Generación de Contraseñas Seguras mediante Grafos ASCII

Introducción

El uso de contraseñas débiles representa uno de los principales vectores de ataque en ciberseguridad. Esta propuesta se enfoca en generar contraseñas seguras a partir de un carácter inicial, utilizando un modelo computacional basado en:

- Un grafo ASCII dirigido con transiciones seguras.
- Una heurística constructiva para generar contraseñas robustas.
- Una metaheurística de optimización (Simulated Annealing) para refinar las soluciones.

Comparación de métodos

Método	Ejemplo	Tiempo (n=8)	Calidad
Fuerza Bruta	A1b@2c#3	> 1 millón años	Óptima
Heurística	A7@m2#Z9	15 ms	85/100
Metaheurística (SA)	A7@#Z!x3	60 ms	95/100

1. Fuerza Bruta

¿Cómo funciona?

Genera todas las combinaciones posibles según la longitud y el conjunto ASCII:

- Ejemplo: longitud 8 → 95^8 ≈ 6.6×10^{15} combinaciones
- Revisa una a una, filtrando según las reglas:
    - Al menos una mayúscula, una minúscula, un número y un símbolo
    - Sin repeticiones consecutivas
    - Sin secuencias comunes (abc, 123, qwerty)

Ejemplo:

Para 3 caracteres desde 'A':

- Aaa → Descartada (repetición)
- Aa1 → Válida
- A1@ → Válida (mejor opción)

Problema:

El crecimiento exponencial hace inviable su uso para contraseñas mayores a 6 caracteres.

2. Heurística Constructiva basada en Grafo

Modelado del espacio como grafo:

- **Nodos:** caracteres ASCII imprimibles.
- **Aristas:** transiciones permitidas según reglas de seguridad.
- **Pesos:** puntuación heurística basada en criterios como diversidad, ausencia de patrones, y seguridad.

Construcción paso a paso:

1. Se parte de un carácter inicial (por ejemplo, `A`).
2. El grafo **no se construye completo desde el inicio**. En su lugar:
    - Para cada nodo, se evalúan todos los posibles caracteres válidos como siguientes candidatos.
    - Se calculan los puntajes.
    - **Solo se crean aristas hacia los 3 vecinos con mayor puntaje**.
    - Este proceso evita la expansión innecesaria del grafo.

Ventajas:

- **Poda temprana**: rutas con puntajes negativos o con reglas violadas se descartan de inmediato.

- **Exploración eficiente**: limita cada nodo a máximo 3 vecinos con las mejores transiciones.

- **Construcción diferida**: los nodos y transiciones solo se generan cuando se necesiten.

Ejemplo:

Construcción de una contraseña de 4 caracteres desde `A`:

1. `A` → candidatos: `7` (+2), `@` (+1), `B` (-2)
    - Se elige `7`
2. `7` → candidatos: `@` (+3), `b` (+2)
    - Se elige `@`
3. `@` → candidatos: `m` (+2), `2` (+1)
    - Se elige `m`

Resultado: `A7@m` (Puntaje: 7)

3. Metaheurística: Simulated Annealing (SA)

¿Por qué usarlo?

- **Refina soluciones generadas por la heurística.**
- **Escapa de óptimos locales** explorando otras combinaciones.
- **Balance entre exploración y explotación.**

Funcionamiento:

1. Se parte de una contraseña heurística: `A7@m2#Z9`
2. Se definen mutaciones permitidas:
    - Intercambio: cambiar un carácter por otro válido.
    - Inserción/sustitución: mantener diversidad.
    - Rotación: reordenar símbolos.
3. Se evalúa la contraseña modificada:
    - Si mejora, se acepta.
    - Si es peor, se acepta con probabilidad `P = e^(ΔE/T)`.
4. Se reduce la temperatura progresivamente: `T_new = 0.95 × T_prev`

Ejemplo:

1. `A1@k3#m` (puntaje: 14)
2. Mutación → `A1@#3#m` (13) → aceptada por alta temperatura
3. Mutación → `A1@#!#m` (16) → aceptada (mejor)
4. Mutación → `A1@#!#7` (18) → aceptada

Resultado final: `A1@#!#7` (puntaje: 18 vs 14 inicial)

Ventajas:

- Se mejora la entropía de las contraseñas en pocos milisegundos.
- Se adapta bien a políticas estrictas de seguridad.