



POLITECNICO
MILANO 1863

Bombenigma: a simulation of WW2 cryptography

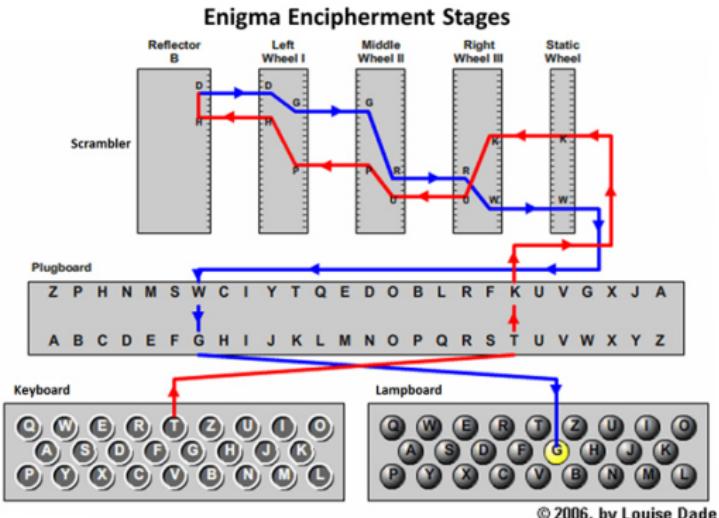
Historical recreation and HPC "bruteforce" solution

Livia Giacomin

September 8, 2023

The problem: Enigma machines

1/20



© 2006, by Louise Dade

Solving the problem

2/20

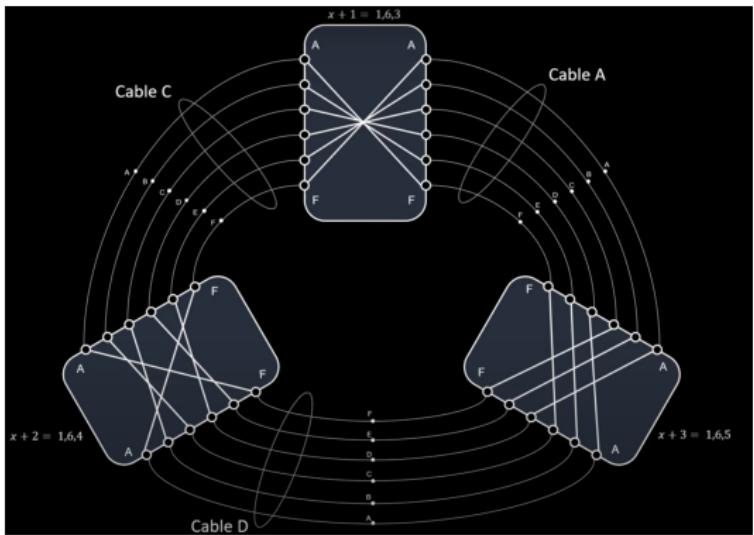
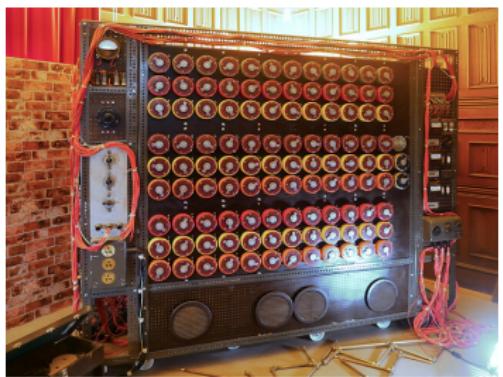
What were the Enigma's weaknesses? What were cribs?

WJSIOAWOTHFULAGRDPNTAZXBBOEBIMGAFRWOR
WEATHERISCLEARHEILHITLER

WJSIOAWOTHFULAGRDPNTAZXBBOEBIMGAFRWOR
WEATHERISCLEARHEILHITLER

The original solution: the Bombe

3/20



Simulating Enigma: the plugboard/reflector class

4/20

```
class Plug_reflect{

private:
    std::vector<int> input_pair1;
    std::vector<int> input_pair2;

public:
    Plug_reflect(const char* path);
    Plug_reflect(std::vector<int>);

    std::vector<int> getpair1();
    std::vector<int> getpair2();
    int map(int input);
};
```

Simulating Enigma: the rotor class

5/20

```
class Rotor{  
private:  
    int curr_pos;  
    int prev_pos;  
    int contacts[ALPH_LEN];  
    int notch;  
  
public:  
    Rotor(const char* path, int start_position);  
    int getCurrentPosition();  
    int getPreviousPosition();  
    void rotate();  
    void changePos(int pos);  
    int shiftUp(int input_index);  
    int shiftDown(int input_index);  
    int mapForward(int input_index);  
    int mapBackward(int contact);  
    bool isItNotch();  
    vector<int> getNotchAndPos();  
    vector<int> getContacts();  
};
```

Simulating Enigma: the enigma class

6/20

```
class Enigma{
private:
vector<Rotor> rotors;
vector<int> rotor_positions;
int num_of_rotors;
Plug_reflect *plugboard=nullptr;
Plug_reflect *reflector=nullptr;

public:
Enigma(int argc, char** argv);
Enigma(vector<int> rotor_indexes, vector<int> rotor_pos, vector<int> plugboard, int reflector_index);
void PlugboardConfig(const char* path, vector<int>& contacts);
void PlugboardConfig2(const char* path, vector<int>& contacts);
void ReflectorConfig(const char* path, vector<int>& contacts);
void RotorConfig(const char* path, vector<int>& contacts);
void RotorPositionConfig(const char* path);
bool PlugboardCheck(const char* path, fstream& in_stream, int& index_num);
bool RangeCheck(int num);
int AppearedBefore(vector<int> contacts, int num, int position);
void timetravel(int time);
void set_pos(vector<int> pos);
void encryptMessage(char& letter);
void printEnigma();
};

};
```

Useful functions: getcribs

7/20

```
vector<string> getCribs(string output, string key){  
    int len = key.size();  
    int len2 = output.size();  
    int check=0;  
    string crib="";  
    vector<string> cribs;  
    for (int i=0; i<len2-len+1; i++){  
        for (int j=0;j<len; j++){  
            if (output[i+j]==key[j]) check = 1;  
            crib += output[i+j];  
        }  
        if (check == 0) cribs.push_back(crib);  
        crib = "";  
        check = 0;  
    }  
    return cribs;  
}
```

Useful structures: codePairs

8/20

```
struct codePair{
    char letter1;
    char letter2;
    int pos;
const {
    if (letter1 != other.letter1) {
        return letter1 < other.letter1;
    } else if (letter2 != other.letter2) {
        return letter2 < other.letter2;
    } else {
        return pos < other.pos;
    }
}
bool operator==(const codePair& other) const {
    return letter1 == other.letter1 && letter2 == other.letter2 && pos == other.pos;
}
};

set<codePair> readMatches(string key, string crib, int cribPos){
    set<codePair> codePairs;
    int len = key.size();
    for (int i=0; i<len;i++){
        if (key[i]<=crib[i]) codePairs.insert({key[i], crib[i],i+cribPos});
        else codePairs.insert({crib[i], key [i],i+cribPos});
    }
    return codePairs;
}
```

Useful functions: findLoop

9/20

```
vector<codePair> findLoop(set<codePair> pairs){
    vector<codePair> loop;
    char temp;
    for (codePair pair1 : pairs){
        if (!loop.empty()) break;
        for (codePair pair2 : pairs){
            temp=pair1.letter2;
            if (!(pair1==pair2)){
                if (pair2.letter1==temp) temp=pair2.letter2;
                else if (pair2.letter2==temp) temp=pair2.letter1;
                else continue;
                for (codePair pair3 : pairs){
                    if (!(pair3==pair2)&&!(pair3==pair1)){
                        if (pair3.letter1==temp) {
                            if (pair1.letter1==pair3.letter2){
                                loop.push_back(pair1);
                                loop.push_back(pair2);
                                loop.push_back(pair3);
                                return loop;
                            }
                            else if (pair3.letter2==temp) {
                                if (pair1.letter1==pair3.letter1){
                                    loop.push_back(pair1);
                                    loop.push_back(pair2);
                                    loop.push_back(pair3);
                                    return loop;
                                }
                            }
                        }
                    }
                }
            }
        }
    }
    return loop;
}
```

Bombe code: finding the crib and loop

10/20

```
int main(int argc, char** argv){
    vector<string> outputs = encodeFullMessages(argc, argv);
    for (string output : outputs){
        string key= getKeyFromFile();
        vector<string> cribList = getCribs(output, key);
        int n_crib=cribList.size();
        cout << "number of cribs found: " <<n_crib<<endl;
        if (n_crib==0) {
            cout << "There are no cribs, so our Bombe cannot decode this message."; return 0; }
        vector<codePair> loop;
        int cribPos;
        for (string crib : cribList){//we try for all cribs
            cribPos=output.find(crib);
            set<codePair> codePairs = readMatches(key,crib,cribPos);
            loop = findLoop(codePairs);
            if (!loop.empty()) {
                cout << "Loop of three pairs of letters found in the crib " << crib <<": " ;
                cout << "(crib position "<<cribPos<< ") ";
                for (const codePair& pair : loop) {
                    cout << "(" << pair.letter1 << ", " << pair.letter2 << ", "<< pair.pos <<") ";
                }
                cout << endl;
            }
            if (loop.empty()){
                cout << "No loops found in the crib "<<crib<<endl;
                continue;
            }
        }
    }
}
```

Bombe code: executing the machine

11/20

```
vector<int> rotor_indexes = {1, 2, 3};  
int reflector_index = 1;  
Enigma *enigma = startEnigma(rotor_indexes, {}, reflector_index);  
string trial;  
int check=0;  
for (int i=0;i<ALPH_LEN;i++){  
    for (int j=0;j<ALPH_LEN;j++){  
        for (int k=0;k<ALPH_LEN;k++){  
            trial={};  
            trial+=’A’;  
            enigma->set_pos({i,j,k});  
            enigma->timetravel(loop[0].pos+cribPos);  
            trial+=encode(enigma,trial).back();  
            enigma->set_pos({i,j,k});  
            enigma->timetravel(loop[1].pos+cribPos);  
            trial+=encode(enigma,trial).back();  
            enigma->set_pos({i,j,k});  
            enigma->timetravel(loop[2].pos+cribPos);  
            trial+=encode(enigma,trial).back();
```

Bombe code: executing the machine

12/20

```
if (trial.back()=='A') {
    check=1;
    cout << "potential settings found with rotor positions "<<i<<" "<<j<<" "<<k<<".<<endl<<
    "The plugboard settings would be: "<< endl;
    vector<char> loop_plugs={};
    loop_plugs.push_back(loop[0].letter2);
    if (loop[0].letter2!=loop[1].letter1) loop_plugs.push_back(loop[1].letter1);
    else loop_plugs.push_back(loop[1].letter2);
    loop_plugs.push_back(loop[0].letter1);
    for (int m=0; m<loop_plugs.size();m++){
        cout << loop_plugs[m]<<" "<<trial[m]<< " // ";
    }
    cout<<endl;
    break;
}
}
}
}
if (check==0) cout << "No potential settings found with this crib."<<endl;
delete enigma;
}
```

Bombe code: output snippet

13/20

```
The message we try to decode is GVNJQZCZMXQJFSSTBLQRKRVXWMANBXZRXAB
number of cribs found: 11
No loops found in the crib VNJQZCZMXQJFSSTBLQR
No loops found in the crib JQZCZMXQJFSSTBLQRKR
No loops found in the crib QZCZMXQJFSSTBLQRKRV
Loop of three pairs of letters found in the crib ZCZMXQJFSSTBLQRKRVX: (crib position 5)
(0, T, 15) (T, X, 9) (0, X, 23)
potential settings found with rotor positions 0 0 22.
The plugboard settings would be:
T A // X I // O E //
potential settings found with rotor positions 0 1 8.
The plugboard settings would be:
T A // X Z // O P //
potential settings found with rotor positions 0 2 12.
The plugboard settings would be:
T A // X E // O H //
potential settings found with rotor positions 0 4 4.
The plugboard settings would be:
T A // X R // O G //
potential settings found with rotor positions 0 5 9.
The plugboard settings would be:
T A // X B // O G //
```

Did the Bombe always work?
Does this code guarantee a solution?

The power of bruteforce

15/20

Modern problems require modern solutions: considering all plugboard cases

```
ofstream outFile("pluglist.txt");
for (int i=0;i<ALPH_LEN;i++){
    for (int j=0;j<i;j++){
        temp.first=i;
        temp.second=j;
        plugs.push_back(temp); }
    }
vector<vector<plug>> plugboards;
vector<plug> plugboard;
int len=plugs.size();
for (int i=0;i<len;i++){
    for (int j=0;j<i;j++){
        if (!shareletter(plugs[i],plugs[j])){
            for (int k=0;k<j;k++){
                if (!shareletter(plugs[k],plugs[j])&&!shareletter(plugs[k],plugs[i])){
                    plugboard={};
                    plugboard.push_back(plugs[k]);
                    plugboard.push_back(plugs[j]);
                    plugboard.push_back(plugs[i]);
                    plugboards.push_back(plugboard);
                    outFile << plugboard[0].first<<" "<<plugboard[0].second<<
                    "<<plugboard[1].first<<" "<<plugboard[1].second<<" "<<plugboard[2].first
                    <<" "<<plugboard[2].second<<endl;
                }}}}}}
```

Bruteforce solution: setup

16/20

```
int main(int argc, char** argv){  
    auto start_time = chrono::high_resolution_clock::now();  
    vector<string> outputs = encodeFullMessages(argc, argv);  
    string output=outputs.front();  
    vector<vector<int>> plugboards={};  
    vector<int> plugboard;  
    string line;  
    ifstream pluglist("pluglist.txt");  
    while (getline(pluglist, line)) {  
        plugboard=readplug(line);  
        plugboards.push_back(plugboard);}  
    string key= getKeyFromFile();  
    vector<string> cribList = getCribs(output, key);  
    reverse(cribList.begin(), cribList.end());  
    int n_cribs=cribList.size();  
    int cribPos;
```

Bruteforce: HPC iterations

17/20

```
int threads=min(n_cribes,8);
#pragma omp parallel for num_threads(threads) shared(cribList) private(cribPos)
for (string crib : cribList){//we try for all cribs
    cribPos=output.find(crib);
    vector<int> rotor_indexes = {1, 2, 3};
    int reflector_index = 1;

#pragma omp parallel for num_threads(threads) private(plugboard)
for (vector<int> plugboard:plugboards) {
    Enigma *enigma = startEnigma(rotor_indexes,plugboard,reflector_index);
    string decoded;
    for (int i=0;i<ALPH_LEN;i++){
        for (int j=0;j<ALPH_LEN;j++){
            for (int k=0;k<ALPH_LEN;k++){
                enigma->set_pos({i,j,k});
                enigma->timetravel(cribPos);
                decoded=encode(enigma,crib);
                if (decoded==key){
                    ...
                }
            }
        }
    }
}
```

Bruteforce: HPC iterations

18/20

```
if (decoded==key){  
    auto end_time = chrono::high_resolution_clock::now();  
    cout<<"Found solution at rotor positions "<<i<<" "<<j<<" "<<k<<". "  
<<endl<<"with plugboard settings ";  
    for (int a : plugboard){  
        char letter = 'A' + a;  
        cout << letter << " ";  
    cout<< endl;  
    enigma->set_pos({i,j,k});  
    cout<<"Decoded messages: "<<endl;  
    for (string code : outputs){  
        cout<<encode(enigma,code)<<endl;  
    }  
    delete enigma;  
  
    auto elapsed_time = chrono::duration_cast<chrono::seconds>(end_time - start_time);  
    cout << "Time taken: " << elapsed_time.count() << " seconds" << endl;  
    exit(0);  
}
```

Found solution at rotor positions 25 1 8.

with plugboard settings D A F C G B

Decoded messages:

LEZIONISOSPESEALPOLITECNICODIMILANO

CLIMATIZZATORIACCESIALPOLITECNICODIMILANO

STUDENTIFUORISEDEAFFOLLANOILPOLITECNICODIMILANO

ATTACCOPROGRAMMATOINPIAZZALEODAVANTIALPOLITECNICODIMILANO

Time taken: 32 seconds

Why was the Bombe relevant? What has it changed in the history of computing?

Thanks for your attention!