```
package gerenciador;

public class SdesEncryption {

	private String txt;
	private String lip;
	private String rip;
	private String sw;


	public SdesEncryption(String txt, int[] key) {


		for (int i = 0; i < txt.length(); i++) {
			String s =
Integer.toBinaryString(txt.charAt(i));
			s = eightbits(s);

			SdesKeys sdesk = new SdesKeys(key);
			int[] k1 = sdesk.getK1();
			int[] k2 = sdesk.getK2();

			String sk = intToString(k1);

			sw = fk(s,sk);

			String sk2 = intToString(k2);

			this.txt = fk(sw, sk2);


		}
	}


	public String getTxt() {
		return txt;
	}


	public String fk(String plaintext, String k) {

		String ep = ip(plaintext);
		String aux = xorwithk(k, ep);

		this.sw = aux + rip;


		return sw;

	}

	private String ip(String s) {
		String aux = "";
```

```
		aux = aux + s.charAt(1);
		aux = aux + s.charAt(5);
		aux = aux + s.charAt(2);
		aux = aux + s.charAt(0);
		aux = aux + s.charAt(3);
		aux = aux + s.charAt(7);
		aux = aux + s.charAt(4);
		aux = aux + s.charAt(6);

		this.lip = aux.substring(0, 4);
		this.rip = aux.substring(4);

		String ep = ExpansionPermution(this.rip);
		return ep;

	}

	private String ExpansionPermution(String r) {
		String aux = "";
		aux = aux + r.charAt(3);
		aux = aux + r.charAt(0);
		aux = aux + r.charAt(1);
		aux = aux + r.charAt(2);
		aux = aux + r.charAt(1);
		aux = aux + r.charAt(2);
		aux = aux + r.charAt(3);
		aux = aux + r.charAt(0);


		return aux;

	}

	private String intToString(int[] k) {
		String sk = "";

		for (int i = 0; i < 8; i++) {
			String c = String.valueOf(k[i]);
			sk = sk + c;
		}
		return sk;
	}

	private String xorwithk(String sk, String ep) {

		String aux = "";
		for (int i = 0; i < 8; i++) {
			if (sk.charAt(i) == ep.charAt(i)) {
				aux = aux + '0';
			} else {
				aux = aux + '1';
			}

		}
```

```
                return s0s1(aux);

        }

        private String s0s1(String s) {


                String line0 = "" + s.charAt(0) + s.charAt(3);
                Integer l0 = Integer.parseInt(line0, 2);
                String column0 = "" + s.charAt(1) + s.charAt(2);
                Integer c0 = Integer.parseInt(column0, 2);

                String line1 = "" + s.charAt(4) + s.charAt(7);
                Integer l1 = Integer.parseInt(line1, 2);
                String column1 = "" + s.charAt(5) + s.charAt(6);
                Integer c1 = Integer.parseInt(column1, 2);

                int[][] matrixS0 = { { 1, 0, 3, 2 }, { 3, 2, 1,
0 }, { 0, 2, 1, 3 }, { 3, 1, 3, 2 } };

                int[][] matrixS1 = { { 0, 1, 2, 3 }, { 2, 0, 1,
3 }, { 3, 0, 1, 0 }, { 2, 1, 0, 3 } };

                String s0 = Integer.toBinaryString(matrixS0[l0]
[c0]);
                String s1 = Integer.toBinaryString(matrixS1[l1]
[c1]);


                s0 = twobits(s0);
                s1 = twobits(s1);

                String p4 = s0 + s1;

                return p4xorf4l(p4);

        }

        private String p4xorf4l(String p4) {
                String aux = "";


                for (int i = 0; i < 4; i++) {
                        if (this.lip.charAt(i) == p4.charAt(i)) {
                                aux = aux + '0';
                        } else {
                                aux = aux + '1';
                        }

                }
                return aux;

        }
```

```
        private String eightbits(String binary) {
                while (binary.length() < 8) {
                        binary = '0' + binary;
                }
                return binary;

        }

        private String twobits(String binary) {
                while (binary.length() < 2) {
                        binary = '0' + binary;
                }
                return binary;

        }

}
```