Teoria da Informação, Segredo Perfeito e Entropia

Bruno da Rocha Braga
Ravel / COPPE / UFRJ
brunorb@ravel.ufrj.br
http://www.ravel.ufrj.br/

20 de Fevereiro, 2003

Resumo

Em um artigo de 1949, Shannon propôs os fundamentos da teoria da informação, que teve implicação em diversos campos como codificação e compressão de dados, telecomunicações e criptografia. Neste artigo, introduzimos estes conceitos e realizamos uma análise das cifras de codificação simétricas mostrando porque elas não são absolutamente seguras.

1. Introdução

Em 1949, Claude Shannon publicou um paper entitulado "Communication Theory of Secrecy Systems" que introduziu diversos conceitos valiosos no estudo da criptografia. Neste artigo, apresentamos uma rápida introdução a sua teoria para que, ao final, mostremos sua implicação prática no uso de sistemas de criptografia. Sugerimos que o leitor possua conhecimentos de teoria de probabilidade e consulte as referências indicadas na última seção deste texto no caso de dúvidas ou procura por provas matemáticas para os teoremas apresentados.

Considere os seguintes conceitos usados em textos de segurança da informação:

Cripto-sistema — sistema constituído por algoritmo matemático para criptografia de dados.

Computational Security — considerando uma medida do esforço computacional do método mais eficiente para quebrar um cripto-sistema, este será considerado computacionalmente seguro se esta medida for muito grande. Tal medida é, em geral, a complexidade de algoritmo.

Unconditional Security — um cripto-sistema é incondicionalmente seguro se, mesmo considerando os recursos computacionais como infinitos, o sistema não pode ser quebrado. Para tanto, utiliza-se a teoria de probabilidades para provar a segurança do cripto-sistema.

Um problema de cripto-análise pode ser classificado como:

Known Plaintext attack — ataque contra um cripto-sistema para o qual possuímos um texto simples (*plaintext*) e o correspondente texto cifrado (*ciphertext*) usando uma chave K.

Plaintext Choosen attack — ataque contra um cripto-sistema que se encontra disponível, ou seja, podemos encriptar quantos *plaintext*s desejarmos e analisarmos o seu correspondente *ciphertext*.

Ciphertext-only attack — ataque contra um cripto-sistema apenas por análise do texto cifrado. Existe uma quantidade *mínima necessária* de texto cifrado para que esta análise seja matematicamente viável — o que não quer dizer computacionalmente viável. Algoritmos de chave simétrica simples como Shift Cipher e Substitution Cipher não são seguros contra este tipo de ataque uma vez ultrapassada essa quantidade mínima.

Em seu trabalho, Shannon apresentou algoritmos simples (do tipo cifra de substituição) e provou que alguns deles são incondicionalmente seguros contra ataques por ciphertext-only. Veremos, na seção seguinte, qual a diferença entres estas classes de algoritmos.

2. O Segredo Perfeito

Nesta seção estaremos avaliando a segurança de cifras de substituição em que cada símbolo (unidade mínima de informação) *x*, da mensagem M que queremos proteger, é criptografado independentemente dos outros símbolos e por uma chave *K*, tal que *K* não é mais utilizada na encriptação de nenhum outro símbolo da mensagem. Neste caso, se nossa mensagem tem tamanho |M|, precisaremos de |M| chaves distintas. Portanto, para facilitar a demonstração, trataremos apenas de mensagens compostas por um único símbolo, ou seja, |M|=1.

Considere então os seguintes cripto-sistemas *alfa* e *beta* (baseados em *matriz de encriptação*) constituídos por (P, C, K, E, D), respectivamente:

- O conjunto dos caracteres que podem estar presente no texto (*plaintext*), P={ a, b, c };
- O conjunto dos caracteres que podem estar presentes no texto-cifrado (*ciphertext*), C={ 1, 2, 3, 4 } para alpha e C={ 1, 2, 3 } para beta;
- O conjunto de chaves passíveis de ser escolhidas K={ k_1, k_2, k_3 };
- A função de encriptação E e a função de decriptação D.

Note que as escolhas de valores para P e K são independentes entre si — você escolhe uma chave e só então seleciona um caractere — ou seja, ambos são *variáveis aleatórias* independentes entre si, cujos valores e respectivas probabilidades associadas são descritos abaixo:

Função de Encriptação *alfa*

	a	**b**	**c**
k_1	1	2	3
k_2	2	3	1

Função de Encriptação *beta*

	a	**b**	**c**
k_1	3	2	1
k_2	1	3	2

k_3	3	4	1

$P(P=a) = 1/4$; $P(P=b) = 1/2$;
$P(P=c) = 1/4$

$P(K=k_1) = 1/2$; $P(K=k_2) = 1/4$;
$P(K=k_3) = 1/4$

k_3	2	1	3

$P(P=a) = 1/4$; $P(P=b) = 2/4$;
$P(P=c) = 1/4$

$P(K=k_1) = 1/3$; $P(K=k_2) = 1/3$;
$P(K=k_3) = 1/3$

Os valores das matrizes de encriptação e suas probabilidades associadas nos permitem escrever as fórmulas de distribuição de probabilidades para cada variável aleatória dos nossos cripto-sistemas:

Conjunto imagem da função P em C, ou seja, os caracteres cifrados de uma linha de k_i.
$C(K) = \{ y=E_k(x) : x \in P \}$

A probabilidade a priori de um caractere cifrado y ser obtido
$P(C=y) = \text{Sum}[k : y \in C(k)] (P(K=k) \cdot P(P=D_k(y)))$ *(1)*

A probabilidade a posteriori de um caractere cifrado y ser obtido dado termos x
$P(C=y \mid P=x) = \text{Sum}[k : x=D_k(y)] (P(K=k))$ *(2)*

A partir destas fórmulas, Shannon simplesmente aplicou o *Teorema de Bayes* para deduzir as fórmulas análogas, implícitas nos dados disponíveis, e usadas em sua análise dos cripto-sistemas:

$P(P=x \mid C=y) = P(C=y \mid P=x) \cdot P(P=x) / P(C=y) \therefore$

$$P(P=x \mid C=y) = \frac{\text{Sum}[k : x=D_k(y)] (P(K=k)) \cdot P(P=x)}{\text{Sum}[k : y \in C(k)] (P(K=k) \cdot P(P=D_k(y)))} \quad (3)$$

Para um exemplo concreto do uso destas fórmulas, considere o cálculo da probabilidade de encontrarmos o *ciphertext 2*, usando a fórmula *(1)*:

$$\begin{aligned} P(C=2) &= [P(K=k_1) \cdot P(P=D_{k1}(2))] + [P(K=k_2) \cdot P(P=D_{k2}(2))] \\ &= [P(K=k_1) \cdot P(P=b)] + [P(K=k_2) \cdot P(P=a)] \\ &= [1/2 \cdot 1/2] + [1/4 \cdot 1/4] = 1/4 + 1/16 = 5/16 \end{aligned}$$

Agora estamos aptos a verificar a diferença entre os dois cripto-sistemas dados. Para isso, considere termos obtido o cripto-texto *2*, usando alpha. Neste caso, podemos tê-lo obtido tanto de *a* quanto de *b* (veja a matriz de alpha). A probabilidade condicional, para cada caractere do conjunto *plaintext*, dado o *ciphertext 2*, é:

$$\begin{aligned} P(P=a \mid C=2) &= [P(K=k_2) \cdot P(P=a)] / \{ [P(K=k_1) \cdot P(P=b)] + [P(K=k_2) \cdot P(P=a)] \} \\ &= [1/16] / \{ 5/16 \} \\ &= 1/5 \neq P(P=a) = 1/4 \end{aligned}$$

$P(\text{P=b} \mid \text{C=2}) = 4/5 \neq P(\text{P=b}) = 1/2$

Perceba que estes resultados nos servem de pistas na tentativa de descobrir de qual caractere *plaintext* (*a* ou *b*) o *ciphertext 2* foi obtido, pois as probabilidades a priori e a posteriori (dado *2*) de *a* e *b* são diferentes! Intuitivamente, podemos considerar que em 4/5 = 80% das vezes que obtivermos um cripto-texto *2*, este terá sido obtido de *b* — obviamente, tendo usado o cripto-sistema alfa.

Considere ainda o caso do cripto-texto *4*, também obtido usando alpha (vale observar aqui que este não pode ser obtido usando beta). Uma rápida consulta à matriz nos informa que *4* só pode ser obtido de *b* usando a chave k_3. Ou seja, uma vez encontrado o *ciphertext* 4, temos absoluta certeza de este corresponder à *b*. Matematicamente falando:

$P(\text{P=b} \mid \text{C=4}) = 1 \neq P(\text{P=b}) = 1/2$

Podemos então concluir que um algoritmo incondicionalmente seguro é aquele que não nos dá pista alguma sobre qual *plaintext* corresponde um dado *ciphertext*, ou seja, a probabilidade de ocorrência todo $x \in P$ permanece inalterada antes e depois da encriptação em um $y \in C$ dado. Assim, podemos apresentar mais uma definição:

Perfect Secrecy — característica de um cripto-sistema onde $P(\text{P=x} \mid \text{C=y}) = P(\text{P=x})$ para todo $x \in P$, $y \in C$. Em outras palavras, a probabilidade de termos x cifrado dado y (a posteriori) é a mesma probabilidade de termos x como *plaintext* (a priori).

Teorema 1: um cripto-sistema onde $|K| = |C| = |P|$ provê Perfect Secrecy se e somente toda chave é passível de ser usada com a mesma probabilidade $1/|K|$ *e* para todo $x \in P$, $y \in C$ há apenas uma única chave k tal que $E_k(x) = y$. Note que esta é uma condição suficiente e não implica que todo o cripto-sistema que provê Perfect Secrecy possua necessariamente estas características.

A função de encriptação beta provê Perfect Secrecy. Para tanto, basta fazermos os cálculos de *(3)* para cada relação (x, y) possível. Por exemplo, façamos para a relação (a, 2):

$$P(\text{P=a} \mid \text{C=2}) = \frac{[\, P(\text{K=k}_3) \cdot P(\text{P=a}) \,]}{\{\, [\, P(\text{K=k}_1) \cdot P(\text{P=b}) \,] + [\, P(\text{K=k}_3) \cdot P(\text{P=a}) \,] + [P(\text{K=k}_2) \cdot P(\text{P=c})] \,\}}$$

$$= [\, 1/3 \cdot 1/4 \,] / \{\, [\, 1/3 \cdot 1/2 \,] + [\, 1/3 \cdot 1/4 \,] + [\, 1/3 \cdot 1/4 \,] \,\}$$
$$= [\, 1/12 \,] / \{\, 1/6 + 1/12 + 1/12 \,\}$$
$$= [\ 1/12 \,] / \{\, 4/12 \,\} = 1/4 \text{ que é o mesmo que } P(\text{P=a}) = 1/4$$

No caso do nosso exemplo, nem precisávamos efetuar este cálculo, pois basta inspecionar os dados sobre os componentes P, C e K do algoritmo beta para concluir que ele cumpre todas as exigências do teorema 1. Já o algoritmo alpha não provê Perfect

Secrecy porque não cumpre as duas exigências: $|C| \neq |P| = |K|$ — C possui um elemento a mais, o *4* — *e* $E_K(c) = 1$ para mais de um K (K_2 e K_3).

3. Entropia e Redundância

Os resultados obtidos na seção anterior ainda são muito teóricos. Qual a utilidade de provarmos que possuímos algoritmos incondicionamente seguros para a encriptação de apenas um único caractere *plaintext* para cada chave disponível? Para o caso de encriptarmos uma mensagem de tamanho *n*, teríamos que fornecer por um canal seguro uma seqüência de *n* chaves distintas. E o que acontece se repetirmos a aplicação de uma chave? É o que veremos a seguir, após apresentarmos mais alguns conceitos definidos por Shannon.

Quando temos uma distribuição de probabilidades, concedemos um conjunto de valores que X pode assumir com uma probabilidade dada *P*(X); tal que a soma das probabilidades de ocorrência de cada um desses valores possíveis seja 1. Em geral, X assume valores naturais e finitos, ou seja, X = { 1, 2, ..., n }.

Imagine agora que precisássemos codificar os valores de X em binário. Poderíamos usar o código ASCII para tanto, mas se tivéssemos menos que 256 valores possíveis, haveria um *desperdício de bits*. Qual seria então a codificação mais eficiente para a representação desses valores de modo a termos uma quantidade mínima de bits empregados? O **Código de Huffman**, utilizado em softwares de compressão de arquivos (como o ZIP), provê uma codificação baseada na freqüência de ocorrência dos caracteres no texto; assim os caracteres mais freqüentes são representados com menos bits enquanto os menos freqüentes são representados com mais bits e, na média, a representação do conjunto de caracteres ASCII é feita com (*bem*) menos que 8 bits. Isso só é possível graças a uma característica própria das linguagens humanas, de usar mais algumas letras em detrimento de outras.

A **Entropia** mede justamente a quantidade de bits *média* necessária para representar um conjunto de valores X, com probabilidades associadas e distintas, da maneira mais eficiente possível, e é dada pela fórmula:

$$H(X) = - \text{Sum}[i=1..n] \left(P(X=x_i) \cdot \log_2 P(X=x_i) \right) \qquad (4)$$

Agora, vamos aplicar este conceito na teoria de cripto-sistemas desenvolvida até aqui. Um cripto-sistema possui *três alfabetos*, ou seja, os conjuntos de valores assumíveis por P, C e K; cada um com probabilidades associadas relativas aos valores pertencentes a cada conjunto. Assim, existe um código ótimo para representação dos valores nestes três conjuntos. Considerando sistemas que provêm Perfect Secrecy, temos que $|C| = |P| = |K|$, ou seja, todos os três conjuntos podem ser representados por alfabetos distintos mas com o a mesma quantidade de caracteres ou então pelo mesmo alfabeto — o que é mais prático e efetivamente ocorre.

Se considerássemos um texto escrito numa língua fictícia, com um alfabeto de tamanho *n* letras, na qual toda letra tem a mesma probabilidade 1/*n* de ocorrer, então teríamos a seguinte entropia para este alfabeto:

$H(\mathrm{P}) = - \mathrm{n} \cdot (1/\mathrm{n} \cdot \log_2(1/\mathrm{n})) = \log_2(\mathrm{n})$ que para n=26 letras é aproximadamente 4.76.

No entanto, não é isso que ocorre em nossos textos cotidianos — strings compostas de caracteres pertencentes à P. As probabilidades distintas associadas ao uso de cada letra implica em diferentes entropias para diferentes idiomas. Para o alfabeto inglês, $H(\mathrm{P}) \approx 4.19$. A distribuição de probabilidades da língua inglesa pode ser calculada com boa precisão se for realizada uma contagem de freqüência de cada letra do alfabeto em um texto grande no dado idioma.

Note que o fato da entropia da língua inglesa ser menor que a da nossa língua fictícia em que cada letra tem a mesma probabilidade de ser usada significa que podemos representar letras de textos em inglês com, em média, 4.76 - 4.19 = 0.57 bits a menos. Se compararmos a necessidade de bits para a codificação desta língua com os 8 bits do código ASCII, vemos uma economia de 3.81 bits por letra, ou seja, 47,6% no caso médio — o que justifica a grande popularidade dos softwares de compressão de dados.

n	**$H(P^n)$**
1	4.19
2	3.90
...	...
∞	1.25

Tabela 2: Entropia para n-grafos do inglês

Na prática, podemos tornar esta codificação ainda mais eficiente — ou seja, reduzirmos a entropia da língua — se ao invés de codificarmos letras, o fizermos para digramas, trigramas, ou mesmo n-gramas. Isso ocorre pois não somente as freqüências de uso de cada letra é distinta das outras como também de pares, triplas e n-tuplas de letras. Por exemplo, qual a probabilidade de, em inglês, você ter um par QQ? Zero. E qual a probabilidade de ter um par QU? 100%, pois o Q sempre é sucedido de um U neste idioma. Entre outras palavras, diz-se que uma língua é representável por um processo estocástico que produz uma seqüência de símbolos de acordo com alguns sistemas de probabilidades. Assim, chegamos a uma fórmula para o cálculo do valor teórico da **entropia de uma linguagem**:

$$H_{\mathrm{L}} = \lim[\mathrm{n} \to \infty] (H(\mathrm{P}^{\mathrm{n}})/\mathrm{n}) \qquad (5)$$

Onde P^{n} é a variável aleatória para strings de tamanho *n* na língua inglesa. Assim como P assume valores de caracteres do alfabeto com probabilidades distintas associadas a cada uma, o mesmo ocorre com strings de caracteres — por exemplo, para n=3 (P^3) a string *the* é muito mais freqüente que a string *ncé*, que só ocorre na palavra *fiancé*. Considere ainda o conceito de **redundância**:

$R_L = 1 - H_L / \log_2|P|$ *(6)*

Para o inglês, $H_L \approx 1.25$ e $R_L \approx 0.75$, ou seja, o conteúdo médio de informação em inglês poderia ser representado em um código de 1.25 bits! Um texto em inglês, suficientemente grande, poderia ser re-escrito com um código em que suas unidades de informação — não se refere às palavras — teriam, em média, 1.25 bits cada; o que implicaria dizer que o inglês, como um código, possui 75% de redundância. Vale reafirmar que isto não quer dizer que a cada quatro letras de uma palavra inglesa podemos omitir três e ainda assim manter o significado; mas que poderíamos ter outro código que expressaria a mesma sentença em termos de outras unidades de informação, cada qual, na média, necessitando de 1/4 dos bits que as atuais unidades de informação (letras) do inglês precisam.

Por fim, enunciamos um teorema que será utilizado na próxima seção e que mostra uma relação entre as entropias dos componentes de um cripto-sistema.

Teorema 2 — Key Equivocation: para um dado cripto-sistema (P, C, K, E, D), temos que $H(K|C) = H(K) + H(P) - H(C)$.

Este conceito determina o quanto pode-se inferir a respeito da chave usada a partir de uma quantidade de cripto-texto dado. Pode-se provar que para a língua fictícia em que todas as letras são equiprováveis e letras subseqüentes são geradas independentemente, temos $H(K|C) = H(K)$ e $H(P|C)$ aumenta linearmente ao longo de uma reta com coeficiente angular log n até cruzar com H(K) constante.

4. Chaves Falsas e Distância de Unicidade

Considere um cripto-sistema análogo ao da primeira seção deste texto mas que em vez de operar sobre caracteres (em P e C), este opera sobre n-gramas de *plaintext* (em P^n e C^n). Com um cripto-texto, de tamanho n, $y \in C^n$, definimos:

$k(y) = \{ k \in K : \exists x \in P^n \ \& \ P(P^n=x) > 0 \ \& \ E_k(x) = y \}$

Ou seja, k(y) é o conjunto de chaves *k* para qual y é o *ciphertext* obtido pela encriptação de um n-grama (1 símbolo = string de tamanho *n*) de *plaintext*. Como o conjunto de chaves possíveis tem tamanho |K|, a quantidade de chaves que não relacionam um valor de P ao y dado é |K| - 1. Além disso, no contexto da mensagem encriptada, este y corresponde a só um *único* $x \in P$, gerado com uma *única chave* $k \in K$, e as demais relações (e suas chaves) possíveis são na verdade "pistas falsas". Por essa razão, chamamos as demais **|k(y)| - 1** chaves de **Spurious Keys** (Chaves Falsas). A quantidade média de chaves falsas para este cripto-sistema é:

$s_n = \text{Sum}[y \in C^n] (P(C=y) \cdot (|k(y)| - 1))$ *(7)*

Acompanhe agora o raciocínio seguinte sobre o conceito de Key Equivocation de um cripto-sistema que visa chegar a uma relação entre a média de chaves falsas e o tamanho *n* de uma string de *ciphertext*.

Do teorema 2 temos que $H(K|C^n) = H(K) + H(P^n) - H(C^n)$

Usando *(5)*, fazemos a aproximação $H(P^n) = n \cdot H_L = n(1 - R_L) \cdot \log_2|P|$

Já que *n* é grande, certamente $H(C^n) \leq n \cdot H(C) = n \cdot \log_2|C|$ dada a hipótese de que nosso cripto-sistema não correlaciona a freqüência dos *ciphertext* com a dos *plaintext* (que segue o padrão da língua utilizada), pelo contrário, idealmente, cada *ciphertext* tem a mesma probabilidade 1/n de ocorrer.

Assim, como |P| = |C|, temos:
$H(K \mid C^n) \geq H(K) + [\, n(1 - R_L) \cdot \log_2|P| \,] - [\, n \cdot \log_2|C| \,] \therefore$
$H(K \mid C^n) \geq H(K) - n \cdot R_L \cdot \log_2|P|$ *(8)*

Considerando os caracteres de C^n isoladamente, podemos escrever:
$H(K \mid C^n) = \text{Sum}[y \in C^n] \, (P(C=y) \cdot H(K \mid C=y))$
$\leq \text{Sum}[y \in C^n] \, (P(C=y) \cdot \log_2|k(y)|)$ considerando cada uma das |k(y)| chaves c/ p=1/ |k(y)|
$\leq \log_2(s_n + 1)$ *(9)*

Assim, como $H(K \mid C^n)$ está entre (8) e (9), temos: $H(K) - n \cdot R_L \cdot \log_2|P| \leq \log_2(s_n + 1)$
O que nos leva ao seguinte teorema.

Teorema 3: suponha um cripto-sistema (P, C, K, E, D) onde |C| = |P| e as chaves são equiprováveis. Considere ainda P o conjunto de *plaintexts* (símbolos), com distribuição de probabilidades de uma língua L, cada um sendo uma string de tamanho *n*, onde *n* é suficientemente grande. Então, dado um *ciphertext* de um símbolo de P, o valor experado de chaves falsas, s_n, é:

$s_n \geq |K| / |P|^{nR_L} - 1$ *(10)*

É fácil ver que $s_n \to 0$ quando $n \to \infty$. Note também que a medida fica distorcida para pequenos valores de n; por exemplo, quando n=1, $s_n = |K|-1 = 25 \geq 1.26$ — aproximação considerando RL = 0.75 e |P| = |K| = 26.

Unicity Distance — é o valor de *n* para o qual o valor esperado de chaves falsas (s_n) se torna zero, ou seja, a quantidade média de *ciphertext* para que uma chave única seja computada, dado tempo de computação suficiente.

$n \approx \log_2|K| / R_L \cdot \log_2|P|$ *(11)*

5. Conclusões

5.1. Histograma de Freqüências

O conjunto P^n de símbolos (palavras, letras, digramas, trigramas ou qualquer n-grama) encontrados em textos de uma língua L possuem freqüências que tendem sempre a um mesmo valor quanto maior for a mensagem M; formando o que chamamos *histograma de freqüências* para um conjunto de símbolos desta língua. Em termos estatísticos, podemos de dizer que P^n é uma variável aleatória para uma população L, estimada a partir de uma amostra grande (um texto nesta língua).

As cifras de substituição (as monoalfabéticas e as polialfabéticas), quando utilizadas com uma única chave k encriptando cada um dos símbolos de uma mensagem grande o suficiente, fazem refletir em C (a variável aleatória para o conjunto de símbolos cifrados) a mesma distribuição de probabilidades de P para a língua L — podendo apenas mascarar esta distribuição através de algumas técnicas — viabilizando uma cripto-análise.

5.2. Segredo Perfeito

Um sistema possui segurança perfeita se não importando a quantidade de cripto-texto que dispomos, a tarefa de descobrir a mensagem original é impossível pois teremos um conjunto de soluções igualmente prováveis. Como vimos, esta é uma condição teórica, e os cripto-sistemas desenvolvidos para alcança-la são inviáveis em aplicações práticas — a chave haveria de ter o mesmo tamanho da mensagem.

Podemos usar a teoria desenvolvida na seção anterior para estimar o menor tamanho de mensagem *n* (a quantidade de *ciphertext*) para o qual não existem cifras falsas, ou seja, uma vez conhecido o algoritmo usado, é matematicamente possível descobrir a chave k usada na cifragem. Uma vez que o tamanho da mensagem seja menor que *n*, teremos um conjunto k(y) de chaves de chaves falsas que traduzem o *ciphertext* em questão para um *plaintext* possível — segundo as regras de cifragem do algortimo usado — ainda que, no caso de estar escrito em uma linguagem humana, seria provavelmente composto de palavras sem sentido, ou seja, inexistentes nesta língua. No entanto, chegar até essas chaves é a questão da cripto-análise no caso destes algoritmos.

Lembramos que matematicamente viável não significa computacionalmente viável. O algoritmo de chave-pública mais usado no mundo, o RSA, não é inquebrável já que é possível fatorar n (a chave-pública) em p*q (a chave privada). Na prática, no entanto, fatorar um número de 200 dígitos com a tecnologia atual requeriria 4 bilhões de anos.

6. Referências

HOEL, P. *Estatística Elementar*, Editora Atlas, 1976.
SHANNON, C. *Communication Theory of Secrecy Systems*, 1949.