades en el anillo de grupo $RG$, para el cifrado y descifrado. Para esto se considera una unidad $u$ y su inverso multiplicativo $u^{-1}$ de $RG$ para cifrar y descifrar el mensaje $m$ multiplicado por $u$ y $u^{-1}$ respectivamente. El método de anillo de grupo se puede combinar con el criptosistema RSA. Este trabajo se enfoca en mostrar de manera detallada como aplicar estos métodos para cifrar mensajes ejemplificandolos en el sistema computacional `SAGE`.

Este trabajo se ha organizado en cuatro capítulos de la siguiente manera

En el Capítulo 1 se dan algunas definiciones y teoremas que serán de utilidad para el desarrollo y comprensión de los temas que se abordan en los capítulos posteriores, en el Capítulo 2 se exponen algunos conceptos y métodos que sirven para la construcción de unidades en anillos de grupo. En el Capítulo 3, se explica la implementación del sistema RSA y una manera de ejecutarlo en el software de álgebra computacional `SAGE`. Finalmente, en el Capítulo 4 se exponen el método de cifrado con unidades en anillo de grupo, el cual incluye el método de cifrado de bloque y el método alternativo de cifrado de bloque. Además, se enseñan dos maneras de combinar el sistema RSA y las unidades en anillo de grupo.

Cabe resaltar que este trabajo contiene el Apéndice A, en el cual se incluyen los algoritmos que se realizaron en el desarrollo de este trabajo y su respectiva implementación en `SAGE`.

# Capítulo 1

# Preliminares

En este capítulo se incluyen algunos conceptos y resultados básicos, necesarios para el entendimiento de la teoría que se presentará durante la realización del trabajo propuesto. Dado que algunos resultados son conocidos, en este escrito no se presentan sus demostraciones. Sin embargo, si es necesario en algunos citar el material de donde se puede encontrar mayor información.

## 1.1. Teoría de congruencias

**Definición 1.1.** Sea $n$ un entero fijo diferente de cero. Dos enteros $a$ y $b$ se dice que son congruentes módulo $n$. Si $n$ divide a $a-b$; es decir si existe un entero $k$ tal que $a-b=kn$. Esto se denota por $a \equiv b \mod n$.

**Teorema 1.1.** *Sean $a$ y $b$ enteros y $n$ un entero no nulo. Entonces $a \equiv b \mod n$ si y solo si $a$ y $b$ dejan el mismo residuo cuando se dividen por $n$.*

A continuación se presentan algunas de las propiedades de las congruencias.

**Teorema 1.2** (Propiedades de congruencias)**.** *Sean $n \neq 0$ y $a, b, c$, enteros cualesquiera. Entonces se tienen las siguientes propiedades.*

1. $a \equiv a \mod n$,
2. *Si* $a \equiv b \mod n$, *entonces* $b \equiv a \mod n$,
3. *Si* $a \equiv b \mod n$ *y* $b \equiv c \mod n$, *entonces* $a \equiv c \mod n$,
4. *Si* $a \equiv b \mod n$ *y* $c \equiv d \mod n$, *entonces* $a+c \equiv b+d \mod n$ *y* $ac \equiv bd \mod n$,
5. *Si* $a \equiv b \mod n$, *entonces* $a^k \equiv b^k \mod n$ *para cualquier entero positivo* $k$.

**Observación 1.1.** Dados dos enteros positivos $a$ y $b$, con $\mathrm{mcd}(a,b)$ se denota el máximo común divisor de $a$ y $b$.

**Definición 1.2.** Se dice que dos números $a$ y $b$ son primos relativos si $\text{mcd}(a, b) = 1$.

**Teorema 1.3.** *Sean $a, m$ enteros con $m > 1$. Si $a$ y $m$ son primos relativos, entonces existe un único entero $s$ tal que $0 < s < m$ y $as \equiv 1$ mód $m$.*

**Definición 1.3** (Función de Euler)**.** Para $n$ un entero mayor que 1, con $\phi(n)$ se denota el número de enteros positivos menores o iguales que $n$ que son primos relativos con $n$.

**Teorema 1.4.** *Sea $n = pq$, con $p$ y $q$ primos. Entonces*

$$\phi(n) = \phi(pq) = \phi(p)\phi(q) = (p-1)(q-1).$$

Los dos teoremas anteriores se utilizarán en la construcción del Criptosistema RSA el cual se presenta en el Capítulo 3.

**Teorema 1.5** (Pequeño Teorema de Fermat)**.** *Si $p$ es primo y $a \in \mathbb{Z}$ tal que $p \nmid a$, entonces $a^{p-1} \equiv 1$ mód $p$.*

**Teorema 1.6** (Teorema de Euler)**.** *Si $a$ es un entero primo relativo con $n$, entonces $a^{\phi(n)} \equiv 1$ mód $n$.*

**Teorema 1.7.** *Sean $p$ y $q$ números primos distintos. Si $c^d \equiv m$ mód $p$ y $c^d \equiv m$ mód $q$, entonces $c^d \equiv m$ mód $pq$.*

## 1.2. Teoría de grupos, anillos y cuerpos

**Definición 1.4** (Grupo)**.** Un grupo $\langle G, \cdot \rangle$, es un conjunto no vacío $G$ junto con una operación binaria $\cdot$, tal que para todo $a, b, c \in G$ se cumplen las siguientes propiedades

1. **Asociativa:** $a \cdot (b \cdot c) = (a \cdot b) \cdot c$,
2. **Identidad (elemento neutro):** Existe un elemento $e$ (identidad) en $G$ tal que $a \cdot e = e \cdot a = a$, $\forall a \in G$,
3. **Inverso:** Para cada elemento $a$ en $G$, existe un elemento $b$ en $G$ (llamado inverso de $a$) tal que $a \cdot b = b \cdot a = e$,

**Observación 1.2.** Usualmente se escribe $ab$ en lugar de $a \cdot b$. La operación binaria en un grupo $G$, no necesariamente es multiplicativa, también puede ser aditiva y se denotada por $\langle G, + \rangle$. Cuando $ab = ba$ $\forall a, b \in G$ decimos que $G$ es un **grupo Abeliano** (conmutativo).

Como ejemplos de lo anteriormente definido se tiene que, $\langle \mathbb{Z}, + \rangle$ (los enteros bajo la suma) forman un grupo Abeliano, $\langle \mathbb{R}, + \rangle$ (reales bajo la suma), forman un grupo Abeliano y $\langle \mathbb{R}[x], + \rangle$ (conjunto de polinomios con coeficientes reales) también forman un grupo Abeliano bajo la suma usual de polinomios. Cabe resaltar que no todos los grupos son Abelianos, por ejemplo $GL(2, R)$ (el conjunto de matrices $2x2$ con entradas reales y determinante diferente de cero) no es Abeliano.

**Definición 1.5** (Orden de un grupo)**.** El número de elementos de un grupo (finito o infinito) se llama orden del grupo. Se usa $|G|$ para denotar el orden de $G$.

**Definición 1.6** (Subgrupo)**.** Un subconjunto $H$ no vacío de un grupo $G$ que es un grupo bajo la operación de $G$, se dice que es un **subgrupo** de $G$ y se denota con $H \leq G$.

De lo anterior, $\langle \mathbb{Z}, + \rangle$ y $\langle \mathbb{R}, + \rangle$ forman grupos Abelianos bajo la suma y como $\mathbb{Z} \subseteq \mathbb{R}$, entonces $\langle \mathbb{Z}, + \rangle \leq \langle \mathbb{R}, + \rangle$.

**Definición 1.7** (Grupo cíclico)**.** Sea $G$ un grupo y $a \in G$. El menor entero positivo $n$ tal que $a^n = 1$, se denomina **el orden** de $a$ en $G$ y se denota con $o(a)$. Si no existe tal entero se dice que el elemento es de orden infinito o se escribe $o(a) = \infty$. Se dice que $G$ es un **grupo cíclico** si existe un elemento $a \in G$ tal que $G = \{a^n : n \in \mathbb{Z}\}$, donde el elemento $a$ se llama generador de $G$. Se dice que $G$ es un grupo cíclico generado por $a$ lo que se denota con $G = \langle a \rangle$.

**Definición 1.8** (Anillo)**.** Un **anillo** es un conjunto $R$ con dos operaciones binarias, $\langle R, +, \cdot \rangle$, tal que

1. $\langle R, + \rangle$ es un grupo Abeliano y para todo $a, b, c \in R$ se tiene que,
2. $a(bc) = (ab)c$,
3. $a(b + c) = ab + ac$ y $(b + c)a = ba + ca$.

**Observación 1.3.** Un anillo no necesariamente tiene una identidad multiplicativa,si existe se dice que es un anillo con identidad. Si $ab = ba$ para todo $a, b \in R$ se dice que $R$ es un anillo conmutativo.

**Definición 1.9.** Sea $R$ un anillo con identidad 1, Si un elemento $a \in R$ tiene inverso multiplicativo, denotado por $a^{-1}$ se dice que $a$ es una unidad. Donde

$$aa^{-1} = a^{-1}a = 1.$$

**Definición 1.10** (Subanillo)**.** Un subconjunto no vacío $S$ de un anillo $R$ es un **subanillo** de $R$, si $S$ es un anillo bajo las operaciones de $R$.

**Definición 1.11** (Cuerpo)**.** Un **cuerpo** o también llamado campo es un anillo conmutativo con identidad donde cada elemento diferente de cero es una unidad.

Los números racionales $\mathbb{Q}$, los números reales $\mathbb{R}$ y los números complejos $\mathbb{C}$ son ejemplos de cuerpos. Otros ejemplos de cuerpos y más ejemplos de grupos y anillos se encuentran en [4].

## 1.3. Módulos y álgebras

A continuación se presenta el concepto de módulo sobre un anillo, el cual es una extensión de espacio vectorial. Para esto se considera un anillo $R$, el cual no necesariamente es conmutativo y no necesariamente tiene identidad.

**Definición 1.12** (Módulo)**.** Un $R$-módulo izquierdo o un módulo izquierdo sobre $R$. Es un conjunto $M$ junto con

1. Una operación binaria sobre $M$. Donde $\langle M, + \rangle$ es un grupo Abeliano.
2. Una acción de $R$ sobre $M$ (que es una función $R \times M \longrightarrow M$) denotado por $rm$, tal que para todo $r, s \in R$ y para todo $m, n \in M$ se satisface que

   *a*) $(r + s)m = rm + sm$,

   *b*) $(rs)m = r(sm)$,

   *c*) $r(m + n) = rm + rn$.

   Si el anillo $R$ tiene una identidad, se adiciona el siguiente axioma

   *d*) $1m = m$, para todo $m \in M$.

Si se considera $K$ un cuerpo, entonces el concepto de $K$-módulo es similar al concepto de espacios vectoriales. Por tanto se puede decir que los módulos son una extensión de los espacios vectoriales.

**Ejemplo 1.1.** Sea $R$ un anillo, entonces $M = R$ es un $R$-módulo izquierdo, donde la acción de un elemento del anillo sobre un elemento del módulo es solo la multiplicación del anillo $R$, es decir $R$ es un módulo izquierdo sobre si mismo. En particular, cada cuerpo puede ser considerado como un espacio vectorial sobre si mismo.

**Ejemplo 1.2.** Sea $R = F$ un cuerpo como se señaló anteriormente, cada espacio vectorial sobre $F$ es un $F$-módulo y viceversa. Sea $n \in \mathbb{Z}$ y sea

$$F^n = \{(a_1, a_2, ..., a_n) : a_i \in F\},$$

se puede dar a $F^n$ la estructura de espacio vectorial definiendo la adición componente a componente y la multiplicación por escalar así,

- $(a_1, a_2, .., a_n) + (b_1, b_2, ..., b_n) = (a_1 + b_1, a_2 + b_2, ..., a_n + b_n)$,
- $\alpha(a_1, a_2, ..., a_n) = (\alpha a_1, \alpha a_2, ..., \alpha a_n)$, donde $\alpha \in F$.

Sea $R$ un anillo con identidad y sea $n \in \mathbb{Z}$. defina

$$R^n = \{(a_1, a_2, ..., a_n) : a_i \in R\}.$$

Entonces se puede dar a $R^n$ la estructura de un $R$-módulo por adición componente a componente y multiplicación por escalar. Similarmente cuando $R$ es un cuerpo, el módulo $R^n$ es llamado módulo libre, de rango $n$ sobre $R$.

**Ejemplo 1.3.** El conjunto de pares de enteros $\mathbb{Z}^2$ con suma por componentes y producto por escalares de $\mathbb{Z}$, también por componentes, es un $\mathbb{Z}$-módulo.

**Definición 1.13** (Álgebra)**.** Sea $R$ un anillo conmutativo, un $R$-módulo $A$, se denomina $R$-álgebra si tiene una multiplicación definida en $A$, tal que con la adición dada en $A$ y esta multiplicación, $A$ es un anillo y satisface que $r(ab) = (ra)b = a(rb)$, para todo $r \in R$ y todo $a, b \in A$.

**Ejemplo 1.4.** El anillo $\langle \mathbb{C}, +, \cdot \rangle$ es una $\mathbb{R}$-álgebra. Sean $r \in \mathbb{R}$ y $a, b \in \mathbb{C}$, tal que $a = x_1 + y_1 i$ y $b = x_2 + y_2 i$ con $x_1, x_2, y_1, y_2 \in \mathbb{R}$. Se tiene que

$$\begin{aligned} r(ab) &= r((x_1 + y_1 i)(x_2 + y_2 i)) \\ &= r((x_1 x_2 - y_1 y_2) + (x_1 y_1 + x_2 y_2)i) \\ &= r(x_1 x_2 - y_1 y_2) + r(x_1 y_1 + x_2 y_2)i \\ &= (rx_1 x_2 - ry_1 y_2) + (rx_1 y_1 + rx_2 y_2)i. \end{aligned}$$

Por otro lado

$$\begin{aligned} (ra)b &= (r(x_1 + y_1 i))(x_2 + y_2 i) \\ &= (rx_1 + ry_1 i)(x_2 + y_2 i) \\ &= (rx_1 x_2 - ry_1 y_2) + (rx_1 y_1 + rx_2 y_2)i. \end{aligned}$$

Finalmente

$$\begin{aligned} a(rb) &= (x_1 + y_1 i)(r(x_2 + y_2 i)) \\ &= (x_1 + y_1 i)(rx_2 + ry_2 i) \\ &= (rx_1 x_2 - ry_1 y_2) + (rx_1 y_1 + rx_2 y_2)i. \end{aligned}$$

**Ejemplo 1.5.** El anillo $R[x]$ de polinomios con entradas reales es una $\mathbb{R}$-álgebra.

## 1.4. Anillo de grupo

En 1837, William R. Hamilton dio la primera formalización de los números complejos, definiendolos como parejas de números reales como se hace hoy en día. Después de esto, Hamilton consideró

elementos de la forma $\alpha = a + bi + cj + dk$ a los cuales llamó *cueterniones*, donde los coeficientes $a, b, c, d$ representan números reales e $i, j$ y $k$ son símbolos formales llamados *unidades básicas*. Estas ideas dieron paso a posteriores generalizaciones como los *octoniones* a los que también se les conoce como *números de Cayley*, los *bicuaterniones*, los *sistemas de hipercomplejos*, y se constituye en un primer aporte hacia la definición de *anillo de grupo*.

En las últimas décadas los anillos de grupos son estructuras algebraicas a las cuales se le han encontrado diferentes aplicaciones, por ejemplo en teoría de Códigos y en *criptografía*, ver [6, 7, 5].

**Definición 1.14.** Sea $G$ un grupo (no necesariamente finito) y $R$ un anillo, se denota por $RG$ al conjunto de todas las combinaciones lineales formales

$$\alpha = \sum_{g \in G} a_g g,$$

donde $a_g \in R$ y solo un número finito de coeficientes $a_g$ es diferente de 0. Se definen dos operaciones de $RG$ de la siguiente forma. Sean $\alpha$ y $\beta$ elementos de $RG$ donde,

$$\alpha = \sum_{g \in G} a_g g \quad \text{y} \quad \beta = \sum_{g \in G} b_g g.$$

- La **suma** de dos elementos en $RG$ es componente a componente, así

$$\alpha + \beta = \sum_{g \in G} a_g g + \sum_{g \in G} b_g g = \sum_{g \in G} (a_g + b_g) g.$$

- El **producto** de dos elementos en $RG$ como sigue

$$\alpha\beta = \sum_{g,h \in G} a_g b_h gh.$$

  Este producto se puede representar de la siguiente manera

$$\alpha\beta = \sum_{u \in G} c_u u, \quad \text{donde} \quad c_u = \sum_{gh=u} a_g b_h.$$

El conjunto $RG$ con las operaciones definidas anteriormente se llama **anillo de grupo** de $G$ sobre $R$.

**Observación 1.4.** Si $R$ es conmutativo, $RG$ se denomina **álgebra de grupo**. Algo particular de este conjunto es que $RG$ es un anillo y un $R$-módulo.

**Teorema 1.8.** *El conjunto $RG$ con las operaciones definidas anteriormente es un anillo.*

*Demostración.* Para demostrar que $RG$ es un anillo, primero se prueba que es un grupo Abeliano bajo la suma.

1. Se prueba que la suma es una operación binaria así. Sean $\alpha, \beta \in RG$ tal que

$$\alpha = \sum_{g \in G} a_g g \quad \text{y} \quad \beta = \sum_{g \in G} b_g g,$$

entonces

$$\alpha + \beta = \sum_{g \in G} a_g g + \sum_{g \in G} b_g g,$$

por la definición se tiene que

$$\alpha + \beta = \sum_{g \in G} (a_g + b_g) g,$$

los elementos $a_g, b_g \in R$ y como $R$ es un anillo, entonces $R$ es un grupo Abeliano, por tanto $a_g + b_g \in R$. Además, $g \in G$, así

$$\sum_{g \in G} (a_g + b_g) g \in RG.$$

2. Se consideran $\alpha, \beta, \delta \in G$ para probar la asociatividad, es decir se debe probar que $\alpha + (\beta + \delta) = (\alpha + \beta) + \delta$, con

$$\alpha = \sum_{g \in G} a_g g \quad , \quad \beta = \sum_{g \in G} b_g g \quad \text{y} \quad \delta = \sum_{g \in G} c_g g.$$

Considere

$$\begin{aligned}
\alpha + (\beta + \delta) &= \sum_{g \in G} a_g g + \left( \sum_{g \in G} b_g g + \sum_{g \in G} c_g g \right) \\
&= \sum_{g \in G} a_g g + \sum_{g \in G} (b_g + c_g) g \\
&= \sum_{g \in G} (a_g + (b_g + c_g)) g \\
&= \sum_{g \in G} (a_g + b_g) g + \sum_{g \in G} c_g g \\
&= \left( \sum_{g \in G} a_g g + \sum_{g \in G} b_g g \right) + \sum_{g \in G} c_g g \\
&= (\alpha + \beta) + \delta.
\end{aligned}$$

3. Dado que $R$ es un anillo, existe un elemento neutro $0_r \in R$, luego el elemento neutro de $RG$ se puede representar como $0 = 0_r g$ donde $g$ es cualquier elemento en el grupo $g \in G$.

4. Dado $\alpha = \sum_{g\in G} a_g g \in RG$, el inverso aditivo de $\alpha$ está dado por

$$-\alpha = -\sum_{g\in G} a_g g = \sum_{g\in G} -a_g g,$$

ya que

$$\begin{aligned}\alpha + (-\alpha) &= \sum_{g\in G} a_g + \sum_{g\in G} -a_g g \\ &= \sum_{g\in G} (a_g - a_g) g \\ &= \sum_{g\in G} 0_r g \\ &= 0.\end{aligned}$$

5. Con las anteriores propiedades se probó que $RG$ es un grupo, ahora se debe probar que $\alpha + \beta = \beta + \alpha$ para que el grupo $RG$ sea Abeliano.

   Se considera

$$\begin{aligned}\alpha + \beta &= \sum_{g\in G} a_g g + \sum_{g\in G} b_g g \\ &= \sum_{g\in G} (a_g + b_g)\, g,\end{aligned}$$

   como $R$ es un anillo, es un grupo Abeliano aditivo, por tanto,

$$\begin{aligned}\alpha + \beta &= \sum_{g\in G} (b_g + a_g)\, g \\ &= \sum_{g\in G} b_g g + \sum_{g\in G} a_g g \\ &= \beta + \alpha.\end{aligned}$$

   Así $\langle RG, + \rangle$ es un grupo Abeliano. Para que $RG$ sea un anillo solo resta por probar que la multiplicación es binaria, asociativa y que satisface la propiedad distributiva.

6. Para demostrar que la multiplicación es una operación binaria se consideran,

$$\alpha = \sum_{g\in G} a_g g \quad \text{y} \quad \beta = \sum_{h\in G} b_h h$$

   ahora,

$$\alpha\beta = \sum_{g,h\in G} a_g b_h gh,$$

   Como $R$ es un anillo, implica que $a_g b_h \in R$ y como $G$ es un grupo, implica que $gh \in G$, por tanto $\alpha\beta \in RG$.

7. Para la propiedad asociativa bajo la multiplicación, se debe probar que $\alpha\left(\beta\delta\right)=\left(\alpha\beta\right)\delta$, con

$$\alpha=\sum_{g\in G}a_g g \quad , \quad \beta=\sum_{h\in G}b_h h \quad \text{y} \quad \delta=\sum_{w\in G}c_w w.$$

Considere

$$\begin{aligned}
\alpha(\beta\delta) &= \sum_{g\in G}a_g g\left(\left(\sum_{h\in G}b_h h\right)\left(\sum_{w\in G}c_w w\right)\right)\\
&= \sum_{g\in G}a_g g\left(\sum_{h,w\in G}b_h c_w hw\right)\\
&= \sum_{g,h,w\in G}a_g\left(b_h c_w\right)g\left(hw\right)\\
&= \sum_{g,h,w\in G}\left(a_g b_h\right)c_w\left(gh\right)w\\
&= \left(\sum_{g,h\in G}a_g b_h gh\right)\sum_{w\in G}c_w w\\
&= \left(\left(\sum_{g\in G}a_g g\right)\left(\sum_{h\in G}b_h h\right)\right)\sum_{w\in G}c_w w\\
&= \left(\alpha\beta\right)\delta.
\end{aligned}$$

8. Finalmente debe cumplir la propiedad distributiva, es decir se debe probar que $\alpha\left(\beta+\delta\right)=\alpha\beta+\alpha\delta$. Así

$$\begin{aligned}
\alpha\left(\beta+\delta\right) &= \sum_{g\in G}a_g g\left(\sum_{g\in G}b_g g+\sum_{g\in G}c_g g\right)\\
&= \sum_{g\in G}a_g g\left(\sum_{g\in G}(b_g+c_g)g\right)\\
&= \sum_{g\in G}a_g(b_g+c_g)g\\
&= \sum_{g\in G}(a_g b_g+a_g c_g)g\\
&= \sum_{g\in G}(a_g b_g)g+\sum_{g\in G}(a_g c_g)g\\
&= \left(\sum_{g\in G}a_g g\right)\left(\sum_{g\in G}b_g g\right)+\left(\sum_{g\in G}a_g g\right)\left(\sum_{g\in G}c_g g\right)\\
&= \alpha\beta+\alpha\delta.
\end{aligned}$$

Por lo tanto, el conjunto $RG$ es un anillo.

□

**Proposición 1.1.** *Si el anillo $R$ tiene una identidad $1_R$ y la identidad del grupo $G$ es $e$. El anillo del grupo $RG$ tiene identidad $1_{RG} = 1_R e$.*

*Demostración.* Sea $R$ un anillo con identidad $1_R$ y $G$ con identidad $e$. Entonces $1_R r = r1_R = r$ para todo $r \in R$. Además, se deduce que el elemento $1_R e$ existe en $RG$.

Sea $\sum_{g\in G} r_g g$ un elemento arbitrario en $RG$. Observe que

$$\begin{aligned}(1_R e)(\sum_{g\in G} r_g g) &= \sum_{g\in G}(1_R r_g e g)\\ &= \sum_{g\in G} r_g g.\end{aligned}$$

$$\begin{aligned}(\sum_{g\in G} r_g g)(1_R e) &= \sum_{g\in G}(r_g 1_R g e)\\ &= (\sum_{g\in G} r_g g)\\ &= \sum_{g\in G} r_g g.\end{aligned}$$

Por lo tanto, $1_R e$ es la identidad en $RG$. □

**Ejemplo 1.6.** El anillo de grupo $\mathbb{Z}_2C_2$, donde $\mathbb{Z}_2 = \{0, 1\}$ es el anillo de los enteros módulo 2 y $C_2 = \{e, a\}$, con $a^2 = e$, es el grupo cíclico de orden 2.

Sus elementos están dados de la siguiente manera

$$\begin{aligned}\mathbb{Z}_2C_2 &= \{me + na : m, n \in \mathbb{Z}_2\}\\ &= \{0e + 0a, 1e + 0a, 0e + 0a, 1e + 1a\}\\ &= \{0, e, a, e + a\}.\end{aligned}$$

Las tablas de la suma y el producto de elementos de $\mathbb{Z}_2C_2$ quedan de la siguiente manera.

| $+$ | $0$ | $e$ | $a$ | $e+a$ |
|---|---|---|---|---|
| $0$ | $0$ | $e$ | $a$ | $e+a$ |
| $e$ | $e$ | $0$ | $e+a$ | $a$ |
| $a$ | $a$ | $e+a$ | $0$ | $e$ |
| $e+a$ | $e+a$ | $a$ | $e$ | $0$ |

Tabla 1.1: Suma en $\mathbb{Z}_2C_2$.

En este ejemplo se puede observar que las unidades de $\mathbb{Z}C_2 = \{e, a\} = C_2$.

| $*$ | $0$ | $e$ | $a$ | $e+a$ |
|---|---|---|---|---|
| $0$ | $0$ | $0$ | $0$ | $0$ |
| $e$ | $0$ | $e$ | $a$ | $e+a$ |
| $a$ | $0$ | $a$ | $e$ | $e+a$ |
| $e+a$ | $0$ | $e+a$ | $e+a$ | $0$ |

Tabla 1.2: Multiplicación en $\mathbb{Z}_2C_2$.

## 1.5. Exponenciación modular binaria

La exponenciación modular consiste en calcular $b^n$ módulo $m$, donde $b, n \in \mathbb{Z}$ y $m \in \mathbb{Z}^+$. En la exponenciación modular binaria primero se representa a $n$ en forma binaria, así

$$n = (a_0 a_1 ... a_{k-2} a_{k-1})_2,$$

donde cada $a_i$ es 0 o 1. Esto significa que se representa como

$$n = a_0 2^0 + a_1 2^1 + \cdots + a_{k-1} 2^{k-1}.$$

Por tanto,

$$b^n = b^{a_0 2^0 + a_1 2^1 + \cdots + a_{k-1} 2^{k-1}} = b^{a_0 2^0} b^{a_1 2^1} \ldots b^{a_{k-1} 2^{k-1}}.$$

Así se va calculando cada $b^{a_i 2^i}$ módulo $m$ cuando $a_i = 1$ y simplificando los $a_i = 0$. En caso de que todos los $a_i$ sean igual a 1 se calcula como sigue

$$b^{2^0} \mod m, \quad b^{2^1} \mod m, \quad ..., \quad b^{2^{k-1}} \mod m,$$

de esta manera se encuentra $b^n \mod m$.

**Ejemplo 1.7.**
Calcular $3^{226} \mod 187$. Primero se representa a 226 en forma binaria así

$$(01000111)_2$$

o lo que es lo mismo

$$226 = 2 + 2^5 + 2^6 + 2^7,$$

entonces

$$3^{226} = 3^{2+2^5+2^6+2^7} = 3^2 \times 3^{2^5} \times 3^{2^6} \times 3^{2^7}.$$

Por tanto,

$3^{226} \mod 187 \equiv 3^2 \times 3^{2^5} \times 3^{2^6} \times 3^{2^7} \mod 187$

$3^{226} \mod 187 \equiv (3^2 \mod 187) \times (3^{2^5} \mod 187) \times (3^{2^6} \mod 187) \times (3^{2^7} \mod 187)$

$3^{226} \mod 187 \equiv (9) \times (86) \times (103) \times (137)$

$3^{226} \mod 187 \equiv 10921914 \mod 187$

$3^{226} \mod 187 \equiv 179$

Finalmente se obtiene el resultado $3^{226} \mod 187 \equiv 179$.

El programa SAGE tiene implementado este algoritmo por medio de la función power_mod. Para calcular $b^n$ mód $m$, se escribe power_mod(b,n,m). Así el ejemplo anterior se puede calcular de la siguiente manera.

```
power_mod(3,226,187)
    179
```

## 1.6. Criptografía

La palabra **criptografía** etimológicamente proviene del griego *kryptos* (oculto) y *graphos* (escibir), lo cual podría traducirse como escritura oculta. La criptografía inicialmente se consideraba como el arte de escribir un mensaje de manera secreta, evitando así los posibles intrusos en la comunicación, cuyo único propósito era mantener la confidencialidad de los mensajes. Sin embargo, esta definición tuvo cambios a finales del siglo XX promovidos por el desarrollo de la informática e internet; por esta razón la criptografía ahora se considera una ciencia aplicada que necesita cada vez más de conocimientos matemáticos y por supuesto de la teoría de la información, la cual ahora incluye otros objetivos a parte de transmitir mensajes de manera oculta, entre los que se destacan los siguientes protocolos para: autenticación de identidad, compartir información secreta, transacciones y votaciones electrónicas seguras. Los problemas de seguridad que la criptografía abarca son la confidencialidad, la integridad, la autenticación y el no-rechazo.

- La confidencialidad otorga una característica a la información con la cual esta solo puede ser revelada a usuarios autorizados.
- La integridad se refiere a que la información no puede ser alterada, de forma accidental o fraudulenta, mientras está siendo transmitida.
- La autenticación es el procedimiento de comprobación de la identidad de un usuario. Mediante el procedimiento se garantiza que el usuario es quien dice ser. Por lo general los sistemas de autenticación están basados en el cifrado mediante una clave privada y secreta.
- El no-rechazo se refiere a que no se pueda negar la autoría de un mensaje enviado.

### 1.6.1. Criptosistema

**Definición 1.15.** Un sistema criptográfico o criptosistema es una quíntupla $(M, C, K, T, D)$, donde

- $M$ es el conjunto finito de mensajes sin cifrar (bits, signos, caracteres, etc.),

$$M = \{m_1, m_2, ..., m_n\}.$$

- $C$ es el conjunto finito de mensajes cifrados,

$$C = \{c_1, c_2, ..., c_n\}.$$

- $K$ es el conjunto finito de claves,

$$K = \{k_1, k_2, ..., k_n\}.$$

- $T$ es el conjunto de funciones de cifrado que se aplica a $M$ para obtener $C$, es decir, $e_k : M \to C$ con $k \in K$, es una función tal que cada $e_k$ es diferente dependiendo del $k$ escogido en el conjunto $K$.

- Finalmente $D$ es el conjunto de funciones de descifrado, similar a $T$, donde $d_k$ es la función inversa de $e_k$, aunque se puede utilizar la misma función $e_k$ pero con clave $k'$ la cual es la inversa de la clave $k$ para descifrar el mensaje, es decir, $d_k : C \to M$ o $e_{k'} : C \to M$.

Todo criptosistema debe cumplir que $d_k(e_k(m)) = m$. Es decir, dado un mensaje $m$, si se cifra empleando la clave $k$ con la función $e_k$ y luego se descifra empleando la misma clave con la aplicación $d_k$, se obtiene el mensaje original $m$.

Los sistemas criptográficos se dividen en sistemas simétricos (clave privada) y asimétricos (clave pública), los cuales se presentan a continuación.

### 1.6.2. Criptografía simétrica

Un sistema de **cifrado simétrico** o también llamado de **clave privada** utiliza la misma clave para cifrar y descifrar el mensaje, por lo tanto el emisor como el receptor deben conocer la clave $k$ para cifrar, el emisor cifra un mensaje $m$ usando la clave, es decir, aplica una función a $m$ con la clave $k$ para obtener $c$, $e_k(m) = c$. El receptor recibe $c$ y aplica $d$ con la clave $k$ para descifrar, $d_k(c) = m$, recuperando el mensaje original $m$.

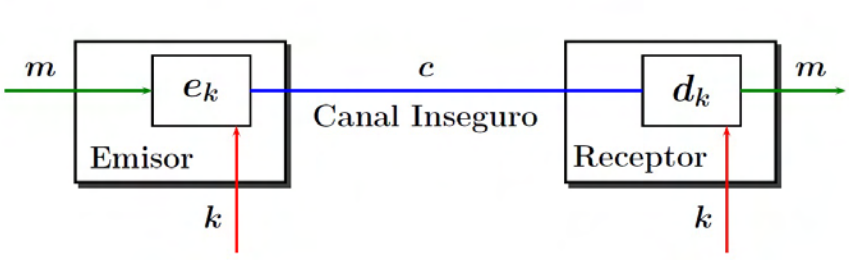


Figura 1.1: Representación gráfica del sistema simétrico.

Dado que en este sistema la clave $k$ debe ser conocida tanto por el emisor como por el receptor, esto sugiere el problema de comunicarse la clave de forma segura.

### 1.6.3. Criptografía asimétrica

Los criptosistemas **asimétricos** o también llamados de **clave pública**, son diferentes a los sistemas simétricos, ya que no usan una misma clave para codificar y decodificar, sino dos claves $(k_p, k_s)$, donde $k_p$ es la clave pública del receptor que el emisor utiliza para cifrar $m$ y obtener $c$, esto es, $e_{k_p}(m) = c$. Por otro lado, $k_s$ es la clave privada que el receptor, el único conocedor de esta, usa para descifrar el mensaje $c$ y recuperar $m$, $d_{k_s}(c) = m$.

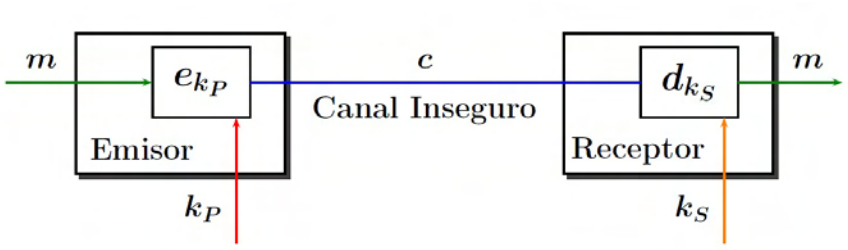


Figura 1.2: Representación gráfica del sistema asimétrico.

En el sistema de clave pública toda persona tiene acceso a conocer la clave con la cual se cifra el mensaje, pero solo el receptor es el conocedor de la clave privada, puesto que la clave de cifrado no permite a nadie encontrar la clave de descifrado sin tener que invertir una gran cantidad de tiempo. Un ejemplo de criptosistema asimétrico es el RSA, el cual se presenta en el Capítulo 3.

### 1.6.4. Firma digital

Una ventaja de la criptografía asimétrica es que sirve para desplegar un método para el desarrollo de firmas digitales. Una firma digital ofrece el soporte de la autenticación e integridad puesto que ayuda al receptor de un mensaje a verificar la autenticidad del origen de la información así como verificar que la información no haya sido modificada. Una firma manuscrita se puede falsificar mientras que la firma digital es imposible en tanto la clave privada del firmante no haya sido descubierta. La firma digital, es un cifrado del mensaje que utiliza la clave privada en lugar de la pública.

## 1.7. Problema de factorización entera

El criptosistema RSA basa su seguridad en el Problema de Factorización Entera (PFE). Este consiste en la dificultad de factorizar completamente un entero $n$, el cual en su presentación más sencilla se asume que tiene dos factores primos $p$ y $q$, la definición formal de este problema es la siguiente.

**Definición 1.16.** Dado $n \in \mathbb{Z}^+$, el cual es el producto de dos números primos $p, q \in \mathbb{Z}^+$, desconocidos y lo suficientemente grandes. Entonces el Problema de Factorización Entera (PFE) consiste en encontrar uno de sus factores bien sea $p$ o $q$.

Por esta razón la dificultad de la factorización de $n$ dependerá de qué tan grandes sean $p$ y $q$, entre más grande sea $n$, mayor será el tiempo requerido para poder factorizarlo.

**Ejemplo 1.8.** Dado $n = 946223925943749769318296542368401703913 93$. Lo único que se sabe es que $n$ es el producto de dos primos $p$ y $q$. El problema de factorización entera es encontrar uno de los factores $p$ o $q$, que en efecto son 307607530132756953133 y 307607530132756953221.

## 1.8. Problema de logaritmo discreto

**Definición 1.17.** Sea $G$ un grupo cíclico finito de orden $n$. Sea $\alpha$ un generador de $G$, y sea $\beta \in G$. El logaritmo discreto de $\beta$ con respecto a la base $\alpha$, denotado $\log_\alpha \beta$, es el único entero $x$, $0 \leq x \leq n-1$, tal que $\beta = \alpha^x$.

El Problema del Logaritmo Discreto (PLD) consisten en encontrar $x$ conociendo a $\alpha$ y $\beta$.

**Ejemplo 1.9.** Dado que 1231 es un número primo se tiene que $\mathbb{F}^*_{2131}$ es un grupo multiplicativo de los enteros módulo 2131 y además que $\mathbb{F}^*_{2131} = \langle 37 \rangle$. Como $1217 \equiv 37^5 \mod 2131$, el logaritmo discreto de 1217 en base 37 es 5, es decir

$$\log_{37} 1217 = 5.$$

Cabe resaltar que entre más grande sea el tamaño del grupo $G$, resolver el problema del logaritmo discreto es cada vez más difícil.

## 1.9. Código ASCII

Un computador guarda la información de manera electrónica, por esta razón cada carácter tiene un código digital, este código es conocido como `ASCII` (American Standard Code for Information Interchange). El código `ASCII` tiene su origen en el año de 1945 a partir de la necesidad de un código para transmitir mensajes por medio del telégrafo. La compañía Bell desarrolló su propio sistema para poder transmitir mensajes de manera estándar a través del telégrafo que consiste en un lenguaje binario de señal y silencio el cual resultó ser práctico en el momento de adoptarlo al lenguaje de maquina; hasta que en 1963, el Comité Estadounidense de Estándares constituyó el código `ASCII` oficialmente.

`ASCII` es un código universal que traduce caracteres alfabéticos a caracteres numéricos. El código `ASCII` básico fue creado en 1967 y representaba 128 caracteres enumerados entre 0 y 127 utilizando

7 bits, contenido que es suficiente para escribir en inglés. Sin embargo, en 1981 se desarrolló el código ASCII extendido de 8 bits donde se remplazan algunos caracteres de control (carácter nulo, avance de línea, inicio de texto y escape) obsoletos por caracteres gráficos y se incorporan 128 caracteres nuevos como símbolos, signos, gráficos adicionales y letras latinas. Caracteres que son necesarios para la escritura en diferentes idiomas. Así el código ASCII extendido se convierte en un estándar oficial.

El código ASCII extendido está formado por caracteres que pertenecen a las siguientes categorías

- Caracteres alfabéticos no ingleses.
- Símbolos de moneda no ingleses.
- Letras griegas.
- Símbolos matemáticos.
- Caracteres para gráficos.
- Caracteres para gráficos de barras.
- Caracteres sombreados.

Una tabla completa del código ASCII extendido puede encontrarse en la página web http://www.elcodigoascii.com.ar/.

# Capítulo 2

# Unidades en anillos de grupo

## 2.1. Introducción

Sea $R$ un anillo con identidad 1, se denota con $\mathcal{U}(R)$ al grupo multiplicativo de unidades de $R$, tal que

$$\mathcal{U}(R) = \{x \in R : (\exists y \in R) \quad xy = yx = 1\}.$$

En particular, dado un grupo $G$ y un anillo $R$ con identidad 1, $\mathcal{U}(RG)$ denota el grupo de unidades del anillo de grupo $RG$.

**Definición 2.1.** El homomorfismo $\varepsilon : RG \to R$ dado por

$$\varepsilon\left(\sum a_g g\right) = \sum a_g$$

con $a_g \in R$ y $g \in G$ se denomina función de aumento de $RG$.

Puede probarse que $\varepsilon$ es un homomorfismo de anillos tal que $\epsilon(u) \in \mathcal{U}(R)$ para todo $u \in \mathcal{U}(RG)$. Se denota por $\mathcal{U}_1(RG)$ al subgrupo de unidades de aumento 1 en $\mathcal{U}(RG)$, esto es

$$\mathcal{U}_1(RG) = \{u \in \mathcal{U}(RG) : \epsilon(u) = 1\}.$$

Para una unidad $u$ del anillo de grupo entero $\mathbb{Z}G$ se tiene que $\epsilon(u) = \pm 1$. Por tanto, se puede probar que

$$\mathcal{U}(\mathbb{Z}G) = \pm\mathcal{U}_1(\mathbb{Z}G).$$

De la misma manera, para un coeficiente arbitrario del anillo $R$. se tiene

$$\mathcal{U}(RG) = \mathcal{U}(R) \times \mathcal{U}_1(RG).$$

**Teorema 2.1** (Producto de unidades)**.** *El producto de unidades también es una unidad. Sean $u$ y $v$ unidades del anillo $RG$. Entonces $uv$ es una unidad. Puesto que, basta con tomar un elemento $v^{-1}u^{-1}$ y calcular*

$$\begin{aligned}(uv)(v^{-1}u^{-1}) &= u(vv^{-1})u^{-1}\\ &= u1u^{-1}\\ &= uu^{-1}\\ &= 1.\end{aligned}$$

**Teorema 2.2** (Potencia de unidades)**.** *Una potencia de una unidad también es una unidad. Sea $u$ una unidad en $RG$. Luego, para $u^n$ con $n \in \mathbb{Z}$ es una unidad. Puesto que, si se toma $(u^{-1})^n$ se tiene que,*

$$\begin{aligned}u^n(u^{-1})^n &= (uu^{-1})^n\\ &= 1^n\\ &= 1.\end{aligned}$$

No hay muchas formas conocidas para la construcción de unidades en anillos de grupo. La mayoría de las construcciones son elementales, algunas de ellas se describen a continuación.

## 2.2. Unidades triviales

Un elemento de la forma $rg$, donde $r \in \mathcal{U}(R)$ y $g \in G$, tiene un inverso $r^{-1}g^{-1}$. Los elementos de esta forma son llamados las **unidades triviales** de $RG$. Por ejemplo, los elementos $\pm g$ con $g \in G$ son las unidades triviales del anillo de grupo entero $\mathbb{Z}G$. Si $K$ es un cuerpo, entonces las unidades triviales de $KG$ son los elementos de la forma $kg$, tales que $k \in K$, $k \neq 0$, $g \in G$.
En términos generales, los anillos de grupos contienen unidades no triviales.

## 2.3. Unidades unipotentes

Si $\eta$ es un elemento de cuadrado cero en un anillo $R$ (es decir, $\eta^2 = 0$), entonces se tiene $(1+\eta)(1-\eta) = 1$. Así, $1+\eta$ y $1-\eta$ son unidades de $R$. En la misma dirección, si $\eta \in R$ tal que $\eta^k = 0$ para algún entero positivo $k$, entonces se tiene que

$$\begin{aligned}(1-\eta)(1+\eta+\eta^2+\cdots+\eta^{k-1}) &= 1-\eta^k = 1\\ (1+\eta)(1-\eta+\eta^2+\cdots\pm\eta^{k-1}) &= 1\pm\eta^k = 1.\end{aligned}$$

entonces $1 \pm \eta$ son unidades de $R$, llamados **unidades unipotentes** de $R$.
En un álgebra de grupo $KG$ sobre un cuerpo de característica $p > 0$ se puede buscar elementos

nilpotentes. Si $g \in G$ es un elemento de orden $p^n$, entonces $(1-g)^{p^n} = 0$, así $\eta = 1-g$ es nilpotente. En este caso $1 - \eta = g$ es una unidad trivial pero $1 + \eta = 2 - g$ siempre es una unidad no trivial.

## 2.4. Unidades bicíclicas

1. Sea $R$ un anillo, $x, y \in R$ tal que $xy = 0$ con $x, y \neq 0$. Si $t \in R$ entonces, considere un elemento $\eta = ytx \in R$, así $\eta^2 = 0$. Por tanto, $1 + \eta$ es una unidad unipotente de $R$.

2. Cuando $R = \mathbb{Z}G$. Se puede obtener divisores de cero considerando un elemento $a \in G$ de orden finito. En efecto, supóngase que $o(a) = n$ entonces

$$a^n - 1 = (a-1)(1 + a + a^2 + \cdots + a^{n-1}) = 0.$$

   Por tanto, $a - 1$ es un divisor de cero, de esta manera si $b \in G$ se puede construir una unidad tomando

$$\mu_{a,b} = 1 + (a-1)b\hat{a}, \quad \text{con} \quad \hat{a} = 1 + a + a^2 + \cdots + a^{n-1},$$

   la cual se denomina bicíclica.

## 2.5. Unidades cíclicas de Bass

**Definición 2.2.** Sea $g$ un elemento de orden $n$ en un grupo $G$. una unidad cíclica de Bass es un elemento del anillo de grupo $\mathbb{Z}G$ de la forma

$$u_i = (1 + g + \cdots + g^{i-1})^{\phi(n)} + \frac{1 - i^{\phi(n)}}{n}\widehat{g},$$

donde $i$ es un entero tal que $1 < i < n - 1$ y $mcd(i, n) = 1$.

**Proposición 2.1.** *Sea $g$ un elemento de orden finito en un grupo $G$. Entonces el elemento*

$$u_i = (1 + g + \cdots + g^{i-1})^{\phi(n)} + \frac{1 - i^{\phi(n)}}{n}\widehat{g},$$

*donde $i$ es un elemento tal que $1 < i < n - 1$ y $\mathrm{mcd}(i, n) = 1$ es en efecto invertible y su inverso está dado por*

$$u_i^{-1} = (1 + g^i + \cdots + g^{i(k-1)})^{\phi(n)} + \frac{1 - k^{\phi(n)}}{n}\widehat{g}.$$

*donde $k$ es un entero tal que $ik \equiv 1 \mod n$.*

**Ejemplo 2.1.** Sea $g = C_{11}$ un elemento de orden $n = 11$, sea $i = 7$ cumple que $\text{mcd}(i, n) = \text{mcd}(7, 11) = 1$ y $\phi(n) = \phi(11) = 10$. Por lo tanto,

$$\begin{aligned} u_7 &= (1 + a + a^2 + a^3 + a^4 + a^5 + a^6)^{10} \\ &\quad + \frac{1 - 7^{10}}{11}(1 + a + a^2 + a^3 + a^4 + a^5 + a^6 + a^7 + a^8 + a^9 + a^{10}) \\ &= (1 + a + a^2 + a^3 + a^4 + a^5 + a^6)^{10} \\ &\quad - 25679568(1 + a + a^2 + a^3 + a^4 + a^5 + a^6 + a^7 + a^8 + a^9 + a^{10}) \\ &= -3190 - 14659a - 21472a^2 - 21472a^3 - 14659a^4 - 3190a^5 \\ &\quad + 9295a^6 + 18832a^7 + 22389a^8 + 18832a^9 + 9295a^{10}. \end{aligned}$$

y su inverso está dado por

$$\begin{aligned} u^{-1} &= (1 + a^2 + a^3 + a^5 + a^6 + a^7 + a^9 + a^{10})^{10} \\ &\quad + \frac{1 - 7^{10}}{11}(1 + a + a^2 + a^3 + a^4 + a^5 + a^6 + a^7 + a^8 + a^9 + a^{10}) \\ &= (1 + a^2 + a^3 + a^5 + a^6 + a^7 + a^9 + a^{10})^{10} \\ &\quad + 25679568(1 + a + a^2 + a^3 + a^4 + a^5 + a^6 + a^7 + a^8 + a^9 + a^{10}) \\ &= 1407 - 3301a - 573a^2 + 3585a^3 - 573a^4 - 3301a^5 + 1407a^6 + 2983a^7 - 2308a^8 - 2308a^9 + 2983a^{10}. \end{aligned}$$

## 2.6. Métodos para construir unidades

En esta sección se presentan algunos teoremas que se pueden encontrar en la literatura que proporcionan formas de construir unidades en anillos de grupo. Aquí no se presentarán las demostraciones, pero para el lector interesado se cita el libro o el artículo donde las puede encontrar.

**Teorema 2.3** ([9, p. 240])**.** *Sea $G$ un grupo tal que $G = \langle a : a^n = 1\rangle$, se tiene que*

1. *Si $n \equiv 1 \mod 4$ entonces*

$$u = \sum_{i=0}^{\frac{n-1}{2}} (-a)^i$$

*es una unidad en $\mathbb{Z}G$ y su inverso está dado por*

$$u^{-1} = \sum_{j=1}^{\frac{n-1}{2}} a^j - \sum_{j=\frac{n+3}{2}}^{n-1} a^j.$$

2. *Si $n \equiv 3 \mod 4$ entonces*

$$u = \sum_{i=0}^{\frac{n-3}{2}} (-a)^i$$

*es una unidad en $\mathbb{Z}G$ y su inverso está dado por*

$$u^{-1} = 1 - \sum_{j=2}^{\frac{n-1}{2}} a^j + \sum_{j=\frac{n+3}{2}}^{n-1} a^j.$$

**Ejemplo 2.2** (Unidades en $\mathbb{Z}C_5$)**.**
En particular en $\mathbb{Z}C_5$. Puesto que $n = 5$ cumple que $n \equiv 1 \mod 4$, se tiene que una unidad está dada por

$$\begin{aligned} u &= \sum_{i=0}^{\frac{n-1}{2}} (-a)^i \\ &= \sum_{i=0}^{2} (-a)^i \\ &= (-a)^0 + (-a)^1 + (-a)^2 \\ &= 1 - a + a^2. \end{aligned}$$

Además

$$\begin{aligned} u^{-1} &= \sum_{j=1}^{\frac{n-1}{2}} a^j - \sum_{j=\frac{n+3}{2}}^{n-1} a^j \\ &= \sum_{j=1}^{2} a^j - \sum_{j=4}^{4} a^j \\ &= a + a^2 - a^4. \end{aligned}$$

Efectivamente

$$\begin{aligned} uu^{-1} &= (1 - a + a^2)(a + a^2 - a^4) \\ &= a + a^2 - a^4 - a^2 - a^3 + a^5 + a^3 + a^4 - a^6 \\ &= a + a^5 - a^6 \\ &= a + 1 - a \\ &= 1. \end{aligned}$$

Como se mencionó en este capítulo, el producto y las potencias de unidades también es una unidad. Así, a partir de la unidad $u = 1 - a + a^2$ se puede construir una nueva unidad $w$, por ejemplo. Una unidad trivial en $\mathbb{Z}C_5$ es $a$, tomando la potencia 4 de $a$, es decir $v = a^4$ cuyo inverso

es $v^{-1} = a$ ya que $(a^4)(a) = a^5 = 1$ y multiplicando con la unidad $u$, se tiene

$$\begin{aligned} w &= vu \\ &= a^4(1 - a + a^2) \\ &= a^4 - a^5 + a^6 \\ &= a^4 - 1 + a \\ &= -1 + a + a^4. \end{aligned}$$

Su inverso sería entonces

$$\begin{aligned} w^{-1} &= u^{-1}v^{-1} \\ &= (a + a^2 - a^4)(a) \\ &= a^2 + a^3 - a^5 \\ &= a^2 + a^3 - 1 \\ &= -1 + a^2 + a^3. \end{aligned}$$

**Ejemplo 2.3** (Unidades en $\mathbb{Z}C_7$)**.**
Como $n = 7$ es congruente con 3 módulo $n \equiv 3 \mod 4$ se tiene que una unidad en el anillo de grupo $\mathbb{Z}C_7$ está dada por

$$\begin{aligned} u &= \sum_{i=0}^{\frac{n-3}{2}} (-a)^i \\ &= \sum_{i=0}^{2} (-a)^i \\ &= (-a)^0 + (-a)^1 + (-a)^2 \\ &= 1 - a + a^2. \end{aligned}$$

Su inverso entonces es

$$\begin{aligned} u^{-1} &= 1 - \sum_{j=2}^{\frac{n-1}{2}} a^j + \sum_{j=\frac{n+3}{2}}^{n-1} a^j \\ &= 1 - \sum_{j=2}^{3} a^j + \sum_{j=5}^{6} a^j \\ &= 1 - (a^2 + a^3) + a^5 + a^6 \\ &= 1 - a^2 - a^3 + a^5 + a^6 \end{aligned}$$

Efectivamente

$$\begin{aligned} uu^{-1} &= (1-a+a^2)(1-a^2-a^3+a^5+a^6) \\ &= 1-a^2-a^3+a^5+a^6-a+a^3+a^4-a^6-a^7+a^2-a^4-a^5+a^7+a^8 \\ &= 1-a+a^8 \\ &= 1-a+a \\ &= 1. \end{aligned}$$

Para resumir el procedimiento de encontrar unidades en $\mathbb{Z}C_n$, en este trabajo se construyó el algoritmo `genunitZCn`, ver Apéndice A.3. De esta manera si se quiere obtener una unidad y su inverso se lo hace de la siguiente manera

```
genunitZCn(5)
  [1 - a + a^2, a + a^2 - a^4]
genunitZCn(7)
  [1 - a + a^2, 1 - a^2 - a^3 + a^5 + a^6]
```

Estos fueron algunos ejemplos de unidades en anillos de grupo. Sin embargo, el lector interesado puede encontrar otras unidades en el capítulo 8 de [9] y en [8, 11].

# Capítulo 3

# Criptosistema RSA

## 3.1. Introducción

Diffie y Hellman de la Universidad de Stanford en 1976 presentaron la noción de criptografía de clave pública, la cual soluciona la dificultad de mantener segura y secreta la clave $k$ necesaria en los sistemas simétricos. Ellos dieron a conocer un algoritmo en el cual se implementa la exponenciación modular, denominado protocolo de intercambio de Diffie-Hellman ver [3]. Además de este protocolo, exponen los requisitos que todo sistema criptográfico asimétrico debe cumplir. Estos son

- El cálculo de claves, pública y privada, debe ser sencillo.
- El proceso de cifrado debe ser sencillo.
- El proceso de descifrado, conociendo la clave secreta, también debe ser sencillo.
- La obtención de la clave secreta, a partir de la pública, debe ser un problema complejo o que requiera de mucho tiempo para solucionarse.
- La obtención del mensaje original, conociendo el mensaje cifrado y la clave pública, también debe ser complejo.

El criptosistema RSA es un ejemplo de criptosistema asimétrico, este fue publicado en el año de 1978 por Ron Rivest, Adi Shamir y Leonard Adleman, del Instituto Tecnológico de Massachusetts (MIT) ver [10], nombre que alude a las iniciales de los apellidos de sus creadores. Este sistema criptográfico cumple con las condiciones anteriormente mencionadas, basa su seguridad en el PFE y utiliza la exponenciación modular, ver secciones 1.7 y 1.5. El criptosistema RSA es muy sencillo de ejecutar y tal vez es uno de los más utilizados actualmente en la seguridad computacional. En este capítulo se presenta cómo se trabaja en este criptosistema.

## 3.2. Criptosistema RSA

En el criptosistema RSA se debe crear una clave pública y una clave privada correspondiente. Para esto se deben considerar dos primos $p$ y $q$ distintos lo suficientemente grandes, calcular $n = pq$ y $\phi(n) = (p-1)(q-1)$. Luego se escoge un entero $e$ tal que

$$1 < e < \phi(n) \quad \text{y} \quad \text{mcd}(e, \phi(n)) = 1.$$

Así la clave pública está dada por $(e, n)$, donde $n$ es el módulo y $e$ es el exponente de cifrado. La clave privada $d$, llamada exponente de descifrado, es un entero tal que

$$1 < d < \phi(n) \quad \text{y} \quad ed \equiv 1 \mod \phi(n).$$

Para cifrar un mensaje el receptor debe escoger un entero $n = pq$ y calcular $e$ y $d$, mantener secreto $d$ y hacer pública la clave $(e, n)$, el emisor cifra el mensaje con esta clave, es decir dado $m$ que es el equivalente numérico de una unidad de texto se debe calcular $m^e$ módulo $n$ para obtener el mensaje cifrado $c$, esto es $c \equiv m^e \mod n$.

Para descifrar el mensaje se calcula $c^d$ módulo $n$, como $d$ es el inverso de $e$ módulo $\phi(n)$, entonces $m \equiv c^d \mod n$. De esta manera, se recupera el mensaje original $m$.

En la práctica, detalles, como la codificación entre letras y números enteros, o proteger la clave privada, son igualmente importantes para la seguridad de las comunicaciones. `SAGE` es un software computacional donde el algoritmo RSA se puede implementar de manera fácil, como sigue.

### 3.2.1. Cifrado RSA en SAGE

Suponga que Bob quiere recibir un mensaje de Alice. Entonces Bob comienza con la construcción de la clave privada y pública. Primero se necesitan dos primos grandes y su producto. En `SAGE` el comando `next_prime(a)` es una forma de obtener un número primo de un tamaño deseado, donde `a` representa el valor a partir del cual se quiere el primo, de esta manera `next_prime(a)` retorna el primer primo mayor que `a`. En el ejemplo que se presenta a continuación se escogen dos primos mayores que $30^{10}$, como se realiza a continuación.

```
p=next_prime(30^10); p
   590490000000023
q=next_prime(p); q
   590490000000037
n=p*q; n
   348678440100035429400000000851
```

Ahora se crea los exponentes de cifrado y descifrado. Se elige el exponente de encriptación $e$ como un número rimo relativo con $\phi(n) = (p-1)(q-1)$. Con `SAGE` se puede factorizar $\phi(n)$ rápidamente

para elegir este valor, pero en la práctica, no sería útil hacer este cálculo para valores grandes de $\phi(n)$. Por tanto, se puede elegir más fácilmente valores aleatorios y verificar que sea primo relativo a $\phi(n)$. El exponente de descifrado es el inverso multiplicativo del exponente de cifrado módulo $\phi(n)$. Si se construye un exponente de encriptación incorrecto, no primo relativo a $\phi(n)$, el cálculo del inverso multiplicativo fallará y `SAGE` lo dirá. Se elige un valor $e$ aleatoriamente y se comprueba si es primo relativo con $\phi(n) = (p-1)(q-1)$.

```
e=47
gcd((p-1)*(q-1),e)==1
   True
```

Luego, buscamos $d$, el inverso de $e$ módulo $\phi(n) = (p-1)(q-1)$, así,

```
d=e.inverse_mod((p-1)*(q-1)); d
   25965415752130209988723404 3143
```

De esta manera Bob le dice a Alice que clave pública es `(e,n)` y mantiene secreta la clave privada `d`.

Cabe anotar que el tamaño del $n$ construido por Bob da la información de la longitud que debe tener el mensaje que Alice quiere cifrar; esto es si $k = [\log_{127} n]$ significa que el mensaje de Alice debe tener a lo más longitud $k$. Sin embargo, si la longitud del mensaje de Alice es mayor que $k$, Alice deberá quebrar el mensaje en bloques de longitud menor que $k$.

Por ejemplo, supóngase que Alice va a cifrar el mensaje "criptosistema" de trece letras. A partir de estas trece letras, se construye un solo número para representar el mensaje. Para esto primero se se usa la función `ord()`, la cual convertirá cada letra del mensaje de Alice a su valor en el código `ASCII`, el cual es un número entre 0 y 127. Se debe resaltar que como se utiliza esta función el mensaje de Alice solo puede contener caracteres en inglés, es decir no acepta palabras con ningún tipo de tilde o letras latinas. Luego Bob utiliza el comando `ZZ(digitos, 128)` para construir el número buscado.

```
palabra = 'criptosistema'
digitos = [ord(letter) for letter in palabra]; digitos
   [99, 114, 105, 112, 116, 111, 115, 105, 115, 116, 101, 109, 97]
M = ZZ(digitos, 128); M
   189284480434524521547081 1491
```

Luego calcula $M^e \mod n$ mediante el siguiente comando.

```
c=power_mod(M,e,n); c
   280178562722461687255388042150
```

El mensaje de Alice ahora está listo para viajar en cualquier red de comunicaciones, sin importar cuán insegura pueda ser.

### 3.2.2. Descifrado RSA en SAGE

Ahora suponga que el valor `c` llegó a Bob. Dado que Bob es el único que conoce la clave privada $d$ sera capaz de descifrar el mensaje; para esto calcula

```
descifrado=power_mod(c,d,n); descifrado
  189284480434524521547081149 1
```

Bob necesita transformar esta representación entera de nuevo en una palabra con letras. Primero utiliza el comando `descifrado.digits(base=128)` para transformar el entero en el listado original. Luego usa la función `chr()` que convierte los valores del código `ASCII` en letras.

```
digitos= descifrado.digits(base=128); digitos
   [99, 114, 105, 112, 116, 111, 115, 105, 115, 116, 101, 109, 97]
letras = [chr(ascii) for ascii in digitos]; letras
   ['c', 'r', 'i', 'p', 't', 'o', 's', 'i', 's', 't', 'e', 'm', 'a']
```

De esa manera se obtuvo el mensaje original **"criptosistema"**.

Para resumir estos procedimientos en `SAGE`, en este trabajo se construyeron los algoritmos `RSA` y `desRSA`, ver apéndices A.1 y A.2. De esta manera por ejemplo Alice pude utilizar el algoritmo `RSA` para encriptar el mensaje "criptosistema" como sigue

```
c=RSA2('criptosistema',e,n); c
 28017856272246168725538804215 0
```

Por otro lado Bob usa el algoritmo `desRSA` para descifrar el mensaje de Alice, así

```
desRSA2(M,d,n)
 ['c', 'r', 'i', 'p', 't', 'o', 's', 'i', 's', 't', 'e', 'm', 'a']
```

# Capítulo 4

# Criptosistema y unidades en anillos de grupo

Sea $G$ un grupo y $R$ un anillo. A continuación se presenta una manera de construir un criptosistema usando unidades en el anillo de grupo $RG$, el cual se puede adaptar para ser asimétrico o simétrico.

## 4.1. Método

Sean $w$ los datos a cifrar, estos se pueden expresar como dígitos en una base cuyo longitud sea menor o igual al orden de $G$. Los dígitos de $w$ se consideran elementos del anillo $R$, en muchos casos $R$ será $\mathbb{Z}$ o $\mathbb{Z}_n$, para algún $n \in \mathbb{Z}^+$. Sin embargo, se podrían usar elementos en cuerpos finitos. De esta manera si $w = w_1 w_2 \cdots w_n$ con $w_i \in R$ se puede representar como un elemento del anillo de grupo $RG$ de la siguiente manera

$$\widehat{w} = \sum_{i=1}^{n} w_i g_i, \quad \text{con} \quad w_i \in R \quad \text{y} \quad g_i \in G.$$

Tomando una unidad en el anillo de grupo $RG$

$$u = \sum_{i=1}^{n} \alpha_i g_i,$$

el cifrado está dado por $w \to \widehat{w}u = \widetilde{w}$. También podría usarse un cifrado $w \to u\widehat{w}$ puesto que no necesariamente los dos cifrados son iguales ya que el anillo de grupo podría no ser conmutativo. Por otro lado, el descifrado puede hacerse multiplicando por el inverso de $u$; esto es, $\widetilde{w} \to \widetilde{w}u^{-1}$.

Vale la pena resaltar que el cifrado depende de la lista de los elementos de $G$ y se podría introducir una mayor complejidad permutando el orden de los elementos en el anillo de grupo.

Como se dijo en el Capítulo 2, el producto y potencias de unidades también es una unidad. Por tanto, el cifrado puede hacerse "más difícil" de descifrar tomando el producto de dos o más unidades. En este caso, solamente el producto extendido de las unidades se da a conocer; situación que es similar a lo que se hace en el criptosistema RSA, en el cual dos primos grandes $p$ y $q$ se multiplican entre sí, pero unicamente el producto $n = pq$ es público.

Por ejemplo, para dos unidades $u, v$ y un elemento $w$, se cifra a $w$ haciendo $w \to \widehat{w}(uv)$ y el descifrado está dado por $\widetilde{w} \to \widetilde{w}(v^{-1}u^{-1})$. Observe que el orden de inversión es importante para las unidades no conmutativas. Así $uv$ es dado y no $u$ y $v$ individualmente. Aunque se conozca la manera de construir $u$ y $v$, la nueva unidad $uv$ no necesariamente debe seguir uno de los métodos de construcción. En general, este procedimiento se puede hacer con el producto de varias unidades.

Además, dada una unidad $u$, se puede utilizar cualquier potencia de $u$ para encriptar el mensaje original. En algunos casos esta nueva potencia será más difícil de obtener para quien no conoce $u$, lo cual introduce el problema del logaritmo discreto expuesto en la Sección 1.8.

**Observación 4.1.** Para cifrar un mensaje, se puede hacerlo de forma simétrica o asimétrica. Si se quiere trabajar sobre un criptosistema simétrico, las dos personas que desean comunicarse deben conocer tanto a $u$ como a $u^{-1}$. De lo contrario, para trabajar sobre un criptosistema asimétrico, el receptor debe hacer pública la unidad $u$ y mantener secreto el inverso $u^{-1}$. Así el emisor cifra el mensaje con la unidad $u$ y el receptor descifra con $u^{-1}$.

**Ejemplo 4.1** (Cifrado usando unidades en $\mathbb{Z}C_5$ )**.**
Considere un mensaje $m = 247$ y asuma que se trabaja con las unidades en el anillo de grupo entero $\mathbb{Z}C_5$, donde $C_5 = \langle a : a^5 = 1 \rangle$ es un grupo cíclico de orden 5. En el ejemplo 2.2 se muestra que una unidad en $\mathbb{Z}C_5$ es

$$w = -1 + a + a^4.$$

Además en el Capítulo 2 se menciona que se puede obtener una unidad $u$ calculando una potencia

$$u = w^{10} = 6051 - 4895a + 1870a^2 + 1870a^3 - 4895a^4.$$

Para cifrar se convierte el mensaje $m = 247$ en un elemento de $\mathbb{Z}C_5$

$$\widehat{m} = 2 + 4a + 7a^2.$$

Luego, se obtiene el cifrado $\widehat{c}$, mediante $\widehat{m}u$

$$\begin{aligned}\widehat{c} &= (2 + 4a + 7a^2)(6051 - 4895a + 1870a^2 + 1870a^3 - 4895a^4) \\ &= 5612 - 19851a + 26517a^2 - 23045a^3 + 10780a^4.\end{aligned}$$

Se puede ver este mensaje como

$$c = [5612, -19851, 26517, -23045, 10780].$$

Para el descifrado, la persona que recibió el mensaje $c$, conoce el inverso $v$ de la unidad $u$, este inverso se puede calcular por medio de la potencia del inverso de $w$, cuyo inverso $w^{-1}$ se calcula en el ejemplo 2.2, por tanto

$$v = (w^{-1})^{10} = 6051 + 1870a - 4895a^2 - 4895a^3 + 1870a^4.$$

Entonces para descifrar el mensaje se calcula

$$\begin{aligned}\widehat{m} &= \widehat{c}v \\ &= \widehat{c}(6051 + 1870a - 4895a^2 - 4895a^3 + 1870a^4) \\ &= 2 + 4a + 7a^2.\end{aligned}$$

Finalmente se obtiene el mensaje original $m = 247$.

A continuación se presenta el ejemplo anteriormente realizado pero implementado en el software computacional `SAGE`.

**Ejemplo 4.2.**
Para implementar este ejemplo en `SAGE`, primero se debe crear el grupo $G$, en este caso, el grupo Abeliano de orden 5, $C_5$. En `SAGE` el anillo de enteros $\mathbb{Z}$ está implementado con la función `IntegerRing()`. Luego se crea el anillo de grupo $RG$ y se introduce la unidad y el inverso.

```
G.<a>=AbelianGroup([5])
RG=GroupAlgebra(G,IntegerModRing())
[a]=RG.gens()
w=-1+a+a^4
iw=-1+a^2+a^3
```

Como ya se ha mencionado, se puede calcular diferentes unidades tomando potencias.

```
u=w^10; u
   6051 - 4895*a + 1870*a^2 + 1870*a^3 - 4895*a^4
v= iw^10; v
  6051 + 1870*a - 4895*a^2 - 4895*a^3 + 1870*a^4
```

Se digita el mensaje 247 como un elemento $m$ del anillo de grupo y se cifra multiplicando el elemento $m$ con la unidad $u$ para obtener $c$

```
m=2+4*a+7*a^2
c=m*u; c
   5612 - 19851*a + 26517*a^2 - 23045*a^3 + 10780*a^4
```

La función `coefficients()` permite obtener los coeficientes, esto es necesario puesto que sólo se envía los coeficientes de $c$

```
C=c.coefficients(); C
   [5612, -19851, 26517, -23045, 10780].
```

Para descifrar, se toma los coeficientes de $C$ y se los escribe como un elemento $c$ del anillo de grupo $\mathbb{Z}C_5$, se calcula $M = cv$ y finalmente se toma los coeficientes de $M$ con la función `M.coefficients()`.

```
  c= c[0] + c[1]*a+ c[2]*a^2+ c[3]*a^3+c[4]*a^4 ; c
     5612 - 19851*a + 26517*a^2 - 23045*a^3 + 10780*a^4
  M=c*v; M
     2 + 4*a + 7*a^2
m=M.coefficients(); m
   [2, 4, 7]
```

Si el número de dígitos en $w$ es más grande o igual al orden de $G$. Entonces, este puede ser cifrado en bloques, es decir, se divide a $w$ en bloques más pequeños, de tal manera que el número de dígitos en cada bloque sea menor o igual que el orden de $G$. Una vez dividido en bloques más pequeños, se cifra cada bloque.

Adicionalmente, un cifrado se puede aplicar a los bloques, llamado cifrado de bloque. A continuación se muestra el proceso.

### 4.1.1. Cifrado de bloque de Hurley

En el artículo [7], los autores proponen aplicar un cifrado a los bloques al cual denominan **cifrado de bloque**. Aquí, el descifrado se realiza conociendo la longitud de cada bloque y la clave para para descifrar el cifrado de bloque.

El procedimiento para el cifrado de bloque es el siguiente. Sea $w$ el elemento que se quiere cifrar, entonces se divide a $w$ en bloques $B_1, B_2, \ldots B_t$ con la misma longitud

$$\widehat{w} = \sum_{i=1}^{q} B_i h_i,$$

donde $h_i$ es un elemento del grupo $H$ cuyo orden es mayor que $t$ y $B_i \in R$, donde se toma $R = \mathbb{Z}$. El cifrado se obtiene mediante $w \to \widehat{w}u = \widetilde{w}$ y se descifra calculando $\widetilde{w}v = \widehat{w}$, donde $u, v \in RH$ y $v$ es el inverso de la unidad $u$.

**Ejemplo 4.3** (Cifrado de bloque en $\mathbb{Z}C_7$)**.**
Se quiere cifrar $w = 2487638945$. Como el número de dígitos de $w$ es mayor al orden de $G = C_7$

(grupo cíclico de orden 7) entonces se procede a cifrar en bloques. Sean $B_1 = 24, B_2 = 87, B_3 = 63, B_4 = 89, B_5 = 45$ los bloques y $u = 1 - a + a^2$ la unidad la cual se mencionó en el ejemplo 2.3.

Para cifrar primero se escribe a $w$ como un elemento del anillo de grupo, así

$$\widehat{w} = 24 + 87a + 63a^2 + 89a^3 + 45a^4.$$

El cifrado está dado por

$$\begin{aligned}\widetilde{w} &= \widehat{w}u \\ &= (24 + 87a + 63a^2 + 89a^3 + 45a^4)(1 - a + a^2) \\ &= -21 + 108a + 24a^3 + 152a^4 - 89a^5 + 134a^6.\end{aligned}$$

El descifrado es entonces

$$\begin{aligned}\widehat{w} &= \widetilde{w}u^{-1} \\ &= (-21 + 108a + 24a^3 + 152a^4 - 89a^5 + 134a^6)(1 - a^2 - a^3 + a^5 + a^6) \\ &= 24 + 87a + 63a^2 + 89a^4 + 45a^6,\end{aligned}$$

donde $u^{-1}$ también se calculó en el ejemplo 2.3.

La implementación del ejemplo en `SAGE` es la siguiente

**Ejemplo 4.4.**
Ya se mencionó en el ejemplo 4.2 que se debe crear el anillo de grupo $RG$ e introducir la unidad $u$ y su inverso $v$ las cuales fueron calculadas en el ejemplo 2.3.

```
G.<a>=AbelianGroup([7])
RG=GroupAlgebra(G,IntegerRing())
[a]=RG.gens()
u=1-a+a^2
v=1-a^2-a^3+a^5+a^6
```

ahora se toma el mensaje que se quiere cifrar, en este caso 2487638945, dividirlo en bloques y escribirlo como elemento $w$ del anillo de grupo $\mathbb{Z}C_7$. Seguidamente se multiplica $w$ por la unidad $u$ para obtener el mensaje cifrado $wc$,

```
w=24+87*a+63*a^2+89*a^4+45*a^6
wc=w*u; wc
   -21 + 108*a + 24*a^3 + 152*a^4 - 89*a^5 + 134*a^6.
```

Finalmente se envía $wc$.

Para descifrar, se toma $wc$ y se multiplica por $v$, el inverso de la unidad $u$, obteniendo $wd$ y finalmente se toma los coeficientes de $wd$ con la función `coefficients()`.

```
wd=wc*v; wd
   24 + 87*a + 63*a^2 + 89*a^4 + 45*a^6
wd.coefficients()
   [24, 87, 63, 89, 45]
```

Así, el mensaje descifrado es la unión de los coeficientes de $wd$ es decir, 2487638945

### 4.1.2. Una extensión al cifrado de bloque de Hurley

Como resultado de trabajo de grado se propone una extensión del método anterior. El método extendido del cifrado de bloque consiste en cifrar cada bloque con una unidad diferente.

Sea $w = w_1 w_2 \cdots w_t$ el mensaje a cifrar, donde $w_i \in R$, se divide a $w$ en bloques $B_1, B_2, \ldots, B_s$, por tanto se puede escribir a cada bloque como un elemento del anillo $RH$

$$\widehat{B_i} = \sum_{j=1}^{m} w_{ij} h_{ij} \quad \text{donde} \quad 1 < i < s,$$

donde $w_j \in R$ y $h_j \in H$. El cifrado se realiza con $B_i \to \widehat{B_i} u_i = \widetilde{B_i}$ donde $u_i$ es una unidad en $RH$, por tanto el mensaje que se envía es cada $\widetilde{B_i}$.

El descifrado se obtiene calculando $\widetilde{B_i} v_i$, donde $v_i$ es el inverso de $u_i$ en $RH$.

**Ejemplo 4.5** (Cifrado de bloque extendido en $\mathbb{Z}C_7$)**.**
Se quiere cifrar $w = 785632159472$ usando unidades en el anillo de grupo $\mathbb{Z}C_7$. Como el número de dígitos de $w$ es mayor al orden de $G = C_7$, entonces se procede a dividir a $w$ en bloques. Sean $B_1 = 7856, B_2 = 3215$ y $B_3 = 9472$ los bloques y $u_1 = 1 - a + a^2$ la unidad que se calculo en el ejemplo 2.3, se procede a calcular más unidades tomando potencias de la unidad $u_1$

$$\begin{aligned} u_1 &= 1 - a + a^2, \\ u_2 &= (1 - a + a^2)^2 = 1 - 2a + 3a^2 - 2a^3 + a^4, \\ u_3 &= (1 - a + a^2)^3 = 1 - 3a + 6a^2 - 7a^3 + 6a^4 - 3a^5 + a^6. \end{aligned}$$

se conoce el inverso $u^{-1} = 1 - a^2 - a^3 + a^5 + a^6$ calculado en el ejemplo 2.3, por tanto los inversos de $u_2$ y $u_3$ se obtiene calculando las potencias del inverso $u^{-1}$, así

$$\begin{aligned} u_1^{-1} &= 1 - a^2 - a^3 + a^5 + a^6, \\ u_2^{-1} &= (1 - a^2 - a^3 + a^5 + a^6)^2 = -1 - 4a - 4a^2 - a^3 + 3a^4 + 5a^5 + 3a^6, \\ u_3^{-1} &= (1 - a^2 - a^3 + a^5 + a^6)^3 = -17 - 17a - 4a^2 + 12a^3 + 19a^4 + 12a^5 - 4a^6. \end{aligned}$$

Para cifrar primero se escribe a cada bloque $B_i$ así

$$\begin{aligned} \widehat{B}_1 &= 7 + 8a + 5a^2 + 6a^3, \\ \widehat{B}_2 &= 3 + 2a + a^2 + 5a^3, \\ \widehat{B}_3 &= 9 + 4a + 7a^2 + 2a^3. \end{aligned}$$

El cifrado está dado por

$$\widetilde{B}_1 = \widehat{B}_1 u_1 = 7 + a + 4a^2 + 9a^3 - a^4 + 6a^5,$$
$$\widetilde{B}_2 = \widehat{B}_2 u_2 = 8 - 4a + 6a^2 + 3a^3 - 8a^4 + 15a^5 - 9a^6,$$
$$\widetilde{B}_3 = \widehat{B}_3 u_3 = 4 - 22a + 51a^2 - 58a^3 + 62a^4 - 40a^5 + 25a^6.$$

Finalmente se envía $\widetilde{B}_1$, $\widetilde{B}_2$ y $\widetilde{B}_3$ .

El descifrado es entonces

$$\widehat{B}_1 = \widetilde{B}_1 u_1^{-1} = 7 + 8a + 5a^2 + 6a^3,$$
$$\widehat{B}_1 = \widetilde{B}_2 u_2^{-1} = 3 + 2a + a^2 + 5a^3,$$
$$\widehat{B}_1 = \widetilde{B}_3 u_3^{-1} = 9 + 4a + 7a^2 + 2a^3.$$

La aplicación de este ejemplo en el software `SAGE` es la siguiente

**Ejemplo 4.6.** Sea 785632159472 el mensaje a cifrar, se crea el anillo de grupo descrito en el ejemplo 4.4, se escribe la unidad $u1$ y su inverso $v1$, nombrado así porque en este ejemplo se necesitan varias unidades y varios inversos.

```
G.<a>=AbelianGroup([7])
RG=GroupAlgebra(G,IntegerRing())
[a]=RG.gens()
u1=1-a+a^2
v1=1-a^2-a^3+a^5+a^6
```

En este ejemplo se va a dividir a 785632159472 en 3 bloques, es decir se tendría los siguientes elementos

```
B1=7+8*a+5*a^2+6*a^3
B2= 3+2*a+a^2+5*a^3
B3= 9+4*a+7*a^2+2*a^3.
```

Por lo tanto, se procede a tomar 2 potencias de la unidad $u1$ y 2 potencias del inverso $v1$

```
u2=u1^2; u2
   1 - 2*a + 3*a^2 - 2*a^3 + a^4
u3=u1^3; u3
   1 - 3*a + 6*a^2 - 7*a^3 + 6*a^4 - 3*a^5 + a^6
v2=v1^2; v2
   -1 - 4*a - 4*a^2 - a^3 + 3*a^4 + 5*a^5 + 3*a^6
v3=v1^3; v3
   -17 - 17*a - 4*a^2 + 12*a^3 + 19*a^4 + 12*a^5 - 4*a^6
```

Para cifrar se toma cada bloque y se multiplica por cada una de las unidades obteniendo los valores $B1c$, $B2c$ y $B3c$ los cuales serán enviado.

```
B1c=B1*u1; B1c
 7 + a + 4*a^2 + 9*a^3 - a^4 + 6*a^5
B2c=B2*u2; B2c
 8 - 4*a + 6*a^2 + 3*a^3 - 8*a^4 + 15*a^5 - 9*a^6
B3c=B3*u3; B3c
 4 - 22*a + 51*a^2 - 58*a^3 + 62*a^4 - 40*a^5 + 25*a^6.
```

Finalmente se envía los elementos $Bic$.

Para descifrar se toma cada $Bic$ y se multiplica por cada inverso $vi$ donde $0 < i < 4$, obteniendo los elementos $B1d$, $B2d$ y $B3d$ que corresponden a los mensajes descifrados

```
B1d= B1c*v1; B1d
 7 + 8*a + 5*a^2 + 6*a^3
B2d= B2c*v2; B2d
 3 + 2*a + a^2 + 5*a^3
B3d= B3c*v3; B3d
 9 + 4*a + 7*a^2 + 2*a^3
```

Como se dijo anteriormente, en algunas aplicaciones es cierto que $\widehat{w}u = u\widehat{w}$, pero no necesariamente es el caso. Si $\widehat{w}u = u\widehat{w}$ para todo $\widehat{w}$ entonces se dice que el cifrado/descifrado es conmutativo, de lo contrario se dice que es no conmutativo. En efecto, la no conmutatividad puede agregar una dimensión de complejidad mayor. Cuando el cifrado no es conmutativo $w \to \widehat{w}u$ es diferente de $w \to u\widehat{w}$. Además, un cifrado se puede realizar por $w \to v\widehat{w}u$, para las unidades $v$ y $u$, que en el caso no conmutativo es diferente de $w \to vu\widehat{w}$.

El inverso $u^{-1}$ de $u$ debe ser conocido para realizar el descifrado. El descifrado es $\widetilde{c} \to u^{-1}\widetilde{c}$ cuando el cifrado es $w \to u\widehat{w}$. En cambio, si el cifrado fue $w \to \widehat{w}u$, entonces el descifrado se realiza por $\widetilde{c} \to \widetilde{c}u^{-1}$.

## 4.2. Criptosistema RSA y unidades de anillos de grupo

El método de cifrado usando unidades de anillos de grupo también se puede combinar con métodos conocidos como el criptosistema RSA y esto proporciona un método de cifrado que se espera sea más fuerte que RSA y más fuerte que el método de unidades en anillos de grupo.

En RSA el entero $n$ es escogido como el producto de dos primos grandes $p$ y $q$. Se puede introducir el cifrado con unidades en anillos de grupo en cualquier etapa, bien sea antes o después de aplicar

el cifrado RSA. En el anillo de grupo $RG$, se considera entonces

$$\widehat{m} = \sum_{i=0}^{k} m_i g_i \quad \text{y una unidad} \quad u = \sum_{i=0}^{k} \alpha_i g_i.$$

En primer lugar se cifra $m$ por $m \to \widehat{m}u = \widetilde{m}$. La unidad es dada por el receptor. Entonces se toman los coeficientes de $\widetilde{m}$ y se cifra cada coeficiente usando RSA para obtener un $\widetilde{c}$. En esta etapa el emisor podría enviar solo los coeficientes de $\widetilde{c}$ y el receptor usaría primero el inverso del sistema RSA para cada coeficiente, luego se escribe los coeficientes como un elemento del anillo de grupo y usar el inverso del cifrado de unidades en anillos de grupo.

Otra opción es cifrar $m$ por $m^e \mod n = c$, escribir al resultado $c$ como un elemento de un anillo de grupo, luego se cifra a $c$ por $c \to \widehat{c}u = \widetilde{c}$ y se envían los coeficientes de $\widetilde{c}$. El receptor para obtener el mensaje original, primero toma los coeficientes recibidos y los escribe como un elemento del anillo de grupo, usa el inverso del cifrado en unidades de anillo de grupo, luego toma los coeficientes y descifra con el inverso del sistema RSA.

**Ejemplo 4.7** (Cifrado de unidades con RSA)**.**
En el Capítulo 2 se menciona como se puede obtener unidades en $\mathbb{Z}G$ si $n = 1 \mod 4$, en este ejemplo se trabajará con $\mathcal{U}(\mathbb{Z}C_{25})$, puesto que $25 \equiv 1 \mod 4$. Así, una unidad está dada por

$$u = 1 - a + a^2 - a^3 + a^4 - a^5 + a^6 - a^7 + a^8 - a^9 + a^{10} - a^{11} + a^{12}$$

y su inverso

$$v = a+a^2+a^3+a^4+a^5+a^6+a^7+a^8+a^9+a^{10}+a^{11}+a^{12}-a^{14}-a^{15}-a^{16}-a^{17}-a^{18}-a^{19}-a^{20}-a^{21}-a^{22}-a^{23}-a^{24}$$

**Cifrado**

Bob quiere enviar el mensaje $m = 164$ a Alice.

- Alice elije dos primos $p = 11$ y $q = 13$ y se calcula $n = pq = (11)(13) = 143$.
- Calcula $\phi(n) = (p-1)(q-1) = 120$ y elije $e$ como $e = 7$.
- Mantiene secreto $d = 103$ tal que $ed \equiv 1 \mod \phi(n)$.
- Alice escoge la unidad $u$
- Alice hace publico $(7, 143, u)$ para Bob.
- Bob calcula $109 \equiv 164^7 \mod 143$ y se representa a 109 como $\widehat{c} = 1 + 9a^2$

- Luego Bob calcula

$$\begin{aligned} c_1 &= \widehat{c}u \\ &= (1+9a^2)(1-a+a^2-a^3+a^4-a^5+a^6-a^7+a^8-a^9+a^10-a^11+a^12) \\ &= 1-a+10a^2-10a^3+10a^4-10a^5+10a^6-10a^7+10a^8-10a^9 \\ &+ 10a^{10}-10a^{11}+10a^{12}-9a^{13}+9a^{14} \end{aligned}$$

**Descifrado** Alice toma $c_1$ y calcula

$$\begin{aligned} \widehat{c} &= c_1 v \\ &= 1+9a^2 \end{aligned}$$

luego se toma a $c$, es decir 109 y calcula

$$164 \equiv 109^{103} \mod 143,$$

es decir 164 corresponde al mensaje original.

## 4.3. Criptosistema alternativo RSA con unidades en $\mathbb{Z}_nG$

A continuación se muestra un cifrado en el cual no es necesario que el receptor haga pública una unidad de $\mathbb{Z}_nG$ ya que el emisor puede ocultar el inverso $v$ de la unidad $u$ que sirve para descifrar, esto lo realiza haciendo uso de la clave del receptor para enviar el mensaje y la inversa $v$ de manera oculta. A continuación se muestra el proceso.

**Cifrado.** Bob quiere enviar un mensaje $m$ a Alice.

- Alice elige dos números primos distintos $p, q$ y calcula $n = pq$.
- Calcula $\phi(n) = (p-1)(q-1)$ y elige $e$ tal que $\text{mcd}(e, \phi(n)) = 1$.
- Se mantiene secreto $d$ tal que $ed \equiv 1 \mod \phi(n)$.
- Alice hace público $(e, n)$ para Bob.
- Bob calcula $c \equiv m^e \mod n$ y se representa a $c$ como un elemento $\widehat{c}$ del anillo de grupo $\mathbb{Z}_nG$, donde $|G|$ es mayor que el número de dígitos de $c$.
- Bob elige una unidad $u$ y su inverso $v$.
- Bob calcula $c_1 = \widehat{c}u$ y finalmente se envía el texto cifrado $C = (c_1, c_2)$, donde $c_2 = v^e$ y este cálculo se efectúa tomando el módulo $n$ de la $e$-ésima potencia de cada uno de los coeficientes de $v$ uno por uno (no directamente de la unidad).

**Descifrado**

- Alice toma $c_2$ y calcula la potencia $d$ módulo $n$ de cada coeficiente de $c_2$ para obtener los coeficientes de $v$, y así obtiene el inverso de la unidad $u$.
- Alice calcula $c_1 v$ para obtener $\widehat{c}$. Finalmente se escribe a $\widehat{c}$ como $c$ y se calcula $m \equiv c^d \mod n$ y así recuperar el mensaje original $m$.

Ahora, se ilustra este esquema utilizando las unidades del anillo de grupo entero de un grupo cíclico de orden 5.

**Ejemplo 4.8** (Cifrado alternativo de RSA con unidades en $\mathbb{Z}_n C_5$)**.**
Para realizar este ejemplo se necesita de $n = pq$, donde $p$ y $q$ son primos, $e$ un entero positivo primo relativo a $\phi(n)$, $d$ el inverso de $e$ módulo $\phi(n)$, una unidad $u$ y su inverso $v$ en $\mathbb{Z}_n C_5$.

**Cifrado**

Bob quiere enviar el mensaje $m = 104$ a Alice.

- Alice elije dos primos $p = 53$ y $q = 59$ y se calcula $n = pq = (53)(59) = 3127$.
- Calcula $\phi(n) = (p-1)(q-1) = 3016$ y elije $e$ como $e = 7$.
- Mantiene secreto $d = 431$ tal que $ed \equiv 1 \mod \phi(n)$.
- Alice hace publico $(17, 10403)$ para Bob.
- Bob calcula $2681 \equiv 104^7 \mod 3127$, se representa a 2681 como $\widehat{c} = 2 + 6a + 8a^2 + 1a^3$
- Luego Bob escoge una unidad $u$ y su inverso $v$, en el ejemplo 2.2 se mencionó que una unidad es

$$w = 1 - a + a^2 \quad \text{y su inverso} \quad w^{-1} = a + a^2 - a^4,$$

para obtener una unidad $u$ se puede calcular una potencia de $w = 1 - a + a^2$, sea

$$u = w^4 = 3112 + 6a + 6a^2 + 3112a^3 + 19a^4$$

por tanto su inverso está dado por

$$\begin{aligned} v &= (w^{-1})^4 \\ &= (a + a^2 - a^4)^4 \\ &= 6 + 19a + 6a^2 + 3112a^3 + 3112a^4. \end{aligned}$$

por tanto para cifrar Bob calcula

$$\begin{aligned} c_1 &= \widehat{c}u \\ &= (2 + 6a + 8a^2 + 1a^3)(3112 + 6a + 6a^2 + 3112a^3 + 19a^4) \\ &= 436 + 451a + 451a^2 + 436a^3 + 907a^4, \end{aligned}$$

y después la segunda parte de su mensaje haciendo

$$\begin{aligned} c_2 &= v^e \\ &= (6 + 19a + 6a^2 + 3112a^3 + 3112a^4)^7 \\ &= 1633 + 27a + 1633a^2 + 3032a^3 + 3032a^4. \end{aligned}$$

Se recuerda que este cálculo se realiza elevando a la $e$ módulo $n$ cada uno de los coeficientes de $v$. Finalmente, Bob envía

$$\begin{aligned} C &= (c_1, c_2) \\ &= [436 + 451a + 451a^2 + 436a^3 + 907a^4, 1633 + 27a + 1633a^2 + 3032a^3 + 3032a^4]. \end{aligned}$$

**Descifrado**

- Alice toma $c_2$ y calcula la potencia $d = 431$ modulo 3127 de cada coeficiente de $c_2$ para obtener $v$

$$\begin{aligned} v &= c_2^{3127} \\ &= (1633 + 27a + 1633a^2 + 3032a^3 + 3032a^4)^{431} \\ &= 6 + 19a + 6a^2 + 3112a^3 + 3112a^4. \end{aligned}$$

- Alice para obtener $\widehat{c}$ calcula

$$\begin{aligned} v &= c_1 v \\ &= (436 + 451a + 451a^2 + 436a^3 + 907a^4)(6 + 19a + 6a^2 + 3112a^3 + 3112a^4) \\ &= 2 + 6a + 8a^2 + 1a^3. \end{aligned}$$

  Finalmente se toma 2681 y se calcula

$$104 \equiv 2681^{431} \mod 3127$$

  Así, se obtiene el mensaje original 104.

Como se mencionó en el Capítulo 2 se puede generar una unidad tomando una potencia arbitraria de la unidad $-1+a+a^4$. las potencias de la unidad $u$ y el inverso $v$ en el anterior ejemplo se realizaron en `SAGE` de la siguiente manera.

```
p=next_prime(50); p
  53
q=next_prime(p); q
  59
```

```
n=p*q; n
  3127
e=7
d= e.inverse_mod((p-1)*(q-1)); d
  431
G.<a>=AbelianGroup([5])
RG=GroupAlgebra(G,IntegerModRing(n))
[a]=RG.gens()
w=1-a+a^2
u=w^4; u
  3112 + 6*a + 6*a^2 + 3112*a^3 + 19*a^4
wi=a + a^2 - a^4
v=wi^4; v
  6 + 19*a + 6*a^2 + 3112*a^3 + 3112*a^4
```

## 4.4. Ejemplos implementados en SAGE.

Estos ejemplos fueron construidos con la ayuda de los algoritmos realizados en SAGE, ver Apéndice A. Los números en los ejemplos y las longitudes aquí se mantienen razonables para que puedan mostrarse.

### 4.4.1. Cifrado con unidades en $RG$

Los ejemplos presentados en esta sección se realizan con los algoritmos A.5 y A.6.

**Ejemplo 4.9** (Cifrado y descifrado en $\mathbb{Z}C_5$)**.**
Se va a cifrar y descifra el mensaje "hola".

**Cifrado**

Se necesita el anillo de grupo $RG = \mathbb{Z}C_5$ y la unidad $u$ para cifrar, así

```
G=AbelianGroup([5]);
RG=GroupAlgebra(G,IntegerModRing())
[a]=RG.gens()
u=-1+a+a^4,
```

seguidamente se hace uso del algoritmo

```
cifraunit('holas',RG,u)
  [7, 101, 100, 11, 201].
```

**Descifrado**

Para descifrar el mensaje $[7, 101, 100, 11, 201]$ se necesita $RG$ y el inverso $v$

```
G=AbelianGroup([5]);
RG=GroupAlgebra(G,IntegerRing())
[a]=RG.gens()
v= -1 + a^2 + a^3,
```

se hace uso del algoritmo que permite descifrar

```
desunit([7, 101, 100, 11, 201],RG,v)
  ['h', 'o', 'l', 'a']
```

**Ejemplo 4.10** (Cifrado y descifrado en $\mathbb{Z}C_7$)**.**

Se va a cifrar y descifra el mensaje "Adios".

**Cifrado**

Se necesita el anillo de grupo $RG = \mathbb{Z}C_7$ y la unidad $u$ para cifrar, así

```
G=AbelianGroup([7]);
RG=GroupAlgebra(G,IntegerRing())
[a]=RG.gens()
u= 1-3*a+6*a^2-7*a^3+6*a^4-3*a^5+a^6
```

seguidamente se hace uso del algoritmo

```
cifraunit('Adios',RG,u)
  [-289, 367, -39, 56, 102, -9, 308].
```

**Descifrado**

Para descifrar el mensaje $[-289, 367, -39, 56, 102, -9, 308]$ se necesita $RG$ y el inverso $v$

```
G=AbelianGroup([7]);
RG=GroupAlgebra(G,IntegerRing())
[a]=RG.gens()
v=-17-17*a-4*a^2+12*a^3+19*a^4+12*a^5-4*a^6,
```

se hace uso del algoritmo que nos permite descifrar

```
desunit([-289, 367, -39, 56, 102, -9, 308],RG,v)
  ['A', 'd', 'i', 'o', 's']
```

### 4.4.2. Cifrado y descifrado RSA con unidades en $RG$

Para realizar estos ejemplos se hizo uso de los algoritmos A.7 y A.8.

**Ejemplo 4.11** (Cifrado RSA con unidades en $\mathbb{Z}C_{25}$)**.**
Para ejecutar este algoritmo necesitamos $n = pq$, donde $p$ y $q$ son primos, se necesita un entero $e$ que sea primo relativo con $\phi(n)$, $d$ el inverso de $e$ módulo $\phi(n)$, el anillo de grupo $RG$, la unidad $u$ y el inverso $v$ en $RG$, así

```
p=next_prime(10); p
   11
q=next_prime(p); q
   13
n=p*q; n
   143
e=7
d= e.inverse_mod((p-1)*(q-1)); d
   103
G.<a>=AbelianGroup([25])
RG=GroupAlgebra(G,IntegerRing())
[a]=RG.gens()
u=1-a+a^2-a^3+a^4-a^5+a^6-a^7+a^8-a^9+a^10-a^11+a^12
v=a+a^2+a^3+a^4+a^5+a^6+a^7+a^8+a^9+a^10+a^11+a^12-a^14-a^15
-a^16-a^17-a^18-a^19-a^20-a^21-a^22-a^23-a^24
```

**Cifrado**
Para cifrar "Mundo" se hace uso del algoritmo

```
RSAunit('Mundo',e,n,RG,u)
  [77, -38, 71, 29, 16, -16, 16, -16, 16, -16, 16, -16, 16, 61, -22, 55, 45].
```

**Descifrado**
Para descifrar $[77, -38, 71, 29, 16, -16, 16, -16, 16, -16, 16, -16, 16, 61, -22, 55, 45]$ se hace uso del algoritmo

```
desRSAunit([77, -38, 71, 29, 16, -16, 16, -16,
16, -16, 16, -16, 16, 61, -22, 55, 45],d,n,RG,v)
   ['M', 'u', 'n', 'd', 'o']
```

### 4.4.3. Cifrado y descifrado alternativo RSA con unidades en $\mathbb{Z}_nG$

En el siguiente ejemplo se usaron los algoritmos A.9 y A.10.

**Ejemplo 4.12** (Cifrado en $\mathbb{Z}_n C_5$)**.**
Para cifrar y descifrar se necesita $n = pq$, donde $p$ y $q$ son primos, un entero $e$ que sea primo relativo con $\phi(n)$, $d$ el inverso de $e$ módulo $\phi(n)$, el anillo de grupo $RG$, la unidad $u$ y el inverso $v$ en $RG$, así

```
p=next_prime(50); p
  53
q=next_prime(p); q
  59
n=p*q; n
  3127
e=7
d= e.inverse_mod((p-1)*(q-1)); d
  431
G.<a>=AbelianGroup([5])
RG=GroupAlgebra(G,IntegerModRing(n))
[a]=RG.gens()
u= 3112 + 6*a + 6*a^2 + 3112*a^3 + 19*a^4
v= 6 + 19*a + 6*a^2 + 3112*a^3 + 3112*a^4.
```

**Cifrado**
Se va a cifrar la palabra "amor", se hace uso del algoritmo

```
RSAunitA('amor',e,d,n,RG,u,v)
[2972 + 601*a + 411*a^2 + 414*a^3 + 2520*a^4,
  1633 + 27*a + 1633*a^2 + 3032*a^3 + 3032*a^4].
```

**Descifrado**
Para descifrar $[2972 + 601a + 411a^2 + 414a^3 + 2520a^4, 1633 + 27a + 1633a^2 + 3032a^3 + 3032a^4]$ se usa el algoritmo

```
desRSAunitA([2972 + 601*a + 411*a^2 + 414*a^3 + 2520*a^4,
1633 + 27*a + 1633*a^2 + 3032*a^3 + 3032*a^4],d,n,RG)
   ['a', 'm', 'o', 'r']
```

# Capítulo 5

# Conclusiones

- En este trabajo, la utilización del software computacional `SAGE` permite hacer ejemplos mayores en comparación con los que se puede realizar a simple mano puesto que en `SAGE` están implementadas funciones que facilitan la construcción de algoritmos. Esto muestra que la teoría de este trabajo puede ejemplificarse mejor con la ayuda del sistema de álgebra computacional `SAGE`.

- Es necesario mejorar la implementación de anillos de grupo en el software de álgebra computacional `SAGE`, ya que en este trabajo no se logró implementar ejemplos en anillos de grupo no conmutativos en `SAGE`.

- Los criptosistemas propuesto en este trabajo se pueden estudiar con anillos de grupo conmutativos y no conmutativos en otro tipo de programas como `GAP`, `Matlab` o `Mathematica`. De esta forma estos softwares son otras herramientas que sirven para la ejemplificación de estos criptosistemas.

- Se debe estudiar más profundamente la construcción de otros tipos de unidades en diferentes anillos de grupo, esto proporcionaría más maneras de ejemplificar los sistemas de cifrado y descifrados que se estudiaron en este trabajo.

- Este trabajo se encaminó en la construcción e implementación de criptosistemas, lo cual permitió presentar ejemplos de la teoría estudiada. Sin embargo, a futuro se puede realizar el criptoanálisis de los criptosistemas propuestos en este trabajo, para determinar su fortaleza y su efectividad en la aplicación como futura investigación. Una posible fortaleza de los criptosistemas que involucran anillos de grupo es que la longitud del mensaje encriptado no necesariamente coincide con la longitud del mensaje original.

- En las implementaciones de los algoritmos A.9 y A.10 se trabajó sobre los anillos de grupo $\mathbb{Z}_nG$ y no en $\mathbb{Z}G$ puesto que al encriptar el inverso de la unidad, se cifra cada coeficiente con

RSA y si algunos de los coeficientes de este inverso son negativos se pierde información, lo que no permite que se obtenga el mensaje original.

# Apéndice A

# Algoritmos en SAGE

En este apéndice se consigna el pseudocodigo de los algoritmos y su implementación en el programa computacional `SAGE` utilizados en el presente trabajo de investigación. De los cuales se hace una breve descripción de su fundamento teórico e implementación.

**Algoritmo A.1.** (Cifrado RSA)
Este algoritmo implementa el criptosistema de cifrado RSA. Para esto recibe el mensaje $m$ entre comillas, dos números positivos $e$ y $n$, donde $\text{mcd}(e, n) = 1$. Retorna un número $c$ que corresponde al mensaje cifrado.

**Algoritmo A.1** RSA

**Entrada:** Recibe un mensaje $m$ entre comillas, $e$ y $n$.

**Salida:** Un número $c$ que representa el mensaje cifrado.

1: Construye una lista $L$ a partir del código `asccii` de cada una de las letras en el mensaje $m$. Usa para esta la función `ord()`.
2: Usa la lista $L$ como dígitos en base 128 y construye un número en base 10. Usa la función `ZZ( ,128)`.
3: Calcula $K \equiv j^e \pmod{n}$. Se usa la función `power_mod(j,e,n)`.
4: **devolver** número $K$.

La implementación de este algoritmo en `SAGE` es la siguiente.

```
def RSA(m,e,n):
    #recibe una palabra la cual debe estar entre comillas, e y n tales que $e$ sea
        primo relativo con $\phi(n)$ y retorna un número $K$.
    L=[ord(letter) for letter in m];
    j=ZZ(L,128);
    K=power_mod(j,e,n);
    return(K)
```

**Algoritmo A.2.** (Descifrado RSA)
Este algoritmo implementa el criptosistema de descifrado RSA. Para esto recibe un numero $M$ , dos números positivos $d$ y $n$, donde $ed \equiv 1 \mod (\phi(n))$. Retorna un mensaje $R$ que corresponde al mensaje descifrado.

**Algoritmo A.2** desRSA

**Entrada:** Recibe un número $M$, $d$ y $n$.

**Salida:** Un mensaje $R$ que representa el mensaje descifrado.

1: Calcula $K \equiv M^d \pmod{n}$. Se usa la función `power_mod(m,d,n)`.
2: Usa la función `k.digits(base=128)` para enlistar el mensaje descifrado en $S$.
3: Usa la función `chr()` que convierte los valores del código `ASCII` en letras y se guarda en $R$.
4: **devolver** El mensaje descifrado $R$.

La implementación de este algoritmo en `SAGE` es la siguiente.

```
def desRSA(M,d,n):
    #recibe un numero $M$, $d$ y $n$ tales que $ed \equiv \mod \phi(n)$ y retorna el
        mensaje descifrado.
          K=power_mod(M,d,n);
    S= K.digits(base=128);
    R=[chr(ascii) for ascii in S];
    return(R)
```

**Algoritmo A.3.** (Generar unidades en $\mathbb{Z}C_n$)
Este algoritmo permite generar unidades en $\mathbb{Z}C_n$ donde $n \equiv 1 \mod 4$ o $n \equiv 3 \mod 4$. Para esto solo recibe un número entero positivo $n$.

**Algoritmo A.3** genunitZCn

**Entrada:** Recibe un entero positivo $n$.

**Salida:** Una lista $E$ que contiene una unidad y su inverso en $\mathbb{Z}C_n$.

1: Se genera un grupo Abeliano $G$ de orden $n$.
2: Se llama al álgebra de grupo $RG$ con $G$ y el anillo de enteros `IntegerRing()`.
3: Nombra los generadores de $RG$. Usa `[a]=RG.gens()`.
4: Condiciona que $n$ sea congruente con 1 módulo 4. Usa `if mod(n,4)==1`.
5: Cumpliendo con la anterior condición realiza la unidad $u = \sum_{i=0}^{\frac{n-1}{2}} (-a)^i$ y el inverso $v = \sum_{j=1}^{\frac{n-1}{2}} a^j - \sum_{j=\frac{n+3}{2}}^{n-1} a^j$.
6: Guarda $u$ y $v$ en una lista $E$.
7: Condiciona que $n$ sea congruente con 3 módulo 4. Usa `if mod(n,4)==3`.
8: Cumpliendo con la anterior condición realiza la unidad $u = \sum_{i=0}^{\frac{n-3}{2}} (-a)^i$ y el inverso $v = 1 - \sum_{j=2}^{\frac{n-1}{2}} a^j + \sum_{j=\frac{n+3}{2}}^{n-1} a^j$.
9: Guarda $u$ y $v$ en una lista $E$.
10: **devolver** La lista $E$.

La implementación de este algoritmo en `SAGE` es la siguiente.

```
def genunitZCn(n):
 #Recibe un entero n tal que n es congruente con 1 o 3 modulo 4 y retorna una
     unidad y su inverso.
 G.<a>=AbelianGroup([n]);
 RG=GroupAlgebra(G,IntegerRing());
 [a]=RG.gens();
 if mod(n,4)==1:
      u=sum([(-a)^i for i in [0..(Integer((n-1)/2))]]);
      v=sum([a^j for j in [1..(Integer((n-1)/2))]])-sum(a^j for j in [Integer((n+3)
          /2)..Integer(n-1)]);
      E=[u,v];
 if   mod(n,4)==3:
      u=sum([(-a)^i for i in [0..(Integer((n-3)/2))]]);
      v=1-sum([a^j for j in [2..Integer((n-1)/2)]])+sum([a^j for j in [Integer((n
          +3)/2)..Integer(n-1)]])  ;
      E=[u,v];
 return(E)
```

**Algoritmo A.4.** (Convertir a elemento de RG)

Este algoritmo permite tomar un vector y darle una forma de un elemento de un anillo de grupo $RG$. Para esto recibe un vector `L[ ]` y el anillo de grupo $RG$. Este sirve como auxiliar para los algoritmos que se implementan más adelante.

**Algoritmo A.4** convertirRG

**Entrada:** Recibe un vector $L$ y el anillo de grupo $RG$.

**Salida:** Un elemento $s$ del anillo de grupo $RG$.

1: Calcula la longitud $z$ del vector $L$ con el comando `lend()`.
2: Nombra un $s$ inicial de $RG$ a partir de cero. Usa `s=RG.zero()`.
3: Toma un $i$ que varia entre 0 y $z-1$, que corresponde a los $i$-ésimos componentes del vector inicial $L$. Usa `for i in [0..z-1]:`.
4: Construye $s$ a partir del $s$ inicial con los componentes del vector $L$. Usa `s=s+L[i]*a^i`.
5: **devolver** El elemento $s$ final.

La implementación de este algoritmo en `SAGE` es la siguiente.

```
def convertirRG(L,RG):
 #Recibe un vector $L$ y el anillo de grupo $RG$. Retorna un elemento del anillo
     de grupo $RG$.
 z=len(L);
 s=RG.zero();
 for i in [0..z-1]:
     s=s+L[i]*a^i;
 return(s);
```

**Algoritmo A.5.** (Cifrado con unidades)
Este algoritmo implementa el criptosistema de cifrado con unidades en el anillo de grupo $RG$. Para esto recibe un mensaje $m$ entre comillas, el anillo de grupo $RG$ y la unidad $u$ en $RG$. Retorna un número $c$ que corresponde a los coeficientes del mensaje cifrado.

**Algoritmo A.5** cifraunit

**Entrada:** Recibe un mensaje $m$, el anillo de grupo $RG$ y la unidad $u$.

**Salida:** Una lista de coeficientes $c$ que representa el mensaje cifrado.

1: Construye una lista $L$ a partir del código `asccii` de cada una de las letras en el mensaje $m$. Usa para esta la función `ord()`.
2: Convierte la lista $L$ en un elemento $s$ del anillo de grupo $RG$. Usa la función auxiliar `convertirRG(L,RG)`.
3: Calcula $w$ que es la multiplicación del elemento $s$ y la unidad $u$.
4: Obtiene $c$ que corresponde a los coeficiente de $w$ mediante la función `w.coefficients()`.
5: **devolver** La lista de coeficientes $c$.

La implementación de este algoritmo en `SAGE` es la siguiente.

```
def cifraunit(m,RG,u):
    #recibe una palabra la cual debe estar entre comillas, $RG$ el anillo de grupo y
        $u$ la unidad en el anillo de grupo $RG$.
```

```
    L=[ord(letter) for letter in m];
    s=convertirRG(L,RG);
    w=s*u;
    c=w.coefficients();
    return(c)
```

**Algoritmo A.6.** (Descifrado con unidades)
Este algoritmo implementa el criptosistema de descifrado con unidades en el anillo de grupo $RG$. Para esto, recibe una lista $L$, el anillo de grupo $RG$ y el inverso $v$ de la unidad $u$ en $RG$. Retorna un número $R$ que corresponde al mensaje descifrado.

**Algoritmo A.6** desunit

**Entrada:** Recibe una lista $L$, el anillo de grupo $RG$ y el inverso $v$.

**Salida:** El mensaje $R$ que representa el mensaje descifrado.

1: Convierte la lista $L$ en un elemento $s$ de $RG$. Usa `convertirRG(L,RG)`.
2: Calcula $t$ que es la multiplicación del elemento $s$ y el inverso $v$.
3: Obtiene $m$ que representa a los coeficientes del elemento $t$. Usa la función `t.coefficients()`
4: Usa la función `chr()` que convierte los valores del código ASCII en letras y se guarda en $R$.
5: **devolver** El mensaje descifrado $R$.

La implementación de este algoritmo en `SAGE` es la siguiente.

```
def desunit(L,RG,v):
    #recibe la lista L, RG el anillo de grupo y v el inverso de la unidad u en el
        anillo de grupo RG
    s=convertirRG(L,RG);
    t=s*v
    m=t.coefficients();
    R=[chr(ascii) for ascii in m];
    return(R)
```

**Algoritmo A.7.** (Cifrado RSA y unidades)
Este algoritmo implementa la combinación del cifrado RSA junto con el cifrado de unidades en el anillo de grupo $RG$. Para esto, recibe un mensaje $M$ entre comillas, dos números positivos $e$ y $n$, donde $gcd(e, n) = 1$, el anillo de grupo $RG$ y la unidad $u$ en $RG$. Retorna una lista $C$ que corresponde al mensaje cifrado.

**Algoritmo A.7** RSAunit

**Entrada:** Recibe un mensaje $M$ entre comillas, dos números positivos $e$ y $n$, el anillo de grupo $RG$ y la unidad $u$ en $RG$.

**Salida:** El mensaje $C$ que representa el mensaje cifrado.

1: Construye una lista $L$ a partir del código `ASCII` de cada una de las letras en el mensaje $M$. Usa la función `ord()`.
2: Calcula la longitud $l$ de la lista $L$. Usa `len()`.
3: Construye un vector vacío `c=[]`
4: Toma un $i$ que varia entre 0 y $l-1$, que corresponde a los $i$-ésimos componentes de la lista $L$. Usa `for i in [0..l-1]`.
5: Llena el vector $c$ con los $i$-ésimos componentes de $L$ elevados a la $e$ módulo $n$. Usa `c.append(power_mod(L[i],e,n))`.
6: Toma los coeficientes de $c$ y lo convierte en un elemento $m$ del anillo $RG$. Usa `convertirRG(c,RG)`.
7: Calcula $w$ que corresponde a la multiplicación de $s$ y la unidad $u$.
8: Crea una lista $C$ con los coeficientes de $w$.
9: **devolver** La lista $C$ que corresponde al mensaje cifrado.

La implementación de este algoritmo en `SAGE` es la siguiente.

```
def RSAunit(M,e,n,RG,u):
    # Recibe un mensaje $M$ entre comillas, dos números enteros positivos $e$ y $n$
        tales que $e$ sea primo relativo con $\phi(n)$, El anillo de grupo $RG$ y la
        unidad $u$ en $RG$.
    L=[ord(letter) for letter in M];
    l=len(L);
    c=[];
    for i in [0..(l-1)]:
        c.append(power_mod(L[i],e,n));
        m=convertirRG(c,RG);
    w=m*u;
    C=w.coefficients();
    return(C)
```

**Algoritmo A.8.** (Descifrado RSA y unidades)

Este algoritmo descifra el cifrado de la combinación del cifrado RSA junto con el cifrado de unidades en el anillo de grupo $RG$. Para esto, recibe una lista $C$, dos números positivos $d$ y $n$, donde $ed$ es congruente con 1 módulo $\phi(n)$, el anillo de grupo $RG$ y el inverso $v$ de la unidad $u$ en $RG$. Retorna una lista $M$ que corresponde al mensaje descifrado.

**Algoritmo A.8** desRSAunit

**Entrada:** Recibe una lista $C$, dos números positivos $d$ y $n$, el anillo de grupo $RG$ y el inverso $v$.

**Salida:** El mensaje $M$ que representa el mensaje descifrado.

1: Convierte la lista $M$ en un elemento $w$ del anillo de grupo $RG$. Usa `convertirRG(C,RG)`.
2: Calcula $m$ que es el producto de $s$ y $v$.
3: Llena un vector $t$ con los coeficientes de $m$. Usa `m.coefficients()`.
4: Calcula la longitud $l$ de $t$. Usa `len()`.
5: Construye un vector vacío `c=[]`.
6: Toma un $i$ que varia entre 0 y $l-1$, que corresponde a los $i$-ésimos componentes de la lista $t$. Usa `for i in [0..l-1]`.
7: Llena el vector $c$ con los $i$-ésimos componentes de $t$ elevados a la $d$ módulo $n$. Usa `c.append(power_mod(t[i],d,n))`.
8: Convierte los valores de $c$ en una lista $M$ de letras. Usa la función `chr()`.
9: **devolver** El mensaje descifrado $M$.

La implementación de este algoritmo en `SAGE` es la siguiente.

```
def desRSAunit(C,d,n,RG,v):
    # Recibe una lista $C$ de coeficientes, un entero positivo $d$ tal que $e$ es el
        inverso de $e$ módulo $\phi(n)$, un entero positivo $n$, el anillo de grupo
        $RG$ y $v$ que es el inverso de la unidad $u$ en $RG$.
    w=convertirRG(C,RG);
    m=w*v;
    t=m.coefficients();
    l=len(t);
    c=[];
    for i in [0..(l-1)]:
        c.append(power_mod(t[i],d,n));
    M=[chr(ascii) for ascii in c];
    return(M)
```

**Algoritmo A.9.** (cifrado alternativo de RSA y unidades)

Este algoritmo implementa el cifrado alternativo de la combinación de RSA y el cifrado de unidades en el anillo de grupo. Este recibe un mensaje $M$ entre comillas, dos números positivos $e$ y $n$, donde $mcd(e.n) = 1$, el anillo de grupo $RG$, la unidad $u$ en $RG$ y el inverso $v$ de la unidad $u$. Retorna una lista $C$ que corresponde al mensaje cifrado.

**Algoritmo A.9** RSAunitA

**Entrada:** Recibe un mensaje $M$, dos números positivos $d$ y $n$, el anillo de grupo $RG$, la unidad $u$ y el inverso $v$.

**Salida:** El mensaje $C$ que representa el mensaje cifrado.

1: Construye una lista $L$ a partir del código `ASCII` de cada una de las letras en el mensaje $M$. Usa la función `ord()`.
2: Calcula la longitud $l$ de la lista $L$. Usa `len()`.
3: Construye un vector vacío `c=[]`.
4: Toma un $i$ que varia entre 0 y $l-1$, que corresponde a los $i$-ésimos componentes de la lista $L$. Usa `for i in [0..l-1]`.
5: Llena el vector $c$ con los $i$-ésimos componentes de $L$ elevados a la $e$ módulo $n$. Usa `c.append(power_mod(L[i],e,n))`.
6: Toma los coeficientes de $c$ y lo convierte en un elemento $m$ del anillo $RG$. Usa `convertirRG(c,RG)`.
7: Calcula $c1$ que corresponde a la multiplicación de $s$ y la unidad $u$.
8: Crea una lista $V$ con los coeficientes de $v$. Usa `v.coefficients()`.
9: calcula $l2$ la longitud de $V$. Usa `len(V)`.
10: Construye un vector vacío `t=[]`.
11: Toma un $j$ que varia entre 0 y $l2-1$, que corresponde a los $i$-ésimos componentes de la lista $V$. Usa `for i in [0..l2-1]`.
12: Llena el vector $t$ con los $i$-ésimos componentes de $V$ elevados a la $e$ módulo $n$. Usa `c.append(power_mod(V[j],e,n))`.
13: Convierte la lista $t$ en un elemento $c_2$ del anillo de grupo $RG$. Usa `convertirRG(t,RG)`.
14: Llena una lista $C$ con los componentes $c1$ y $c2$.
15: **devolver** La lista $C$ que es el mensaje cifrado.

La implementación de este algoritmo en `SAGE` es la siguiente.

```
def RSAunitA(M,e,n,RG,u,v):
    #recibe un mensaje $M$ entre comillas, dos entero positivo $e,n$ tal que $e$ es
        primo relativo con $\phi(n)$, el anillo de grupo $RG$, la unidad $u$ y su
        inverso $v$ en $RG$.
    L=[ord(letter) for letter in M];
    l=len(L);
    c=[];
    for i in [0..(l-1)]:
        c.append(power_mod(L[i],e,n));
        m=convertirRG(c,RG);
    c1=m*u;
    V=v.coefficients();
    l2=len(V);
```

```
    t=[];
    for j in [0..(l2-1)]:
        t.append(power_mod(V[j],e,n));
        c2=convertirRG(t,RG)
    C=[c1,c2];
    return(C)
```

**Algoritmo A.10.** (Desencriptar el cifrado alternativo de RSA y unidades)
Este algoritmo desencripta el cifrado alternativo de la combinación del cifrado RSA y el cifrado de unidades en el anillo de grupo $RG$. Para esto, recibe una lista $C$, dos números positivos $d$ y $n$, donde $ed$ es congruente con 1 módulo $\phi(n)$ y el anillo de grupo $RG$. Retorna una lista $m$ que corresponde al mensaje descifrado.

**Algoritmo A.10** desRSAunitA

**Entrada:** Recibe una lista $C$, dos números positivos $d$ y $n$ y el anillo de grupo $RG$.

**Salida:** El mensaje $m$ que representa el mensaje descifrado.

1: Separa los elementos de la lista $C$ en $c1$ y $c2$.
2: Toma una lista $C2$ con los coeficientes de $c2$. Usa `c2.coefficients()`.
3: Calcula la longitud $l2$ de $c2$. Usa `len()`.
4: Construye un vector vacío `V=[]`.
5: Toma un $i$ que varia entre 0 y $l2-1$, que corresponde a los $i$-ésimos componentes de la lista $c2$. Usa `for i in [0..l2-1]`.
6: Llena el vector $V$ con los $i$-ésimos componentes de $c2$ elevados a la $d$ módulo $n$. Usa `V.append(powe_mod(C2[i],d,n))`.
7: Convierte la lista $V$ en un elemento $v$ del anillo de grupo $RG$. Usa `convertirRG(V,RG)`.
8: Calcula $w$ que es el producto de $c1$ y $v$.
9: Calcula los coeficientes de $w$ y los llena en un un vector $W$. Usa `w.coefficients()`.
10: Calcula la longitud $l$ de $W$. Usa `len()`.
11: Construye un vector vacío `M=[]`.
12: Toma un $j$ que varia entre 0 y $l-1$, que corresponde a los $i$-ésimos componentes de la lista $W$. Usa `for j in [0..l-1]`.
13: Llena el vector $M$ con los $i$-ésimos componentes de $W$ elevados a la $d$ módulo $n$. Usa `M.append(power_mod(t[i],d,n))`.
14: Convierte los valores de $M$ en una lista $m$ de letras. Usa la función `chr()`.
15: **devolver** El mensaje descifrado $m$.

La implementación de este algoritmo en `SAGE` es la siguiente.

```
def desRSAunitA(C,d,n,RG):
    #Este algoritmo recibe una lista $C$ con dos coeficientes, dos entero positivo
```

```
        $d$ y $n$ tal que que $d$ es el inverso de $e$ módulo $\phi(n)$ y el anillo
        de grupo $RG$.
    c1=C[0];
    c2=C[1];
    C2=c2.coefficients();
    l2=len(C2);
    V=[];
    for i in [0..(l2-1)]:
        V.append(power_mod(C2[i],d,n));
        v=convertirRG(V,RG);
    w=c1*v;
    W=w.coefficients();
    l=len(W);
    M=[];
    for j in [0..(l-1)]:
        M.append(power_mod(W[j],d,n));
        m=[chr(ascii) for ascii in M];
    return(m)
```

# Referencias

[1] I. Anshel, M. Anshel, and D. Goldfeld. An algebraic method for public-key cryptography. *Math. Res. Lett.*, 6(3-4):287–291, 1999.

[2] J. A. Buchmann. *Introduction to cryptography*. Undergraduate Texts in Mathematics. Springer-Verlag, New York, 2001.

[3] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Trans. Information Theory*, IT-22(6):644–654, 1976.

[4] J. Gallian. *Contemporary Abstract Algebra*. F. Cengage Learning, 2009.

[5] M. Guerreiro. Group algebras and coding theory. *São Paulo Journal of Mathematical Sciences*, 10(2):346–371, Dec 2016.

[6] T. Hanoymak and O. Küsmüş. On construction of cryptographic systems over units of group rings. *Int. Electron. J. Pure Appl. Math.*, 9(1):37–43, 2015.

[7] B. Hurley and T. Hurley. Group ring cryptography. *Int. J. Pure Appl. Math.*, 69(1):67–86, 2011.

[8] E. Jespers and M. M. Parmenter. Units of group rings of groups of order 16. *Glasgow Math. J.*, 35(3):367–379, 1993.

[9] C. Polcino Milies and S. K. Sehgal. *An introduction to group rings*, volume 1 of *Algebra and Applications*. Kluwer Academic Publishers, Dordrecht, 2002.

[10] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Comm. ACM*, 21(2):120–126, 1978.

[11] S. K. Sehgal. Units of integral group rings—a survey. In *Algebraic structures and number theory (Hong Kong, 1988)*, pages 255–268. World Sci. Publ., Teaneck, NJ, 1990.