Aplicaciones de la teoría de números en Criptografía

Pablo Sebastián Herrera

Universidad del Valle de Guatemala

Seminario I de Matemática Aplicada

Introducción

- La criptografía moderna se apoya en problemas de teoría de números que se consideran **difíciles desde el punto de vista computacional**.
- Nos centraremos en tres familias de problemas y sus criptosistemas públicos asociados:
  - **Factorización de enteros** $\Rightarrow$ criptosistema **RSA**.
  - **Problema del logaritmo discreto** (DLP) en grupos finitos $\Rightarrow$ **ElGamal**.
  - **Logaritmo discreto en curvas elípticas** (ECDLP) $\Rightarrow$ criptosistemas **ECC**.
- En todos los casos aparece la misma idea:
  - ciertas operaciones (multiplicar, exponenciar, multiplicación escalar en curvas elípticas) son eficientes,
  - mientras que la operación inversa (factorizar, tomar logaritmos discretos) es **computacionalmente inviable** para parámetros bien elegidos.
- El objetivo es:

$$\text{Problema matemático complicado} \longrightarrow \text{Diseño de un criptosistema público.}$$

Problema de factorización de enteros

- En RSA se trabaja con un entero compuesto

$$N = pq,$$

  donde p y q son primos grandes y secretos.
- **Problema de factorización**:

Dado un entero N que es producto de dos (o más) primos grandes, encontrar una factoración no trivial

$$N = pq, \quad 1 < p, q < N.$$

- Para RSA, conocer una factorización de N permite calcular

$$\varphi(N) = (p-1)(q-1),$$

  y de ahí la clave privada.
- La seguridad de RSA descansa en que, para tamaños típicos (por ejemplo N de 2048 bits), no se conoce un algoritmo de factorización que sea eficiente en el peor caso.

Equivalencia con el cálculo de $\varphi(N)$

- Si se conoce la factorización $N = pq$, entonces

$$\varphi(N) = (p-1)(q-1)$$

  se calcula de forma inmediata.
- Recíprocamente, bajo ciertas condiciones, conocer $\varphi(N)$ permite recuperar p y q.
- En particular, dado N y $\varphi(N)$, se tiene

$$p + q = N - \varphi(N) + 1, \qquad pq = N,$$

  y se pueden hallar p y q resolviendo una ecuación cuadrática.
- Intuitivamente **calcular la clave privada** a partir de la clave pública es esencialmente tan difícil como **factorizar** N.

Experimento: dificultad de factorizar semiprimos

n	# dígitos	Factores primos	Tiempo (s)
2 307 323	7	$[1093, 2111]$	0.0001
2 914 107 281	10	$[46327, 62903]$	0.0085
195 035 301 437	12	$[281023, 694019]$	0.0400
58 266 774 862 333	14	$[6\,759\,707, 8\,619\,719]$	0.8420
2 058 228 245 360 593	16	$[28\,087\,351, 73\,279\,543]$	5.1180

- Cada n es un **semiprimo**, producto de dos primos grandes: $n = p \cdot q$.
- La columna “# dígitos” muestra el tamaño de n en base 10.
- La columna “Factores primos” muestra los primos p y q hallados por *trial division*.
- La columna “Tiempo (s)” es el tiempo que tarda el algoritmo ingenuo en factorizar n.
- Se observa que, al aumentar sólo unos pocos dígitos, el tiempo de factorización crece rápidamente.
- Para módulos RSA reales (centenares de dígitos), este tipo de ataque por fuerza bruta es totalmente inviable.

Criptosistema RSA: generación de claves

- **Parámetros** (se eligen una sola vez):
  1. Elegir dos primos grandes p y q, típicamente de tamaño similar.
  2. Calcular
  $$N = pq, \qquad \varphi(N) = (p-1)(q-1).$$
  3. Escoger e tal que
  $$1 < e < \varphi(N), \quad \gcd(e, \varphi(N)) = 1.$$
  4. Calcular el inverso multiplicativo
  $$d \equiv e^{-1} \pmod{\varphi(N)}.$$
- **Clave pública**: (N, e).
- **Clave privada**: d (y, en la práctica, también se guardan p y q para acelerar el descifrado).
- Dado (N, e) no se conoce un método eficiente para recuperar d sin factorizar N.

Problema del logaritmo discreto

- Sea G un grupo abeliano finito escrito multiplicativamente y $g \in G$ un generador de un subgrupo de orden n.
- Para $h \in \langle g \rangle$ se define el **logaritmo discreto** de h en base g como el entero x tal que

$$g^x = h, \qquad 0 \leq x \leq n-1.$$

- **Problema del logaritmo discreto (DLP)**:

Dados (G, g, h), encontrar x tal que $g^x = h$.

- Exponenciar $g \mapsto g^x$ es eficiente (exponenciación rápida), pero no se conoce un algoritmo general eficiente para invertir esta operación en grupos grandes bien elegidos.

DLP en criptografía: asimetría computacional

- En la práctica se usan grupos como:
  - $G = \mathbb{Z}_p^*$ con p primo grande,
  - grupos de puntos de curvas elípticas (caso ECC).
- Para cada intento de ataque:
  - hay algoritmos subexponenciales (baby-step giant-step, Pollard rho, index calculus) en algunos grupos,
  - pero su complejidad sigue siendo prohibitiva para tamaños de clave recomendados.
- **Asimetría**: computar g^x mód p es "barato", pero recuperar x a partir de g^x es "caro".
- ElGamal explota exactamente esta asimetría: el atacante ve potencias de g pero no tiene acceso a los exponentes.

Criptosistema ElGamal: parámetros y generación de claves

- **Parámetros públicos**:
  - Primo grande p,
  - generador g de un subgrupo grande de $\mathbb{Z}_p^*$.
- **Clave privada** del receptor:

$$a \in \{1, \ldots, p-2\} \quad \text{escogido al azar.}$$

- **Clave pública**:

$$h = g^a \text{ mód } p.$$

- Cualquiera puede usar (p, g, h) para cifrar; sólo quien conoce a puede descifrar.
- Recuperar a a partir de (g, h) es un ejemplo del DLP en $\mathbb{Z}_p^*$.

Ejemplo de la dificultad del DLP

Buscamos un exponente k tal que

$$2^k \equiv 24\,024 \pmod{300\,007}.$$

Iteración k	2^k mód 300 007	Tiempo acumulado (s)
0	1	0.0000
25 000	263 759	0.0034
50 000	186 851	0.0068
75 000	282 991	0.0109
100 000	281 583	0.0175
125 000	17 570	0.0243
149 994	24 024	0.0278

Ejemplo de la dificultad del DLP (cont.)

- Usamos una búsqueda ingenua: recorrer $k = 0, 1, 2, \ldots$ y actualizar iterativamente

$$v_k = 2^k \bmod 300\,007$$

  hasta que v_k coincide con 24 024.

- En el experimento,

$$k = 149\,994, \quad 2^{149\,994} \equiv 24\,024 \pmod{300\,007},$$

  lo que implica **149 995 intentos** en total.

- Aunque el tiempo total medido es pequeño ($\approx 0{,}028$ s), ya estamos haciendo del orden de 10^5 evaluaciones de la potencia modular para un primo relativamente pequeño.

- La complejidad de la búsqueda ingenua crece proporcionalmente al tamaño del grupo: si escalamos a grupos criptográficos reales (por ejemplo, de tamaño $\approx 2^{256}$), este tipo de ataque se vuelve completamente **inviable**.

- Este ejemplo ilustra la **asimetría** central del DLP: computar $2^k \bmod p$ es barato, pero invertir la operación (recuperar k) puede ser extremadamente costoso.

Curvas elípticas sobre cuerpos finitos

- Sea $\mathbb{F}_q$ un cuerpo finito con q elementos.
- Una **curva elíptica** sobre $\mathbb{F}_q$ (en forma corta de Weierstrass) se define por

$$E : y^2 = x^3 + ax + b, \qquad a, b \in \mathbb{F}_q,$$

con discriminante

$$\Delta = -16(4a^3 + 27b^2) \neq 0$$

para evitar singularidades.

- El conjunto de puntos racionales

$$E(\mathbb{F}_q) = \{(x, y) \in \mathbb{F}_q^2 : y^2 = x^3 + ax + b\} \cup \{\mathcal{O}\}$$

forma un **grupo abeliano** bajo la operación de suma definida geométricamente (regla de la cuerda y la tangente), con $\mathcal{O}$ como elemento neutro.

- El tamaño de este grupo satisface el acotamiento de Hasse:

$$|E(\mathbb{F}_q)| = q + 1 - t, \qquad |t| \leq 2\sqrt{q}.$$

Problema del logaritmo discreto en curvas elípticas (ECDLP)

- Sea $E(\mathbb{F}_q)$ el grupo de puntos de una curva elíptica y sea $P \in E(\mathbb{F}_q)$ un punto de orden grande n.
- Para un entero k se define la **multiplicación escalar**

$$Q = kP = \underbrace{P + P + \cdots + P}_{k \text{ veces}}.$$

- **ECDLP** (Elliptic Curve Discrete Logarithm Problem):

Dados P y $Q = kP$ en $E(\mathbb{F}_q)$, encontrar el entero k.

- Computar $Q = kP$ es eficiente (algoritmo "doblar y sumar"), pero no se conoce ningún algoritmo subexponencial general para recuperar k en curvas adecuadamente seleccionadas.
- Esta diferencia de complejidad es la base de la seguridad de los criptosistemas ECC.

Ventajas criptográficas de ECC

- Para un mismo nivel de seguridad, los tamaños de clave en ECC son mucho menores que en RSA/DLP clásico.
- Ejemplo típico:
  - RSA de 3072 bits $\approx$ ECC con claves de 256 bits.
- Consecuencias prácticas:
  - menos almacenamiento de claves,
  - operaciones más rápidas en dispositivos con recursos limitados,
  - menor ancho de banda para certificados y firmas.
- La seguridad se basa en la ausencia de algoritmos subexponenciales eficientes para el ECDLP en curvas “seguras” (evitando curvas con estructuras especiales que faciliten ataques).

Criptosistema ECC tipo ElGamal: parámetros y claves

- **Parámetros públicos**:
  - cuerpo finito $\mathbb{F}_q$,
  - curva elíptica $E/\mathbb{F}_q$,
  - punto generador $G \in E(\mathbb{F}_q)$ de orden primo grande n.
- **Clave privada** del receptor:

$$d \in \{1, \ldots, n-1\} \quad \text{escogido al azar.}$$

- **Clave pública**:

$$Q = dG \in E(\mathbb{F}_q).$$

- Conocer d a partir de (G, Q) requiere resolver una instancia del ECDLP.

Criptosistema ECC tipo ElGamal: cifrado y descifrado

- Supongamos que el mensaje m se representa como un punto $M \in E(\mathbb{F}_q)$ (codificación estándar).
- **Cifrado** de M usando la clave pública Q:
  1. El emisor elige un escalar efímero $k \in \{1, \ldots, n-1\}$ al azar.
  2. Calcula
  $$C_1 = kG, \qquad C_2 = M + kQ.$$
  3. El criptograma es el par (C_1, C_2).
- **Descifrado** con la clave privada d:
  1. Se calcula
  $$dC_1 = dkG = k(dG) = kQ.$$
  2. Se recupera el mensaje
  $$M = C_2 - dC_1.$$
- Seguridad: para extraer M a partir de (C_1, C_2, G, Q) el atacante debería resolver instancias del ECDLP.

Ejemplo de la dificultad del ECDLP

- Trabajamos sobre la curva elíptica

$$E : y^2 = x^3 + 2x + 3 \pmod{40\,009}.$$

- Punto base:

$$P = (3, 6), \quad \text{con } \operatorname{ord}(P) = 2240.$$

- Elegimos un exponente secreto

$$k_{\text{secreto}} = 1234, \qquad Q = k_{\text{secreto}} P = (30\,790, 5425).$$

Iteración k	$R = kP$	Tiempo acumulado (s)
1	$(3, 6)$	0.0000
400	$(4519, 2131)$	0.0008
800	$(39\,485, 15\,527)$	0.0015
1200	$(34\,035, 39\,010)$	0.0023
1234	$(30\,790, 5425)$	0.0024

Ejemplo de la dificultad del ECDLP (cont.)

- El atacante conoce la curva, el primo p, el punto base P y el punto objetivo Q.
- Ataque ingenuo: recorrer $k = 1, 2, 3, \ldots$ y computar iterativamente

$$R_k = kP$$

  hasta que R_k coincida con Q.
- En el experimento se obtuvo

$$k = 1234 \quad \Rightarrow \quad kP = Q,$$

  tras **1234 sumas de puntos** en el grupo $E(\mathbb{F}_{40\,009})$, con un tiempo total de $\approx 0{,}0024$ s.
- De nuevo, el tiempo absoluto es pequeño porque el grupo es diminuto (orden $\approx 10^3$), pero el número de operaciones crece linealmente con el tamaño del grupo.
- En curvas elípticas usadas en criptografía real (orden del grupo $\approx 2^{256}$), una búsqueda lineal de k como esta sería completamente **inviable**, lo que ilustra la dureza del ECDLP.

Comparación experimental de RSA, ElGamal y EC–ElGamal

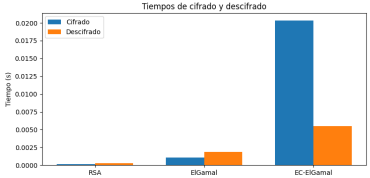
Mensaje cifrado en los tres criptosistemas:

```
Me robe unas pizzas de la pizza party y espero nunca se enteren
```

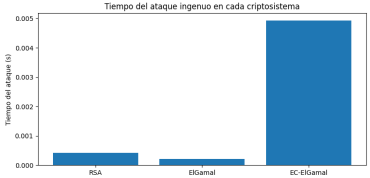
Resumen numérico de los experimentos (ataques ingenuos):

Esquema	Problema duro	Iteraciones	Tiempo ataque (s)	Cif./Descif. (s)
RSA	Factorización de enteros	2578	0.000427	0.000130 / 0.000253
ElGamal	DLP en $\mathbb{Z}_p^*$	1234	0.000206	0.001104 / 0.001880
EC–ElGamal	ECDLP en $E(\mathbb{F}_p)$	789	0.004932	0.020347 / 0.005519

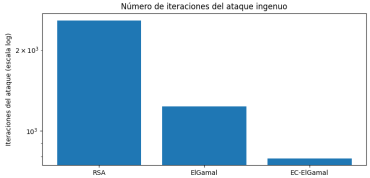
Visualización de los resultados:



Tiempos de cifrado y descifrado



Tiempo del ataque ingenuo



Iteraciones del ataque (escala log)

Conclusiones

- La teoría de números proporciona los **problemas duros** que hacen posible la criptografía de clave pública: factorización de enteros (RSA), logaritmo discreto en grupos multiplicativos (ElGamal) y logaritmo discreto en curvas elípticas (ECC).
- En los tres casos aparece la misma idea central:
  - las operaciones “directas” (multiplicar, exponenciar, multiplicar un punto por un escalar) son eficientes;
  - las operaciones inversas (factorizar, calcular logaritmos discretos o resolver ECDLP) son computacionalmente costosas para parámetros bien elegidos.
- RSA, ElGamal y EC–ElGamal son ejemplos concretos de cómo un **problema matemático abstracto** se traduce en un esquema de cifrado práctico, con propiedades como confidencialidad e incluso firma digital.
- La seguridad de la criptografía moderna no depende de “ocultar” los algoritmos, sino de la **dificultad computacional** de problemas de teoría de números bien estudiados y del uso cuidadoso de parámetros estandarizados.

Referencias


Lobillo, F. J. (2019).
Teoría de Números y Criptografía.


Santamaría Fernández, J. (2012).
Criptografía basada en el problema del logaritmo discreto.


Martínez, J. (s.f.).
Aritmética modular.


Judson, T. W. (2019).
Abstract Algebra: Theory and Applications.


Menezes, A. J., van Oorschot, P. C., & Vanstone, S. A. (1996).
Handbook of Applied Cryptography.
CRC Press.


Katz, J., & Lindell, Y. (2015).
Introduction to Modern Cryptography (2nd ed.).
Chapman & Hall/CRC.