Koliokviumas: Lapkričio 6d., 13:30, 103f aud., 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

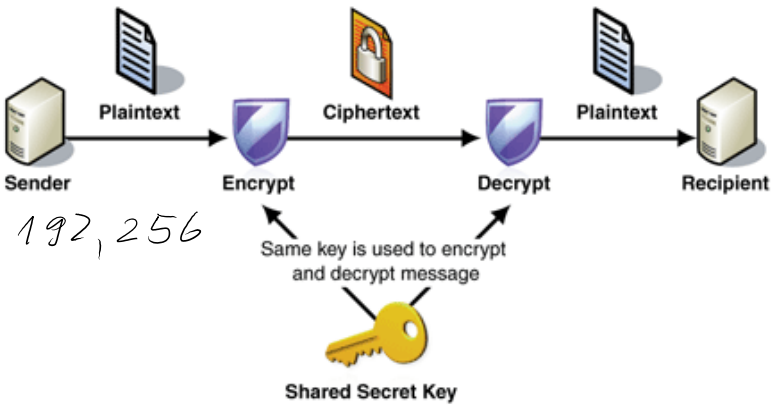
Cryptography:
Information confidentiality, integrity, authenticity
person identification

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



AES - 128, 192, 256

Asymmetric - Public Key Cryptography

Principles of Public Key Cryptography

Instead of using single symmetric key shared in advance by the parties for realization of symmetric cryptography, asymmetric cryptography uses two ***mathematically*** related keys named as private key and public key we denote by **PrK** and **PuK** respectively.

PrK is a secret key owned ***personally*** by every user of cryptosystem and must be kept secretly. Due to the great importance of **PrK** secrecy for information security we labeled it in red color. **PuK** is a non-secret ***personal*** key and it is known for every user of cryptosystem and therefore we labeled it by green color. The loss of **PrK** causes a dramatic consequences comparable with those as losing password or pin code. This means that cryptographic identity of the user is lost. Then, for example, if user has no copy of **PrK** he get no access to his bank account. Moreover his cryptocurrencies are lost forever. If **PrK** is got into the wrong hands, e.g. into adversary hands, then it reveals a way to impersonate the user. Since user's **PuK** is known for everybody then adversary knows his key pair (**PrK**, **Puk**) and can forge his Digital Signature, decrypt messages, get access to the data available to the user (bank account or cryptocurrency account) and etc.

Let function relating key pair (**PrK**, **Puk**) be ***F***. Then in most cases of our study (if not declared opposite) this relation is expressed in the following way:

$$\mathbf{PuK}=\boldsymbol{F}(\mathbf{PrK}).$$

In open cryptography according to Kerchoff principle function ***F*** must be known to all users of cryptosystem while security is achieved by secrecy of cryptographic keys. To be more precise to compute **PuK** using function ***F*** it must be defined using some parameters named as public parameters we denote by **PP** and color in blue that should be defined at the first step of cryptosystem creation. Since we will start from the cryptosystems based on discrete exponent function then these public parameters are

$$\mathbf{PP} = (p, g).$$

Notice that relation represents very important cause and consequence relation we name as the direct relation: when given **PrK** we compute **PuK**.

Let us imagine that for given ***F*** we can find the inverse relation to compute **PrK** when **PuK** is given. Abstractly this relation can be represented by the inverse function $\boldsymbol{F}^{-1}$. Then

$$\mathbf{PrK}=\boldsymbol{F}^{-1}(\mathbf{PuK}).$$

In this case the secrecy of **PrK** is lost with all negative consequences above. To avoid these undesirable consequences function ***F*** must be **one-way function** – OWF. In this case informally OWF is defined in the following way:

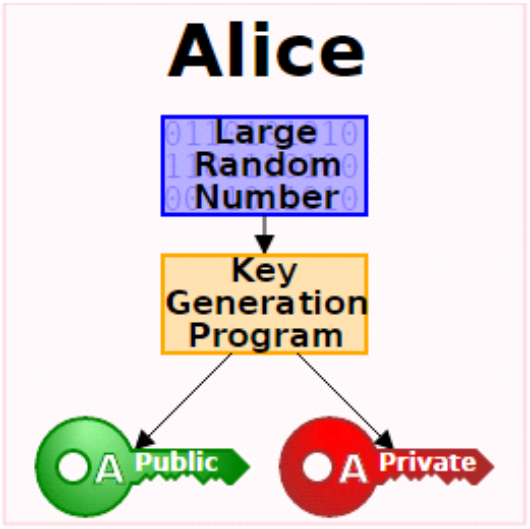
1. The computation of its direct value **PuK** when **PrK** and ***F*** in are given is effective.
2. The computation of its inverse value **PrK** when **PuK** and ***F*** are given is infeasible, meaning that to find $\boldsymbol{F}^{-1}$ is infeasible.

The one-wayness of ***F*** allow us to relate person with his/her **PrK** through the **PuK**. If ***F*** is 1-to-1, then the pair (**PrK**, **Puk**) is unique. So **PrK** could be reckoned as a unique secret parameter associated with certain person. This person can declare the possession or **PrK** by sharing his/her **PuK** as his public parameter related with **PrK** and and at the same time not revealing **PrK**.

So, every user in asymmetric cryptography possesses key pair (**PrK**, **PuK**). Therefore, cryptosystems based on asymmetric cryptography are named as **Public Key CryptoSystems** (PKCS).

We will consider the same two traditional (canonical) actors in our study, namely Alice and Bob. Everybody is having the corresponding key pair ($\mathbf{PrK_A}$, $\mathbf{PuK_A}$) and ($\mathbf{PrK_B}$, $\mathbf{PuK_B}$) and are exchanging with their public keys using open communication channel as indicated in figure below.

Asymmetric Key Generation



PrK and PuK are related

PuK = **F**(PrK)

F is one-way function

Having PuK it is infeasible to find

PrK = $\mathbf{F}^{-1}$(PuK)

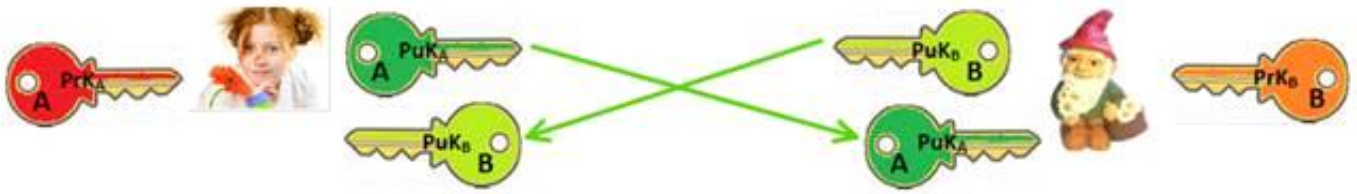
F(*x*)=a is OWF, if:

1.It easy to compute *a*, when *F* and *x* are given.

2.It is infeasible compute *x* when *F* and *a* are given.

PrK = *x* <-- randi ==> PuK = *a* = g^x mod p

Public Parameters PP = (p, g)



$PrK \sim 2^{2048} \rightarrow |PrK| = 2048 \text{ bits}$; $1T = 2^{40}$

$|Puk| = 2048 \text{ bits}$

Asymmetric Signing - Verification

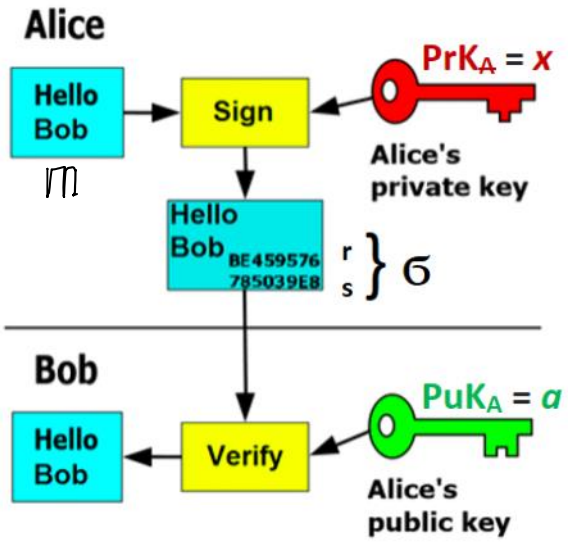
δ=Sign(PrK_A, m)

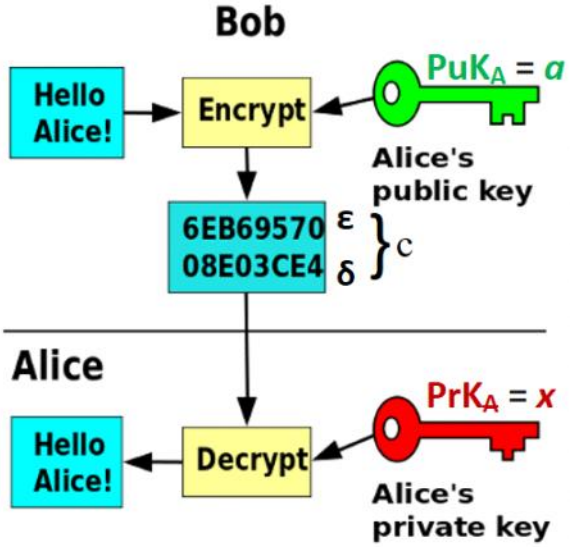
V=Ver(PuK_A, m, δ), V∈{True, False} ≡ {1, 0}

Asymmetric Encryption - Decryption

c=Enc(PuK_A, m)

m=Dec(PrK_A, c)





RSA Cryptosystem:

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

$\varphi(n) = \varphi \equiv$ **fy**.

If $n = p*q$ where ***p***,***q***-primes then $\varphi(n) = \varphi = (p-1)*(q-1) \equiv$ **fy**.

Let ***n***=3*5=15 --> $\varphi(n) = \varphi = (3-1)*(5-1) = 2*4 = 8 \equiv$ **fy**.

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

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

According to Euler theorem exponents are computed mod ϕ.

$\mathbb{Z}'_{15} = \{1, 2, 3, \ldots, 14\}$ * 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
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

Exp. Tab. **Z15**

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

$2^8 = 256 \bmod 15 = (255+1) \bmod 15 = \underbrace{255 \bmod 15}_{0} + 1 \bmod 15 = 1$

$\gcd(2,15)=1 \rightarrow 2^8 = 1 \bmod 15$

$\gcd(3,15)=3 \neq 1 \rightarrow 3^8 \neq 1 \bmod 15$

$\gcd(4,15)=1 \rightarrow 4^8 = 1 \bmod 15$

$\mathcal{I}_{15}^* \subset \mathcal{I}'_{15} = \{1, 2, \dots 4\}$

$\mathcal{I}_{15}^* = \{1, 2, 4, 7, 8, 11, 13, 14\}$

$|\mathcal{I}_{15}^*| = 8 = \phi(15) = \phi(3*5) = (3-1)*(4-1)$

RSA key generation:

1. Two primes p, q are generated at random.
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. $PrK = d$; $PuK = (n, e)$.

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

RSA textbook encryption

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

$m \bmod n = m$

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

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

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

$= c^d \bmod n =$

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

$= m^{ed \bmod \phi} \bmod n =$ ($ed \bmod \phi = 1$)

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

RSA textbook encryption is not randomised – is not probabilistic.

RSA textbook signature

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

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

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

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

$\sigma^e \bmod n =$

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

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

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

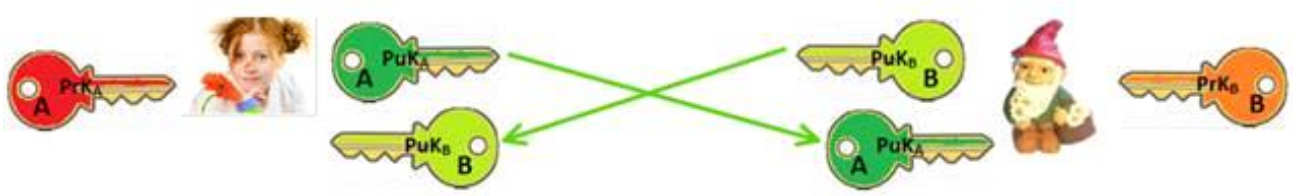
RSA textbook signature is a signature with message recovery.

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

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

σ_A σ_A t_A
σ_B σ_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) $Ver(PuK_A, \sigma_A, t_A) \in \{1, 0\}$
$t_A' = (\sigma_A)^{e_A} \bmod n_B = (t_A)^{d_A e_A} \bmod n$
$= t_A^1 \bmod n_B = t_A$
2) $sign(d_B, t_B) = \sigma_B$
3) $k_{BA} = (t_A)^v \bmod p$

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

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 placed on h value:

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

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

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

$Ver(\ ) = 1$ 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 $m = m_1 \circ m_2 \bmod n$

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

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

$c = c_1 \circ c_2 \bmod n$

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

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

$= \ldots \bmod n = Sig(PrK_A, m_1) \cdot Sig(PrK_A, m_2) \bmod n.$

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

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

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

Generalized isomorphic property

Encryption.

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

Pascal Paillier enc.

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 → Sig & PuK → PrK

→{

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
```

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

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

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 $n = p \cdot q \Rightarrow$ Euler Totient Function ϕ can be computen $\Rightarrow \phi(n) = (p-1)\cdot(q-1) = \phi$ is computed $\Rightarrow$ having $PuK = (n, e)$ the $PrK = d$ can be computed by the relation $e \cdot d = 1 \bmod \phi$.

this computation is effective using classical computers

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

>> $d = \text{mulinv}(e, \phi)$ $\not\Leftarrow$ $d = e^{-1} \bmod \phi$

Masking technique:

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

A: $t \leftarrow rand$

$\mu = m \cdot t^e \bmod n$

$\xleftarrow{PuK}$ B: $PuK = (n, e)$; $PrK = d$.

$\xrightarrow{\mu}$ $S'_t = Sig(PrK, \mu) = \mu^d \bmod n =$

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

$\xleftarrow{S'_t}$ $= m^d \cdot t \bmod n$

$S'_t \cdot t^{-1} \bmod n =$

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

$= m^d = S'_m$.

$m, S'_m \rightarrow$ 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 ≠ 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.