Cryptography and light bulbs: simplified attempt to explain "computer"

David Morgan

Did you ever wonder why conversations about computers always end up mentioning something about "binary numbers?" Also, "ones and zeros?"

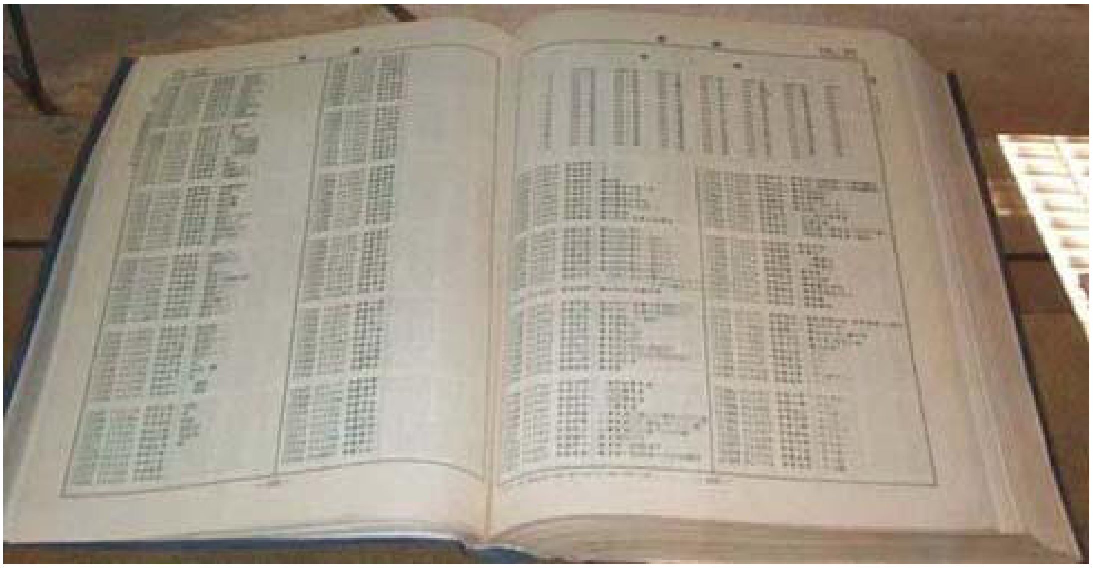
Symbolic representation

First of all, computers are instruments for symbolically representing things. Things in the world. Things in the imagination. Just things. A thing like, for example, a tree. There is that thing with branches and bark and leaves, then there's the word. "tree," I mean. We use that word, that utterance, to represent the thing. The word is no tree; it's just a 4-letter word. Just a sound. You can't climb the word's branches. But between people, at least those that use the set of words called "English," you can bring to somebody's mind the understanding of a tree without need to actually have one nearby to show to him. You substitute the word for the object. The word is a symbol.

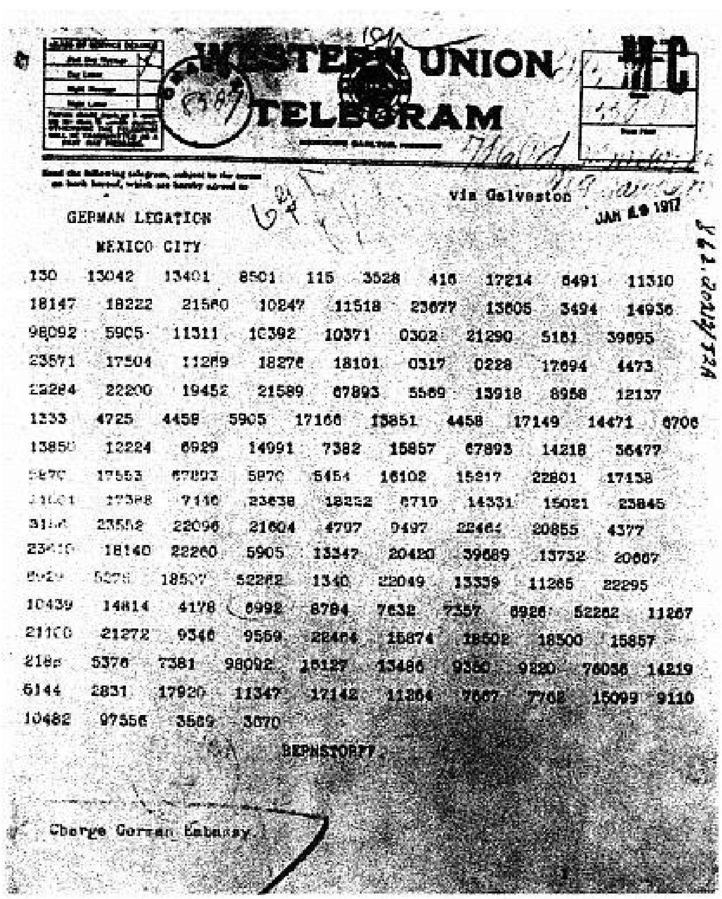
Since the symbol is not the object, you could choose a different symbol for the object if you (and your people) wanted. "arbol" will do. Or "ki," "arbre," or "mti." Some people actually did choose those and do use them (Spanish people, Japanese, French, Swahili).

One set of symbols for trees and things might be numbers. We could number all the things (nouns), and all the activities (verbs), and all the characteristics (adjectives) and compose our communication with strings of numbers instead of strings of words. We could abandon the English, Spanish, Japanese symbol sets, use none of those, and replace their old symbols with our new ones, the numbers. Maybe we would assign 17812 to that trunk-ful, root-ful, leaf-ful object we're discussing here. It would be "code" for a tree. We could choose numbers for everything we want to represent and keep track of them in a lookup book, a dictionary of all those code numbers.

Don't laugh. This is not far fetched. It was the basis of cryptography at one time. Here's picture of a so-called code book used by the Japanese in World War II:



They put a different number to each Japanese word and sent messages among themselves in numbers. The Germans did the same thing in World War I. Here's the telegram sent from Germany in 1917 to their ambassador in Mexico telling him to invite the Mexicans to please declare war on the United States:



WESTERN UNION
TELEGRAM

via Galveston

JAN 19 1917

GERMAN LEGATION

MEXICO CITY

130 13042 13401 8501 115 3528 416 17214 6491 11310
18147 18222 21560 10247 11518 23677 13605 3494 14936
98092 5905 11311 10392 10371 0302 21290 5161 39695
23571 17504 11269 18276 18101 0317 0228 17694 4473
22284 22200 19452 21589 67893 5569 13918 8958 12137
1333 4725 4458 5905 17166 13851 4458 17149 14471 6706
13850 12224 6929 14991 7382 15857 67893 14218 36477
5870 17553 67893 5870 5454 16102 15217 22801 17138
[illegible] 17388 7440 23638 18222 6719 14331 15021 23845
[illegible] 23552 22096 21604 4797 9497 22464 20855 4377
[illegible] 18140 22260 5905 13347 20420 39689 13732 20667
[illegible] 5275 18507 52262 1340 22049 13339 11265 22295
10439 14814 4178 6992 8784 7632 7357 6926 52262 11267
21100 21272 9346 9559 22464 15874 18502 18500 15857
[illegible] 5376 7381 98092 16127 13486 9350 9220 76036 14219
5144 2831 17920 11347 17142 11264 7667 7762 15099 9110
10482 97556 3569 3670

BERNSTORFF.

Charge German Embassy.

The numbers are alternative symbols for German words. The Germans made up this system because they wanted secrecy and it's better than speaking German since nobody knew their number system and too many people do know their word system (i.e., German). The people in Berlin and the German embassy in Mexico City both had a copy of the relevant code book but nobody else did. They could understand each other but nobody else could This telegram is called the Zimmerman note. It passed through communication channels in Britain and the U.S. So along the way it was intercepted and figured out. As it turned out the Mexicans never did declare war on the U.S. But not long afterward the U.S. did declare it against Germany.

I want to make up a symbol code for representing things. I have to start by deciding what things I want to represent, and then what code symbols to represent each of them with.

<u>Two aspects of symbolic representation</u>

What I want to represent - I'll start small scale. I only want to represent the four seasons.

What I want to represent it with - I don't want to use words (no "summer," no "primavera," no "hiver"), nor even numbers. I want to use pairs of coins. I'll use wooden nickels. The thing about a pair of wooden nickels that interests me is whether they be head or tails. Each nickel has to be one or the other. So the pair can be one of four ways. Here are the four ways:

Each pair is a distinct symbol I could use. So I'll use them for the four seasons, assigning one of the seasons to each of the pairs (or one of the pairs to each season if you prefer), perhaps this way:

= *Winter*

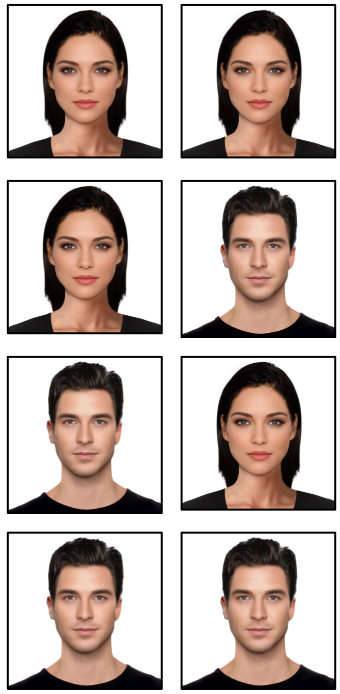
= *Spring*

= *Summer*

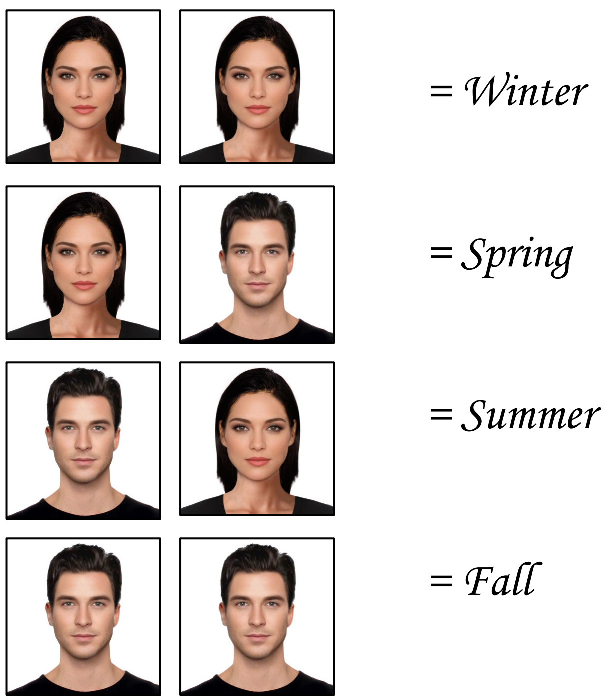
= *Fall*

Changing what I represent - I could instead use my coin pairs to represent some other four things. Maybe earth, air, fire, and water. Maybe north, south, east, west. Maybe spades, hearts, diamonds, clubs. Maybe twenty, forty, sixty, eighty. Maybe zero, one, two, three. Or something. *Any four things*. Same symbols, different meanings.

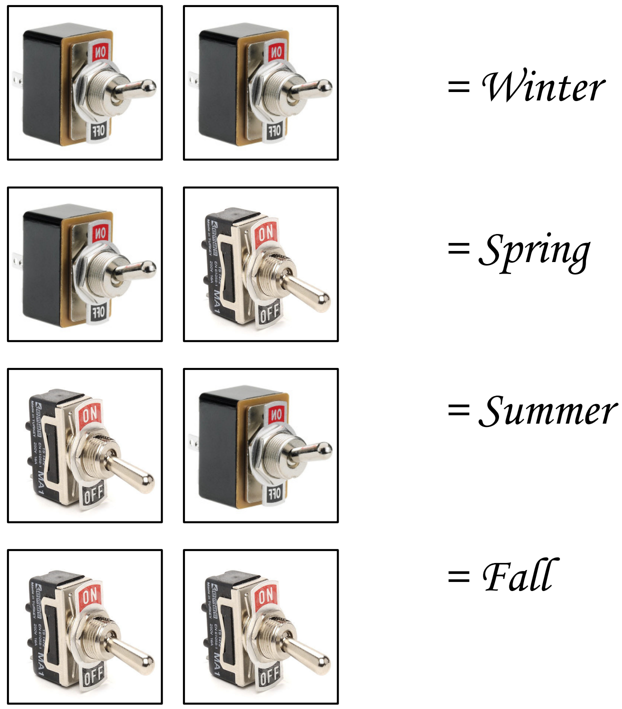
Changing what I represent it with - or if I remain specifically interested in the seasons, I could represent them with some other symbol pairs. Say, people pairs instead of coin pairs like this:



And assign pairs to seasons:

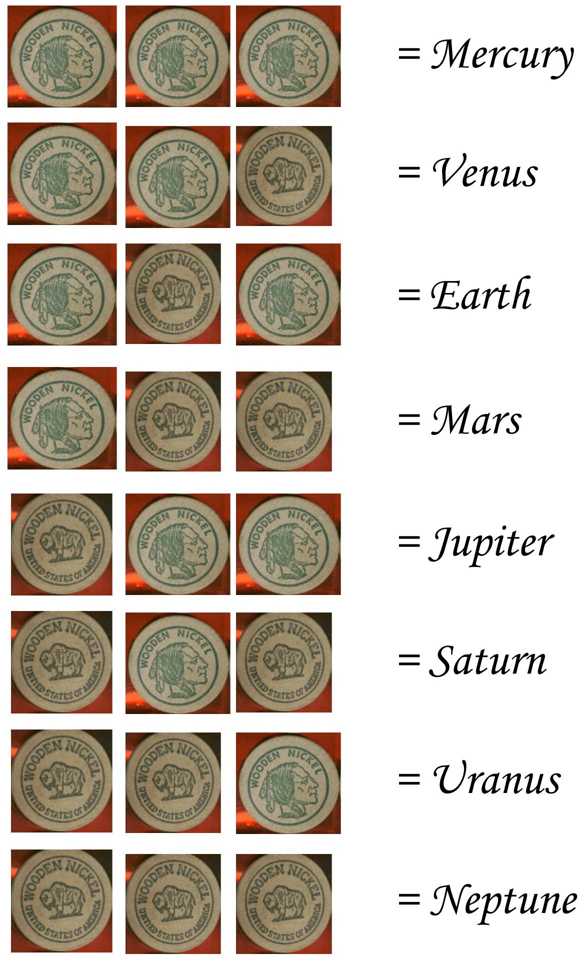


Same meanings, different symbols. Or how about a third symbol set for these seasons, using switch pairs instead of coin or people pairs.



I've chosen my symbol elements carefully. They are special, because they have either of two possible dispositions or configurations: head or tail, boy or girl, on or off. They are "binary" and each coin, person, or switch is an element of the pair. Each is a "binary element," which computer people abbreviate and call a "bit."

If I need to represent more things, say eight of them, maybe the planets (since Pluto got fired, there are only eight). I can do that if I use more elements-- add a third coin, person, or switch. Now I have triples instead of pairs. The different configurations of the triples become eight in number (ranging from, e.g., all three heads to all three tails and the other six ways in between).



Computers

Now let's talk about computers, because this is what they do. We have the same two questions: what do they represent, and what do they represent it with. From experience, it seems they can represent just about anything.

What do they represent it with? Suppose you were going to make a computer. You need it to contain something that can represent stuff. Would you choose to make it out of wooden nickels, people, or switches? They are all three equally good for representing stuff. So maybe it would be just as well to build your machine out of a lot of people instead of a lot of switches? That doesn't feel right. Computers choose switches for this job. That's because first of all, they are smaller than people and nickels. More importantly, they have a side effect that is electrical and that people and wooden nickels don't. A coin flipped head or tail, or a switch flipped up or down, that's a positional thing. But a switch flipped up or down has a side effect of physics that a wooden nickel utterly lacks. Namely, concomitant to its position it makes or breaks a circuit. It blocks or passes electricity. It conducts or insulates. Wooden nickels and people don't have that.

From the above, you get the idea that if you decide on a system you can pretty much represent anything you want using symbols and, for the symbols, you can use combinations of any elements that are binary (i.e., always in either of two ways). Computers use electric switches to do it. (Computer people more often talk about "transistors" than "switches." But a transistor *is* a switch that has a special way of flipping on or off. But it's a switch.) And they use combinations that go beyond pairs or triples, usually gangs of 8- or 16- or 32- or 64-at-a-time instead of 2- or 3-. Computer memory is a massive, massive collection of switches. Each independently can be on or off. You can check whether on or off by trying to pass electricity through and see what happens. That's called reading. You can also go flip them individually on or off in some collective pattern you choose. That's called writing. Now you have your system for representing stuff.

What stuff? Well, whatever you want. Kind of like homonyms, the exact same spoken symbol double-employed for two different purposes. When somebody puts the sound "say-ell" into your ear it might mean canvas sheet on a mast or economic transaction by a merchant (sail or sale), for example. I have also a real-world example of a "computer homonym." Namely, if a computer assembled 8 switches and put them in this pattern:

off - off - on - off - on - on - off - on

I'm aware that this is sometimes used to represent the number 45 and sometimes used to represent the hyphen character. The system where is it used for 45 is called the binary number system; the system where it is used to represent hyphen is called the ASCII system (American Standard Code for Information Interchange). If as above you're free to use symbols for either the seasons or the points on the compass, your choice totally, then you can use symbols for either number quantities (45) or for keyboard characters (hyphen). The massive collection of switches we're talking about here, in the computer is called its "memory."

There's one other thing to say about setting switches. They can be hooked up in a larger circuit, as elements in the circuit. The ones we're talking about so far are not really part of anything beyond themselves. The only thing they do that interests us is be on or be off. The switches in the picture above are just sitting there in their respective positions, each either on or off. They aren't hooked up to anything. When you move the switch from one position to the other nothing else happens. No ramification. Here's a switch where something else *does* happen:

The bulb goes bright or dark in direct sympathy with the switch going on or off. The difference between this switch and an isolated one is that we surrounded this one with circuitry (wires, bulb) so it became part of a larger whole.

The computer too can surround switches with circuitry to give them this kind of active effect. They embed something like the above diagram, but much more elaborate and complicated, into a product called a CPU (central processing unit) or microprocessor. It looks like a black plastic square. Inside it are both switches and surrounding circuitry, miniaturized. The switches are ganged in groups (of 16, or 32, or 64) called "registers." The surrounding circuitry is called the ALU (arithmetic and logic unit). When you turn switches on and off, different thing happen in the ALU circuitry. The engineers have designed it so that whatever particular combination of switch settings you make, different side-effect actions may take place.

In particular, there are two switch sets (general-purpose registers) in the CPU for holding numbers you want to manipulate (remember that off-off-on-off-on-on-off-on can be understood to express 45). And there's a switch set that's wired to do different things to your two numbers. That set is called the "instruction register." And when you flip the instruction register's switches in one pattern, the electrical result will be to set some result register's switches to express the sum of the two numbers in the general-purpose registers. But some other pattern of flips in the instruction register produces their difference. And a third instruction register pattern may produce their product, and a fourth their quotient. The CPU is wired to be able to do whichever of those. The instruction pattern that generates sums is called the "add" instruction, the one that produces differences is called the "subtract" instruction, and the other two of course, the "multiply" and "divide" instructions. There are more. Altogether the set of available instructions, the CPU's repertoire, is called its "instruction set."

Lie down
Roll over
Shake hands
Sit
Stay

That's the instruction set for your dog. Your dog is designed (trained) to have that instruction set. My dog has a different one from yours. My dog is smarter than yours. He also knows "fetch." Like different dog owners, different CPU manufacturers have somewhat different instruction sets. Intel is the best-known manufacturer of CPUs for PCs. ARM is the best-known manufacturer of CPUs for cell phone. For both, you can look up their instruction sets. Though they may overlap, they differ (incompatible). But all of them at least know how to add, subtract, multiply, and divide. That is, how to compute. That's why these things are called "computers."

I've likened the CPU to a simple circuit that controls a light bulb. A better example would be more elaborate. Think of the dashboard on your car. Switches are there. What do they do? Different things. More elaborate still, think of the cockpit of a commercial jet.

Switches, surrounded with circuits that make them active, that make them potent. A computer, a CPU, is just yet another outcome of human engineering effort. More miniature, less recognizable, but just the same.

Binary numbers

I said something about those at the beginning but didn't explain. You'll learn more about those later. Here's a hint:

0	0	0	= *zero*
0	0	1	= *one*
0	1	0	= *two*
0	1	1	= *three*
1	0	0	= *four*
1	0	1	= *five*
1	1	0	= *six*
1	1	1	= *seven*

Remember the lesson here goes something like, "You can represent anything with combinations of any binary element." Binary elements are things that are either of two ways. Our numerals aren't binary elements because a numeral can be any of ten ways (i.e., there are ten different ones). But if we confine the numerals to only two, let's pick zero (the circle) and one (the vertical line), that makes them binary. So why not?