2550 Intro to cybersecurity

L8: Crypto: PKC

abhi shelat

Basic Number theory

$$a \bmod p$$

$17 \bmod 11$

$135433238 \bmod 11$

Basic number theory

Modular arithmetic

Claim 28.1. *For* $n > 0$ *and* $a, b \in \mathbb{Z}$*,*

1. $(a \bmod n) + (b \bmod n) = (a + b) \bmod n$
2. $(a \bmod n)(b \bmod n) \bmod n = ab \bmod n$

Modular Exponentiation $(a, x, n) \to a^x \bmod n$

$$5^{19} \bmod 31$$

Modular Exponentiation

$$(a, x, n) \to a^x \bmod n$$

Algorithm 2: ModularExponentiation(a, x, n)

Input: $a, x \in [1, n]$

$r \leftarrow 1$
while $x > 0$ **do**
    if x *is odd* **then**
        $r \leftarrow r \cdot a \bmod n$
    $x \leftarrow \lfloor x/2 \rfloor$
    $a \leftarrow a^2 \bmod n$
Return r

Modular Exponentiation

$$(a, x, n) \to a^x \bmod n$$

$$a^x \bmod n = \prod_{i=0}^{\ell} x_i a^{2^i} \bmod n$$

Algorithm 2: ModularExponentiation(a, x, n)

Input: $a, x \in [1, n]$

$r \leftarrow 1$
while $x > 0$ **do**
  if x *is odd* **then**
    $r \leftarrow r \cdot a \bmod n$
  $x \leftarrow \lfloor x/2 \rfloor$
  $a \leftarrow a^2 \bmod n$
Return r

Greatest Common Divisor

```
GCD(A,B) = GCD(            )
```

Greatest Common Divisor

```
GCD(6809,1639)
```

given (a,b), finds (x,y) s.t.
ax + by = gcd(a,b)

Algorithm 1: ExtendedEuclid(a, b)

Input: (a, b) s.t $a > b \geq 0$
Output: (x, y) s.t. $ax + by = \gcd(a, b)$

```
if a mod b = 0 then
    Return (0, 1)
else
    (x, y) ← ExtendedEuclid (b, a mod b)
    Return (y, x − y(⌊a/b⌋))
```

groups

$$(G, \oplus)$$

closure

associativity

identity

inverse

example of groups

$$(\mathbb{Z}_n, +)$$

Example of groups

$$(\mathbb{Z}_n, \star)$$

$\{a \mid \gcd(a, n) = 1\}$

multiplicative group, mod n

$$\mathbb{Z}_n^\star$$

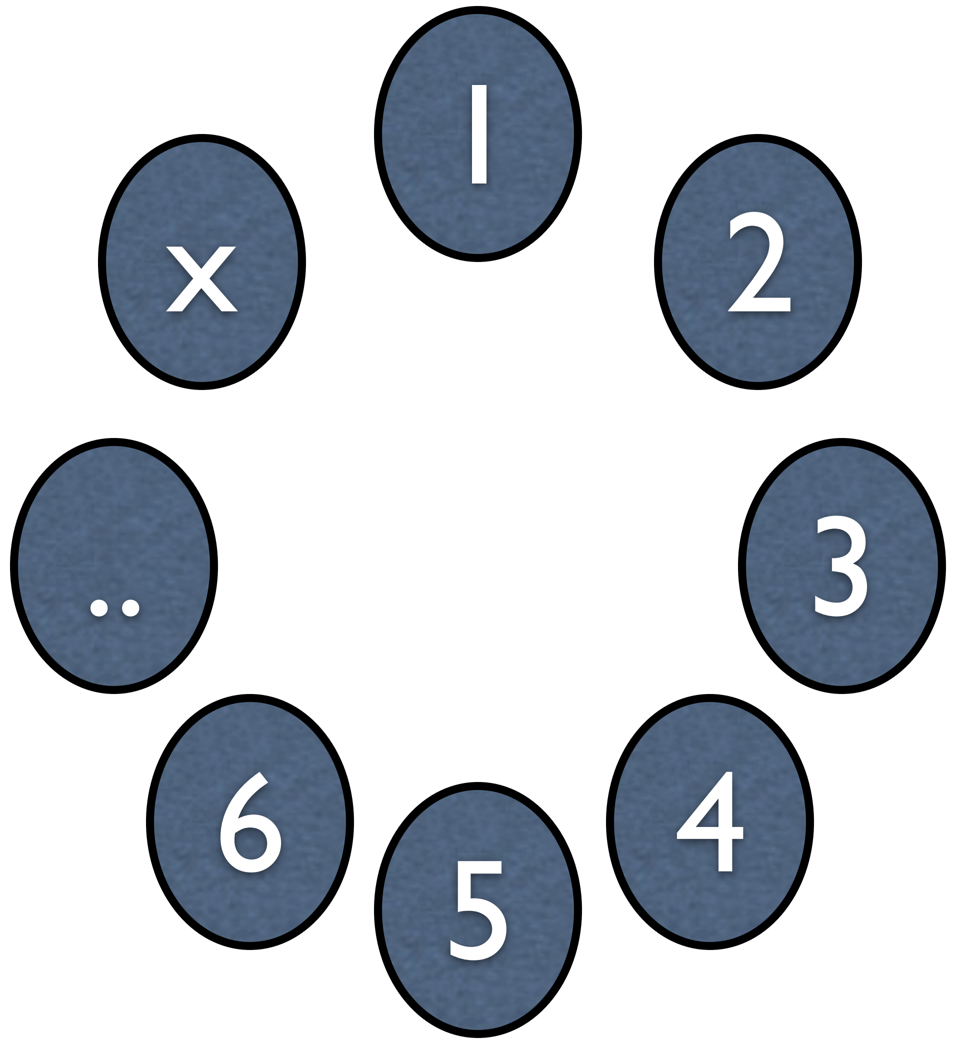
Euler totient

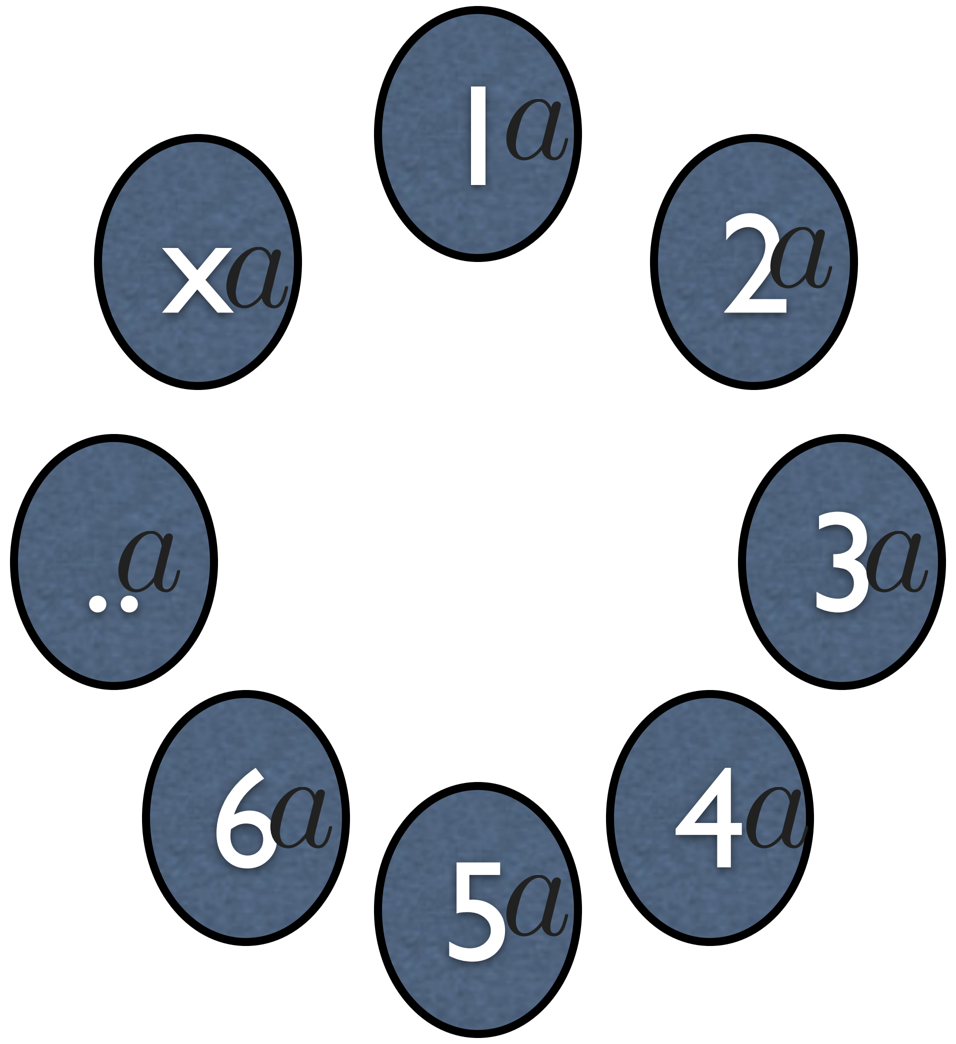
Euler theorem

$$\forall a \in \mathbb{Z}_n^\star,\ a^{\Phi(n)} = 1 \bmod n$$

Euler theorem

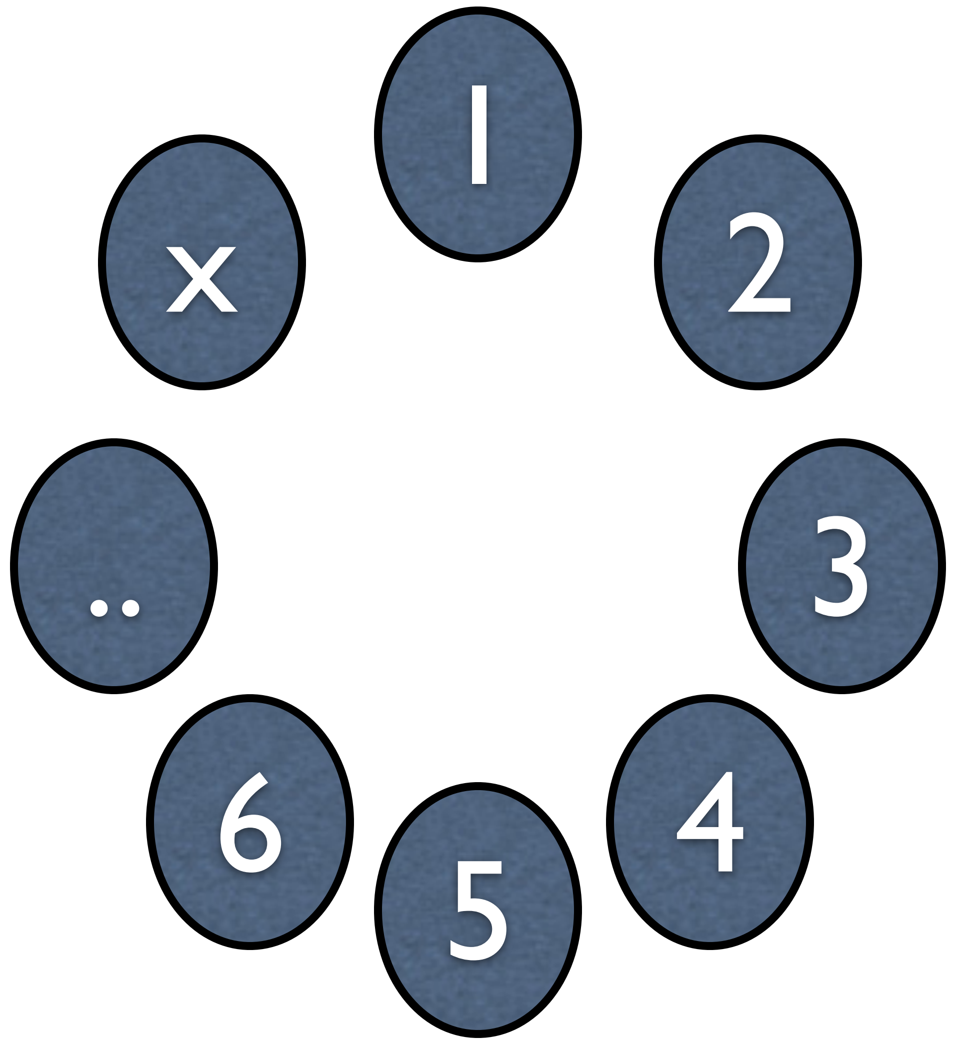
$$\forall a \in \mathbb{Z}_N^\star,\ a^{\Phi(N)} = 1 \bmod N$$

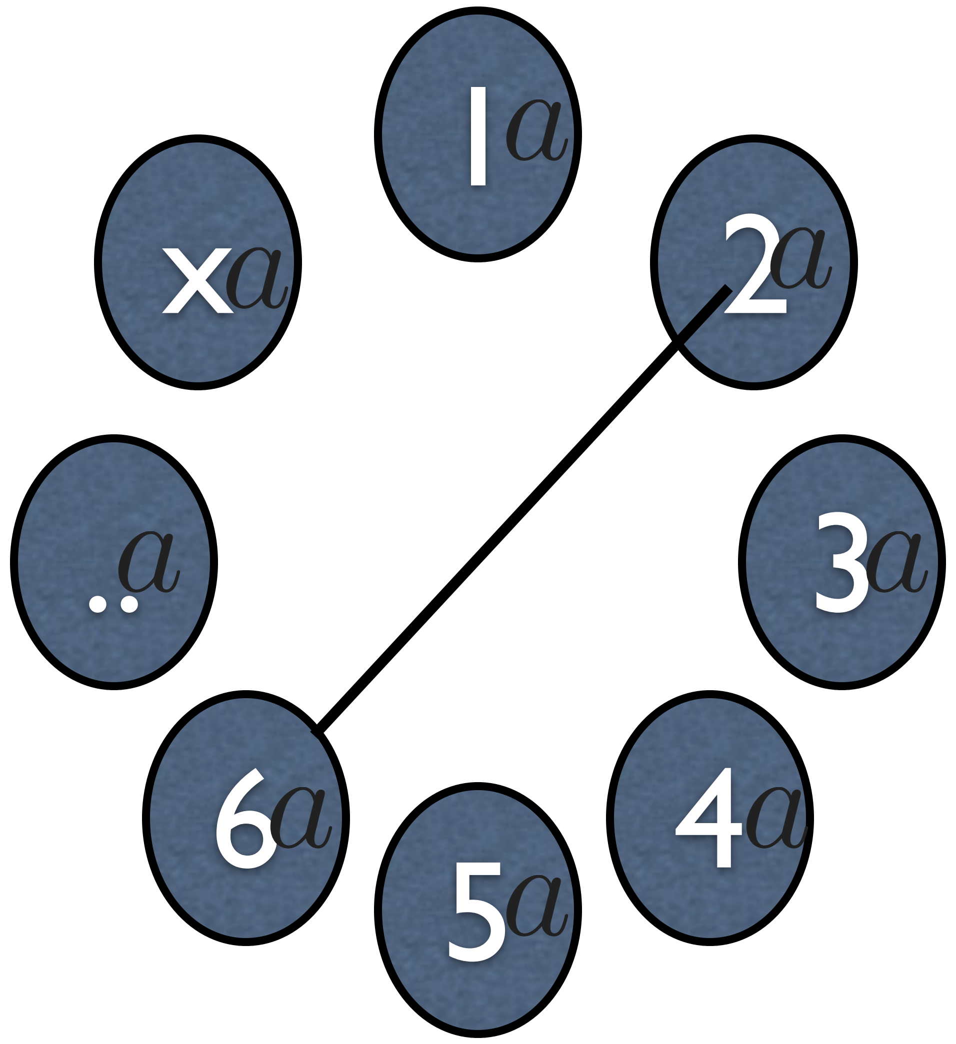




Euler theorem

$$\forall a \in \mathbb{Z}_N^\star,\ a^{\Phi(N)} = 1 \bmod N$$





argue: all are distinct

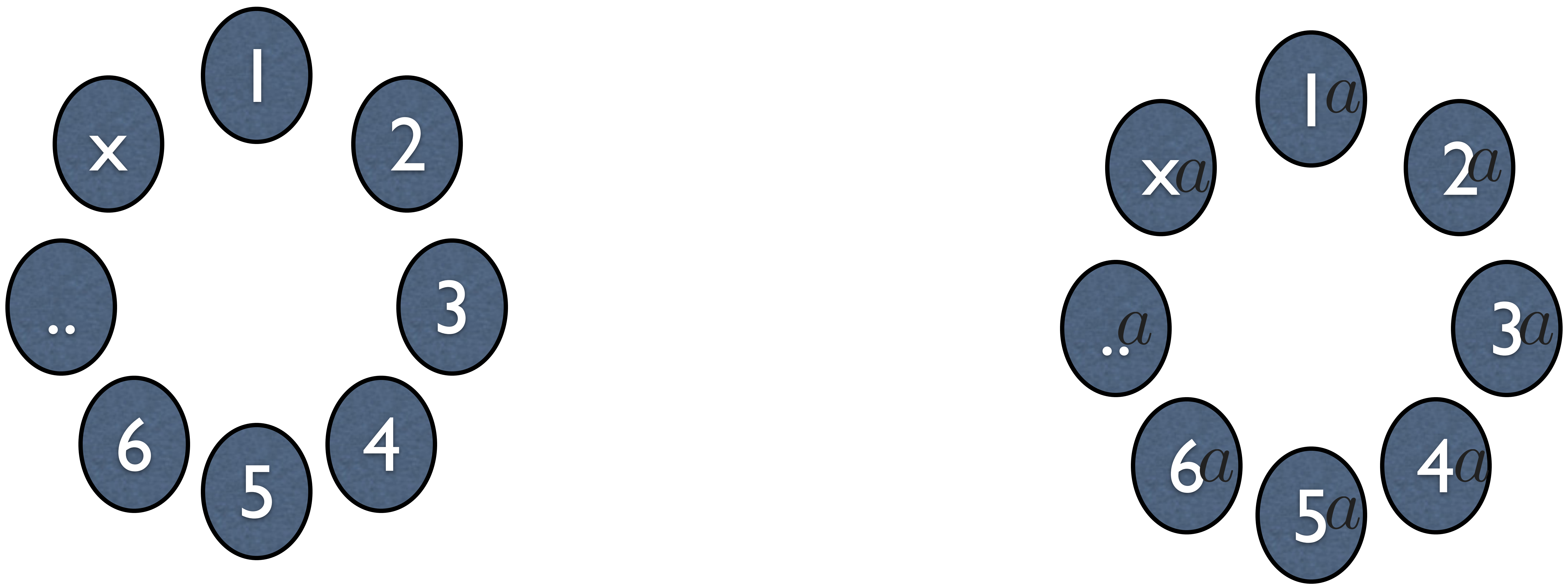
spse two are equal.

multiply by a^{-1}

this implies 2=6!

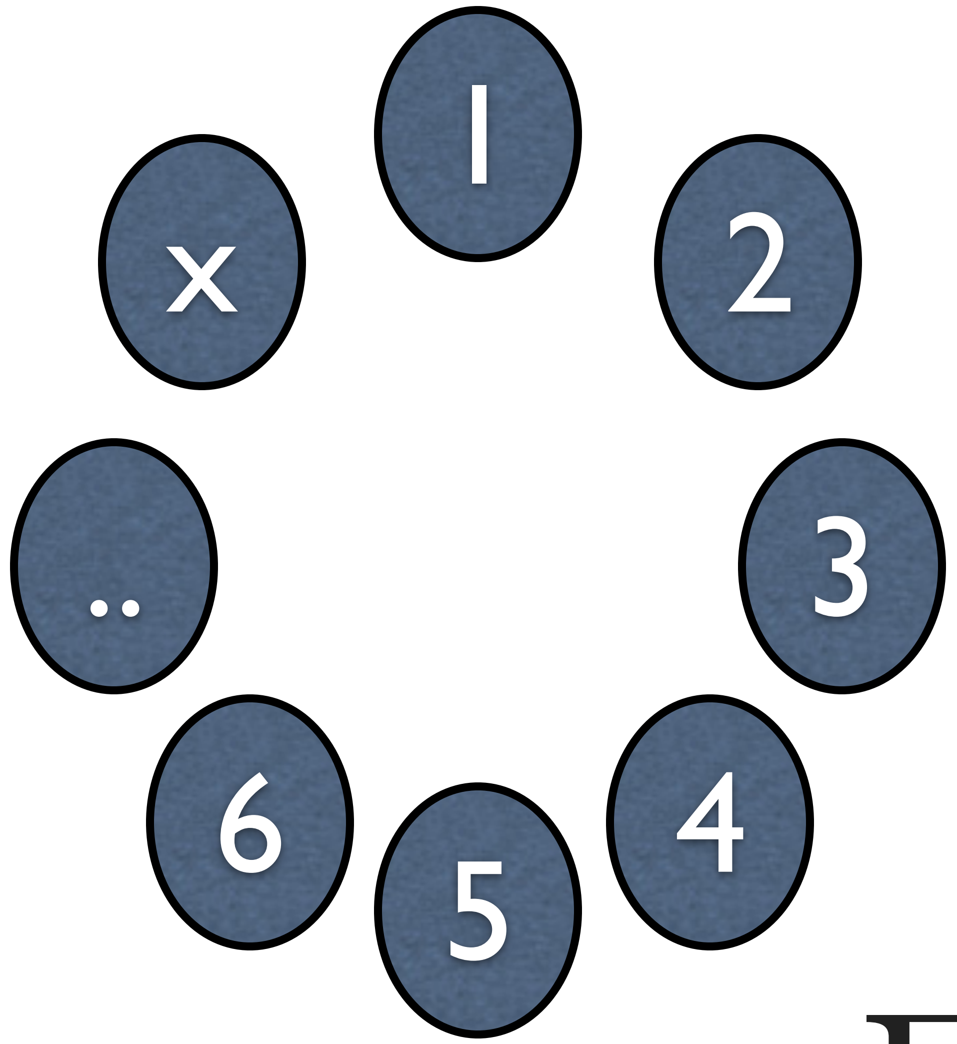
Euler theorem

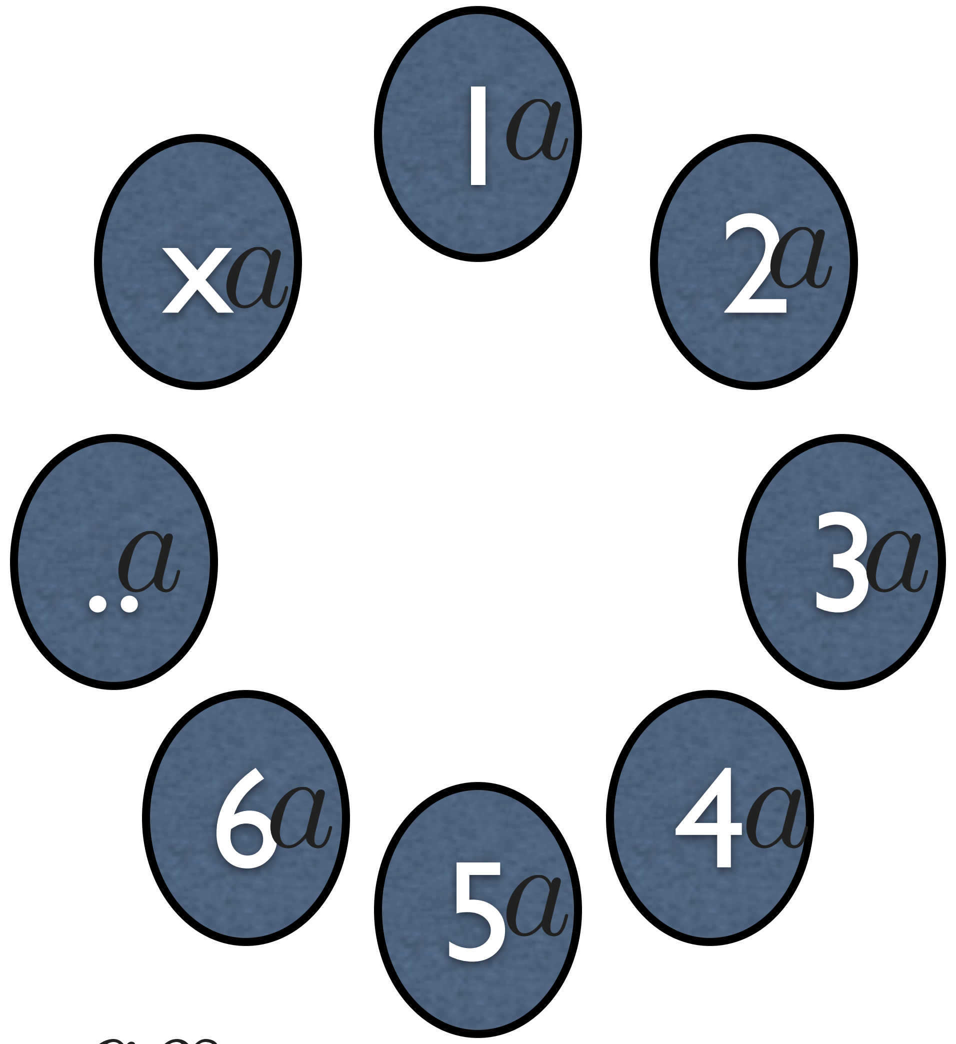
$$\forall a \in \mathbb{Z}_N^\star,\ a^{\Phi(N)} = 1 \bmod N$$



Euler theorem

$$\forall a \in \mathbb{Z}_N^\star,\ a^{\Phi(N)} = 1 \bmod N$$

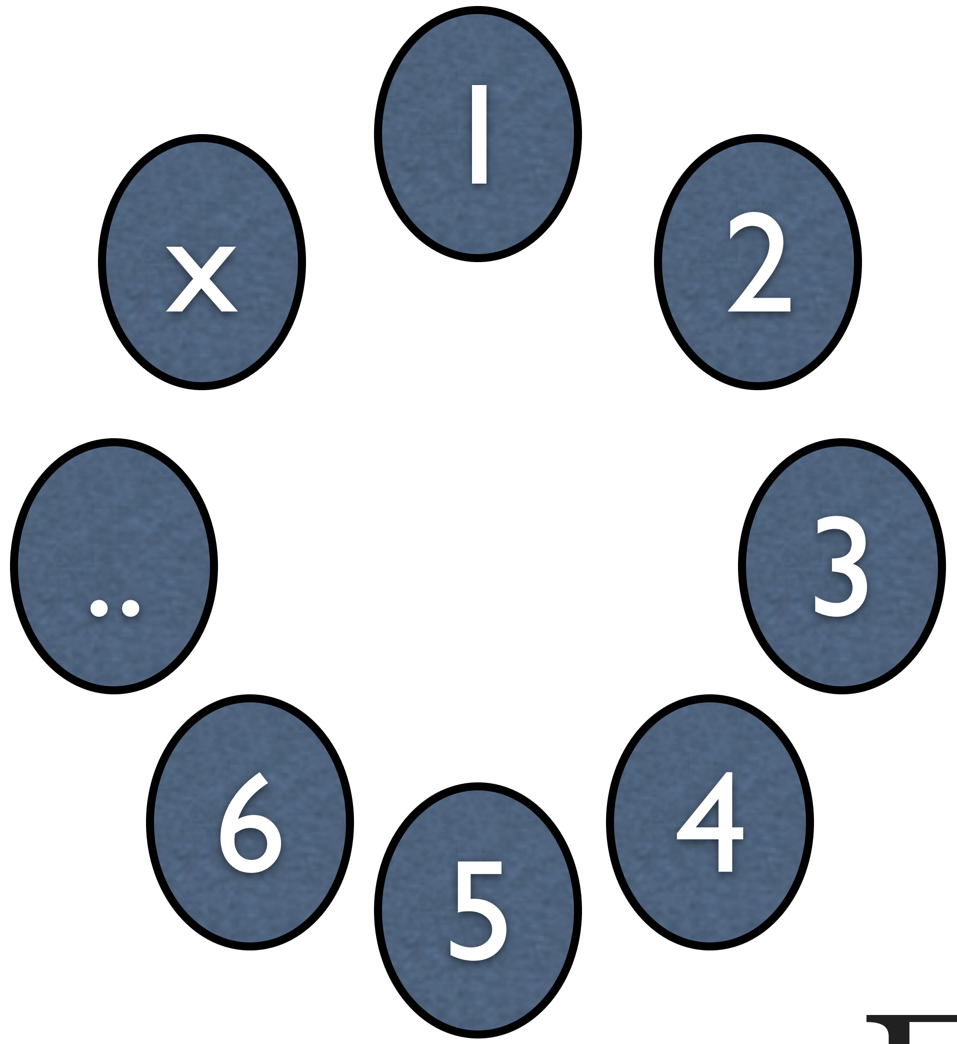


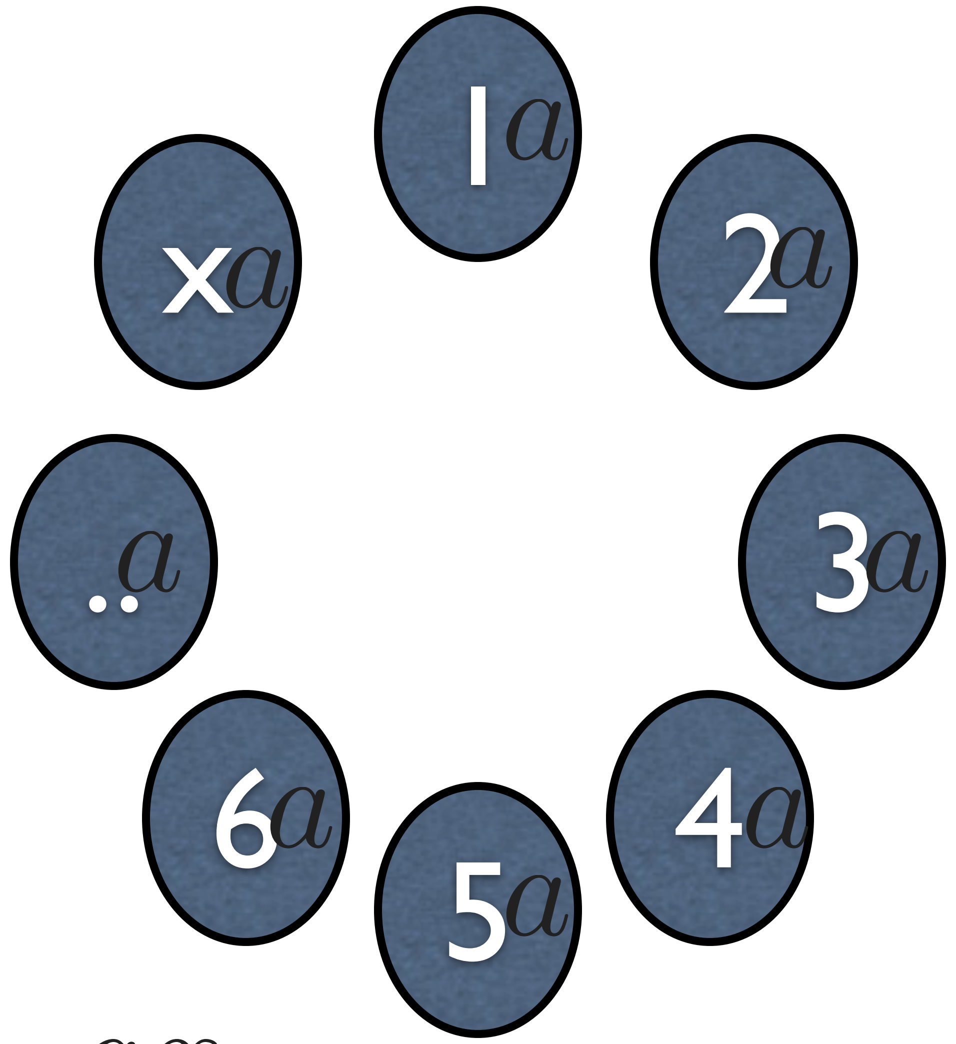


$$\prod_{x \in Z_N^\star} x \qquad \prod_{x \in Z_N^\star} ax$$

Euler theorem

$$\forall a \in \mathbb{Z}_N^\star,\ a^{\Phi(N)} = 1 \bmod N$$

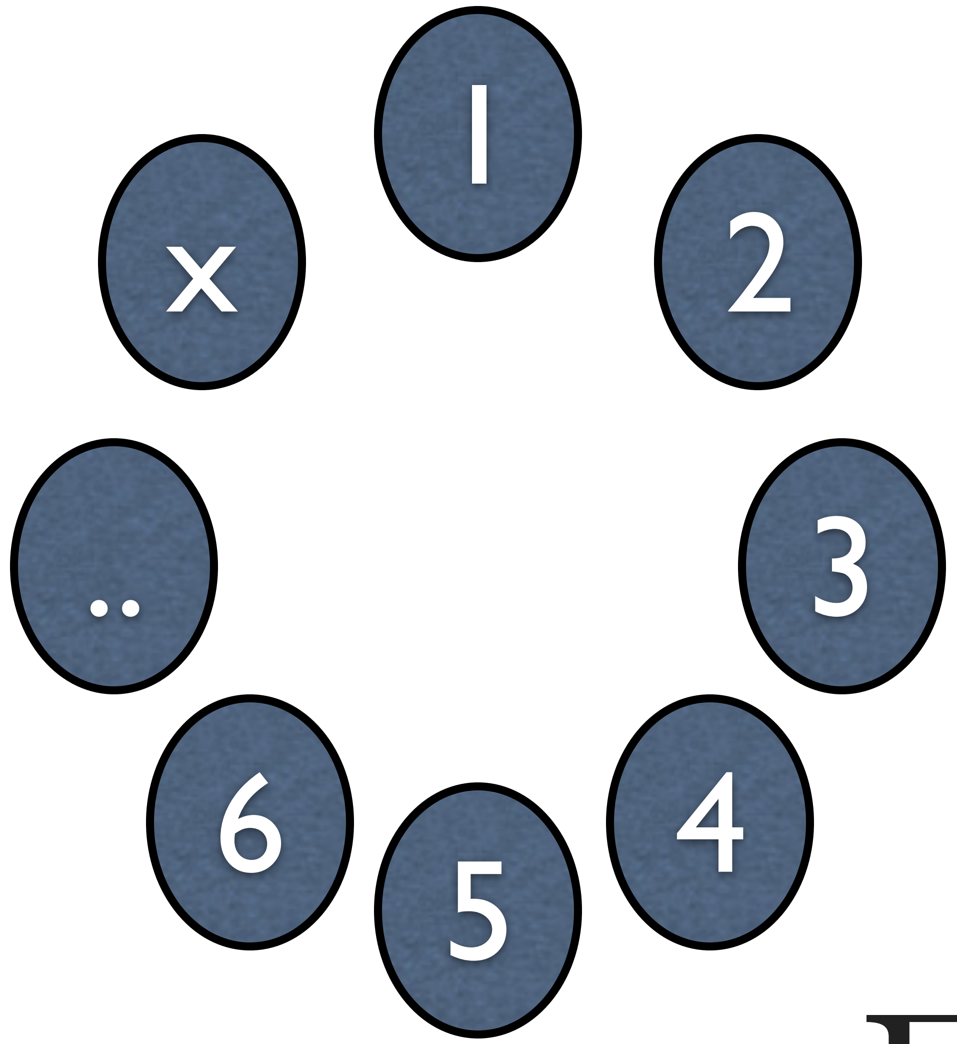


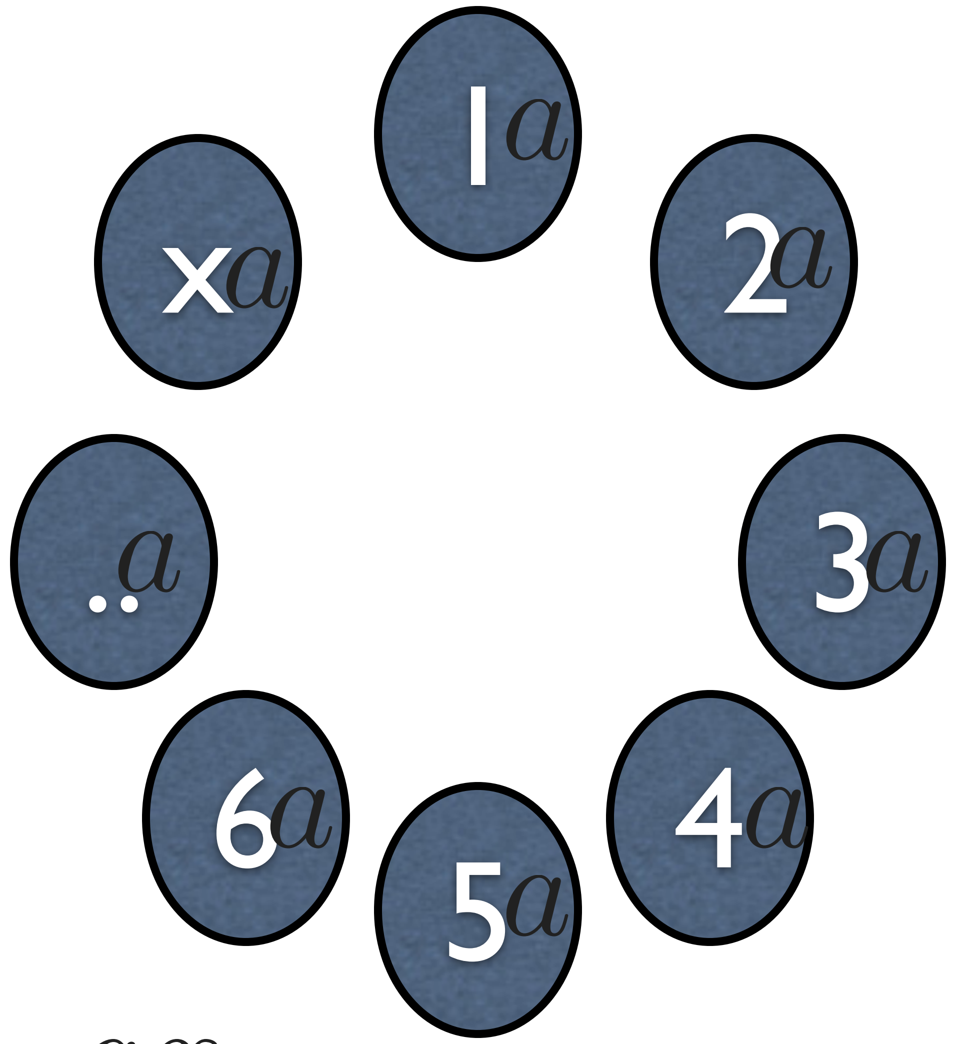


$$\prod_{x \in Z_N^\star} x = \prod_{x \in Z_N^\star} ax$$

Euler theorem

$$\forall a \in \mathbb{Z}_N^\star,\ a^{\Phi(N)} = 1 \bmod N$$



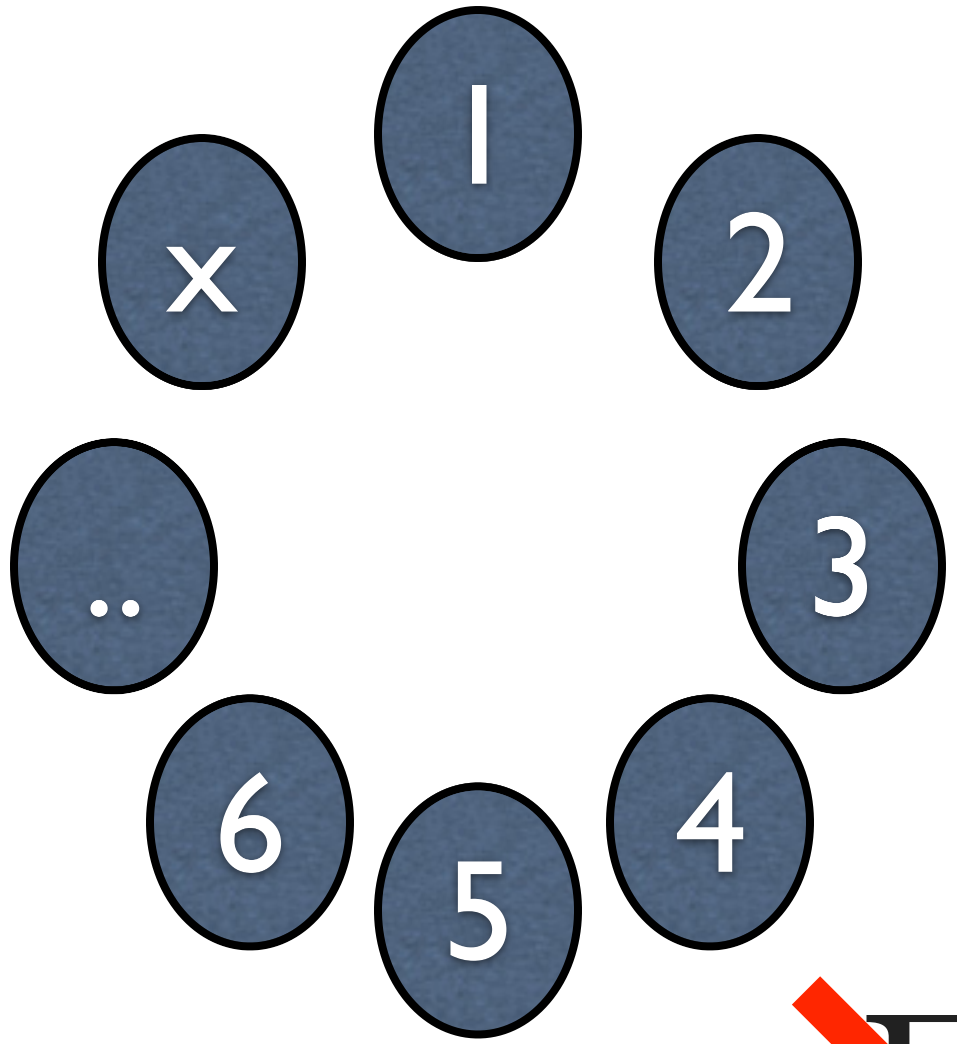


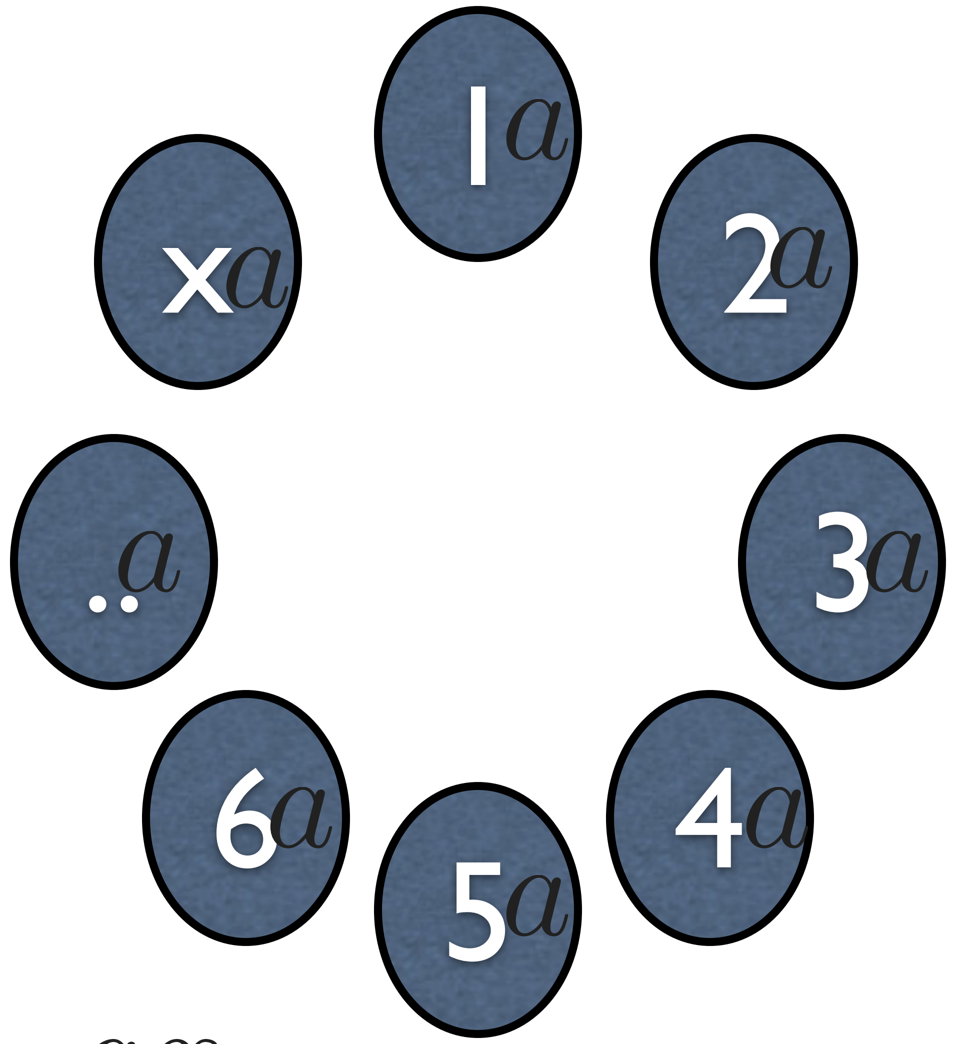
$$\prod_{x \in Z_N^\star} x = \prod_{x \in Z_N^\star} ax$$

$$a^{\Phi(N)} \prod_{x \in Z_N^\star} x$$

Euler theorem

$$\forall a \in \mathbb{Z}_N^\star,\ a^{\Phi(N)} = 1 \bmod N$$





$$\cancel{\prod_{x\in Z_N^\star}} x = \prod_{x\in Z_N^\star} ax$$

$$a^{\Phi(N)} \cancel{\prod_{x\in Z_N^\star}} x$$

compute

$$11^{31^{2020}} \bmod 23$$

(show your work)

El-Gamal encryption

gen(1^n)

$$p \leftarrow \Pi_n \quad g \leftarrow \text{Generators}_p$$

$\text{enc}_{pk}(m)$

$\text{dec}_{sk}(c)$

El-Gamal encryption

gen(1^n)

$$p \leftarrow \Pi_n \quad g \leftarrow \text{Generators}_p$$

$$a \leftarrow \mathbb{Z}_p$$

$$pk \leftarrow (g, g^a) \quad sk \leftarrow (g, a)$$

enc$_{pk}$(m)

$$r \leftarrow \mathbb{Z}_p$$

$$(g^r, (g^a)^r \cdot m)$$

dec$_{sk}$(c)

$$(c_1, c_2) \leftarrow c$$

$$m \leftarrow c_2/(c_1)^a$$

Example ElGamal

msg="   "

$pk \leftarrow (g, g^a) \quad sk \leftarrow (g, a)$

enc$_{pk}$(m)

$r \leftarrow \mathbb{Z}_p$

$c \leftarrow g^r, pk^r \cdot m$

dec$_{sk}$(c)

$(c_1, c_2) \leftarrow c$

$m \leftarrow c_2/(c_1)^a$

Why is ElGamal secure?

decisional Diffie-Hellman assumption (DDH)

$$p \leftarrow \Pi_n \quad g \leftarrow \text{Generators}_p$$

$$a, b, c \leftarrow \mathbb{Z}_p \quad \text{(work in a prime order group)}$$

$$\{p, g, g^a, g^b, g^{ab}\}_n \approx \{p, g, g^a, g^b, g^c\}_n$$

"Textbook" RSA (insecure)

Pick N = p*q where p,q are primes.

Pick e,d such that $e \cdot d = 1 \bmod \phi(N)$

$\mathrm{Enc}_{N,e}(m) = m^e \mod N$

$\mathrm{Dec}_{N,d}(c) = c^d \mod N$

$(m^e)^d \bmod N =$

“Textbook” RSA (insecure) Example

Pick N = p*q where p,q are primes.

N=11*13 = 143

Pick e,d such that $e \cdot d = 1 \mod \phi(N)$

$\text{Enc}_{N,e}(m) = m^e \mod N$

$\text{Dec}_{N,d}(c) = c^d \mod N$

"Textbook" RSA (insecure)

Pick N = p*q where p,q are primes.

Pick e,d such that $e \cdot d = 1 \bmod \phi(N)$

$\text{Enc}_{N,e}(m) = m^e \mod N$

$\text{Dec}_{N,d}(c) = c^d \mod N$

Why is it insecure against IND-CPA attack?

pkcs1.5

$\text{enc}_{pk}(m)$

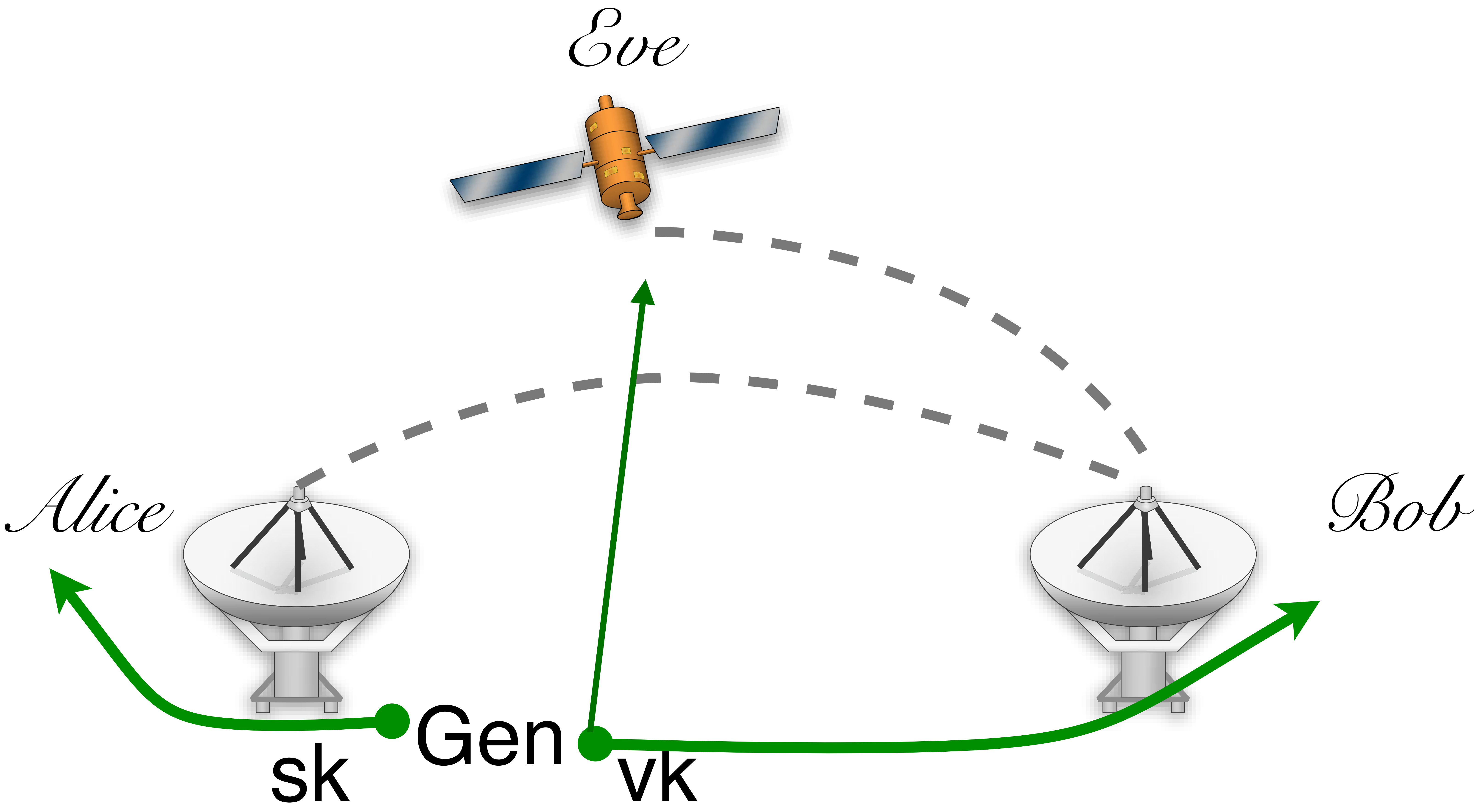
pick r as a random string with no 0s

(typically 8 bytes)

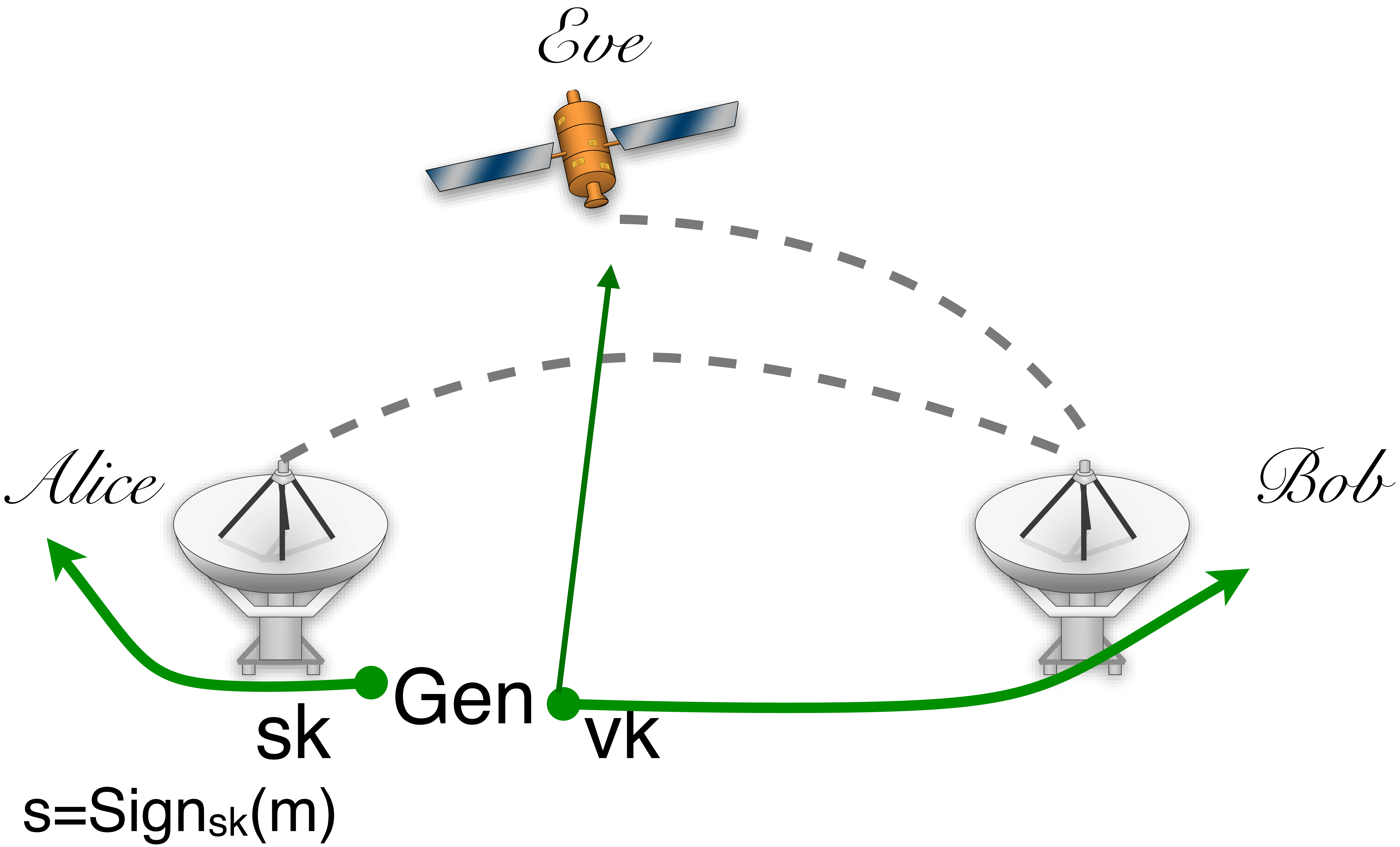
$$c \leftarrow (0||2||r||0||m)^e \bmod N$$

"padding oracle" attack against this scheme

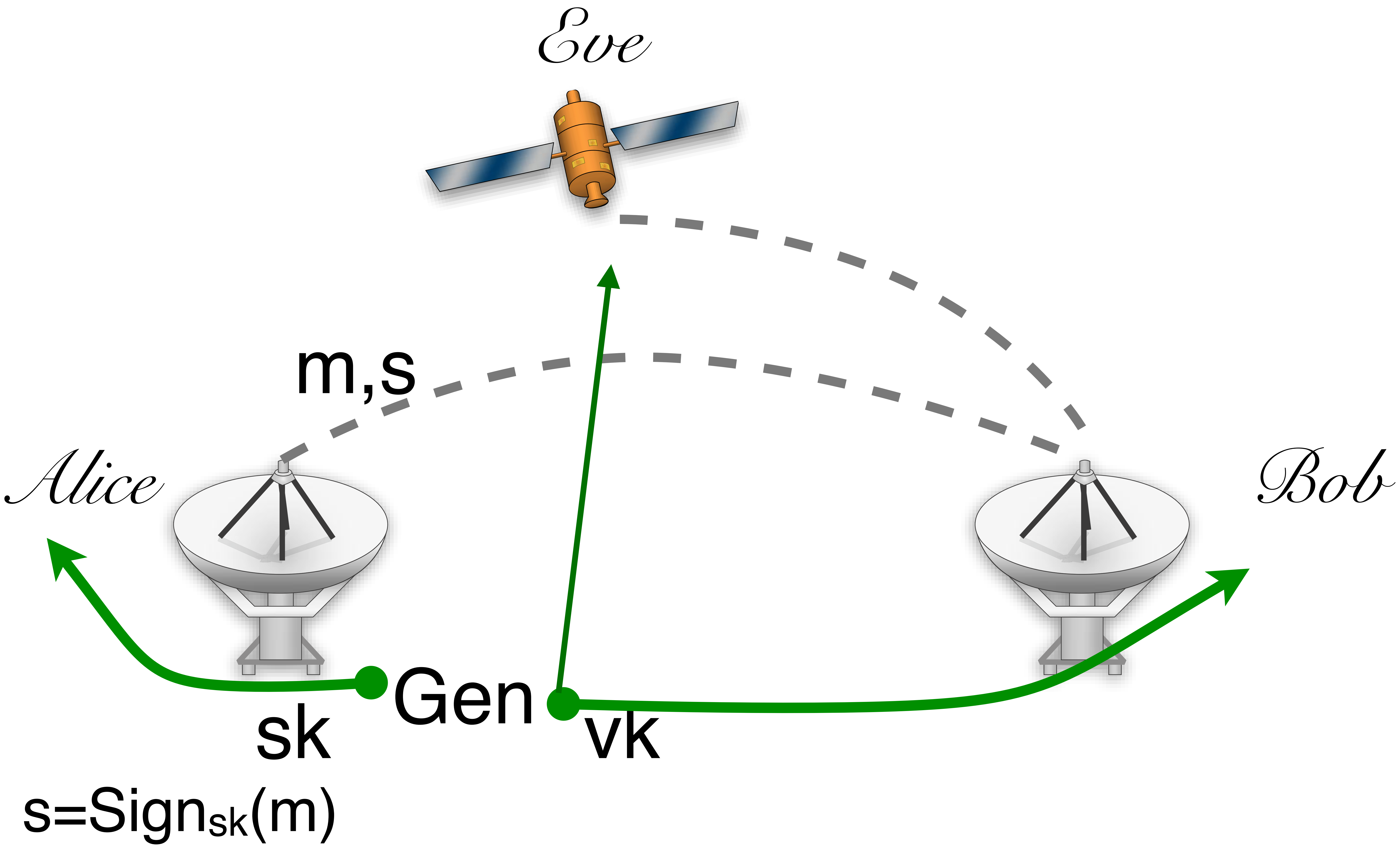
Public key digital signature



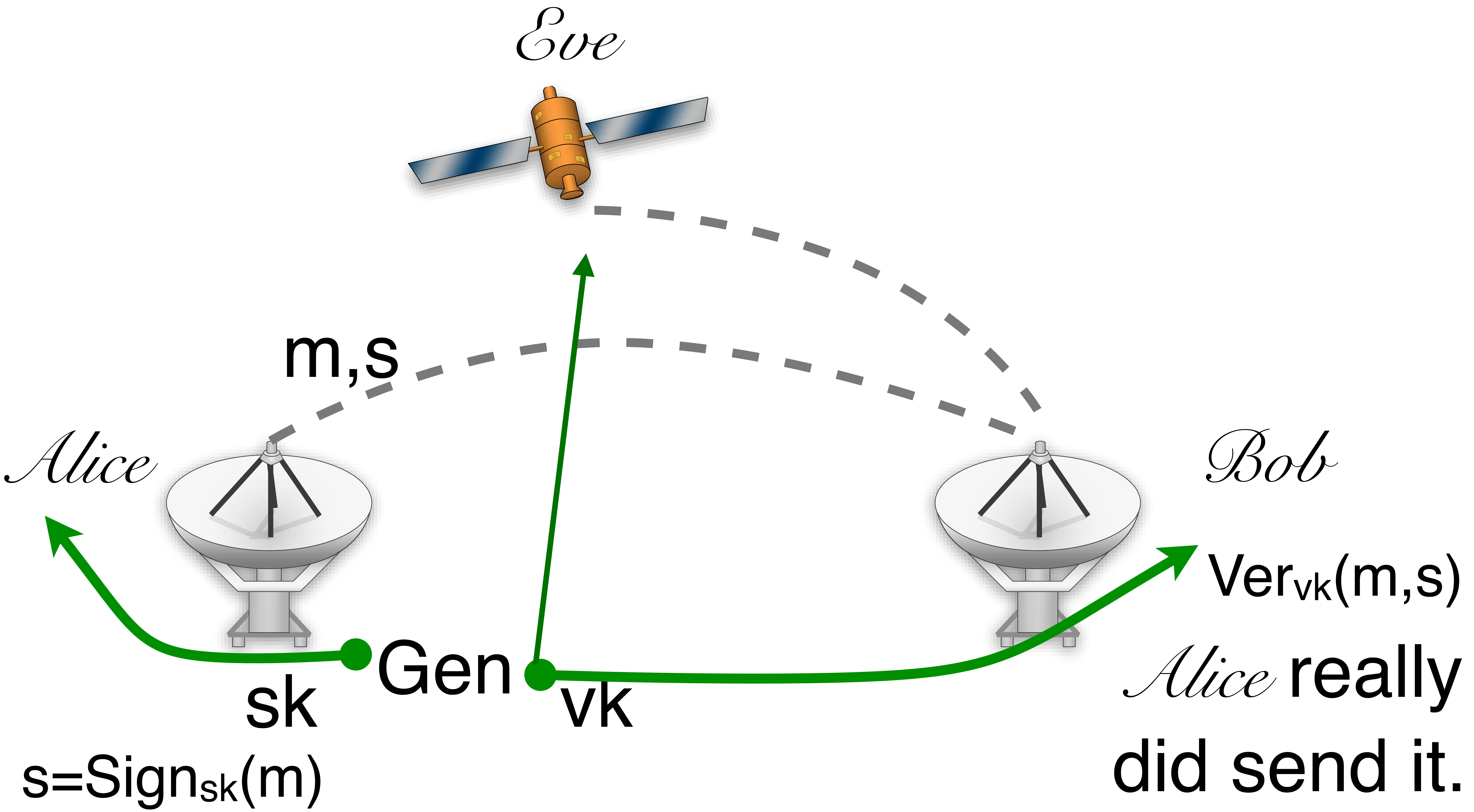
Public key digital signature



Public key digital signature



Public key digital signature



Public key digital signature

message space $\{\mathcal{M}\}_n$

$\mathrm{Gen}(1^n)$

$\mathrm{Sign}_{sk}(m)$

$\mathrm{Ver}_{vk}(m,s)$

Public key digital signature

message space $\{\mathcal{M}\}_n$

$\mathrm{Gen}(1^n)$ generates a key pair sk,vk

$\mathrm{Sign}_{sk}(m)$

$\mathrm{Ver}_{vk}(m,s)$

Public key digital signature

message space $\{\mathcal{M}\}_n$

$\mathrm{Gen}(1^n)$ generates a key pair sk,vk

$\mathrm{Sign}_{sk}(m)$ generates a signature s for $m \in \mathcal{M}_n$

$\mathrm{Ver}_{vk}(m,s)$

Public key digital signature

message space $\{\mathcal{M}\}_n$

$\text{Gen}(1^n)$ generates a key pair sk,vk

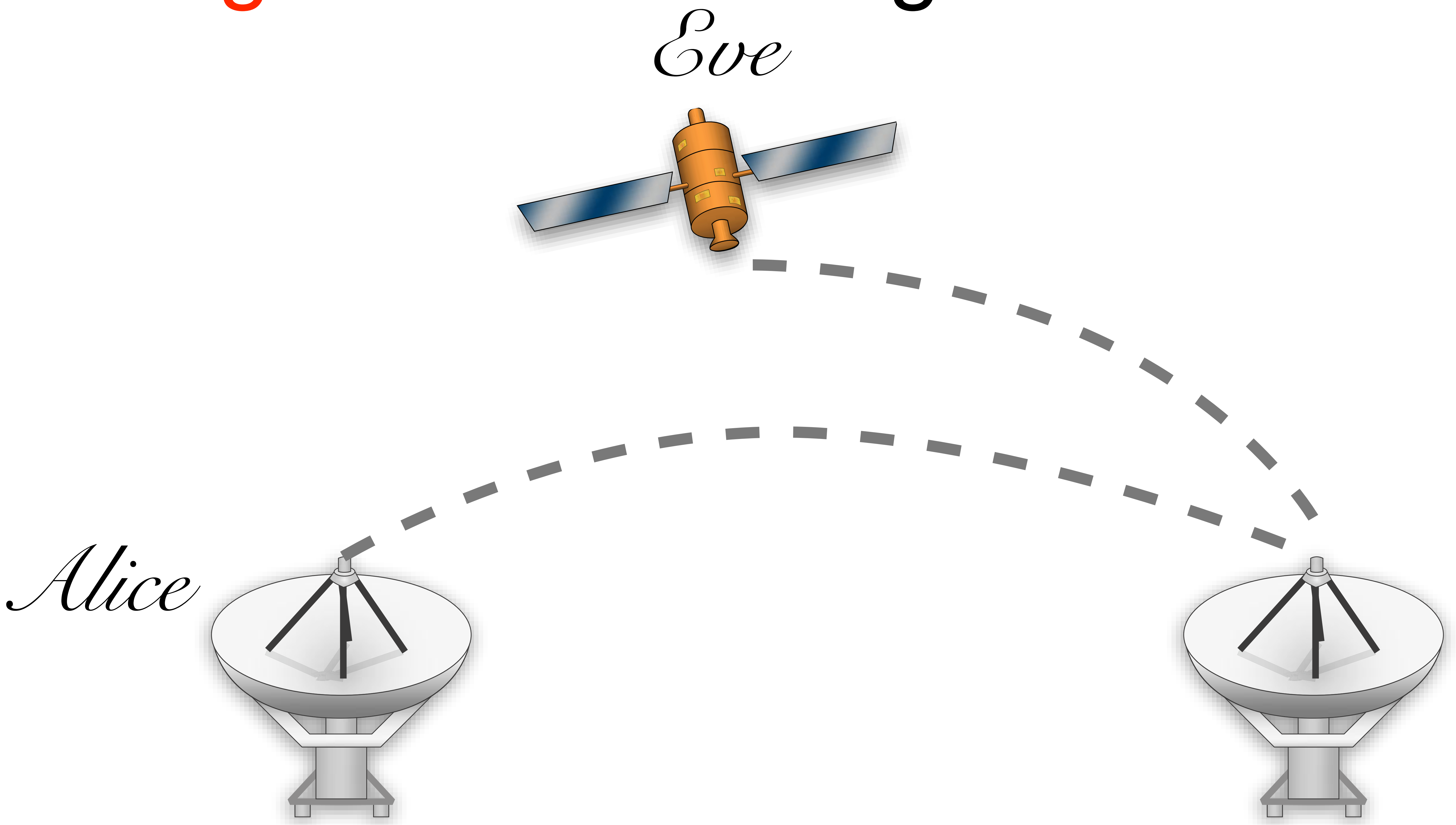
$\text{Sign}_{sk}(m)$ generates a signature s for $m \in \mathcal{M}_n$

$\text{Ver}_{vk}(m,s)$ accepts or rejects a msg,sig pair

$$\Pr[k \leftarrow Gen(1^n) : Ver_{vk}(m, Sign_{sk}(m)) = 1] = 1$$

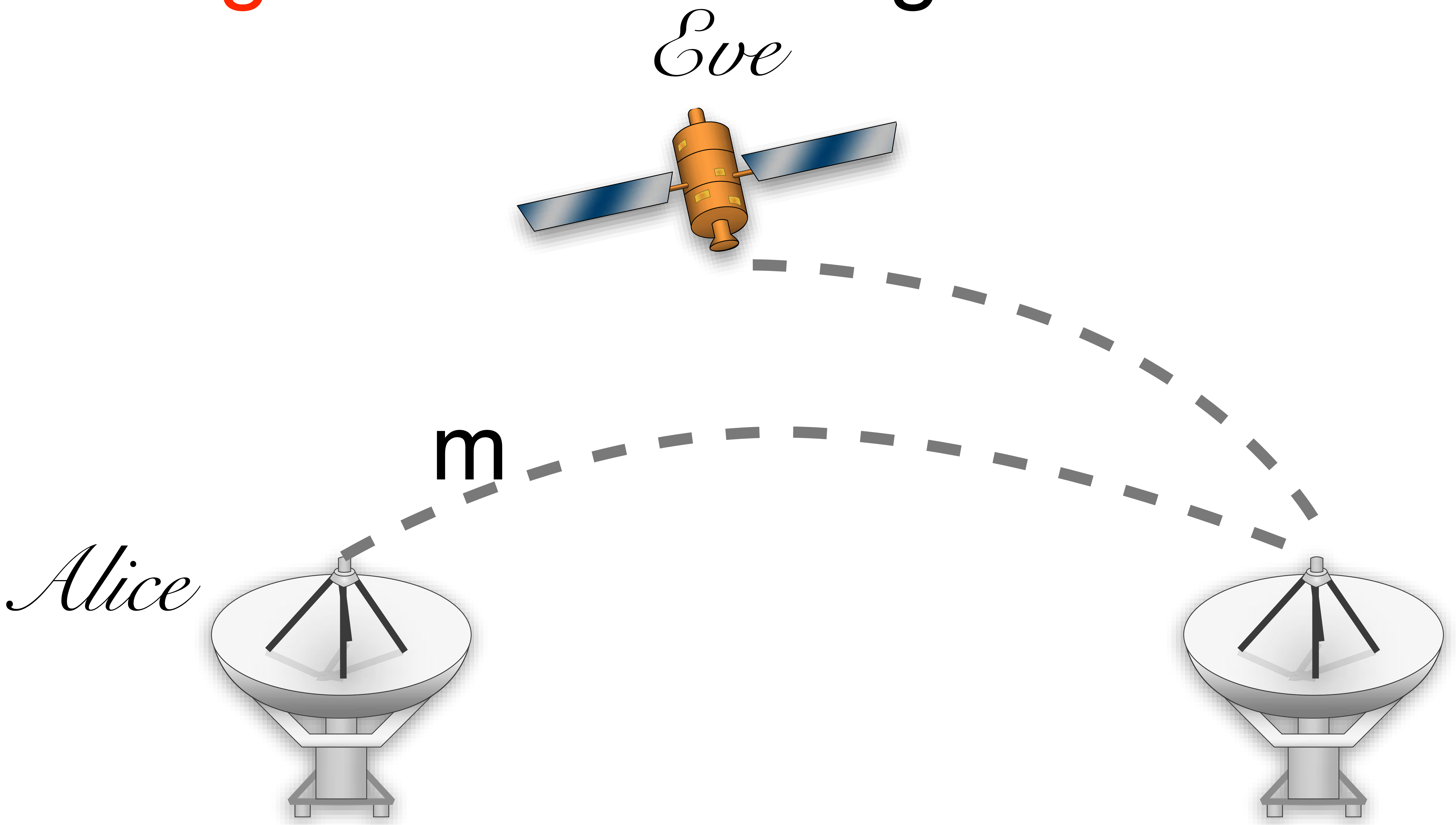
existential unforgability

“even when given a signing oracle,
an adversary cannot forge a signature for
any message of its choosing ”



existential unforgability

"even when given a signing oracle,
an adversary cannot forge a signature for
any message of its choosing "



for all non-uniform ppt A

$$\Pr[\qquad\qquad\qquad\qquad] < \mu(n)$$

for all non-uniform ppt A

$$\Pr\left[\begin{array}{l}(vk, sk) \leftarrow Gen(1^n); (m, s) \leftarrow A^{Sign_{sk}(\cdot)} : \\ \quad Ver_{vk}(m, s) = 1 \\ \quad \text{and A didn't query m}\end{array}\right] < \mu(n)$$

Textbook RSA Signatures (insecure)

Pick N = p*q where p,q are primes.

Pick e,d such that $e \cdot d = 1 \bmod \phi(N)$

Sign((sk=d, N) m):

Compute the signature: $\sigma \leftarrow m^d \bmod N$

Verify((pk=e, N), σ, m):

$$m \stackrel{?}{=} \sigma^e \bmod N$$

RSA Signatures in GPG

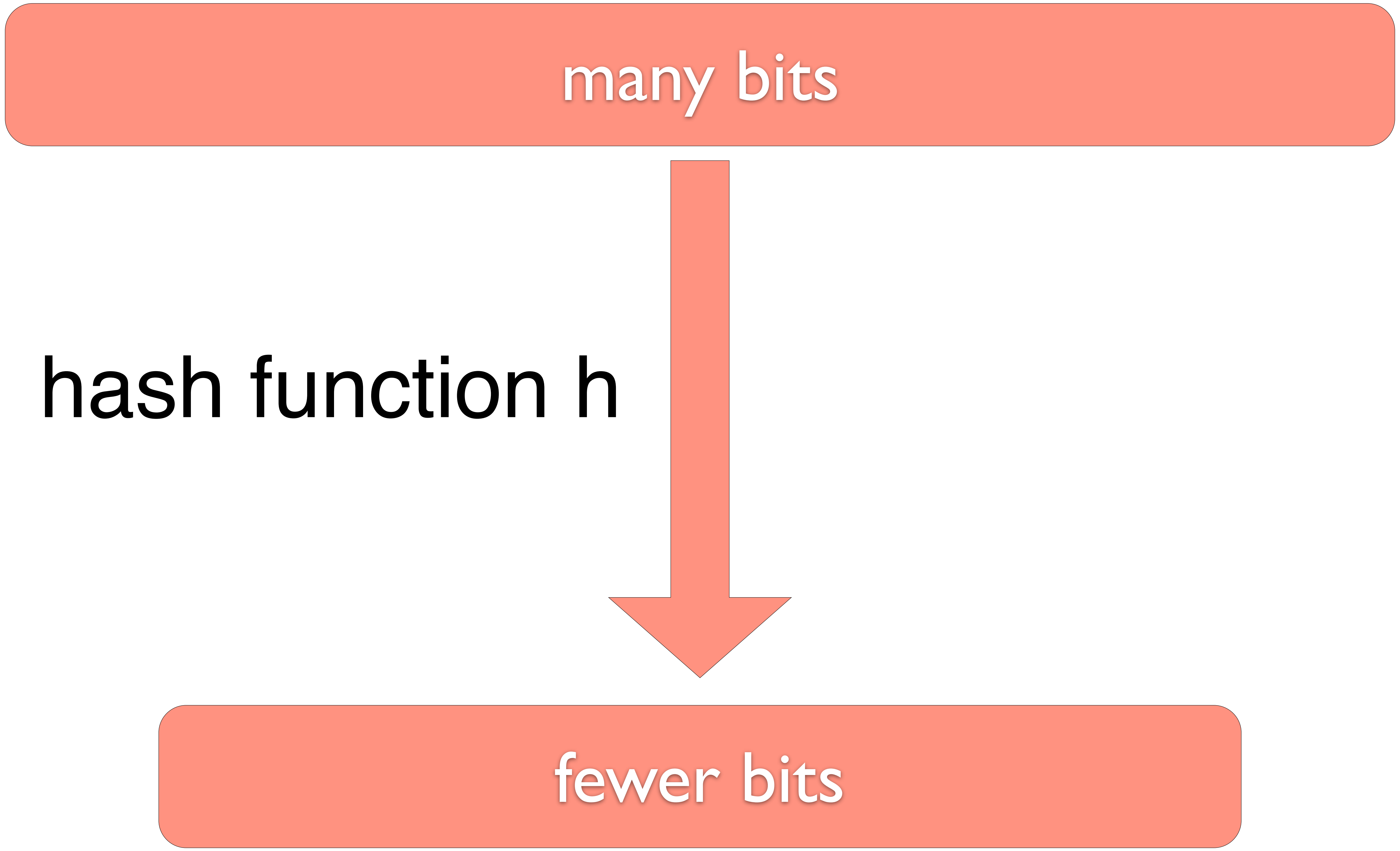
Sign((sk, N) m):

Compute the padding: $z \leftarrow 00 \cdot 01 \cdot FF \cdots FF \cdot 00 \cdot \mathsf{ID}_H \cdot H(m)$

Compute the signature: $\sigma \leftarrow z^{sk} \bmod N$

What is this H() function?

goal of a hash function



a hash function is a function

$$h : \{0, 1\}^d \rightarrow \{0, 1\}^r$$

such that h is easy to evaluate
and r < d

useful in data structures

```
public class test
{
    public static void main(String[] args)
    {
        System.out.println(args[0].hashCode());
    }
}
```

```
abhi$ java test HHHHHHHHHHHHHHHHHHHHGGGDD
-1644493785
```

collisions should be rare

```
public class test
{
    public static void main(String[] args)
    {
        System.out.println(args[0].hashCode());
    }
}
```

```
abhi$ java test HHHHHHHHHHHHHHHHHHHHGGGDD
-1644493785

abhi$ java test “hello world”
1794106052
```

java hash function

$$h(s) = \sum_{i=0}^{n} s[i]31^{n-i}$$

java hash function

$$h(s) = \sum_{i=0}^{n} s[i]31^{n-i}$$

it is thus easy to find a pair s_1, s_2
such that $h(s_1) = h(s_2)$

```
public class test
{
    public static void main(String[] args)
    {
        System.out.println(args[0].hashCode());
    }
}
```

```
abhi$ java test HHHHHHHHHHHHHHHHHHHHGGGDD
-1644493785
```

```
public class test
{
    public static void main(String[] args)
    {
        System.out.println(args[0].hashCode());
    }
}
```

```
abhi$ java test HHHHHHHHHHHHHHHHHHHHGGGDD
-1644493785

abhi$ java test HHHHHHHHHHHHHHHHHHHHGGGCc
-1644493785
```

```
public class test
{
    public static void main(String[] args)
    {
        System.out.println(args[0].hashCode());
    }
}
```

```
abhi$ java test HHHHHHHHHHHHHHHHHHHHGGGDD
-1644493785
```

```
abhi$ java test HHHHHHHHHHHHHHHHHHHHGGGCc
-1644493785
```

‘D’ - ‘c’ + 31(‘D’-‘C’) = 0

Collision resistant hash function

in addition to being easy to compute,
it should be "hard" for a p.p.t. adversary
to find a hash collision.

md4	1990
md5	1992
sha1	1994
sha256	2005
Sha3	2015

md4	1990	128 bit
md5	1992	128 bit
sha1	1994	160 bit
sha256	2005	256 bit
Sha3	2015	

md4	1990	128 bit	1995
md5	1992	128 bit	1998
sha1	1994	160 bit	2005*
sha256	2005	256 bit	
Sha3	2015		

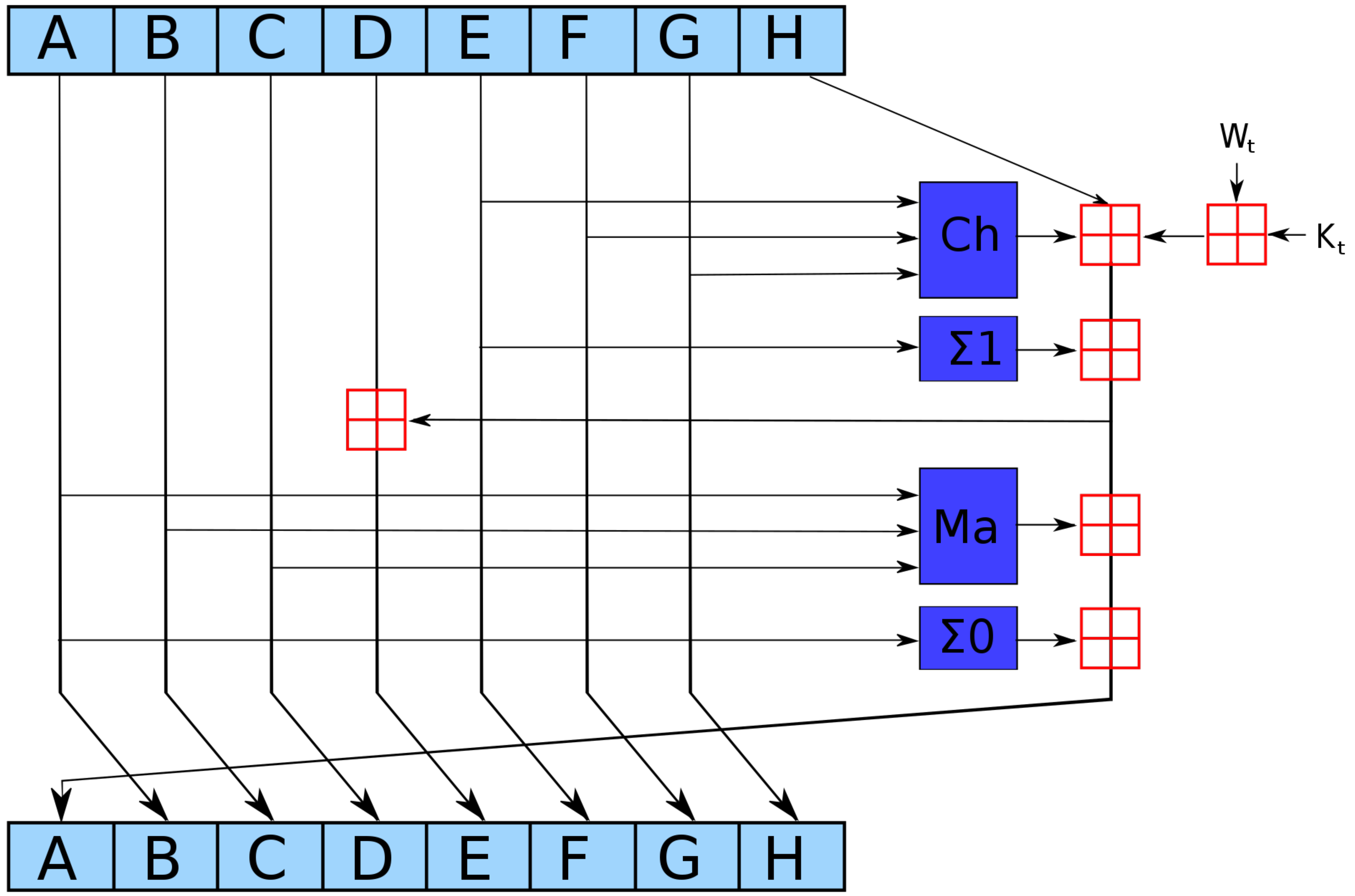
```
abhi18:neu abhi$ shasum -a 256
Noble patricians, patrons of my right,
Defend the justice of my cause with arms.
0c3c007b97cf8b75cfbd717804414a6a79b2defb4400eca9ea764a531a9ff193  -
```

Sha256

Pre-process the input

Break input into chunks

For each "chunk", repeat this 64 times:



Most cryptographers consider SHA256 to be indistinguishable from a “Random oracle”, i.e., a random function on arbitrary length messages.

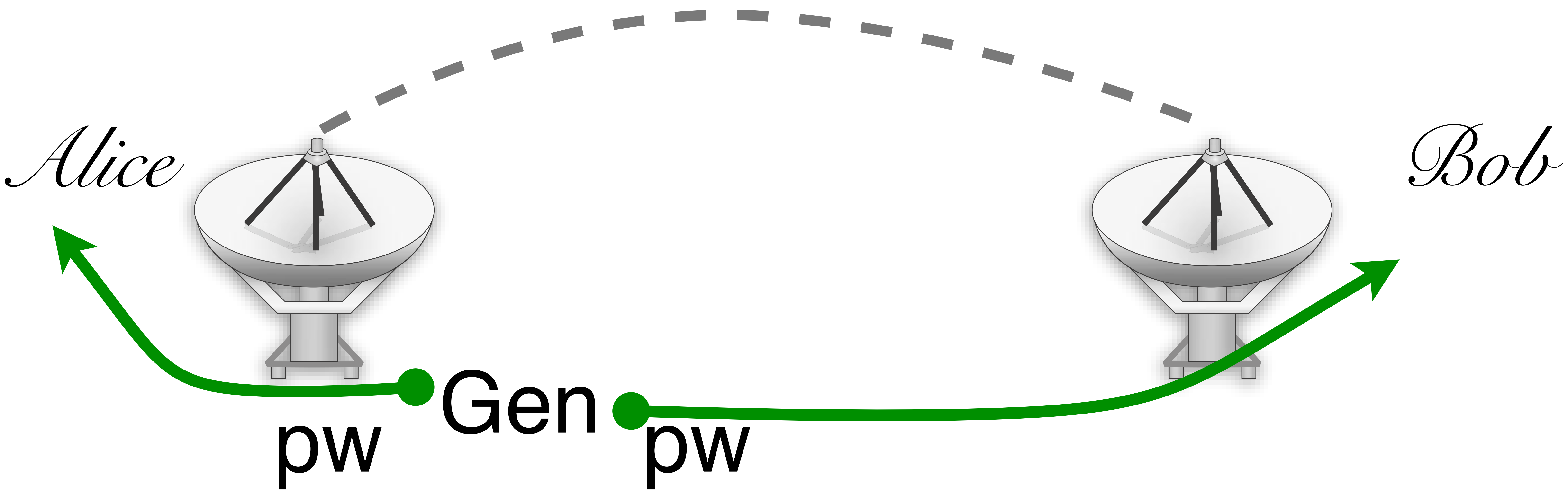
Recap:

Passwords

Main problem:

Passwords

Main problem:



Natural authenticators