UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

HÉLIO CARLOS BRAUNER FILHO

Um simulador para a máquina Lorenz SZ40

Prof. Dr. Raul Fernando Weber
Orientador

Porto Alegre, dezembro de 2014

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Brauner Filho, Hélio Carlos

Um simulador para a máquina Lorenz SZ40 / Hélio Carlos Brauner Filho. – Porto Alegre: PPGC da UFRGS, 2014.

36 f.: il.

Trabalho de conclusão (graduação) – Universidade Federal do Rio Grande do Sul. Bacharelado em Ciência da Computação, Porto Alegre, BR–RS, 2014. Orientador: Raul Fernando Weber.

I. Weber, Raul Fernando. II. Título.

“When the seagulls... follow the trawler... it’s because they think... sardines... will be thrown... into the... sea. Thank you very much”

— Eric Cantona

AGRADECIMENTOS

Aos meus pais, Aida e Hélio, por quase tudo. À minha irmã, Liziane, pela amizade e apoio durante toda a vida. Ao meu primo Vítor, grande amigo e colaborador ao longo do curso inteiro. Ao João e ao Jefferson, bons amigos e que também ajudaram bastante ao longo do curso. Ao meu orientador, Raul Weber, sempre solícito durante todo o trabalho e cujas aulas inspiraram a sua realização. A Sid Meier e Masashi Kishimoto, cujas obras me ajudaram a desestressar nas horas mais complicadas. Aos amigos e amigas, que inspiram e fazem desse mundo um lugar mais palatável. A Kukulkan. À Maria Casadevall.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

BP	Bletchley Park
CIMT	Centre for Innovation in Mathematics Teaching
ITA2	International Telegraphy Alphabet Number 2
OTAN	Organização do Tratado do Atlântico Norte
RC4	Rivest Cipher 4
SZ40	Schlüsselzusatz 40
TLS	Transport Layer Security
USPTO	United States Patent and Trademark Office
WEP	Wired Equivalent Privacy
Wren	Women's Royal Naval Service

LISTA DE FIGURAS

RESUMO

A criptografia é uma ciência estudada desde tempos muito remotos. Ao longo da história, diversos sistemas criptográficos foram usados. Na Segunda Guerra Mundial, várias máquinas que implementavam diferentes sistemas criptográficos foram criadas. Nesse contexto é que surgiu a Lorenz SZ40, criada pelos alemães, que implementava um sistema criptográfico baseado em cifra de fluxo.

Embora esse sistema fosse baseado em outro tipo de cifra, chamada de *cifra de Vernam* ou *one-time pad*, a cifra de fluxo da forma como foi feita nesta máquina foi uma inovação, e utilizada em larga escala.

Com o objetivo de melhor compreender e auxiliar no ensino sobre o funcionamento destas máquinas e das cifras de fluxo em geral em cursos de criptografia, um simulador para a máquina Lorenz SZ40 foi construído e integrado a um ambiente já existente, o Oedipus, que contém vários outros sistemas de criptografia.

Este simulador implementa o sistema de geração de chaves utilizado pela SZ40, e utiliza uma interface que tem por objetivo aproximar o usuário da idéia geral do funcionamento da máquina de uma maneira simples e similar àquela apresentada nos outros simuladores do ambiente Oedipus.

Palavras-chave: Criptografia, simulador, simulador criptográfico, cifras de fluxo, one-time pad, Lorenz SZ40, criptoanálise, criptografia na Segunda Guerra Mundial.

A simulator for the Lorenz SZ40 machine

ABSTRACT

Cryptography is a science that has been studied since ancient times. Throughout history, several cryptographic systems have been used. In World War II, lots of machines that implemented different cryptographic systems were created. It was in this context that the Lorenz SZ40 came to exist, created by the Germans and implementing a cryptographic system based on stream ciphers.

Even though that system has been based in another kind of cipher called *Vernam Cipher* or *one-time pad*, the stream cipher that has been implemented on that machine was a great innovation that has been used in large scale.

With the objective of attaining a better understanding and to aid in the teaching about the properties of those machines and stream ciphers in general in cryptography courses, a simulator for the Lorenz SZ40 machine has been built and integrated into a previously existent environment called Oedipus, which contains several other cryptography systems implemented.

This simulator implements the key generation system used by the SZ40, and implements an interface that aims to approximate the user to the broad idea of how the machine worked in a simple manner and similar to the one presented in the other simulators in the Oedipus environment.

Keywords: cryptography, simulator, cryptographic simulator, stream ciphers, one-time pad, Lorenz SZ40, criptoanalysis, criptography in World War II.

1 INTRODUÇÃO

Sistemas de criptografia são utilizados há muito tempo, e em guerras, sempre foram cruciais para que fosse feita comunicação sem que o inimigo pudesse descobrir o conteúdo de mensagens interceptadas. Durante a Segunda Guerra Mundial, os alemães criaram uma máquina de criptografia, chamada Lorenz SZ40, que implementava um sistema de geração de chaves, chamado sistema de cifra de fluxo, em que a cada caractere digitado em um teletipo era gerada uma nova chave de criptografia.

Esta máquina, além de gerar interesse pela sua história e por ter influenciado a criptoanálise, bem como a computação, também é um exemplo de inovação da época, e fornece uma forma simples de demonstrar o funcionamento de cifras de fluxo.

Com isso em mente, neste trabalho é apresentado um simulador para a máquina SZ40, construído no ambiente de criptografia Oedipus, bem como diversas informações sobre as cifras anteriores que a influenciaram, sua criptoanálise e sua influência histórica, e cifras posteriores baseadas no mesmo conceito de cifra de fluxo.

1.1 Estrutura do texto

O Capítulo 2 introduz as cifras de fluxo e as cifras que as antecederam. O Capítulo 3 descreve a máquina SZ40, conta um pouco de sua história, explica como foi feita sua criptoanálise, conta sobre as máquinas criadas para auxiliar neste processo, a influência da SZ40 e destas máquinas e cita algumas cifras de fluxo posteriores. O Capítulo 4 apresenta o ambiente Oedipus, o simulador e uma representação formal para a SZ40, o pseudo-código para o simulador e a descrição das principais classes do código verdadeiro. O Capítulo 5 fala sobre trabalhos relacionados e os compara com o trabalho feito no simulador para a SZ40. O Capítulo 6 apresenta a conclusão e discute possíveis trabalhos futuros.

2 CIFRAS DE FLUXO

Este capítulo apresenta a idéia básica da criptografia com cifras de fluxo e *one-time pad*, bem como as características e a história de duas das primeiras cifras a utilizar estes conceitos: Cifra de Miller e Cifra de Vernam.

2.1 Conceitos básicos

Uma cifra de fluxo é um sistema de chave simétrica, onde a mesma chave de criptografia é usada para criptografar e descriptografar texto, em que os dígitos utilizados para representar uma informação são combinados através de alguma operação, um a um, com dígitos pseudo-randômicos que compõem a chave de criptografia. Na maioria dos casos em que este método de criptografia é ou foi utilizado, os dígitos utilizados para representar informação são bits, sobre os quais é aplicada uma operação de adição em módulo 2, também conhecida como *ou-exclusivo* ou ainda *disjunção exclusiva* (ROBSHAW, 1995).

As cifras de fluxo são inspiradas no conceito de *one-time pad*, em que um texto em claro é combinado com uma chave da mesma maneira como é feito com cifras de fluxo. Por este método, diferentes chaves, inteiramente randômicas, eram criadas e anotadas em pedaços de papel, com cópias feitas de acordo com o número de destinatários da mensagem. A chave era então utilizada para criptografar um texto em claro e então destruída. A primeira cifra *one-time pad* reconhecida como tal foi criada por um banqueiro chamado Frank Miller, em 1882, para criptografar mensagens enviadas por telégrafo. O método consistia na utilização de um livro de códigos com palavras e frases associadas a um número, de 1 a 14000. Após encontrar o número correspondente à palavra ou frase desejada, o remetente deveria anotar um número aleatório, também entre 1 e 14000, e fazer a soma dos dois números. Caso o resultado da soma fosse maior do que 14000, uma subtração de 14000 era realizada sobre este resultado. O destinatário deveria então subtrair a sequência de números aleatórios dos números recebidos na mensagem e, caso o resultado de uma subtração fosse menor do que 0, deveria somar 14000. Estas somas e subtrações são equivalentes a utilizar uma operação de módulo 14000. As sequências anotadas de números aleatórios eram posteriormente descartadas. O método criado por Miller, no entanto, nunca foi utilizado (RIJMENANTS, 2013).

Mais tarde, em 1917, Gilbert Vernam criou um método baseado no sistema de teletipo com código Baudot. Com este método, a primeira máquina a usar uma fita com caracteres aleatórios para criptografar foi feita. No entanto, a versão original de Vernam tinha uma fraqueza, já que a fita operava em loop, e a mesma chave poderia ser usada várias vezes. Um capitão do exército americano, Joseph Mauborgne, sugeriu a Vernam utilizar fitas inteiramente randômicas e apenas uma vez. Os dois patentearam este novo método, sendo o primeiro *one-time pad* a ser usado na prática (KAHN, 1967). Em 1949, foi pro-

vado por Claude Shannon que este método, com chaves inteiramente randômicas e pelo menos do mesmo tamanho do texto em claro a ser criptografado, não pode ser quebrado (SHANNON, 1949). O modelo original de Vernam foi feito em lógica de relés, e nas explicações que ele apresentou para o United States Patent and Trademark Office (USPTO), os sinais + e - são utilizados para representar os sinais do código Baudot. Com esta codificação, Vernam apresentou como exemplo o resultado da combinação do caractere *A* com o caractere *B*, tendo como resultado *G*, e a operação de *G* com *B*, resultando em *A*. Desta operação (Figura 2.1) podemos derivar uma tabela (Figura 2.2). Ao fazer a análise da tabela, fica evidente que na verdade a operação implementada por Vernam se trata de um *ou-exclusivo* (VERNAM, 1919).

Apesar de ser um sistema que na teoria não tem falhas, dificuldades de aspecto prático evitaram que *one-time pads* fossem utilizados mais amplamente. Entre as dificuldades, a principal encontrada é a criação de uma chave inteiramente randômica, principalmente para utilização em criptografia de textos longos. Por esta razão, as cifras de fluxo com métodos para geração de chaves pseudo-randômicas passaram a ser utilizadas em seu lugar, o que permitiu uma utilização em maior escala, mas sacrificando a segurança incondicional oferecida pelas cifras *one-time pad* (ROBSHAW, 1995).

Caractere	Representação
A	+ + - - -
B	+ - - + +
RES	- + - + +
RES	- + - + +
B	+ - - + +
A	+ + - - -

Figura 2.1: Representação do exemplo de Vernam. Em azul, os operandos, em verde, os resultados

Entrada		Saída
A	***B***	***A*** $\oplus$ ***B***
-	-	-
-	+	+
+	-	+
+	+	-

Figura 2.2: Operações derivadas do exemplo de Vernam

3 CIFRAS DE FLUXO EM TEMPOS DE GUERRA: LORENZ SZ40

Durante a Segunda Guerra Mundial, mais de 20 anos depois da criação original de Vernam, o exército alemão adotou o uso de uma máquina que utilizava os conceitos de cifra de fluxo para criptografar mensagens: a Schlüsselzusatz 40 (SZ40). Neste capítulo, é apresentado um pouco da história da máquina, bem como o funcionamento e a estrutura básica deste sistema, os métodos utilizados e inovações introduzidas pelos britânicos na criptoanálise para decifrar as mensagens criptografadas pela SZ40. Também são feitas algumas considerações sobre a influência da SZ40 e do processo inglês de decifragem sobre o desenvolvimento da computação. Alguns métodos criptográficos modernos baseados em cifra de fluxo são apresentados, com foco sobre dois dos mais conhecidos, o RC4 e o WEP.

3.1 A SZ

3.1.1 História

Em 1918, um inventor alemão, chamado Arthur Scherbius, inventou uma máquina de criptografar baseada em rotores, a Enigma, que passou a ser utilizada pelo exército e pela marinha alemã durante a década de 20 (CHURCHHOUSE, 2002). No entanto, esta máquina possuía falhas conhecidas pelos alemães, e algum trabalho de criptoanálise sobre o método já havia sido feito antes do começo da guerra (KAHN, 1991). Com isso em mente, o alto comando alemão decidiu fazer um pedido para a maior fabricante de teletipos da Alemanha, a Lorenz AG, para que criasse uma máquina de criptografia. Em 1940, o primeiro teste com a SZ40 foi realizado. Tratava-se de um módulo de acoplamento às máquinas de teletipo da companhia, baseado em rotores, assim como a Enigma, mas que implementava um método de criptografia bem diferente. A SZ40 foi então utilizada até 1941, sendo substituída por modelos mais avançados em 1942 (CHURCHHOUSE, 2002).

3.1.2 Estrutura e funcionamento

A SZ40 era acoplada aos fios de saída para o transmissor de rádio dos teletipos padrão da Lorenz. Cada caractere era formado por 5 impulsos, ou bits, codificados de acordo com o *International Telegraphy Alphabet Number 2* (ITA2), também conhecido como *código Baudot* (Figura 3.3). Uma chave pseudo-randômica era gerada através dos rotores e uma operação de *ou-exclusivo* era aplicada pelo dispositivo acoplado sobre cada um dos bits (GOOD; MICHIE; TIMMS, 1945). O caractere cifrado era então enviado por um transmissor de rádio e, quando recebido por um destinatário utilizando a mesma chave, era

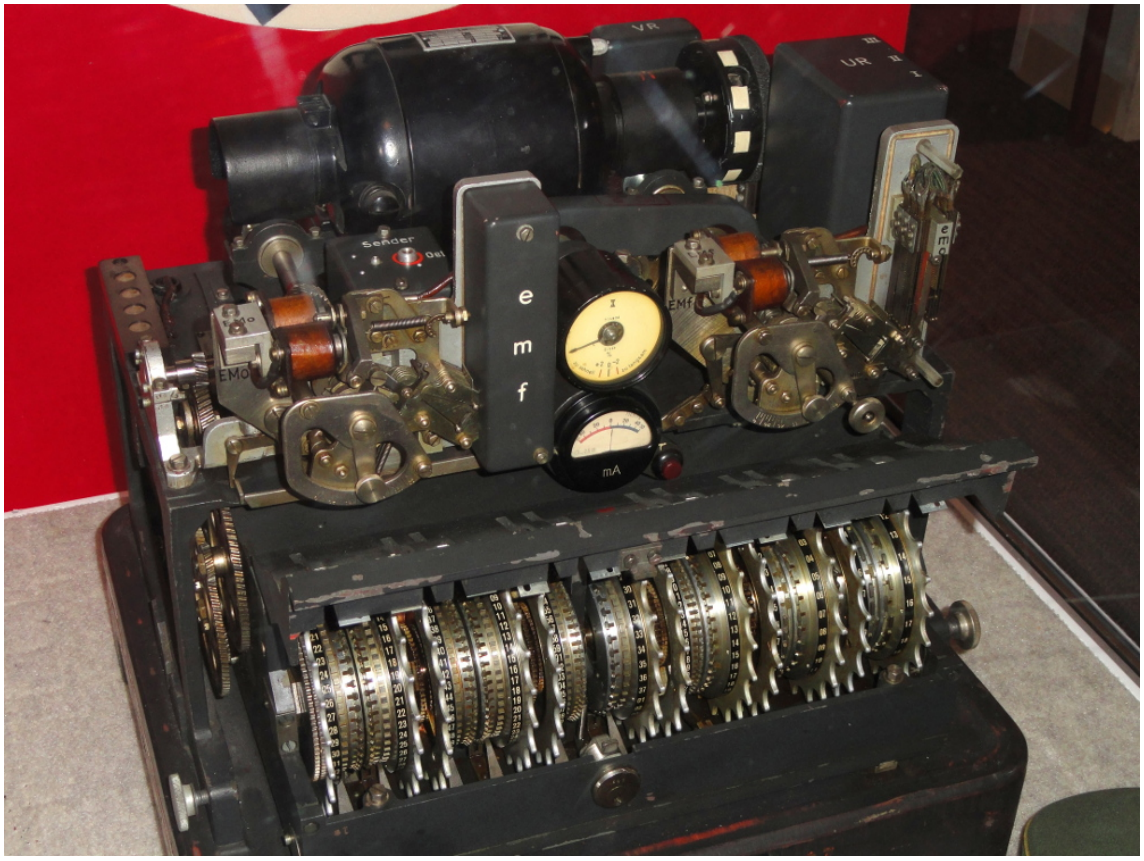

Figura 3.1: A máquina Lorenz SZ40

impresso já decifrado, tornando o processo transparente para os operadores dos teletipos.

A máquina utilizava um sistema de cifra de fluxo baseado em 12 rotores, que possuíam uma configuração especial para rotacionar e gerar as chaves para cada caractere. Os criptoanalistas britânicos em Bletchley Park criaram uma nomenclatura para especificar a estrutura dos rotores, com a divisão em 3 tipos de rotores: χ(*chi*), ψ(*psi*) e μ(*mu*) (GOOD; MICHIE; TIMMS, 1945). Cada rotor possuía um número diferente de posições (Figura 3.2), e cada posição era configurada em *ativa*, equivalente a um bit *1*, ou em *inativa*, equivalente a um bit *0*. Os 5 rotores χ giravam todos juntos a cada vez em que um caractere era digitado. Os rotores ψ também giravam juntos, mas não giravam caractere a caractere. A rotação era controlada pelos dois rotores μ. Quando $\mu 37$ se encontrava em uma posição com configuração *ativa*, os rotores ψ giravam e, quando $\mu 37$ se encontrava em posição com configuração *inativa*, permaneciam na mesma posição. O rotor $\mu 37$, por sua vez, só girava quando o rotor $\mu 61$ se encontrava em posição com configuração *ativa*, enquanto que $\mu 61$ girava sempre.

Rotor	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*
Nome em BP	ψ_1	ψ_2	ψ_3	ψ_4	ψ_5	$\mu 1$	$\mu 2$	χ_1	χ_2	χ_3	χ_4	χ_5
Posições	*43*	*47*	*51*	*53*	*59*	*37*	*61*	*41*	*31*	*29*	*26*	*23*

Figura 3.2: Configurações de rotores da máquina Lorenz SZ40. Com as 501 posições dos rotores, o número de combinações possíveis era de 2^{501}, aproximadamente $1,6 \times 10^{19}$

Padrão de impulsos x e •	Binário	Letter shift	Figure shift	Interpretação sem shifts
•••••	00000	null	null	/
••x••	00100	space	space	9
••x•x	00101	H	#	H
••••x	00001	T	5	T
•••xx	00011	O	9	O
••xxx	00111	M	.	M
••xx•	00110	N	,	N
•••x•	00010	CR	CR	3
•x•x•	01010	R	4	R
•xxx•	01110	C	:	C
•xxxx	01111	V	;	V
•x•xx	01011	G	&	G
•x••x	01001	L	)	L
•xx•x	01101	P	0	P
•xx••	01100	I	8	I
•x•••	01000	LF	LF	4
xx•••	11000	A	-	A
xxx••	11100	U	7	U
xxx•x	11101	Q	1	Q
xx••x	11001	W	2	W
xx•xx	11011	FIGS		+ or 5
xxxxx	11111		LTRS	- or 8
xxxx•	11110	K	(	K
xx•x•	11010	J	'	J
x••x•	10010	D	$	D
x•xx•	10110	F	!	F
x•xxx	10111	X	/	X
x••xx	10011	B	?	B
x•••x	10001	Z	"	Z
x•x•x	10101	Y	6	Y
x•x••	10100	S	'	S
x••••	10000	E	3	E

Figura 3.3: Esquema de código Baudot conforme utilizado pela Lorenz SZ40

Os valores de χ e ψ eram combinados através de *ou-exclusivo* de $\chi 1$ e $\psi 1$ (bits mais significativos) a $\chi 5$ e $\psi 5$ (bits menos significativos). O resultado era combinado com os bits do caractere na mesma ordem, do bit mais significativo ao menos significativo (CHURCHHOUSE, 2002).

3.1.3 Operação

As estações de operação de teletipo onde os alemães transmitiam mensagens geralmente continham duas máquinas SZ, uma ligada a um teletipo para receber e a outra ligada a um teletipo para transmitir. Antes de começar as operações de transmissão, os rotores tinham as configurações de suas posições ajustadas em *ativo* e *inativo* (representados pelos criptoanalistas em Bletchley Park por **x** e •, respectivamente), e uma posição inicial era definida para cada rotor. Até 1944, os padrões dos rotores eram alterados com uma freqüência relativamente baixa, com rotores ψ sendo mudados a cada trimestre, rotores χ sendo mudados a cada mês e rotores μ sendo mudados diariamente. A partir de 1944, os rotores ψ passaram a ser mudados mensalmente, e em 1º de Agosto de 1944, todos os rotores passaram a ter seus padrões mudados diariamente (GOOD; MICHIE; TIMMS, 1945).

As configurações iniciais para cada rotor eram sinalizadas através de uma mensagem em claro com 12 nomes próprios alemães. As iniciais indicavam as configurações que deviam ser utilizadas, registradas em um livro. Em outubro de 1942, este método foi substituído por um esquema em que os caracteres "*QEP*" eram enviados em claro, seguidos por dois dígitos que indicavam a posição no livro onde estavam as configurações que deveriam ser utilizadas. Assim que eram usadas todas as configurações no livro, que passou a ser conhecido como *QEP*, este deveria ser descartado e as estações envolvidas deveriam obter uma nova versão (COPELAND, 2006).

3.1.4 Interceptação e decifragem

Os britânicos possuíam estações para captura de sinais alemães que utilizavam código Morse, chamadas de *Y-stations*. Em 1940, uma destas estações identificou sinais diferentes do código Morse. Por causa disso, uma nova estação foi construída em 1941 em Knockholt, unicamente para detectar sinais deste novo tipo de tráfego, que passou a ser chamado pelos britânicos de *Tunny* (HINSLEY F. H.; STRIPP, 1993; GOOD; MICHIE; TIMMS, 1945).

Em 30 de Agosto de 1941, uma mensagem de cerca de 4000 caracteres foi transmitida, mas com erros na recepção, entre as cidades de Atenas e Viena. O destinatário enviou então uma requisição, escrita em claro, para que a mensagem fosse retransmitida, chamando a atenção dos operadores em Knockholt. O texto foi reenviado utilizando as mesmas configurações de rotores utilizadas no primeiro envio. Nesta segunda mensagem, alguns elementos do texto haviam mudado, em particular por causa do uso de abreviaturas. Estes dois textos forneceram aos criptoanalistas britânicos o que se chama *depth*, um caso particular em que duas ou mais mensagens são transmitidas com as mesmas configurações iniciais (CHURCHHOUSE, 2002). Os textos foram decifrados por John Tiltman, juntamente com o fluxo utilizado para criptografar os caracteres, e então se passou a tentar descobrir como funcionava a máquina que gerava esta chave.

Após 3 meses de tentativas frustradas neste sentido, o matemático Bill Tutte foi incumbido desta tarefa. Através de uma técnica que buscava repetições na chave fazendo anotações à mão, Tutte conseguiu reconhecer um padrão de repetição de 41 caracteres. Em janeiro de 1942, Tutte e os demais pesquisadores da equipe de Tiltman em Bletchley

Za	JSH5N ZYZY5 GLFRG
Zb	JSH5N ZYMFS /885I
Za ⊕ Zb	///// //FOU GFL4M

Figura 3.4: Resultados obtidos por Tiltman através dos primeiros caracteres das duas mensagens *in depth*

Park conseguiram descobrir toda a lógica de funcionamento da máquina SZ40 (SALE, 2013; TUTTE, 1998).

Após este sucesso, uma equipe de criptoanalistas, encabeçada por Ralph Tester e composta em sua maioria por membros da equipe de Alan Turing designada para investigar a máquina Enigma, foi encarregada da maior parte dos trabalhos de investigação posteriores. Esta equipe, que passou a ser chamada de *Testery*, também contou com a ajuda de máquinas criadas por um setor liderado por Max Newman e chamado de *Newmanry* (ROBERTS, 2009).

3.1.5 Criptoanálise

3.1.5.1 Passos iniciais

Ao receber o texto cifrado pelos alemães para estudar, Tiltman conseguiu identificar que os alemães haviam utilizado o mesmo esquema da cifra de Vernam. Sabendo disso, Tiltman se valeu de propriedades conhecidas deste tipo de cifra e das mensagens *in depth* interceptadas para obter a chave utilizada. Uma destas propriedades pode ser explicada da seguinte forma: Sejam *Za* e *Zb* os textos cifrados, *K* a chave de criptografia e *Pa* e *Pb* os textos em claro, e $\oplus$ representando a soma em módulo 2, temos que:

$$Za \oplus Zb = Pa \oplus Pb.$$

Com o conteúdo dos dois textos em claro conhecidos, a chave pode ser obtida através de uma das seguintes operações:

$$Za \oplus Pa = K \text{ ou } Zb \oplus Pb = K.$$

Com o resultado da operação de soma em módulo 2 entre *Za* e *Zb* (Figura3.4), Tiltman começou a utilizar elementos prováveis do texto em claro para tentar encaixar neste resultado. Ao descobrir palavras de *Pa*, tentava aplicá-las em *Pb*, e vice-versa (COPELAND; COPELAND; ROBINSON, 2010). Com isso, Tiltman obteve cerca de 4000 mil caracteres da chave utilizada nestes dois textos, sobre os quais ele e sua equipe trabalharam para descobrir o processo de criação da chave. Ao se juntar à equipe em 41, Tutte começou a utilizar o método de Kasiski, que consiste em utilizar repetições no texto cifrado e a distância entre estas repetições para tentar encontrar o intervalo de repetição da chave através da escrita do texto cifrado dividido em *n* colunas ou linhas, sendo *n* o suposto tamanho da chave (KASISKI, 1863). As mensagens de sinalização da configuração dos rotores deram a Tutte a idéia de que o período de repetição de um dos impulsos deveria ser 23, pois os alemães só haviam utilizado 23 letras para indicar a configuração de seu rotor correspondente. Para todos os outros rotores, 25 letras foram utilizadas (TUTTE, 1998).

Tutte então tentou utilizar um período inicial de 575, resultado da multiplicação de 23 por 25, e escreveu o primeiro impulso de cada caractere cifrado em linhas, mas não encontrou as repetições esperadas ao comparar verticalmente cada um dos impulsos escritos nas linhas. No entanto, percebeu que estas repetições aconteciam na diagonal, e decidiu realizar o mesmo processo com um intervalo de 574, desta vez obtendo resultados melhores. Utilizando os fatores primos de 574, Tutte chegou à conclusão de que o intervalo para um dos rotores responsáveis pelo primeiro impulso era 41. Com isso em mente, pôde descobrir que o outro rotor tinha intervalo 43, descobrindo assim o intervalo de geração para os dois rotores que compunham o primeiro dos 5 impulsos da chave, e chamando-os, respectivamente, de $\chi 1$ e $\psi 1$. Tutte e o resto da divisão de pesquisa realizaram então o mesmo processo para os outros rotores, descobrindo todos os intervalos de repetição e deduzindo, a partir de anomalias nos padrões, o propósito dos dois rotores μ (TUTTE, 1998).

3.1.5.2 Turingery

Em julho de 1942, Turing visitou o setor de pesquisas e teve contato com o trabalho relativo à quebra das cifras geradas pela SZ (COPELAND, 2006). No mesmo mês, criou um método para descobrir as configurações de ativo e inativo de cada uma das posições dos rotores da máquina (GOOD; MICHIE; TIMMS, 1945). Para tanto, Turing utilizou um conceito chamado em Bletchley Park de *diferenciação*. Uma diferença entre dois caracteres adjacentes é dada pela operação de *ou-exclusivo* entre os dois, ou, considerando *K* como sendo um caractere da chave criptográfica usada, $\overline{K}$ um caractere subsequente e ΔK representando a diferença entre os dois:

$$\Delta K = K \oplus \overline{K}$$

Daí tem-se que:

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

E então:

$$\Delta K = \Delta\chi \oplus \Delta\psi$$

E, representando o texto em claro por *P*, o texto cifrado por *Z* e relações de diferença, similares a ΔK, ΔP e ΔZ, é válida a seguinte propriedade:

$$\Delta Z = \Delta P \oplus \Delta\chi \oplus \Delta\psi$$

E ainda esta:

$$\Delta P = \Delta Z \oplus \Delta\chi \oplus \Delta\psi$$

Como os rotores ψ não se moviam sempre, quando dois caracteres eram repetidos em sucessão, $\Delta\psi$, por vezes, resultava em 5 bits em *0*, que é o código para o caractere conhecido pelos criptoanalistas britânicos por '/', ou *nulo*. Nestes casos, sabia-se então que $\Delta K = \Delta\chi$ (TUTTE, 1998). Os textos costumavam ter uma boa quantidade de caracteres repetidos um após o outro, tanto pelas características do idioma alemão como pelo hábito dos operadores alemães de enviar duas vezes seguidas caracteres de alternância entre letras e símbolos/números do teletipo, para o caso de haver algum erro na transmissão (COPELAND, 2006).

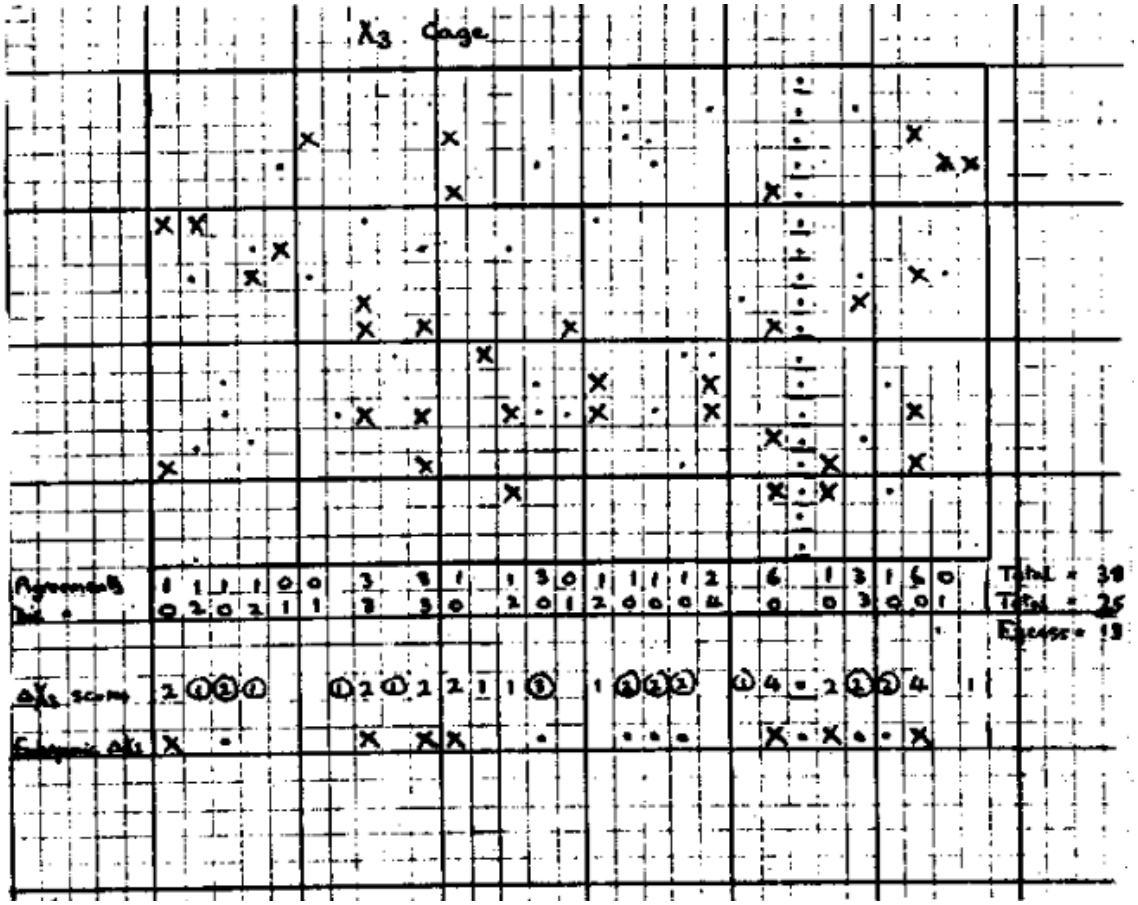

Figura 3.5: Cage para $\chi 3$, conforme reproduzido em (GOOD; MICHIE; TIMMS, 1945)

Turing então usou estas propriedades e o fato de que, em média, $\Delta\psi$ resultava no caractere nulo em metade das vezes e $\Delta\chi = \Delta K$ tinha 50% de ser verdade para criar seu método, que começava partindo do pressuposto de que um caractere resultante de ΔK provinha somente de $\Delta\chi$ (TUTTE, 1998). A partir disto, o padrão de bits imaginado era representado em uma tabela contendo cinco linhas, uma para cada um dos impulsos de χ, e um número de colunas igual ao tamanho do texto cifrado. Os cinco impulsos do primeiro elemento de χ eram anotados no começo, e a partir daí a seqüência imaginada, gerada de acordo com a estrutura conhecida dos rotores, era anotada. Folhas com anotações sobre os resultados obtidos, chamadas de *cages* (Figura 3.5), eram então utilizadas para anotar resultados progressivos sobre as deduções de $\Delta\chi$, e quando havia muitas divergências para uma determinada dedução, concluía-se que $\Delta\psi$ não era nulo para aquele caso e a dedução era descartada (COPELAND, 2006).

3.1.5.3 Abordagem 1+2 de Tutte

Em novembro de 1942, Tutte introduziu um método estatístico para descobrir a posição inicial dos rotores. O método se baseava na descoberta da posição inicial dos rotores χ através de testes exaustivos com todas as posições iniciais possíveis e então na busca por evidências de não-uniformidade da distribuição de caracteres que seria similar àquela do texto em claro (GOOD; MICHIE; TIMMS, 1945). As configurações de *ativo* e *inativo* deveriam ter sido obtidas corretamente pelos outros métodos para poder gerar a seqüência de caracteres de χ corretamente. Como o número de caracteres de chave gerados por χ ultrapassava a conta dos 22 milhões, esses testes eram feitos par a par, iniciando por $\chi 1$ e

$\chi 2$, que podiam resultar em 1271 combinações diferentes. O método então se desenvolve com base no fato de que:

$$Zi = \chi i \oplus \psi i \oplus Pi$$

onde *i* é o índice de cada um dos cinco impulsos de um caractere. A partir da fórmula anterior, temos que:

$$Pi = Zi \oplus \chi i \oplus \psi i$$

E daí, para os primeiros dois impulsos:

$$(P1 \oplus P2) = (Z1 \oplus Z2) \oplus (\chi 1 \oplus \chi 2) \oplus (\psi 1 \oplus \psi 2)$$

Utilizando um chute para *P1* $\oplus$ *P2* para cada uma das posições iniciais de $\chi 1$ e $\chi 2$ seriam geradas diferentes seqüências de *1* e *0* (ou **x** e •, respectivamente). Quando a posição inicial correta era utilizada para gerar as seqüências, a quantidade de bits em *0* era maior. Para obter um resultado melhor, Tutte utilizou os caracteres diferenciados, pois assim caracteres repetidos no texto em claro gerariam *0*, e $\Delta\psi 1 \oplus \Delta\psi 2$ sempre geraria *0* quando os rotores ψ não se movessem, e 50% das vezes caso tivessem se movido, o que resultava em uma ocorrência de *0* em 70% das vezes no total.

A função utilizada por Tutte podia ser expressada então como:

$$(\Delta P1 \oplus \Delta P2) = (\Delta Z1 \oplus \Delta Z2) \oplus (\Delta\chi 1 \oplus \Delta\chi 2) \oplus (\Delta\psi 1 \oplus \Delta\psi 2)$$

Tutte utilizou esta função em um texto cifrado que já havia sido decifrado e percebeu que 55% das vezes o resultado era um 0, de onde concluiu que resultados com maiores níveis de ocorrências de 0 para a etapa $(\Delta Z1 \oplus \Delta Z2) \oplus (\Delta\chi 1 \oplus \Delta\chi 2)$ deveriam indicar o posicionamento inicial correto de χ (COPELAND, 2006).

Após a aplicação deste método, era possível remover o elemento χ do texto cifrado e então tentar descobrir os valores de ψ por outros métodos mais simples.

3.2 Máquinas britânicas

3.2.1 Tunny

A Tunny britânica era uma máquina feita para funcionar da mesma forma que as SZ40 e SZ42, gerando texto em claro a partir de texto cifrado após a descoberta das configurações dos rotores (HINSLEY F. H.; STRIPP, 1993). A entrada e saída de texto era feita através de cartões perfurados, e era construída a partir dos sistemas de relés e outros equipamentos utilizados nas centrais telefônicas britânicas (COPELAND, 2006). Por isto, a programação não envolvia o posicionamento de rotores exatamente como nas SZ, mas sim a ligação de fios tal como se fazia nos terminais de telefone destas centrais (COPELAND, 2006).

Esta máquina foi projetada e criada por Gil Hayward, Allen Coombs, Bill Chandler e Sid Broadhurst no laboratório de Tommy Flowers, que viria a criar a máquina Colossus (COPELAND, 2006).

3.2.2 Heath Robinson

A Heath Robinson, batizada assim pelas operadoras da máquina em homenagem a um cartunista que criava máquinas mirabolantes para realizar tarefas triviais, foi introduzida em junho de 1943, e foi criada em um esforço conjunto por Max Newman, Frank Morrell (NOVEMBER 1943 : SLOW PROGRESS IN ITALY, 2006), Tommy Flowers (GOOD; MICHIE; TIMMS, 1945) e Charles Wynn-Williams. O objetivo desta máquina era automatizar a abordagem 1+2 de Tutte, e, para tanto, utilizava duas fitas em *loop*: uma com o texto cifrado e a outra com o par de χ utilizado. A máquina então realizava as combinações e fazia uma contagem dos bits em *0*. Quando a contagem ultrapassava um valor determinado a máquina retornava este valor, inicialmente através apenas de um painel luminoso que precisava ser interpretado e era fonte de erros manuais, posteriormente através de um dispositivo de impressão (SALE, 2001).

Apesar de novos modelos terem sido criados mais tarde, a máquina possuía alguns problemas causados por sua natureza mecânica, o mais sério deles sendo a dificuldade em sincronizar as duas fitas a uma velocidade de tratamento de cerca de 1000 caracteres por segundo, o que levou à sua posterior substituição pela máquina Colossus (SALE, 2001).

3.2.3 Colossus

Em dezembro de 1943, um protótipo da Colossus, o Mark 1, já se encontrava em funcionamento, e entrou oficialmente em operação em fevereiro de 1944, quando foi utilizado com sucesso para decifrar uma mensagem (COPELAND, 2006). A Colossus executava a mesma operação que a Heath Robinson, mas dispensava a necessidade de sincronizar duas fitas, graças ao fato de que a entrada da chave criptográfica não era mais feita com uma fita, mas sim através de um painel eletrônico programável (FLOWERS, 1983).

O projeto original de Tommy Flowers incluía diversos dispositivos diferentes para a máquina, como um sistema para leitura e movimento de fita perfurada, um codificador e somador para simular a operação da Lorenz, um painel de controle com contadores eletrônicos, uma impressora e uma unidade lógica para realizar operações booleanas (FLOWERS, 1983). A parte eletrônica foi projetada principalmente com o auxílio de William Chandler e Sidney Broadhurst (COPELAND, 2006). A Mark 1 era capaz de operar a uma velocidade de 5000 caracteres por segundo, limitada pela velocidade de leitura da fita. A sincronia era mantida entre a fita e os sinais eletrônicos do painel programável através de um sinal de *clock* gerado através da leitura das marcações da fita (FLOWERS, 1983).

Mais tarde, a versão Mark 2 foi construída, sendo utilizada pela primeira vez em 1º de Junho de 1944 (COPELAND, 2006). Esta versão possuía duas inovações tecnológicas que lhe permitiam operar 5 vezes mais rápido do que a Mark 1: um registrador de deslocamento e um *array* sistólico (FLOWERS, 1983).

3.3 Influência e cifras de fluxo posteriores

3.3.1 Influência dos trabalhos em Bletchley Park

Além da influência óbvia sobre a ciência da criptoanálise, com novos métodos e técnicas, que acabaram por resultar em desenvolvimentos futuros não só na decifragem, mas também na criação de novas cifras muito mais difíceis de serem quebradas (KAHN, 2002), é de particular importância a influência do Colossus sobre trabalhos futuros. Apesar do fato de todas as máquinas Colossus terem sido desativadas e desmontadas, com

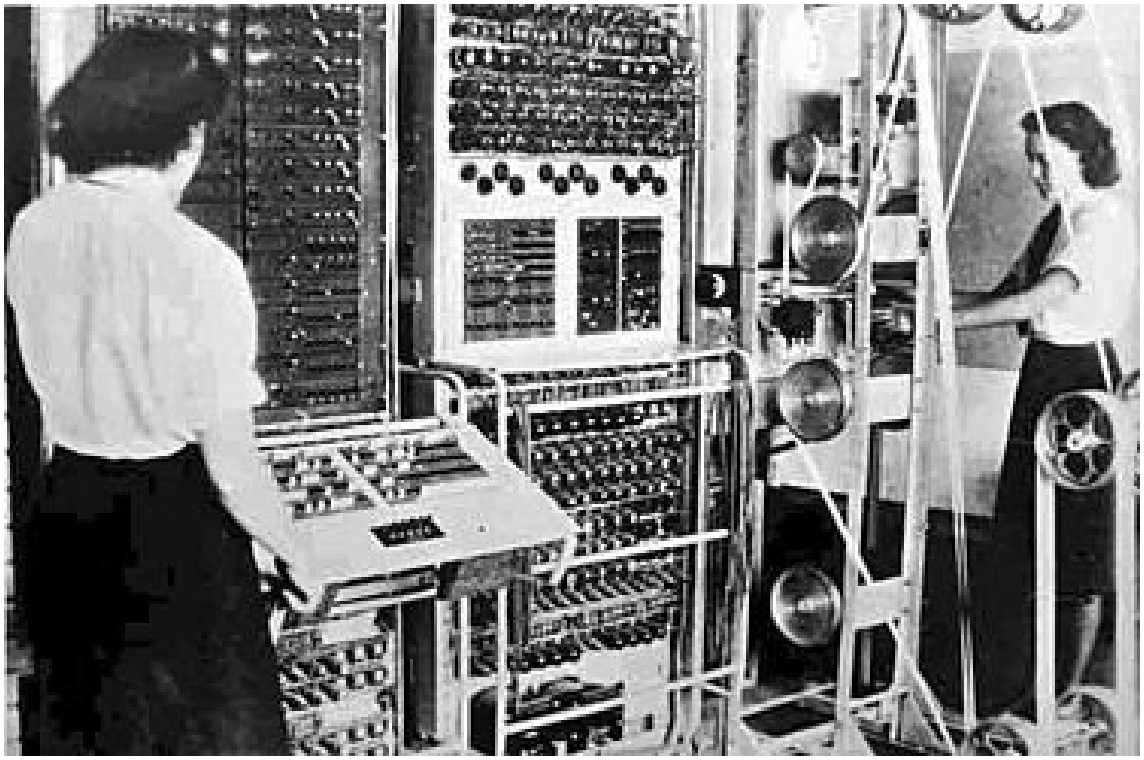

Figura 3.6: Wrens operando uma máquina Colossus Mark 1 em 1943

exceção de duas que foram mantidas em segredo e desmontadas apenas no final da década de 50 (COPELAND, 2006), e a documentação destruída para manter segredo, o conhecimento técnico e a experiência ficaram com os envolvidos, que depois trabalharam em outros projetos. Max Newman e Alan Turing viriam a trabalhar no Manchester Mark 1, um dos primeiros computadores de programa armazenado (LAVINGTON, 1978).

Allen Coombs, que havia ficado como responsável por construir as máquinas Colossus posteriores ao protótipo, trabalhou alguns anos mais tarde no desenvolvimento de sistemas de leitura ótica para código postal, criando um dos primeiros sistemas para este fim (RANDELL, 1995).

Flowers utilizou sua experiência com dispositivos eletrônicos para criar novos sistemas de centrais telefônicas (ANDREWS, 2012) e um gerador de números randômicos, o ERNIE, utilizado para fazer sorteios de loteria (OXBERRY, 2004).

Uma das discussões que emerge sobre a Colossus é a sua definição como um computador Turing-completo ou não. Sendo um computador criado com um propósito específico e com operações limitadas, a resposta inicial é não. No entanto, um conjunto de 10 máquinas Colossus poderia ser capaz de simular o funcionamento de uma máquina de Turing, e esse conjunto, se considerado como uma máquina só, seria Turing-completo (WELLS, 2009).

3.3.2 Cifras de fluxo posteriores

3.3.2.1 Siemens & Halske T52

Em 1942, os britânicos conseguiram interceptar um tipo de tráfego diferente daquele gerado pelas máquinas Lorenz. Este tráfego era gerado pelas máquinas T52, e, posteriormente, pelas máquinas T43, criadas pela Siemens & Halske. Estas máquinas também funcionavam através de rotores e da operação de *ou-exclusivo* sobre bits individuais que

compunham os caracteres, mas além disso, também possuíam um passo de permutação dos bits, que tornava a cifra mais complexa (WEIERUD, 2005-2006).

Os primeiros modelos da máquina foram quebrados tanto pelos britânicos como pelos suecos. No entanto, o T43 nunca foi quebrado. Vários países tentaram utilizar a T52 modificada ou a T43, com níveis de sucesso variáveis, sendo o principal deles a Noruega (WEIERUD, 2005-2006).

3.3.2.2 KW-26

Desenvolvida na década de 50, a KW-26 foi utilizada primeiramente pelo governo dos Estados Unidos, e posteriormente pelo governo de países membros da OTAN. Esta máquina também utilizava a mesma idéia da SZ40, aplicando *ou-exclusivo* sobre os bits do código Baudot para cada caractere. No entanto, a geração de chave não era feita através de rotores, mas sim através de registradores de deslocamento.

Como as ligações destas máquinas geravam sinais o tempo todo, mesmo quando não havia uma mensagem a ser transmitida, a análise do tráfego de mensagens era impossibilitada. A KW-26 foi modificada em 1968, após a captura de um navio americano que levava algumas dessas máquinas por parte dos norte-coreanos. Foi descontinuada durante a década de 80 (KLEIN, 2006).

3.3.2.3 Cifras modernas

Muitas cifras de fluxo foram criadas a partir da década de 80, mas várias delas com fragilidades, tais como A5/1 (POST, 2013), FISH (ANDERSON, 1995), PANAMA (DAEMEN; VAN ASSCHE, 2007) e PHELIX (WU; PRENEEL, 2007), bem como duas das mais conhecidas, RC4 (RIVEST, 2001) e WEP (IEEE, 1997).

A cifra RC4, criada em 1987 por Ron Rivest, utiliza um algoritmo para fazer permutações de bits e depois de bytes da chave, a qual pode ter tamanho variável, e não utiliza registradores de deslocamento (RIVEST, 1992). Várias formas de atacar o RC4 foram exploradas, sendo que a base destes ataques se dá através do fato de que o protocolo RC4 reutiliza chaves com certa freqüência (PAUL; PRENEEL, 2004), e que alguns dos valores gerados para a chave possuem tendências inerentes ao algoritmo de permutação (PAUL; RATHI; MAITRA, 2008) . Apesar disso, RC4 ainda foi largamente utilizado através do protocolo TLS (Transport Layer Security) até 2013, quando ataques considerados plausíveis foram apresentados (ALFARDAN et al., 2013).

O algoritmo WEP (Wired Equivalent Privacy) é baseado no RC4, e possui as mesmas falhas, sendo o maior problema o reuso de chaves por causa de um vetor de inicialização randômico para estas chaves de apenas 24 bits (FLUHRER; MANTIN; SHAMIR, 2001).

Alguns outros algoritmos, que ainda não tiveram nenhuma fraqueza significativa descoberta, incluem o MUGI (WATANABE et al., 2002) e o MULTI-S01 (FURUYA et al., 2002), sendo avaliados pelo CRYPTREC, uma iniciativa do governo japonês para estudar soluções de criptografia (CRYPTREC, 2013). Similarmente, o SOSEMANUK, o Trivium e outros, estudados pela eSTREAM, pelo menos até 2012 (ESTREAM, 2012), também entram nesta lista. Todos estes métodos foram projetados para serem implementados em software, mas estudos sobre implementação em hardware já foram feitos, por exemplo, para o MUGI (KITSOS; SKODRAS, 2006). Os métodos de geração de chave são bem variados entre estas cifras, mas têm em comum a utilização de métodos polinomiais.

4 SIMULANDO A SZ40

Um simulador simples para a SZ40, integrado ao ambiente de simulação Oedipus, foi desenvolvido com o intuito de facilitar o ensino sobre o seu funcionamento e sobre cifras de fluxo.

Neste capítulo, será apresentado o ambiente Oedipus, o simulador, suas funções, uma representação formal da máquina SZ40 e uma descrição em pseudo-código de seu funcionamento básico.

4.1 O ambiente Oedipus

O ambiente Oedipus foi desenvolvido em dois trabalhos anteriores, (MELLO, 2006) e (LOCATELLI, 2008). Ele inclui opções para utilizar os métodos de criptografia Vigenere, transposição, Playfair, ADFGVX, Beale, Enigma, substituição, deslocamento e, a partir da versão descrita neste trabalho, Lorenz, e podem ser acessados através da opção *Métodos* da barra de ferramentas principal do ambiente.

O ambiente foi desenvolvido anteriormente em Java, com o uso extensivo de duas bibliotecas, a *Java Reflections*, para carregar bytecodes em tempo de execução, e a *Java Swing*, para a utilização da interface gráfica.

O ambiente possui opções e definições padrão para cada um dos métodos. Todos eles possuem a opção de salvar e carregar texto claro ou cifrado, trocar entre cifragem e decifragem, excetuando-se aqui a Enigma e a Lorenz, que utilizam o mesmo processo para cifrar e decifrar, e a opção para limpar dados, removendo tanto o texto claro como o texto cifrado. Estas opções se encontram no menu *Arquivos* da barra de ferramentas.

4.2 Método: Lorenz

O simulador que foi integrado ao ambiente Oedipus foi construído a partir do simulador da máquina Enigma, no mesmo IDE utilizado para este, o Eclipse.

A representação adotada para cada rotor é uma tabela com número de linhas igual ao número de posições de cada rotor da SZ40 e duas colunas. A primeira coluna indica o número da posição e a segunda indica se a posição está em *ativo* ou *inativo*, usando a representação binária com *1* e *0*. Os rotores receberam a mesma nomenclatura dada pelos britânicos em Bletchley Park, ao invés da numeração de 1 a 12 utilizada pelos alemães.

O mesmo display de teclas presente no simulador da Enigma foi mantido, embora ele não existisse na máquina Lorenz, pois facilita a visualização do resultado para o usuário, e fica sobre o teclado virtual. Uma parte especial da interface exibe quais são os valores de χ, ψ e μ como um todo, o que também não existia, mas auxilia na realização de cálculos

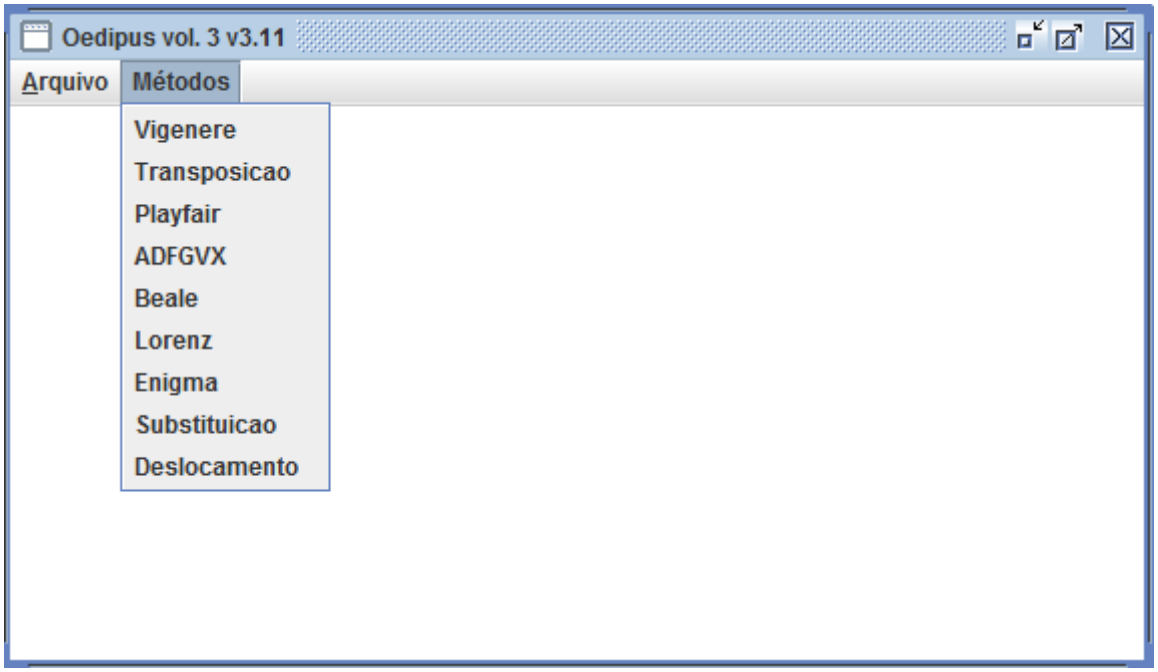

Figura 4.1: Tela principal do ambiente Oedipus

manuais para melhor compreender o funcionamento da máquina.

Os menus do topo incluem as opções para salvar texto claro e texto cifrado, limpar dados das caixas de texto claro e texto cifrado, ver a tabela de caracteres com sua representação em código Baudot, e, também, a janela para configuração das posições iniciais de cada rotor.

Os exemplos apresentados na Figura 4.2 e na Figura 4.3 podem ser acompanhados através da Figura 4.4 e da Figura 4.5.

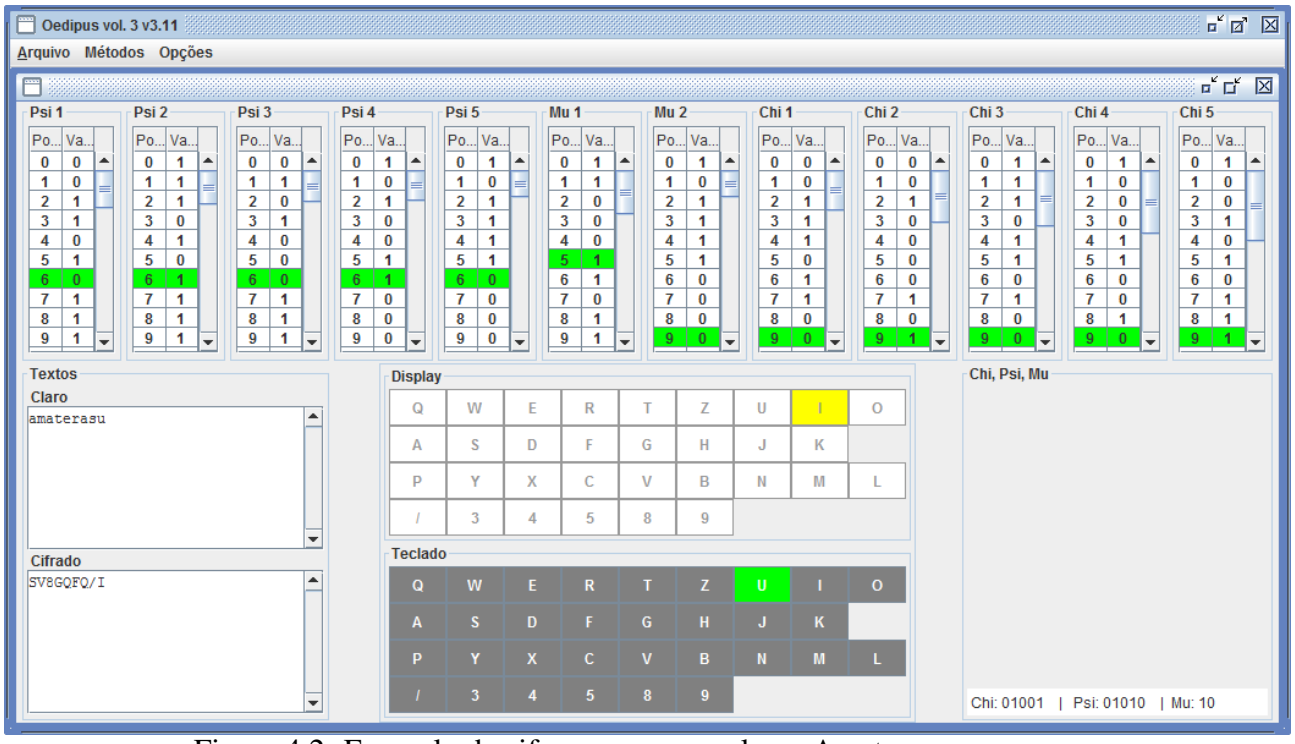

Figura 4.2: Exemplo de cifragem com a palavra Amaterasu

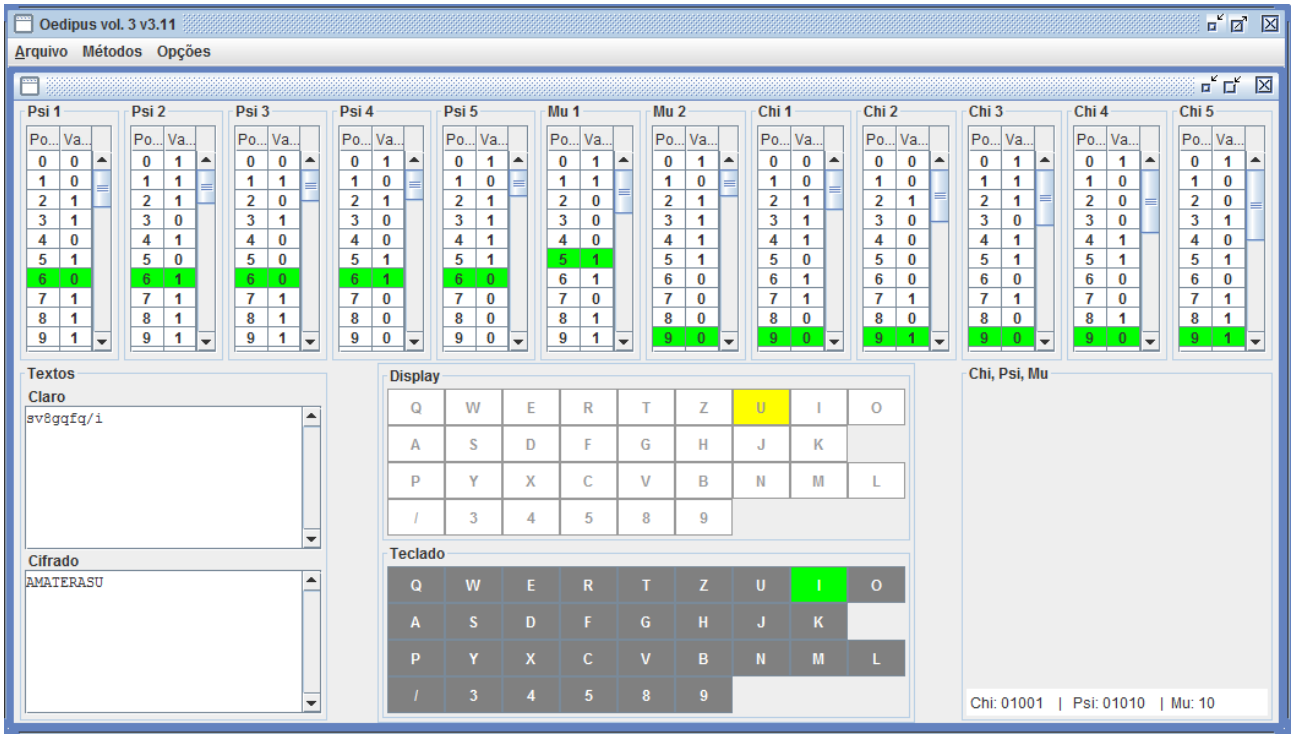

Figura 4.3: Exemplo de decifragem com a palavra Amaterasu

Caractere de entrada	*A*	*M*	*A*	*T*	*E*	*R*	*A*	*S*	*U*
Chi	00111	00100	11100	10001	10110	00111	10000	11101	00011
Psi	01011	01100	11011	11011	11011	11011	10101	01001	10011
Chave	01100	01000	00111	01010	01101	11100	00101	10100	10000
Entrada em Baudot	11000	00111	11000	00001	10000	01010	11000	10100	11100
Resultado em Baudot	10100	01111	11111	01011	11101	10110	11101	00000	01100
Caractere resultante	*S*	*V*	*8*	*G*	*Q*	*F*	*Q*	*/*	*I*

Figura 4.4: Processo de cifragem da palavra Amaterasu com a configuração utilizada nos exemplos

Caractere resultante	*S*	*V*	*8*	*G*	*Q*	*F*	*Q*	*/*	*I*
Chi	00111	00100	11100	10001	10110	00111	10000	11101	00011
Psi	01011	01100	11011	11011	11011	11011	10101	01001	10011
Chave	01100	01000	00111	01010	01101	11100	00101	10100	10000
Entrada em Baudot	10100	01111	11111	01011	11101	10110	11101	00000	01100
Resultado em Baudot	11000	00111	11000	00001	10000	01010	11000	10100	11100
Caractere de entrada	*A*	*M*	*A*	*T*	*E*	*R*	*A*	*S*	*U*

Figura 4.5: Processo de decifragem da palavra Amaterasu com a configuração utilizada nos exemplos

4.3 Representação Formal

Uma cifra de fluxo é geralmente representada por um autômato finito determinístico (CANTEAUT; FILIOL, 2001). A melhor representação neste caso é uma máquina de Mealy, com entrada e saída representados nas transições entre estados. Se imaginarmos que cada combinação de configurações de *ativo* e *inativo* para cada uma das posições de cada rotor é uma máquina de Mealy diferente, uma definição para cada uma dessas

máquinas pode ser dada como:

$$SZ40\{\eta\} = \{S, S_0, \Sigma, \Lambda, T, G\}$$

S é um conjunto de estados finitos, definidos pela forma $\chi 1\{P_1\}$, $\chi 2\{P_2\}$, $\chi 3\{P_3\}$, $\chi 4\{P_4\}$, $\chi 5\{P_5\}$, $\mu 1\{P_6\}$, $\mu 2\{P_7\}$, $\psi 1\{P_8\}$, $\psi 2\{P_9\}$, $\psi 3\{P_{10}\}$, $\psi 4\{P_{11}\}$, $\psi 5\{P_{12}\}$, onde P_x indica a posição inicial do rotor para cada rotor, com cada P_x limitado pelo tamanho do rotor correspondente.

S_0 é o estado inicial, definido pelas posições P_x iniciais para cada rotor.

Σ é o alfabeto de entrada, definido pelo conjunto de todos os caracteres representados em código Baudot.

Λ é o alfabeto de saída, também dado pelo conjunto de todos os caracteres representados em código Baudot.

T é a função de transição definida por T: $S \times \Sigma \rightarrow S$.

G é a função de saída definida por G: $S \times \Sigma \rightarrow \Lambda$.

η indica a configuração de *ativo* e *inativo* dos rotores.

Embora esta representação possa ser utilizada, ela não é prática, pois o número de configurações possíveis e o número de estados para cada uma das máquinas resultantes ultrapassa 2^{500}.

4.4 Código

O funcionamento principal da máquina SZ40, como foi implementado no código em Java, pode ser apresentado através de pseudo-código (Figura 4.6).

No código em Java do simulador, existem quatro classes principais. A primeira, chamada *Lorenz*, implementa a lógica apresentada no pseudo-código da Figura 4.6 e controla os elementos da interface. A segunda, *LorenzRotor*, implementa a estrutura de cada um dos rotores e define as configurações de *ativo* e *inativo* padrão para cada um deles. As últimas duas, *CellRendererLorenzType* e *TableLorenzType* estendem as classes padrão do Java *DefaultTableCellRenderer* e *JTable*, respectivamente, e são responsáveis pela estrutura e controle das tabelas e suas células para representar os rotores na interface.

```
1: function XOR(arg1[5], arg2[5])
2:
3:     resultado[5]
4:     for i from 1 to 5 do
5:         if arg1[i] == arg2[i] then
6:             resultado[i] = 0
7:         else
8:             resultado[i] = 1
9:     return resultado
10:
11: function SZ40()
12:
13:     posicaoChi[5] = inicialChi()
14:     tamanhoChi[5] = {TCHI1, TCHI2, TCHI3, TCHI4, TCHI5}
15:
16:     posicaoMu[2] = inicialMu()
17:     tamanhoMu[2] = {TMU1, TMU2}
18:
19:     posicaoPsi[5] = inicialPsi()
20:     tamanhoPsi[5] = {TPSI1, TPSI2, TPSI3, TPSI4, TPSI5}
21:
22:     while true do
23:
24:         if teclaPressionada() then
25:
26:             // Calcula caractere cifrado e o imprime
27:
28:             claro = pegaUltimaTecla()
29:             cifrado = xor(pegaPsi(), pegaChi())
30:             output(xor(claro, cifrado))
31:
32:             // Passa para o próximo estado dos rotores
33:
34:             for i from 1 to 5 do
35:                 posicaoChi[i] = mod(posicaoChi[i] + 1, tamanhoChi[i])
36:
37:             posicaoMu[1] = mod(posicaoMu[1] + 1, tamanhoMu[1])
38:
39:             if pegaValorMu1(posicaoMu[1]) == 1 then
40:                 posicaoMu[2] = mod(posicaoMu[2] + 1, tamanhoMu[2])
41:             if pegaValorMu2(posicaoMu[2]) == 1 then
42:                 for i from 1 to 5 do
43:                     posicaoPsi[i] = mod(posicaoPsi[i] + 1, tamanhoPsi[i])
```

Figura 4.6: Pseudo-código representando as operações principais do simulador da SZ40

5 TRABALHOS RELACIONADOS

A idéia de usar um simulador para máquinas de criptografia é bastante comum, e mesmo simuladores para a SZ40 já foram feitos. Alguns dos trabalhos anteriores na mesma área serão discutidos neste capítulo.

5.1 Simulador Online

Um dos trabalhos utilizados como fonte de inspiração para o simulador apresentado foi o simulador online feito por Adam Grove (GROVE, 2013). As funcionalidades implementadas são as mesmas, mas existem algumas diferenças essenciais quando é feita uma comparação entre os dois simuladores.

Além das diferenças no aspecto gráfico, a maneira de realizar a entrada e a saída de dados é bem diferente. Enquanto no simulador integrado ao Oedipus esta entrada é feita através de um teclado virtual, com a saída feita em duas caixas de texto, no simulador online esta entrada é feita diretamente através do teclado físico, com a saída em uma caixa de texto apenas. Esta opção, embora mais simples, resulta em alguns problemas, pois permite a entrada de caracteres não reconhecidos, que são tratados, mas podem ocasionar algumas falhas na cifragem. Outros problemas ocorrem ao se tentar apagar o texto cifrado e o texto em claro, problemas estes que são evitados no ambiente Oedipus graças à opção de limpar dados.

Uma diferença positiva deste simulador online em relação ao simulador integrado é o display, que mostra não apenas a tecla digitada e a saída, como também a chave intermediária gerada por χ e ψ.

5.2 *Toolkit* para cifra simplificada

Um *toolkit* para a cifra de Lorenz simplificada foi feito pelo *Centre for Innovation in Mathematics Teaching* (CIMT) da *University of Plymouth* (REYNOLDS, 2005) para auxiliar no ensino sobre o funcionamento da SZ40 original. Neste *toolkit*, além de diversas ferramentas de auxílio para compreender os métodos de Tutte e Turing para decifrar as mensagens da Lorenz, é possível encontrar um simulador de dois rotores que auxilia na compreensão da idéia de uma cifra de fluxo e do funcionamento da SZ40 em geral. Não apresenta o sistema inteiro da SZ40 e nem possui um equivalente para a função dos rotores μ, mas é uma ferramenta útil no ensino de cifras de fluxo e criptoanálise.

5.3 Simuladores para outras máquinas

Existem diversos simuladores para outras máquinas de criptografia, incluindo aqui as que não necessariamente utilizam cifras de fluxo. Alguns dos mais interessantes foram feitos por Dirk Rijmenants, e incluem simuladores para a conhecida máquina Enigma, para a M-209 utilizada pelos americanos na Segunda Guerra Mundial e também para as máquinas BC-52 e KL-7 usadas durante a Guerra Fria (RIJMENANTS, 2014). Estes simuladores são particularmente interessantes pois a representação gráfica das máquinas é muito realista, sendo possível visualizar o interior delas e analisar melhor seu funcionamento.

6 CONCLUSÃO

No trabalho foram explicados alguns aspectos sobre cifras de fluxo e suas origens, bem como o funcionamento da máquina Lorenz SZ40, a sua criptoanálise e as máquinas criadas pelos britânicos para auxiliar no trabalho de decifrar as mensagens geradas por ela, e alguns exemplos de cifras de fluxo posteriores.

A fim de melhor elucidar o entendimento da SZ40 e das cifras de fluxo em geral no contexto de um curso sobre criptografia, foi construído um simulador, integrado ao ambiente Oedipus, que implementa as funcionalidades desta máquina. Para realizar esta integração, optou-se por manter a mesma identidade visual e algumas das mesmas funcionalidades já presentes nos outros simuladores contidos neste ambiente.

O código feito foi baseado no código do simulador da Enigma incluído no ambiente Oedipus, para simplificar o trabalho e também para manter a coesão, de maneira a facilitar o entendimento para alguém que já tenha tido ou venha a ter contato com o código da Enigma.

Por fim, uma representação formal baseada na máquina de Mealy foi apresentada, de onde saiu justamente a conclusão de que esta forma, embora possível, não é prática para representar a SZ40 por causa do grande número de estados que precisam ser representados. Após esta representação, uma versão em pseudo-código da lógica da SZ40 e uma breve descrição das classes no código em Java que implementa o simulador foram feitas.

Como sugestão para trabalhos futuros, seria interessante a criação de um simulador para a máquina Colossus, que tornaria mais fácil o entendimento dos métodos utilizados para a criptoanálise da SZ40. Outra sugestão é colocar tutoriais explicando alguns aspectos básicos de cada método criptográfico e dados históricos, para que o simulador possa ser usado mais facilmente por alguém que tenha interesse no assunto mas não realiza nenhum curso na área.

REFERÊNCIAS

ALFARDAN, N. J. et al. On the security of RC4 in TLS and WPA. In: USENIX SECURITY SYMPOSIUM. **Anais...** [S.l.: s.n.], 2013. p.83.

ANDERSON, R. On Fibonacci keystream generators. In: FAST SOFTWARE ENCRYPTION. **Anais...** [S.l.: s.n.], 1995. p.346–352.

ANDREWS, J. The Sacking of Tommy Flowers. **Resurrection: The Bulletin of the Computer Conservation Society**, [S.l.], v.1, n.59, p.29–34, 2012.

CANTEAUT, A.; FILIOL, E. Ciphertext only reconstruction of stream ciphers based on combination generators. In: FAST SOFTWARE ENCRYPTION. **Anais...** [S.l.: s.n.], 2001. p.165–180.

CHURCHHOUSE, R. **Codes and Ciphers**: julius caesar, the enigma and the internet. Cambridge, East Anglia, UK: Cambridge University Press, 2002.

COPELAND, B. J.; COPELAND, B. J.; ROBINSON, H. Colossus: breaking the german"tunny"code at bletchley park. an illustrated history. **The Rutherford Journal: The New Zealand Journal for the History and Philosophy of Science and Technology**, [S.l.], v.3, 2010.

COPELAND, J. **The German Tunny Machine**. Oxford, South East England, UK: Oxford University Press, 2006.

CRYPTREC. **About CRYPTREC**. Acesso em novembro de 2014, `http://www.cryptrec.go.jp/english/about.html`.

DAEMEN, J.; VAN ASSCHE, G. Producing collisions for PANAMA, instantaneously. In: FAST SOFTWARE ENCRYPTION. **Anais...** [S.l.: s.n.], 2007. p.1–18.

ESTREAM. **eSTREAM**: the ecrypt stream cipher project. Acesso em novembro de 2014, `http://www.ecrypt.eu.org/stream/`.

FLOWERS, T. H. The design of Colossus. **Annals of the History of Computing**, [S.l.], v.5, n.3, p.239–252, 1983.

FLUHRER, S.; MANTIN, I.; SHAMIR, A. Weaknesses in the key scheduling algorithm of RC4. In: SELECTED AREAS IN CRYPTOGRAPHY. **Anais...** [S.l.: s.n.], 2001. p.1–24.

FURUYA, S. et al. On non-negligible bias of the first output byte of RC4 towards the first three bytes of the secret key. **IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences**, [S.l.], v.E85-A, n.1, p.58–65, 2002.

GOOD, J.; MICHIE, D.; TIMMS, G. **General Report on Tunny**: with emphasis on statistical methods. City of London, London, UK: UK Public Record Office, 1945. (HW 25/4 e HW 25/5).

GROVE, A. **Lorenz cipher simulator**. Acesso em novembro de 2014, `http://adamsgames.com/lorenz/index.htm`.

HINSLEY F. H.; STRIPP, A. **Codebreakers**: the inside story of bletchley park. Oxford, South East England, UK: Oxford University Press, 1993.

IEEE. IEEE Standard for Information Technology- Telecommunications and Information Exchange Between Systems-Local and Metropolitan Area Networks-Specific Requirements-Part 11: wireless lan medium access control (mac) and physical layer (phy) specifications. **IEEE Std 802.11-1997**, [S.l.], p.i–445, 1997.

KAHN, D. **The Codebreakers**. New York City, NY, USA: Macmillan, 1967.

KAHN, D. **Seizing the Enigma**: the race to break the german u-boat codes, 1939-1943. Boston, Mass., USA: Houghton Mifflin Co, 1991.

KAHN, D. **Remarks of David Kahn Commemorating the 50th Anniversary of the National Security Agency**. Acesso em novembro de 2014, `http://fas.org/irp/eprint/kahn.html`.

KASISKI, F. W. **Die Geheimschriften und die Dechiffrir-Kunst**. Berlin, Brandenburg, DE: E. S. Mittler und Sohn, 1863.

KITSOS, P.; SKODRAS, A. N. On the Hardware Implementation of the MUGI Pseudorandom Number Generator. In: FIFTH INTERNATIONAL SYMPOSIUM ON COMMUNICATIONS SYSTEMS, NETWORKS AND DIGITAL SIGNAL PROCESSING,(CSNDSP'2006), PATRAS, GREECE. **Anais...** [S.l.: s.n.], 2006. p.19–21.

KLEIN, M. **Securing record communications**: the tsec/kw-26. Acesso em novembro de 2014, `https://www.nsa.gov/about/_files/cryptologic_heritage/publications/misc/tsec_kw26.pdf`.

LAVINGTON, S. H. The Manchester Mark I and atlas: a historical perspective. **Communications of the ACM**, [S.l.], v.21, n.1, p.4–12, 1978.

LOCATELLI, A. P. **Um ambiente didático para o aprendizado dos algoritmos clássicos de criptografia**. 2008. Dissertação (Mestrado em Ciência da Computação) — Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

MELLO, P. E. **Um ambiente didático para o aprendizado dos algoritmos clássicos de criptografia**. 2006. Dissertação (Mestrado em Ciência da Computação) — Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

NOVEMBER 1943 : slow progress in italy. Acesso em novembro de 2014, `https://web.archive.org/web/20121002235131/http://www.bletchleypark.org.uk/content/archive/nov1943.rhtm`.

OXBERRY, D. **Inside out**: premium bonds. Acesso em novembro de 2014, http://www.bbc.co.uk/insideout/northwest/series6/premium_bonds.shtml.

PAUL, G.; RATHI, S.; MAITRA, S. On non-negligible bias of the first output byte of RC4 towards the first three bytes of the secret key. **Designs, Codes and Cryptography**, [S.l.], v.49, n.1-3, p.123–134, 2008.

PAUL, S.; PRENEEL, B. A New Weakness in the RC4 Keystream Generator and an Approach to Improve the Security of the Cipher. In: FAST SOFTWARE ENCRYPTION. **Anais...** [S.l.: s.n.], 2004. p.245–259.

POST, W. **How the NSA pinpoints a mobile device**. Acesso em novembro de 2014, http://apps.washingtonpost.com/g/page/world/how-the-nsa-pinpoints-a-mobile-device/645/#document/p1/a135574.

RANDELL, B. **Obituary**: allen coombs. Acesso em novembro de 2014, http://www.independent.co.uk/news/people/obituary-allen-coombs-1611270.html.

REYNOLDS, A. J. **Simplified Lorenz Cipher Toolkit**. Acesso em novembro de 2014, http://www.cimt.plymouth.ac.uk/resources/codes/lorenz/.

RIJMENANTS, D. **The Complete Guide To Secure Communications With The One Time Pad Cipher**. 2013.

RIJMENANTS, D. **Dirk Rijmenants' Cipher Machines and Cryptology**. Acesso em novembro de 2014, http://users.telenet.be/d.rijmenants/.

RIVEST, R. RSA Security response to weaknesses in key scheduling algorithm of RC4. **Technical note, RSA Data Security, Inc**, [S.l.], 2001.

RIVEST, R. L. The RC4 encryption algorithm. **RSA Data Security Inc**, [S.l.], 1992.

ROBERTS, J. **My Top-Secret Codebreaking During World War II**: the last british survivor of bletchley park's testery. Acesso em novembro de 2014, http://www.ucl.ac.uk/news/news-articles/0903/09031601.

ROBSHAW, M. J. B. **Stream Ciphers Technical Report, version 2.0**. Bedford, Mass., USA: RSA Laboratories, 1995. (TR-701).

SALE, T. **The Rebuild of Heath Robinson**. Acesso em novembro de 2014, http://www.codesandciphers.org.uk/virtualbp/hrob/hrrbld05.htm.

SALE, T. **The Lorenz Cipher and how Bletchley Park broke it**. Acesso em novembro de 2014, http://www.codesandciphers.org.uk/lorenz/fish.htm.

SHANNON, C. E. Communication Theory of Secrecy Systems. **Bell System Technical Journal**, [S.l.], v.28, n.4, p.656–715, 1949.

TUTTE, W. T. **Fish and I**. Acesso em novembro de 2014, http://www.usna.edu/Users/math/wdj/papers/cryptoday/tutte_fish.pdf.

VERNAM, G. **Secret signaling system**. 1919. n.US1310719 A.

WATANABE, D. et al. A new keystream generator MUGI. In: FAST SOFTWARE ENCRYPTION. **Anais...** [S.l.: s.n.], 2002. p.179–194.

WEIERUD, F. Bletchley Park's Sturgeon, the Fish that Laid No Eggs. **The Rutherford Journal: The New Zealand Journal for the History and Philosophy of Science and Technology**, [S.l.], v.1, 2005-2006.

WELLS, B. Advances in I/O, speedup, and universality on Colossus, an unconventional computer. In: **Unconventional Computation**. [S.l.]: Springer, 2009. p.247–261.

WU, H.; PRENEEL, B. Differential-linear attacks against the stream cipher Phelix. In: FAST SOFTWARE ENCRYPTION. **Anais...** [S.l.: s.n.], 2007. p.87–100.