CTF – UNE HISTOIRE DE COLOSSE

Cryptographie - Stéganographie

Table des matières

1. Cryptographie – Partie 1 3

1.1. Présentation de la mission 3
1.2. Compétences CTF et BTS 4
1.3. Démarche de résolution de la mission 4
1.3.1. Calculs mathématiques et clé de chiffrement 4
1.3.2. Code Baudot 5
1.3.3. Programme déchiffrement 5

2. Stéganographie – Partie 2 7

2.1. Présentation de la mission 7
2.2. Compétences CTF et BTS 7
2.3. Démarche de résolution de la mission 7

3. Annexes 9

3.1. Document « NOTES » - fourni par le CTF 9
3.2. Document « PHOTOCOPIE » - fourni par le CTF 10
3.3. Script du programme decode Lorenz 11

Une histoire de colosse

CRYPTOGRAPHIE – STÉGANOGRAPHIE

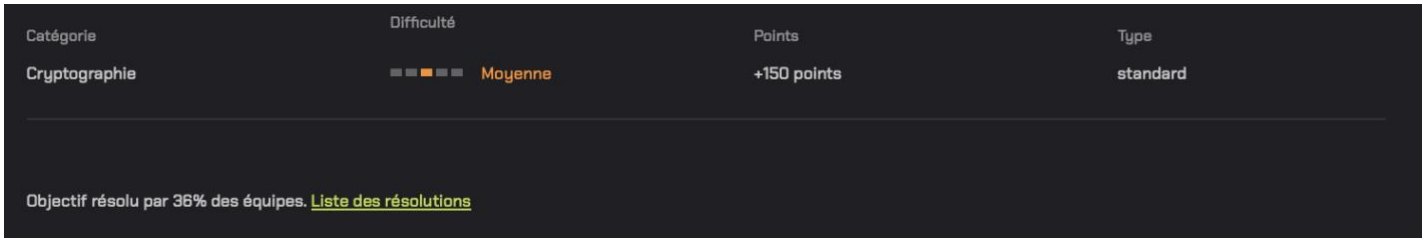
Les missions « Une histoire de colosse » sont séparées en deux parties. Elles concernent toutes deux un chiffrement effectué par la machine de Lorenz, très puissante et utilisée lors de la Seconde Guerre Mondiale pour des échanges militaires allemands de haute importance. Par ailleurs, l'ordinateur Colossus est le premier calculateur électronique fondé sur le système binaire et construit par des chercheurs de Bletchey Park afin de déchiffrer les messages codés envoyés par des machines allemandes à cette même période.

Nos missions vont être de déchiffrer le contenu de ce message à l'aide du chiffrement de Lorenz, aussi appelé le chiffrement de Tunny, afin de découvrir le message secret de l'armée adressé à un ancien officier anglais dans les années 1940.

Afin de mener à bien nos missions de la plus haute importance, nous utiliserons des méthodes de cryptographie et de stéganographie.
Si nous réussissons, deux flags nous seront communiqués dans ce message secret, nous permettant ainsi de valider ces deux missions.

1. Cryptographie – Partie 1

1.1. Présentation de la mission



Une histoire de colosse - Partie 1

La cible de votre mission se trouve devant vous : une vieille bâtisse de l'époque Victorienne, au sud de Londres.

Votre enquête vous mène à l'intérieur du bâtiment ; vous avez reçu l'ordre de récupérer les effets personnels d'un ancien officier anglais, qui aurait gardé des documents de l'armée chez lui.

Vous ne trouvez rien de très intéressant dans un premier temps, les pièces principales et le grenier semblent calmes et rangées.

Un détail cependant vous marque : un coffre en bois, un peu en retrait, semble fermé par un cadenas. Vous réussissez à l'ouvrir avec un peu de mal en forçant.

Surprise ! A l'intérieur se trouve une pochette contenant des documents... Assez étrange. Une suite de chiffres et de lettres sans vraiment de sens.

Vous décidez d'en prendre une pour analyse, avec ce qui semble être une documentation pour déchiffrer le contenu.

Voici les documents récupérés : NOTES.pdf et PHOTOCOPIE.pdf Peut-être que ce code secret cache des informations sur les documents que vous cherchez ?

1.2. Compétences CTF et BTS

Dans cette mission, il nous est demandé de déchiffrer un document qui a été chiffré à l'aide de la machine de Lorenz, utilisée lors de la Seconde Guerre mondiale pour crypter les communications militaires allemandes de haute importance. Réputée beaucoup plus complexe et encore mieux protégée que la machine Enigma, elle était utilisée pour les communications entre Hitler et ses généraux.
Cette mission fait appel à l'usage de la cryptographie, discipline employée pour protéger des messages en les cryptant et en utilisant généralement des clés de chiffrement. Pour la résolution de cette mission, différentes compétences du référentiel du BTS Services Informatiques aux Organisations ont été mises en application :

- Travailler en mode projet
- Protéger les données à caractère personnel
- Préserver l'identité numérique de l'organisation

1.3. Démarche de résolution de la mission

Pour cette mission, les deux documents mis à notre disposition sont :

- Le document anglais « Notes », qui nous explique comment le chiffrement de la machine de Lorenz fonctionne, avec les variables définies
- Le document « Photocopie », qui est le document crypté reçu par l'ancien officier anglais lors de la Seconde Guerre Mondiale

Ces documents sont par ailleurs disponibles en annexes à la fin de ce document.

1.3.1. Calculs mathématiques et clé de chiffrement

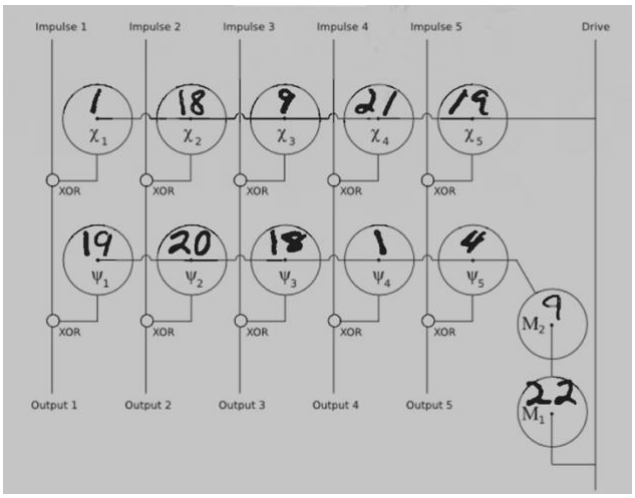
Les informations suivantes ont été données dans le document « Notes » :

The five Chi-Wheels and Psi-Wheels both produce 5-bit teleprinter letters which are XORed onto the incoming teleprinter message as shown in figure 1. Thus applying the theory of additive ciphers gives immediately as the keystream

$$K = \chi \oplus \psi'$$

and the encrypted message is

$$Z = K \oplus C.$$



Les variables sont également définies dans ce document, dans lequel les Chi et les Psi sont représentés.

On peut alors tenter de trouver la clé de chiffrement, pour laquelle on dispose de cette information :

K = chi XOR psi

Comme mentionné dans le texte, les caractères sont tous représentés en binaire, soit en base deux.

On commence alors par convertir chacune des valeurs Chi et Psi en binaire, ce qui nous donne :

Chi binaire	1 : 00001	18 : 10010	9 : 01001	21 : 10101	19 : 10011
Psi binaire	19 : 10011	20 : 10100	18 : 10010	1 : 00001	4 : 00100
Chi XOR Psi	10010	00110	11011	10100	10111

On obtient donc la clé suivante : **K = 10010 00110 11011 10100 10111**

Dans la deuxième formule : **Z = K XOR C**
Le K correspond à la clé qu'on vient de trouver, le Z est la valeur chiffrée qu'on souhaite connaître, et le C correspond à la valeur claire avant d'avoir été chiffrée.
Il faudrait alors faire un calcul inversé pour pouvoir déchiffrer les caractères présents dans le document, en passant toujours par le binaire.

1.3.2. Code Baudot

Dans le document « Notes », le code Baudot est abordé. Après une recherche sur internet, voici la grille du Code Baudot pour les différents caractères :

Nr.	Code	Char.	Z	Nr.	Code	Char.	Z	Nr.	Code	Char.	Z	Nr.	Code	Char.	Z
1	00011	A	-	9	00110	I	8	17	10111	Q	1	25	10101	Y	6
2	11001	B	?	10	01011	J	BEL	18	01010	R	4	26	10001	Z	/
3	01110	C	:	11	01111	K	(	19	00101	S	'	27	01000	CR	
4	10001	D	‡	12	10010	L	)	20	10000	T	5	28	00010	LF	
5	00001	E	3	13	11100	M	.	21	00011	U	7	29	11111	Character	
6	01101	F		14	01100	N	,	22	11110	V	=	30	11011	Digits/Signs	
	11010	G		15	11000	O	9	23	10011	W	2	31	00100	Space	
	10100	H		16	10110	P	0	24	11101	X	/	32	00000	not occupied	

On nous parle également de « Cross and Dot » dans le document, où une croix représenterait un 1, et un point représenterait un 0. Dans ce cas, il faut alors inverser les caractères dans le code Baudot (les 0 et les 1), ce qui nous donne la chose suivante :

A -	11100
B ?	00110
C :	10001
D	01110
E 3	11110
F	10010
G	00101
H	01011
I 8	11001

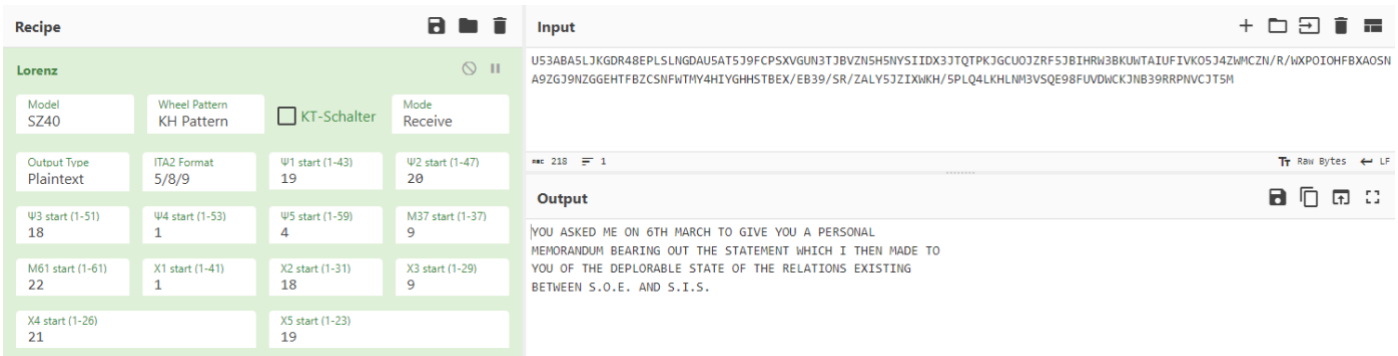
J	10100
K	10000
L	01101
M	00011
N	10011
O 9	00111
P 0	01001
Q 1	01000
R 4	10101

S	11010
T 5	01111
U 7	11100
V =	00001
W 2	01100
X	00010
Y 6	01010
Z	01110

1.3.3. Programme déchiffrement

Finalement, dans un concours de cybersécurité, tout est solvable sur Internet, sans passer par de longs calculs mathématiques à la main… On a pu trouver un programme présent sur GitHub, nous permettant de déchiffrer totalement le texte, après avoir renseigné les variables de cette manière.
Il faut bien faire attention à ces catégories qui changent selon le bloc à déchiffrer :

- Model : SZ40 pour les blocs 1 et 3 / SZ42a pour les blogs 2 et 4
- Wheel Pattern : KH Pattern pour les blocs 1 et 2 / ZMUG Pattern pour les blocs 3 et 4
- ➔ Ce message d'erreur peut s'afficher « Invalid ITA2 character : Carriage Return », penser à bien supprimer chaque espace et retour à la ligne dans le texte chiffré avant d'essayer de le déchiffrer



Le script du programme est disponible en annexe à la fin du document.
Ce programme nous a permis de déchiffrer le texte dans son intégralité, ce qui nous donne :

YOU ASKED ME ON 6TH MARCH TO GIVE YOU A PERSONAL
MEMORANDUM BEARING OUT THE STATEMENT WHICH I THEN MADE TO
YOU OF THE DEPLORABLE STATE OF THE RELATIONS EXISTING
BETWEEN S.O.E. AND S.I.S.

FLAG - 0QP40P9W
I SHOULD LIKE TO SAY AT ONCE THAT MY PERSONAL
RELATIONSHIP WITH THE HEAD OF S.I.S. AND HIS CHIEF OFFICER
(A.C.S.S.) HAS ALWAYS BEEN VERY GOOD INDEED, AND THROUGHOUT
THE TWO ORGANISATIONS THERE ARE INSTANCES WHERE
CORRESPONDING
SECTIONS HAVE MANAGED TO GET ALONG VERY WELL TOGETHER - BUT
GENERALLY SPEAKING I HAVE NOT THE SLIGHTEST HESITATION IN
STATING QUITE CATEGORICALLY THAT THE GENERAL KEY WORD FROM
TOP TO BOTTOM IN THE S.I.S. ORGANISATION HAS BEEN TO DELAY
RATHER THAN TO EXPEDITE THE NATURAL EXPANSION OF S.O.E.

ALTHOUGH I CANNOT JUSTIFY SUCH AN ATTITUDE ON THE PART
OF S.I.S., I CAN AT THE SAME TIME UNDERSTAND TO A CERTAIN
EXTENT THEIR POINT OF VIEW, NAMELY THAT WHEREAS THEY
NATURALLY DESIRE QUIET WATERS IN WHICH TO FISH FOR INTELLIGENCE,
THE ACTIVITIES OF S.O.E. IN THE SAME REGIONS OR DISTRICTS
INEVITABLY STIR UP TROUBLE. CONSEQUENTLY THERE IS A
FUNDAMENT AL CLASH OF INTERESTS IN WHICHEVER PART OF THE
WORLD THE TWO ORGANISATIONS ARE WORKING.

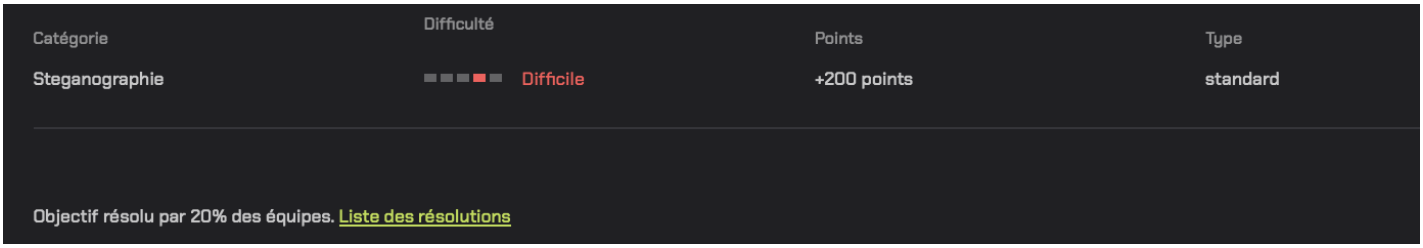
19(5)/2(5)/5(2)/26(1)/47(1)/2(2)/7(2)/2(5)/59(1)/2(4)/24(1)/9(1)/59(1)/24(1)/81(2)/5(2)/
10(1)
WHEN WE BEGAN TO GET INTO OUR STRIDE AS A NEW ORGANISATION
TOWARDS THE END OF 1940 THE S.I.S. ATTITUDE WAS THAT WE WERE
A RATHER INEFFECTIVE AND RIDICULOUS COLLECTION OF AMATEURS,
WHO MIGHT ENDANGER S.I.S. AND ALL THEIR WORKS IF WE WERE NOT
KEPT QUIET SOMEHOW. NOW THEIR ATTITUDE APPEARS TO BE THAT
WE ARE DANGEROUS RIVALS AND THAT IF WE ARE NOT SQUASHED
QUICKLY WE SHALL EVENTUALLY SQUASH THEM.

Nous avons alors le texte déchiffré totalement, qui correspond au message militaire envoyé à l'ancien officier anglais.
Le flag est donc :

FLAG - 0QP40P9W

2. Stéganographie – Partie 2

2.1. Présentation de la mission



Une histoire de colosse – Partie 2

Les deux documents étaient bien liés ! Mais malheureusement, aucune information sur ce que vous cherchiez au départ.

Vous décidez de rendre compte de vos découvertes à la hiérarchie. Leur curiosité est attisée, ils vous demandent de comprendre le document chiffré dans sa totalité.

A votre grande surprise, dans le dernier bloc, un code secret est présent... Il semble être lié au deuxième bloc de texte déchiffré.

Peut-être est-ce au final ce que vous cherchiez depuis le début ?

2.2. Compétences CTF et BTS

Dans cette mission, il nous est demandé de déchiffrer un document qui a été chiffré à l'aide de la machine de Lorenz, utilisée lors de la Seconde Guerre mondiale pour crypter les communications militaires allemandes de haute importance. Réputée beaucoup plus complexe et encore mieux protégée que la machine Enigma, elle était utilisée pour les communications entre Hitler et ses généraux.
Cette mission fait appel à l'usage de la stéganographie, discipline qui dissimule discrètement de l'information dans un autre média.
Pour la résolution de cette mission, différentes compétences du référentiel du BTS Services Informatiques aux Organisations ont été mises en application :

- Travailler en mode projet
- Protéger les données à caractère personnel
- Préserver l'identité numérique de l'organisation

2.3. Démarche de résolution de la mission

Dans cette mission, il nous est indiqué qu'un code secret est présent dans le dernier bloc du texte déchiffré, et qu'il serait directement lié au deuxième bloc de ce même texte. En regardant le texte déchiffré, on s'aperçoit qu'une ligne de nombres est présente, étrange et peu commune. On peut alors penser qu'il s'agit du fameux code secret, le voici :

19(5)/2(5)/5(2)/26(1)/47(1)/2(2)/7(2)/2(5)/59(1)/2(4)/24(1)/9(1)/59(1)/24(1)/81(2)/5(2)/10(1)

On remarque que les nombres présents ici sont compris entre 2 et 81, et les chiffres entre parenthèses entre 1 et 5. On pourrait alors penser que ces nombres correspondent à un endroit précis dans ce texte, par exemple :

- Le premier chiffre pourrait correspondre à un numéro de ligne, et le second entre parenthèses à un numéro de mot présent sur cette ligne,
  Exemple pour 19(5) : le cinquième mot de la dix-neuvième ligne du texte, qui correspond au mot « IN »

- Le premier chiffre pourrait correspondre à un numéro de mot, et le second entre parenthèses à un numéro de lettre présente dans ce mot, en comptant dans le deuxième bloc du texte,
  Exemple pour 19(5) : la cinquième lettre du dix-neuvième mot du texte, qui correspond à la lettre « F » du mot « CHIEF »

Après l'étude de ces deux options différentes possibles, on peut se rendre compte que la première ne fonctionnerait pas car le texte déchiffré ne comporte que 32 lignes et que les chiffres du code secret s'étendent jusqu'à 81. De plus, cette solution ne serait pas cohérente avec l'indice de l'énoncé, qui est le suivant :

« Un code secret est présent... Il semble être lié au deuxième bloc de texte déchiffré. »

On considère alors que cette première solution pour résoudre l'énigme n'est pas la bonne, et on s'intéresse désormais à la seconde.
On peut s'apercevoir que le deuxième bloc du texte contient 86 mots au total, en considérant que la première ligne n'appartient pas à ce bloc, soit la ligne suivante : « FLAG - 0QP40P9W »
Ainsi, l'hypothèse du nombre de mots pourrait fonctionner.

En effectuant la même démarche pour chacun des nombres du code secret, on peut obtenir cette première suite de lettres :

FLAGBHNLCUBMCHAP

Les premières lettres « FLAG » au début nous laissent penser qu'on est sur la bonne voie, mais cette suite de lettres ne correspond pas au FLAG demandé pour résoudre la mission.

En observant la typologie d'écriture du FLAG de la première partie de « Une histoire de colosse » : FLAG - 0QP40P9W, on remarque qu'un tiret est présent entre les lettres « FLAG » et les autres.
Si l'on regarde plus attentivement le deuxième bloc du texte déchiffré, on remarque qu'un même tiret y est présent. Il est placé juste après le quarante-sixième mot de ce bloc. On pourrait alors tenter de prendre en compte ce tiret comme un caractère à part entière, et considérer qu'il est alors le quarante-septième « mot » de ce bloc, il correspondrait ainsi au 47(1).
De cette manière, toutes les lettres correspondant à des nombres supérieurs à 47 dans le code seraient faussées, on les modifie en les comptant à nouveau pour obtenir un nouveau FLAG qui correspondrait davantage à celui attendu pour la résolution de cette mission.

I SHOULD LIKE TO SAY AT ONCE THAT MY PERSONAL
RELATIONSHIP WITH THE HEAD OF S.I.S. AND HIS CHIEF OFFICER
(A.C.S.S.) HAS ALWAYS BEEN VERY GOOD INDEED, AND THROUGHOUT
THE TWO ORGANISATIONS THERE ARE INSTANCES WHERE
CORRESPONDING
SECTIONS HAVE MANAGED TO GET ALONG VERY WELL TOGETHER - BUT
GENERALLY SPEAKING I HAVE NOT THE SLIGHTEST HESITATION IN
STATING QUITE CATEGORICALLY THAT THE GENERAL KEY WORD FROM
TOP TO BOTTOM IN THE S.I.S. ORGANISATION HAS BEEN TO DELAY
RATHER THAN TO EXPEDITE THE NATURAL EXPANSION OF S.O.E.

Le FLAG est donc :

FLAG – HNLQUBMQBXAP

3. Annexes

3.1. Document « NOTES » - fourni par le CTF

Report from Stanford University.
Added on the encryption scheme my personal notes.

One of the goals of the codebreakers at Bletchley Park was to break this Tunny machine to decrypt the messages of the German Army. Unfortunately this machine was much stronger than the known Enigma machine.

The Lorenz Shluessel zusatz consists of 12 wheels. The message to be encrypted or decrypted comes in as Baudot teleprinter code. Each bit is either a pulse or the absence of a pulse representing a cross or dot (1 or 0). Each character is decomposed into its five bits which are processed in parallel. Wheels are used to generate a pseudo-random keystream which is XORed with the plaintext bits. Each wheel has pins on it where each pin can represent two states, either a one or a zero. Thus a wheel is a bit sequence.

The Lorenz machine has two sets of wheels, a regularly stepping set called the Chi-Wheels and an irregularly stepping set called the Psi-Wheels. The wheels all have different periods which are relatively prime to obtain the largest possible number of wheel setting combinations. This ensures that the maximum length of a message is the product of the wheel lengths which is for the Chi-Wheels 23 x 26 x 29 x 31 x 41 = 22,041,682. This is long enough to ensure that a message is not encrypted more than once with the same keystream

The irrugularly stepping Psi-Wheels are controlled by two motor wheels in series. The first motor wheel steps every time and the second motor wheel steps only if the first motor wheel is a one. The Psi-Wheels step only if the second moter wheel is a one.

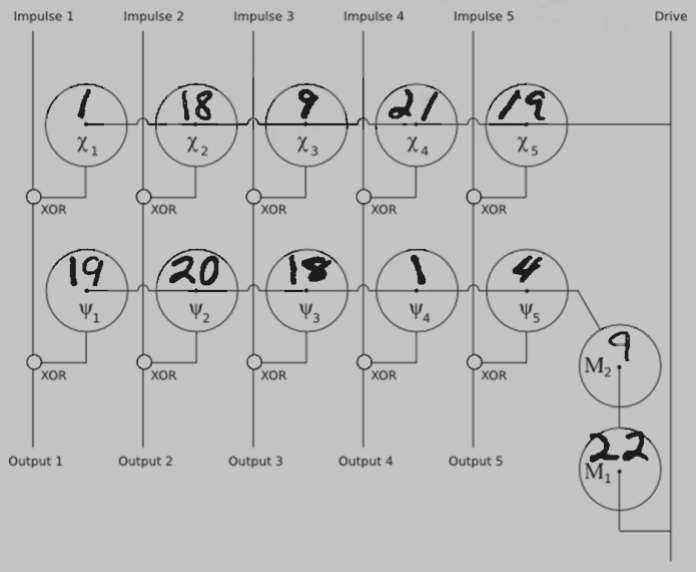
For the following analysis we denote the generated Psi-Wheel sequence with Psi'. This sequence is different from the wheel sequence since the wheels don't step every time.

The five Chi-Wheels and Psi-Wheels both produce 5-bit teleprinter letters which are XORed onto the incoming teleprinter message as shown in figure 1. Thus applying the theory of additive ciphers gives immediately as the keystream

$$K = \chi \oplus \psi'$$

and the encrypted message is

$$Z = K \oplus C.$$



3.2. Document « PHOTOCOPIE » - fourni par le CTF

UNCLASSIFIED

Detail of test sequence No.89170/ [redacted]

Report No.	Test designation	Test Result	Approbation code No.
No.1943/1	SZ40/KH/PT/5/8/9	U53ABA5LJKGDR48EPLSLNGDAU5AT5J9FCPSXVGUN3TJB VZN5H5NYSIIDX3JTQTPKJGCUOJZRF5JBIHRW8BKUWTAI UFIVKC6J4ZWMCZN/R/WKPCICHFBXACSNA9ZGJ9NZGGEH TFBZCSNFWTMY4HIYGHHSTBEX/EB39/SR/ZALY5JZIXW KH/5PLQ4LKHLNM3VSQE98FUVDWCKJN839RRPNVCJT5M	GREEN/O.K. ACCEPTED
No.[redacted]	SZ42a/KH/PT/5/8/9	8ZNXVXKTXH4WCNNGGBX/KIC4TVKSIIWTS8XWWWLEVY/ HQELVCPKJ4CAVBFUC5WHUNTL5K9/IMD/NJMJXGU93NK GMHWJEUG3S8C8ZWKJGS4559AQKPG5JCGBN/LWD34/SSG N95CE/ND8Q8IJQVZAD58CDL/94YT/GT3ZMHNTXS4OADO 4H8HBOSCCLHCXSWM58UUYAXCVLQUCIZA5HYBUOA3BM9 SJZRYWH/RJFIY49L3SBUKLU3QFCZMPJ9RLCMKIRCX5S ZCB3GIBGEHETLCAN9H4G53VZ9SGLHMKW9C8YCKRGCJFB 4PVD8CVYV8WR9JCFJRKWURR49F8C5D9XNYN343TWWJY GBHQPTMYS4JCD8LC3YG8LHV9TIKU9CALVU3VFD5H8DVN MXWKY/ITPV48GEVP9DKI5KP8RTKXJNKBK9GWPG8GA4 TETWFYFT3CA/TH4JJ9S9DGSCKSEIQ5QKCWD8ESCEJ4BS HK/DKRQUHXARYXVLE3SNL5C//CN8HJNDJ4MCLRXLAZ8 C5KWYD5CDM444HWYQCZ4RSA4LNVK5SPP/55QIEAZCTSM ZNCRHCPKJJXNZTNASQGD3SKCVNKAAVMGN/C53CWL95N 8RSPZKVUZPRVH/GZLXGCB	GREEN/O.K. ACCEPTED FLAGGED
No.[redacted]	SZ40/ZMUG/PT/5/8/9	5WH9M5ZAPYWKCHWZNYEYKLCH9PXL4ET3E9W8P/L/PPK MPRGNYCYRX/EUXGCH4IEHJW4NLPNJKYMFZPTR5ABHAE W83RP8A9/QMUFZLSAU5VVMWMVGCTIEQTXQQFJUVPCTLL LJIUAKKYXQBILM84HIWBDR854BF5VG8MCIHJKCSLLUJ 4FTJEXA4/CXFFJACGCKBUW/KV55RH8SX544YAAPGYN8 PV5CMR/YNMLV/OGQVM94ZQ3ZI8HQWC3IH4/KXBPRN/L FV/3V/UVBZQUTLA/VPICFJMACF4SU5ZDCH48JUZSC48C 4/UHWJRJVKP/44FGIPESKTBBUZ4ZTJBI33HGKSCBVXN /KM89UJGX/CQCYMBURT8T3C/PJGDDHTVDRIQ8SZX3CD D3DDB9DCRFXGEK8MBCMWUG5HXPL83PF/PWIWAXMDR K5CPI3CMLG48RTXXS/HWSDSIJRRWLVX5VNXCUDR4MT8 EVDHNVRYYZKAIGS	GREEN/O.K. ACCEPTED
No.1943/2	SZ42a/ZMUG/PT/5/8/9	AG8ZBBNY8ACRWK5R9VJY9/EEVR5LMDE/UKTHP4JEZPZ B3IYFCYWKWQMY/3NY/LCJPFP3VLU/EUIAZ45USM8K8O L/54T/4LDRXWATKB/NS/ZFI93AXFBMRHECYKAS3I/KR UCUSJMCXMJJCCENEA5ECITA9NMBCLJ9QCJBMBAPMGN8 IJCWDA3T4QCZQJNGPKGXGDHU385W5UFTRAH8QWUAPGUG 4TX8P/5GQE8TA4TVZYGX5EL33I3ZEYKJGNIWWYXU5KR UTZOCYYKEESUPPBCUDKO4YWKLSM9WGQ9YXG5A/WET/H H8QETS5GOFILCF3PIRJHAEQCXYRCUJXQBYMSS8PZH4TG BZEKZPAHFGZ534RHZ3LITNE/PKKDTTBVQYPKSUPH5/E 44/CTC5Q9AEG4FRCIWK8CMTH5Z/HDGFXYK94F5IER9P UENO8YBG9VXH98WHBJT5DLHHPNAXZFIPBUMJ3WLHPXI E/PP/AY8TSPP95MWHQHUNDJNMJJB4AJFFZS89YRBIGP 4PEQZHQQH9Y8N3RKTETMFP8KQ	[redacted]

page. 58

3.3. Script du programme decode Lorenz

```
/**
 * Emulation of the Lorenz SZ40/42a/42b cipher attachment.
 *
 * Tested against the Colossus Rebuild at Bletchley Park's TNMOC
 * using a variety of inputs and settings to confirm correctness.
 *
 * @author VirtualColossus [martin@virtualcolossus.co.uk]
 * @copyright Crown Copyright 2019
 * @license Apache-2.0
 */

import Operation from "../Operation.mjs";
import OperationError from "../errors/OperationError.mjs";

/**
 * Lorenz operation
 */
class Lorenz extends Operation {

    /**
     * Lorenz constructor
     */
    constructor() {
        super();

        this.name = "Lorenz";
        this.module = "Bletchley";
        this.description = "The Lorenz SZ40/42 cipher attachment was a WW2 German rotor cipher machine with twelve rotors which attached in-line between remote teleprinters.<br><br>It used the Vernam cipher with two groups of five rotors (named the psi(ψ) wheels and chi(χ) wheels at Bletchley Park) to create two pseudorandom streams of five bits, encoded in ITA2, which were XOR added to the plaintext. Two other rotors, dubbed the mu(μ) or motor wheels, could hold up the stepping of the psi wheels meaning they stepped intermittently.<br><br>Each rotor has a different number of cams/lugs around their circumference which could be set active or inactive changing the key stream.<br><br>Three models of the Lorenz are emulated, SZ40, SZ42a and SZ42b and three example wheel patterns (the lug settings) are included (KH, ZMUG & BREAM) with the option to set a custom set using the letter 'x' for active or '.' for an inactive lug.<br><br>The input can either be plaintext or ITA2 when sending and ITA2 when receiving.<br><br>To learn more, Virtual Lorenz, an online, browser based simulation of the Lorenz SZ40/42 is available at <a href='https://lorenz.virtualcolossus.co.uk' target='_blank'>lorenz.virtualcolossus.co.uk</a>.<br><br>A more detailed description of this operation can be found <a href='https://github.com/gchq/CyberChef/wiki/Lorenz-SZ' target='_blank'>here</a>.";
        this.infoURL = "https://wikipedia.org/wiki/Lorenz_cipher";
        this.inputType = "string";
        this.outputType = "string";
        this.args = [
            {
                name: "Model",
                type: "option",
                value: ["SZ40", "SZ42a", "SZ42b"]
            },
            {
                name: "Wheel Pattern",
                type: "argSelector",
                value: [
                    {
                        name: "KH Pattern",
                        off: [19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30]
                    },
                    {
                        name: "ZMUG Pattern",
                        off: [19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30]
                    },
                    {
                        name: "BREAM Pattern",
                        off: [19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30]
                    },
                    {
                        name: "No Pattern",
                        off: [19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30]
                    },
                    {
                        name: "Custom",
                        on: [19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30]
                    }
                ]
            },
            {
                name: "KT-Schalter",
                type: "boolean",
                value: false
            },
            {
                name: "Mode",
                type: "argSelector",
                value: [
                    {
                        name: "Send",
                        on: [4],
                        off: [5]
                    },
                    {
                        name: "Receive",
                        off: [4],
                        on: [5]
                    }
                ]
            },
            {
                name: "Input Type",
                type: "option",
                value: ["Plaintext", "ITA2"]
            },
            {
                name: "Output Type",
                type: "option",
                value: ["Plaintext", "ITA2"]
            },
            {
                name: "ITA2 Format",
                type: "option",
                value: ["5/8/9", "+/-/."]
            },
            {
                name: "Ψ1 start (1-43)",
                type: "number",
                value: 1
            },
            {
                name: "Ψ2 start (1-47)",
                type: "number",
                value: 1
            },
            {
                name: "Ψ3 start (1-51)",
                type: "number",
                value: 1
            },
            {
                name: "Ψ4 start (1-53)",
                type: "number",
                value: 1
            },
            {
                name: "Ψ5 start (1-59)",
                type: "number",
                value: 1
            },
            {
                name: "M37 start (1-37)",
                type: "number",
                value: 1
            },
            {
                name: "M61 start (1-61)",
                type: "number",
                value: 1
```

```
        },
        {
            name: "X2 start (1-31)",
            type: "number",
            value: 1
        },
        {
            name: "X3 start (1-29)",
            type: "number",
            value: 1
        },
        {
            name: "X4 start (1-26)",
            type: "number",
            value: 1
        },
        {
            name: "X5 start (1-23)",
            type: "number",
            value: 1
        },
        {
            name: "Ψ1 lugs (43)",
            type: "string",
            value: ".x...xx.x.x..xxx.x.x.xxxx.x.x.x.x.x..x.xx.x"
        },
        {
            name: "Ψ2 lugs (47)",
            type: "string",
            value: ".xx.x.xxx..x.x.x..x.xx.x.xxx.x....x.xx.x.x.x..x"
        },
        {
            name: "Ψ3 lugs (51)",
            type: "string",
            value: ".x.x.x..xxx....x.x.xx.x.x.x..xxx.x.x..x.x.xx..x.x.x"
        },
        {
            name: "Ψ4 lugs (53)",
            type: "string",
            value: ".xx...xxxxx.x.x.xx...x.xx.x.x..x.x.xx.x..x.x.x.x.x.x."
        },
        {
            name: "Ψ5 lugs (59)",
            type: "string",
            value: "xx...xx.x..x.xx.x...x.x.x.x.x.x.x.x.xx..xxxx.x.x...xx.x..x."
        },
        {
            name: "M37 lugs (37)",
            type: "string",
            value: "x.x.x.x.x.x...x.x.x...x.x.x...x.x...."
        },
        {
            name: "M61 lugs (61)",
            type: "string",
            value: ".xxxx.xxxx.xxx.xxxx.xx....xxx.xxxx.xxxx.xxxx.xxxx.xxx.xxxx..."
        },
        {
            name: "X1 lugs (41)",
            type: "string",
            value: ".x...xxx.x.xxxx.x...x.x..xxx....xx.xxxx.."
        },
        {
            name: "X2 lugs (31)",
            type: "string",
            value: "x..xxx...x.xxxx..xx..x..xx.xx.."
        },
        {
            name: "X3 lugs (29)",
            type: "string",
        },
        {
            name: "X4 lugs (26)",
            type: "string",
            value: "xx..x..xxxx..xx.xxx....x.."
        },
        {
            name: "X5 lugs (23)",
            type: "string",
            value: "xx..xx....xxxx.x..x.x.."
        }
    ];
}

/**
 * @param {string} input
 * @param {Object[]} args
 * @returns {string}
 */
run(input, args) {

    const model = args[0],
        pattern = args[1],
        kt = args[2],
        mode = args[3],
        intype = args[4],
        outtype = args[5],
        format = args[6],
        lugs1 = args[19],
        lugs2 = args[20],
        lugs3 = args[21],
        lugs4 = args[22],
        lugs5 = args[23],
        lugm37 = args[24],
        lugm61 = args[25],
        lugx1 = args[26],
        lugx2 = args[27],
        lugx3 = args[28],
        lugx4 = args[29],
        lugx5 = args[30];

    let s1 = args[7],
        s2 = args[8],
        s3 = args[9],
        s4 = args[10],
        s5 = args[11],
        m37 = args[12],
        m61 = args[13],
        x1 = args[14],
        x2 = args[15],
        x3 = args[16],
        x4 = args[17],
        x5 = args[18];

    this.reverseTable();

    if (s1<1 || s1>43) throw new OperationError("Ψ1 start must be between 1 and 43");
    if (s2<1 || s2>47) throw new OperationError("Ψ2 start must be between 1 and 47");
    if (s3<1 || s3>51) throw new OperationError("Ψ3 start must be between 1 and 51");
    if (s4<1 || s4>53) throw new OperationError("Ψ4 start must be between 1 and 53");
    if (s5<1 || s5>59) throw new OperationError("Ψ5 start must be between 1 and 59");
    if (m37<1 || m37>37) throw new OperationError("M37 start must be between 1 and 37");
    if (m61<1 || m61>61) throw new OperationError("M61 start must be between 1 and 61");
    if (x1<1 || x1>41) throw new OperationError("X1 start must be between 1 and 41");
    if (x2<1 || x2>31) throw new OperationError("X2 start must be between 1 and 31");
    if (x3<1 || x3>29) throw new OperationError("X3 start must be between 1 and 29");
    if (x4<1 || x4>26) throw new OperationError("X4 start must be between 1 and 26");
    if (x5<1 || x5>23) throw new OperationError("X5 start must be between 1 and 23");
```

```
// Initialise chosen wheel pattern
let chosenSetting = "";
if (pattern === "Custom") {
    const re = new RegExp("^[.xX]+$");
    if (lugs1.length !== 43 || !re.test(lugs1)) throw new OperationError("Ψ1 custom lugs must be 43 long and can only include . or x ");
    if (lugs2.length !== 47 || !re.test(lugs2)) throw new OperationError("Ψ2 custom lugs must be 47 long and can only include . or x");
    if (lugs3.length !== 51 || !re.test(lugs3)) throw new OperationError("Ψ3 custom lugs must be 51 long and can only include . or x");
    if (lugs4.length !== 53 || !re.test(lugs4)) throw new OperationError("Ψ4 custom lugs must be 53 long and can only include . or x");
    if (lugs5.length !== 59 || !re.test(lugs5)) throw new OperationError("Ψ5 custom lugs must be 59 long and can only include . or x");
    if (lugm37.length !== 37 || !re.test(lugm37)) throw new OperationError("M37 custom lugs must be 37 long and can only include . or x");
    if (lugm61.length !== 61 || !re.test(lugm61)) throw new OperationError("M61 custom lugs must be 61 long and can only include . or x");
    if (lugx1.length !== 41 || !re.test(lugx1)) throw new OperationError("X1 custom lugs must be 41 long and can only include . or x");
    if (lugx2.length !== 31 || !re.test(lugx2)) throw new OperationError("X2 custom lugs must be 31 long and can only include . or x");
    if (lugx3.length !== 29 || !re.test(lugx3)) throw new OperationError("X3 custom lugs must be 29 long and can only include . or x");
    if (lugx4.length !== 26 || !re.test(lugx4)) throw new OperationError("X4 custom lugs must be 26 long and can only include . or x");
    if (lugx5.length !== 23 || !re.test(lugx5)) throw new OperationError("X5 custom lugs must be 23 long and can only include . or x");
    chosenSetting = INIT_PATTERNS["No Pattern"];
    chosenSetting.S[1] = this.readLugs(lugs1);
    chosenSetting.S[2] = this.readLugs(lugs2);
    chosenSetting.S[3] = this.readLugs(lugs3);
    chosenSetting.S[4] = this.readLugs(lugs4);
    chosenSetting.S[5] = this.readLugs(lugs5);
    chosenSetting.M[1] = this.readLugs(lugm61);
    chosenSetting.M[2] = this.readLugs(lugm37);
    chosenSetting.X[1] = this.readLugs(lugx1);
    chosenSetting.X[2] = this.readLugs(lugx2);
    chosenSetting.X[3] = this.readLugs(lugx3);
    chosenSetting.X[4] = this.readLugs(lugx4);
    chosenSetting.X[5] = this.readLugs(lugx5);
} else {
    chosenSetting = INIT_PATTERNS[pattern];
}
const chiSettings = chosenSetting.X; // Pin settings for Chi links (X)
const psiSettings = chosenSetting.S; // Pin settings for Psi links (S)
const muSettings = chosenSetting.M; // Pin settings for Motor links (M)

// Convert input text to ITA2 (including figure/letter shifts)
const ita2Input = this.convertToITA2(input, intype, mode);

let thisPsi = [];
let thisChi = [];
let m61lug = muSettings[1][m61-1];
let m37lug = muSettings[2][m37-1];
const p5 = [0, 0, 0];

const self = this;
const letters = Array.prototype.map.call(ita2Input, function(character) {
    const letter = character.toUpperCase();

    // Store lugs used in limitations, need these later
    let x2bptr = x2+1;
    if (x2bptr===32) x2bptr=1;
    let s1bptr = s1+1;
    if (s1bptr===44) s1bptr=1;

    thisChi = [
        chiSettings[1][x1-1],
        chiSettings[2][x2-1],
        chiSettings[3][x3-1],
        chiSettings[4][x4-1],
        chiSettings[5][x5-1]
    ];

    thisPsi = [
        psiSettings[1][s1-1],
        psiSettings[2][s2-1],
        psiSettings[3][s3-1],
        psiSettings[4][s4-1],
        psiSettings[5][s5-1]
    ];

    if (typeof ITA2_TABLE[letter] == "undefined") {
        return "";
    }

    // The encipher calculation

    // We calculate Bitwise XOR for each of the 5 bits across our input ( K XOR Psi XOR Chi )
    const xorSum = [];
    for (let i=0;i<=4;i++) {
        xorSum[i] = ITA2_TABLE[letter][i] ^ thisPsi[i] ^ thisChi[i];
    }
    const resultStr = xorSum.join("");

    // Wheel movement

    // Chi wheels always move one back after each letter
    if (--x1 < 1) x1 = 41;
    if (--x2 < 1) x2 = 31;
    if (--x3 < 1) x3 = 29;
    if (--x4 < 1) x4 = 26;
    if (--x5 < 1) x5 = 23;

    // Motor wheel (61 pin) also moves one each letter
    if (--m61 < 1) m61 = 61;

    // If M61 is set, we also move M37
    if (m61lug === 1) {
        if (--m37 < 1) m37 = 37;
    }

    // Psi wheels only move sometimes, dependent on M37 current setting and limitations

    const basicmotor = m37lug;
    let totalmotor;
    let lim = 0;

    p5[2] = p5[1];
    p5[1] = p5[0];
    if (mode==="Send") {
        p5[0] = parseInt(ITA2_TABLE[letter][4], 10);
    } else {
        p5[0] = parseInt(xorSum[4], 10);
    }

    // Limitations here
    if (model==="SZ42a") {
        // Chi 2 one back lim - The active character of Chi 2 (2nd Chi wheel) in the previous position
        lim = parseInt(chiSettings[2][x2bptr-1], 10);
        if (kt) {
            // p5 back 2
            if (lim===p5[2]) {
                lim = 0;
            } else {
                lim=1;
            }
        }

        // If basic motor = 0 and limitation = 1, Total motor = 0 [no move], otherwise, total motor = 1 [move]
        if (basicmotor===0 && lim===1) {
            totalmotor = 0;
        } else {
            totalmotor = 1;
        }

    } else if (model==="SZ42b") {
        // Chi 2 one back + Psi 1 one back.
        const x2b1lug = parseInt(chiSettings[2][x2bptr-1], 10);
```

```
            const s1b1lug = parseInt(psiSettings[1][s1bptr-1], 10);
            lim = 1;
            if (x2b1lug===s1b1lug) lim=0;
            if (kt) {
                // p5 back 2
                if (lim===p5[2]) {
                    lim=0;
                } else {
                    lim=1;
                }
            }
            // If basic motor = 0 and limitation = 1, Total motor = 0 [no move], otherwise, total motor = 1 [move]
            if (basicmotor===0 && lim===1) {
                totalmotor = 0;
            } else {
                totalmotor = 1;
            }

        } else if (model==="SZ40") {
            // SZ40 - just move based on the M37 motor wheel
            totalmotor = basicmotor;
        } else {
            throw new OperationError("Lorenz model type not recognised");
        }

        // Move the Psi wheels when current totalmotor active
        if (totalmotor === 1) {
            if (--s1 < 1) s1 = 43;
            if (--s2 < 1) s2 = 47;
            if (--s3 < 1) s3 = 51;
            if (--s4 < 1) s4 = 53;
            if (--s5 < 1) s5 = 59;
        }

        m61lug = muSettings[1][m61-1];
        m37lug = muSettings[2][m37-1];

        let rtnstr = self.REVERSE_ITA2_TABLE[resultStr];
        if (format==="5/8/9") {
            if (rtnstr==="+") rtnstr="5"; // + or 5 used to represent figure shift
            if (rtnstr==="-") rtnstr="8"; // - or 8 used to represent letter shift
            if (rtnstr==".") rtnstr="9"; // . or 9 used to represent space
        }
        return rtnstr;
    });

    const ita2output = letters.join("");

    return this.convertFromITA2(ita2output, outtype, mode);

}

/**
 * Reverses the ITA2 Code lookup table
 */
reverseTable() {
    this.REVERSE_ITA2_TABLE = {};
    this.REVERSE_FIGSHIFT_TABLE = {};

    for (const letter in ITA2_TABLE) {
        const code = ITA2_TABLE[letter];
        this.REVERSE_ITA2_TABLE[code] = letter;
    }
    for (const letter in figShiftArr) {
        const ltr = figShiftArr[letter];
        this.REVERSE_FIGSHIFT_TABLE[ltr] = letter;
    }
}

/**
 * Read lugs settings - convert to 0|1
 */
readLugs(lugstr) {
    const arr = Array.prototype.map.call(lugstr, function(lug) {
        if (lug===".") {
            return 0;
        } else {
            return 1;
        }
    });
    return arr;
}

/**
 * Convert input plaintext to ITA2
 */
convertToITA2(input, intype, mode) {
    let result = "";
    let figShifted = false;

    for (const character of input) {
        const letter = character.toUpperCase();

        // Convert input text to ITA2 (including figure/letter shifts)
        if (intype === "ITA2" || mode === "Receive") {
            if (validITA2.indexOf(letter) === -1) {
                let errltr = letter;
                if (errltr==="\n") errltr = "Carriage Return";
                if (errltr===" ") errltr = "Space";
                throw new OperationError("Invalid ITA2 character : "+errltr);
            }
            result += letter;
        } else {
            if (validChars.indexOf(letter) === -1) throw new OperationError("Invalid Plaintext character : "+letter);

            if (!figShifted && figShiftedChars.indexOf(letter) !== -1) {
                // in letters mode and next char needs to be figure shifted
                figShifted = true;
                result += "55" + figShiftArr[letter];
            } else if (figShifted) {
                // in figures mode and next char needs to be letter shifted
                if (letter==="\n") {
                    result += "34";
                } else if (letter==="\r") {
                    result += "4";
                } else if (figShiftedChars.indexOf(letter) === -1) {
                    figShifted = false;
                    result += "88" + letter;
                } else {
                    result += figShiftArr[letter];
                }

            } else {
                if (letter==="\n") {
                    result += "34";
                } else if (letter==="\r") {
                    result += "4";
                } else {
                    result += letter;
                }
            }

        }

    }

    return result;
}
```

```
    /**
     * Convert final result ITA2 to plaintext
     */
    convertFromITA2(input, outtype, mode) {
        let result = "";
        let figShifted = false;
        for (const letter of input) {
            if (mode === "Receive") {

                // Convert output ITA2 to plaintext (including figure/letter shifts)
                if (outtype === "Plaintext") {

                    if (letter === "5" || letter === "+") {
                        figShifted = true;
                    } else if (letter === "8" || letter === "-") {
                        figShifted = false;
                    } else if (letter === "9") {
                        result += " ";
                    } else if (letter === "3") {
                        result += "\n";
                    } else if (letter === "4") {
                        result += "";
                    } else if (letter === "/") {
                        result += "/";
                    } else {

                        if (figShifted) {
                            result += this.REVERSE_FIGSHIFT_TABLE[letter];
                        } else {
                            result += letter;
                        }

                    }

                } else {
                    result += letter;
                }

            } else {
                result += letter;
            }
        }

        return result;

    }

}

const ITA2_TABLE = {
    "A": "11000",
    "B": "10011",
    "C": "01110",
    "D": "10010",
    "E": "10000",
    "F": "10110",
    "G": "01011",
    "H": "00101",
    "I": "01100",
    "J": "11010",
    "K": "11110",
    "L": "01001",
    "M": "00111",
    "N": "00110",
    "O": "00011",
    "P": "01101",
    "Q": "11101",
    "R": "01010",
    "S": "10100",
    "T": "00001",
    "U": "11100",
    "V": "01111",
    "W": "11001",
    "X": "10111",
    "Y": "10101",
    "Z": "10001",
    "3": "00010",
    "4": "01000",
    "9": "00100",
    "/": "00000",
    " ": "00100",
    ".": "00100",
    "8": "11111",
    "5": "11011",
    "-": "11111",
    "+": "11011"
};

const validChars = "ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890+-'()/:=?,. \n\r";
const validITA2 = "ABCDEFGHIJKLMNOPQRSTUVWXYZ34589+-./";
const figShiftedChars = "1234567890+-'()/:=?,.";

const figShiftArr = {
    "1": "Q",
    "2": "W",
    "3": "E",
    "4": "R",
    "5": "T",
    "6": "Y",
    "7": "U",
    "8": "I",
    "9": "O",
    "0": "P",
    " ": "9",
    "-": "A",
    "?": "B",
    ":": "C",
    "#": "D",
    "%": "F",
    "@": "G",
    "£": "H",
    "": "J",
    "(": "K",
    ")": "L",
    ".": "M",
    ",": "N",
    "'": "S",
    "=": "V",
    "/": "X",
    "+": "Z",
    "\n": "3",
    "\r": "4"
};

const INIT_PATTERNS = {
    "No Pattern": {
        "X": {
            1: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
            2: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
            3: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
            4: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
            5: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
        },
        "S": {
            1: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
            2: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
            3: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
            4: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
```

```
            5: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
        },
        "M": {
            1: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
            2: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
        }

    },
    "KH Pattern": {
        "X": {
            1: [0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0],
            2: [1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0],
            3: [0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0],
            4: [1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0],
            5: [1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0]
        },
        "S": {
            1: [0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1],
            2: [0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1],
            3: [0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1],
            4: [0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0],
            5: [1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0]
        },
        "M":  {
            1: [0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0],
            2: [1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0]
        }
    },
    "ZMUG Pattern":  {
        "X": {
            1: [0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0],
            2: [1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0],
            3: [0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0],
            4: [1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1],
            5: [0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0]
        },
        "S": {
            1: [1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0],
            2: [0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1],
            3: [0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1],
            4: [0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1],
            5: [1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0]
        },
        "M": {
            1: [1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1],
            2: [0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1]
        }
    },
    "BREAM Pattern": {
        "X": {
            1: [0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0],
            2: [0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1],
            3: [1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0],
            4: [1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0],
            5: [0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1]
        },
        "S": {
            1: [0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0],
            2: [1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0],
            3: [1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1],
            4: [0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1],
            5: [1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1]
        },
        "M": {
            1: [1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1],
            2: [0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0]
        }
    }
};

export default Lorenz;
```

Ce script est sur cette page web : https://github.com/gchq/CyberChef/blob/master/src/core/operations/Lorenz.mjs

Le code est présent dans les dossiers suivants sur la page GitHub de CyberChef : **src/core/operations/Lorenz.mjs**