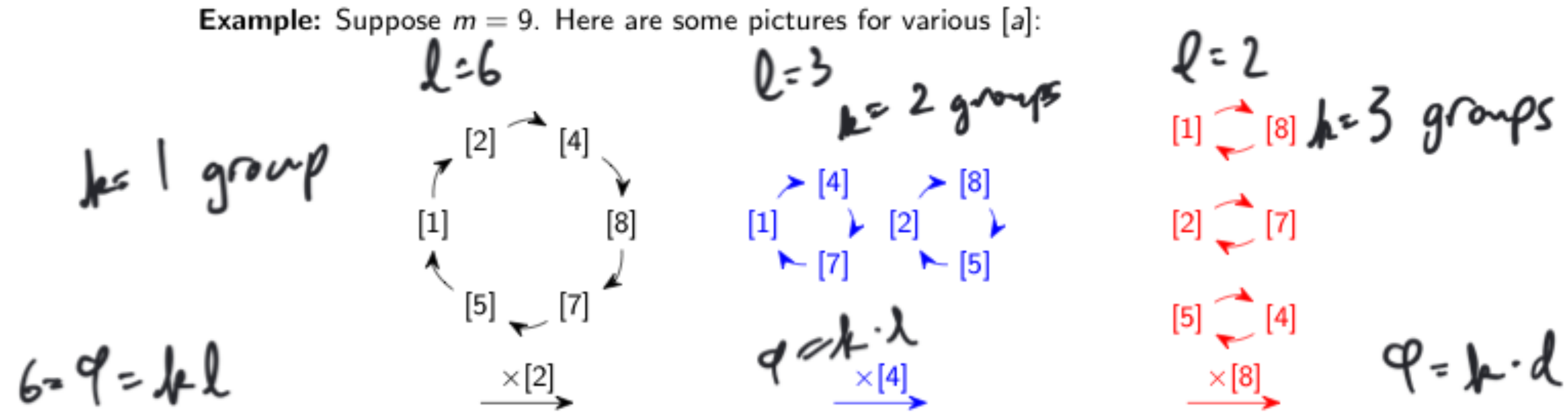
Lecture 18: Public key cryptography

Last time:

- **Claim (Euler's theorem):** if $[a]_m$ is a unit, then $[a]_m^{[b]_{\varphi(m)}} := [a^b]_m$ is well-defined
- We reduced this to $[a]_m^{\varphi(m)} = [1]_m$.

Proof: Choose an arbitrary m and unit $[a]_m$. We consider what happens to the elements of $\mathbb{Z}_m^*$ when we multiply by $[a]$.

Example: Suppose $m = 9$. Here are some pictures for various $[a]$:



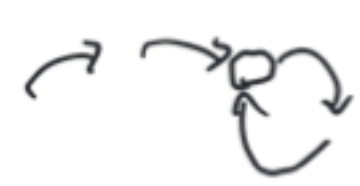
Back to proof: we'll always have the units divided into loops all same length.

$$[a^{-1}\cdot c] \to [c] \to [a\cdot c]$$

So units form "chains"

$$[1] \to [a] \to [a^2] \to [a^3] \to \cdots$$

Can't go forever without repeating, only finitely many units.



Can't happen.

$1 = a^l = a^0$

$$[1] \to [a] \to [a^2] \to [a^3] \to \cdots$$

for some l.

(the length of loop).

$$[b] \to [ab] \to [a^2 b] \to \cdots$$

$= [a^l b]$

starting at any $[b]$, taking l steps brings me back to $[b]$

So all loops are same length.

---

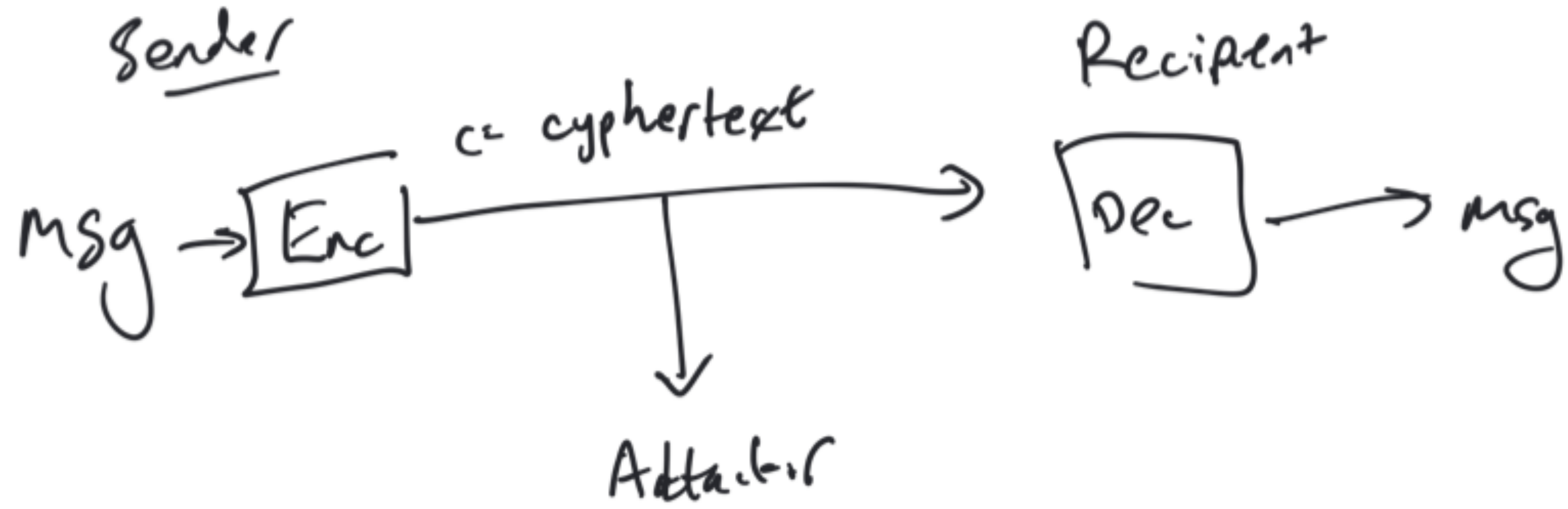
if there are k loops, each of length l, there are $k\cdot l$ total elements (i.e. units), so $\varphi(m) = k\cdot l$.

$$\text{So } [a]^{\varphi(m)} = [a]^{kl} = \Big(\underbrace{[a]^l}_{[1]}\Big)^k = [1]. \checkmark$$

$[1]$ because of defn of l.

Cryptography

Goal: the **sender** wants to send a message to the **recipient** without the **attacker** being able to decipher it.



Example: add 7 to each letter, wrapping around past 26

$a = 1 \xrightarrow{+7} 8 = g.$

$a \to g$
$b \to h$
$\vdots$

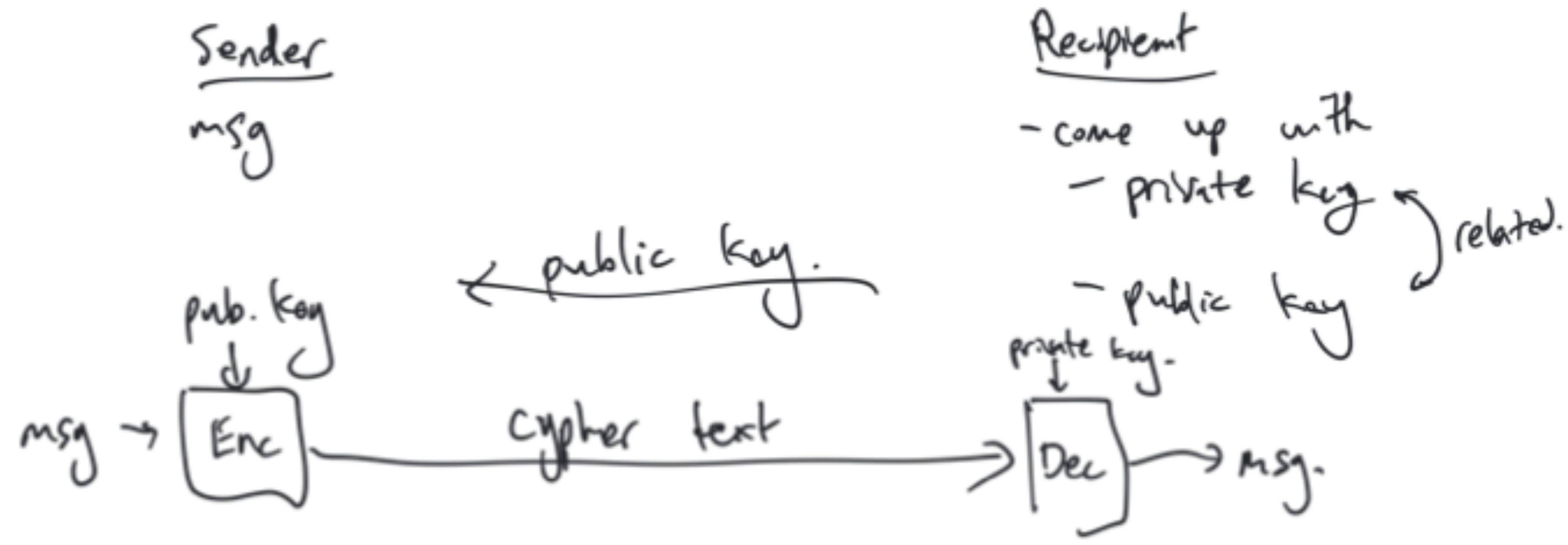
One time pad: pick a different # randomly for each letter, add it to character.

"hi"
$(3, 7)$ one-time pad
↓ ↓
"k m"

problem: have to tell the recipient what the pad is, so they can decrypt.

Public key cryptography

Key idea: there's a $\overset{\text{private key}}{\text{secret}}$ that only the recipient knows. Related "public key" that everyone knows



RSA cryptosystem.

Sender

$[msg]_m$

$\xleftarrow{\overset{\text{public key.}}{(m,k)}}$

$C = [msg]_m^k \longrightarrow$

Recipient

choose large primes p, q.

let $m = p \cdot q$. So $\varphi(m) = (p-1)(q-1)$.

idea: $\sqrt[k]{C}$ needs to find $[k]^{-1}_{\varphi(m)}$ ← need to know k, $\boxed{\varphi(m)}$ secret.

actually: $C^{[k]^{-1}_{\varphi(m)}}$

$= \left([msg]_m^{[k]_{\varphi(m)}}\right)^{[k]^{-1}_{\varphi(m)}}$

$= [msg]_m^{[k \cdot k^{-1}]_{\varphi(m)}} = [msg]^{[1]}$

- there's an efficient way to test to see if a # is prime.
- there's lots of primes: pick a large odd # n, check whether prime, if not, try $n+2$ and repeat, there's enough primes that you'll quickly find one.

Conjecture: if p, q are large, then it is difficult to factor m into p & q.

Naive algorithm: To factor m, divide by every prime # $2 \ldots \sqrt{m}$.

This algorithm takes exponential time in # digits of m.

Note: Nobody has been able to prove that there's no fast factoring algorithm.