1. Encryption & Security

Overview

- Security issues
- Encryption and cryptanalysis
- Encryption in the digital age
  - Symmetric encryption
  - Asymmetric encryption
- Applications of encryption
- Encryption is not security!

Security issues

Networking is a security issue

- Why?
- If you want a really secure machine, lock it in an electromagnetically shielded room and don't connect it to any networks or other sources of data beyond your control (totally isolated island)
- Not much fun, is it?

The Problem

- The Internet is public
  - Messages sent pass through many machines and media

- Anyone intercepting a message might
  - read it and/or
  - replace it with a different message

- The Internet is anonymous
  - IP addresses don’t establish identity

- Anyone may send messages under a false identity

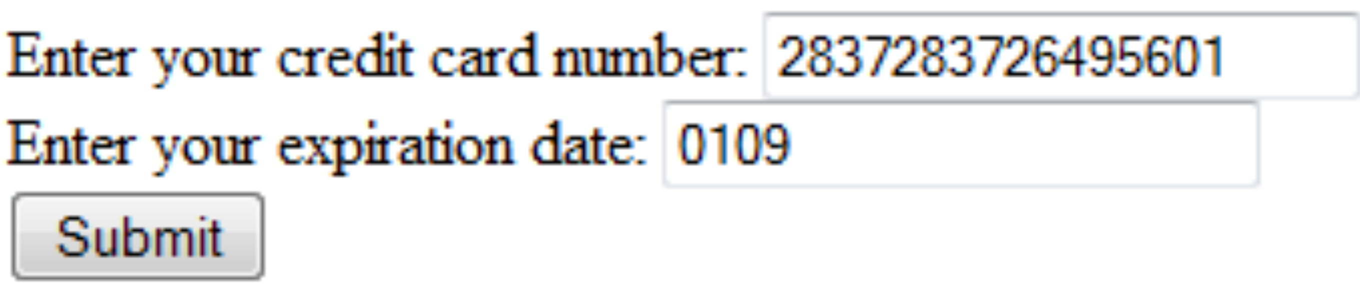
The Problem

- The Internet is public
  - Messages sent pass through many machines and media
- Anyone intercepting a message might
  - read it and/or
  - replace it with a different message
- The Internet is anonymous
  - IP addresses don’t establish identity
- Anyone may send messages under a false identity

Cryptography offers *partial* solutions to all of these problems

A Shady Example

- I want to make a purchase online and click a link that takes me to http://www.sketchystore.com/checkout.jsp
- What I see in my browser:



A Shady Example (cont’d)

- When I press SUBMIT, my browser sends this:

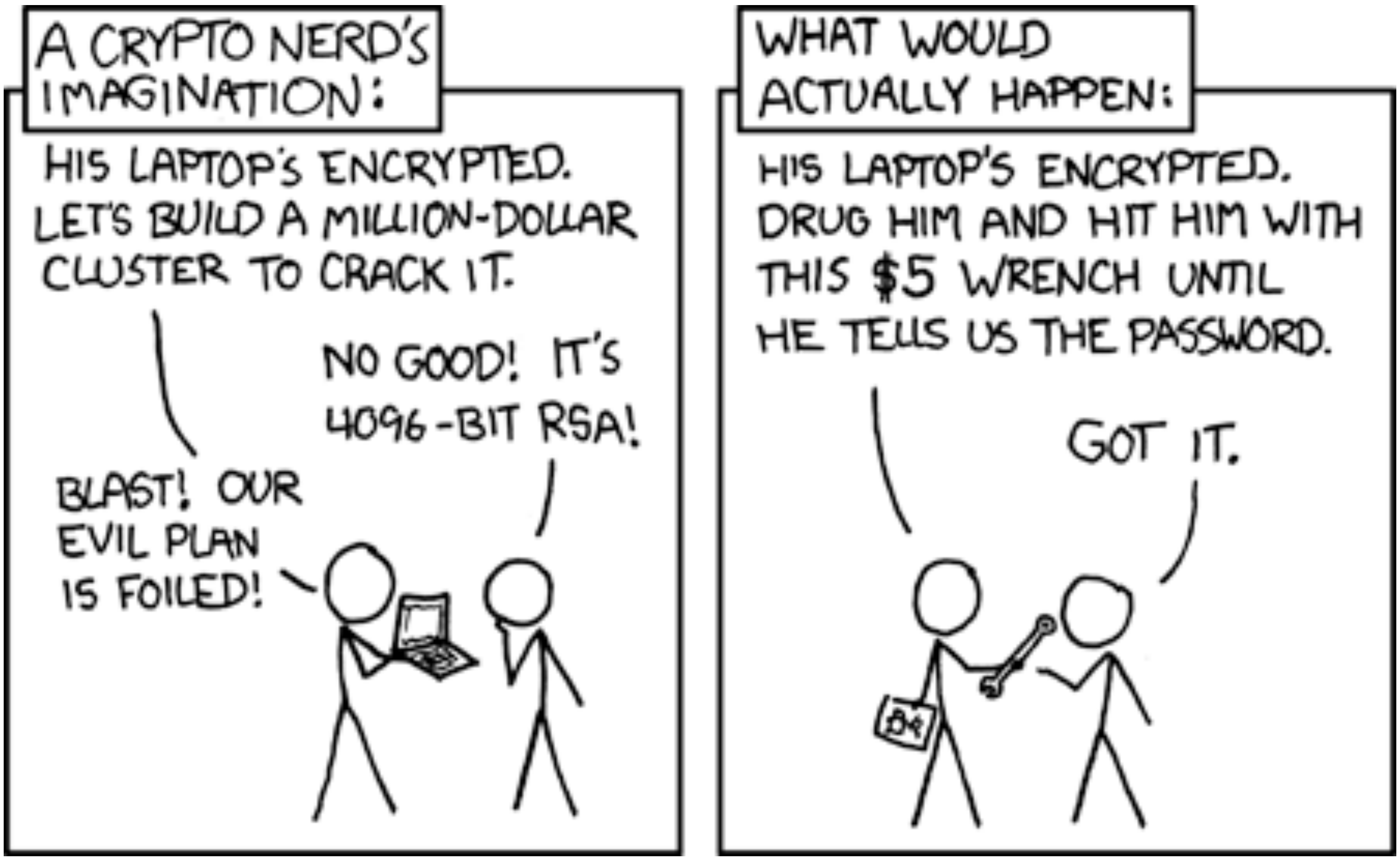
```
POST /purchase.jsp HTTP/1.1

Host: www.sketchystore.com

User-Agent: Mozilla/4.0

Content-Length: 48

Content-Type: application/x-www-form-urlencoded

userid=rbd&creditcard=2837283726495601&
  exp=01/09
```

A Shady Example (cont’d)

- If this information is sent unencrypted, who has access to my credit card number?
  - Other people who can connect to my wireless ethernet
  - Other people physically connected to my wired ethernet
  - …
- Packets are passed from router to router.
  - *All* those routers have access to my data.

A caveat

cryptography is not security


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.

Encryption and cryptanalysis

basic concepts

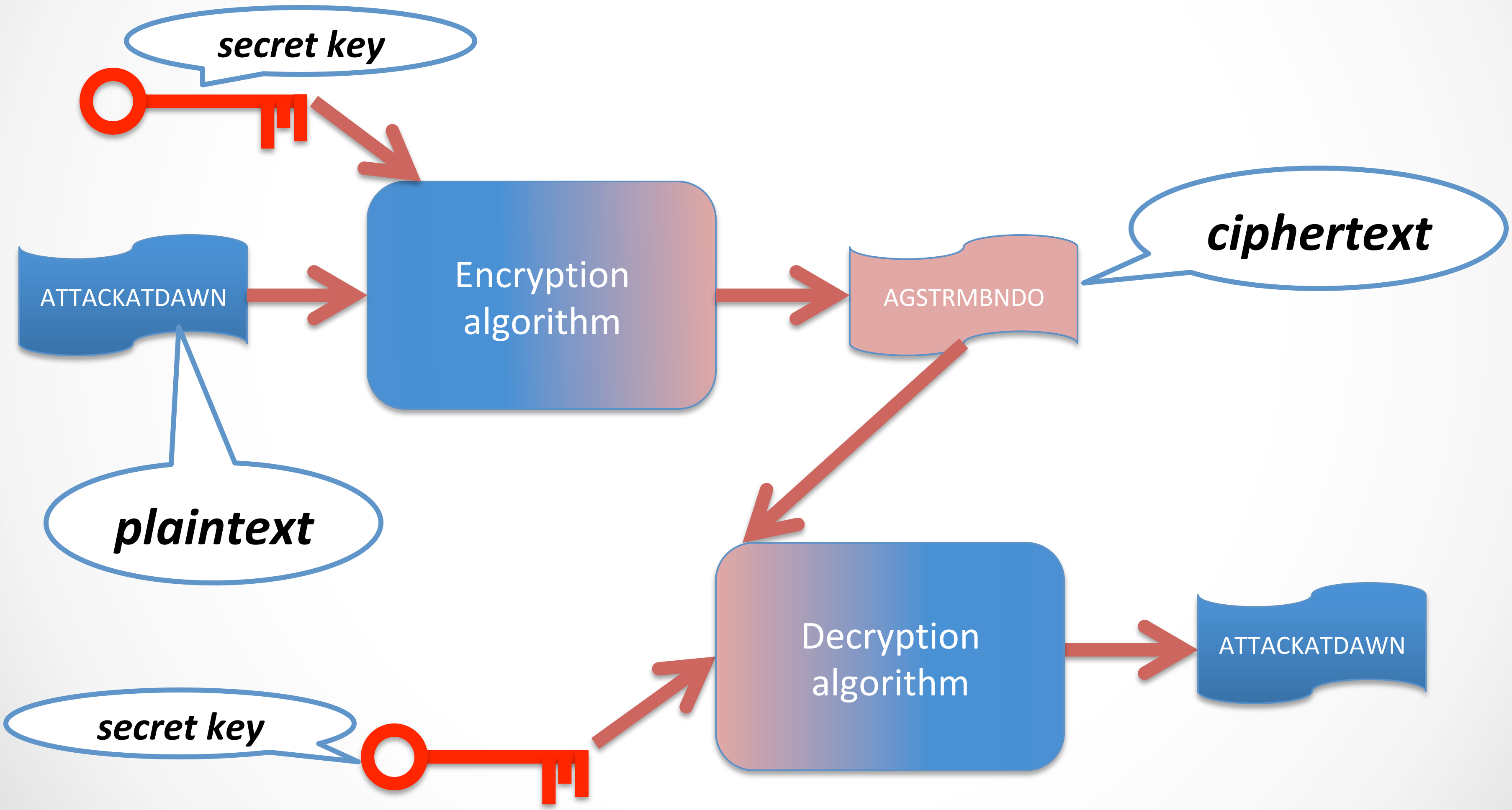
Encryption

- We encrypt (encode) our data so others can’t understand it (easily) except for the person who is supposed to receive it.

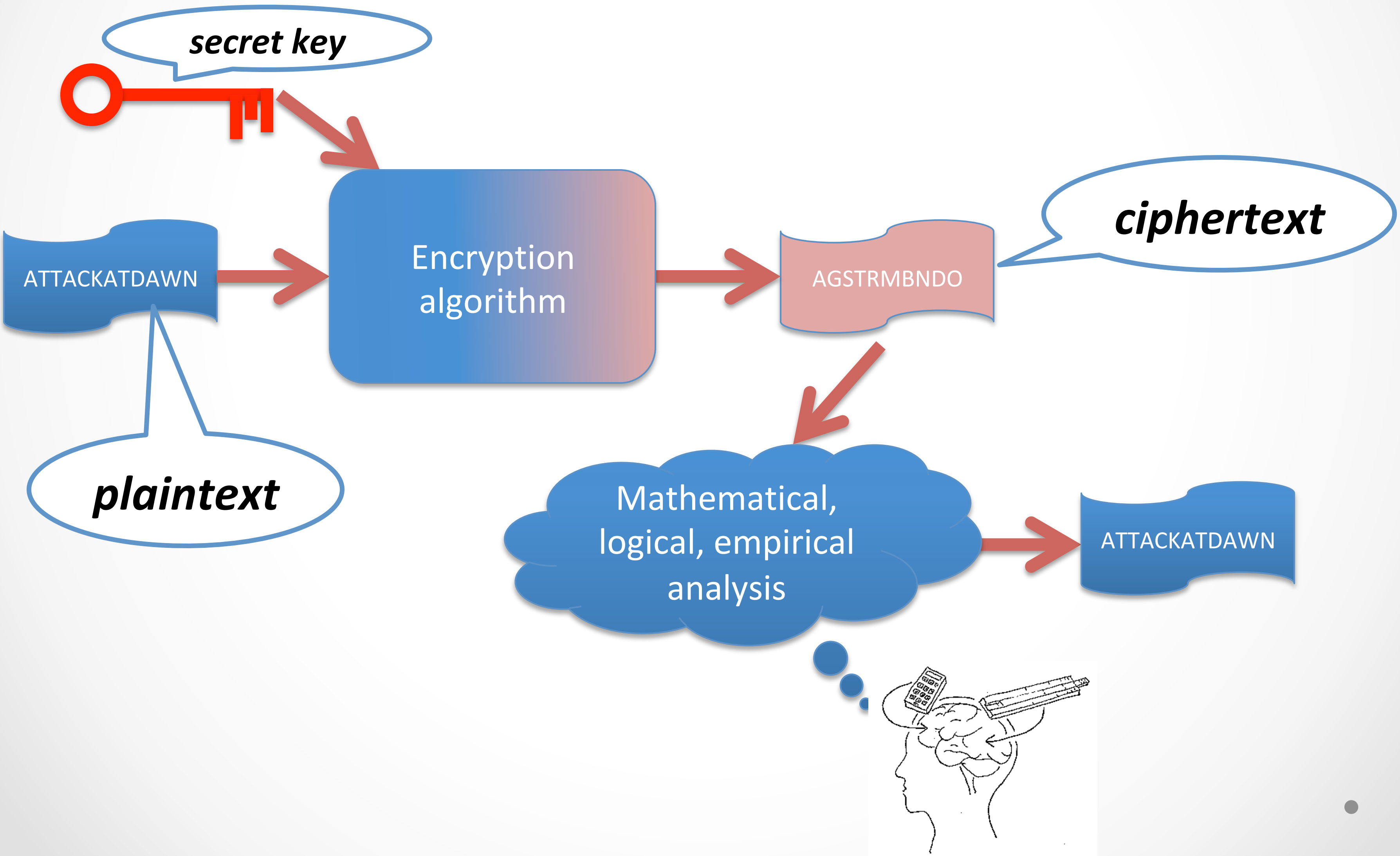
- We call the data to encode **plaintext** and the encoded data the **ciphertext**.

- Encoding and decoding are *inverse functions* of each other

Encryption/decryption



Cryptanalysis



Encryption techniques

substitution and transposition

Two basic ways of altering text to encrypt/decrypt

- Substitute one letter for another using some kind of rule

Substitution cipher

- Scramble the order of the letters using some kind of rule

Transposition cipher

Substitution Ciphers

Substitution Ciphers

- Simple encryption scheme using a substitution cipher:
  - Shift every letter forward by 1:

    A → B, B → C, ..., Z → A

Substitution Ciphers

ABCDEFGHIJKLMNOPQRSTUVWXYZ
ABCDEFGHIJKLMNOPQRSTUVWXYZ

ABCDEFGHIJKLMNOPQRSTUVWXYZ
ABCDEFGHIJKLMNOPQRSTUVWXYZ

ABCDEFGHIJKLMNOPQRSTUVWXYZ
BCDEFGHIJKLMNOPQRSTUVWXYZA

Substitution Ciphers

- Simple encryption scheme using a substitution cipher:
  - Shift every letter forward by 1:

    A $\longrightarrow$ B, B $\longrightarrow$ C, ..., Z $\longrightarrow$ A

- Example:
  MESSAGE $\longrightarrow$ NFTTBHF

Substitution Ciphers

MESSAGE → NFTTBHF

ABCDEFGHIJKLMNOPQRSTUVWXYZ
BCDEFGHIJKLMNOPQRSTUVWXYZA

Substitution Ciphers

- Simple encryption scheme using a substitution cipher:
  - Shift every letter forward by 1:

    A ⟶ B, B ⟶ C, …, Z ⟶ A

- Example:
  MESSAGE ⟶ NFTTBHF

- Can you decrypt TFDSFU?

Substitution Ciphers

TFDSFU

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
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	A

Substitution Ciphers

- Simple encryption scheme using a substitution cipher:
  - Shift every letter forward by 1:

    A $\longrightarrow$ B, B $\longrightarrow$ C, ..., Z $\longrightarrow$ A

- Example:
  MESSAGE $\longrightarrow$ NFTTBHF

- Can you decrypt TFDSFU? SECRET

Caesar Cipher

- Shift forward *n* letters; *n is the secret key*
- For example, shift forward 3 letters:
  A ⟶ D, B ⟶ E, ..., Z ⟶ C
  - This is a Caesar cipher using a **key** of 3 (n=3).
- MESSAGE ⟶ PHVVDJH
- How can we crack this encrypted message if we don't know the key?
  DEEDUSEKBTFEIIYRBOTUSETUJXYI

Caesar Cipher (cont' d)

```
DEEDUSEKBTFEIIYRBOTUSETUJXYI
EFFEVTFLCUGFJJZSCPUVTFUVKYZJ
FGGFWUGMDVHGKKATDQVWUGVWLZAK
GHHGXVHNEWIHLLBUERWXVHWXMABL
HIIHYWIOFXJIMMCVFSXYWIXYNBCM
IJJIZXJPGYKJNNDWGTYZXJYZOCDN
JKKJAYKQHZLKOOEXHUZAYKZAPDEO
KLLKBZLRIAMLPPFYIVABZLABQEFP
LMMLCAMSJBNMQQGZJWBCAMBCRFGQ
MNNMDBNTKCONRRHAKXCDBNCDSGHR
NOONECOULDPOSSIBLYDECODETHIS
OPPOFDPVMEQPTTJCMZEFDPEFUIJT
PQQPGEQWNFRQUUKDNAFGEQFGVJKU
QRRQHFRXOGSRVVLEOBGHFRGHWKLV
RSSRIGSYPHTSWWMFPCHIGSHIXLMW
STTSJHTZQIUTXXNGQDIJHTIJYMNX
TUUTKIUARJVUYYOHREJKIUJKZNOY
UVVULJVBSKWVZZPISFKLJVKLAOPZ
VWWVMKWCTLXWAAQJTGLMKWLMBPQA
WXXWNLXDUMYXBBRKUHMNLXMNCQRB
XYYXOMYEVNZYCCSLVINOMYNODRSC
YZZYPNZFWOAZDDTMWJOPNZOPESTD
ZAAZQOAGXPBAEEUNXKPQOAPQFTUE
ABBARPBHYQCBFFVOYLQRPBQRGUVF
BCCBSQCIZRDCGGWPZMRSQCRSHVWG
CDDCTRDJASEDHHXQANSTRDSTIWXH
```

- How long would it take a computer to try all 25 shifts?

Vigenère Cipher

- Shift different amount for each letter. Use a *key word*; each letter in the key determines how many shifts we do for the corresponding letter in the message.
- Example: key word "`cmu`": shift by 2, 12, 20
- Message "`pittsburgh`"

  `cmucmucmuc`

  encrypted: `runvevwdaj`

- Try it yourself at http://www.simonsingh.net/The_Black_Chamber/v_square.html

	ABCDEFGHIJKLMNOPQRSTUVWXYZ	
A	ABCDEFGHIJKLMNOPQRSTUVWXYZ	no shift
B	BCDEFGHIJKLMNOPQRSTUVWXYZA	shift by 1
C	CDEFGHIJKLMNOPQRSTUVWXYZAB	shift by 2
D	DEFGHIJKLMNOPQRSTUVWXYZABC	shift by 3
E	EFGHIJKLMNOPQRSTUVWXYZABCD	etc.
F	FGHIJKLMNOPQRSTUVWXYZABCDE	

...

- Message: ATTACKATDAWN
- Pick a secret key DECAFDECAFDE
- Encrypted: D

1st letter in the message is shifted by 3, 2nd letter is shifted by 4, …

```
    ABCDEFGHIJKLMNOPQRSTUVWXYZ
A | ABCDEFGHIJKLMNOPQRSTUVWXYZ
B | BCDEFGHIJKLMNOPQRSTUVWXYZA
C | CDEFGHIJKLMNOPQRSTUVWXYZAB
D | DEFGHIJKLMNOPQRSTUVWXYZABC
E | EFGHIJKLMNOPQRSTUVWXYZABCD
F | FGHIJKLMNOPQRSTUVWXYZABCDE
...
```

- Message: ATTACKATDAWN
- Pick a secret key DECAFDECAFDE
- Encrypted: DX

1st letter in the message is shifted by 3, 2nd letter is shifted by 4, …

```
    ABCDEFGHIJKLMNOPQRSTUVWXYZ
A | ABCDEFGHIJKLMNOPQRSTUVWXYZ
B | BCDEFGHIJKLMNOPQRSTUVWXYZA
C | CDEFGHIJKLMNOPQRSTUVWXYZAB
D | DEFGHIJKLMNOPQRSTUVWXYZABC
E | EFGHIJKLMNOPQRSTUVWXYZABCD
F | FGHIJKLMNOPQRSTUVWXYZABCDE
...
```

- Message: ATTACKATDAWN
- Pick a secret key DECAFDECAFDE
- Encrypted: DXV

1st letter in the message is shifted by 3, 2nd letter is shifted by 4, …

	ABCDEFGHIJKLMNOPQRSTUVWXYZ
A	ABCDEFGHIJKLMNOPQRSTUVWXYZ
B	BCDEFGHIJKLMNOPQRSTUVWXYZA
C	CDEFGHIJKLMNOPQRSTUVWXYZAB
D	DEFGHIJKLMNOPQRSTUVWXYZABC
E	EFGHIJKLMNOPQRSTUVWXYZABCD
F	FGHIJKLMNOPQRSTUVWXYZABCDE

...

- Message: ATTACKATDAWN
- Pick a secret key DECAFDECAFDE
- Encrypted: DXVAHNEVDFZR

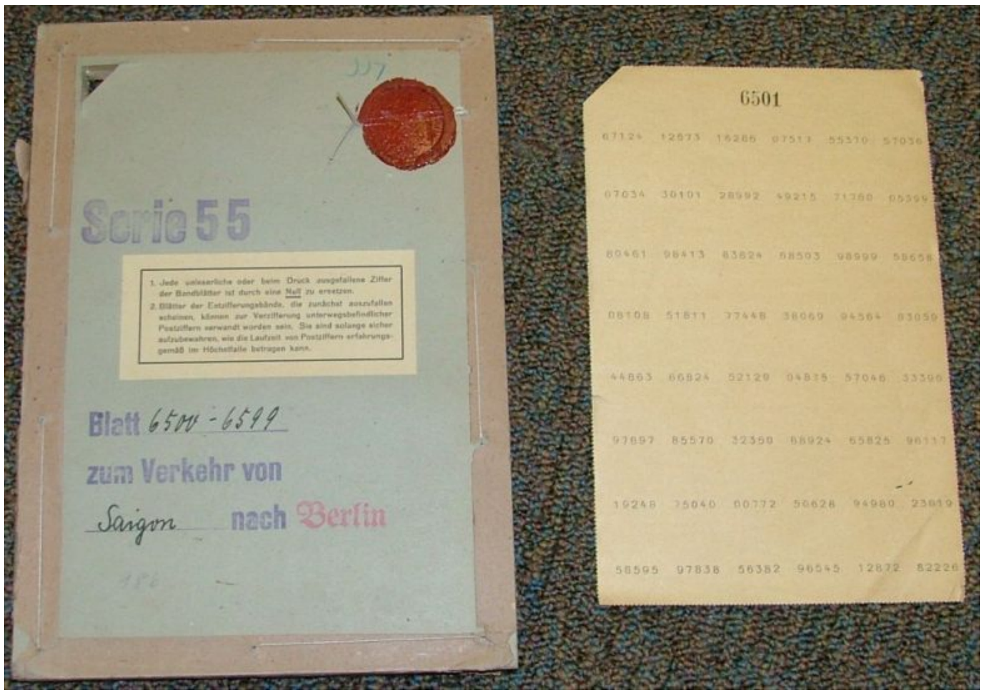
1st letter in the message is shifted by 3, 2^{nd} letter is shifted by 4, …

Vernam Cipher

- Vigenère cipher was broken by Charles Babbage in the mid 1800s by exploiting the repeated key
  - The length of the key determines the cycle in which the cipher is repeated.
- Vernam cipher: make the key the same length as the message; Babbage's analysis doesn't work.

One-time Pads

- Vernam cipher is commonly referred to as a one-time pad.



Alice and Bob have identical "pads" (shared keys)

- If random keys are used one-time pads are unbreakable in theory.

Transposition Ciphers

Transposition ciphers

an ancient Greek method

STSF...EROL...NOUA...DOTN…MPHK…OSEA…RTRN…EOND…

image:http://crypto.interactive-maths.com/simple-transposition-ciphers.html

Encryption in computing

fast computation makes encryption usable by all of us

Encryption in computing

- One-time pads impractical on the net (why?)
- Basic assumption: the encryption/decryption *algorithm* is known; only the key is secret (why?)
- Very complicated encryptions can be computed fast:
  - typically, elaborate combinations of substitution and transposition

HTTPS

- Security protocol for the Web, the peoples' encryption
- Purpose:
  - confidentiality (prevent eavesdropping)
  - message integrity and authentication (prevent "man in the middle" attacks that could alter the messages being sent)
- Techniques:
  - asymmetric encryption ("public key" encryption) to exchange secret key
  - certificate authority to obtain public keys
  - symmetric encryption to exchange actual messages

Keyspace

- *Keyspace* is jargon for the number of possible secret keys, for a particular encryption/decryption algorithm

- Number of bits per key determines *size of keyspace*
  - important because we want to make *brute force attacks* infeasible

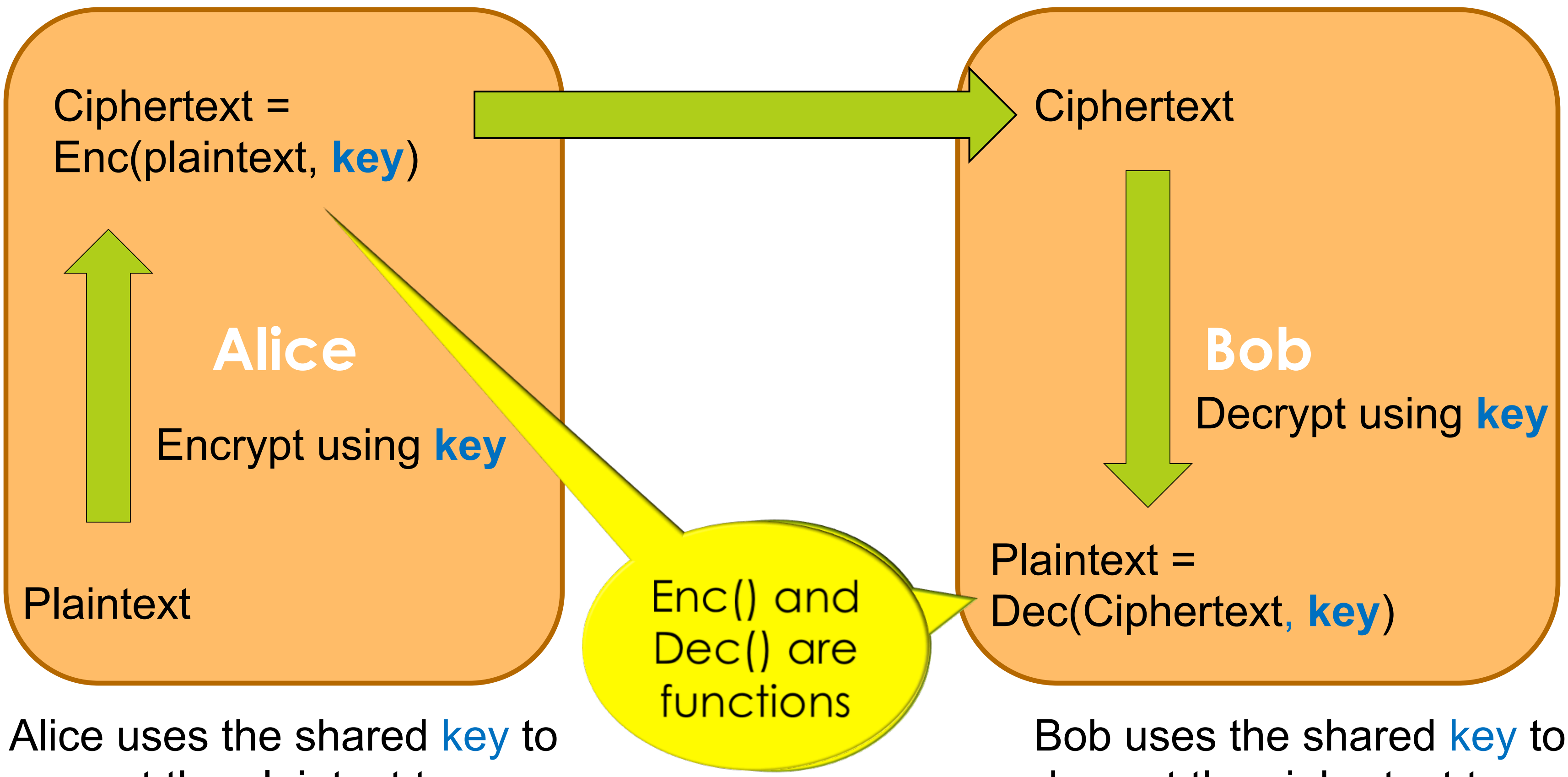
- *Brute force attack*: run the (known) decryption algorithm repeatedly with **every possible key** until a sensible plaintext appears

- Typical key sizes: several hundred bits

Symmetric vs. asymmetric encryption

- **Symmetric** (shared-key between sender and receiver) encryption: commonly used for long messages
  - Often a complicated mix of substitution and transposition encipherment
  - Reasonably fast to compute
  - Examples (Caesar Cipher)
  - Requires a shared secret key usually communicated using (slower) *asymmetric encryption*

Symmetric (Shared Key) Encryption



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

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

- Solution: asymmetric encryption based on *number theory*
  - Alice has one secret, Bob has a different secret; working together they establish a shared secret
  - Examples: Diffie-Hellman key exchange, RSA public key encryption

Symmetric vs. asymmetric encryption

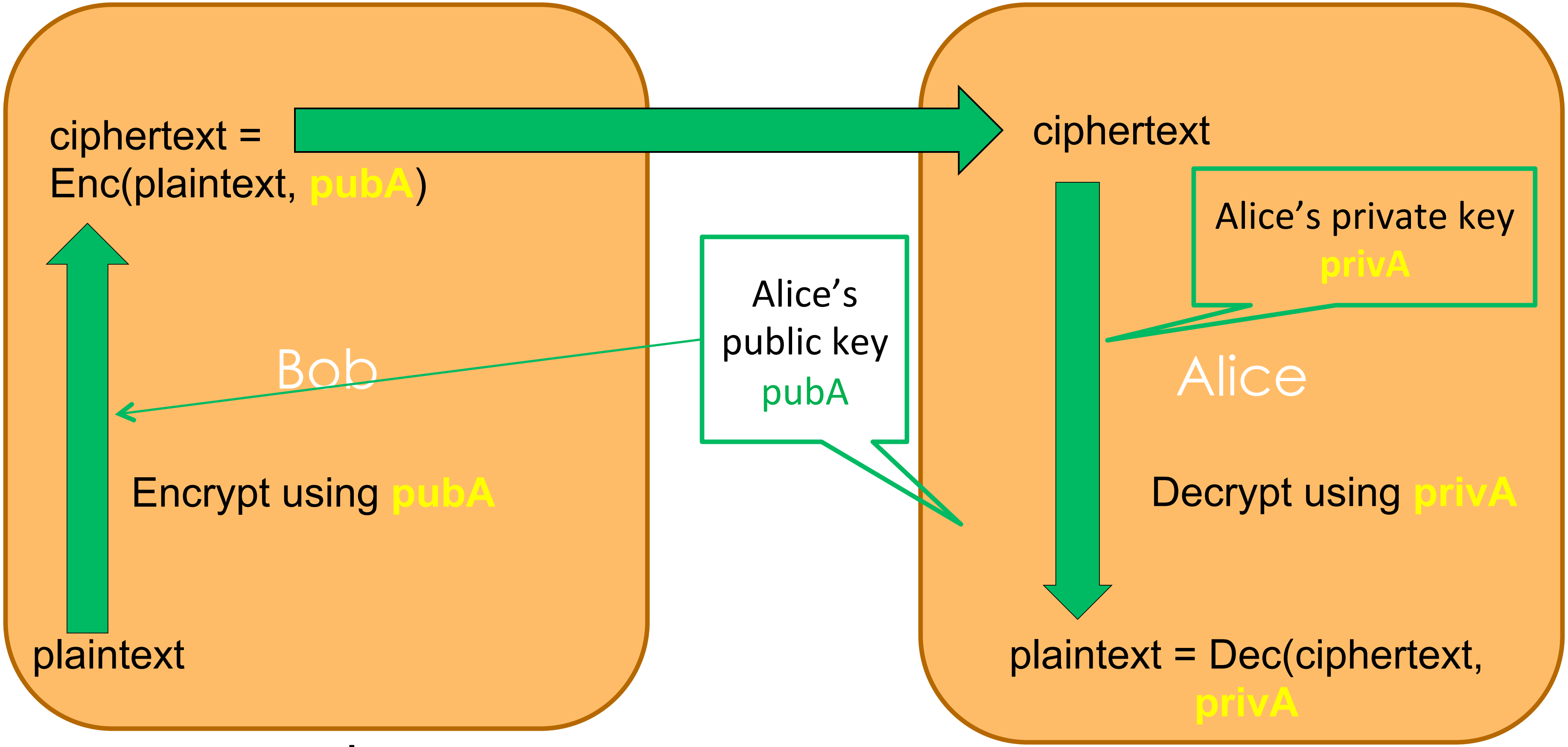
- **Symmetric** (shared-key between sender and receiver) encryption: commonly used for long messages
  - Often a complicated mix of substitution and transposition encipherment
  - Reasonably fast to compute
  - Examples (Caesar Cipher)
  - Requires a shared secret key usually communicated using (slower) *asymmetric encryption*

- **Asymmetric** encryption (two keys): different keys are used to encrypt and to decrypt
  - Public key: available to everyone, used to encrypt
  - Private key: available only to receiver, used to decrypt
  - Anyone with the public key can encrypt, only the private key can be used to decrypt!

One type of asymmetric encryption: RSA

- Common encryption technique for transmitting symmetric keys on the Internet (https, ssl/tls)
  - Named after its inventors: **R**ivest, **S**hamir and **A**dleman
  - Used in https (you know when you're using it because you see the URL in the address bar begins with http**s**://)

Asymmetric Public Key Encryption



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

Alice uses herprivate key to decrypt the ciphertext to recover the plaintext

How RSA works

- First, we must be able to represent any message as a single number (it may already be a number as is usual for a symmetric key)
- 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

Sender encrypts using public key

•Bob is sender/transmitter

Receiver decrypts using private key

- Alice is the receiver

Public and Private Keys

Sender encrypts using public key

- Bob is sender/transmitter
- Every sender has the receiver's public key:
    - public key (e, n)

Receiver decrypts using private key

- Alice is the receiver
- Every receiver has a:
    - public key (e, n)
    - private key (d, n)

Public and Private Keys

Sender encrypts using public key

- Bob is sender/transmitter
- Every sender has the receiver's public key:
  - public key (e, n)
- Encryption: The sender **encrypts** a (numerical) message M into ciphertext C using the receiver's **public key**:
  - M^e modulo $n \rightarrow C$ *(ciphertext)*

Receiver decrypts using private key

- Alice is the receiver
- Every receiver has a:
  - public key (e, n)
  - private key (d, n)

Public and Private Keys

Sender encrypts using public key

- Bob is sender/transmitter
- Every sender has the receiver's public key:
  - public key (e, n)
- Encryption: The sender **encrypts** a (numerical) message M into ciphertext C using the receiver's **public key**:
  - M^e modulo n $\rightarrow C$ *(ciphertext)*

Receiver decrypts using private key

- Alice is the receiver
- Every receiver has a:
  - public key (e, n)
  - private key (d, n)
- Decryption: The receiver **decodes** the encrypted message C to get the original message M using the **private key** (which no one else knows).
  - C^d modulo n $\rightarrow M$ *(plaintext)*

RSA Example

Sender encrypts using public key

- Bob is sender/transmitter
- Every sender has the receiver's public key:
  - public key (e, n)
- Encryption: The sender **encrypts** a (numerical) message M into ciphertext C using the receiver's **public key**:
  - M^e modulo $n \rightarrow C$ *(ciphertext)*

Receiver decrypts using private key

- Alice is the receiver
- Every receiver has a:
  - public key (e, n)
  - private key (d, n)
- Decryption: The receiver **decodes** the encrypted message C to get the original message M using the **private key** (which no one else knows).
  - C^d modulo $n \rightarrow M$ *(plaintext)*

- Bob wants to send Alice the message M=**4**

RSA Example

Sender encrypts using public key

- Bob is sender/transmitter
- Every sender has the receiver's public key:
  - public key (e, n)
- Encryption: The sender **encrypts** a (numerical) message M into ciphertext C using the receiver's **public key**:
  - M^e modulo n $\rightarrow C$ *(ciphertext)*

Receiver decrypts using private key

- Alice is the receiver
- Every receiver has a:
  - public key (e, n)
  - private key (d, n)
- Decryption: The receiver **decodes** the encrypted message C to get the original message M using the **private key** (which no one else knows).
  - C^d modulo n $\rightarrow M$ *(plaintext)*

- Bob wants to send Alice the message M=**4**
- Bob knows Alice's Public Key: (3, 33) (e = 3, n = 33)

- Alice's Public Key: (3, 33) (e = 3, n = 33)
- Alice's Private Key: (7, 33) (d = 7, n = 33)
  - Usually these are really huge numbers with many hundreds of digits!

RSA Example

Sender encrypts using public key

•Bob is sender/transmitter
•Every sender has the receiver's public key:

- public key (*e, n*)

•Encryption: The sender **encrypts** a (numerical) message *M* into ciphertext *C* using the receiver's **public key**:

- M^e modulo n → C *(ciphertext)*

Receiver decrypts using private key

- Alice is the receiver
- Every receiver has a:
  - public key (*e, n*)
  - private key (*d, n*)
- Decryption: The receiver **decodes** the encrypted message *C* to get the original message *M* using the **private key** (which no one else knows).
  - C^d modulo n → M *(plaintext)*

- Bob wants to send Alice the message M=**4**
- Bob knows Alice's Public Key: (3, 33) (e = 3, n = 33)
- Bob encrypts the message using *e* and *n*
  - (M^e modulo n → C):
  - 4^3 modulo 33 → 31
  - ... Bob sends **31**

- Alice's Public Key: (3, 33) (e = 3, n = 33)
- Alice's Private Key: (7, 33) (d = 7, n = 33)
  - Usually these are really huge numbers with many hundreds of digits!

RSA Example

Sender encrypts using public key

•Bob is sender/transmitter
•Every sender has the receiver's public key:

- public key (*e, n*)

•Encryption: The sender **encrypts** a (numerical) message *M* into ciphertext *C* using the receiver's **public key**:

- M^e modulo *n* → *C* *(ciphertext)*

Receiver decrypts using private key

- Alice is the receiver
- Every receiver has a:
  - public key (*e, n*)
  - private key (*d, n*)
- Decryption: The receiver **decodes** the encrypted message *C* to get the original message *M* using the **private key** (which no one else knows).
  - C^d modulo *n* → *M* *(plaintext)*

- Bob wants to send Alice the message M=**4**
- Bob knows Alice's Public Key: (3, 33) (e = 3, n = 33)
- Bob encrypts the message using *e* and *n*
  - *(M^e modulo n → C)*:
  - 4^3 modulo 33 → 31
  - ... Bob sends **31**

- Alice's Public Key: (3, 33) (e = 3, n = 33)
- Alice's Private Key: (7, 33) (d = 7, n = 33)
  - Usually these are really huge numbers with many hundreds of digits!
- Alice receives the encoded message **31**
  - Alice decrypts the message using d and n
  - (C^d modulo *n* → *M*):
  - 31^7 modulo 33 → **4**

Generating n, e and d

- p and q are (big) random primes.
- $n = p \times q$
- $\varphi = (p - 1)(q - 1)$
- e is small and relatively prime to φ
- d, such that: $e \times d \mod \varphi = 1$

$p = 3, q = 11$

$n = 3 \times 11 = 33$

$\varphi = 2 \times 10 = 20$

$e = 3$

$3 \times d \mod 20 = 1$

$d = 7$

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

Cracking RSA

Every sender has the receiver's public key:

- public key (e, n)

Every receiver has a:

- public key (e, n)
- private key (d, n)

- Everyone knows (e, n). Only Alice knows d.
- If we know e and n, can we figure out d?
  - If so, we can read secret messages to Alice.
- 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.
- … to factor a 15-digit number would take 4 months.
- … 20-digit number … 30,000 years.
- … 25-digit number… 3 billion years.
- We're safe with RSA! (at least, from factoring with digital computers)

Certificate Authorities

- **How do we know we have the right public key for someone?**
- *Certificate Authorities* sign digital certificates indicating authenticity of a sender who they have checked out in the real world.
- Senders provide copies of their certificates along with their message or software.
- But can we trust the certificate authorities? (only some)

Encryption is not security!

It’s just a set of techniques

How (in)secure is the Internet?

- The NSA has a budget of $11B; we know from Edward Snowden how some of it is used
- Corporations and criminals also spy on us
- What can go wrong?
  - Insecure pseudo-random number generators
  - Untrustworthy certificate authorities
  - Malware
  - “Social engineering” attacks like phishing
  - Deliberately built-in insecurity in crypto products
  - Physical tapping of Internet routers

Security is an unsolved problem

Your cyber systems continue to function and serve you not due to the expertise of your security staff but solely due to the sufferance of your opponents.

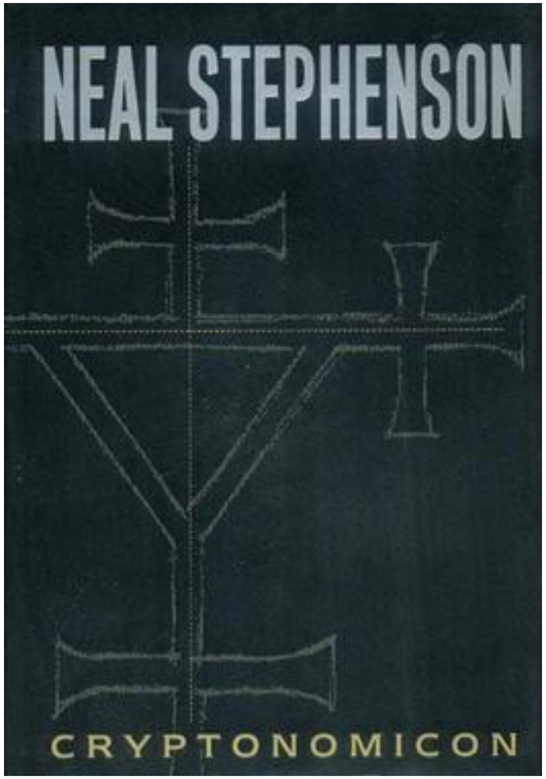
– former NSA Information Assurance Director Brian Snow (quoted by Bruce Schneier, https://www.schneier.com/blog/archives/2013/03/phishing_has_go.html)

Summary

- Cryptography is cool mathematics and protocol design
- But cryptography is not security, only a set of techniques
- Security is a broader issue involving
  - Other technology
  - Social and legal factors

"Only amateurs attack machines; professionals target people" –Bruce Schneier

Two closing thoughts



Use Signal…