CRYPTOGRAPHY

Masonry for software security

WHAT IS IT

- Cryptography: code-making / secret keeping
- Cryptanalysis: code-breaking / secret revealing

WHAT ISN'T IT

masonry is to **castle defense**

what

cryptography is to **software security**

OUTLINE

- **Randomness**
- **Hashing**
- **Encryption**
  - Caesar shift cipher
  - One time pad
- **Asymmetric-key (RSA)**
- **Diving deeper**

WORKSHOP OUTLINE

- Randomness*
- Hashing*
- Encryption*

** "don't try this at home"*

RANDOMNESS

"Any old bunch of crap that is thrown together."

–URBANDICTIONARY

RANDOMNESS

- “unpredictable”
- uniformly distributed output
- next is unrelated to previous
- next is unrelated to previous previous
- etc.

RANDOM NUMBER GENERATOR (RNG)

- **because sometimes we need a random number**
- **for cryptography!**
- **improving algorithms**
  - quick sort is better with a random pivot
  - parallel programming, avoiding deadlocks and “starvation”
- **for simulations / games**
- **just for kicks**

WHICH SEQUENCE IS MORE LIKELY?

1 Coin Toss

0 1

½ = ½

SAME

WHICH SEQUENCE IS MORE LIKELY?

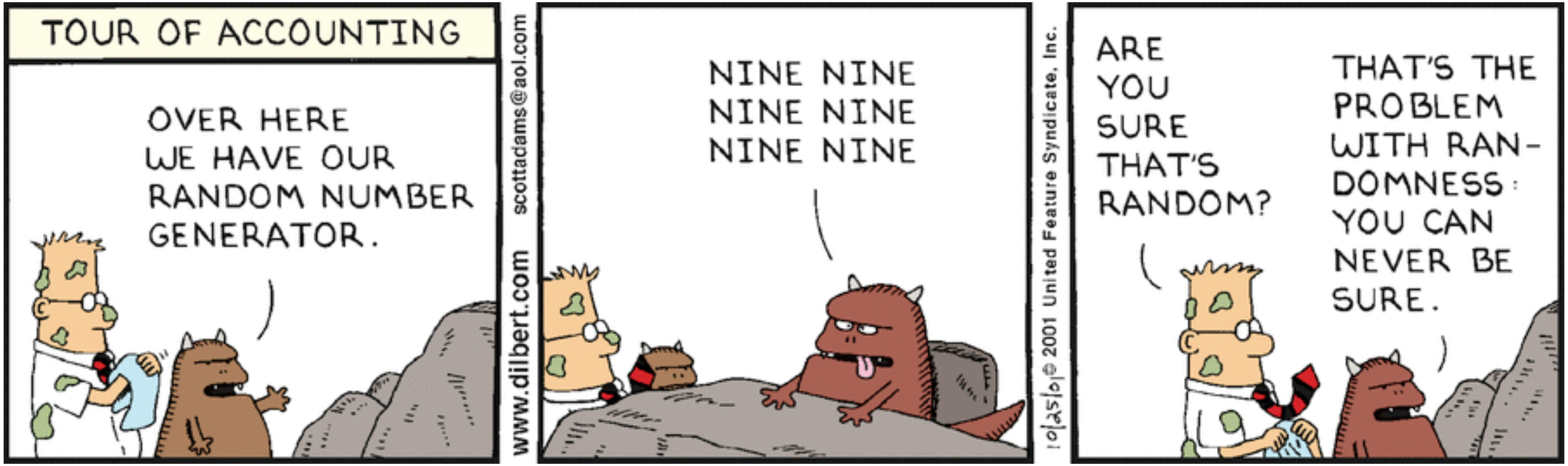
10 Coin Tosses

0000000000

1010100111

$\frac{1}{2}^{10} = \frac{1}{2}^{10}$

SAME



[HTTP://DILBERT.COM/STRIP/2001-10-25](http://dilbert.com/strip/2001-10-25)

ACTUAL RNG

- quantum mechanics*
- thermal noise
- 10th character of most recent tweet?
- “subjective”

** barring changes in our understanding of physics*

PSEUDO-RNG (PRNG)

- easier and also deterministic (repeatable)
- you pick and choose important properties of randomness
- “die hard” tests
- PRNG < cryptographically secure PRNG (CSPRNG)

PRNGS

- involve a seed
- middle squares method
- Mersenne twister

CSPRNG

- **for example, generating a random secret**
- **pass statistical tests (i.e. are good PRNGs)**
- **but also two more things**
  - satisfies “next-bit test”
  - withstands “state compromise extensions”

HASHING ALGORITHMS

- message (string) => digest (string)
- digest is fixed size
- same message always produces same digest
- irreversible!
- problem: collisions

EXAMPLES

- +*
- sha1
- md5
- neurons

**not a good one, but still*

HASHING: WHY

- **Identity**
  - hash tables, bloom filters
  - git commit ids
  - equality testing
- **Cryptography**
  - password verification
  - integrity verification
  - some PRNGs
  - proof-of-work (e.g. in cryptocurrency)

HASHING: HOW

1. pad
2. partition
3. combine

AKA “Merkle–Damgård construction”

"SIMPLE HASH"*

1. **pad** string to *at least* twice the desired length
2. **partition** into chunks of desired length
3. **combine** (*reduce*) these partitions using XOR

** read: "not cryptographically secure"*

ASIDE: XOR

AND

0	0	0
0	1	0
1	0	0
1	1	1

OR

0	0	0
0	1	1
1	0	1
1	1	1

XOR

0	0	0
0	1	1
1	0	1
1	1	0

Evenly Distributed

CRYPTOGRAPHIC HASHING

- Pre-image resistance: can’t find collision given hash result
- Second pre-image resistance: can’t find collision given input message
- Collision resistance: can’t find some arbitrary collision

HMAC

- key + message => digest
- key is a secret
- can be used to confirm message receipt without exposing the message or the identity of the receiver
- seems like it'd only be useful for James Bond
- actually useful for you

PBKDF2

- Password-based key derivation function 2
- Used on a password
- Call some hashing algorithm *repeatedly* on “salted” text
- Salt = secret = key
- I.e. it’s just a repetitive HMAC

PBKDF2...BUT WHY?

- Why hash passwords at all?
- Why salt our hashes?
- Why *repeatedly* hash?
- Another answer to "why": avoid $5 million lawsuit

HTTP://SECURITY.STACKEXCHANGE.COM/A/63421

WORKSHOP PARTS I & II

- Generate random strings
- Simple (but BAD) hashing algorithm
- HMAC
- pbkdf2

ENCRYPTION

- string <=string=> string
- plaintext <=key=> ciphertext
- reversible!

ENCRYPTION: WHY

- Secure communication
- Secure storage

ENCRYPTION: HOW

- Language
- Caesar shift
- Vignere cipher
- One time pad
- DES
- AES
- …many more

CAESAR SHIFT: HOW

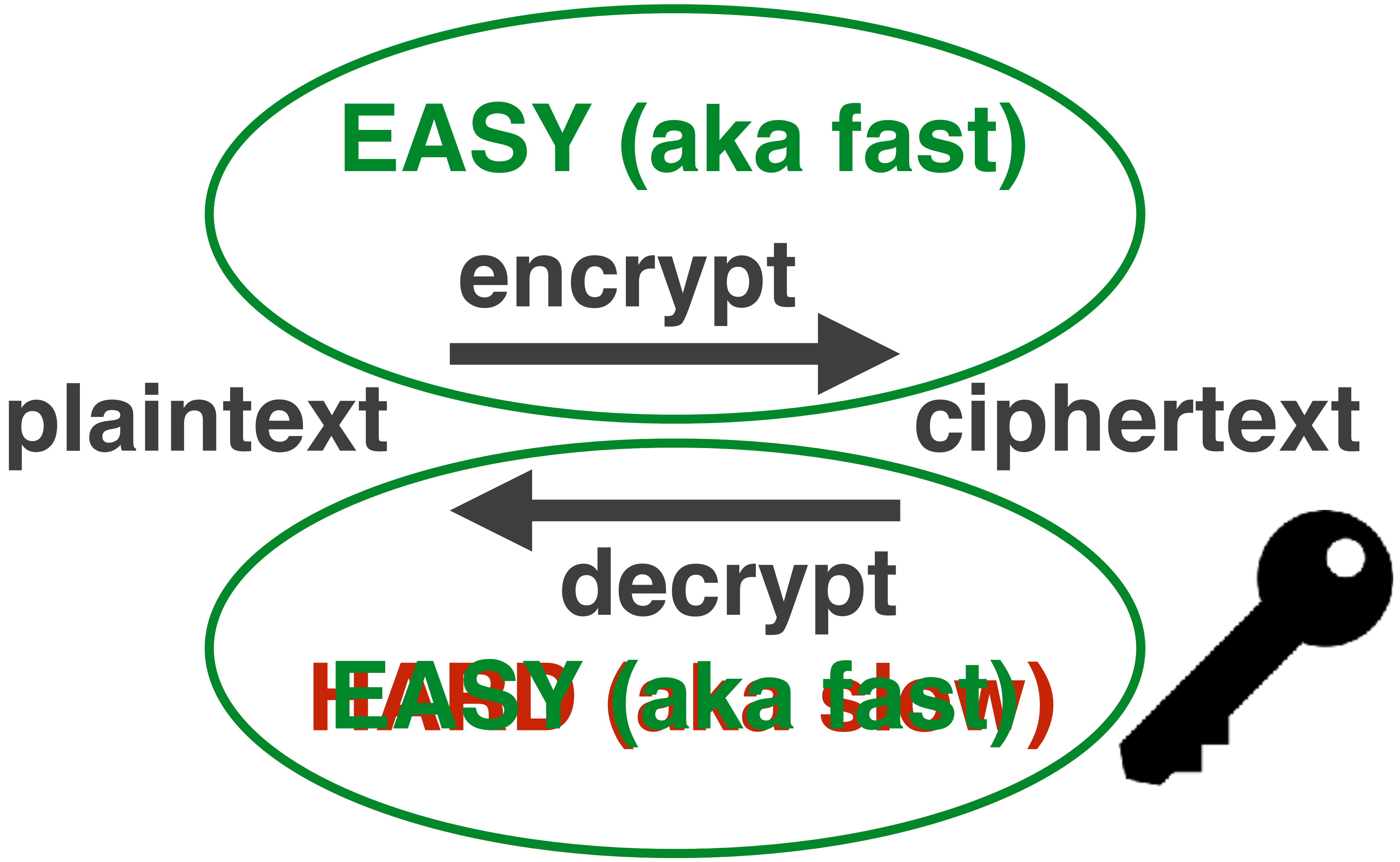
- Shift alphabet by some number
- Replace every character with corresponding shift
- Quiz: what’s the key?
- Quiz: why is this not great (cryptography)?

ONE TIME PAD: HOW

- Generate random key of *same length* as message
- XOR corresponding characters
- Never use that key for another message

ASYMMETRIC-KEY CRYPTOGRAPHY

- plaintext =keyA=> ciphertext =keyB=> plaintext
- $\boxed{\text{“one way”}}$ function with a $\boxed{\text{trapdoor}}$



ASYMMETRIC: WHY

- solves key distribution problem
- can provide “digital signatures”

ASYMMETRIC: HOW

- RSA
- ElGamal
- Elliptic curve algorithms
- …many more

RSA: HOW

- TL;DR Math!

RSA: HOW

- Wouldn’t it be nice if we could do…
- encrypt: m^e mod (n) = c
- decrypt: c^d mod (n) = m

- Because that would be easy for encryption, given e and n
- And hard for decryption unless given d
- So e would be the public key
- And d would be the private key

RSA: HOW

- So we're looking for
- encrypt: $m^e \bmod (n) = c$
- decrypt: $c^d \bmod (n) = m$

- Or, simplified, we're looking for a system where:
- $m^{(e*d)} \bmod(n) = m$

RSA: HOW

- We're looking for a system where:
- $m^{(e*d)} \mod(n) = m$

- Oh wow, look at Euler's theorem:
- $a^{(\phi(n))} \mod(n) = 1$
- Which can be changed to:
- $a^{(1+k*\phi(n))} \mod(n) = a$

RSA: HOW

- phi(n): count the numbers less than n that share no factors with it
- e.g. phi(9)…
- Numbers less than 9: 8, 7, 6, 5, 4, 3, 2, 1
- That share no factors with 9: 8, 7, 5, 4, 2, 1
- That is 6 numbers so phi(9) = 6

RSA: HOW

- OK, so again, Euler's theorem:
- a^(1+k*phi(n)) mod(n) = a

- Our goal
- m^(e*d) mod(n) = m

- So in in order to achieve our goal:
- e*d = 1+k*phi(n)

RSA: HOW

- New goal:
- e*d = 1+k*phi(n)

- For example we choose some n, say 33
- phi(33) = 20
- We could then choose: e = 3, d = 7, k = 1
- Because: 3*7 = 1+1*20
- Public key = e = 3; private key = d = 7

RSA: HOW

- Remember…
- encrypt: m^e mod (n) = c
- decrypt: c^d mod (n) = m
- And e, d, and n need to fit…
- e*d = 1+k*phi(n)
- BUT d needs to be secret, e and n need to be public
- …which sounds problematic

RSA: HOW

- Wouldn't it be nice if for…
- e*d = 1+k*phi(n)
- …it would be easy for US to calculate e and d given n
- …but hard for OTHER PEOPLE
- Well calculating phi(n) given n is HARD in general
- …but easy if we know the prime factors of n
- So n should be the product of two large primes

RSA: HOW

- If n is the product of two primes, p and q
- Then phi(n) is just (p-1)*(q-1)
- Then choose any e and d that satisfy:
- e*d = 1+k*phi(n)
- …and throw away those primes

RSA: HOW

- Generate two large primes, multiply them (yields n)
- Find any e where e shares no prime factors with phi(n)
- Find d where e*d shares no prime factors with phi(n)
- e and n are the public key for encryption via:
- m^e mod (n) = c
- d and n are the private key for decryption via:
- c^d mod (n) = m

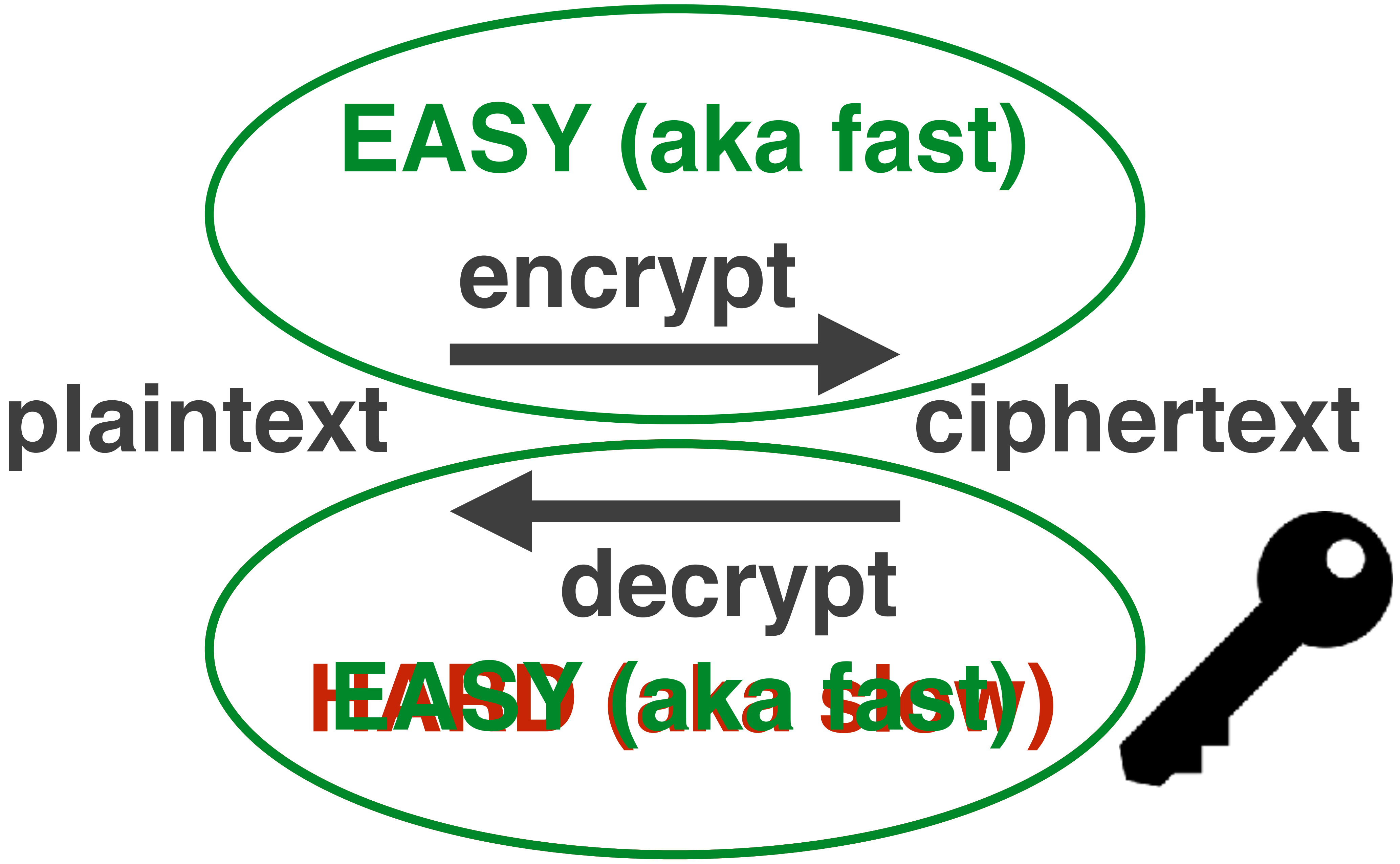
RSA: HOW

- Generating e and d, more
- Loop from e = 2, increment
- Find any e where e shares no prime factors with phi(n)
- Find d where e*d shares no prime factors with phi(n)
- e and n are the public key for encryption via:
- m^e mod (n) = c
- d and n are the private key for decryption via:
- c^d mod (n) = m

BACK TO THE BIG PICTURE, WE DID ALL OF THIS IN ORDER TO ACHIEVE…

ASYMMETRIC-KEY CRYPTOGRAPHY

- plaintext =keyA=> ciphertext =keyB=> plaintext
- “one way” function with a trapdoor



WORKSHOP PARTS III & IV

- Caesar shift
- One time pad
- RSA (!)