RSA and the Cloud

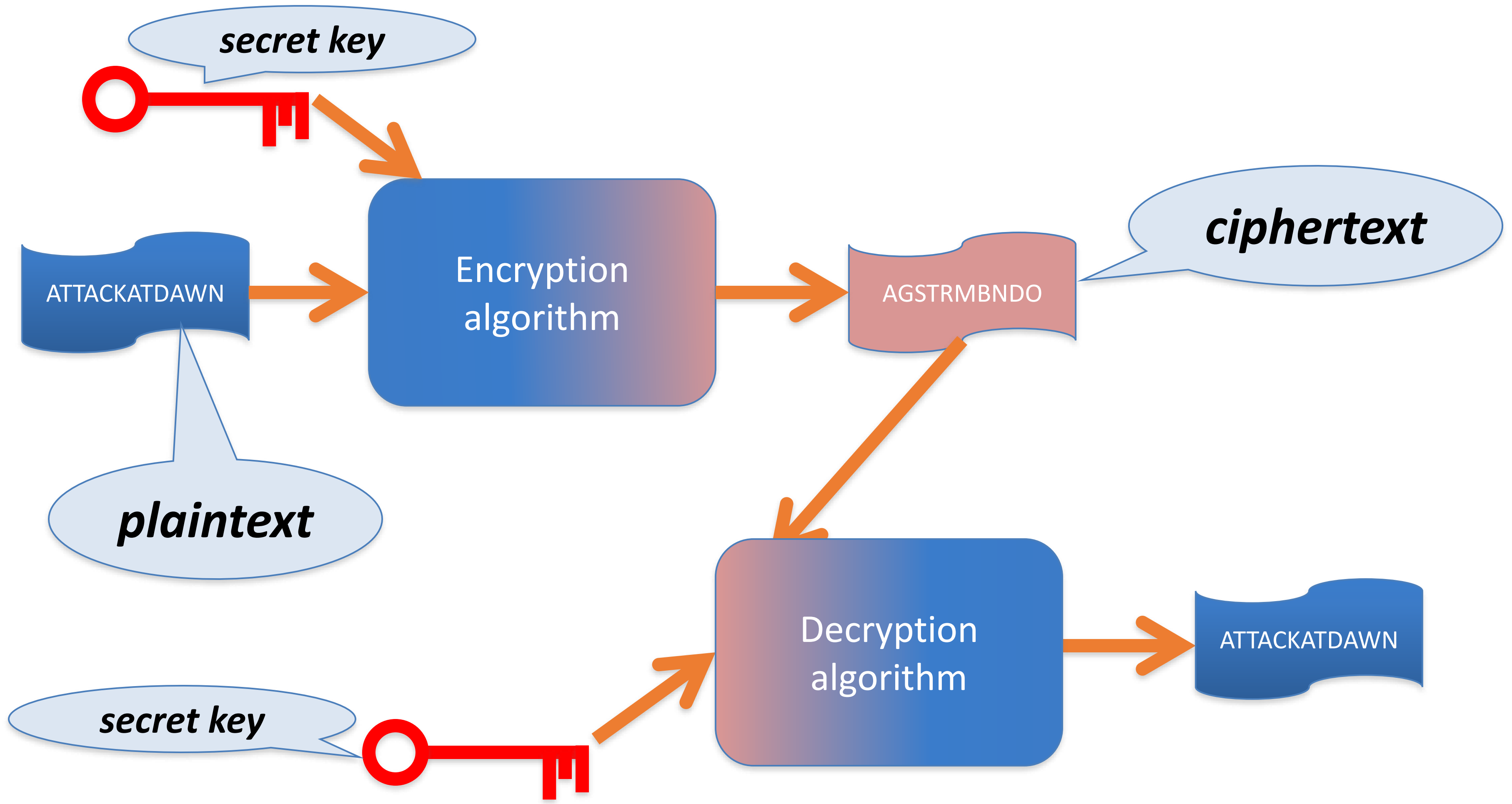
Kelly Rivers and Stephanie Rosenthal

15-110 Fall 2019

Encryption

- We encrypt (encode) our data so others can't understand it (easily) except for the person who is supposed to receive it.
- We call the data to encode plaintext and the encoded data the ciphertext.
- Encoding and decoding are *inverse functions* of each other.
- Basic assumption: the encryption/decryption *algorithm* is known; only the key is secret
  - The key is the password that helps someone decrypt a message
  - As long as the key is strong, it will be near-impossible for others to guess it

Encryption/decryption



Common Encryption Encodings

Caesar Cipher

Key idea – shift the letters in the alphabet by a certain amount to encrypt the message. Shift it the same number of letters back in the other direction to decrypt.

Example: “Hi, my name is Stephanie” -> shifted 5 characters (and lowercase)

“mn, rd sfrj nx xyjumfsj”

If your message receiver knows 5, they can decode by shifting by -5 letters

Common Encryption Encodings

Substitution Cipher

Key idea – since there are only a finite (26) number of Caesar ciphers, instead mix up all the letters randomly and substitute the ith letter for the ith index in the substitution

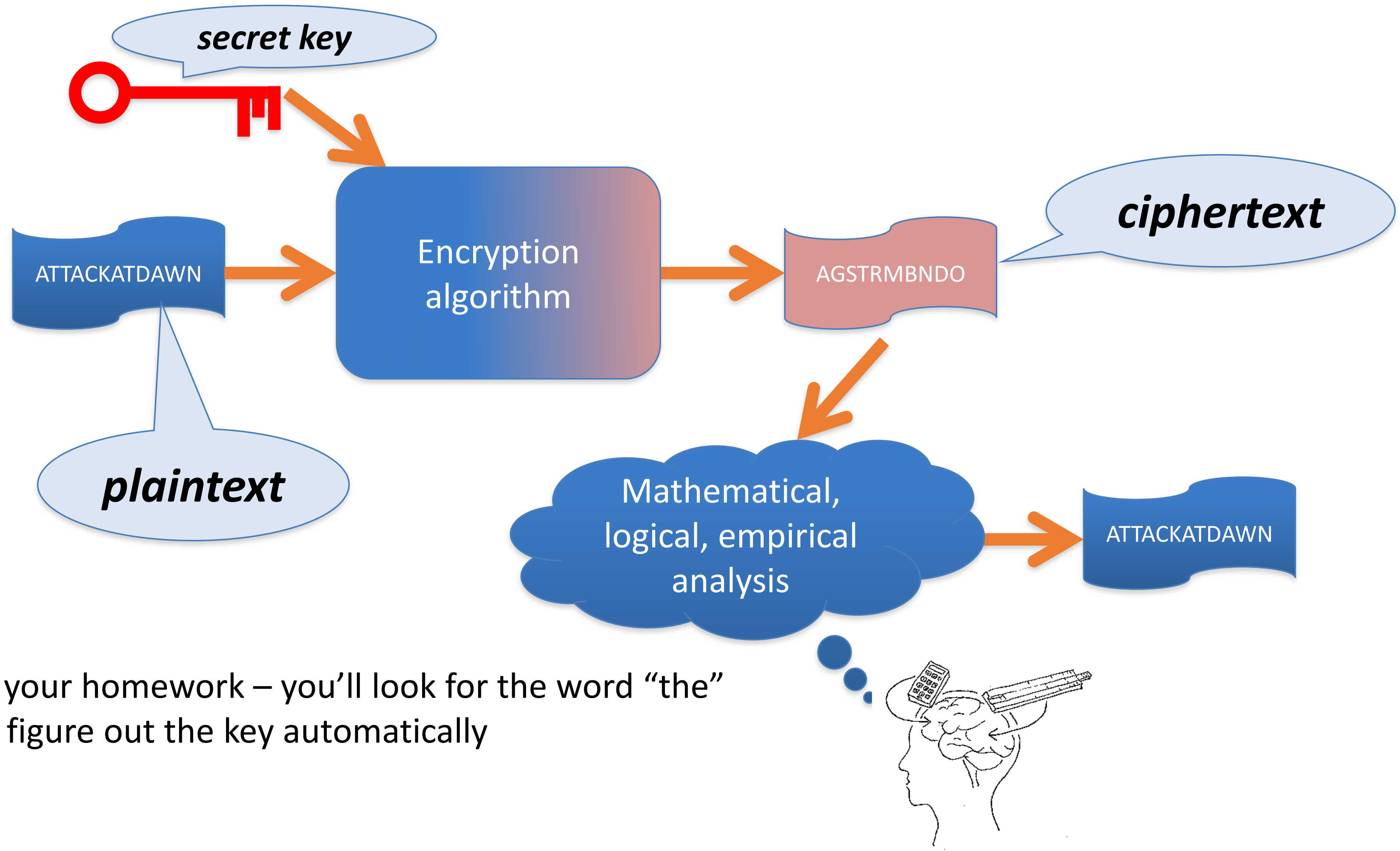
Example: “Hi, my name is Stephanie” -> [qwertyuiopasdfghjklzxcvbnm]

h is the 7th letter (0 index), so use the 7th substitution i

i is the 8th letter, so use the 8th substitution o, …

Complete message: “io, dn fqdt ol lzthiqfot”

Cryptanalysis



In your homework – you'll look for the word "the" to figure out the key automatically

Common Encryption Encodings

Substitution Cipher

Key idea – since there are only a finite (26) number of Caesar ciphers, instead mix up all the letters randomly and substitute the ith letter for the ith index in the substitution

There are 26! 4x10^23 combinations of letters, so the likelihood of decoding a message is very low unless you have the key (the substitution list)

Many other encodings

Most popular today is to multiply the message by really big numbers to get different bit encodings

- A message in ASCII can also be interpreted as a binary number.
- Multiply this number by another really big number to encrypt

Keyspace

- Keyspace is the number of possible secret keys, for a particular encryption or decryption algorithm (26 for Caesar cipher, 26! for substitution)

- Number of bits per key determines size of keyspace
    - important because we want to make ***brute force attacks*** infeasible
    - brute force attack: run the (known) decryption algorithm repeatedly with **every possible key** until a sensible plaintext appears

- Typical key sizes: several hundred bits (numbers on the order of 2^{100})

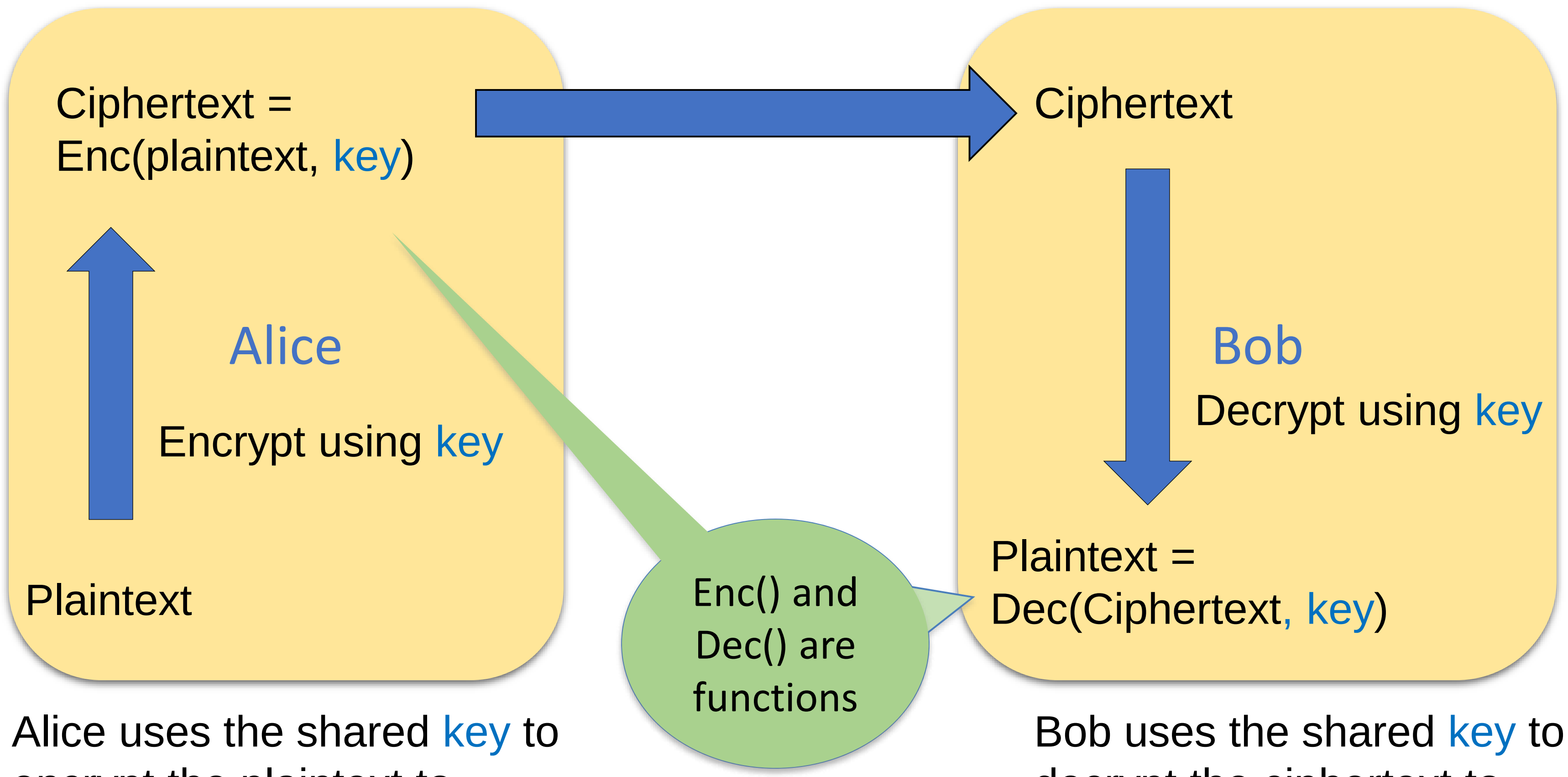
Symmetric Encryption

Symmetric (shared-key) encryption: commonly used for long messages

The sender and receiver know a single key and can use it together

- A sender couldn’t use the same key for different receivers
- Often a complicated mix of substitution and transposition encipherment
- Reasonably fast to compute
- Requires a shared secret key usually communicated using (slower) *asymmetric encryption*

Symmetric (Shared Key) Encryption


Ciphertext = Enc(plaintext, key)
Alice
Encrypt using key
Plaintext
Enc() and Dec() are functions
Ciphertext
Bob
Decrypt using key
Plaintext = Dec(Ciphertext, key)

Alice uses the shared key to encrypt the plaintext to produce the ciphertext

Bob uses the shared key to decrypt the ciphertext to recover the plaintext

Establishing Shared Keys

Substitution ciphers and Caesar ciphers require shared keys

Problem: how can Alice and Bob secretly agree on a key, using a public communication system?

- Diffie-Hellman Key Exchange protocol achieves this
- Not going to go into detail in this class

Asymmetric Encryption

Symmetric (shared-key) encryption: commonly used for long messages

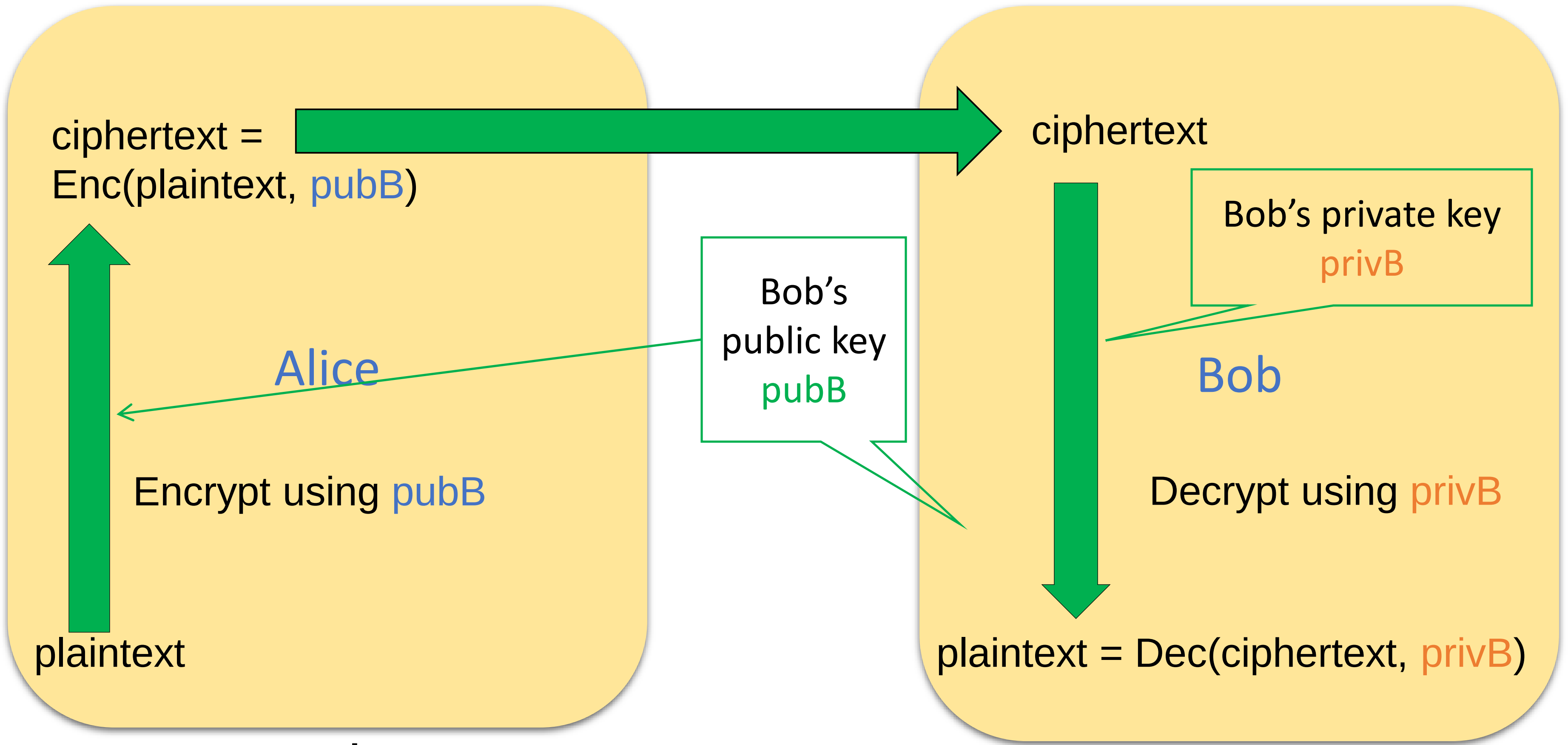
The sender and receiver know a single key and can use it together

- A sender couldn't use the same key for different receivers
- Often a complicated mix of substitution and transposition encipherment
- Reasonably fast to compute
- Requires a shared secret key usually communicated using (slower) *asymmetric encryption*

Asymmetric encryption: different keys are used to encrypt and to decrypt

- Make one key public for senders to encrypt, no one can decrypt except the receiver

Asymmetric Public Key Encryption



Alice uses Bob's public key to encrypt the plaintext to produce the ciphertext

Bob uses his private key to decrypt the ciphertext to recover the plaintext

One type of asymmetric encryption: RSA

- Named after its inventors: Rivest, Shamir and Adleman
- Used in https (you know when you're using it because you see the URL in the address bar begins with https://)

How RSA works

First, we must be able to represent any message as a single number

For example:

A T **T** A **C** K **A** T **D** A **W** N

0120**20**01**03**11**01**20**04**01**23**14

Public and Private Keys

used for encryption

Every receiver has a public key (e, n) and a private key (d, n).

used for decryption

The transmitter encrypts a (numerical) message M into ciphertext C using the receiver's public key:

$$M^{e} \text{ modulo } n \longrightarrow C \quad \textit{(ciphertext)}$$

The receiver decodes the encrypted message C to get the original message M using the private key (which no one else knows).

$$C^{d} \text{ modulo } n \longrightarrow M \quad \textit{(plaintext)}$$

RSA Example

Bob’s Public Key: (3, 33) (e = 3, n = 33)

Bob’s Private Key: (7, 33) (d = 7, n = 33)

(Usually these are really huge numbers with many hundreds of digits!)

Alice wants to send the message 4

Alice encrypts the message using *e* and *n*: 4^3 modulo 33 $\longrightarrow$ 31.

Alice sends 31

Bob receives the encoded message 31

Bob decrypts the message using d and n: 31^7 modulo 33 $\longrightarrow$ 4

Bob receives 4

In case you're curious: generating *n*, *e* and *d*

• p and q are (big) random primes.	$p = 3$, $q = 11$
• $n = p \times q$	$n = 3 \times 11 = 33$
• $\varphi = (p - 1)(q - 1)$	$\varphi = 2 \times 10 = 20$
• e is small and relatively prime to φ (only positive integer that divides both of them is 1)	$e = 3$
• d, such that $e \times d \bmod \varphi = 1$	$3 \times d \bmod 20 = 1$ $d = 7$

Usually the primes are huge numbers--hundreds of digits long.

Cracking RSA

Everyone knows (e, n). Only Bob knows d. Cracking RSA requires learning d.

If we know e and n, can we figure out d?

- If so, we can read secret messages to Bob.

We **can** determine d from e and n.

- Factor n into p and q.
  $n = p \times q$
  $\varphi = (p - 1)(q - 1)$
  $e \times d = 1 \pmod{\varphi}$
- We know e (which is public), so we can solve for d.

But only if we can factor n

RSA is safe (for now)

Suppose someone can factor my 5-digit *n* in 1 ms,

At this rate, to factor a 10-digit number
would take 2 minutes. *(2 minutes is about 10^5 times 1 ms)*

- ... to factor a 15-digit number
  would take 4 months. *(4 months is about 10^5 times 2 minutes)*
- ... 20-digit number ... 30,000 years.
- ... 25-digit number... 3 billion years.

We're safe with RSA! (at least, from factoring with digital computers)

RSA Takeaways

RSA includes a public key and a private key.

Anyone can have access to the public key and encrypt messages.

Only the person with the private key can decrypt the message using their private key.

If we could multiply numbers really quickly, we could try a lot of different encodings, but in general we cannot so this encoding scheme is pretty safe for now

- If P = NP, then multiplication would be very fast and encryption would break!

Security is only as good as your weakest link

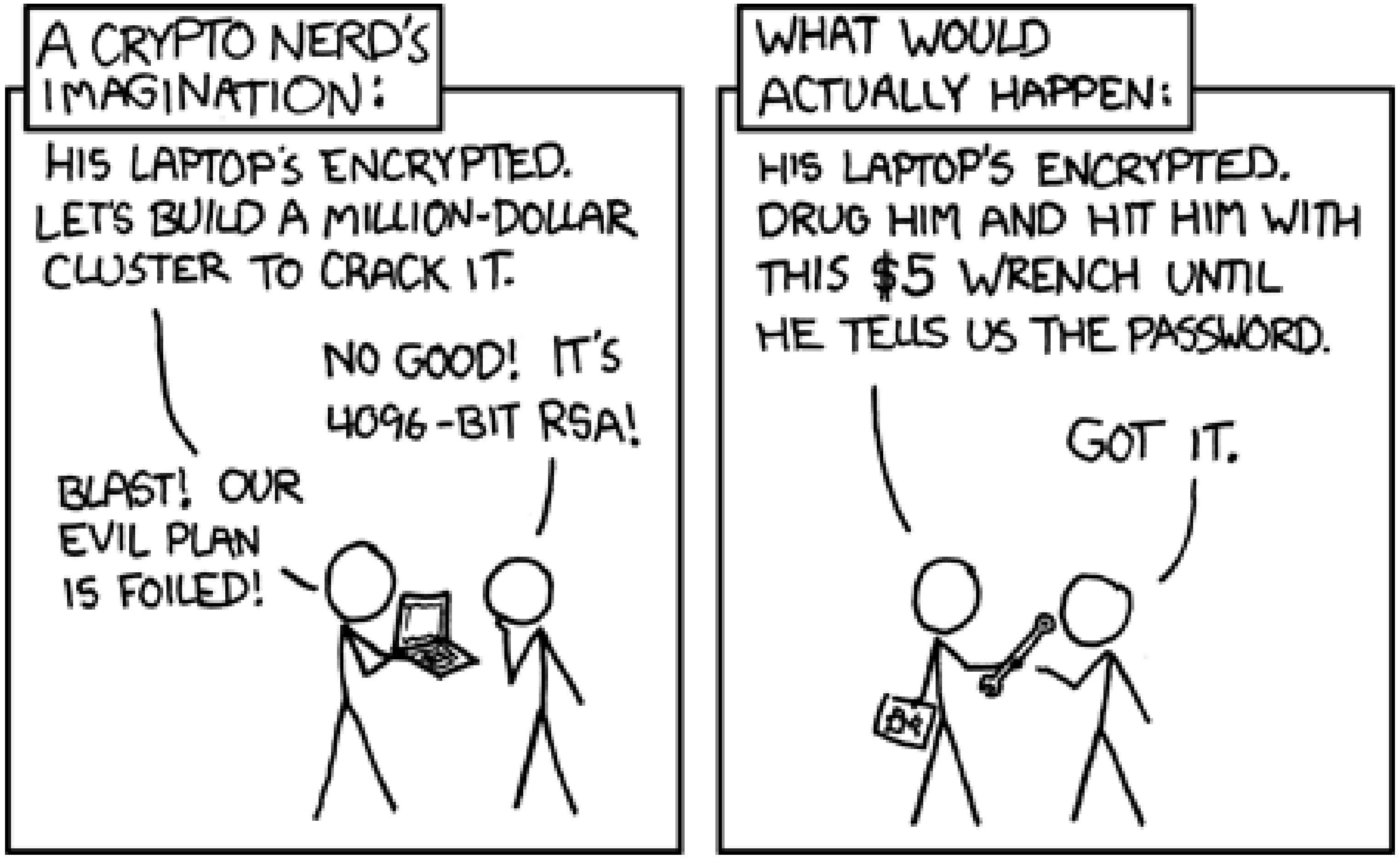

A CRYPTO NERD'S IMAGINATION:
HIS LAPTOP'S ENCRYPTED. LET'S BUILD A MILLION-DOLLAR CLUSTER TO CRACK IT.
NO GOOD! IT'S 4096-BIT RSA!
BLAST! OUR EVIL PLAN IS FOILED!
WHAT WOULD ACTUALLY HAPPEN:
HIS LAPTOP'S ENCRYPTED. DRUG HIM AND HIT HIM WITH THIS $5 WRENCH UNTIL HE TELLS US THE PASSWORD.
GOT IT.

image: http://xkcd.com/538/

The Cloud

What is Cloud Computing?

Cloud computing is a set of services which allow users to access a number of resources in a way that is elastic, cost-efficient, and on-demand.

Cloud computing is an umbrella term used to refer to Internet based development and services.

Characteristics of cloud data, applications, services, and infrastructure:

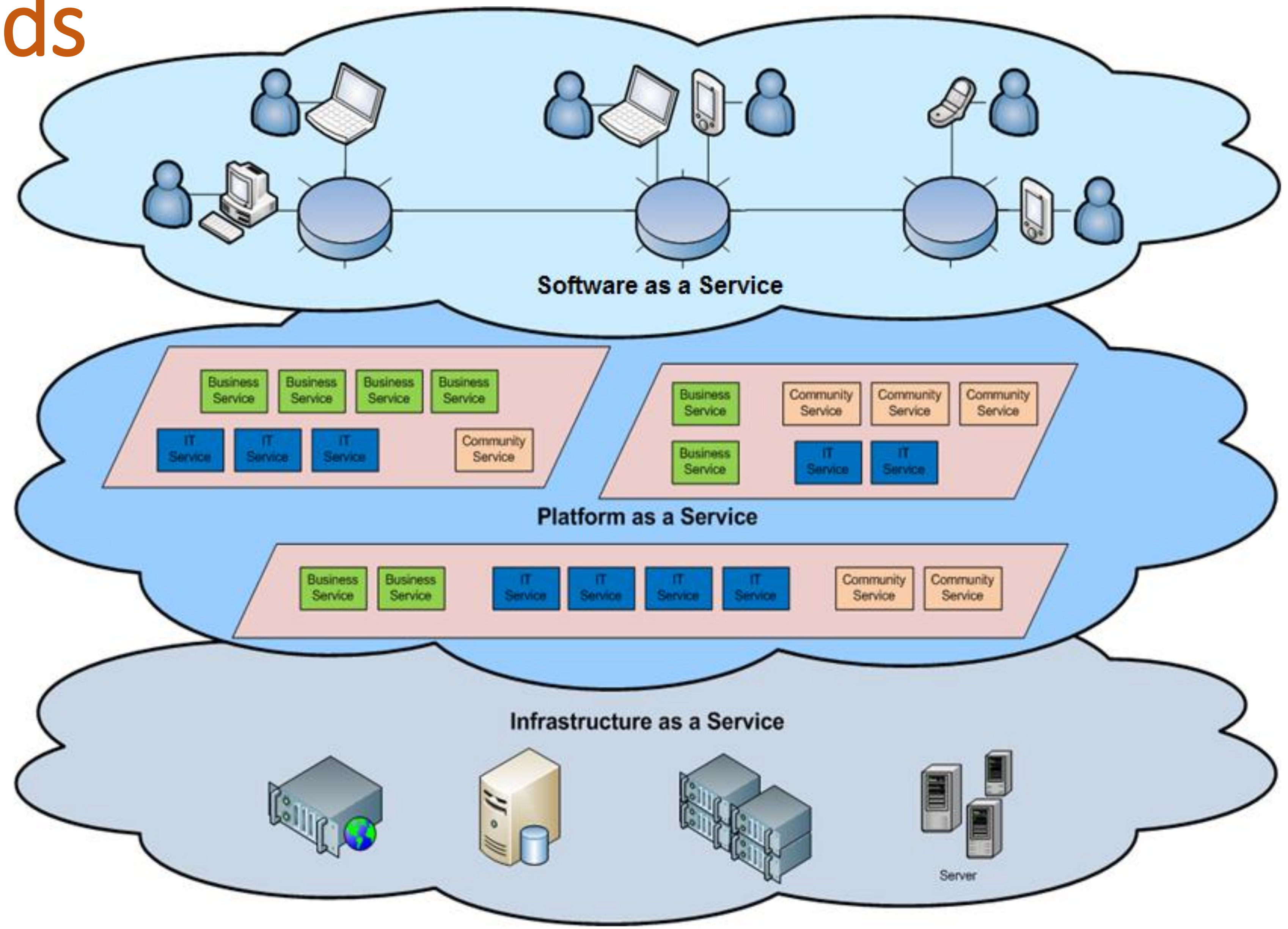
- Remotely hosted: Services and data are hosted on remote resources.
- Ubiquitous: Services and data are available from anywhere.
- Commodified: The result is a utility computing model similar to traditional utilities such as electricity and water. You pay for what you use!

What is the cloud used for?

Everyone has an opinion on what to use a cloud for

- Applications on the internet – email, tax prep, word processing
- Memory/Storage for business, personal data
- Web services for photos, maps, GPS, building websites
- Computing for machine learning and artificial intelligence

Types of Clouds



Types of Clouds

Software as a Service (SaaS) – an application hosted on a website

- Gmail, Google Docs, Microsoft 365, Salesforce.com, Quicken Online

Platform as a Service (PaaS) – place for others to built apps

- Alexa/Echo, Google App Engine, Squarespace, Amazon Web Services

Infrastructure as a Service (IaaS) – available hardware to use

- Storage - Apple Cloud, Google Drive, Dropbox, Box
- Computers - Google Compute Engine, Amazon EC2

Another Look at Types of Clouds

	Services	Description
Application Focused	**Services**	**Services – Complete business services such as PayPal, OpenID, OAuth, Google Maps, Alexa**
	Application	**Application – Cloud based software that eliminates the need for local installation such as Google Apps, Microsoft Online**
	Development	**Development – Software development platforms used to build custom cloud based applications (PAAS & SAAS) such as SalesForce**
Infrastructure Focused	**Platform**	**Platform – Cloud based platforms, typically provided using virtualization, such as Amazon ECC, Sun Grid**
	Storage	**Storage – Data storage or cloud based NAS such as iCoud, Dropbox, CloudNAS**
	Hosting	**Hosting – Physical data centers such as those run by IBM, HP, Amazon, etc.**

How does it work?

- Exactly like any other distributed system
- At cloud companies, many many computers are networked together and store data redundantly in case any of them go down.
- They have servers that you connect to through “front end” websites. Those websites access your data within their network and give you applications and personalized content
- There is a lot of work that goes into making these cloud services robust to failures (computers and memory crash or die all the time) and super fast so that you don’t even think about your data not being on your computer

Internet of Things

- Cloud of sensors, actuators (things that act on the world), and computers
- What are the considerations of robustness, security, and privacy for IoT that are the same and different from other cloud and distributed systems?

Questions about Cloud Services