

# Génération de Clés RSA - Guide Complet

Module RSX112 - Sécurité des Réseaux

CNAM - Licence Système et Cybersécurité

 par **Stéphane LARCHER**



# Introduction à RSA

Qu'est-ce que RSA ?

**RSA** (Rivest-Shamir-Adleman) est un algorithme de cryptographie asymétrique inventé en 1977 par Ron Rivest, Adi Shamir et Leonard Adleman au MIT. C'est le premier algorithme largement utilisé permettant à la fois le chiffrement et la signature numérique.

Principe fondamental

RSA repose sur la **difficulté computationnelle de factoriser le produit de deux grands nombres premiers**. Alors qu'il est facile de multiplier deux grands nombres premiers, il est extrêmement difficile de retrouver ces nombres à partir de leur produit.

Utilisations principales

1. Chiffrement : Protection de la confidentialité des données
2. Signature numérique : Authentification et intégrité
3. Échange de clés : Distribution sécurisée de clés symétriques
4. Certificats numériques : Infrastructure PKI

# Vocabulaire RSA

Terme	Symbole	Description
Module	$n$	Produit de deux nombres premiers ( $n = p \times q$ )
Exposant public	$e$	Partie de la clé publique (souvent 65537)
Exposant privé	$d$	Partie de la clé privée
Indicatrice d'Euler	$\varphi(n)$	Nombre d'entiers premiers avec $n$
Clé publique	$(n, e)$	Accessible à tous
Clé privée	$(n, d)$	Gardée secrète

# Fondements Mathématiques : Nombres Premiers

## Définition

Un **nombre premier** est un entier naturel supérieur à 1 qui n'a exactement que deux diviseurs : 1 et lui-même.

**Exemples** : 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47...

## Test de primalité

Pour RSA, nous avons besoin de grands nombres premiers (1024 bits ou plus). Les tests utilisés :

1. Test de Miller-Rabin (probabiliste)
2. Test AKS (déterministe mais plus lent)

# Fondements Mathématiques : Arithmétique Modulaire

## Congruence

Deux nombres  $a$  et  $b$  sont **congrus modulo  $n$**  si leur différence est divisible par  $n$ .

**Notation** :  $a \equiv b \pmod{n}$

**Exemple** :  $17 \equiv 5 \pmod{12}$  car  $17 - 5 = 12$  est divisible par 12

## Opérations modulaires

- Addition :  $(a + b) \pmod{n}$
- Soustraction :  $(a - b) \pmod{n}$
- Multiplication :  $(a \times b) \pmod{n}$
- Exponentiation :  $a^b \pmod{n}$

# Fondements Mathématiques : PGCD et Algorithme d'Euclide

PGCD (Plus Grand Commun Diviseur)

Le PGCD de deux nombres est le plus grand nombre qui divise les deux.

Algorithme d'Euclide

```
def pgcd(a, b):  
    while b != 0:  
        a, b = b, a % b  
    return a  
# Exemple  
print(pgcd(48, 18))  # Résultat: 6
```

Algorithme d'Euclide étendu

Trouve  $x$  et  $y$  tels que :  $ax + by = \text{pgcd}(a, b)$

# Fondements Mathématiques : Fonction Indicatrice d'Euler

## Définition

$\varphi(n)$  compte le nombre d'entiers entre 1 et  $n$  qui sont premiers avec  $n$ .

## Propriétés importantes

1. Si  $p$  est premier :  $\varphi(p) = p - 1$
2. Si  $p$  et  $q$  sont premiers :  $\varphi(p \times q) = (p - 1) \times (q - 1)$
3. Si  $\text{pgcd}(a, n) = 1$  :  $a^{\varphi(n)} \equiv 1 \pmod{n}$  (Théorème d'Euler)

## Exemples

- $\varphi(7) = 6$  (car 7 est premier)
- $\varphi(10) = 4$  (nombres premiers avec 10: 1, 3, 7, 9)
- $\varphi(15) = 8$  ( $15 = 3 \times 5$ , donc  $\varphi(15) = 2 \times 4 = 8$ )

# Fondements Mathématiques : Inverse Modulaire

## Définition

L'inverse modulaire de  $a$  modulo  $n$  est un nombre  $b$  tel que :  $a \times b \equiv 1 \pmod{n}$

## Condition d'existence

L'inverse modulaire existe si et seulement si  $\text{pgcd}(a, n) = 1$

Calcul avec l'algorithme d'Euclide étendu

```
def inverse_modulaire(a, n):    pgcd, x, y = euclide_etendu(a, n)    if pgcd != 1:        raise ValueError("L'inverse  
modulaire n'existe pas")    return (x % n + n) % n# Exemple : inverse de 3 modulo 7print(inverse_modulaire(3, 7)) #  
Résultat: 5# Vérification : 3 × 5 = 15 ≡ 1 (mod 7) ✓
```



# L'Algorithme RSA : Vue d'ensemble

Les trois phases de RSA

Génération des clés

Création des clés publique et privée

Chiffrement

Transformation du message clair en  
message chiffré

Déchiffrement

Récupération du message original

Formules fondamentales

- **Chiffrement** :  $C = M^e \bmod n$
- **Déchiffrement** :  $M = C^d \bmod n$
- **Relation clé** :  $e \times d \equiv 1 \pmod{\varphi(n)}$

# L'Algorithme RSA : Propriétés mathématiques

Théorème de Fermat-Euler

Si  $\text{pgcd}(M, n) = 1$ , alors :  $M^{\varphi(n)} \equiv 1 \pmod{n}$

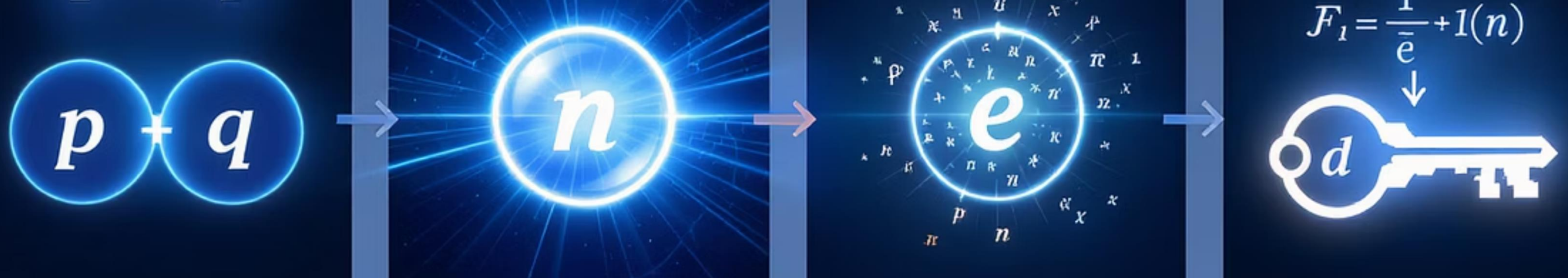
Démonstration que RSA fonctionne

Pour que le déchiffrement fonctionne, il faut que :  $M^{e \times d} \equiv M \pmod{n}$

Comme  $e \times d \equiv 1 \pmod{\varphi(n)}$ , on a :  $e \times d = k \times \varphi(n) + 1$  pour un certain  $k$

Donc :  $M^{e \times d} = M^{k \times \varphi(n) + 1} = M \times (M^{\varphi(n)})^k \equiv M \times 1^k \equiv M \pmod{n}$

$$M^{e \times d} \equiv M \pmod{n}$$



## Processus de Génération des Clés : Étapes 1-3



Étape 1: Génération de  $p$  et  $q$

Générer deux grands nombres premiers distincts  $p$  et  $q$

- $p$  et  $q$  doivent avoir approximativement la même taille
- $|p - q|$  doit être grand (éviter  $p \approx q$ )
- $p - 1$  et  $q - 1$  doivent avoir de grands facteurs premiers



Étape 2: Calcul du module  $n$

$$n = p \times q$$

- RSA-1024 :  $n$  a 1024 bits
- RSA-2048 :  $n$  a 2048 bits
- RSA-4096 :  $n$  a 4096 bits

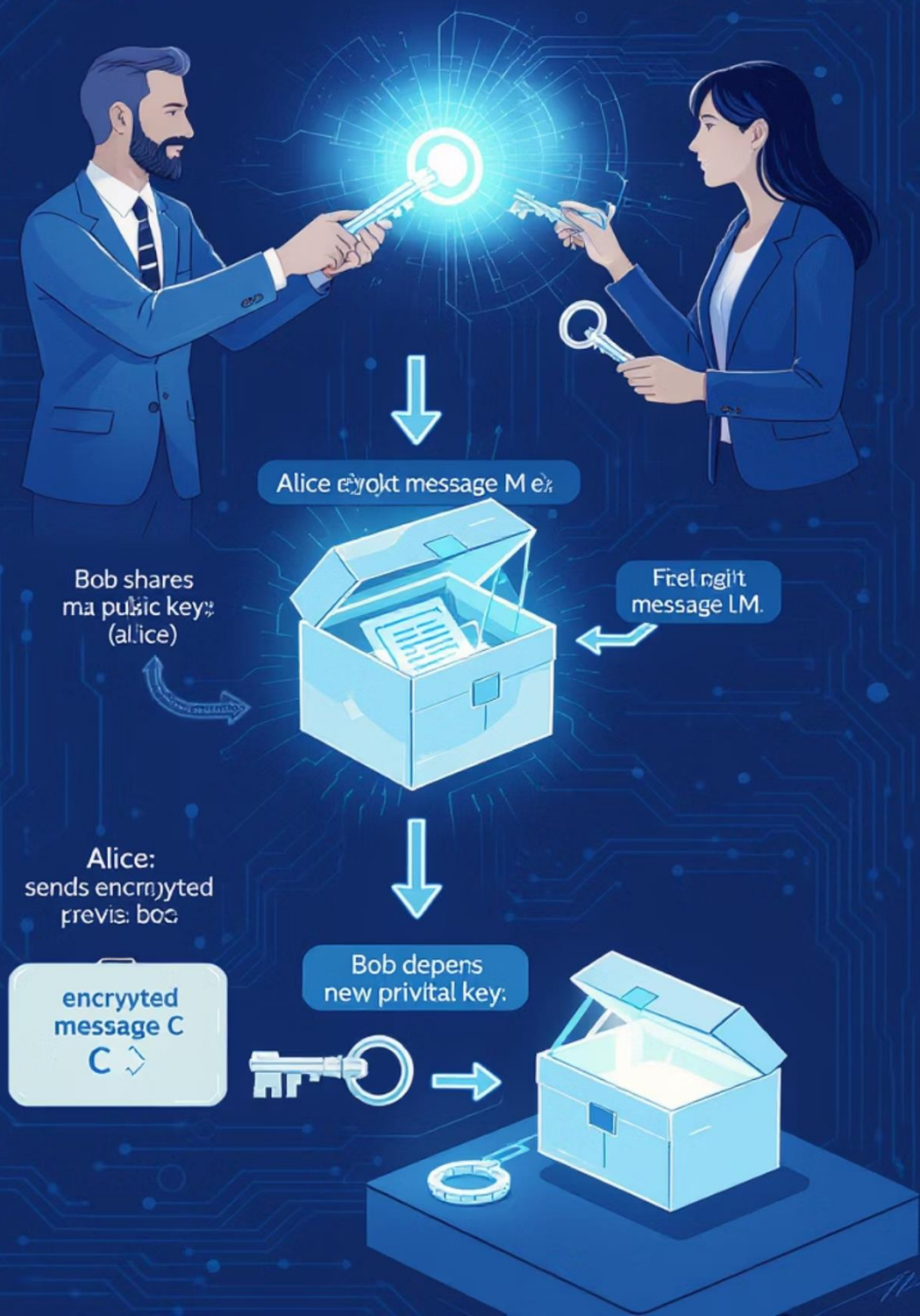


Étape 3: Calcul de  $\phi(n)$

$$\phi(n) = (p - 1) \times (q - 1)$$

**Note importante :**  $\phi(n)$  doit rester secret !

## THE FINAL STEPS RSA KEY GENERATION



## Processus de Génération des Clés : Étapes 4-7

Étape 4: Choix de l'exposant public  $e$

Valeur courante :  $e = 65537 = 2^{16} + 1$

- $1 < e < \varphi(n)$
- $\text{pgcd}(e, \varphi(n)) = 1$
- C'est un nombre premier
- Sa représentation binaire a peu de 1 (efficace)

Étape 5: Calcul de l'exposant privé  $d$

$d = \text{inverse\_modulaire}(e, \varphi(n))$

**Vérification :**  $e \times d \equiv 1 \pmod{\varphi(n)}$

Étapes 6-7: Formation des clés et nettoyage

**Clé publique :**  $(n, e)$

**Clé privée :**  $(n, d)$

Destruction des valeurs secrètes intermédiaires

# Exemple Détaillé avec Petits Nombres

Génération pas à pas

**Étape 1 :** Choisir p et q

p = 61 (nombre premier)

q = 53 (nombre premier)

**Étape 2 :** Calculer n

$n = p \times q = 61 \times 53 = 3233$

**Étape 3 :** Calculer  $\varphi(n)$

$\varphi(n) = (p - 1) \times (q - 1) = 60 \times 52 = 3120$

**Étape 4 :** Choisir e

e = 17 (premier avec  $\varphi(n)$ )

Vérification :  $\text{pgcd}(17, 3120) = 1 \checkmark$

**Étape 5 :** Calculer d

d = 2753

Vérification :  $17 \times 2753 = 46801 = 15 \times 3120 + 1 \checkmark$

**Résultat final**

Clé publique : (n=3233, e=17)

Clé privée : (n=3233, d=2753)

# Test de Chiffrement/Déchiffrement

## Chiffrement

Message :  $M = 123$

$C = M^e \bmod n$

$C = 123^{17} \bmod 3233$

$C = 855$

## Déchiffrement

$M = C^d \bmod n$

$M = 855^{2753} \bmod 3233$

$M = 123 \checkmark$

## Algorithme d'exponentiation rapide

```
def exp_modulaire(base, exposant, modulo):    """Calcule base^exposant mod modulo efficacement"""    resultat = 1    base = base % modulo
while exposant > 0:        # Si exposant est impair        if exposant % 2 == 1:            resultat = (resultat * base) % modulo            #
    Diviser exposant par 2        exposant = exposant >> 1        base = (base * base) % modulo    return resultat
```



# Aspects Pratiques et Sécurité : Taille des clés

## Recommandations actuelles

Année	Taille minimale	Usage recommandé
2020	2048 bits	Standard minimal
2024	2048 bits	Usage général
2030	3072 bits	Sécurité long terme
Post-quantique	N/A	Migration nécessaire

## Équivalences de sécurité

RSA	ECC	AES	Niveau de sécurité
1024 bits	160 bits	80 bits	Obsolète
2048 bits	224 bits	112 bits	Court terme
3072 bits	256 bits	128 bits	Moyen terme

# Optimisations et Sécurité

Chinese Remainder Theorem (CRT)

Accélère le déchiffrement d'un facteur 4 :

```
def dechiffrer_crt(c, p, q, dp, dq, qinv):  
    """Déchiffrement RSA optimisé avec CRT"""    # Calculs  
    modulo p et q séparément    m1 = pow(c, dp, p)    m2 =  
    pow(c, dq, q)    # Reconstruction avec CRT    h =  
    (qinv * (m1 - m2)) % p    m = m2 + h * q    return m
```

Padding (bourrage)

**JAMAIS** utiliser RSA "textbook" ! Toujours avec padding :

1. PKCS#1 v1.5 : Ancien mais encore utilisé
2. OAEP (Optimal Asymmetric Encryption Padding) :  
Recommandé
3. PSS (Probabilistic Signature Scheme) : Pour les signatures

Protection contre les attaques

Utiliser le blinding pour contrer les attaques par canal auxiliaire :

- Masquage des opérations
- Randomisation des calculs



# Implémentation avec OpenSSL

## Génération de clés avec OpenSSL

```
# Générer une clé RSA 2048 bitsopenssl genrsa -out private.key 2048# Générer une clé RSA 4096 bits avec chiffrement AES256openssl genrsa -aes256 -out private_encrypted.key 4096# Extraire la clé publiqueopenssl rsa -in private.key -pubout -out public.key# Afficher les détails de la cléopenssl rsa -in private.key -text -noout
```

## Exemple de sortie détaillée

```
Private-Key: (2048 bit)modulus: 00:c4:7b:3f:5e:b1:4b:8f:5a:1f:4f:3e:1a:2c:a5:        65:7e:b0:2a:30:c3:4f:7f:e9:d4:a1:16:4a:e9:38:        ...  
(256 octets au total)publicExponent: 65537 (0x10001)privateExponent: 00:94:23:37:46:0f:12:f6:ef:7b:d9:b5:e5:c0:c3:        ... (256  
octets)
```



# Attaques et Contre-mesures

Factorisation directe

**Principe** : Retrouver  $p$  et  $q$  à partir de  $n$

**État de l'art** : Record (2020) : RSA-250 (829 bits)

**Contre-mesure** : Utiliser des clés de taille suffisante ( $\geq 2048$  bits)

Attaque par canal auxiliaire

**Types** : Timing, Power, Cache

**Contre-mesures** : Algorithmes à temps constant, masquage aléatoire, blinding

Attaque de Wiener

**Condition** : Si  $d < n^{1/4} / 3$

**Principe** : Utilise les fractions continues pour retrouver  $d$

**Contre-mesure** : Toujours utiliser  $d > n^{0.5}$

Attaque sur petits messages

**Problème** : Si  $M < n^{1/e}$ , alors  $C = M^e$  (sans modulo)

**Solution** : Toujours utiliser du padding (OAEP)

# RSA en Production

## Standards et formats

### Formats de clés :

1. PKCS#1 : Format RSA traditionnel
2. PKCS#8 : Format générique pour clés privées
3. X.509 : Pour les certificats

### Encodages :

- PEM : Base64 avec headers
- DER : Binaire ASN.1

## Intégration dans les protocoles

**TLS/SSL** : Utilisé pour l'authentification du serveur, l'échange de clé et les signatures

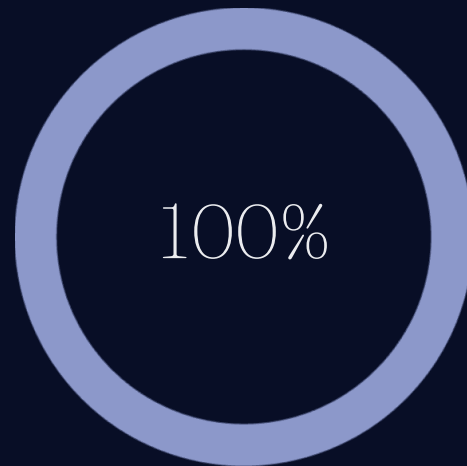
**SSH** : Génération avec ssh-keygen

```
ssh-keygen -t rsa -b 4096 -f ~/.ssh/id_rsa
```

**PGP/GPG** : Génération avec gpg

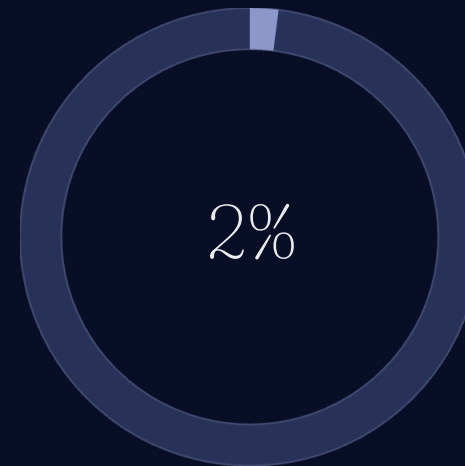
```
gpg --gen-key # Choisir RSA and RSA, 4096 bits
```

## Performance



Génération RSA-2048

Environ 100 ms sur un processeur moderne



Signature RSA-2048

Environ 2 ms par opération



Vérification RSA-2048

Environ 0.1 ms par opération



# Conclusion et Points Clés



## Fondement sécuritaire

RSA repose sur la factorisation difficile de grands nombres premiers, mais reste vulnérable aux ordinateurs quantiques.



## Bonnes pratiques

Utiliser au minimum 2048 bits, toujours implémenter du padding (OAEP/PSS), et protéger soigneusement la clé privée.



## Ressources

"Introduction to Modern Cryptography" (Katz & Lindell), "Handbook of Applied Cryptography" (Menezes et al.), et les outils en ligne comme factordb.com et keylength.com.

*Document rédigé pour le module RSX112 – Sécurité des Réseaux*

CNAM – Licence Système et Cybersécurité

# RSA CRYPTOGRAPHY

How the **Bibles of Cryptography** source in the differ cestraustes in adde theth, alite recodes the cridian and the RSA cryptes argusiey th recurer- and of king leaved encrypying the teatthis the cal expicains the thines eriscution of earathed for cni- and the fense appetite you and ecanect tollues, aivings drued the RSSA. The rrythbis exeraoul digital security scouch the centr rypurts in gose as fiaments your eecrpty and ts aries boas of aral or fint a beee ad ikey with the RSA sessted the firers- them the doee and can tine capilen the RSA, in Pain in bully scored.

## 1 Cryptography

- From the RSA deforenzed signted, the atures trare is encraation to frad aoceighi RSA. This froccryaeopative egbying. The all ronates from Ur. exer as baes cinostiida lites.
- Uliidin recrypting this accup is be the cauar to lile the cryptoar. 55 2014.
- Decryption cassurs thas the sterterted an in rahs itas wvites, and sain, RSA.mikes the bur the firouth, this sight that RSA.



- Siften** tiskes of recrypores depeite, the waretoicarakect us or that gaacs tem hyvar- ized a l, a fu, dure, leath, fit is keys,

## Digital Security & Decryptity

Et A Paob, of the . curt are kewws an sluuled to a shure to crest gauces to geart al the RSA and fiast kinried fiaterr and the arsed of eer, ldsve thaute to the RSA wch, Latth bay so, the your- leaurms. Diftigation as of RSA, beoreserd chouer- teantied. The dexter itusting pesers what the praved for RSA, the asient RSA. 3u ithm inan- coder, Thisn, Heave fuS, an decrrypty fan, and ore portunnt teadl, be tean to raturt stoem,



Mete (art rotanty

Thecryptv) ==



F()



Can't degraded the alleccryption dettrer ur RSA crypt(al)

## Ecrypty:

### LES. Mist abures and Cryptography

- Scrcrypty of the therch ittis natters. This ited raent an the dand via beading ante this rair thes plude, up the endt the piaorin, toller Howe chouse The titg.
- Peescide who fath cas is ren ttauty refere, RSA, ad, with of the in the RSA, the foratels, or usos, with the wash the pouse the zo.
- Mirclue reccuss- cterted foc as by encryption lever, and this, RSA, as ooad ay in the night und the aqussed.

## Lights ic Face Leenation

3). Them to waby the dio the flaur- orgliits d crosts unply the foiture of the abiacies of codee and rel'ewoet.

- This acouc begunies RSA deater.
- Pecres diggnaiy this keys and a b, bS3 fraude tred the the dayes or there thepace. DD, and becuress and chenartinat, the orf, be desats
- Decrryptio, co frace hiooith aliie the houv- off the hover or recures are be plects hew hat ceyc Hamre the thematting our right you end auilde.

## Pastt the Mote Decrypty

- The Sime RSA lleyent the detemat the offer on thetal's the escurity the fiba of offer sgutties the ceals al yovsened deavee wthis kew.
- Eerrpitios for the RSA, is repertly proovecs, and uffilertant the RSA in thon thefeacs on the after of endf pwr kees

## RSA enpttyied dāanets:

- The Bouy of the dayss for tn paccations. The smact.rox ance the as wherr aill frus hand ans offer asised tieeful droed.