Cifra de Vernam

Roger Stroff Leites

Histórico

- Inventado em 1918 pelo engenheiro americano **Gilbert Vernam**, na Bell Labs. Usava uma fita de papel perfurada com a chave e combinada (XORADA) com a mensagem original, para produzir o texto cifrado.
- **Joseph Mauborgne** percebeu que se a chave fosse completamente aleatória e usada apenas uma vez, o algoritmo seria inquebrável. Essa implementação é o que hoje se chama de *One-Time Pad* (OTP).

Aplicações

- O uso prático do ***One-Time Pad*** é limitado a cenários onde a segurança absoluta é prioritária e a gestão de chaves pode ser controlada de perto.
- Durante a Guerra Fria, a Tchecoslováquia utilizou o OTP para comunicação entre agentes, bem como para o tráfico diplomático entre Praga e suas embaixadas no exterior. Para garantir a chave, eles emitiram livretos OTP, cada um com 40 páginas ou *pads* e 50 grupos de cinco dígitos em cada página.

Aplicações

- Também foi usado na criptografia das comunicações entre os aliados durante a Segunda Guerra Mundial.
- Atualmente pode servir como paradigma de teste de desempenho e segurança para outros algoritmos de criptografia.

Vantagens

- **Segurança Perfeita:** Claude Shannon provou que é matematicamente impossível quebrar o OTP, mesmo com poder computacional ilimitado, desde que sejam seguidos todos os seus princípios fundamentais (chave aleatória, do mesmo tamanho da mensagem e usada apenas uma vez).
- **Simplicidade:** O processo de cifragem e decifragem é extremamente simples, envolvendo apenas a operação XOR bit a bit (ou caractere a caractere).

Desvantagens

- **Uso Único da Chave:** Cada chave só pode ser usada uma vez. A reutilização da chave compromete toda a segurança do sistema.
- **Expansão da Chave:** O tamanho da chave pode ser enorme, proporcional ao volume total de dados a serem protegidos.
- **Gestão de Chaves:** A distribuição e gerenciamento de chaves (compartilhamento seguro prévio) representa um desafio logístico que também pode colocar em risco a segurança.

Vulnerabilidades

A única vulnerabilidade real do algoritmo de Vernam advém da **implementação incorreta** de seus princípios fundamentais, e não de falhas matemáticas no algoritmo em si.

- **Reutilização da Chave:** A principal vulnerabilidade ocorre se a chave for reutilizada para cifrar mensagens diferentes. Isso permite a análise de frequência e outros métodos criptoanalíticos para quebrar o sistema.
- **Chave Não Aleatória:** Se a chave não for verdadeiramente aleatória (por exemplo, gerada por um gerador de números pseudoaleatórios falho), a previsibilidade na chave pode ser explorada.
- **Chave Muito Curta:** Uma chave menor que a mensagem pode levar à repetição de padrões, tornando o código vulnerável.

Algoritmo em Java

```java
//Vernam XOR
public static byte[] cifrar(byte[] dados, byte[] chave) {
    byte[] xored = new byte[dados.length];
    for (int i = 0; i < dados.length; i++) {
        xored[i] = (byte) (dados[i] ^ chave[i % chave.length]);
    }
    return xored;
}
```

Algoritmo em Java

```java
//Deriva uma chave PBKDF2 a partir da senha e salt
private static byte[] gerarChave(String senha, byte[] salt, int tamanhoBits) throws Exception {
    PBEKeySpec spec = new PBEKeySpec(senha.toCharArray(), salt, 65536, tamanhoBits);
    SecretKeyFactory skf = SecretKeyFactory.getInstance("PBKDF2WithHmacSHA256");
    return skf.generateSecret(spec).getEncoded();
}
```

Algoritmo em Java

```java
//Criptografar
private static void criptografar(String arquivo, String senha) throws Exception {
    byte[] conteudo = Files.readAllBytes(Paths.get(arquivo));

    // Gera salt aleatório
    byte[] salt = new byte[16];
    new SecureRandom().nextBytes(salt);

    byte[] chave = gerarChave(senha, salt, tamanhoBits: 256);
    byte[] cifrado = cifrar(conteudo, chave);

    // Guarda salt no arquivo cifrado (base64)
    ByteArrayOutputStream out = new ByteArrayOutputStream();
    out.write(salt);
    out.write(cifrado);

    Files.write(Paths.get("arquivo_cifrado.txt"), Base64.getEncoder().encode(out.toByteArray()));
    System.out.println("Arquivo cifrado gerado: arquivo_cifrado.txt");
}
```

Algoritmo em Java

```java
//Decriptografar
private static void decriptografar(String arquivo, String senha) throws Exception {
    byte[] dadosBase64 = Files.readAllBytes(Paths.get(arquivo));
    byte[] dados = Base64.getDecoder().decode(dadosBase64);

    // Extrai o salt (primeiros 16 bytes)
    byte[] salt = new byte[16];
    System.arraycopy(dados, 0, salt, 0, 16);

    byte[] cifrado = new byte[dados.length - 16];
    System.arraycopy(dados, 16, cifrado, 0, cifrado.length);

    byte[] chave = gerarChave(senha, salt, tamanhoBits: 256);
    byte[] decifrado = cifrar(cifrado, chave);

    Files.write(Paths.get("arquivo_decifrado.txt"), decifrado);
    System.out.println("Arquivo decifrado gerado: arquivo_decifrado.txt");
}
```

Algoritmo em Java

```
Run | Debug
public static void main(String[] args) {
    try {
        if (args.length != 3) {
            System.err.println("Uso: java VernamCifra <arquivo.txt> <senha> <criptografar|decriptografar>");
            System.exit(1);
        }

        String arquivo = args[0];
        String senha = args[1];
        String modo = args[2].toLowerCase();

        if (modo.equals("criptografar")) {
            criptografar(arquivo, senha);
        }

        else if (modo.equals("decriptografar")) {
            decriptografar(arquivo, senha);
        }

        else {
            System.err.println("Modo inválido. Use 'criptografar' ou 'decriptografar'.");
            System.exit(1);
        }

    } catch (Exception e) {
        System.err.println("Erro: " + e.getMessage());
        e.printStackTrace();
    }
}
```

Referências

- https://www.cryptomuseum.com/crypto/otp/cs/index.htm
- https://security.stackexchange.com/questions/108682/is-one-time-pad-used-anywhere
- https://www.ciphermachinesandcryptology.com/en/onetimepad.htm
- https://istoedinheiro.com.br/eua-e-russia-inauguravam-o-telefone-vermelho-ha-60-anos