Secure Software Design

Andey Robins

Spring 23 - Week 5

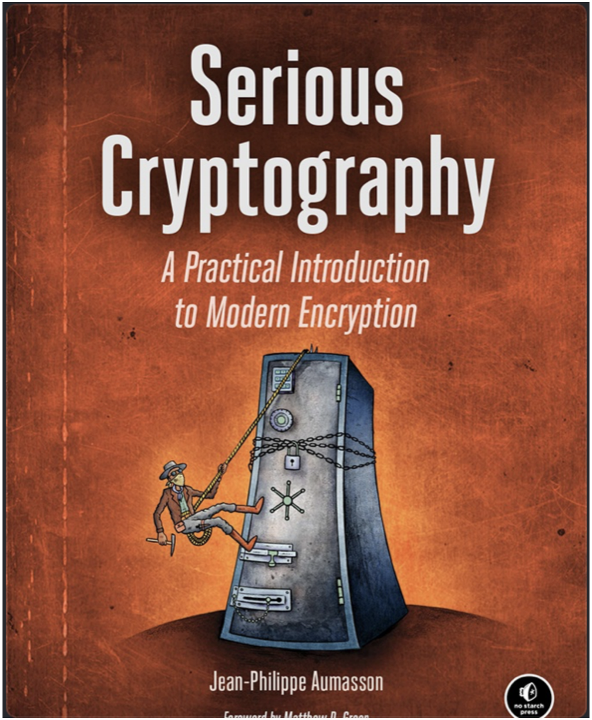
Cryptography

Cryptography

Cryptography is a set of mathematical tools we use to ensure the security of information in a hostile environment.

Outline

- Cryptographic Primitives
- Hashing
- Symmetric Encryption
- Asymmetric Encryption
- KDAs
- Signing
- CAs and Certificates
- Applied Crypto



Serious Cryptography

A Practical Introduction to Modern Encryption

Jean-Philippe Aumasson

Foreword by Matthew D. Green

Crypto Primitives

Crypto Primitives

- Hashes
- Entropy
- SRNGs
- Keys
- Ciphers

Hashes

Hashes are built upon the idea of one-way functions. Something easy to compute in one direction, but very difficult to reconstruct with just the hash.

$$\forall x, y; (\exists f; f(x) = y \iff \nexists f'; f'(y) = x)$$

In the above formula, the function `f` is the hashing function.

Entropy

Entropy is a measure of randomness. Formally, if your probability distribution is p_1, p_2, . . . , p_n, then entryopy is:

$$-p_1 * log(p_1) - p_2 * log(p_2) - ... - p_n * log(p_n)$$

Therefore, random 128 bit keys created over a uniform distribution have 128 bits of entropy:

$$2^{128} * (-2^{-128} * log(2^{-128})) = -log(2^{-128}) = 128 \text{ bits}$$

Key Takeaway: If you use a uniform distribution, you get as many bits as you expect.

RNGs

Cryptographically Secure RNGs vs Pseudo RNGs

I got a question teaching 1010 last semester: “Can you pick a random index into pi and use that to generate random numbers?”

RNGs

Cryptographically Secure RNGs vs Pseudo RNGs

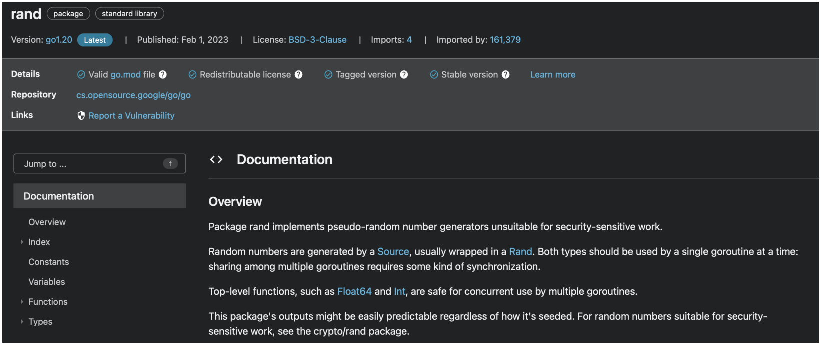
I got a question teaching 1010 last semester: "Can you pick a random index into pi and use that to generate random numbers?"

It depends on the application. If we need cryptographically secure numbers, no because the next number is not independent from the previous since we could look it up and predict future numbers perfectly.

RNGs

Two parts:

1. A source of entropy
2. A crytpographic algorithm to produce random bits given a source of entropy



Overview

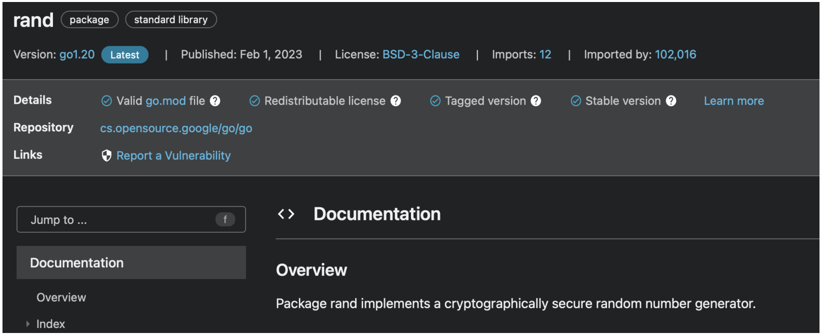
Package rand implements pseudo-random number generators unsuitable for security-sensitive work.

Random numbers are generated by a Source, usually wrapped in a Rand. Both types should be used by a single goroutine at a time: sharing among multiple goroutines requires some kind of synchronization.

Top-level functions, such as Float64 and Int, are safe for concurrent use by multiple goroutines.

This package's outputs might be easily predictable regardless of how it's seeded. For random numbers suitable for security-sensitive work, see the crypto/rand package.

Figure 2: math/rand package for golang


rand package standard library

Version: go1.20 Latest | Published: Feb 1, 2023 | License: BSD-3-Clause | Imports: 12 | Imported by: 102,016

Details ✓ Valid go.mod file ✓ Redistributable license ✓ Tagged version ✓ Stable version Learn more

Repository cs.opensource.google/go/go

Links Report a Vulnerability

Documentation

Overview

Package rand implements a cryptographically secure random number generator.

Figure 3: crypto/rand package for golang

Keys

Keys are an additional secret used in various crytpographic operations. The simplest form is seen in One Time Pad encryption, but it's probably better known in the sense of public and private keys.

One Time Pad

Given a Plaintext, P, and random key, K, we generate the ciphertext, C, with the following operation:

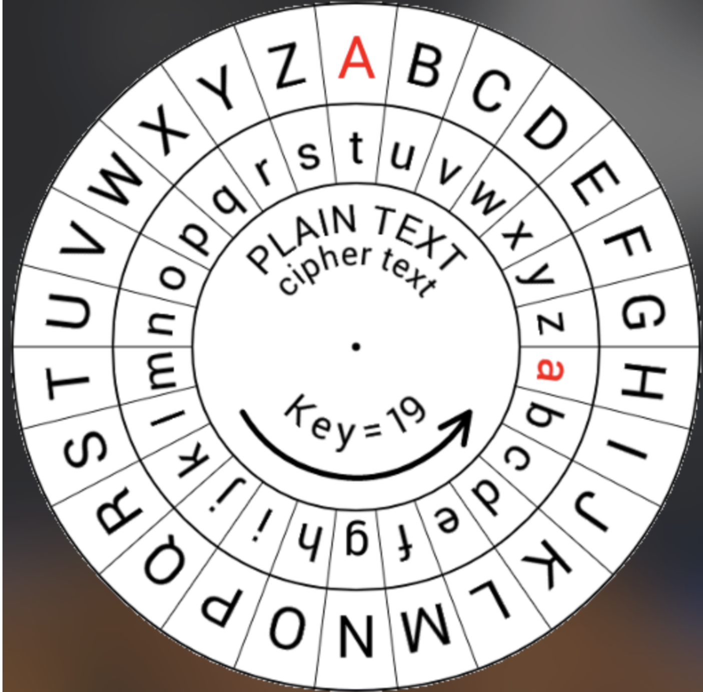
$$C = P \oplus K$$

Which decrypts due to the following rule:

$$C \oplus K = K \oplus K \oplus P = P$$

Ciphers

Ciphers, such as One Time Pad, are ways to *encrypt* information so that only those with a secret are able to read it. Some of the earliest are the Caesar cipher or Scytale cipher.



A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
t u v w x y z a b c d e f g h i j k l m n o p q r s
PLAIN TEXT
cipher text
Key = 19

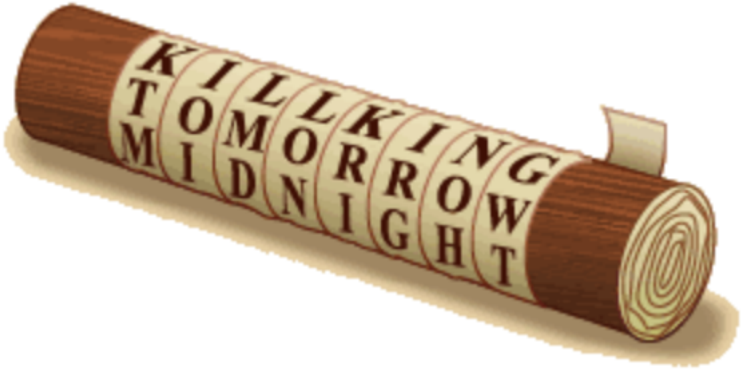


Figure 5: A scytale cipher

Kerckhoff's Principle

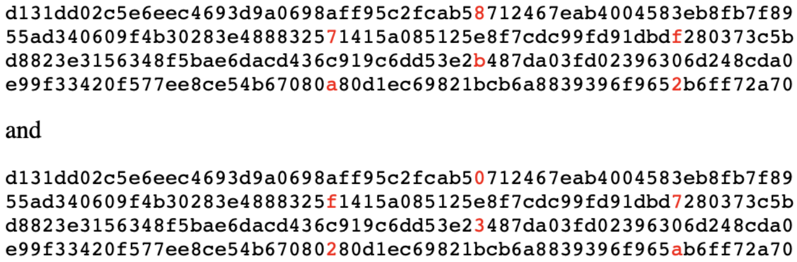
Kerckhoff's Principle states that the secrecy of a cryptographic message should rely on the secrecy of the key and not the secrecy of the cipher.

Hashing

Collisions

The viability of a hashing algorithm relies on it being reistant to collisions. Collisions are two inputs which hash to the same value.

Collision Attacks



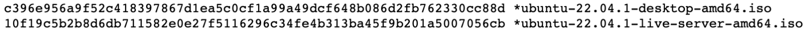
```
d131dd02c5e6eec4693d9a0698aff95c2fcab58712467eab4004583eb8fb7f89
55ad340609f4b30283e488832571415a085125e8f7cdc99fd91dbdf280373c5b
d8823e3156348f5bae6dacd436c919c6dd53e2b487da03fd02396306d248cda0
e99f33420f577ee8ce54b67080a80d1ec69821bcb6a8839396f9652b6ff72a70
```

and

```
d131dd02c5e6eec4693d9a0698aff95c2fcab50712467eab4004583eb8fb7f89
55ad340609f4b30283e4888325f1415a085125e8f7cdc99fd91dbd7280373c5b
d8823e3156348f5bae6dacd436c919c6dd53e23487da03fd02396306d248cda0
e99f33420f577ee8ce54b67080280d1ec69821bcb6a8839396f965ab6ff72a70
```

Figure 6: Collision in MD5 with digest 79054025255fb1a26e4bc422aef54eb4

Example: Checksums



```
c396e956a9f52c418397867d1ea5c0cf1a99a49dcf648b086d2fb762330cc88d *ubuntu-22.04.1-desktop-amd64.iso
10f19c5b2b8d6db711582e0e27f5116296c34fe4b313ba45f9b201a5007056cb *ubuntu-22.04.1-live-server-amd64.iso
```

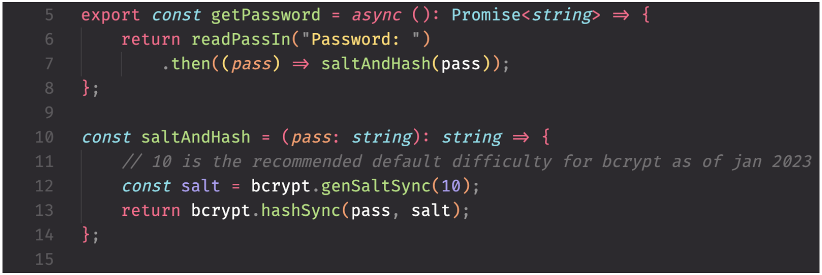
Figure 7: Ubuntu SHA256 digests for LTS version 22.04

Makes it easy to verify the authenticity of a file. When coupled with signatures, makes it very easy to verify the authenticity and ownership of very large files.

Password Hashes

Given our assumptions about hashes: they are irreversible and collision resistant, a very useful application becomes hashing passwords! Now, even if your password store is stolen, the user passwords aren't lost.

Example: Bcrypt



```
export const getPassword = async (): Promise<string> => {
    return readPassIn("Password: ")
        .then((pass) => saltAndHash(pass));
};

const saltAndHash = (pass: string): string => {
    // 10 is the recommended default difficulty for bcrypt as of jan 2023
    const salt = bcrypt.genSaltSync(10);
    return bcrypt.hashSync(pass, salt);
};
```

Figure 8: An example of using bcrypt

Salt and Pepper

Salt and pepper are both pieces of information a principal can add to a password to secure it.

Salt - Added by the “backend,” definitely the much more common, completely best practice

Pepper - Added by the “user,” useful for redundancy or in low trust environments (i.e. putting the name of the website after the password in your password keeper.)

Symmetric Encryption

Encryption Definitions

- **Plaintext** is the message to be encrypted, in other words, the original message
- **Ciphertext** is the encrypted message. This is illegible.

Symmetric Key Encryption

Both principles have the same key and perform the same operations to encrypt and decrypt the information. Due to the similarity and equivalent actions, we call this symmetric encryption.

One Time Pad

There is only one value for K, even though ideally there would be a way for us to

$$C = P \oplus K$$

Which decrypts due to the following rule:

$$C \oplus K = K \oplus K \oplus P = P$$

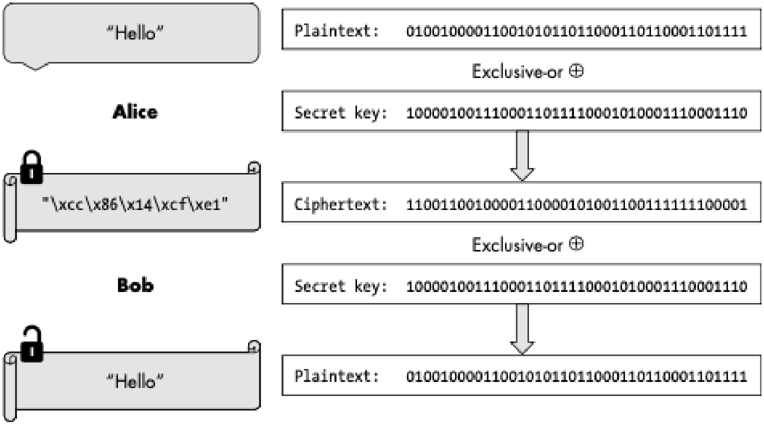

"Hello"
Alice
"\xcc\x86\x14\xcf\xe1"
Bob
"Hello"
Plaintext: 0100100001100101011011000110110001101111
Exclusive-or ⊕
Secret key: 1000010011100011011110001010001110001110
Ciphertext: 1100110010000110000101001100111111100001
Exclusive-or ⊕
Secret key: 1000010011100011011110001010001110001110
Plaintext: 0100100001100101011011000110110001101111

Figure 9: Concrete example of one time pad encryption

AES

AES is the default “work horse” of the encryption world. It’s what is referred to as a *block cipher*. A long message is broken up into pieces, or blocks, and these blocks are then encrypted.

The same key is used for encryption and decryption.

Problem: Replay Attacks

If I encrypt the text “We attack at dawn. Come in on their west flank,” what prevents this message from being repeated in the future?

Change our message to: “We attack at dawn on 6/6/44. Come in on their west flank.”

Now this message has some means to check whether it becomes invalidated with time, but you could still wait a century and re-use the message to potentially dangerous outcomes.

We need some value that will be guaranteed to be only used once. A *nonce*.

AES Modes

- Galois/Counter Mode (GCM)
  - Good for parallelization
- Counter with Cipher block chaining message authentication code Mode (CCM)
  - Accounts for both authentication and confidentiality
- Synthetic Initialization Vector mode (SIV)
  - Nonce-misuse resistant
- AES-GCM-SIV
  - Combines GCM mode with nonce resistance
- Electronic Code Book mode (ECB)
  - Susceptible to replay attacks
  - Largely just insecure
- Cipher Block Chaining mode (CBC)
  - Sequential encryption

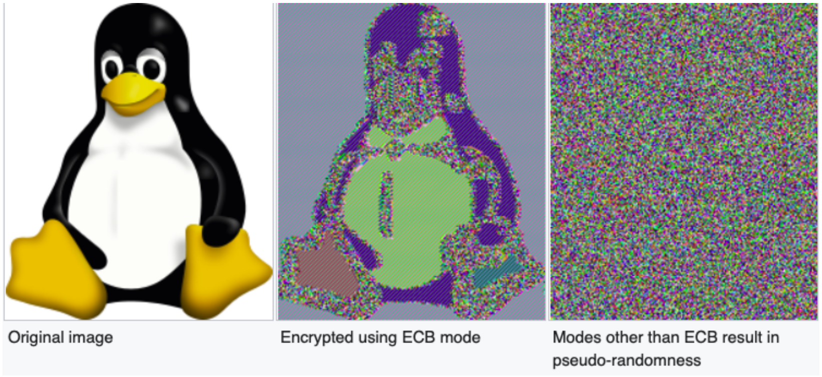


Figure 10: Visualization of ECB Mode

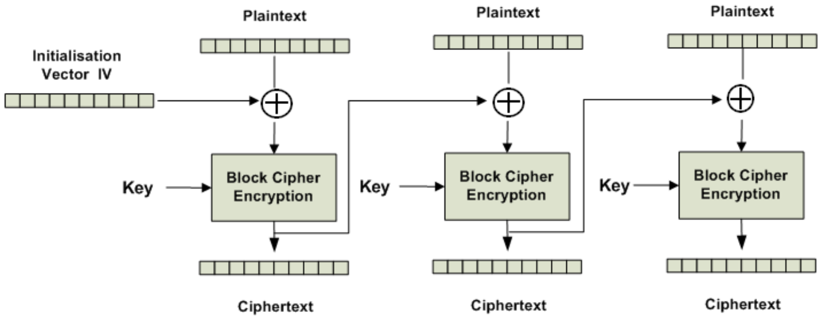


Figure 11: CBC Mode AES Encryption (invented 1976)

Asymmetric Encryption

Diffie-Helman Key Exchange

Revolutionary way to establish a shared secret in a public channel.

"DH works its magic by combining either public value with the other private value, such that the result is the same in both cases."

Requires the generation of "safe" primes.

`time openssl dhparam 2048` vs `time openssl genrsa 2048`

RSA

RSA was one of the first asymmetric, public-key cryptosystems. First published in 1977, emerged the year after the Diffie-Helman key exchange algorithm was published.

Since it relies on the hardness of factoring, it's not quantum-proof.

Elliptic Curve

Current best level of encryption, relies on the difficulty of the discrete-logarithm problem. Functions through math on prime curves in these logarithmic fields.

A different class of algorithms, which operates on a subset of mathemtical objects, many of which are quantum resistant.

NIST vs Curve25519

The NIST curve P-256 has an equation:

$$y^2 = x^3 - 3x + b$$

where b is a 256 bit number. (Similar curves exist for 192, 224, 384, aand 521 bit prime curves as well.)

Curve25519 has form:

$$y^2 = x^3 + 486662x^2 + x$$

Nobody disputes 25519, there are concerns about using P-256. There's a reason, Chrome, Apple systems, and OpenSSH use 25519.

Why Use Asymmetric or Symmetric Encryption?

- Asymmetric involves less trust of both parties
- Symmetric is faster and usually easier to use

Key Derivation Functions

KDFs

An algorithm which dervies one or more secret keys from a secret.

Two major uses: 1. Padding keys to required length 2. Converting form of secrets

A subset of hashing

KDFs are usually some form of a keyed hash. They therefore leverage many of the techniques we learn about with hashing.

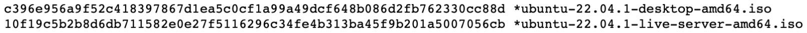
Applications for Encryption

A common application is to use a KDF after DH to create a symmetric secret.

- An alternative option in this scenario would of course also be using DH to encrypt a secret, sharing it, and then using it

Signing

Signing is authentication with encryption or with hashing. It's usually discussed on top of either a hash or encryption algorithm. Using the same principles (and in many cases similarly named algorithms) as our previous cryptographic systems.



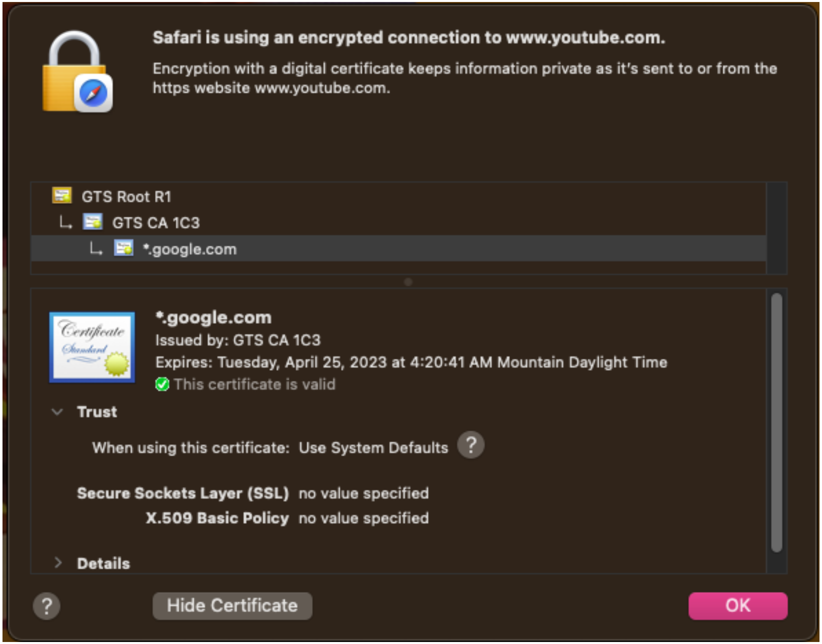
```
c396e956a9f52c418397867d1ea5c0cf1a99a49dcf648b086d2fb762330cc88d *ubuntu-22.04.1-desktop-amd64.iso
10f19c5b2b8d6db711582e0e27f5116296c34fe4b313ba45f9b201a5007056cb *ubuntu-22.04.1-live-server-amd64.iso
```

Figure 12: Ubuntu checksums and signatures

CAs and Certificates

Certificate Authorities are groups which are largely granted trust and we trust to sign the certificates and authorize/authenticate other principles in the net.

Certificates are the digital signatures they sign out. It's basically just like any form of ID, but backed by a cryptographic identity.



Safari is using an encrypted connection to www.youtube.com.

Encryption with a digital certificate keeps information private as it's sent to or from the https website www.youtube.com.

GTS Root R1
└ GTS CA 1C3
  └ *.google.com

***.google.com**
Issued by: GTS CA 1C3
Expires: Tuesday, April 25, 2023 at 4:20:41 AM Mountain Daylight Time
✓ This certificate is valid

Trust

When using this certificate: Use System Defaults

Secure Sockets Layer (SSL)	no value specified
X.509 Basic Policy	no value specified

Details

Figure 13: YouTube certificate as of 2/14/23

Questions?

Next Time

This wraps up our fundamentals of cybersecurity.

Next are Security Design Patterns