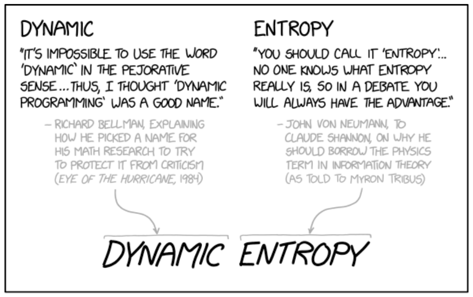
CPSC 418/MATH 318 Introduction to Cryptography
Entropy, Product Ciphers, Block Ciphers

Renate Scheidler

Department of Mathematics & Statistics
Department of Computer Science
University of Calgary

Week 3


DYNAMIC
"IT'S IMPOSSIBLE TO USE THE WORD 'DYNAMIC' IN THE PEJORATIVE SENSE... THUS, I THOUGHT 'DYNAMIC PROGRAMMING' WAS A GOOD NAME."
– RICHARD BELLMAN, EXPLAINING HOW HE PICKED A NAME FOR HIS MATH RESEARCH TO TRY TO PROTECT IT FROM CRITICISM (EYE OF THE HURRICANE, 1984)
ENTROPY
"YOU SHOULD CALL IT 'ENTROPY'... NO ONE KNOWS WHAT ENTROPY REALLY IS, SO IN A DEBATE YOU WILL ALWAYS HAVE THE ADVANTAGE."
– JOHN VON NEUMANN, TO CLAUDE SHANNON, ON WHY HE SHOULD BORROW THE PHYSICS TERM IN INFORMATION THEORY (AS TOLD TO MYRON TRIBUS)
DYNAMIC ENTROPY

SCIENCE TIP: IF YOU HAVE A COOL CONCEPT YOU NEED A NAME FOR, TRY "DYNAMIC ENTROPY."

Outline

Recap: One-Time Pad

Definition 1 (Vernam one-time pad)

$\mathcal{M} = \mathcal{C} = \mathcal{K} = \{0,1\}^n$ $(n \in \mathbb{N})$.
Encryption of $M \in \{0,1\}^n$ under key $K \in \{0,1\}^n$ is bitwise XOR, *i.e.*

$$C = M \oplus K \ .$$

Decryption of C under K is done the same way, *i.e.* $M = C \oplus K$.

Decryption is the inverse of encryption, since $K \oplus K = (0, 0, \ldots, 0)$ and $M \oplus (0, 0, \ldots, 0) = M$.

Security of the One-Time Pad

Theorem 1

The one-time pad provides perfect secrecy if each key is chosen with equal likelihood. Under this assumption, each ciphertext occurs with equal likelihood (regardless of the probability distribution on the plaintext space).

Proof sketch

The first assertion follows immediately from Shannon's Theorem (Theorem **??**). The second assertion is proved by computing $p(C)$ for all $C \in \mathcal{C}$ using the formula. □

This means that in the one-time pad, any given ciphertext can be decrypted to *any* plaintext with equal likelihood (def'n of perfect secrecy). There is no "distinguished" (e.g. meaningful) decryption. So even exhaustive search doesn't help.

Cryptanalysis of the One-Time Pad

It is imperative that each key is only used once:

- Immediately falls to a KPA: if a plaintext/ciphertext pair (M, C) is known, then the key is $K = M \oplus C$.
- Vulnerable to a COA if a key K is used twice:

  $$C_1 = M_1 \oplus K \ , C_2 = M_2 \oplus K \implies C_1 \oplus C_2 = M_1 \oplus M_2 \ .$$

  Note that this is encryption with a *coherent running key* cipher (adding two coherent texts M_1 and M_2), which is like a Vigenère cipher with an extremely long key word (shift rotation pattern) and thus vulnerable to frequency analysis (can find M_1 and M_2 from $M_1 \oplus M_2$).

For the same reason, we can't use shorter keys and "re-use" portions of them. Keys must be randomly chosen and at least as long as messages. This makes the one-time pad impractical.

Practical Issues

Main disadvantages of one-time pad:

- requires a random key which is as long as the message
- each key can be used only once.

One-time schemes are used when perfect secrecy is crucial and practicality is less of a concern, for example, Moscow-Washington hotline.

One-Time Pad: Conclusion

The major problem with the one-time pad is the cost. As a result, we generally rely on *computationally secure* ciphers.

- These ciphers would succumb to exhaustive search, because there is a unique "distinguished" (e.g. meaningful) decipherment.
- The computational difficulty of finding this solution foils the cryptanalyst.
- A *proof* of security does not exist for any proposed computationally secure system (just a reduction, subject to certain assumptions, to presumably computationally intractable problem)

Measuring Information

Recall that information theory captures the amount of information in a piece of data.

Measured by the average number of bits needed to encode all possible outcomes in an *optimal prefix-free* encoding.

- optimal – the average number of bits is as small as possible
- prefix-free – no code word is the beginning of another code word (*e.g.* can't have code words 01 and 011 for example)

Formally, the amount of information in an outcome is measured by the *entropy* of the associated random variable (function of the probability distribution over the set of possible outcomes).

Example

The four messages

UP, DOWN, LEFT, RIGHT

could be encoded in the following ways:

String	Character	Numeric	Binary
"UP"	"U"	1	00
"DOWN"	"D"	2	01
"LEFT"	"L"	3	10
"RIGHT"	"R"	4	11
(40 bits)	(8 bits)	(16 bits)	(2 bits)
(5 char string)	8-bit UTF-8	(2 byte integer)	2 bits

Coding Theory

In the example, all encodings carry the same information (which we will be able to measure), but some are more efficient (in terms of the number of bits required) than others.

Note: *Huffmann encoding* can be used to improve on the above example if the directions occur with different probabilities.

This branch of mathematics is called *coding theory* (and has nothing to do with the term "code" defined previously).

Entropy

Definition 2

Let X be a random variable with outcomes $X_1, X_2, \ldots, X_n$ and a probability distribution

$$p(X_1), p(X_2), \ldots, p(X_n) \quad \text{where} \quad \sum_{i=1}^{n} p(X_i) = 1$$

The *entropy* of X is defined by the weighted average

$$H(X) = \sum_{\substack{i=1 \\ p(X_i)\neq 0}}^{n} p(X_i) \log_2\left(\frac{1}{p(X_i)}\right) = -\sum_{\substack{i=1 \\ p(X_i)\neq 0}}^{n} p(X_i) \log_2\left(p(X_i)\right) \ .$$

Intuition

- An event occurring with probability $1/2^n$ can be optimally encoded with n bits.
- An event occurring with probability p can be optimally encoded with $\log_2(1/p) = -\log_2(p)$ bits.
- The weighted sum $H(X)$ is the expected number of bits (*i.e.* the amount of information) in an optimal encoding of X (*i.e.* one that minimizes the number of bits required).
- If $X_1, X_2, \ldots, X_n$ are outcomes (e.g. plaintexts, ciphertexts, keys) occurring with respective probabilities $p(X_1), p(X_2), \ldots, p(X_n)$, then $H(X)$ is the average amount of information required to represent an outcome.

Example 1

Suppose $n = 1$ (only one outcome). Then

$$p(X_1) = 1 \iff \frac{1}{p(X_1)} = 1 \iff \log_2 \frac{1}{p(X_1)} = 0 \iff H(X) = 0 \ .$$

No information is needed to represent X (you already know the outcome with certainty in advance).

In fact, for arbitrary n, $H(X) = 0$ if and only of $p_i = 1$ for exactly one i and $p_j = 0$ for all $j \neq i$ (formal proof later).

Example 2

Suppose $n > 1$ and $p(X_i) > 0$ for all i. Then

$$0 < p(X_i) < 1 \quad (i = 1, 2, \ldots, n)$$
$$\frac{1}{p(X_i)} > 1$$
$$\log_2 \left(\frac{1}{p(X_i)} \right) > 0,$$

hence $H(X) > 0$ if $n > 1$.

If there are at least 2 outcomes, both occurring with nonzero probability, then some amount of information is needed to represent X.

Example 3

Suppose there are two possible outcomes which are equally likely:

$$p(\text{heads}) = p(\text{tails}) = \frac{1}{2},$$
$$H(X) = \frac{1}{2}\log_2(2) + \frac{1}{2}\log_2(2) = 1 \ .$$

So one bit of information is needed to represent X.

In fact, either outcome needs 1 bit of information (heads or tails).

Example 4

Suppose we have

$$p(UP) = \frac{1}{2}, \quad p(DOWN) = \frac{1}{4}, \quad p(LEFT) = \frac{1}{8}, \quad p(RIGHT) = \frac{1}{8} \ .$$

Then
$$\begin{aligned} H(X) &= \frac{1}{2}\log_2(2) + \frac{1}{4}\log_2(4) + \frac{1}{8}\log_2(8) + \frac{1}{8}\log_2(8) \\ &= \frac{1}{2}\cdot 1 + \frac{1}{4}\cdot 2 + \frac{1}{8}\cdot 3 + \frac{1}{8}\cdot 3 = \frac{7}{4} = 1.75 \ . \end{aligned}$$

An optimal prefix-free (Huffman) encoding is

$$UP = 0, \quad DOWN = 10, \quad LEFT = 110, \quad RIGHT = 111 \ .$$

Because UP is more probable than the other messages, receiving UP is more certain (i.e. reveals less information) than receiving one of the other messages. The *average* amount of information required is 1.75 bits.

Example 5

Suppose we have n outcomes which are equally likely: $p(X_i) = 1/n$.

$$H(X) = \sum_{i=1}^{n} \frac{1}{n} \log_2 n = \log_2(n) \ .$$

So if all outcomes are equally likely, then $H(X) = \log_2(n)$.

If $n = 2^k$ (*e.g.* each outcome is encoded with k bits), then $H(X) = k$.

Applications

For a random variable on a plaintext space $\mathcal{M}$, its entropy measures the uncertainty of plaintexts.

Gives the amount of partial information that must be learned about a message in order to know its whole content when it has been

- distorted by a noisy channel (coding theory) or
- hidden in a ciphertext (cryptography)

For example, consider a ciphertext C = X\$7PK that is known to correspond to a plaintext $M \in \mathcal{M} = \{$"heads", "tails"$\}$ in a fair coin toss.

- The random variable on $\mathcal{M}$ has entropy 1, so the cryptanalyst only needs to find the distinguishing bit in the first character of M, not all of M.

Extremal Entropy

Recall that the entropy of n equally likely outcomes (*i.e.* each occurring with probability $1/n$) is $\log_2(n)$. This is indeed the maximum:

Theorem 2

$H(X)$ is maximized if and only if all outcomes are equally likely. That is, for any n, $H(X) = \log_2(n)$ is maximal if and only if $p(X_i) = 1/n$ for $1 \leq i \leq n$.

$H(X) = 0$ is minimized if and only if $p(X_i) = 1$ for or exactly one i and $p(X_j) = 0$ for all $j \neq i$.

Intuitively, $H(X)$ decreases as the distribution of messages becomes increasingly skewed.

Minimal Entropy – Proof

Proof.

If one probability is 1, say $p(X_1) = 1$, and all the others are 0, then

$$H(X) = -p(X_1)\log_2(p(X_1)) = -1 \cdot 0 = 0 \ .$$

Conversely:

$$\begin{aligned} & H(X) = 0 \\ \Rightarrow\quad & p(X_i)\log_2(p(X_i)) = 0 \text{ for each } i \text{ with } p(X_i) > 0 \\ \Rightarrow\quad & \log_2(p(X_i)) = 0 \text{ for each } i \text{ with } p(X_i) > 0 \\ \Rightarrow\quad & p(X_i) = 1 \text{ for each } i \text{ with } p(X_i) > 0 \ , \end{aligned}$$

but since all probabilities sum to one, this means there can only be one non-zero probability which is 1. □

Maximal Entropy – Proof Sketch

Proof.

- $n = 1$: this is Example 1: $p(X_1) = 1 \iff H(X) = 0$.
- Arbitrary n: see Theorem 3.6, pp. 72-73, of Paterson-Stinson.

  Applies *Jensen's inequality* for concave functions (Theorem 3.5, p. 72 of Stinson-Paterson) to $\log_2$:

$$\begin{aligned} H(X) &= \sum p(X_i) \log_2 \left(\frac{1}{p(X_i)} \right) \\ &\leq \log_2 \left(\sum p(X_i) \cdot \frac{1}{p(X_i)} \right) \quad \text{by Jensen's inequality} \\ &= \log_2 \left(\sum 1 \right) \leq \log_2(n) \end{aligned}$$

  with equality iff all $p(X_i)$ are equal (i.e. equal to $1/n$).

$\square$

Entropy of Keys

The entropy of the random variable on a key space $\mathcal{K}$ measures the amount of partial information that must be learned about a key to actually uncover it (*e.g.* the number of bits that must be guessed correctly to recover the whole key).

For a k-bit key, the best scenario is that all k bits must be guessed correctly to know the whole key (*i.e. no* amount of partial information reveals the key, only full information does).

- Entropy of the random variable on the key space should be maximal.
- By Theorem 2, this happens exactly when each key is equally likely.
- Best strategy to select keys in order to give away as little as possible is to choose them with equal likelihood (*uniformly at random*).

Cryptosystems are assessed by their key entropy, which ideally should just be the key length in bits (*i.e.* maximal).

Example: Plaintext Versus Key Entropy

$\mathcal{M} = \{0, 1\}$ (bits)

$\mathcal{C} = \mathcal{K} = \{0, 1\}^{1,000,000}$ (bit strings of length one million)

- For each key K, the encryptions of '0' and '1' under K differ by at least one bit (because encryptions are injective).
- Knowledge of the value of *one* distinguishing bit for every encryption makes it possible to deduce the correct plaintext from any ciphertext. For example, if an attacker intercepts the ciphertext

$$C = 010110 \cdots 111001$$

  and has knowledge that the 3rd bit of the encryption of '0' under the (unknown) key that was used is 1, then she knows that C is the encryption of '1' (without knowing which key was used).

So on average, one ciphertext bit reveals the entire plaintext.

Plaintext entropy is 1, even though the key entropy may be as much as 1,000,000.

Lessons Learned from Previous Example

- The security level (*i.e.* key entropy) of a cryptosystem may not tell the whole story in some applications and may in fact convey a false sense of security.
- Small message spaces are problematic (more later).
- The concept of *indistinguishability* is crucial in the context of security (more later).

Product Ciphers

Shannon also introduced the idea of product ciphers (multiple encryption):

Definition 3 (Product cipher)

The *product* of two ciphers is the result of applying one cipher followed by the other.

AKA *superencipherment* and various other names.

Note: All modern symmetric key ciphers in use are product ciphers.

Properties of Product Ciphers

If different ciphers are used in a product cipher, ciphertexts of one cipher need to have the correct format to be plaintexts for the next cipher to be applied.

- This is composition of encryption maps.

Applying a product cipher potentially increases security. E.g. n-fold encryption with one cipher and n keys potentially corresponds to a cipher that has n times longer keys.

Of course it also results in a loss of speed by a factor of n, but this might be worth it for added security.

Caveat

Be careful with this reasoning!

Note 1

The product of two substitution ciphers is a substitution cipher. The product of two transposition ciphers is a transposition cipher.

Such ciphers are *closed* under encryption, so multiple encryption under different keys provides no extra security:

E.g. double encryption $E_{K_1}(E_{K_2}(M)) = E_{K_3}(M)$ for a third key K_3.

Confusion and Diffusion

Shannon suggested applying two simple (substitution) ciphers with a fixed mixing transformation (transposition) in between to

- *diffuse* language redundancy into long-term statistics and
- *confuse* the cryptanalyst by obscuring the relationship between the ciphertext and the key.

Definition 4 (Confusion)

Make the relationship between the key and ciphertext as complex as possible (accomplished by applying substitutions or *S-boxes*).

Definition 5 (Diffusion)

Dissipate the statistical properties of the plaintext across the ciphertext (accomplished by applying transpositions or *P-boxes*).

Examples of Historic Product Ciphers

ADFGX/ADFGVX Ciphers – employed by the Germans in WW I

Hayhanen Cipher

- Reino Hayanen was KGB officer who defected to the US in 1957 and solved the *hollow nickel* espionage case for the FBI (who couldn't break the cipher!)
- This led to arrest of Russian spy Rudolph Abel and the 1961 prisoner exchange of Abel for US Air Force pilot Francis Powers whose U-2 spy plane was shot down over Russia in 1960
- Inspired Steven Spielberg's 2015 movie *Bridge of Spies* (which portrayed Hayhanen rather unfavourably)

Examples of Modern Product Ciphers

Example 6

IBM's *Lucifer* system uses permutations (transpositions) on large blocks for the mixing transformation, and substitution on small blocks for confusion.

This type of design is called a *Feistel* network, after Lucifer's designer Horst Feistel.

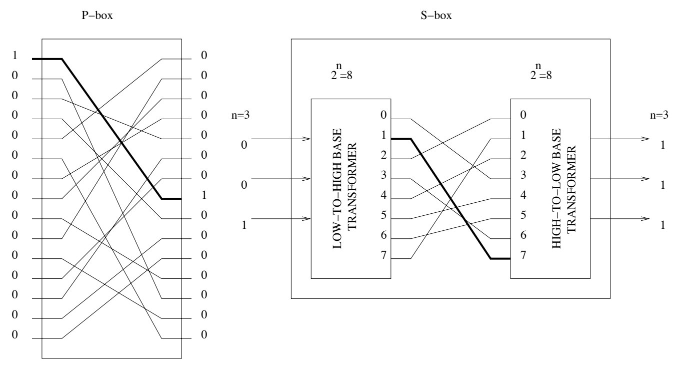
Feistel originally wanted to call the product cipher "Dataseal".

IBM instead shortened the term *demonstration cipher* to "Demon."

Later, it was changed to *Lucifer*, because it retained the "evil atmosphere" of Demon, and (more or less) contained the word *cipher*.

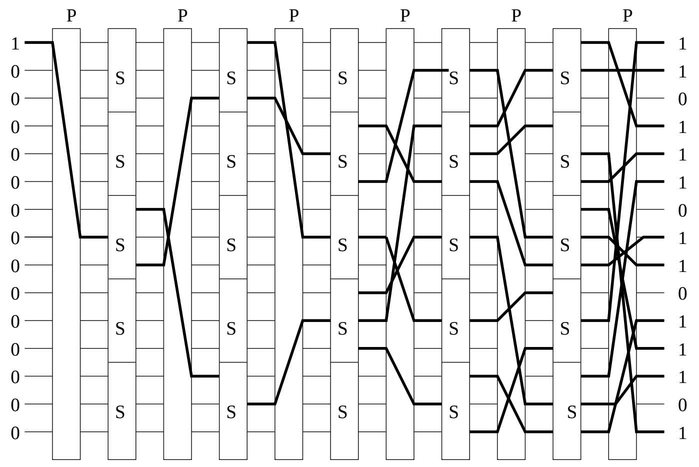
Lucifer: P-boxes and S-boxes

Since Lucifer was set up in hardware, they called the chips which did the permutation "P-boxes" and those that did the substitution "S-boxes."



The Lucifer system simply consisted of a number of P and S boxes in alternation.

Diffusion in Lucifer



The thicker lines in the graphic indicate how the first input bit '1' dissipates over the entire ciphertext.

Error Propagation

Definition 7 (Error Propagation)
The degree to which a change in the input leads to changes in the output.

Definition 8 (Avalanche Effect)
Changing one input bit leads to significant changes in the output (e.g. half the output bits flip).

Good error propagation is a desirable property of a cryptosystem (a user can easily tell if a message has been modified).

Not necessarily good for decryption though (where one might want one error in the process to still lead to a mostly correct decryption).

Block Ciphers

All modern ciphers in use are block ciphers (although not necessarily used as such — we'll talk about modes of operation of block ciphers in Week 5).

Definition 9 (Block cipher)
Encrypts plaintext blocks of some fixed length to ciphertext blocks of some fixed (possibly different) length.

Usually, a message M will be larger than the plaintext block length, and must hence be divided into a series of sequential message blocks $M_1, M_2, \ldots, M_n$ of the desired length.

- A block cipher operates on these blocks one at a time.
- May need to *pad* last block M_n to the block length

Examples of Block Ciphers

Example 10
The shift cipher is a block cipher where the blocks consists of one character (*i.e.* 8 bits on 32-bit architecture, 16 bits on 64-bit architecture).

Two main block ciphers in use today:

- Data Encryption Standard (DES)
  - Obsolete (key space too small)
  - Still used in legacy code as triple encipherment (3DES)
- Advanced Encryption Standard (AES)

NIST

NIST: National Institute of Standards and Technology

Everything about NIST's cryptographic standards, recommendations, and guidance can be found at the NIST cryptographic standards and guidelines website
`https://csrc.nist.gov/projects/cryptographic-standards-and-guidelines`.

- Extremely useful website for both practitioners and scholars of cryptography.
- There is a link on the "references" page on the course website.

NIST Publications:

- Older designation: FIPS (Federal Information Processing Register)
- Newer designation: SP (Special Publication)
  - All the crypto publications appear under SP 800

Data Encryption Standard (DES)

- Described in FIPS 46, 46-2, 46-3 (see also docs on "handouts" page)
- Developed by IBM around 1972 in secret (based on Lucifer), with input from NSA
- Block cipher that encrypts 64-bit plaintext blocks to 64-bit ciphertext blocks using 64-bit keys.
  - Note that 8 of the key bits are parity bits, resulting in 56 actual bits of the key.
- So $\mathcal{M} = \mathcal{C} = \{0,1\}^{64}$ and $\mathcal{K} = \{0,1\}^{56}$.
- Algorithm consists of 16 rounds of permutations and substitutions

$$DES_{Key}(M) = IP^{-1}(S_{16}(S_{15}(\dots(S_2(S_1(IP(M))))\dots)))$$

Multiple DES Encryption

What about multiple DES encryptions? Does this foil exhaustive attacks due to longer key sizes?

Campbell and Wiener (1992) proved that DES is *not* closed, so multiple DES encryptions/decryptions could potentially provide additional security.

- size of the group generated by all the keys (*i.e.* the number of distinct encryptions obtained by applying repeated DES encryptions) has been shown to have size at least $10^{2499} \approx 2^{8302}$. (Estimated number of atoms in the universe: 2^{240}.)

Later, we will show that on double encryption is essentially no more secure than single encryption (but twice as slow).

What about three DES encryption? 3DES (triple DES) is still used.

Triple DES

Use three successive DES operations: $C = E_{K_1}(D_{K_2}(E_{K_3}(M)))$

- See NIST Special Publication SP 800-67.

Advantages:

- Same as single key if $K_2 = K_1$ or $K_2 = K_3$.
- Exhaustive search has complexity 2^{112} via the meet-in-the-middle attack (see Week 5), but with a 168-bit key and a factor of 3 in speed.
- Can use $K_1 = K_3$ with no loss of security.
- No other known practical attacks.

The main disadvantage is that 3-DES is three times slower than single key DES while only doubling the key size.

Skipjack and the Clipper Chip

A lesson on how **not** to introduce standardized crypto!

After DES became obsolete, the United States *National Security Agency* (NSA) wanted to take control of the cipher standard selection process

- Proposed the *Skipjack Algorithm* implemented on the *Clipper Chip*
- Standardized by NIST as Escrowed Encryption Standard (EES) in Feb. 1994 (see FIPS 185) and still used by US Government.

The details of Clipper and Skipjack were initially classified and kept secret.

Due to the secrecy and wide distrust of the NSA in the US and abroad, this cipher never caught on in the public sector.

AES Competition

A lesson on how to **definitely** introduce standardized crypto!

In 1997, NIST initiated a world-wide process of candidate submission and evaluation for the *Advanced Encryption Standard* to replace DES.

The process was completely transparent and public!

Requirements:

- possible key sizes of 128, 192, and 256 bits
- plaintexts and ciphertexts of 128 bits
- should work on a wide variety of hardware (from chip cards to supercomputers)
- fast
- secure
- world-wide royalty-free availability (!)

Selection Criteria

Candidates were selected according to:

- security – resistance against all known attacks
- cost — speed and code compactness on a wide variety of platforms
- simplicity of design

Most important: *public* evaluation process

- series of three conferences: algorithms, attacks, evaluations presented and discussed
- 21 submissions from all over the world evaluated during 1998-1999
- final selection done by NIST