# Steganography

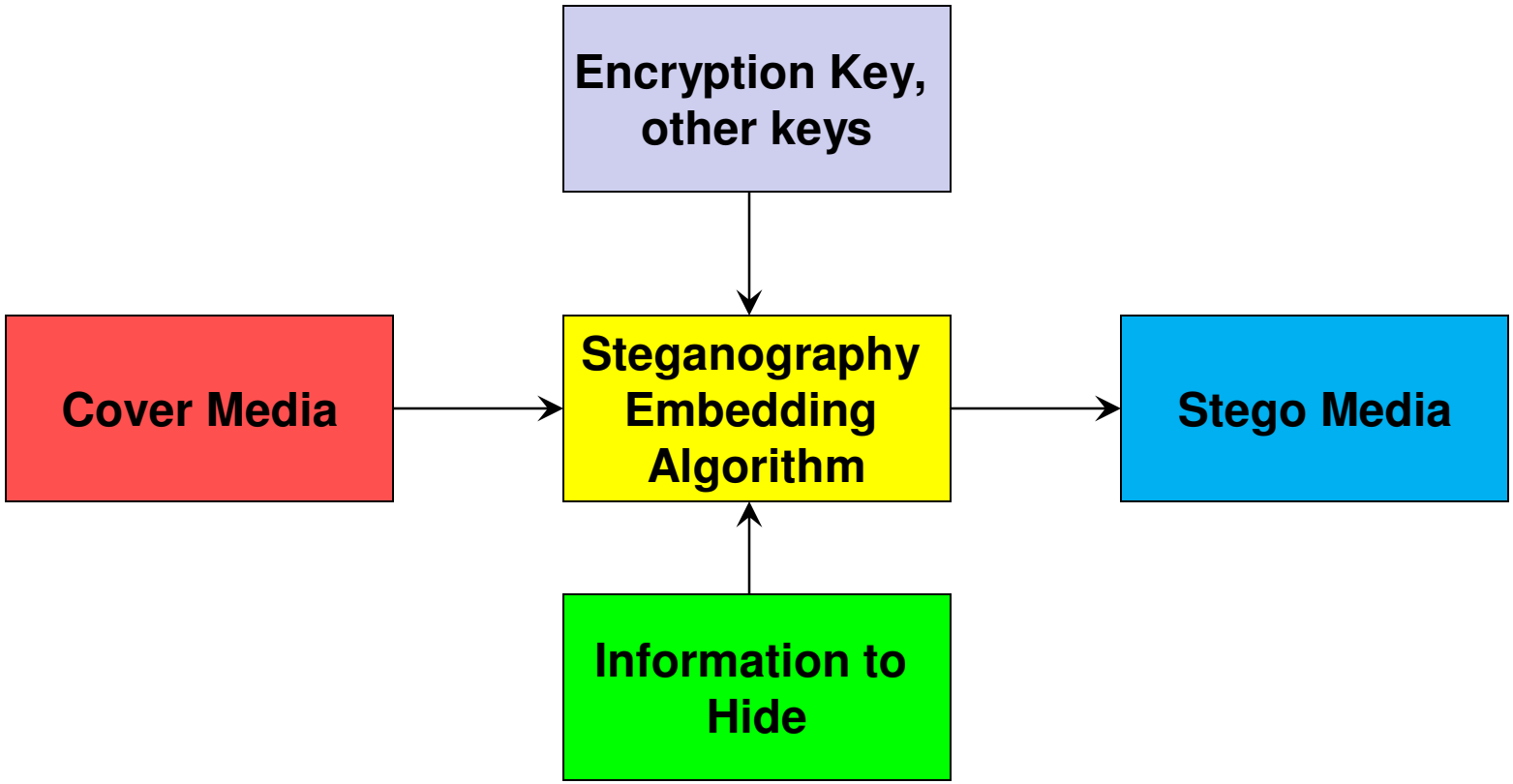
Dr. Natarajan Meghanathan
Associate Professor of Computer Science
Jackson State University, Jackson MS 39217
E-mail: natarajan.meghanathan@jsums.edu

# Steganography

- Steganography is the science of hiding information by embedding the hidden (secret) message within a cover media (for example, an image, audio or video carrier file) in such a way that the hidden information cannot be easily perceived to exist for the unintended recipients of the cover media.
- Steganography hides the fact that the secret communication does not exist.
- Different from Cryptography:
  - Cryptography techniques have been widely used to encrypt the plaintext data, transfer the ciphertext over the Internet and decrypt the ciphertext to extract the plaintext at the receiver side.
  - However, with the ciphertext not really making much sense when interpreted as it is, a hacker or an intruder can easily perceive that the information being sent on the channel has been encrypted and is not the plaintext.
  - This can naturally raise the curiosity level of a malicious hacker or intruder to conduct cryptanalysis attacks on the ciphertext (i.e., analyze the ciphertext vis-à-vis the encryption algorithms and decrypt the ciphertext completely or partially).

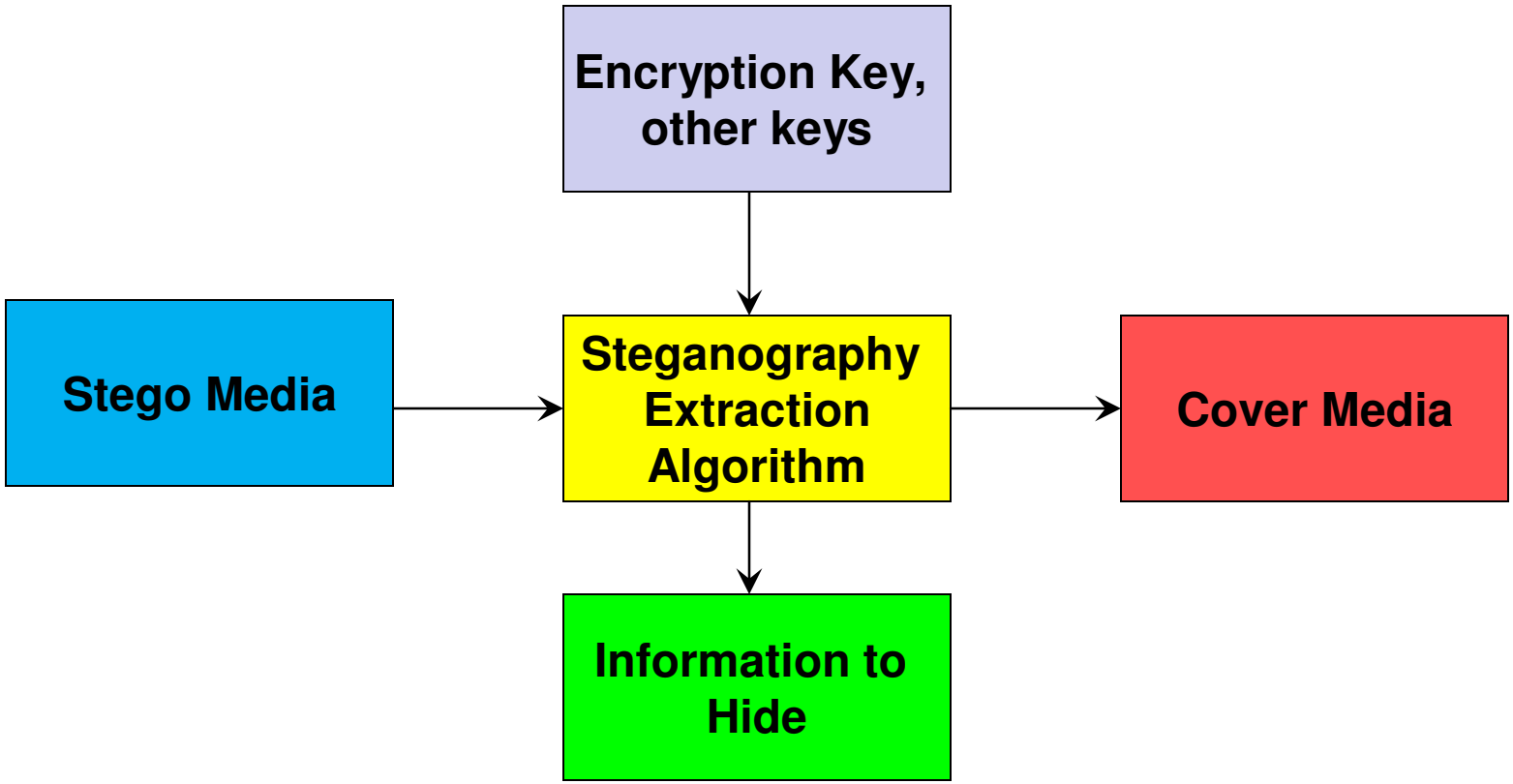
# Steganography: Hiding Model

- Steganography takes advantage of the redundancy of data in the cover media to insert the “extra” information so that there could be no perceptible change to the cover media.



# Steganography: Extraction Model

- The extraction process should be possible without the cover.
- The extraction algorithm can also be applied on any cover, whether or not it contains a secret message.



# Information Hiding in Noisy Data

- The cover media (images or digital sound) typically has redundancies in the form of a noise component (i.e., insignificant parts) and the general principle underlying most steganographic methods is to place the secret message in the noise component of the signal.
- The cover can be typically considered as a sequence of bits
  - In the case of a digital sound, this sequence is the sequence of samples over time
  - In the case of a digital image, a sequence can be obtained by vectorizing the image (i.e., by lining up the gray-scale or color values in a left-to-right and top-to-bottom order)
- A simple way to hide information in a sequence of binary numbers (bytes) is to replace the least significant bit (LSB) of each byte with a bit of the secret message or replace the LSB of bytes according to a sequence of pseudo random numbers generated at both sides.
- For better security, the embedding algorithm should not significantly alter the statistical profile of the cover.

# Random Access Cover vs. Stream Cover

- A cover is called a random access cover if the sender has access to the entire sequence of elements, a priori, in the embedding process.
- A cover is called a stream cover if the sender has no access to the entire sequence of numbers in the embedding process – for example, an application that stores information in digital audio files while they are being recorded.

- With a stream cover, it may be difficult to evenly spread the secret information all over the cover and one has to insert the secret info into the stream as it is transmitted; whereas, with a random access cover, the cover bits used to hide the secret information can be evenly selected with a suitable pseudo random selection function from all parts of the cover.
- The main drawback with random access covers is that the cover size is often shorter than that of stream covers; there could be chances of collisions if the same cover bit is repeatedly selected for hiding the secret information.

# Types of Steganography

- Secret Key Steganography
  - The sender chooses a cover and embeds the secret message into the cover using a secret key (for example: the seed for the pseudo random number generator for the byte sequence) known only to the sender and receiver.
  - The receiver can only extract the hidden message if he knows the secret key.
  - Anyone who does not know the secret key should not be able to obtain evidence of the encoded information.
  - The cover and the stego-objects can be perceptually similar.
  - Secret key steganography requires the prior exchange of a secret key before the actual communication (subverts the original intention of invisible communication!!)

  - Solution: Send the secret key using public-key cryptography

# Public Key Steganography

- Alice (sender), Bob (receiver), Wendy (attacker)
- Alice encrypts the secret message with Bob's public key to obtain a random-looking ciphertext and embeds it in the cover and transmits the same as the stego media.
- Bob will first extract the hidden ciphertext from the stego media received and then decrypt the ciphertext using his private key.
- The embedding/extraction algorithms and the encryption/decryption algorithms are assumed to be publicly known.
- If Wendy happens to receive the Stego media, she can extract the hidden ciphertext, which will merely appear as some random bits; but, she will not be able to decrypt it.
- Steganographic Key Exchange Protocol
- If the public key of Bob is not known to Alice, Bob can send his public key to Alice embedded in a cover. However, this could be prone to Man-in-the-Middle attacks and to avoid such attacks, the public-key certificate of Bob has to be sent instead.
- Alice could generate a session key and send it as the secret message to Bob through public key steganography. The session key could be later used for embedding the actual secret information through secret key steganography.

# Security of Steganography Systems

- A steganography system is insecure if an attacker is able to prove the existence of a secret message.
- To analyze the security of a steganography system, we must assume that an attacker has unlimited computation power and is able and willing to perform a variety of attacks.
- If he cannot confirm his hypothesis that a secret a message is embedded in a cover, then a system is theoretically secure.

- Perfect Security:
- If the selection of a cover is represented as a random variable *C*, a steganography system is (theoretically) perfectly secure if the process of embedding a secret message in a cover does not alter the probability distribution of C.
- The relative entropy (measure of the difference) between the probability distributions of the cover and the stego should be zero for a perfectly secure steganography system.
- If the relative entropy between the two probability distributions is $\leq \varepsilon$, then the steganography system is said to be $\varepsilon$-secure ($\varepsilon > 0$).

# Security of Steganography Systems

- Detecting Secret Messages
- *Type-I error:* An attacker falsely detects a hidden message in a cover that does not actually contain information.
- *Type-II error:* An attacker fails to detect a hidden message.
- For a perfectly secure steganography system, the probability that an attacker makes a Type-II error is 1.
- For a ε-secure steganography system with the probabilities α and β that a passive attacker makes a Type-I and Type-II error respectively, the following relationship holds. Note that for a ε-secure system, α + β need not be 1.

$$\left[\alpha \log_2 \frac{\alpha}{1-\beta} + (1-\alpha)\log_2 \frac{1-\alpha}{\beta}\right] \leq \varepsilon$$

- For ε-secure systems with no Type-I errors (i.e., α = 0), $\beta \geq 2^{-\varepsilon}$
- If ε = 0 (i.e., perfectly secure systems for which α + β = 1), β = 1
- As ε increases, β decreases (i.e., as the probability distributions of the cover and stego become increasing dissimilar, the probability that an attacker fails to detect a hidden message decreases).

# Security of Steganography Systems

$$\left[\alpha \log_2 \frac{\alpha}{1-\beta} + (1-\alpha)\log_2 \frac{1-\alpha}{\beta}\right] \leq \varepsilon$$

| β | α = 0.1 | α = 0.2 | α = 0.3 | α = 0.4 | α = 0.5 | α = 0.6 | α = 0.7 | α = 0.8 | α = 0.9 |
|---|---|---|---|---|---|---|---|---|---|
| 0.1 | 2.53594 | 1.966015 | 1.48966 | 1.083007 | 0.736966 | 0.449022 | 0.22169 | 0.06406 | 0 |
| 0.2 | 1.652933 | 1.2 | 0.840637 | 0.550978 | 0.321928 | 0.150978 | 0.040637 | 0 | 0.052933 |
| 0.3 | 1.145731 | 0.770559 | 0.488957 | 0.277058 | 0.125769 | 0.03258 | 0 | 0.037124 | 0.167817 |
| 0.4 | 0.794436 | 0.483007 | 0.265148 | 0.116993 | 0.029447 | 0 | 0.031163 | 0.13203 | 0.326466 |
| 0.5 | 0.531004 | 0.278072 | 0.118709 | 0.029049 | 0 | 0.029049 | 0.118709 | 0.278072 | 0.531004 |
| 0.6 | 0.326466 | 0.13203 | 0.031163 | 0 | 0.029447 | 0.116993 | 0.265148 | 0.483007 | 0.794436 |
| 0.7 | 0.167817 | 0.037124 | 0 | 0.03258 | 0.125769 | 0.277058 | 0.488957 | 0.770559 | 1.145731 |
| 0.8 | 0.052933 | 0 | 0.040637 | 0.150978 | 0.321928 | 0.550978 | 0.840637 | 1.2 | 1.652933 |
| 0.9 | 0 | 0.06406 | 0.22169 | 0.449022 | 0.736966 | 1.083007 | 1.48966 | 1.966015 | 2.53594 |

If $\alpha + \beta > 1$ or $\alpha + \beta < 1$, then $\varepsilon > 0$. This is because either of the two terms is $> 0$.

If $\alpha + \beta = 1$, then $\varepsilon \geq 0$. Thus, for a perfectly secure steganography system, we need the probability of a Type-1 error and probability of a Type-2 error to add up to 1.

# Least Significant Bit (LSB)-based Substitution

- The embedding process consists of choosing a subset $\{j_1, \ldots, j_{l(m)}\}$ of cover elements and performing the substitution operation $LSB(c_{j_i}) = m_i$ ($m_i$ can be either 1 or 0).
  - One can also change more than one bit of the cover-element – for example, by storing two message bits in the two least significant bits of one cover-element.
- In the extraction process, the LSB of the selected cover-elements are extracted and used to reconstruct the secret message.

**A Generic LSB-based Embedding Algorithm**

**Input**: cover $\mathbf{C}$
**for** $i = 1$ to Length($m$) **do**
  Compute index $j_i$ where to store the $i^{\text{th}}$ message bit of $m$
  $s_{j_i} \leftarrow \text{LSB}(\mathbf{c}_{j_i}) = m_i$
**end for**
**Output**: stego-object $\mathbf{S}$

# (LSB)-based Substitution

**A Generic LSB-based Extraction Algorithm**

**Input:** cover $\mathbf{C}$

**for** $i = 1$ to Length($m$) **do**

Compute index $j_i$ where to store the $i^{\text{th}}$ message bit of $m$

$S_{j_i} \leftarrow \text{LSB}(\mathbf{C}_{j_i}) = m_i$

**end for**

**Output:** stego-object $\mathbf{S}$

- Embedding Strategies
  - Use every cover-element starting from the first one. Since, the number of bits in the secret message is typically less than the length of the cover, the strategy would lead to a situation where the embedding process will be finished long before the end of the cover and the first part of the cover will have different statistical properties than the second part, where no modifications have been made.

# Embedding Strategies for LSB-based Substitution

- Random Interval Method
- Both communication partners share a stego-key K that would be used as a seed for a pseudorandom number generator that can create a random sequence k1, …, kl(m) and use the cover-elements with indices $j_1 = k_1$; $j_i = j_{i-1} + k_i$, for $2 \le i \le$ Length(m)

**Input**: cover $\mathbf{C}$
Generate random sequence $k_i$ using seed $k$
$n \leftarrow k_1$
**for** $i = 1$ to Length($m$) **do**
  $Sn \leftarrow \mathrm{LSB}(\mathbf{C}n) = m_i$
  $n \leftarrow n + k_i$
**end for**
**Output**: stego-object $\mathbf{S}$

**Embedding Algorithm**

**Input**: stego-object $\mathbf{S}$
Generate random sequence $k_i$ using seed $k$
$n \leftarrow k_1$
**for** $i = 1$ to Length($m$) **do**
  $m_i = \mathrm{LSB}(\mathbf{S}n)$
  $n \leftarrow n + k_i$
**end for**
**Output**: message m

**Extraction Algorithm**

# Problem of Collisions with the Random Interval Method-based LSB

- Collision – One index (jk) could appear more than once in the pseudorandom sequence as the output of the pseudorandom number generator is not restricted in any way.
- If a collision occurs, the sender will possibly try to insert more than one message bit into a cover-element, leading to corruption of the message.
- If the message is quite short compared with the number of cover-elements, the probability of collisions is negligible and the corrupted bits could be reconstructed using an error-correcting code.
- This is however feasible only for quite a short secret message, as the probability of a collision, *p*, increases rapidly as the length of the secret message to be embedded increases.

$$p \approx 1 - \exp\left(-\frac{\ell(m)[\ell(m)-1]}{2\ell(c)}\right)$$

| **Length(c)** | **Length(m)** | **p** |
|---|---|---|
| 600 x 600 | 200 | 0.05 |
| pixels | 600 | 0.4 |

# Handling Collisions with the Random Interval Method-based LSB

- Tracking the Pseudorandom sequence
- The sender could keep track of all the cover-element indices, that have already been used for the communication, in a set B.
- If during the embedding process, one specific index generated according to the pseudorandom sequence has not been used before (i.e., is not in the set B), then the sender adds the index to B and continues to use it. If the index generated is already in B, the sender discards the index and chooses another index pseudorandomly.
- At the receiver side, a similar technique is applied.
- Using a Pseudorandom Permutation of Cover-bit Indices
- Instead of embedding the message bits sequentially in the increasing order of the index values for the cover, each element in the pseudorandom number sequence could be generated anywhere from 1 to Length(*c*), where *c* is the cover and the secret message bits could be embedded at the cover-bits corresponding to the index values generated.
- Collisions are handled similar to the approach suggested above.

# Cover-Regions and Parity-bits

- A cover-region is any non-empty subset of the cover **C** = $\{c_1, \ldots, c_{l(c)}\}$
- The idea is to generate a pseudorandom sequence of disjoint cover-regions, using a stego-key as the seed, and store only one bit of the secret message in a whole cover-region rather than in a single element.
- The secret bit to be hidden inside a cover-region is embedded as the parity-bit $p(I)$ for the cover-region $I$ chosen.

$$p(I) = \sum_{j \in I} LSB(c_j) \bmod 2$$

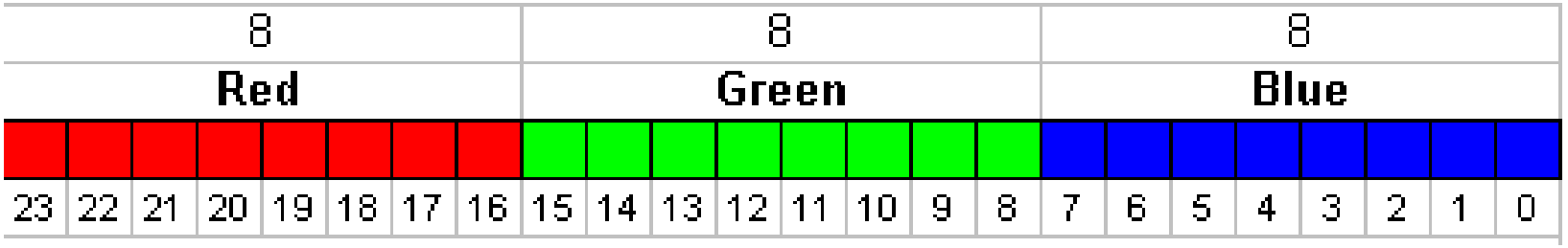
- During the embedding step, $l(m)$ disjoint cover-regions $I_i$ ($1 \leq i \leq l(m)$) are selected, each encoding one secret bit $m_i$ in the parity bit $p(I_i)$.
- If the parity bit of the cover-region $I_i$ does not match with the secret bit $m_i$ to encode, one LSB of a randomly chosen cover-element in $I_i$ is flipped. This will result in $p(I_i) = m_i$.
- During the extraction process at the receiver, the parity bits of all the selected cover-regions (generated according to the pseudorandom sequence) are calculated and lined up to reconstruct the message.

# Pixel

- A pixel (picture element) represents a single point, the smallest unique element, in an image.
- Each pixel has its own address and the address correspond to its co-ordinates.
- Pixels are typically arranged in a two-dimensional grid, and are often represented using dots or squares.
- Each pixel is a sample of the original image; the larger the number of pixels, the greater the resolution.
  - A 6 MP camera captures the image and stores it via 6 Million pixels.
- The number of distinct colors that can be represented by a pixel depends on the number of bits per pixel (bpp).
- Commonly used coloring models:
  - 8-bit monochromatic model – $2^8$ = 256 colors (gray-scale model) – A value of 0 indicates black and 255 indicates white color.
  - 24-bit RGB model that can represent $2^{24}$ = 16.8 million colors; Used in BMP (Bitmap), JPEG, TIFF image formats.

# 24-bits per Pixel - RGB Model

- 24 bits per pixel - Three 8-bit unsigned integer (0 through 255) represent the intensities of red, green, and blue (R, G, B).
- An (R, G, B) value of (0, 0, 0) represents Black and an (R, G, B) value of (255, 255, 255) represents White.




- (255, 0, 0) – Red; (0, 255, 0) – Green; (0, 0, 255) - Blue

# Palette-based Images

- In a palette-based image, only a subset of the colors from a specific color space is used to colorize the image.
- Every palette-based image consists of two parts: (I) a palette specifying the N colors as a list of indexed pairs ($i$, $\mathbf{c}_i$), assigning a color vector $\mathbf{c}_i$ to every index $i$ and (II) the actual image with its pixels identified with the palette index values corresponding to the color value they represent.

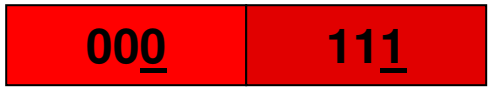
- The above approach can significantly reduce the file size if only a small number of colors are used throughout the image.
- Examples: Graphics Interchange Format (GIF) and Bitmap (BMP) format.

# Information Hiding in Palette-based Images

- Information is typically encoded in the image data and not the palette as there are only limited entries in the latter.
- Note that the image data has only the index values for the color representing the pixel and not its RGB values. The index values for neighboring pixels (that may have perceptually similar colors) may not be closer to each other. Hence, a simple LSB encoding of the index values will lead to perceptually different colors in the stego-object and an attacker may quiet easily determine the presence of hidden information.

| | |
|---|---|
| 000 | 255, 0, 0 |
| 001 | 125, 0, 0 |
| 010 | 75, 0, 0 |
| 011 | 100, 0, 0 |
| 100 | 200, 0, 0 |
| 101 | 150, 0, 0 |
| 110 | 25, 0, 0 |
| 111 | 225, 0, 0 |

←Palette Table (not sorted)




**Original Image Data**

**Before LSB-based Information Hiding**

The underlined bits are the LSBs


**LSB-encoded Image Data**

Assume data to be hidden is: **1 0**

**After LSB-based Information Hiding**

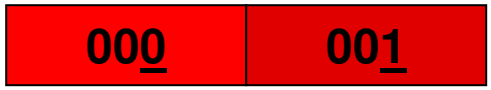
# Information Hiding in Palette-based Images

- Solution-1: "Sort" the colors in the palette so that the adjacent pixels are still perceptually similar after the embedding process.
  - Sort according to the Euclidean distance in the RGB space

$$d = \sqrt{R^2 + G^2 + B^2}$$

  - Sort according to the Luminance $Y = 0.299R + 0.587G + 0.114B$

| | |
|---|---|
| 000 | 255, 0, 0 |
| 001 | 225, 0, 0 |
| 010 | 200, 0, 0 |
| 011 | 150, 0, 0 |
| 100 | 125, 0, 0 |
| 101 | 100, 0, 0 |
| 110 | 75, 0, 0 |
| 111 | 25, 0, 0 |

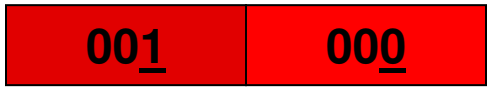



**Original Image Data**

←Palette Table (Sorted)

**Before LSB-based Information Hiding**

The underlined bits are the LSBs




**LSB-encoded Image Data**

Assume data to be hidden is: **1 0**

**After LSB-based Information Hiding**

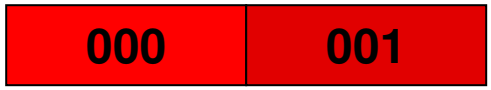
# Information Hiding in Palette-based Images

- Solution-2: For every pixel (or a pixel chosen for information hiding), the index value corresponds to the color of the pixel if its parity value [(R + G + B)/mod 2] matches with the secret bit to encode; otherwise the index value will correspond to that of the next-closest color, in the Euclidean space (assume ties are broken based on the index values; in our case – the larger index value is assumed to be the next-closest color), whose parity value matches with the secret bit to encode.

Assume data to be hidden is: **1 0**

| | |
|---|---|
| 000 | 255, 0, 0 |
| 001 | 225, 0, 0 |
| 010 | 200, 0, 0 |
| 011 | 150, 0, 0 |
| 100 | 125, 0, 0 |
| 101 | 100, 0, 0 |
| 110 | 75, 0, 0 |
| 111 | 25, 0, 0 |

←Palette Table (Sorted)

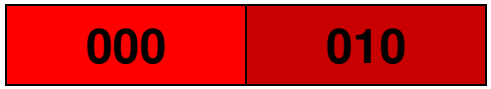



**Original Image Data**

**Before Information Hiding**

000 → (255 + 0 + 0) mod 2 = 1 matches with the first secret bit '1'

001 → (225 + 0 + 0) mod 2 = 1; the Next closest color whose parity matches with the second secret bit '0' is (200, 0, 0) → 010

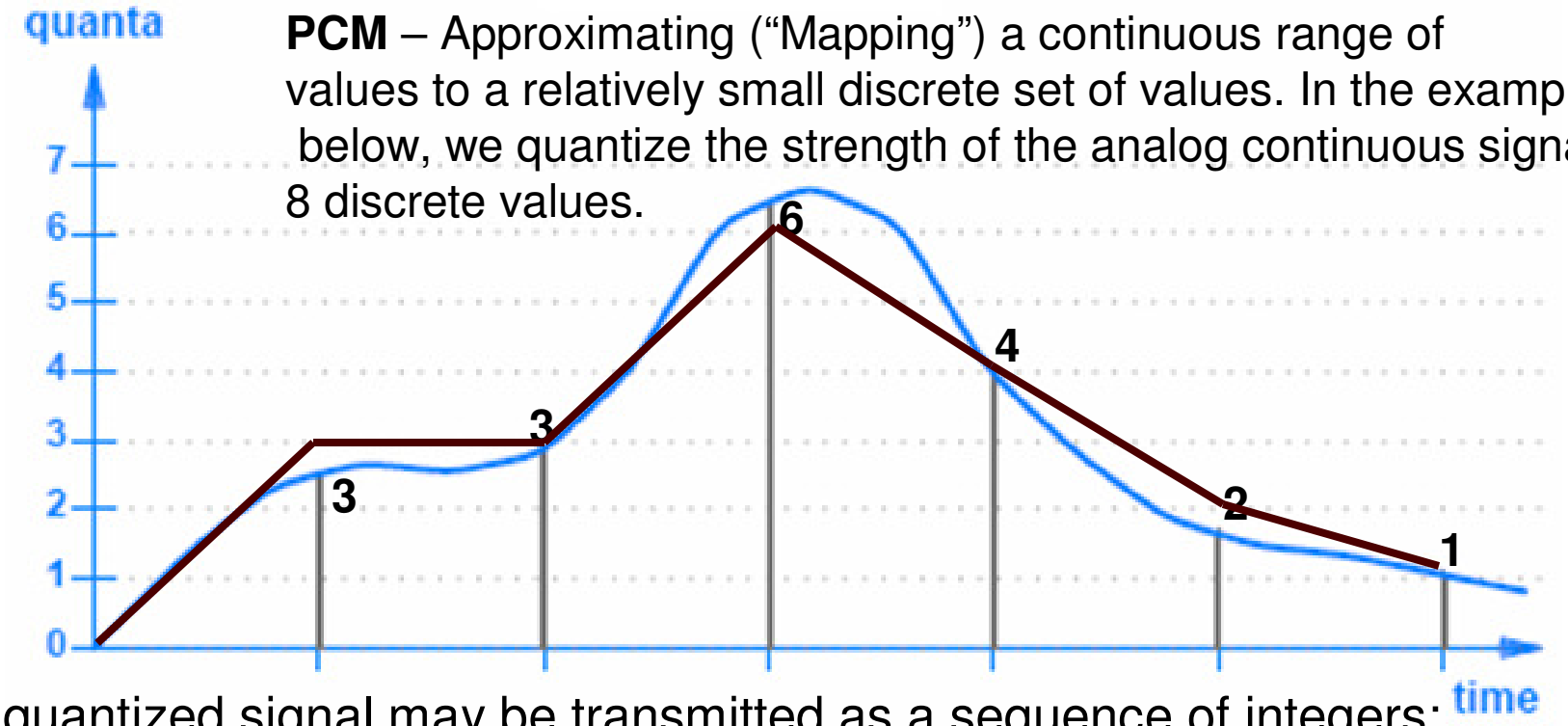



**After Information Hiding**

# Quantization of Continuous Signals using Pulse Code Modulation

- Continuous signals, such as audio signals, could be quantized to discrete signals using techniques like Pulse Code Modulation (PCM) and Predictive coding.

**PCM** – Approximating ("Mapping") a continuous range of values to a relatively small discrete set of values. In the example below, we quantize the strength of the analog continuous signal to 8 discrete values.



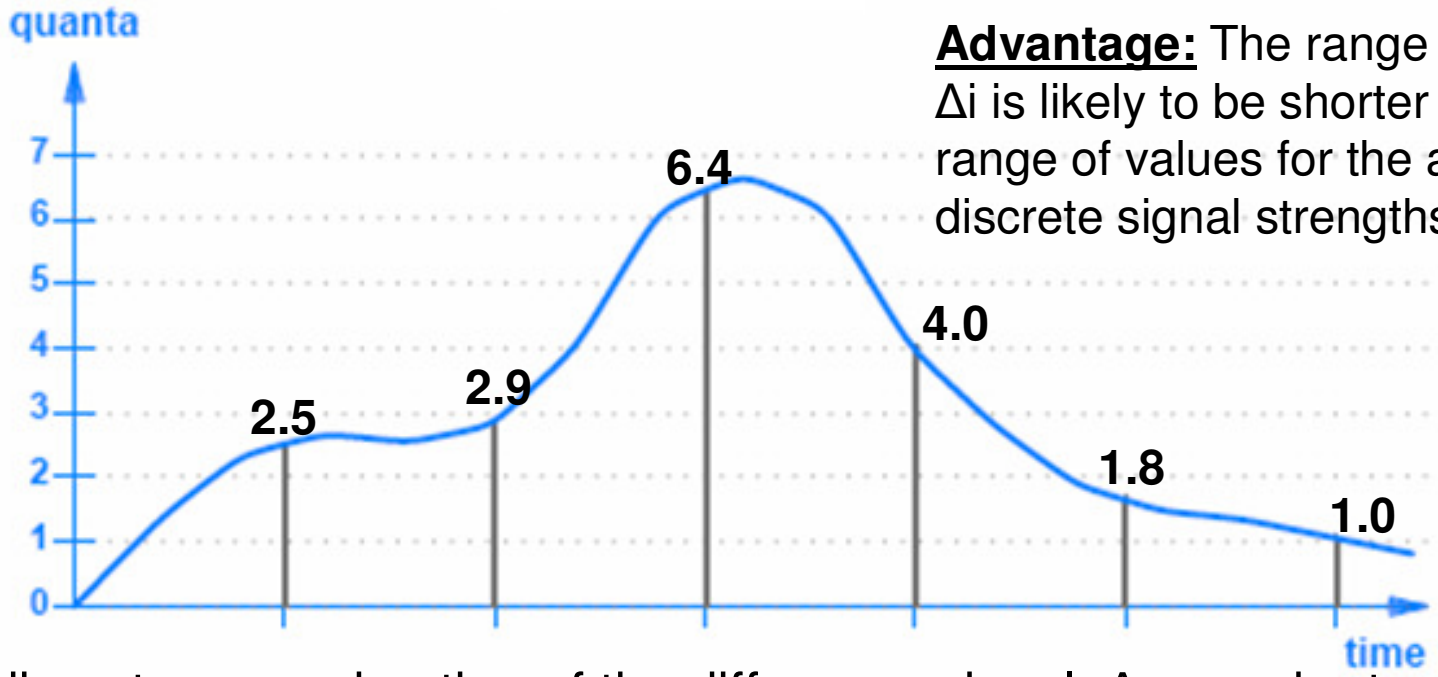

The quantized signal may be transmitted as a sequence of integers:

011 011 110 100 010 001

3 3 6 4 2 1

# Quantization of Continuous Signals using Predictive Coding

- With Predictive Coding, we quantize the difference between the signal strengths of the successive samples ($x_i - x_{i-1}$) and transmit the difference as a discrete value, rounded to the nearest integer.




**Advantage:** The range of values for Δi is likely to be shorter than the range of values for the absolute discrete signal strengths.

The discrete approximation of the difference signal, $\Delta_i$, may be transmitted as a sequence of signed integers:

| +3 | 0 | +4 | -2 | -2 | -1 |
|---|---|---|---|---|---|
| (2.5-0) | (2.9-2.5) | (6.4-2.9) | (4-6.4) | (1.8-4) | (1-1.8) |

# Steganography through Quantization using Predictive Coding

- The stego-key could consist of a table which assigns a specific bit to every possible value of the quantized difference signal, $\Delta_i$.
- An example Stego-key table

| $\Delta_i$ | -4 | -3 | -2 | -1 | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |

- In order to store the ith message bit in the cover-signal, the quantized difference signal $\Delta_i$ is computed. If $\Delta_i$ does not match (according to the secret table) with the secret bit to be encoded, $\Delta_i$ is replaced by the nearest $\Delta_j$ where the associated bit equals the secret message bit.
- At the receiver side, the message is decoded according to the difference signal $\Delta_i$ and the stego-key table.

# Example for Steganography through Quantization using Predictive Coding

- Stego-key table

| $\Delta_i$ | -4 | -3 | -2 | -1 | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |

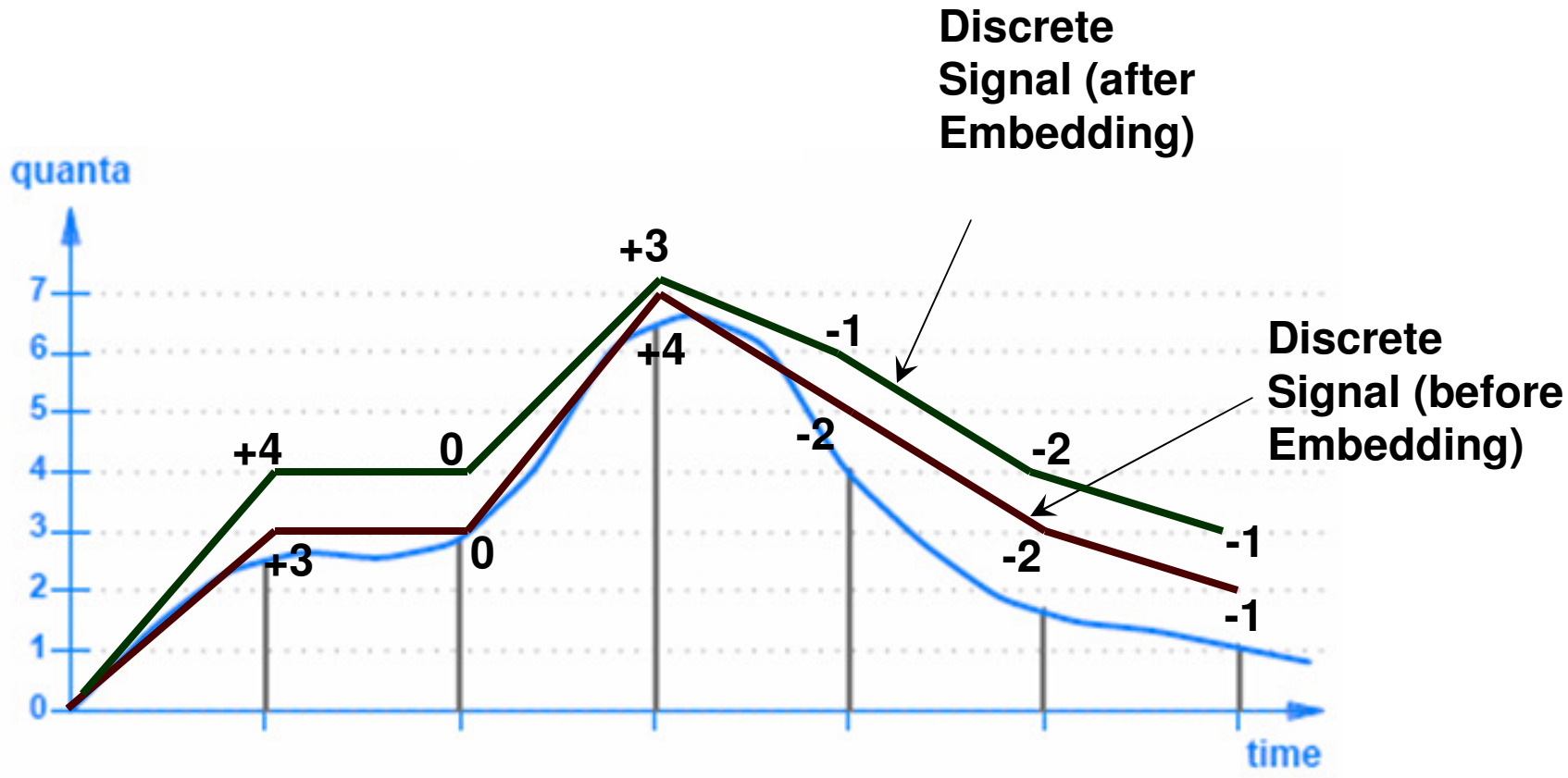
The discrete approximation of the difference signal, $\Delta_i$, may be transmitted as a sequence of signed integers:

| +3 | 0 | +4 | -2 | -2 | -1 |
|---|---|---|---|---|---|
| (2.5-0) | (2.9-2.5) | (6.4-2.9) | (4-6.4) | (1.8-4) | (1-1.8) |

- Let the secret message to be hidden be: 1 1 0 1 0 1
- Rule: Check the entry for the Δi value in the Stego-key table. If the associated bit does not match with the secret message bit, then look at the entries for ($\Delta_i \pm 1$, $\Delta_i \pm 2$ and so on) until there is a match. The '+' could be assumed to be given preference over '-' and vice-versa (i.e., alternatively – to balance out) in the ±, where ever possible. The transmitted quantized difference signals, in the above example, would be:

| | | | | | | |
|---|---|---|---|---|---|---|
| | +4 | 0 | +3 | -1 | -2 | -1 |
| Secret bits: | 1 | 1 | 0 | 1 | 0 | 1 |

# Example for Steganography through Quantization using Predictive Coding



# Information Hiding through Cover Generation

- All of the embedding methods discussed so far, hide the secret information in a specific cover by applying an embedding algorithm.
- There exists steganographic applications that generate a digital object only for the purpose of being a cover for secret communication. We will see one such application that is based on context-free grammars.
- Review of Context-free Grammar (CFG)
- Let $G = <V, \Sigma, \Pi, S>$ be a CFG, where V is the set of variables, $\Sigma$ the set of terminal symbols, $\Pi \subseteq V \times (V \cup \Sigma)^*$ the set of productions; $S \in V$ the start symbol.
- The productions can be seen as a substitution rule; they convert a variable into a string containing terminal or variable symbols.
- A string $s \in L(G)$ is a sequence of terminal symbols produced successively from the start symbol S by substituting variables by sequences of terminal or variable symbols according to $\Pi$.
- If for every string $s \in L(G)$, there exists only one way *s* can be generated from the start symbol, the grammar G is said to be unambiguous.
- Note that a grammar is context-free if the left hand side of a production rule is a single non-terminal symbol.

# Example for Context-free Grammar

- CFG G = < {S, A, B, C}, {A, …., Z, a, …, z}, Π, S} >

$$\Pi = \{\ S \to \texttt{Alice}\ B,\ S \to \texttt{Bob}\ B,\ S \to \texttt{Eve}\ B,\ S \to \texttt{I}\ A,$$
$$A \to \texttt{am working},\ A \to \texttt{am lazy},\ A \to \texttt{am tired},$$
$$B \to \texttt{is}\ C,\ B \to \texttt{can cook},$$
$$C \to \texttt{reading},\ C \to \texttt{sleeping},\ C \to \texttt{working}\}$$

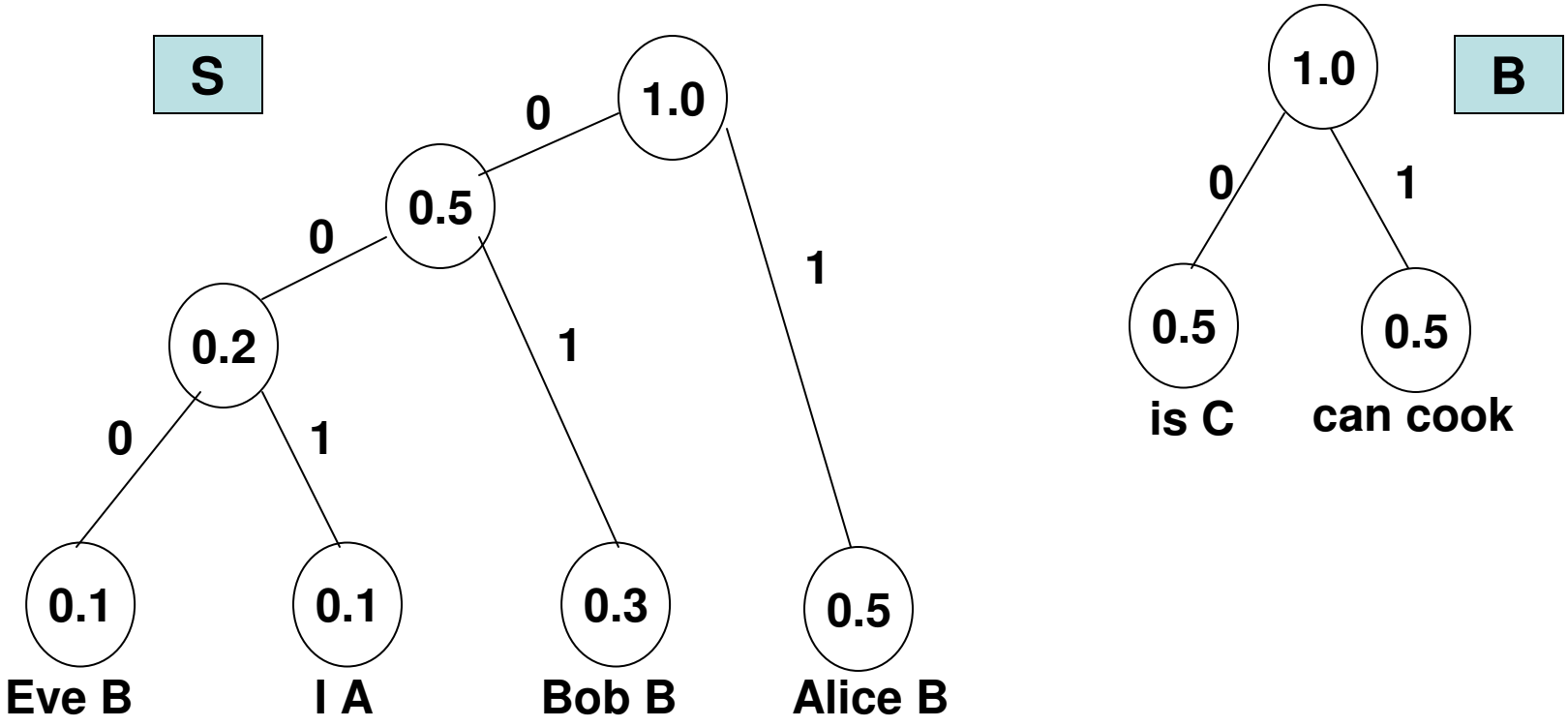
- Example 1 for Derivation of Sentence: I am lazy
- S → I A
- S → I am lazy
- Example 2 for Derivation of Sentence: Alice is reading
- S → Alice B
- S → Alice is C
- S → Alice is reading

# Steganography through Automated Generation of English Texts using CFG

- Unambiguous grammars can be used as a steganographic tool.
- Given a set of productions (used as stego-key), we assign a probability to each possible production for variable $V_i$.
- The sender and receiver, each, could construct a Huffman tree for the set of all productions associated with variable $V_i$.
- For a given secret message to be sent, a sequence of English sentences are generated using the CFG and the Huffman tree.
- The bits of the secret message are scanned from left to right and an English sentence is derived by traversing the Huffman tree according to the next bits of the secret message until a node of the tree is reached. The start symbol is then substituted by the production which can be found at this node of the tree. This process is repeated until all the message bits are used and the string consists only of terminal symbols.
- The receiver would be able to decode the secret message by iteratively parsing through the Huffman trees of the production rules (beginning from that of the Start symbol), according to the sequence of sentences received.
- Important: The size of the secret message is assumed to be known to the receiver so that it can terminate the excess bits that get encoded due to the structure of the Huffman tree.

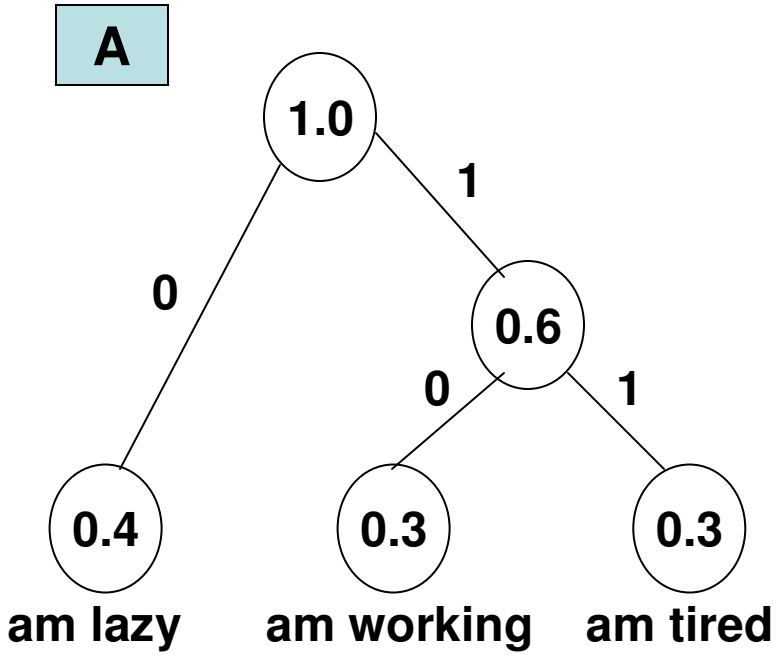
# Example for Automated Generation of English Texts using CFG

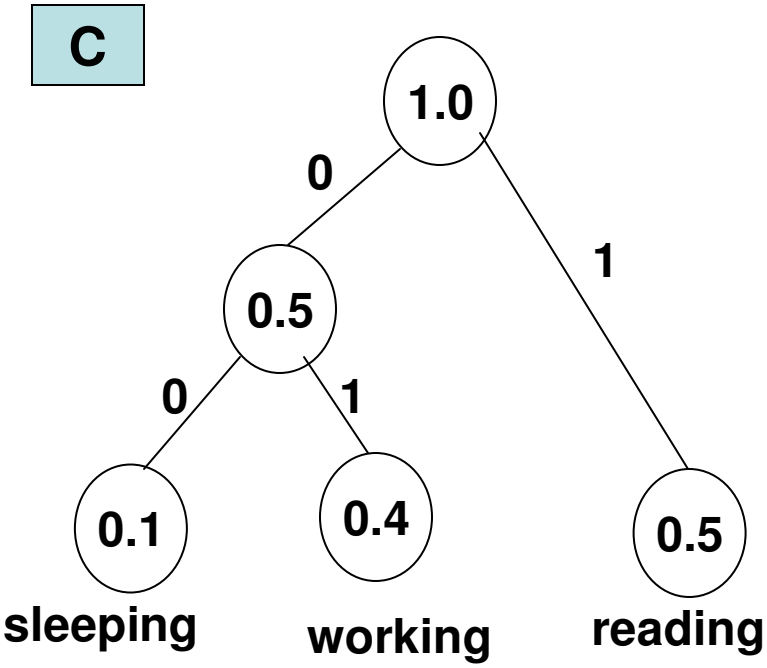
$$\Pi = \{\ S \to_{0.5} \texttt{Alice}\ B,\ S \to_{0.3} \texttt{Bob}\ B,\ S \to_{0.1} \texttt{Eve}\ B,\ S \to_{0.1} \texttt{I}\ A,$$
$$A \to_{0.3} \texttt{am working},\ A \to_{0.4} \texttt{am lazy},\ A \to_{0.3} \texttt{am tired},$$
$$B \to_{0.5} \texttt{is}\ C,\ B \to_{0.5} \texttt{can cook},$$
$$C \to_{0.5} \texttt{reading},\ C \to_{0.1} \texttt{sleeping},\ C \to_{0.4} \texttt{working}\}$$



# Example for Automated Generation of English Texts using CFG

$$\Pi = \{\ S \to_{0.5} \texttt{Alice}\ B,\ S \to_{0.3} \texttt{Bob}\ B,\ S \to_{0.1} \texttt{Eve}\ B,\ S \to_{0.1} \texttt{I}\ A,$$
$$A \to_{0.3} \texttt{am working},\ A \to_{0.4} \texttt{am lazy},\ A \to_{0.3} \texttt{am tired},$$
$$B \to_{0.5} \texttt{is}\ C,\ B \to_{0.5} \texttt{can cook},$$
$$C \to_{0.5} \texttt{reading},\ C \to_{0.1} \texttt{sleeping},\ C \to_{0.4} \texttt{working}\}$$



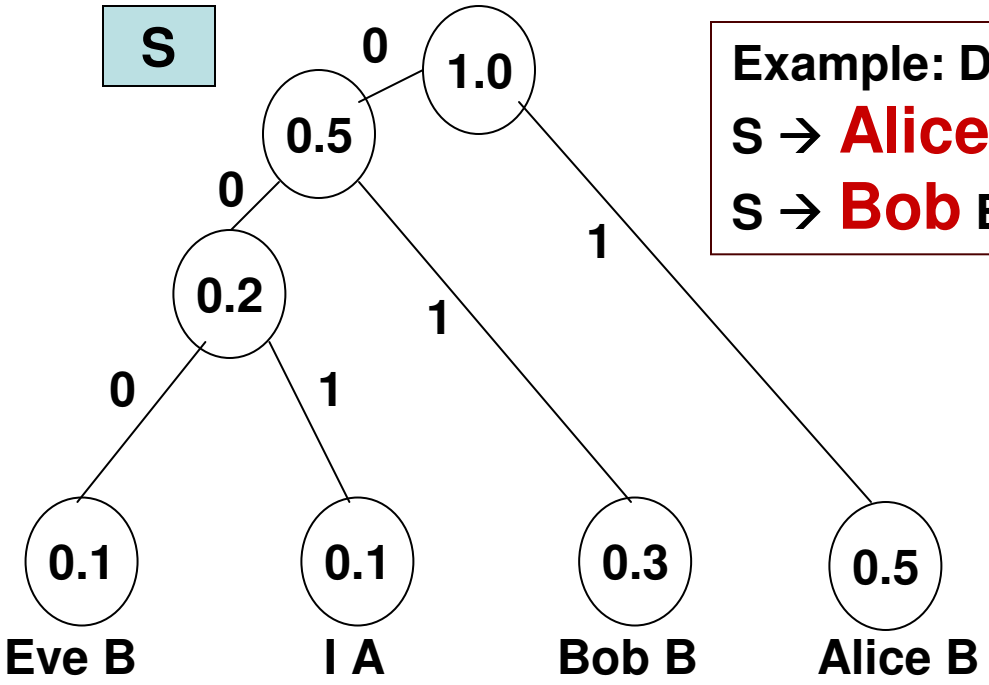



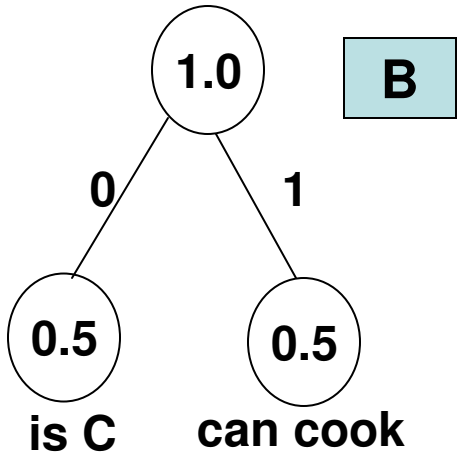
## Example 1

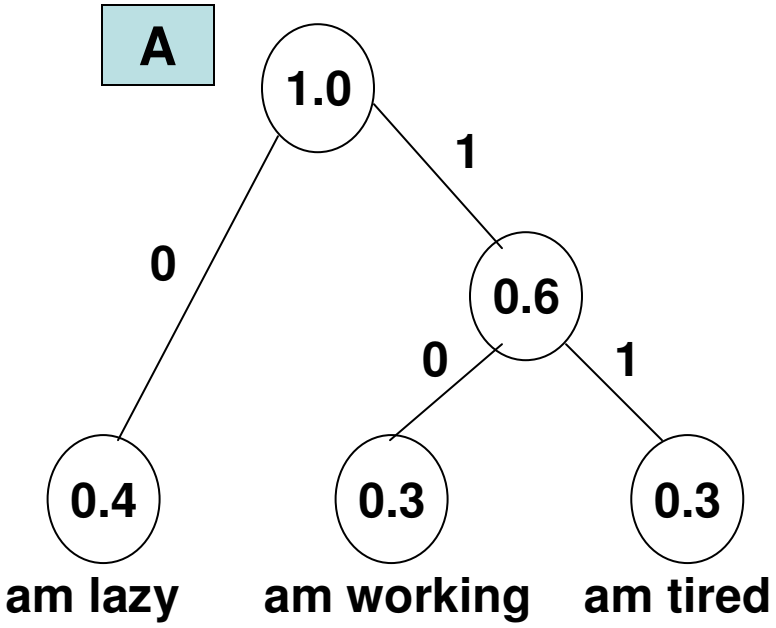
**Example: Derivation of Sentence for 110101**

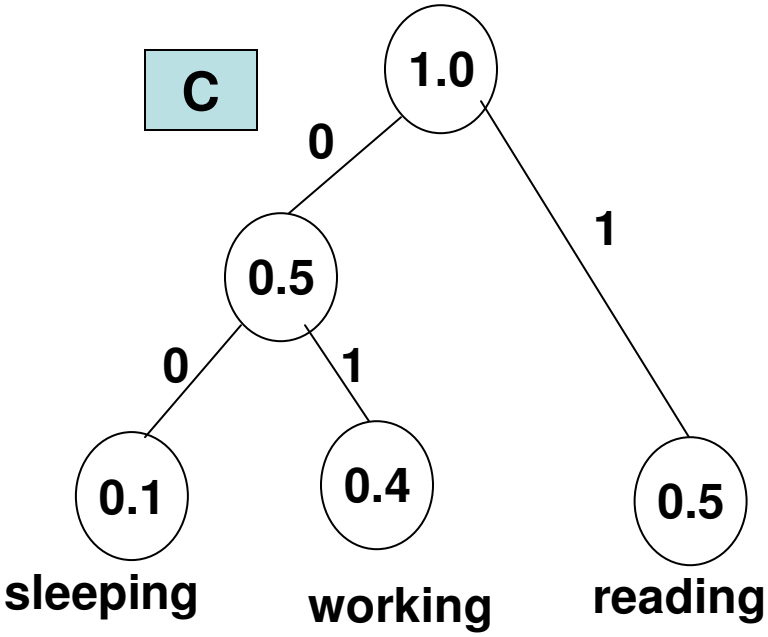
S → **Alice** B (1); B → **can cook** (1): 11

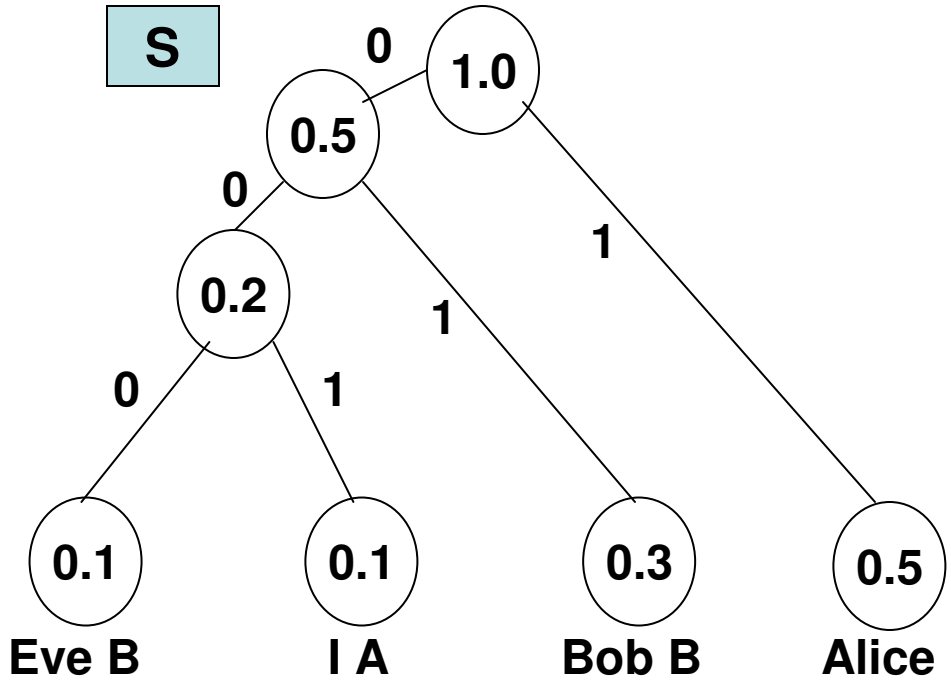
S → **Bob** B (01); B → **is** C (0); C → **reading** (1)





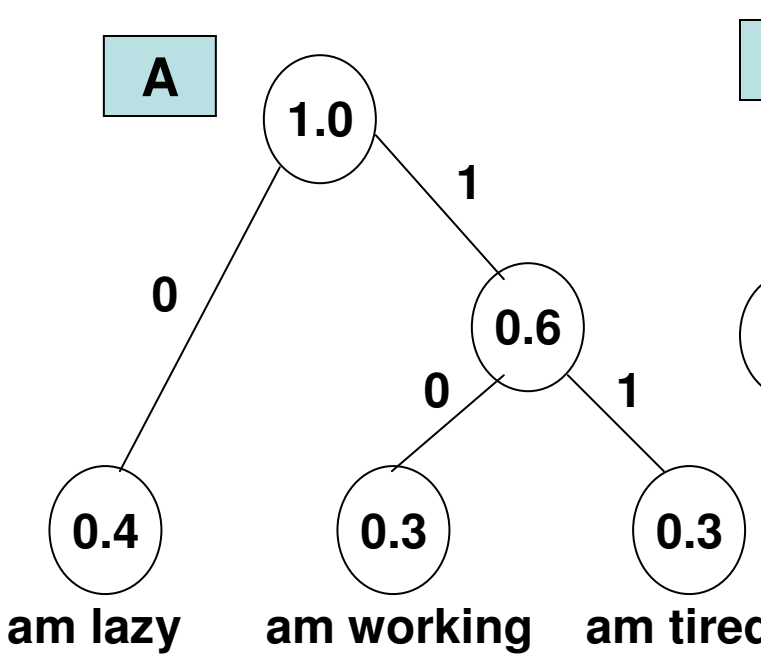

## Example 2

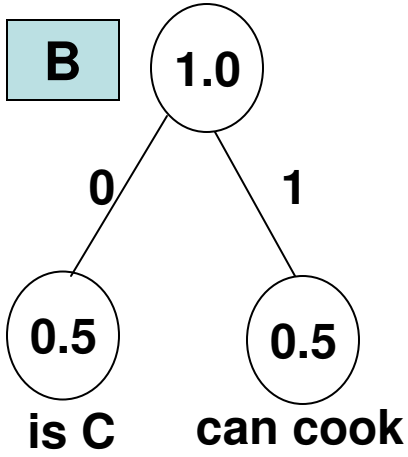
**Secret Message:** 1011010001110

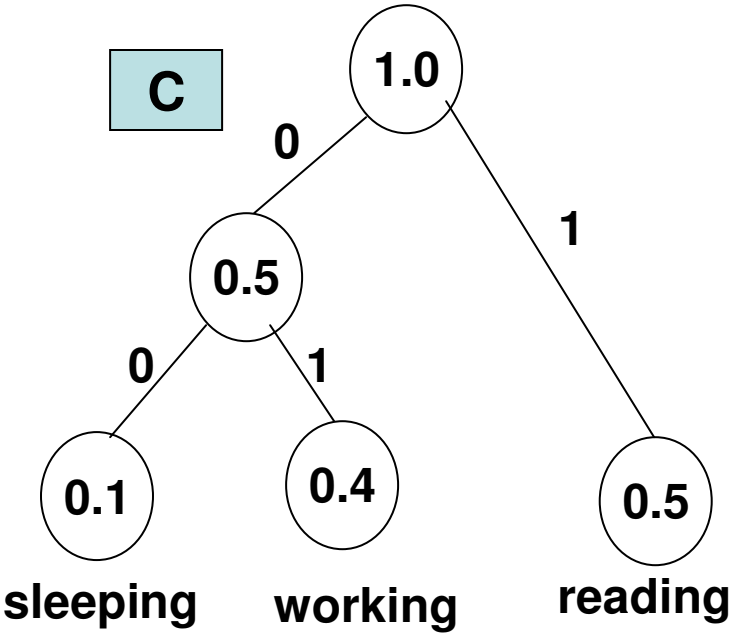
**Sequence of Sentences Generated**

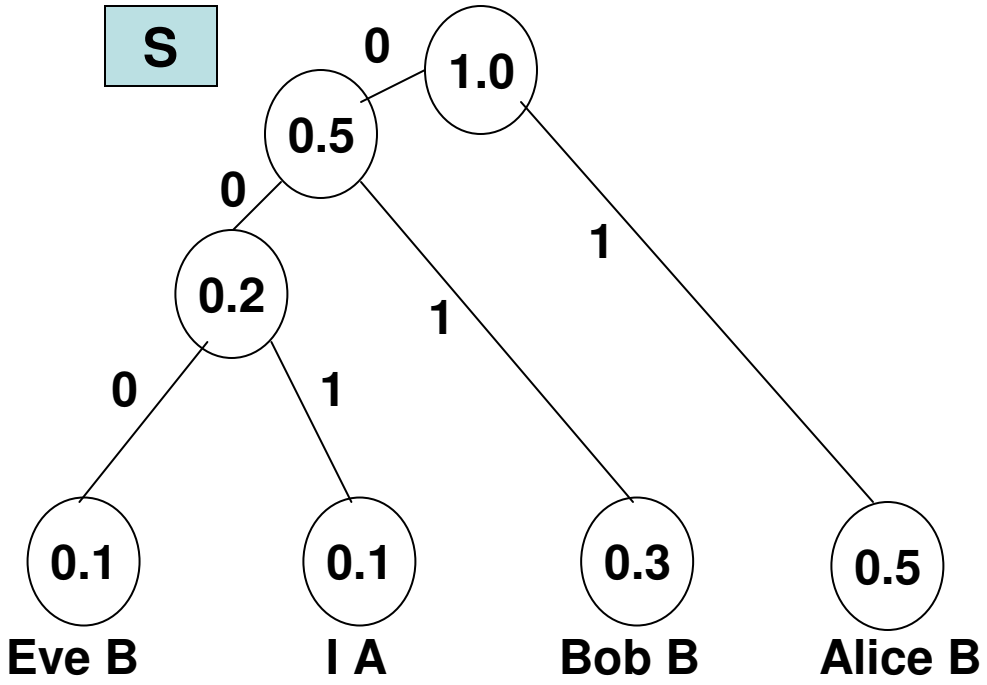
Alice is reading (101)
Alice is reading (101)
Eve can cook (0001)
Alice can cook (11)
Eve can cook (0001)

001 are dummy bits
Receiver knows message size



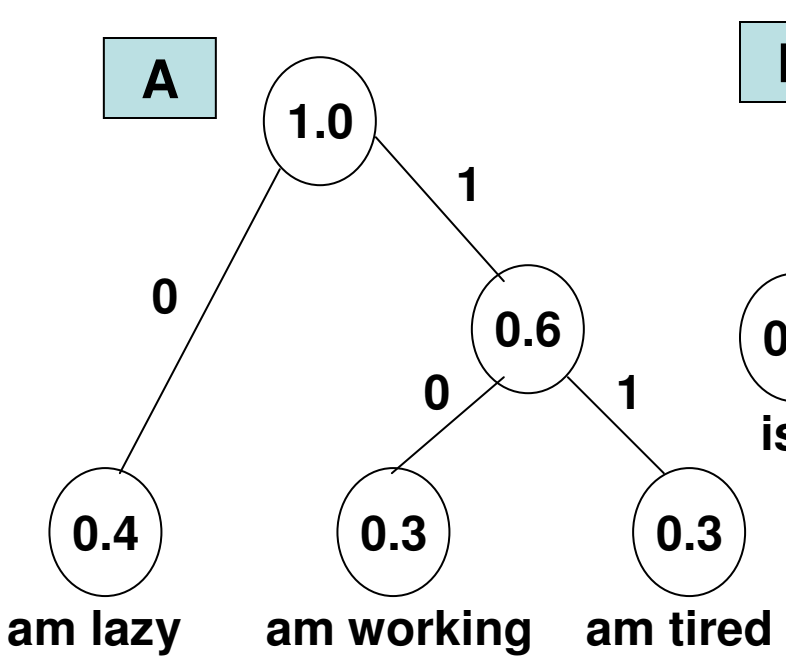


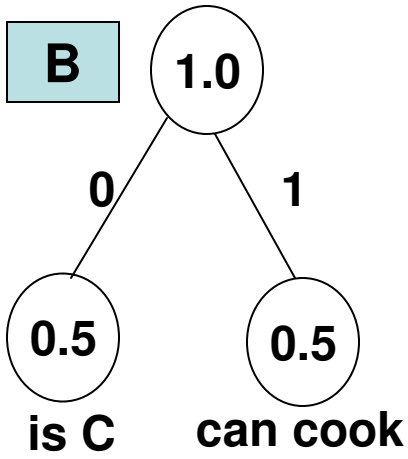

## Example 3

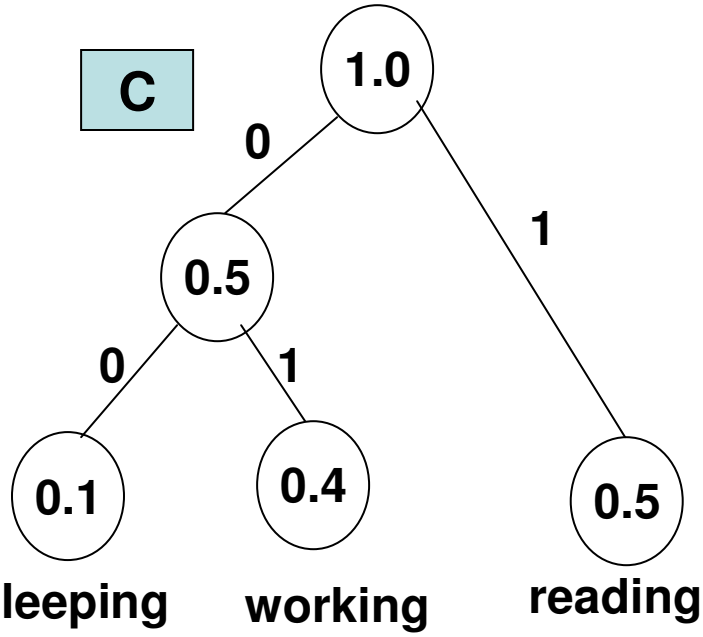
**Secret Message:** 011101010010001

**Sequence of Sentences Generated**

Bob can cook (011)
Alice is reading (101)
Bob is working (01001)
Eve can cook (0001)



# Steganography vs. Watermarking

- Steganography focuses covert (hidden) point-to-point communication between two parties.
- Steganographic methods are usually not robust against modification of the data, or have only limited robustness and cannot protect the embedded information against technical modifications that may occur during transmission.
- Watermarking has the additional notion of resilience against attempts to remove the hidden data.
  - Even if the existence of the hidden information is known, it should be hard for an attacker to destroy the embedded watermark (could be a number, text or image) without knowledge of a key.
- A popular application of watermarking is to give proof of ownership of digital data by embedding copyright statements.
  - For this application, the embedded information should be robust against manipulations that may attempt to remove it.
- A practical implication of the robustness requirement is that watermarking methods can typically embed much less information into cover-data than steganographic methods.
- Desired Characteristics of a Watermarking System: Imperceptibility, Robustness (cannot be modified), Noninvertibility (cannot be removed), Ability to recover with or without the original data