Practical Cryptography

For The Digital Citizen

Marshall Williams
GitHub @mwthink

What is Cryptography?

- “Cryptography” is a broad subject that involves the securing of information being sent amongst multiple parties
- Cryptography covers everything from simple number-ciphers (A=1, B=2…) to advance modern algorithms that are only possible with computers
- Computer-enhanced cryptography dates back to Alan Turing cracking the German *Enigma* machine during WW2
- The US government previously classified cryptography as “munitions” and attempted to regulate it via arms-export laws. During the late 90s, the Supreme Court ruled that published software code is protected by free speech (*Bernstein v US*), thus protecting the right to access strong encryption technologies.

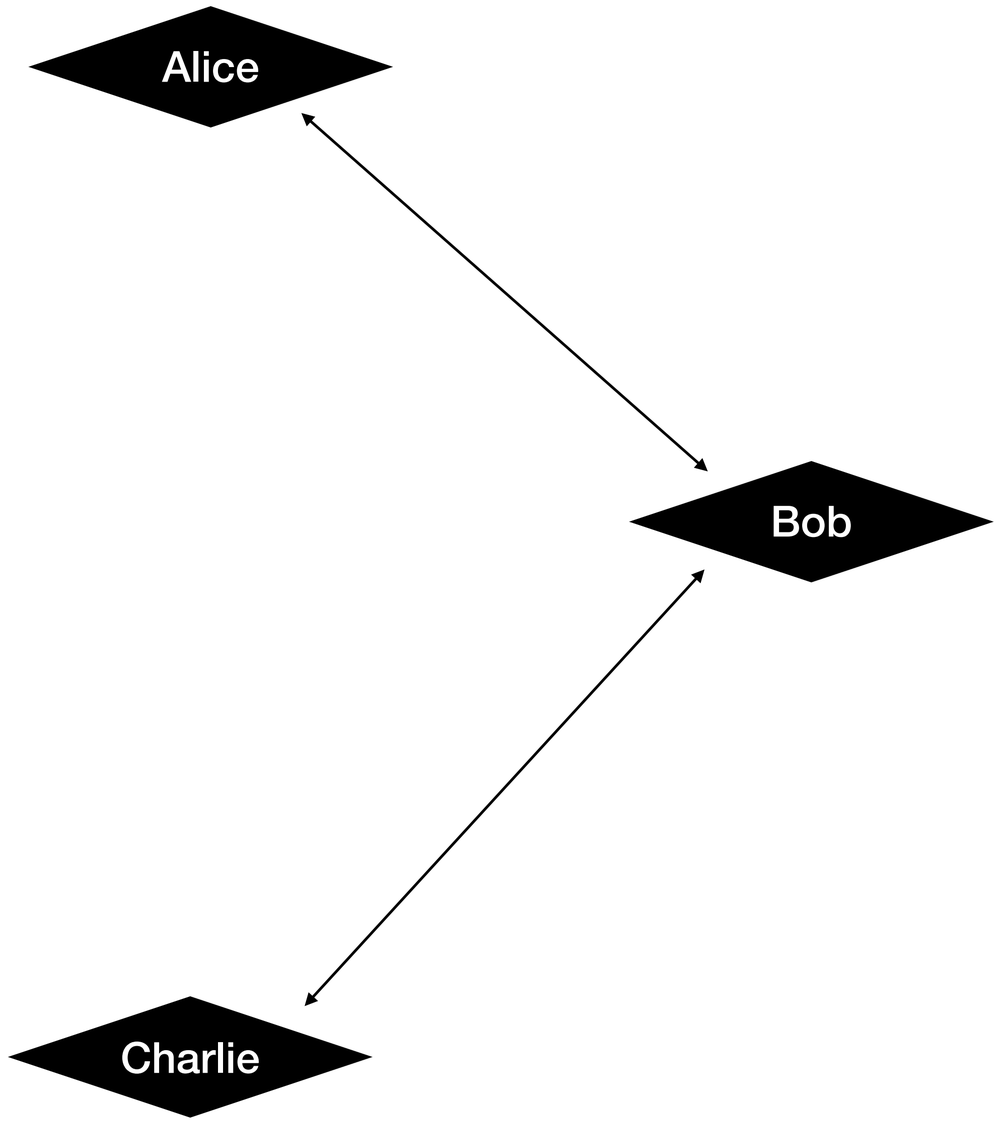
The Basic Model

Alice, Bob and Charlie

The Alice-Bob-Charlie (ABC) scenario is commonly used to model cryptographic problems.

The scenario is simple:

- Alice (A) and Charlie (C); 2 long-time friends, want to communicate privately
- All of their communication is sent via their messenger, Bob (B)
- A+C do not trust B and do not want B to be able to read their messages



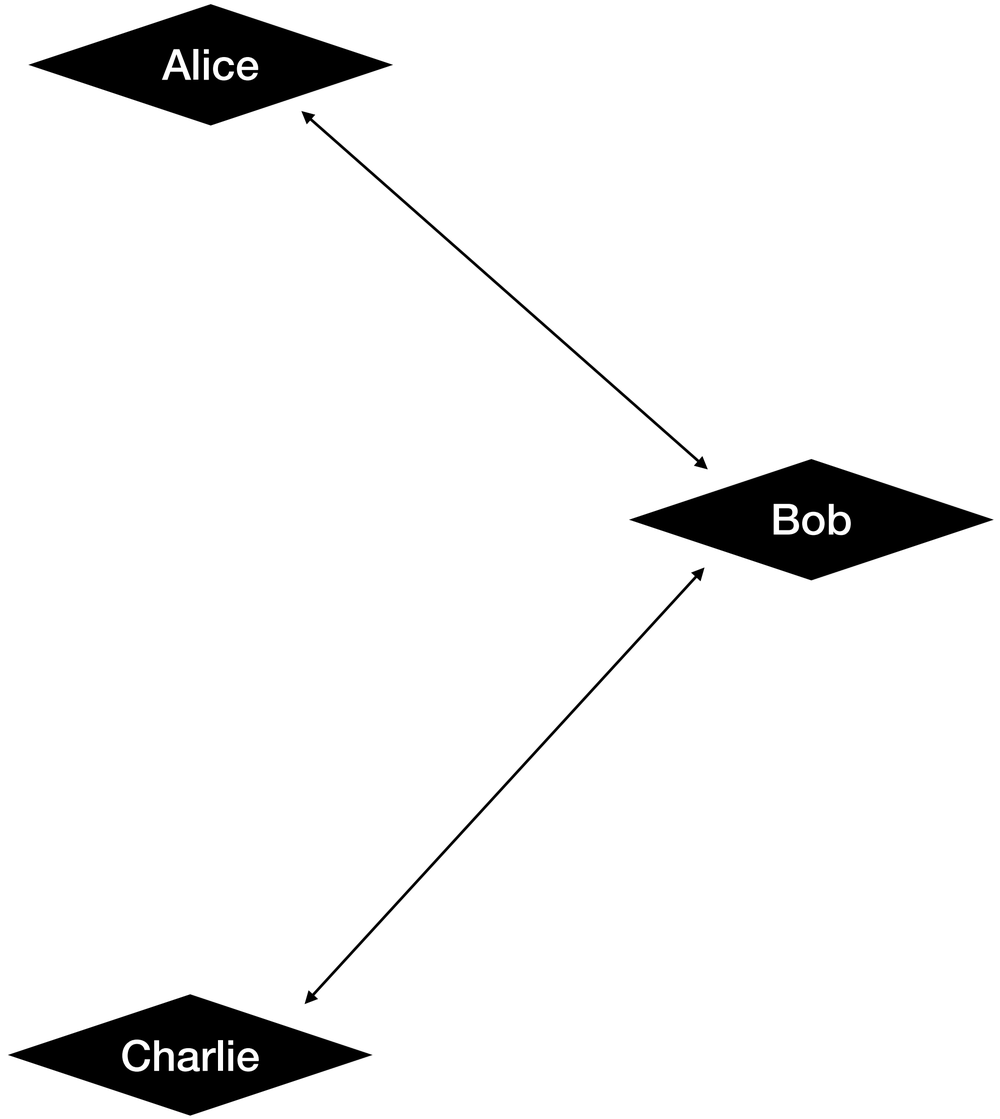
The Basic Model
Problems of ABC

In simple terms, we want to “secure” the message between A+C, via B.

We will break this vague concept of “security” into 2 definable goals:

1. ***Protecting the content*** of a message from unauthorized parties (“*Prevent Bob from reading the message”*)

2. ***Ensuring the integrity*** of the delivered message (“*Prevent Bob from delivering a fake message*”)

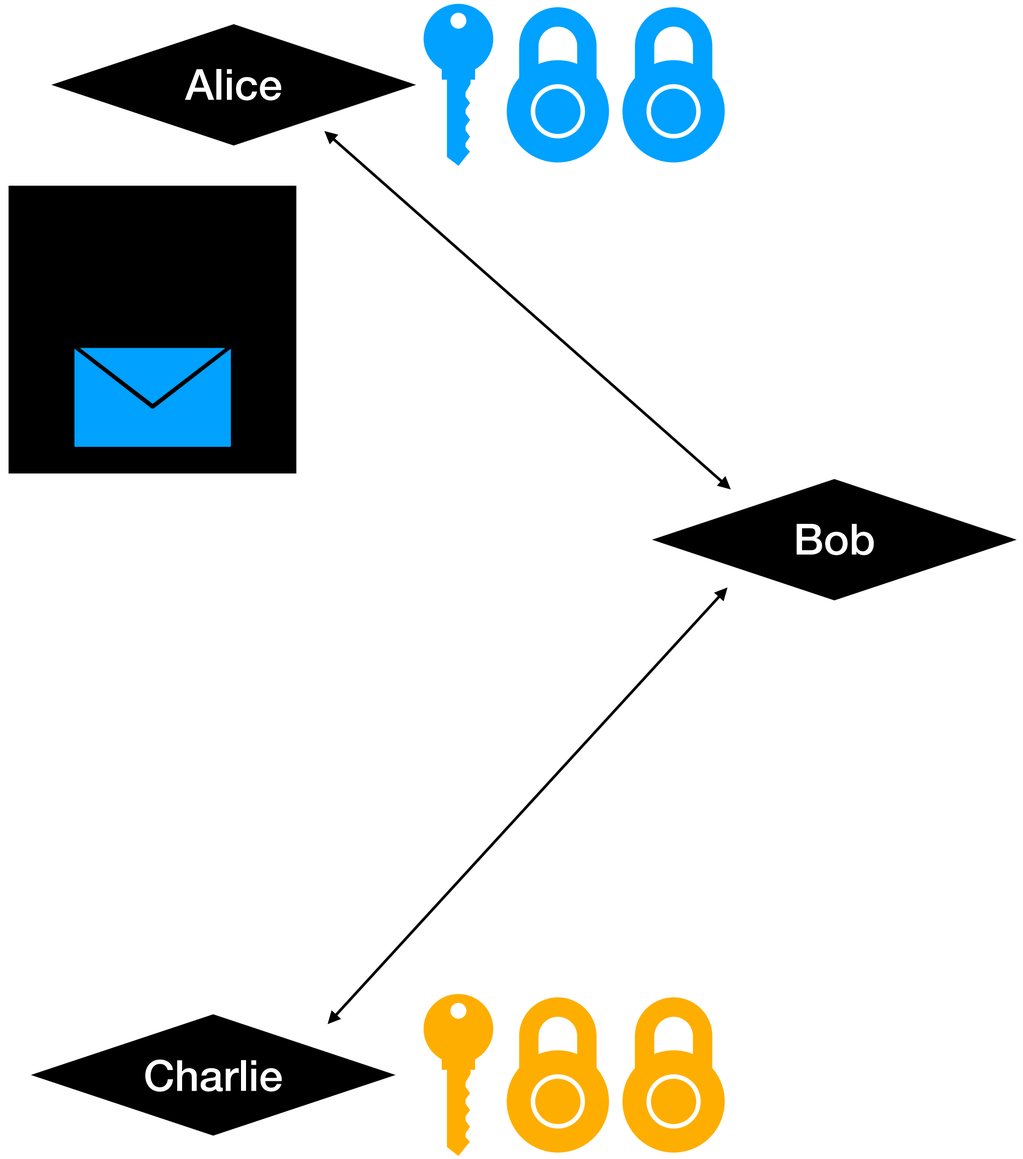


The Basic Model

Lockbox Solution

Let’s try to solve this in a low-tech way:

- A+C both purchase 2 identical padlocks and a matching key
- A+C meet in person and exchange one of their locks **unlocked** (each keeps their own key)
- After writing a message intended for the other, A/C will put the message into a box along with their *own personal* lock which is **unlocked**
- The box itself is then locked with *the recipient’s* lock
- A/C provide the locked box to B for delivery to the other person

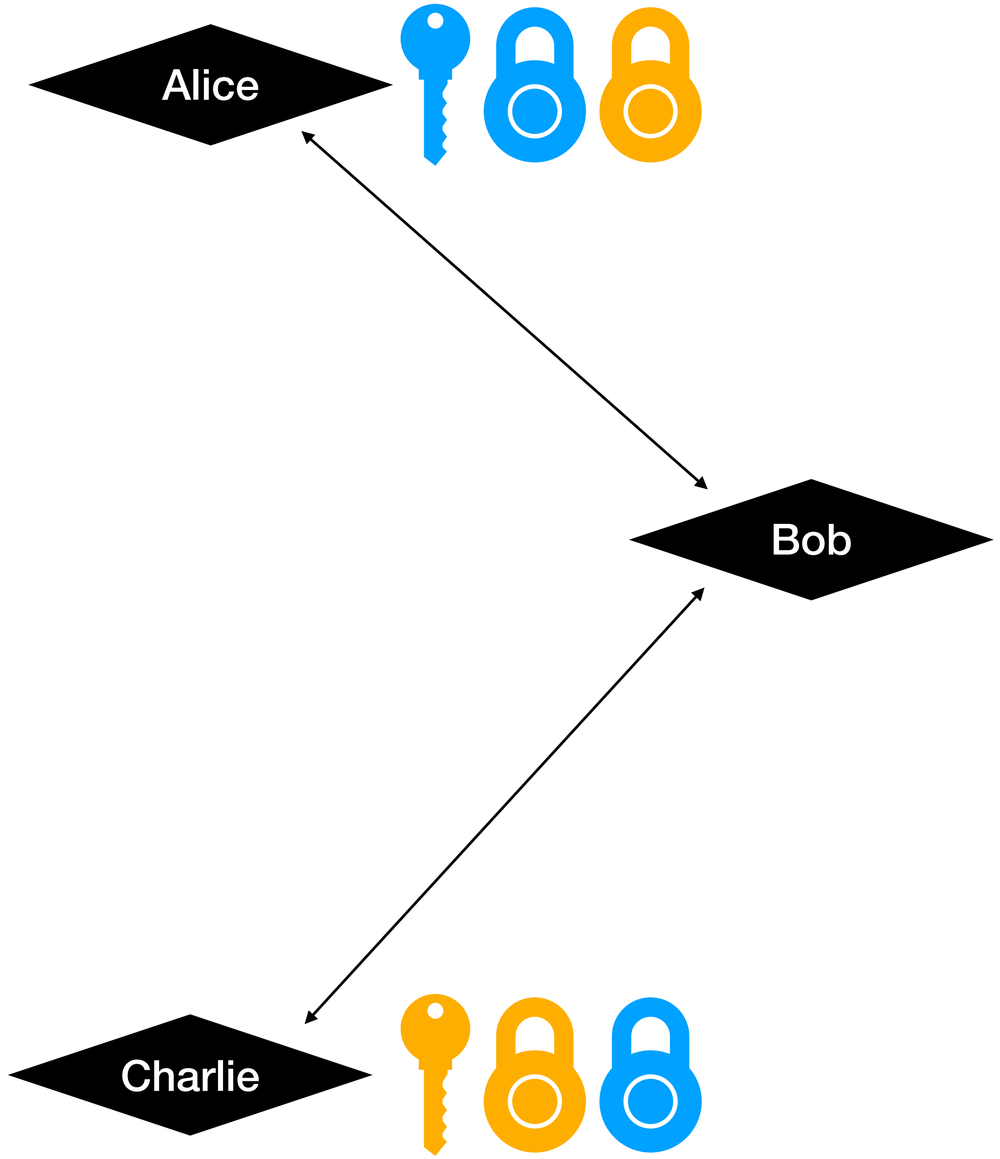


The Basic Model

Lockbox Problems

This solution is simple, but has major pitfalls:

- Locks do not *prevent* B from accessing the contents of the box if determined
- Locks can be physically destroyed (this tampering would be visually obvious)
- Locks can be picked non-destructively
- Internally, a physical key-based lock is a negative image of its key. *Access to the lock + time =* ***the ability to create a key copy***
- You can only have as many ongoing conversations as you have padlocks. Each party must send their own lock to get a secure response.

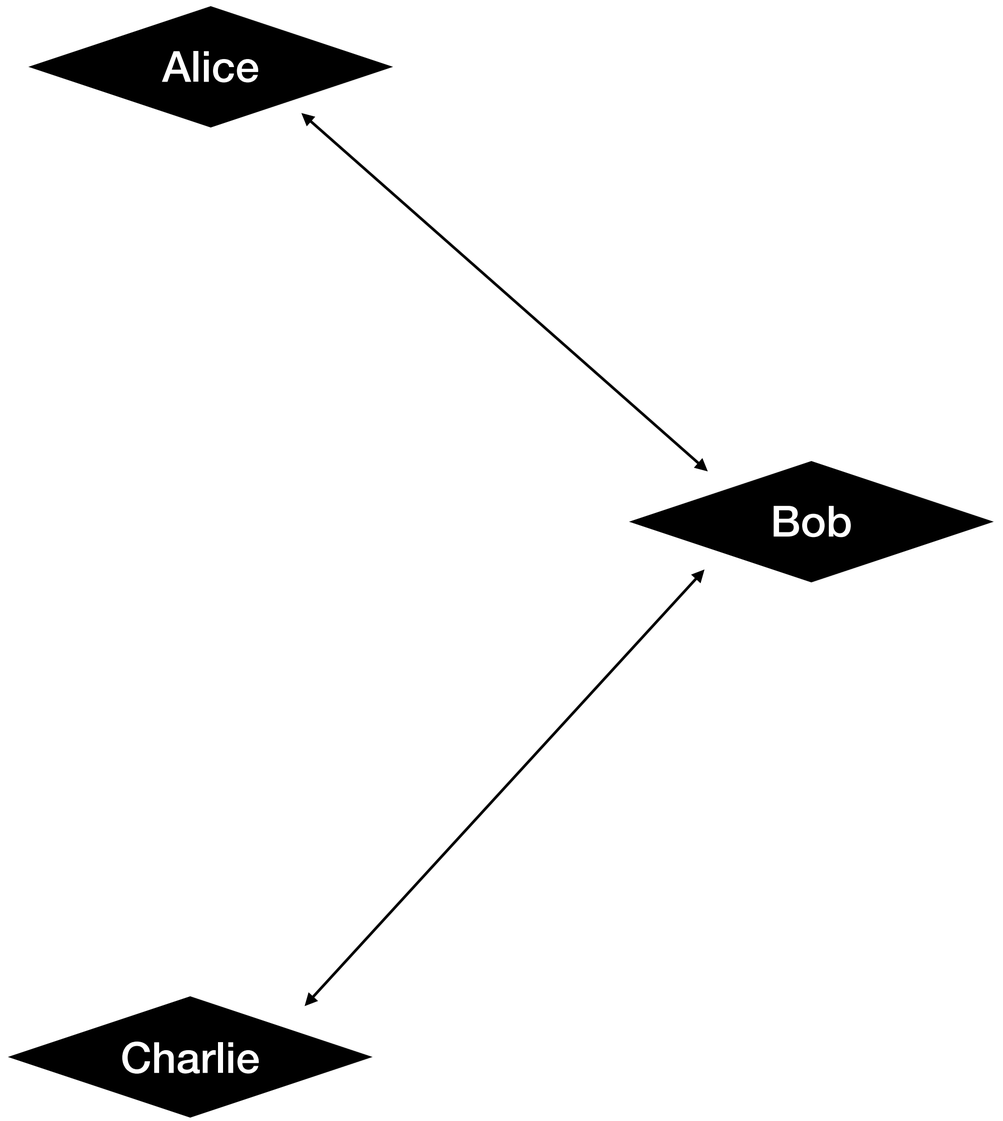


The Basic Model

Digital Lockbox

What if we kept the same basic idea, but rather than use physical-locks, we could use computers to create a “mathematical lock”?

- Can generate an infinite number of locks
- Not vulnerable to physical threats such as key-copying or destructive entry



Basics of Digital Data

Just a little byte

For our purposes today:

- “***Data***” refers to any sort of information in a digital format. This could be a simple line of text such as “*Hello!*” or a downloaded 2GB movie file.
- A ***“string”*** is a specific type of data, more commonly known as “human-readable text”
- The word ***“hexadecimal”*** *(*or “***hex***”) refers to data which can be represented entirely with the numbers *0 through 9* and the letters *A through F*
- A “***function***” refers to a process that is given data as an *input* and produces data as *output.*
- “***The wire***” refers to the medium through which data is transmitted. This can be a physical wire, a cellular connection, a package-carrier or even the air carrying spoken-word.

Basic Crypto Functions

Things To Do With Data

Although there are various algorithms that can be used to produce results, all modern cryptographic methods have these core functions:

- **Hashing** - With regards to data, a ***hash*** *is a* hexadecimal string that can be used as a *unique fingerprint* for a specific input. The act of generating a hash is called *hashing* (e.g. “I am hashing this file”).
    - The same input to a hash function should always produce the *exact same* result.
    - Can be used to verify that a file copied successfully and had no 0s change to 1s. In this context may also be called a “checksum”.
- **Signing** - A cryptographic ***signature*** is a unique, verifiable proof of identity. You could think about it as the digital equivalent of the signature you put on a legal document. It
    - Often times, a signature is combined with a hash of the content being signed. This enables not only verifying the authenticity of the source, but also that the data itself was not altered along the way.
- **Verifying -** Verification is the other half of signing. This function allows you to read a cryptographic signature and confirm its authenticity (or lack thereof)
- **Encrypting** - Encrypting is the act of taking input data (“*plaintext*”) and producing output data that is mathematically locked-up (“*ciphertext*”).
- **Decrypting** - Decrypting is the act of taking in *ciphertext* and producing output *plaintext.*

Asymmetric Key Cryptography

Scary Name; Easy Concepts

If you use a letter-to-number (A=1, B=2…) cipher system, decoding a message is just a matter of doing the process in reverse. Knowing the method to encrypt also reveals the way to decrypt. This type of process is a “symmetric cipher”.

Modern cryptography has introduced “asymmetric” systems. In an asymmetric system we define 2 linked components:

- ***Private key***
    - The actual “secret” component of the entire system
    - Enables **decryption**
    - Enables **signing**
- ***Public key***
    - Generated from the private key
    - Enables **encryption**
    - Enables **verification**
    - Shared with other people. No security risk from being public-knowledge.

Using the padlock analogy, a *private key* is akin to your actual key and a *public key* is like the padlock itself.

The term ***keypair*** can be used to refer to both items as a single set.

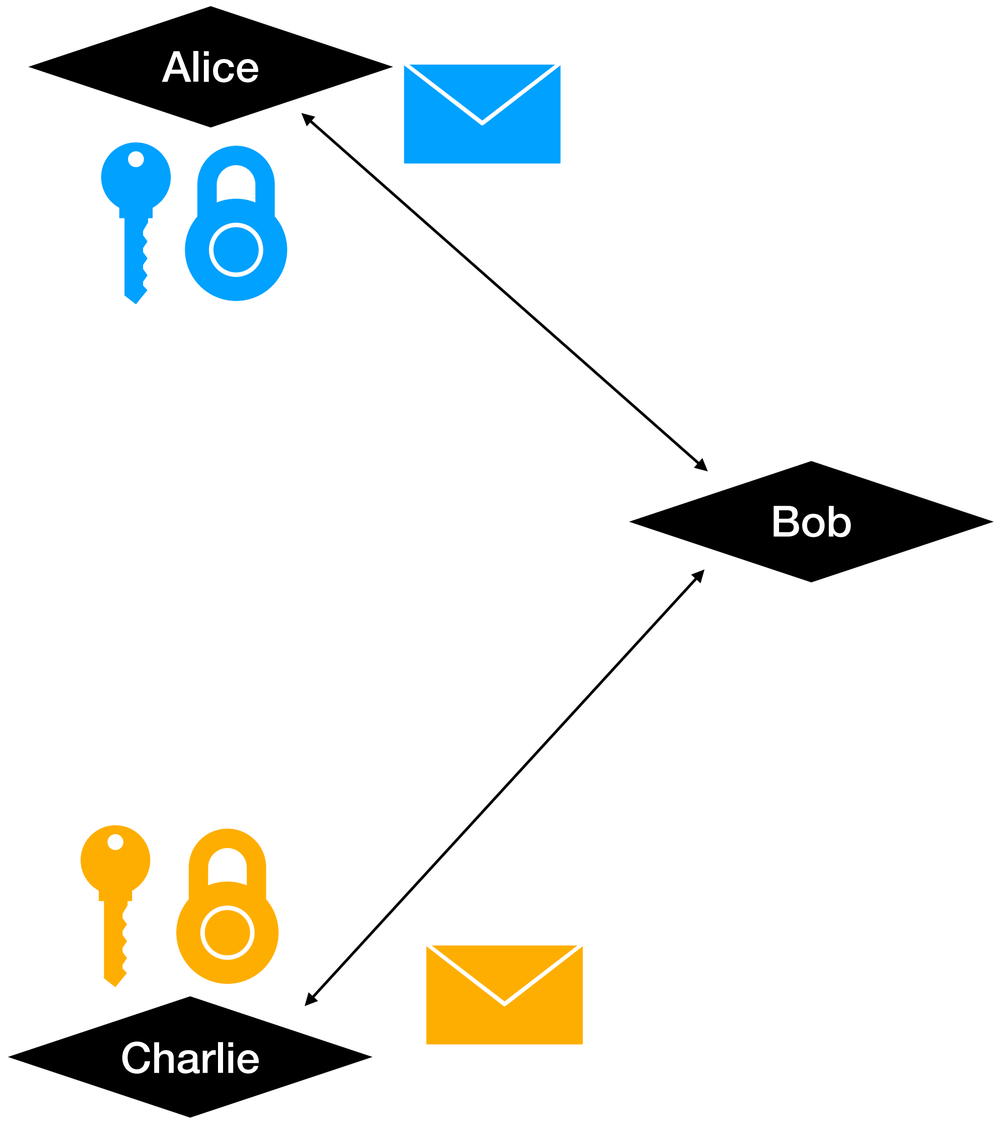
You would distribute the publicKey to anyone you wanted to be able to send you a private message. You would keep the private key for yourself and use it to decrypt messages once you got them.

Back To Basics

Lockbox + Crypto

Lets revisit the ABC problem with our new toolkit:

- A+C both generate their own keypairs
- A+C acquire a copy of each other's public keys (Can be directly exchanged or via mutual sources)
- A+C can encrypt messages for each other at any time an infinite amount of times
- We don't have to trust the security of "the wire"

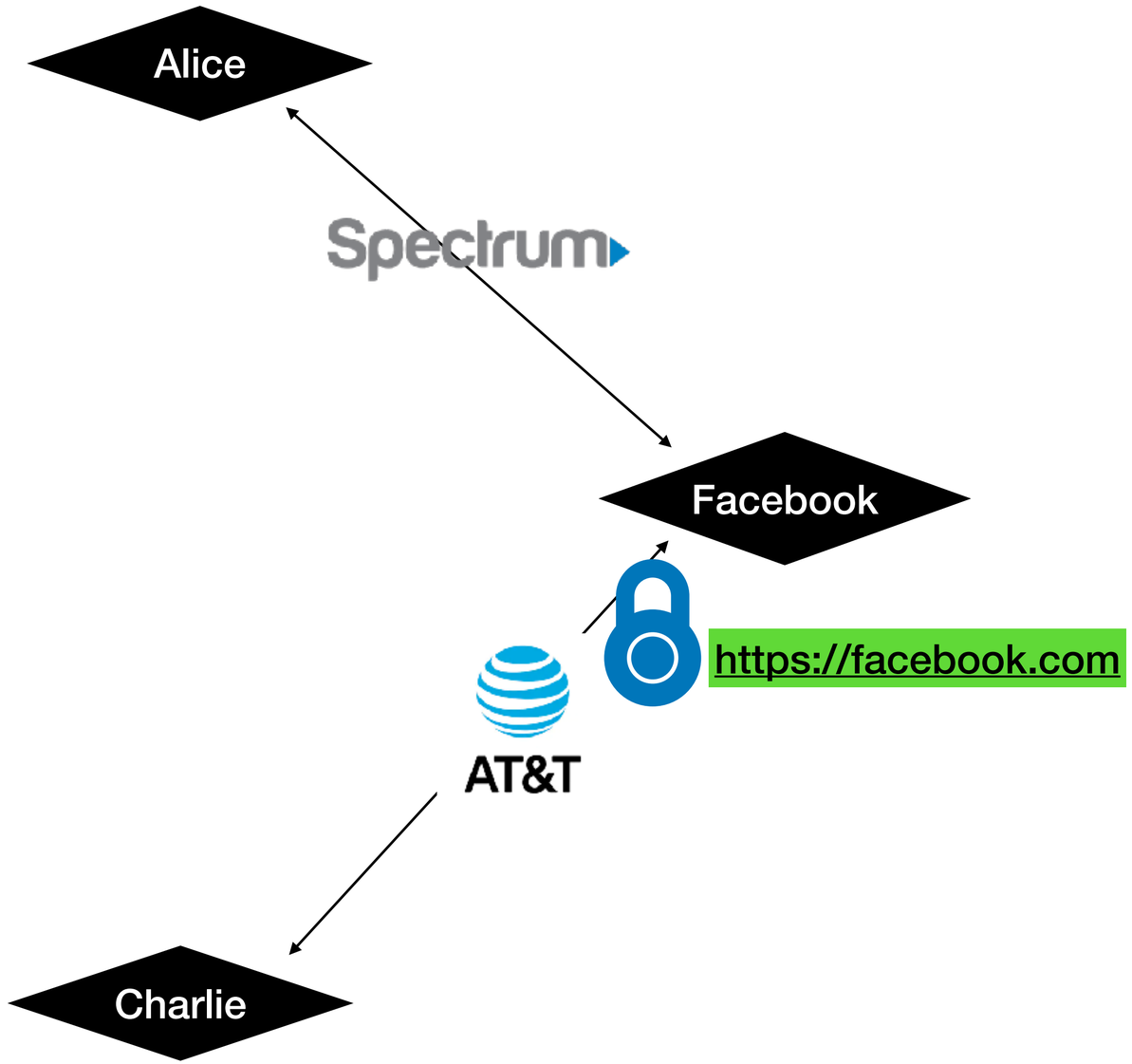


Demo Time

- This is a simple web app that will allow you to generate and use PGP keys
- https://mwthink.github.io/crypto-demo/

Expanding ABC
Real-World Application

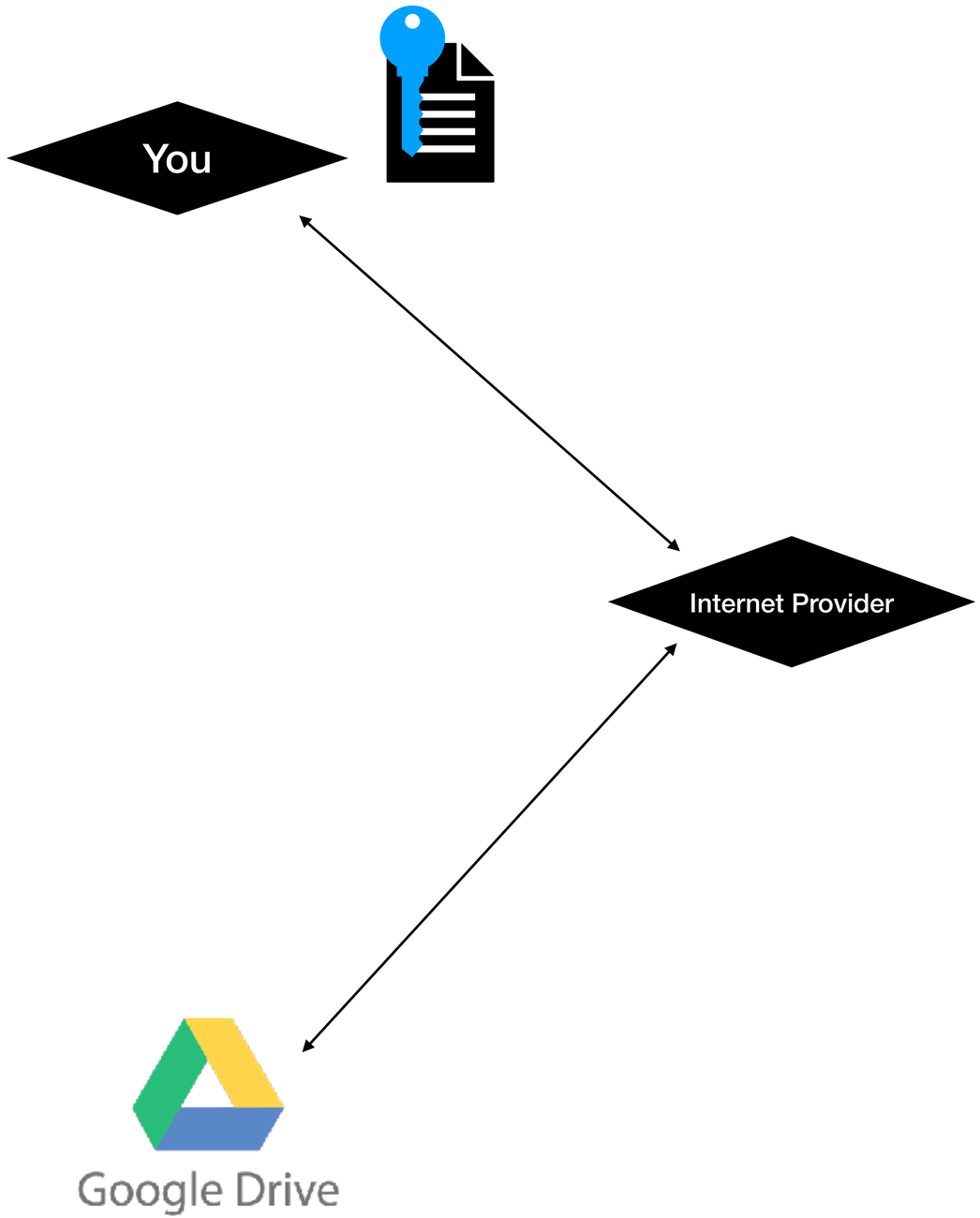
- In the real world, problems tend to be more complex than ABC, but we can take this model and adapt it for many scenarios.
- Consider that Alice and Charlie are both communicating on Facebook.
- Both are sending their messages to Facebook via their personal ISP.
- Not only is the data protected by the other side's personal key, but the transit to/from Facebook is also encrypted with their web server's public key (HTTPS)



Expanding ABC

More Real Uses

- This model isn’t restricted to humans + text communications
- Let’s say that you have some important personal files.
- You want to store them on a convenient cloud storage service.
- You want your files to be secure from prying eyes looking both during transit and at-rest.
- You can encrypt the data using your own public key and then upload the encrypted file(s).
- The storage provider is unable to read the contents at any point. You are responsible storage of the private key; it will be used to decrypt the files when downloaded later



Key Storage

Keeping the secret, secret

There are too many variables involved in key storage needs and possibilities to define a universal rule for secret storage. Many of the problems involved are non-technical and every good solution will be use-case specific.

That being said, here are some common considerations:

- Keys can be ***stored on a computer disk as a file***. This is the default assumption for most systems, but it leaves keys ***prone to attack*** on a live-system.
- ***Dedicated hardware devices*** such as a Trezor can be connected via USB to a computer. These devices allow generating a key and interacting with it ***without ever exposing the key*** to the (potentially infected) host computer.
- Ultimately, keys are just a set of numbers represented in hex-values. These letters and numbers can be ***written on paper or etched into steel*** and stored in a ***bank deposit box*** (or, wherever)

Dedicated hardware devices are often combined with steel-backups, offering the convenience of digital access but the durability of steel.

Pictured: Trezor Model T & Cryptosteel Capsule

Rationalizing Cryptographic Systems

Making Sense of Anything, Anywhere

- The basic principles just discussed are universal. Any modern system using encryption will fit into the ABC model.
    - By not “trusting the wire”, we can securely communicate over any channel, even ones we know to be under active surveillance.
- Using this model, you can assess basic security of almost any service by asking “Who owns the keys?”
    - Virtually all (properly implemented) modern cryptography can be generally treated as “equals” in terms of strength (At time of writing, in 2022)
    - Phrases like “we use the strongest encryption” are marketing gimmicks. *Everyone uses the strongest encryption; ready-access is widespread since the Internet.*
- Services may use the same encryption standards, but key management practices can vary wildly
    - Consumer-facing cloud services (e.g. iCloud) often tout good encryption, but they **tend to sacrifice good *security* for good *customer service***. If you lose your password/backup key and the service is able to restore your access after a 1-800 call, they have access to the encryption key and could read that data without you.
    - Password managers generally have you save a “backup phrase” onto paper as a backup key. Usually the company storing your passwords is not capable of reading them even if they wanted to. If you locked yourself out of this account, companies cannot restore the data; at most they could wipe the account back to a “new” state
- We have only scratched the surface. There are many different concepts and features built atop these principles:
    - Multi-recipient encryption
    - Public-key expiration times
    - Multi-party key sharding

Q&A

> *Insert audience questions here*