Ingeniería Matemática e Inteligencia Artificial

Memoria Cripto Chat

Práctica 2

Matemática Discreta

Sergio Herreros Pérez
Daniel Sánchez Sánchez

rsa.py

- **Generar claves**:
  Primero comprobamos que el rango dado es mayor que 2. Ahora, hacemos un primer bucle para encontrar p, que empieza en un numero aleatorio en el rango y itera cíclicamente este hasta que encuentra un primo. Si da la vuelta completa, significa que no hay ningún primo, y levanta una excepción. En caso positivo, pasamos a buscar otro primo q, pero en este caso, empezamos en un numero aleatorio en la zona no explorada por el anterior bucle, para no repetir, y hacemos dos bucles, uno hacia arriba y otro hacia abajo, cubriendo todo el rango no explorado buscando a pp. En caso de no encontrar ningún primo, levantamos una excepción. Una vez que hayamos encontrado dos primos, calculamos n y $\phi(n)$. Elegimos que nuestra e será 65537, siempre que no sea divisor de $\phi(n)$. Si lo fuese, buscamos una e que no lo sea sumando de 2 en 2 hasta encontrarla. Una vez tenemos todo, calculamos la clave privada d como $e^{-1}\left(mod\varphi(n)\right)$.

- **Aplicar padding:**
  Generamos *digitos_padding* dígitos aleatoriamente y los añadimos al final de nuestro número m.

- **Eliminar padding:**
  Eliminamos del número los últimos *dígitos_padding* dígitos.

- **Cifrar RSA:**
  Se aplica padding a m y se calcula $c = m^e(modn)$

- **Descifrar RSA:**
  Se calcula $m = c^d(modn)$ y se le pasa por eliminar padding.

- **Romper clave:**
  Factorizamos *n* usando Pollard rho con la detección de ciclos de Brent y la variación de paquete, que consiste en acumular $packet = \prod(x - y)\ (modn)$ por varias iteraciones, y luego calcular $mcd(packet, n)$, ahorrándonos calcular el mcd en cada iteración, ya que es en esta función donde recae el mayor peso. Una vez sacados p y q calculamos $\varphi(n)$, y con este la clave privada d.

- **Ataque de texto plano elegido:**
  Aprovechando las dos vulnerabilidades principales de nuestro sistema, que son que no tiene padding y que cifra letra por letra, podemos utilizar un ataque de diccionario, el en que ciframos todos los caracteres de Unicode (aunque con caracteres Alpha-numericos y signos de puntuación debería bastar para sacar casi todo el mensaje) y con ello ir letra por letra cifrada buscando a que carácter de Unicode corresponde.

- **Ataque cíclico:**
  Un posible ataque para recuperar un mensaje es el ataque cíclico. Funcionaria tanto con padding como sin él, sin embargo, tiene bastantes limitaciones. Solo es eficiente cuando el exponente es pequeño, y $\lambda(\lambda(n))$ es pequeño también.
  Este ataque se basa en encontrar un l tal que $c^{e^l} = c$ ya que esto implicaría que $c^{e^{l-1}} = m$. Sabemos que dicha $l \mid \lambda(\lambda(n))$ (https://repositorio.unican.es/xmlui/bitstream/handle/10902/16920/Ataques+al+RSA+FINAL.pdf?sequence=1 , Proposición 2.9), por lo que siempre que dicha l sea pequeña y el exponente también lo sea, podemos probar valores de l hasta que terminemos el ciclo.

criptochat.py

Nuestro criptochat.py contiene una clase CriptoChat, que se encarga de controlar la funcionalidad. Hemos orientado esta como si fuese un servicio donde te registras como usuario con una contraseña, y puedes acceder a la funcionalidad requerida. Toda la información se guarda y se carga de un fichero "users.json".

Al iniciar la interfaz, lo primero que te pide es registrarte o hacer un log in si ya lo estabas. Después ya puedes acceder a las funciones requeridas, y a alguna extra:

1. Generar un nuevo par de claves, usando la función generar_claves de rsa.py
2. Registrar un nuevo par de claves, donde puedes introducir cuales quieres que sean tus claves.
3. Añadir contacto, donde puedes añadir un contacto con su clave publica para poder enviarle mensajes en un futuro
4. Enviar mensaje, que te pide que selecciones un contacto ya registrado y escribas el mensaje que deseas enviar, y a continuación te muestra el mensaje cifrado con su clave.
5. Recibir mensaje, donde introduces un mensaje cifrado y te muestra el mensaje descifrado con tu clave privada.
6. Cambiar padding, que permite cambiar el número de dígitos de padding de nuestro sistema RSA.
7. Cambiar de usuario, que permite cambiar de usuario
8. Modificar contacto, que permite cambiar el nombre, la clave publica o eliminar un contacto
9. Salir, que cierra el programa