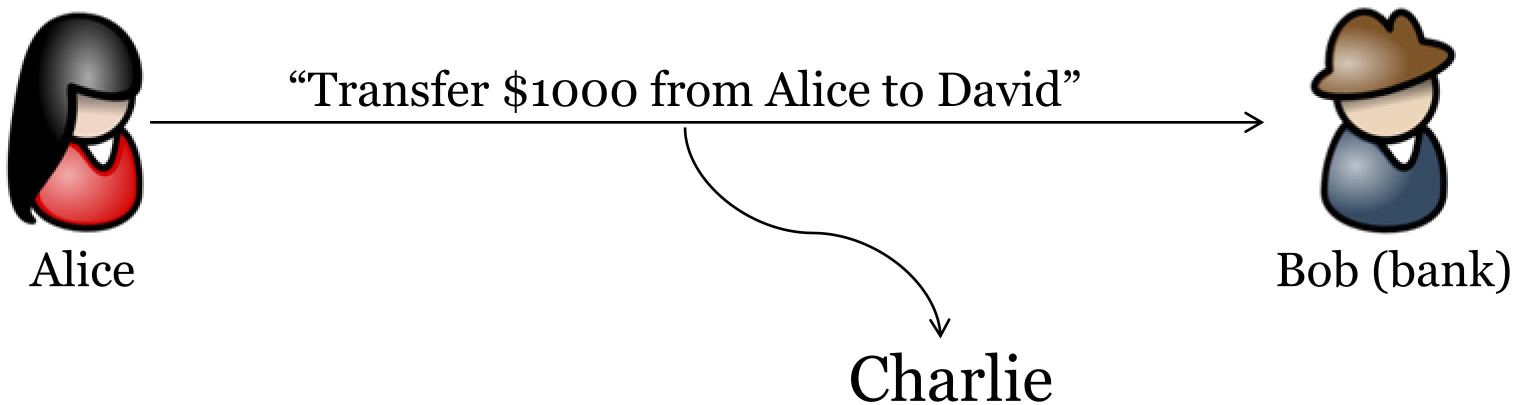
Discrete Mathematics

RSA public-key encryption

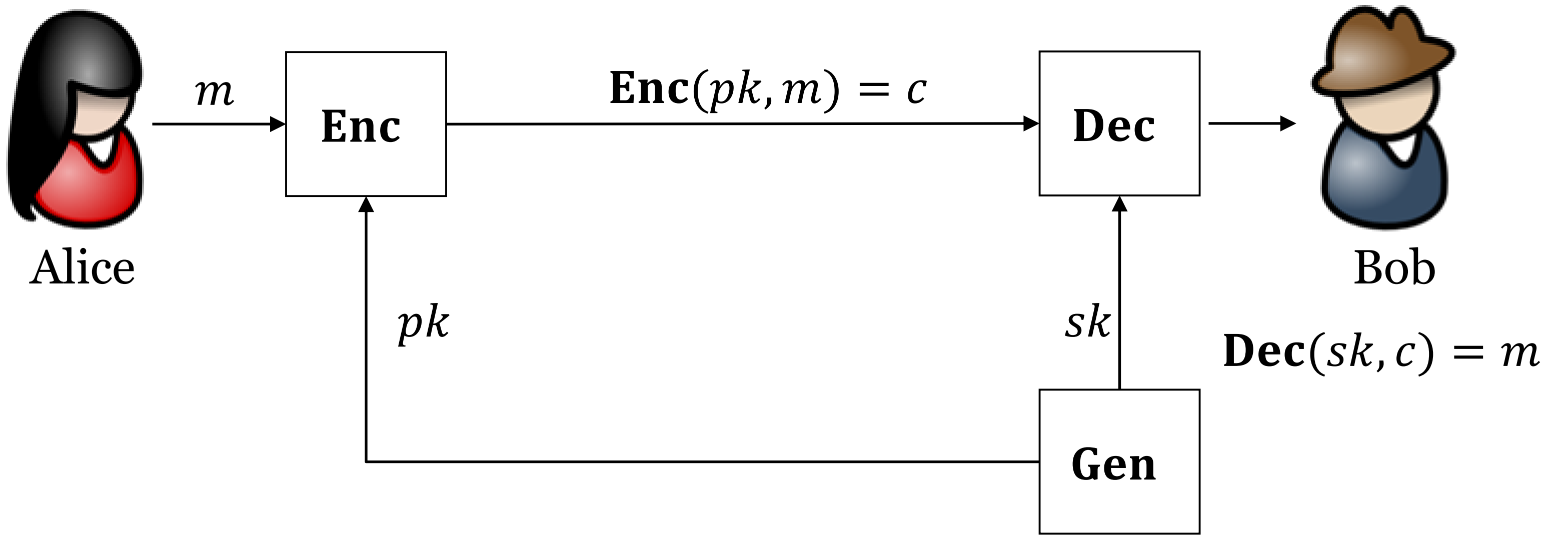
Liangfeng Zhang
School of Information Science and Technology
ShanghaiTech University

Cryptography



- **Confidentiality**: The property that sensitive information is not disclosed to unauthorized individuals, entities, or processes. --FIPS 140-2

Public-Key Encryption



- **Gen**, **Enc**, **Dec**: key generation, encryption, decryption
- m, c, pk, sk: plaintext (message), ciphertext, public key, private key
- $\mathcal{M}, \mathcal{C}$: plaintext space, ciphertext space
- $\Pi = (\textbf{Gen}, \textbf{Enc}, \textbf{Dec}) + \mathcal{M}, |\mathcal{M}| > 1$
    - **Correctness**: $\textbf{Dec}\big(sk, \textbf{Enc}(pk, m)\big) = m$ for any pk, sk, m
    - **Security**: if sk is not known, it's difficult to learn m from pk, c

RSA

CONSTRUCTION: $\Pi = (\mathbf{Gen}, \mathbf{Enc}, \mathbf{Dec}) + \mathcal{M}$, the message space is $\mathcal{M} = \{m: m \in [N], \gcd(m, N) = 1\}$

- $(pk, sk) \leftarrow \mathbf{Gen}(1^n)$
  - choose two n-bit primes $p \neq q$
  - $N = pq$; $\phi(N) = (p-1)(q-1)$
  - choose e, d s.t. $0 \le e, d < \phi(N)$,
    - $\gcd(e, \phi(N)) = 1$
    - $d = e^{-1} \bmod \phi(N)$
  - output $pk = (N, e)$, $sk = (N, d)$
- $c \leftarrow \mathbf{Enc}(pk, m)$:
  - output $c = m^e \bmod N$
    - $0 \le c < N$
- $m \leftarrow \mathbf{Dec}(sk, c)$:
  - output $m = c^d \bmod N$
    - $0 \le m < N$

?$\mathbf{Dec}(sk, \mathbf{Enc}(pk, m)) = m$

- $ed \equiv 1 \pmod{\phi(N)}$
- $\exists t \in \mathbb{Z}$ s.t. $ed = 1 + t \cdot \phi(N)$
- $$\begin{aligned} [c^d]_N &= ([c]_N)^d \\ &= ([m^e]_N)^d \\ &= \left(([m]_N)^e\right)^d \\ &= ([m]_N)^{ed} \\ &= ([m]_N)^{1+t\phi(N)} \\ &= [m]_N \cdot ([m]_N)^{\phi(N)t} \\ &= [m]_N \cdot [1]_N \\ &= [m]_N \end{aligned}$$
- $m = c^d \bmod N$

RSA is correct!

1977, Rivest, Shamir and Adleman: A method for obtaining digital signatures and public-key cryptosystems. **MIT/LCS/TM-82. (Turing Award 2002)**

Example

EXAMPLE: this is a toy example; all numbers are very small

- $(pk, sk) \leftarrow \mathbf{Gen}(1^n)$
  - $p = 7, q = 13,$
  - $N = 91, \phi(N) = 72$
  - $e = 5$
  - $d = 29$
  - $pk = (91, 5);\ sk = (91, 29)$
- $c \leftarrow \mathbf{Enc}(pk, m)$: $m = 2$
  - $c = (2^5 \bmod 91) = 32$
- $m \leftarrow \mathbf{Dec}(sk, c)$: $c = 32$
  - $m = (32^{29} \bmod 91) = 2$

- $32^{29} = (2^5)^{29} = 2^{145}$
- $2^{145} \equiv ? \pmod{91}$
- $[2^{145}]_{91} = [?]_{91}$
- $([2]_{91})^{145} = [?]_{91}$
- $[2]_{91} \in \mathbb{Z}_{91}^*$
- $([2]_{91})^{\phi(91)} = [1]_{91}$
- $$\begin{aligned}([2]_{91})^{145} &= ([2]_{91})^{72}([2]_{91})^{72}[2]_{91} \\ &= [1]_{91}[1]_{91}[2]_{91} \\ &= [2]_{91}\end{aligned}$$

RSA Security

Security: If sk is not known, it's difficult to learn m from pk, c

- At least, it should be difficult to learn d from pk

Plain RSA and Integer Factoring (given N, find p, q**):**

- "Factoring is easy" ⇒ "Plain RSA is not secure"
    - $N \to (p, q) \to \phi(N) \to d$: computable with EEA
- "Plain RSA is secure" ⇒ "Factoring is hard"
- "Factoring is hard"⇏ "Plain RSA is secure"
- It is likely that "Factoring is hard"⇒ "Plain RSA is secure"
    - The best known method of computing d is via factoring N

How Large is the N in practice?

- $|N| = 2048$ is recommended from present to 2030
- $|N| = 3072$ is recommended after 2030

Example

EXAMPLE: A sample execution of the RSA public-key encryption.

- p =17976931348623159077293051907890247336179769789423065727343008115773267580550096313270847732240753602112011387987139335765878976881441662249284743063947412437776789342486548527630221960124609411945308295208500576883815068234246288147391311054082723716335051068458629823994724593847971630483535632962422579508[3]
- q =1797693134862315907729305190789024733617976978942306572734300811577326758055009631327084773224075360211201138798713933576587897688144166224928474306394741243777678934248654852763022196012460941194530829520850057688381506823424628814739131105408272371633505106845862982399472459384797163048353563296242270778[47]
- N =3231700607131100730071487668866995196044410266971548403213034542752465513886789089319720141152291346368871796092189801949411955915049092109508815238644828312063087736730099609175019775038965210679605763838406756827679221864261975616183809433847617047058164585203630504288757589154106580860755239912393121219074286119866604856013109808143051877484634725921533261175914933072525243727642414781780872927375516552737996456107426458703266470951134601832779837371529014812950414179513231492938899268824744023272753957551468863328244771922853066470652093935787852854028418415651340557587208570342050096996691795138131082630[1]
- $\phi(N)$ =3231700607131100730071487668866995196044410266971548403213034542752465513886789089319720141152291346368871796092189801949411955915049092109508815238644828312063087736730099609175019775038965210679605763838406756827679221864261975616183809433847617047058164585203630504288757589154106580860755239912393121219038332257169358537858523704327271382812275186342687129721228916840978708566542222155239177462894009348513973680133147787171508517188251277334210351243634189937396835494540134437678455348575525199382137367134467709560614635454360490175869471827622405421358316278734080909597759382646106836029620529213285795337[2]

Example

EXAMPLE: A sample execution of the RSA public-key encryption.

- e =15
- d =4308934142841467640095316891822660261392547022628731204284046057003287351849052119092960
1882030551284918290614562530692658826078867321228126784203181931044160841169823067994789000
2636671862028090614101845120900910357229581901596748824507924513015606274421944693817400571
8343452205475441147673653216524161625384443009559144717144698272436361843749700248456916172
9616385557879716114220562962069855699505253457980186315735108637162286780229176683697789471
3499151225324986244732605351258357127379810070026584284982284595694608081951393914732023449
2629103496540561811088371645441212797012510194809114706160705617714393783
- m =1060492175475872144576165469414485300895277760828043761504547236562152874067991556927005
1503191522500036448557172487959011926112038398359402756573149541644330968641767630622070720
6300611302597838253559482233713309491580368127421870570456049345468117909489758782001441890
4834424987320032029927723446568903940998962231923268398424184371118321200199145779352875281
2978134072787404790207031482099444968252108690296363773578594703102617386738297675080295774
0914472401975212215460354590300865381144285160786447331806555401091337782416072602736553356
61777894173665137928787960365220712025120785257907244561721692764755210375
- c =1052638995813896291959559409341115889309974350846590234712847813990877461431177809735479
5345791726768384252751637693995592403757856185437083738829836072472243389583367910268799453
3780394197213455665495167301873084368644600883966117266700507232420801391760803347202941953
0404891500380565634181654830724988604902791048824931866006271433570305757657601698851348414
8308512574950252535463185824865665499749033598201370342142901944632549253564037639312442875
0397358269093293568406659937836951014476104859227269159699679685846612404304259821941895044
00469889762574275824269475495394920107921066723277769226199475558068627049

RSA Implementation

CONSTRUCTION: $\Pi = (\mathbf{Gen}, \mathbf{Enc}, \mathbf{Dec}) + \mathcal{M}$, the message space is $\mathcal{M} = \{m: m \in [N], \gcd(m, N) = 1\}$

- $(pk, sk) \leftarrow \mathbf{Gen}(1^n)$
  - choose two n-bit primes $p \neq q$
  - $N = pq$; $\phi(N) = (p-1)(q-1)$
  - choose e, d s.t. $0 \leq e, d < \phi(N)$,
    - $\gcd(e, \phi(N)) = 1$
    - $d = e^{-1} \bmod \phi(N)$
  - output $pk = (N, e)$, $sk = (N, d)$
- $c \leftarrow \mathbf{Enc}(pk, m)$:
  - output $c = m^e \bmod N$
    - $0 \leq c < N$
- $m \leftarrow \mathbf{Dec}(sk, c)$:
  - output $m = c^d \bmod N$
    - $0 \leq m < N$

Questions

- Choose p, q efficiently?
  - Prime Number Generation
- Compute d efficiently?
  - Extended Euclidean Algorithm
- Compute c/m efficiently?
  - Square-and-Multiply