Security-101: Notes that would make my mum proud

Josh Felmeden

January
2019

Contents

1 Encoding

The first thing to note is that encoding is NOT encryption. They're actually very different. Encoding is used for safe transfer of data. Say you want to send a message by form of an e-mail, or telegraph etc. E-mails are all encoded by the e-mail program, so you don't need to worry about it. Morse code is an example of encoding, but all of the characters have different lengths.

In modern terms, a **bit** can be either 0 or 1 (obviously, please tell me you knew that). Baudot code is kinda similar to Morse code, except that each character has a fixed length of 5 holes, that can be either covered or not. This is like bits, because if the hole is covered, we can represent this as a 1, and the uncovered holes are represented by a 0.

The formal definitions are:

- **Encoding** is a publicly known way to transform a message so that you can send it over a particular channel. There will be no secrecy from this so anyone can decode it.
- **Encryption** is transforming a message so that only the intended recipients can decrypt it. It requires a key.

A way to encode is a code, while the way to encrypt is a cipher.

1.1 ASCII

ASCII was introduced in the US in 1960-68. It stands for the American standard code for information interchange and has 7 bits per character. There are no support for accents, umlauts etc, but that's the yanks for you.

There is a code chart that has four bits in the column and 3 bits in the rows, so to access an ASCII value, you use the three bits from the row, followed by the four from the column.

ASCII was replaced by Unicode later on which is a single standard that is the be all and end all. It covers most known languages and even languages like Elvish for Christ sake.

You can't use Unicode to write anything directly because it's a standard and not an encoding. There are a number of 'flavours' of Unicode:

- UCS-2: each character is exactly 2 bytes long
- UCS-4: each character is exactly 4 bytes long
- UTF-8/16: This is variable length encoding
- UTF-7: "e-mail safe" format (only uses bits per byte)

A file is in **text format** if it contains text in some form of encoding. For example, the format that I'm writing in at the moment is UTF-8. A file is in binary format if it can contain arbitrary bytes. Some of these may not print right if you open it.

Binary files can be really hard to display/e-mail. We can encode these again before we email them. This is crucial for cryptographic data such as keys, ciphertexts and signatures which are binary data. The encoding is nothing to do with secrecy or security – it's just so that you can type, send, and work with these objects.

1.2 Binary and Hex

There are different bases that you can work with: hex, binary, and decimal. You already know this though. Hex encoding is really common because each byte can be encoded into exactly two characters. Unfortunately, the encoded data is twice as long as the original. An example of this is colours. Since computers usually encode colours as 3 bytes, each representing the red, green, and blue components, it makes sense to make each of these using 6 hexadecimal characters.

1.3 Base64

We're not talking about base 64 encryption, because it's not encryption. It's encoding. Base64 splits data into 3-byte blocks and then encodes each block as 4 characters. This means that the encoded data is only a third longer than the original, rather than twice the length as with hex encoding. If we take an example of the string "hi!":

$$h = 01111000$$
$$i = 01101001$$
$$! = 00100001$$

It then takes this and splits it up into 3:

$$011010|000110|100100|100001$$
$$011010000110100100100001$$

After this, it converts these into letters using a lookup table that I'm not prepared to write out here. Basically, from this sequence of bits, we end up with "hi!" = "aGkh". Dope, right? It turns any 3-byte sequence into 4 printable characters, therefore turning binary data into a text format.

2 Cryptography

2.1 Block ciphers

A block cipher such as AES works by having a message, a key, and an encryption function. The message can be of any length, and the ciphertext should be about the same size as the message. It's a bad idea to have something that produces a pattern, because this means that the code can be decrypted very easily. Using a block cipher that returns the same message when used for the same message is called ECB, which stands for electronic code book.

The subtleties of what makes a good encryption function is beyond this unit, but the point is that we can prove that if the block cipher kinda looks random on single blocks, then a good mode of encryption looks random as well. Good modes include another 'random' block at the start, known as the initialisation vector, so that you don't get the same thing if you encrypt the same message twice. Did you get that? If you encrypt the same thing twice, you don't get the same message!

An example of this is CBC, which is cipher block chaining. CBC mode is cool because it has a chain of ciphers (kind of like DES). It's really important that **patterns** and **repetition** are be hidden. Also, that you don't learn anything about the contents of a message, other than it's length.

2.2 MACs

In a passive attack, the adversary only observes the data. It's not interfered with at all. If the attack is *active* then data can be inserted, deleted, or meddled with. Even the delivery destination of the data could be compromised. A message authentication code, or MAC, is an algorithm that generates a *tag* that is based on the message and a shared secret key. The tag is sent with the message, and the recipient confirms this by running the same algorithm with the same key and comparing the value. If it is the same, then the recipient is assured that the message is the same as the one originally sent. If there is a mismatch in any way then this means that the message that was received has changed in some way to the one that was sent in the first place. A MAC only does what it's told to do, and since they're not required to preserve confidentiality, sending a MAC of the plaintext message in the clear actually compromises the confidentiality provided by the encryption protocol. It makes much more sense to encrypt the message and then MAC it, which ensures the integrity of the ciphertext along with the confidentiality of the plaintext, or the other option is to MAC, and then encrypt. Most cryptographers recommend encrypt and then MAC.

Checksums are not MACs. They're just used for detecting errors in stored or transmitted data.

2.3 Key management

Symmetric cryptography provides the facility to exchange encrypted documents between people who share the same key. This raises the question, how do they even get those keys to start with? For n people to be able to communicate in pairs requires n^2 keys. They need to be agreed over secure channels to prevent eavesdroppers from obtaining them and the just going ham on the messages. They also need to be updated sometimes. But, what's the next step?

3 Public-Key Cryptography

The operation of public key cryptography is kind of like a padlock. You can click it, and it's shut. But, when it's shut you need a key. This is really useful for being able to communicate with a lot of people, since they can all send you encrypted messages, but no one can read them except from you.

It all relies on two things: A public key, and a secret, private key. The public key must be made available to the public y making it available in some directory. Then, anyone who wants to send a

message to you encrypts it under the public key, but it can only be decrypted by the secret key.

For this to even work, you need both of the keys (which are long and random numbers) to be mathematically linked in some way. Normally, the public key is made by some one-way function from the private key, and is easily computed by the owner of the private key, while the inverse transformation is practically impossible.

A public key encryption has three algorithms:

1. **K**: the key generation algorithm. This returns the private and public keypair.
2. **E**: the encryption algorithm, which encrypts the message with the public key, and returns the ciphertext
3. **D**: the decryption algorithm that takes the ciphertext and the secret key as the input, and returns the original message in plaintext.

Public key encryption by itself offers literally no authenticity. Anyone at all can send you encrypted messages.

We're going to look at signing messages now. There can be such a thing as a Public-Key Infrastructure (or PKI), which is a framework for generating, managing, and distributing keypairs and digital certificates (proof of key ownership). This means that if someone trusts a person, then they can add their signature to the key.

3.1 Digital signatures

A signature is equivalent to a handwritten signature. It binds someone's identity to a piece of information. With a digital signature verification algorithm, one verifies that a digital signature is authentic. A digital signature scheme consists of a key generation algorithm, a signature creation algorithm, and an associated verification algorithm. There are different kinds of signature schemes. Digital signatures with appendix require the original message as an input to the verification algorithm. Digital signatures with message recovery do not require the original message as an input to the verification algorithm. In this case, the original message is just recovered from the message itself.

The digital signature standard (or DSS) is a standardised signature scheme with appendix. The DSS describes an algorithm called DSA, which is based on the mathematically difficult problem of computing discrete logarithms in certain groups it also includes some variant of the same algorithm on elliptic curves, the ECDSA. In order to sign a message, the owner first must compute the hash value of the message. This hash value and the private key are then subjected to the signing algorithm. The output of that algorithm is the signature of the input message. Anyone who wishes to verify the signature can simple take the message, the public key and the signature and complete the verification algorithm with these inputs.

4 Malware

Malware is a word that means malicious software. It refers to files that run on a machine that have intentions of the not-so-nice kind. They can do things like:

- Collect information about the user.
- Encrypt data on the machine with the aim to get the user to pay for decryption (also known as ransomware)
- Linking the user up to a paid service (that is useless)
- Installing a piece of software that will be activated in the future.

The problems are getting worse, and they're already pretty bad. Before the internet, systems could get corrupted by malicious floppy disk freebies, but because everyone uses the internet, it's way easier to spread vicious programs.

There are a lot of types of malware:

- Some types, such as viruses and worms, spread by replicating.
- Some are concealed, such as trojans and logic bombs.
- Types such as adware, spyware, pornware and ransomware are financially motivated.
- Some, such as rootkits, are designed to gain control of the machine, and then like them up to a botnet.
- Viruses are pieces of code that rely on some other piece of code to propagate. They have three components:
  - An infection mechanism
  - A trigger, which is some condition such as time, under which it will begin to execute
  - A payload, which is the malicious code
- A worm is a *self-replicating* virus. This means that it makes copies of itself, without relying on a host to spread. It achieves this by exploiting weaknesses of networks, networked services and operating systems, like:
  - Using a (command line) email client to email itself to other users.
  - Using (unprotected) user credentials to log into other systems remotely, to copy itself over and execute
- Logic bombs are pieces of code that are inserted into a system. It functions as a trigger when specified conditions are met, such as the commencement of a process to delete particular files from an employer's system if their employment is terminated.

- A trojan is a piece of code that has secretly been inserted with harmful intent. They're often activated by logic bombs. Most forms of adware etc. are trojans.

Malware is everywhere. Main attack vectors are USB devices sometimes, but more often from adware and downloads and emails.

It's pretty hard to detect malware