Information Security HW1

分工

- 四資工三甲 B100632020 余柏葳 Encrypt
- 四資工三甲 B100632021 艾奎華 Decrypt

開發環境

- Win10
- Visual Studio 2017
- C++

程式說明

Caesar cipher

1. 先將cipherText轉小寫後減去key
2. 判斷是否超出字母範圍，有則+26循環

```
//Decrypt caesar 5 ITDTZWGJXYFSIYMJSQJYLT
string Caesar(string key, string cipherText){
        int offset = stoi(key);
        string plainText;
        char c;
        for (int i = 0; i < cipherText.size(); i++){
                c = tolower(cipherText[i]) - offset;
                c = c < 'a' ? c + 26 : c;
                plainText.push_back(c);
        }
        return plainText;
}
```

Playfair cipher

1. 建出table(重複去除、j視為i)
2. 判斷

> 同行 -> 上一格
> 同列 -> 左一格
> 其他 -> 對角線

```
//Decrypt playfair COMP IDPWQSDFTUGURFBKHNSFAD
void Find(int& x, int& y, const char table[5][5], const char& c){
	for (x = 0; x < 5; x++)
		for (y = 0; y < 5; y++)
			if (table[y][x] == c) return;
}

string Playfair(string key, string cipherText){
	string plainText;
	const char alphabets[26] = "ABCDEFGHIKLMNOPQRSTUVWXYZ";
	int count = 0, fill = 0;
	bool repeated[25] = { false };
	char table[5][5];

	for (int i = 0; i < 5; i++){
		for (int j = 0; j < 5; j++){
			if (count == key.length()){
				while (repeated[fill])
					fill++;
				table[i][j] = alphabets[fill++];
			}
			else{
				while (repeated[key[count] - 'A']) count++;
				table[i][j] = (key[count] == 'J') ? 'I' : key[count];
				repeated[(key[count] > 'I') ?
				    key[count++] - 'B' : key[count++] - 'A'] = true;
			}
		}
	}

	for (int i = 0; i < cipherText.length(); i += 2){
		int x1 = 0, y1 = 0, x2 = 0, y2 = 0;
		Find(x1, y1, table, cipherText[i]);
		Find(x2, y2, table, cipherText[i + 1]);

		if (x1 == x2){
			plainText.push_back(table[y1 ? y1 - 1 : 4][x1] + 32);
			plainText.push_back(table[y2 ? y2 - 1 : 4][x2] + 32);
		}
		else if (y1 == y2){
			plainText.push_back(table[y1][x1 ? x1 - 1 : 4] + 32);
			plainText.push_back(table[y2][x2 ? x2 - 1 : 4] + 32);
		}
		else{
			plainText.push_back(table[y2][x1] + 32);
			plainText.push_back(table[y1][x2] + 32);
		}
	}
	return plainText;
}
```

Vernam proposed the autokey system

1. 將key跟cipherText的首個字母轉成五位數的bits並xor
2. 將得出的數字加進plainText跟key
3. 重複執行直到cipherText全部試完

```
//Decrypt vernam TEC QK[N[JPQDSE`QTKH_MA_NK
string Vernam(string key, string cipherText){
	string plainText;
	bitset<5> cipherBits;
	bitset<5> keyBits;
	for (int i = 0; i < cipherText.length(); i++){
		keyBits = key[i] - 'A';
		cipherBits = cipherText[i] - 'A';
		//cout << keyBits << "  " << cipherBits << endl;
		char c = (keyBits^cipherBits).to_ulong();
		c = c > 25 ? c + 'a' - 26 : c + 'a';
		//cout << c << endl;
		key.push_back(c);
		plainText.push_back(c);
	}
	return plainText;
}
```

Row transposition

1. 依照key回推每個row的長度
2. 將cipherText依長度分開
3. 將分開後的字串依key的順序放回

```
//Decrypt row 45362178 RTOUDGYAEDSNOTLONTBHEE
string Row(string key, string cipherText){
        const int cipherLen = cipherText.length();
        const int keyLen = key.length();
        int div = cipherLen / keyLen;
        int mod = cipherLen % keyLen;
        string plainText;
        map<int, string> table;
        int index = 0;
        int count = mod > 0 ? div + 1 : div;
        for (int i = 0; i < keyLen; i++){
                table.insert(make_pair(i, cipherText.substr(index, count)));
                index += count;
                count = --mod > 0 ? div + 1 : div;
        }
        for (int i = 0; i < cipherLen; i++){
                plainText.push_back(table[key[i % keyLen] - '1'][0] + 32);
                table[(key[i % keyLen] - '1')].erase(0, 1);
        }
        return plainText;
}
```

Rail fence cipher

1. 依照key建出每個字母的index
2. 依照index的順序放回字母

```
//Decrypt rail_fence 2 DYUBSADHNEGOORETNTELTO
string RailFence(string key, string cipherText){
        string plainText;
        vector<vector<int>> rails(stoi(key));
        int nowIndex = 0;
        int offset = 1;
        for (int i = 0; i < cipherText.length(); i++){
                rails[nowIndex].push_back(i);
                if (nowIndex == stoi(key) - 1) offset = -1;
                if (nowIndex == 0) offset = 1;
                nowIndex += offset;
        }
        plainText.resize(cipherText.length());
        nowIndex = 0;
        for (int i = 0; i < rails.size(); i++)
                for (int j = 0; j < rails[i].size(); j++)
                        plainText[rails[i][j]] = cipherText[nowIndex++] + 32;
        return plainText;
}
```