Koliokviumas: dalyvavimas gyvai.
https://imimsociety.net/en/cryptography/29-diffie-hellman-key-agreement-protocol.html
https://imimsociety.net/en/cryptography/32-man-in-the-middle-attack.html
https://imimsociety.net/en/cryptography/34-textbook-rsa-signature.html
https://imimsociety.net/en/cryptography/35-textbook-rsa-encryption.html
https://imimsociety.net/en/cryptography/41-authentictaed-diffie-hellman-dh-key-agreement-protocol-kap.html

KD
http://crypto.fmf.ktu.lt/xdownload/

- Course_Work-Example.7z
- Course_Work-Requirements-2022.doc

Cryptography:

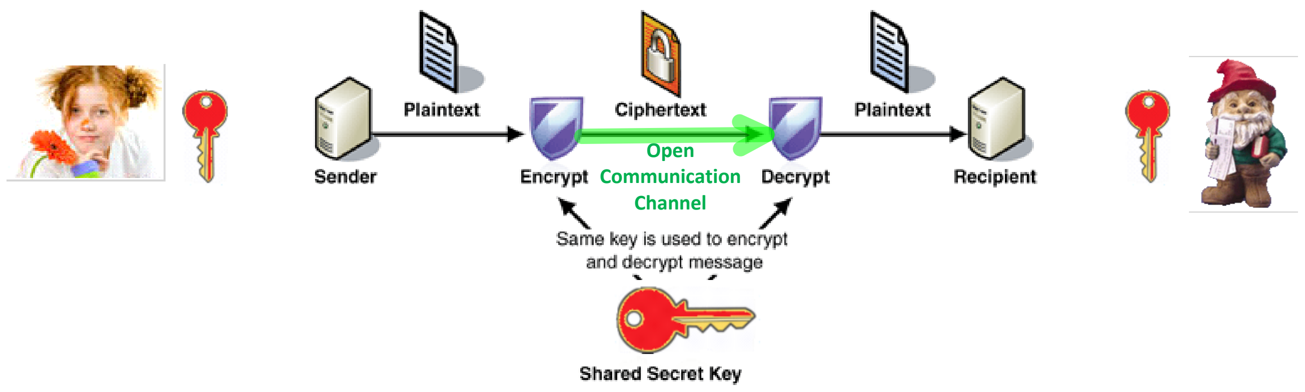
Information confidentiality, integrity, authenticity
person identification

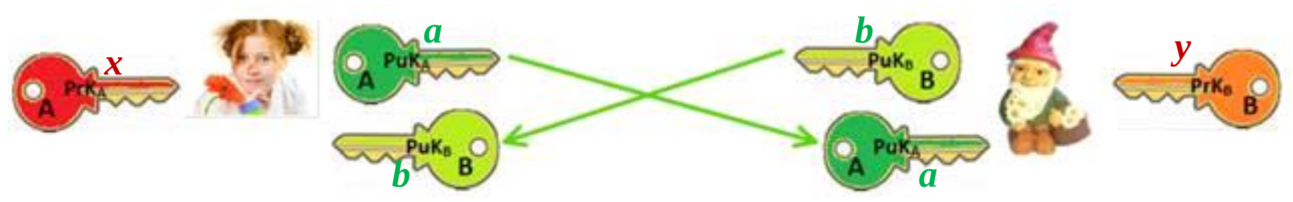
Symmetric cryptography --------------------- Asymmetric cryptography

Symmetric encryption
H-functions, Message digest
HMAC H-Message Authentication Code

Asymmetric encryption
E-signature - Public Key Infrastructure - PKI
E-money, cryptocurrencies, blockchain
E-voting
Digital Rights Management - DRM
Etc.

Symmetric - Secret Key Encryption





Message m < p

Asymmetric Signing - Verification

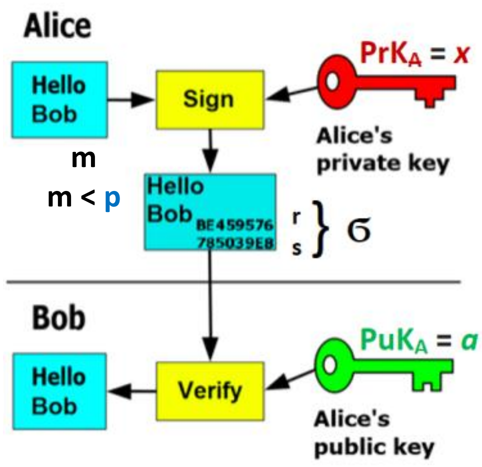
Sign(PrK_A, m) = σ = (r, s)

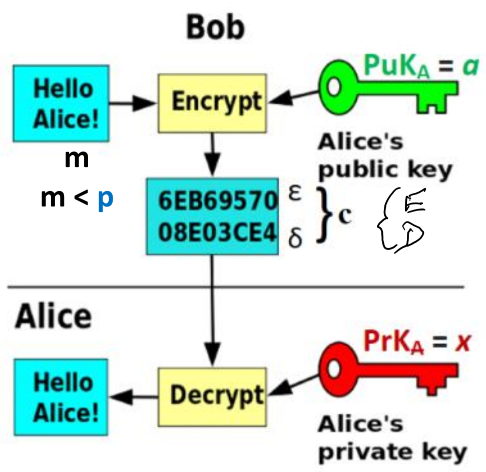
V=Ver(PuK_A, m, σ), V$\in\{$True, False$\}\equiv\{1, 0\}$

Asymmetric Encryption - Decryption

c=Enc(PuK_A, m)

m=Dec(PrK_A, c)





ϕ

RSA Cryptosystem:

Euler totient function **ϕ**(***n***): defines number of numbers **z** less than **n** that gcd(***z***,***n***)=1.

ϕ(***n***) = **ϕ** ≡ **fy**.

If ***n***=***p********q*** where ***p***,***q***-primes then **φ**(***n***) = **φ** = (***p***-1)*(***q***-1) ≡ **fy**.

Let ***n***=3*5=15 --> **ϕ**(***n***) = **ϕ** = (3-1)*(5-1) = 2*4 = 8 ≡ **fy**.

Euler theorem. If gcd(***z***,***n***)=1 then

$$z^{\phi} = 1 \bmod n$$

According to Euler theorem exponents are computed mod ϕ.

$Z_{15}' = \{1, 2, 3, ..., 14\}$ $*$ mod 15

16 | 15
15 | 1
(1)

Multiplication Tab. Z′15														
*	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14

*	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14
2	2	4	6	8	10	12	14	1	3	5	7	9	11	13
3	3	6	9	12	0	3	6	9	12	0	3	6	9	12
4	4	8	12	1	5	9	13	2	6	10	14	3	7	11
5	5	10	0	5	10	0	5	10	0	5	10	0	5	10
6	6	12	3	9	0	6	12	3	9	0	6	12	3	9
7	7	14	6	13	5	12	4	11	3	10	2	9	1	8
8	8	1	9	2	10	3	11	4	12	5	13	6	14	7
9	9	3	12	6	0	9	3	12	6	0	9	3	12	6
10	10	5	0	10	5	0	10	5	0	10	5	0	10	5
11	11	7	3	14	10	6	2	13	9	5	1	12	8	4
12	12	9	6	3	0	12	9	6	3	0	12	9	6	3
13	13	11	9	7	5	3	1	14	12	10	8	6	4	2
14	14	13	12	11	10	9	8	7	6	5	4	3	2	1

$^{-1}$

RSA key generation:

1. Two primes p, q are generated at random. $|q| = 1024$ bits $|p| = 1024$ bits
2. RSA module $n = p \cdot q$ is computed & $\phi(n) = (p-1)\cdot(q-1) = \phi$.
3. Random RSA exponent e : $\gcd(e, \phi) = 1$ is computed.
   According to RSA standard $e = 2^{16}+1$.
4. The inverse element to $e \bmod \phi$ is computed:
   $d = e^{-1} \bmod \phi \Rightarrow d\,e \bmod \phi = 1$.
5. $PrK = d$; $PuK = (n, e)$.

We use $|n| = 28$ bits $\longrightarrow$ $|p| = |q| = 14$ bits

```
>> e=2^16+1
e = 65537
>> isprime(e)
ans = 1
```

Key Generation

```
>> p=genprime(14)
p = 11491
>> q=genprime(14)
q = 14087
>> n=p*q
n = 161873717
>> e=2^16+1
e = 65537
>> fy=(p-1)*(q-1)
fy = 161848140
>> d=mulinv(e,fy)
d = 34529513
>> mod(e*d,fy)
ans = 1
```

RSA textbook encryption

m - message : $m < n \sim 2^{2048}$; $|m| < 2048$ bits

$\boxed{m \bmod n = m}$

B : $\xleftarrow{PuK_A}$ A : $PuK_A = (n, e)$; $PrK_A = d$.

$c = Enc(PuK_A, m) = m^e \bmod n$ $\xrightarrow{c}$ $Dec(PrK_A, c) = m =$

$= c^d \bmod n =$

$= (m^e)^d = m^{ed} =$

$= m^{\boxed{ed \bmod \phi}} \bmod n =$ $\rightarrow = 1$

$= m^1 \bmod n = m \bmod n = m$

```
>> m=int64(111222333)
m = 111222333
>> c=mod_exp(m,e,n)
c = 51722206
```

```
>> mm=mod_exp(c,d,n)
mm = 111222333
```

RSA textbook encryption is not randomised - is not probabilistic.

RSA textbook signature

A : $PuK_A = (n, e)$; $PrK_A = d$.

m - message : $m < n \Rightarrow m \bmod n = m$

$\sigma = Sign(PrK_A, m) =$ $\xrightarrow{m, \sigma}$ B : $PuK_A = (n, e)$

$= m^d \bmod n$

$Ver(PuK_A, \sigma, m) = \begin{cases} True \equiv 1 \\ False \equiv 0 \end{cases}$

```
>> sigma=mod_exp(m,d,n)
sigma = 149550780
```

$\sigma^e \bmod n =$

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

$= m^{de \bmod \phi} \bmod n =$

$= m^1 \bmod n = m'$.

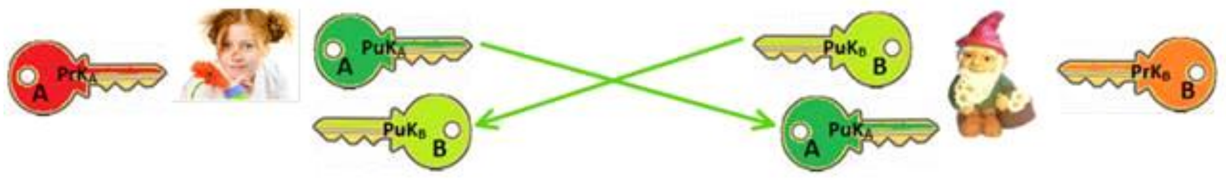
```
>> ver=mod_exp(sigma,e,n)
ver = 111222333
```

RSA textbook signature is a signature with message recovery.

If $m' = m$, then signature σ is formed by d corresponding to $PuK_A = (n, e) \Rightarrow$ True

RSA AKAP

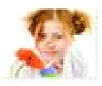
$PrK_A = d_A$; $PuK_A = (n_A, e_A)$. $PrK_B = d_B$; $PuK_B = (n_B, e_B)$.



AKAP using RSA signature

$PuK_A = (n_B, e); PrK_A = d_A.$
PuK_B

$PuK_B = (n_B, e); PrK_B = d_B.$
PuK_A



$u \leftarrow rand(Z_p^*)$
$g^u \bmod p = t_A \xrightarrow{\sigma_A} \quad \xrightarrow{\sigma_A} t_A$
$t_B \xleftarrow{\sigma_B} \quad \xleftarrow{\sigma_B}$

$v \leftarrow rand(Z_p^*)$
$t_B = g^v \bmod p$

$k_{AB} = (t_B)^u \bmod p =$
$= (g^v)^u \bmod p = g^{vu} \bmod p$

$k_{BA} = (t_A)^v \bmod p =$
$= (g^u)^v \bmod = g^{uv} \bmod p$

$k_{AB} = k = k_{BA}$

1) $Sign(d_A, t_A) = \sigma_A$
$\sigma_A = (t_A)^{d_A} \bmod n$

2) $Ver(PuK_B, \sigma_B, t_B) \in \{1, 0\}$
$t'_B = (\sigma_B)^{e_B} \bmod n_B = t_B$

3) $k_{AB} = (t_B)^u \bmod p$

1) $Ver(PuK_A, \sigma_A, t_A) \in \{1, 0\}$
$t'_A = (\sigma_A)^{e_A} \bmod n_A = (t_A)^{d_A e_A} \bmod n$
$= t_A^1 \bmod n_B = t_A$

2) $Sign(d_B, t_B) = \sigma_B$

3) $k_{BA} = (t_A)^v \bmod p$

$$K_{AB} == k == K_{BA}$$

```
>> p=genstrongprime(27)
p = 110918987
>> q=(p-1)/2
q = 55459493
>> g=2
g = 2
>> mod_exp(g,q,p)
ans = 110918986
```

```
>> u=int64(randi(2^27-1))
u = 48423797
>> tA=mod_exp(g,u,p)
tA = 14603504
>> n
n = 161873717
>> signA=mod_exp(tA,d,n)
signA = 20854858
```

```
n = 161873717
>> signA=mod_exp(tA,d,n)
signA = 20854858
>>
>> ttA=mod_exp(signA,e,n)
ttA = 14603504
```

Till this place

The "Hash-and-Sign" Paradigm.
The hashed RSA signature scheme can be viewed as an attempt to prevent certain attacks on the textbook RSA signature scheme.

M - message to be signed: $|M| \sim 1\,GB$

But signature must be placed on $m < n$: $|m| < 2048$ bits.

$H(M) = h$; $|h| = 256$ bit $\Rightarrow$ $|h| < 2048$ bits.

Signature is placen on h value:

$\mathrm{Sign}(PrK_A, h) = h^{d_A} \bmod n = \sigma_h$

$$A: \xrightarrow{M';\ \sigma_h} B: PuK_A = (n_A, e_A)$$

1. $h' = H(M')$.
2. $\mathrm{Ver}(PuK_A, \sigma_h, h') = 1$

$\mathrm{Ver}(\ ) = 1$ if $h' = H(M) = h$

If $h' = h \Rightarrow M' = M$.

3. B trust that M' is authentic.

$|n| \sim 28 \Rightarrow |p| = |q| = 14$ bits

```
>> p=genprime(14)
p = 8863
>> q=genprime(14)
q = 9497
>> n=p*q
n = 84171911
>> dec2bin(n)
ans = 101 0000 0100 0101 1100 1000 0111
```

```
>> e=2^16+1
e = 65537
>> isprime(e)
ans = 1
>> fy=(p-1)*(q-1)
fy = 84153552
>> gcd(e,fy)
ans = 1
>> e_m1=mulinv(e,fy)
e_m1 = 18083441
>> mod(e*e_m1,fy)
ans = 1
>> d=e_m1
```

Homomorphic property of RSA cryptosystem

Encryption. Let m_1, m_2 be messages to be encrypted

Let $m = m_1 \cdot m_2 \bmod n$

$\mathrm{Enc}(PuK_A, m) = c = m^e \bmod n = (m_1 \cdot m_2)^e \bmod n =$

$= m_1^e \cdot m_2^e \bmod n = \underbrace{\mathrm{Enc}(PuK_A, m_1)}_{c_1} \cdot \underbrace{\mathrm{Enc}(PuK_A, m_2)}_{c_2} \bmod n.$

$c = c_1 \cdot c_2 \bmod n$

Signing. Let $m = m_1 \odot m_2 \bmod n$.

$$Sig(PrK_A, m) = s = m^d \bmod n = (m_1 \cdot m_2)^d \bmod n =$$

$$= m_1^d \cdot m_2^d \bmod n = \underbrace{sig(PrK_A, m_1)}_{s_1} \cdot \underbrace{Sig(PrK_A, m_2)}_{s_2} \bmod n.$$

$$s = s_1 \odot s_2 \bmod n.$$

Generalized isomorphic property

Encryption.

$$\begin{cases} \text{If } m^* = m_1 \cdot m_2 \ \& \ m^+ = m_1 + m_2 \\ Enc(PuK, m^*) = c^* = c_1^* \cdot c_2^* = Enc(PuK, m_1) \cdot Enc(PuK, m_2) \\ Enc(PuK, m^+) = c^+ = c_1^+ + c_2^+ = Enc(PuK, m_1) + Enc(PuK, m_2) \end{cases}$$

Pascal Paillier enc.

$$\text{If } m^+ = m_1 \oplus m_2 \Rightarrow Enc(PuK, m^+) = c = c_1 \odot c_2$$

$$c_1 = Enc(PuK, m_1);\ c_2 = Enc(PuK, m_2).$$

signing. $Enc \rightarrow Sig$ & $PuK \rightarrow PrK$

$\rightarrow \{$

Security:

1.Hardness of factoring.

If $n = p \cdot q$, where p, q - primes, then RSA encryption & signing is secure if the factoring of n is a hard problem.

```
>> n = 15
>> factor(n)
 3   5
```

$$Z = p_1^{\alpha_1} \cdot p_2^{\alpha_2} \cdot \ldots \cdot p_N^{\alpha_n}$$

$$\cancel{1 \cdot 1} \cdot p_1^{\alpha_1} \cdot \ldots$$

If n sufficiently large, then factoring of n into multipliers p, q is infeasible with non-quantum computers.

Peter Shor in IBM corporation published a paper of quantum cryptanalysis.

Lattice Based and Hidden Field Equations based CS are reconed to be resistant to quantum crypt. anal.

$$A = {}^{x}W^{y}$$

Breakin RSA by factorization of n.

If p, q are found, when $n = p \cdot q \Longrightarrow$ Euler Totient Function ϕ can be computen $\Longrightarrow \phi(n) = (p-1)\cdot(q-1) = \phi$ is computed $\Longrightarrow$ havind $PuK = (n, e)$ the $PrK = d$ can be computed by the relation $\underline{e \cdot d = 1 \bmod \phi}$.

this computation is effective using classical computers

If factoring of n is known, then RSA CS is totally breaked $\Longrightarrow$ total breaking means PrK recovery (compromis.)

$>> d = \text{mulinv}(e, \phi) \Leftarrow d = e^{-1} \bmod \phi$

Masking technique:

Let m be a sum of money A would like to withdraw from Bank.

B: $PuK = (n, e)$; $PrK = d$.

$\xleftarrow{PuK}$

A: $t \leftarrow \text{rand}$

$\mu = m \cdot t^e \bmod n \xrightarrow{\mu}$

$S_t = Sig(PrK, \mu) =$

$= \mu^d \bmod n =$

$= m^d \cdot t^{ed} \bmod n =$

$= m^d \cdot t \bmod n$

$\xleftarrow{S_t}$

$S_t \cdot t^{-1} \bmod n =$

$= m^d \cdot \cancel{t} \cdot \cancel{t^{-1}} \bmod n =$

$= m^d = S_m$.

$m, S_m \xrightarrow{\quad}$ Seller

Till this place

2.Vulnerabilities.
  2.1.Common modulus attack in encryption.
  2.2.Forging signatures.

Non-randomness property

Necessity of probabilistic encryption.

Encrypting a message with textbook RSA always yields the same ciphertext, and so we actually obtain that any deterministic scheme must be insecure for multiple encryptions.

RSA padded encryption

PKCS # 1 v1.5. A widely-used and standardized encryption scheme, RSA Laboratories Public-Key Cryptography Standard (PKCS) # 1 version 1.5, utilizes what is essentially padded RSA encryption.

Hardness of factoring assumption serves as a useful bacground to the secure construction based on RSA **padding**.

One simple idea is to randomly pad the message before encrypting.

For a public key PuK = (**n**, **e**) of the usual form, let ***k*** denote the length of **n** in bytes; i.e.,
k is the integer satisfying $2^{8(k-1)} < n < 2^{8k}$.

Messages ***m*** to be encrypted are assumed to be a multiple of 8 bits long, and can have length up to ***k*** - 11 bytes.

Encryption of a message ***m*** that is ***D***-bytes long is computed as

$$c = (00000000||00000010||r||00000000||m)^e \bmod n \quad \text{//concatenation}$$

where ***r*** is a randomly-generated string of (***k*** - ***D*** - 3) bytes, with none of these bytes equal to 0.

Common modulus attack II. The attack just shown allows any employee to decrypt messages sent to any other employee.

This still leaves the possibility that sharing the modulus *n* is fine as long as all employees trust each other (or, alternatively, as long as confidentiality need only be preserved against outsiders but not against other members of the company) .

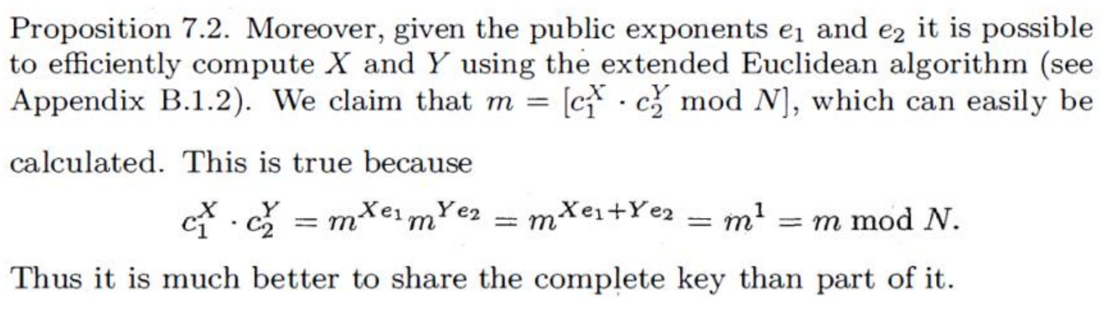
Here we show a scenario indicating that sharing a modulus is still a bad idea,. at least when textbook RSA encryption is used.

Say the same message m is encrypted and sent to two different (known) employees with public keys (*n*, *e1*) and (*n*, *e2*) where e1 $\neq$ e2 .

Assume further that gcd(e1 , e2) = 1 .

Then an eavesdropper sees the two ciphertexts *c1 = me1* mod *n* and *c2 = me2* mod *n*.

Since gcd(e1 , e2) = 1 , there exist integers X, Y such that X(e1) + Y(e2) = 1.



Proposition 7.2. Moreover, given the public exponents e_1 and e_2 it is possible to efficiently compute X and Y using the extended Euclidean algorithm (see Appendix B.1.2). We claim that $m = [c_1^X \cdot c_2^Y \bmod N]$, which can easily be calculated. This is true because

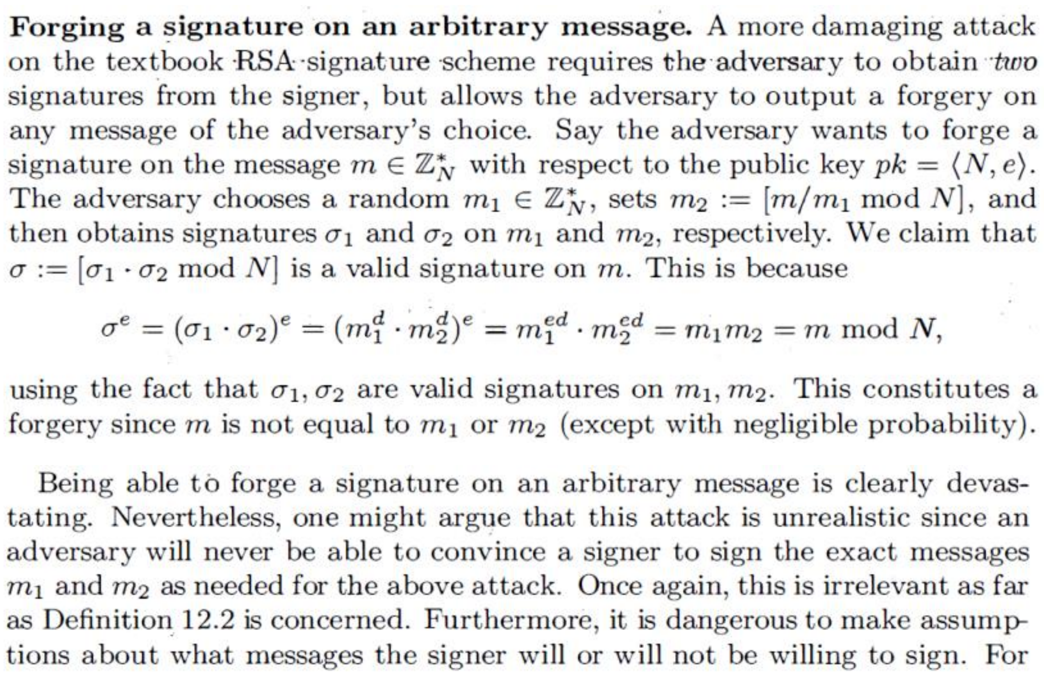
$$c_1^X \cdot c_2^Y = m^{Xe_1} m^{Ye_2} = m^{Xe_1+Ye_2} = m^1 = m \bmod N.$$

Thus it is much better to share the complete key than part of it.

This example and those preceding it should serve as a warning to only ever use RSA (and any other cryptographic scheme) in the exact way that it is specified.

Even minor and seemingly harmless modifications can open the door to attack.

RSA Textbook signature



Forging a signature on an arbitrary message. A more damaging attack on the textbook RSA signature scheme requires the adversary to obtain *two* signatures from the signer, but allows the adversary to output a forgery on any message of the adversary's choice. Say the adversary wants to forge a signature on the message $m \in \mathbb{Z}_N^*$ with respect to the public key $pk = \langle N, e \rangle$. The adversary chooses a random $m_1 \in \mathbb{Z}_N^*$, sets $m_2 := [m/m_1 \bmod N]$, and then obtains signatures σ_1 and σ_2 on m_1 and m_2, respectively. We claim that $\sigma := [\sigma_1 \cdot \sigma_2 \bmod N]$ is a valid signature on m. This is because

$$\sigma^e = (\sigma_1 \cdot \sigma_2)^e = (m_1^d \cdot m_2^d)^e = m_1^{ed} \cdot m_2^{ed} = m_1 m_2 = m \bmod N,$$

using the fact that σ_1, σ_2 are valid signatures on m_1, m_2. This constitutes a forgery since m is not equal to m_1 or m_2 (except with negligible probability).

Being able to forge a signature on an arbitrary message is clearly devastating. Nevertheless, one might argue that this attack is unrealistic since an adversary will never be able to convince a signer to sign the exact messages m_1 and m_2 as needed for the above attack. Once again, this is irrelevant as far as Definition 12.2 is concerned. Furthermore, it is dangerous to make assumptions about what messages the signer will or will not be willing to sign. For

The "Hash-and-Sign" Paradigm.
The hashed RSA signature scheme can be viewed as an attempt to prevent certain attacks on the textbook RSA signature scheme.
We omit considerations of these attacks.
But nevertheless, in general, it is not proved this signature to be secure.
RSA offers another advantage relative to textbook RSA: it can be used to sign arbitrary-length bit-strings.

In any case the randomization of signature should be implemented.