# Foundations of Computer Security

## Lecture 31: Languages and Encodings

Dr. Bill Young
Department of Computer Sciences
University of Texas at Austin

# Languages

A *language* may refer to a natural language such as English or Japanese. But it may also refer to any set of symbols used to form "strings."

Often, we'll use "language" to refer to the results of a series of experiments.

**Example 1:** Assume a two-sided coin. Symbols in the language are "H" (heads) and "T" (tails). A string describes the result of repeatedly flipping the coin. E.g., "HHTHTTTHTHTTHTT..."

**Example 2:** Assume a six-sided die. Symbols in the language are "1", "2", "3", "4", "5", "6". A string describes the result of repeatedly rolling the die. E.g., "425146261435261..."

# Encoding Properties

For any language, our goal is a binary encoding, with the following properties:

**lossless:** it must be possible to recover the entire original sequence of symbols from the transmission;

**uniquely decodable:** for any encoded string, there must be only one possible decoding;

**streaming:** there should be no breaks in the encoding.

# Unique Decodability

Suppose you roll a 6-sided die repeatedly. The following are some possible codes. *Are they lossless, uniquely decodable, streaming? Which is "best"?*

| **Roll** | **Naïve** | **Code 1** | **Code 2** |
|---|---|---|---|
| 1 | 000 | 0 | 00 |
| 2 | 001 | 10 | 01 |
| 3 | 010 | 110 | 10 |
| 4 | 011 | 1110 | 110 |
| 5 | 100 | 11110 | 1110 |
| 6 | 101 | 11111 | 1111 |

Sufficient (but not necessary) for unique decodability is the property of being *prefix-free*: That is, the string representing any symbol cannot be an initial prefix of the string representing any other symbol.

# Unique Decodability

Imagine a language containing the symbols A, B, C. *What's wrong with the following encoding?* A clue is that it is not prefix-free.

| **Sym** | **Code** |
|---|---|
| A | 1 |
| B | 0 |
| C | 10 |

An encoding can fail to have the prefix-free property, but still be uniquely decodable.

| **Sym** | **Code** |
|---|---|
| A | 1 |
| C | 111111110 |

Parsing such a language may require arbitrary "look-ahead."

# Finding a Coding

*How do you come up with an efficient encoding?* Use fewer bits for the symbols that occur more frequently.

Samuel Morse knew this instinctively. But Morse code doesn't satisfy our criteria for encodings. *Why not?*

| | | | |
|---|---|---|---|
| E | · | S | ··· |
| T | – | R | ·–· |
| M | –– | Q | ––·– |

*Huffman coding* is guaranteed to find an efficient code for a given language *if* you know the probabilities of symbols in the language.

# Lessons

- We use "language" for any scheme to generate a series of symbols.
- For any language, we want an efficient binary encoding that is lossless, uniquely decodable and streaming.
- Huffman encoding will provide such an encoding assuming we know the probabilities of symbols in the language.

**Next lecture:** Entropy