Cryptography
Week #11:
PKI & Homomorphic Encryption

Rogério Reis, rogerio.reis@fc.up.pt
MSI/MCC/MIERSI - 2024/2025
DCC FCUP

December, 5th 2025

Why *PKI*?

Why *PKI*?

All *PK* cryptography primitives assume public-keys are authentic.
If not true, protocols are vulnerable to man-in-the-middle attacks.
In the real-world this problem can be solved in an ad-hoc way:

- manually confirm public-key belongs to intended party;
- systems (e.g., GPG/PGP) supporting ad-hoc *PK* authentication.

When legal/regulatory coverage is required $\Longrightarrow$ *PKI*:

- Technical standards: which algorithms/encoding formats to use
- Regulations: how technical standards should be used
- More Regulations: responsibilities and rights of involved parties
- Laws: formal guarantees and penalties wrt regulations

Public-key certificates

Public-key certificates

Goal:

- Alice sends Bob a public key p_k over an insecure channel
- Bob must be able to check Alice holds associated secret key

Trivial solution:

- Bob has authenticated channel to Trusted-Third-Party (TTP)
- Alice has previously proved to TTP that she owns p_k (how?)
- Bob asks TTP (on-line) if p_k belongs to Alice

Problems in practice:

1. How does Bob build authenticated channel to TTP?
2. What happens if TTP is off-line?
3. How do Bob and Alice get to work with the same TTP?
4. What does "Trust" in TTP mean?

Public-key certificates (2)

Public-key certificates use signatures to solve points 1 and 2:

- *TTP* is called a Certification Authority (*CA*)
- Alice proves to *CA* that she owns p_k
  - By signing a certificate request (PKCS#11)
  - Because *CA* itself provides secret key to Alice
- *CA* provides/checks data Alice wants on certificate:
  - Alice identity + public key
  - *CA*-specific information: serial number, issuer identity
  - Validity (start and end dates)
- *CA* signs data as a byte-encoded ASN.1 data structure.

PK **Certificate := Alice's data and** *PK* + *CA* **signature**
Trust in certificate $\leq$ **Trust in** *CA*

Public-key certificates (3)

What is ASN.1[1]?

- Abstract Syntax Notation 1: platform/language independent
- Legacy specification language from networking standards
- Standards use ASN.1 to specify data structures (packets)
- DER (Distinguished Encoding Rules) specify byte encoding

How do certificates solve points 1 and 2:

- Digital signature guarantees certificate is authentic to Bob
- *CA* can be off-line: Bob can get certificate via Alice!

So can certificates be sent over insecure channels?

Other natural questions:

- How does Bob know *CA* and verifies the *CA* signature?
- What are Alice/Bob actually trusting the *CA* to do?

[1] See here https://datatracker.ietf.org/doc/html/rfc8017#appendix-C for some examples

Verifying a Public-Key Certificate

Suppose Alice sends Bob a public-key certificate with:

- Alice's identity and public key
- A validity period (start and end dates)
- Some additional meta-information
- All signed by certification authority *CA*

This is what Bob should do:

1. Check Alice's identity is correct (e.g., DNS name for server)
2. Check current time is within validity period
3. Check meta-information makes sense for application
4. Check *CA* is *trustworthy* to certify this public-key
5. Obtain *CA*'s public key and verify signature in certificate

The first three are self-explanatory. *PKI* solves 4 and 5.

Sanity check: did you understand how this works?

Who sends and who receives/validates a PK certificate in:

- Asymmetric encryption:
  - Public key belongs to receiver
  - Sender must get certificate beforehand
- Digital signatures
  - Public key belongs to signer
  - OK to sign and send certificate along (M, σ)
- Key agreement
  - If mutually authenticated, then both must send certificates
  - What happens usually in TLS?

Example: in S/MIME (signed email) clients usually

- Allow signing a message as soon as personal certificate installed
- Needs signed message from Alice before allowing encryption
- Does this make sense?

Technical details about public-key certificates

Standardized in X.509 and transposed to internet by IETF

Important data structures have unique object identifiers

Current version is 3, which includes basic fields:

- subject, issuer, validity, public key info, serial

Extensions (attachments), some of which may be marked critical

- all extensions carry an object identifier
- if marked critical but not recognized $\Rightarrow$ reject!

Important extensions:

- Subject/authority key identifier: fingerprint of public key
- Basic constraints: flag that signals special CA certificate
- Key usage: CA can restrict purpose of certificate

Public Key Infrastructure

Public Key Infrastructure

[Wikipedia]

A public key infrastructure (PKI) is a set of roles, policies, hardware, software and procedures needed to create, man- age, distribute, use, store and revoke digital certificates.

All of these components serve a purpose and follow rules so that:

- A certificate user (end entity) can be assured
- By a trustworthy certification authority
- That a PK belongs to another end entity (person, server,...)
- And can be used for a given purpose
- Under well-defined rights/responsibilities for all parties

PKI Architecture

```
+---+
| C |                      +------------+
| e | <------------------->| End entity |
| r |      Operational     +------------+
| t |      transactions          ^
| i |     and management         |  Management
| f |      transactions          |  transactions       PKI
| i |                            |                    users
| c |                            v
| a | ======================  +--+------------+  ==============
| t |                         ^               ^
| e |                         |               |         PKI
|   |                         v               |     management
| & |                      +------+           |      entities
|   | <--------------------|  RA  |<----+     |
| C |  Publish certificate +------+     |     |
| R |                                   |     |
| L |                                   |     |
|   |                                   v     v
| R |                               +------------+
| e | <-----------------------------|     CA     |
| p |   Publish certificate         +------------+
| o |   Publish CRL                    ^      ^
| s |                                  |      |  Management
| i |               +------------+     |      |  transactions
| t | <-------------| CRL Issuer |<----+      |
| o |   Publish CRL +------------+            v
| r |                                     +-------+
| y |                                     |  CA   |
+---+                                     +-------+
```

Operational/Management transactions

How do certificates go around?
Operational protocols specify how certificates are:

- stored in repositories (e.g., LDAP)
- transferred to client software (HTTP, FTP, MIME)
- encoded in non-ambiguous formats

You have seen several instances of operational protocols:

- In TLS the RFC specifies how certificates are exchanged
- In S/MIME certificates are included in the PKCS#7 attachments
- OS certificates are managed via standard cryptographic modules

PKI Management: Initialization

We asked an important question before:

- How do users get to know a *CA*
- How does Bob verify a *CA* signature in a certificate?

Answer:

- All public keys are encoded in X.509 certificates
- **Some certificates contain the public keys of *CA*s**
- Bob obtains the *CA*'s public key from a certificate
- Bob uses the *CA*'s PK to verify signature on Alice's certificate
- If certificate OK $\implies$ Bob can use Alice's public key

Therefore, Alice's public key is authenticated if:

- Bob has certificate for *CA* that issued Alice's certificate
- Bob trusts CA to have checked data on Alice's certificate

PKI Management: Initialization (2)

How does Bob know to trust *CA*?
In the simplest settings:

- Bob gets certificate directly from *CA*
- Bob implicitly trusts *CA* certificate

Examples:

- We get many *CA* certificates pre-installed in OS
- Portuguese citizen's card is certified by state-run *CA*

These are examples of initialization operations.
Key generation, if done by the end entity, also part of initialization.

PKI Management: Registration and Certification

Registration Authorities (RA):

- Front-end: direct contact with end-entities
- Responsible for checking data that goes into certificates
- Responsible for ensuring (unique) entity possesses secret key

Certification Authorities:

- Back-end: infrastructure where certificates are signed
- Typically high-security: air gaps, physical security, etc.

Example: Portuguese Citizen's Card

- RA is Registo Civil, Loja do Cidadão, etc.
- CA is deployed in protected facilities at INCM
- CA generates keys, signs certificates and issues smartcards
- RA delivers them to citizens after physical identification

PKI Management: Revokation

Certificates outside of validity dates are, by definition, invadid.
What happens if they need to be invalidated?

- E.g., lost secret key, data breach, meta-data becomes incorrect.

Certificates need to be revoked while they still look valid.
This is formally done using Certificate Revokation Lists (CRL):

- CA periodically publishes a black-list of revoked certificates
- Certificate consumers should check most-recent CRL
- Exceptional CRL may also be published, as best-effort

How do we get revokation information?
Certificate extensions typically indicate URLs for CRLs
Traditionally low support from client software

PKI Management: Revokation (2)

Three solutions used in the real-world.

1. Trusted Service Provider Lists (TSL):
   - up to date white list of trusted certificates
   - closed small groups (e.g., banking) and high-security applications
2. On-line Certificate Status Protocol (OCSP)):
   - a trusted server checks CRLs for you
   - usually managed by CAs themselves
   - typically used in large organizational contexts (e.g., eGov)
3. Certificate pinning:
   - web servers/browsers/applications carry their own white lists
   - identify good certificates for important entities (e.g., Google)

Certificate Chains and CA Hierarchy

We have seen a simple case: Bob trusts Alice's CA implicitly.
In general this is not the case:

- Bob is initialized with certificates for root CAs
- Bob trusts implicitly in these CAs
- Certificates for root CAs are self-signed:
  - CA generates a key pair (sk, pk)
  - CA creates its own certificate with subject = issuer = CA name
  - Certificate includes pk and CA signs it with sk

Note: self-signed certificates can be generated by anyone.
Validating a self-signed certificate implies:

- belief that whoever owns that secret key is a CA
- belief that this CA only generates *good* certificates

Certificate Chains and CA Hierarchy (2)

Root CAs typically do not issue end-entity certificates.

- There is a hierarchy of CAs
- If CA A signs certificate of CA B
- Then trust in CA B $\leq$ trust in CA A

We can have many levels in this hierarchy/tree, so:

- To authenticate Alice's public key, Bob gets Alice's certificate
- To validate Alice's certificate, Bob gets certificate of Alice's CA
- Bob verifies that Alice's certificate is valid wrt Alice's CA

Bob still needs to decide whether to trust Alice's CA.
Trust = Alice's CA is descendent of Root CA trusted by Bob

Certificate Chains and CA Hierarchy (3)

Bob enters a loop starting with Current CA = Alice's CA.
The loop works as follows:

- **If** Bob implicitly trusts Current CA certificate: **Accept**!
- **Else If** Current CA is subordinate to some $\widehat{CA}$
  - Bob gets $\widehat{CA}$ certificate
  - Bob verifies Current CA certificate is valid wrt $\widehat{CA}$
  - Bob re-enters loop with Current $CA = \widehat{CA}$
- **Else Reject**!

Note: this process fails if Bob cannot get certificates

- All certificates can be sent by Alice except the root of trust.

Certificate Policies

PKI can be used to give cryptography a legal meaning.
A Certificate Policy is a set of PKI operation rules:

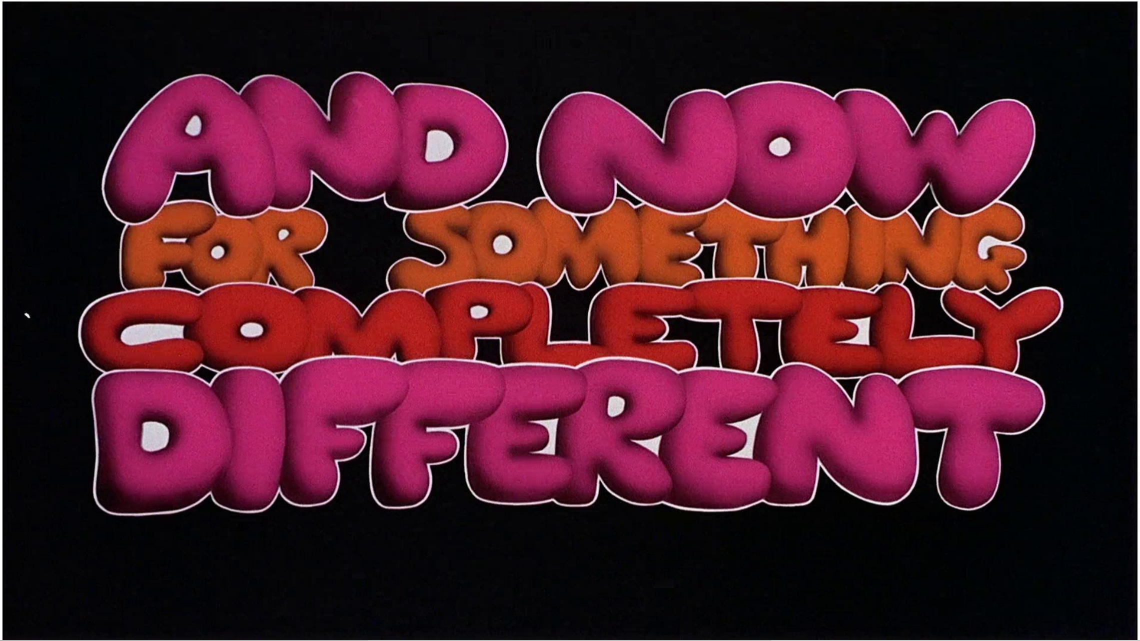
- Rights and responsibilities of end-entities
- Rights and responsibilities of *CA*s

These rights and responsibilities can be written in law.
A certificate policy is assigned an object identifier (*OID*):

- Certificates can be flagged to comply with a policy

This implies an accreditation system:

- *CA* must be audited before it is authorized to use *OID*
- Any *CA* that uses *OID* without authorization is breaking the law


AND NOW
FOR SOMETHING
COMPLETELY
DIFFERENT

Homomorphic Encryption

Shortly after the original RSA paper, a question was posed by Rivest, Adleman, and Dertouzos: *would it be possible to have a database of encrypted information (such as financial or health data), stored in an external location, that would nonetheless allow computations on the encrypted data without decrypting it?* This would permit, for example, external storage, and computation on the encrypted data stored at the external site, without having to trust the owner or operator of the external site.

The idea of "cloud computing" is not an invention of the last years!

Somewhat and Fully Homomorphic Encryption

There are a number of cryptosystems that have been characterized as **somewhat homomorphic**. Paillier's scheme is an example of that type of cryptography as it is only homomorphic for one operation.
Others, that support ring homomorphisms, i.e. support two operations, are called **fully homomorphic** encryptions. These normally rely heavily on algebra of ideals, thus a little more exigent in algebra knowledge.

Paillier's scheme

Let $n = pq$, product of two primes of equal bit size, and such that

$$(p, q-1) = 1 \wedge (p-1, q) = 1.$$

Then, $\mathbb{Z}_n \times \mathbb{Z}_n^*$ is isomorphic to $\mathbb{Z}_{n^2}^*$, given by $f(a,b) = (1+n)^a b^n \mod n^2$. why?
As a consequence a uniform element $y \in \mathbb{Z}_{n^2}^*$ corresponds to a uniform element $(a,b) \in \mathbb{Z}_n \times \mathbb{Z}_n^*$.
Call $y \in \mathbb{Z}_{n^2}^*$ an nth residue module n^2 if exists $x \in \mathbb{Z}_{n^2}^*$ with $y = x^n \mod n^2$. Denote the set of the nth residues modulo n^2 by $Res(n^2)$. Let us characterize the nth residues in $\mathbb{Z}_{n^2}^*$. Taking any $x \in \mathbb{Z}_{n^2}^*$ with $x \leftrightarrow (a,b)$ and raising it to the nth power, we have

$$(x^n \mod n^2) \leftrightarrow (a,b)^n = (na \mod n, b^n \mod n) = (0, b^n \mod n).$$

Moreover, we claim that that any element y that $y \leftrightarrow (0,b)$ is a nth residue.

To see this recall $(n, \Phi(n)) = 1$ thus $d = (n^{-1} \mod \Phi(n))$ exists.So

$$(a, (b^d \mod n))^n = (na \mod n, b^{dn} \mod n) = (0, b) \leftrightarrow y$$

for any $a \in \mathbb{Z}_n$. Thus

$$Res(n^2) = \{y | b \in \mathbb{Z}_n^* \wedge y \leftrightarrow (0, b)\}.$$

This shows that the number of nth roots of any $y \in Res(n^2)$ is exactly n and computing the nth power is an n-to-1 function. As such, if $r \in \mathbb{Z}_{n^2}^*$ is uniform then $(r^n \mod n^2)$ is a uniform element of $Res(n^2)$. The *decision composite residuosity problem* is to distinguish a uniform element of $\mathbb{Z}_{n^2}^*$ from a uniform element of $Res(n^2)$.

Formally, let $\textsc{GenModulus}$ be a polynomial-time algorithm that, on input 1^n, outputs (n, p, q), where $n = pq$, and p, q are ℓ-bit primes (except with a probability negligible in ℓ). Then

Definition

The *decisional composite residuosity problem* is hard relative to $\textsc{GenModulus}$ if for all probabilistic polynomial-time algorithms D there exists a negligible function ε s.t.

$$|Pr[D(n, (r^n \mod n^2)) = 1] - Pr[D(n, r) = 1]| \leq \varepsilon(n),$$

where in each case the probabilities are taken over the experiment in which $\textsc{GenModulus}(1^n)$ outputs (n, p, q) and then a uniform $r \in \mathbb{Z}^*_{n^2}$ is chosen.

The *decisional composite residuosity (DCR) assumption* is the assumption that there is a $\textsc{GenModulus}$ relative to which the decisional composite residuosity is **hard**.

This suggests the following way to encrypt a message $m \in \mathbb{Z}_n$ with respect to a public key n: choose a uniform nth residue $(0, r)$ and set the cyphertext to

$$c \leftrightarrow (m, 1)(0, r) = (m + 0, 1 \cdot r) = (m, r).$$

Since a uniform nth residue $(0, r)$ cannot be distinguished from a uniform element (r', r), the cyphertext is indistinguishable (from the point of view of someone that does not know how to factorise n) from the cybertext

$$c' \leftrightarrow (m, 1) \cdot (r', r) = ((m + r' \mod n), r)$$

for uniform $r' \in \mathbb{Z}_n$ and $r \in \mathbb{Z}_n^*$. As $(m + r \mod n)$ is uniformly distributed in $\mathbb{Z}_n$, c' is independent of the message m.

Encryption: The sender generates the cyphertext $c \in \mathbb{Z}^*_{n^2}$ by choosing a uniform $r \in \mathbb{Z}^*_n$ and then computing

$$c = ((1+n)^m r^n \mod n^2).$$

Observe that

$$c = (((1+n)^m 1^n)((1+n)^0 r^n) \mod n^2) \leftrightarrow (m,1)\cdot(0,r),$$

thus,

$$c \leftrightarrow (m,r)$$

as required.

Decryption: Now, knowing $n = pq$, we claim to be able to decrypt efficiently c and recover m using this steps.

$$\begin{aligned}\hat{c} &= (c^{\Phi(n)} \mod n^2),\\ \hat{m} &= (\hat{c} - 1)/n,\\ m &= (\hat{m}\,\Phi(n)^{-1} \mod n).\end{aligned}$$

Let us see why this works. Let $c \leftrightarrow (m, r)$, for arbitrary $r \in \mathbb{Z}_n^*$. Then

$$\begin{aligned}\hat{c} &= (c^{\Phi(n)} \mod n^2)\\ &\leftrightarrow (m, r)^{\Phi(n)}\\ &= ((m\,\Phi(n) \mod n, r^{\Phi(n)} \mod n)\\ &= (m\,\Phi(n) \mod n, 1).\end{aligned}$$

This means that $\hat{c} = (1+n)^{(m\,\Phi(n) \mod n)} \mod n^2$, and we know

$$\hat{c} = (1+n)^{(m\,\Phi(n) \mod n)} = (1 + (m\,\Phi(n) \mod n)n) \mod n^2$$

(we proved that $(1+n)^a \equiv 1 + an \pmod{n^2}$)
Since $1 + (m\,\Phi(n) \mod n)n < n^2$ we can drop $\pmod{n^2}$. Thus

$$\hat{m} = (\hat{c} - 1)/n = (m\,\Phi(n) \mod n).$$

Finally,

$$m = (\hat{m}\,\Phi(n)^{-1} \mod n).$$

($\Phi(n)$ is invertible modulo n since $(\Phi(n), n) = 1$)

An example

Let $n = 11 \cdot 17 = 187$ ($n^2 = 34969$), and let $m = 175$.
Choosing $r = 83 \in \mathbb{Z}^*_{187}$ we compute

$$c = ((1+187)^{175} \cdot 83^{187} \mod 34969) = 23911 \leftrightarrow (175, 83).$$

To decrypt, knowing $\Phi(187) = 10 \cdot 16 = 160$. So

$$\begin{aligned} \hat{c} &= (23911^{160} \mod 34969) = 25620, \\ \hat{m} &= (25620 - 1)/187 = 137, \quad \text{since } 90 = (160^{-1} \mod 187), \\ m &= (137 \cdot 90 \mod 187) = 175. \end{aligned}$$

Paillier as a homomorphic encryption

The Paillier encryption scheme is useful in a number of settings because it is homomorphic. Roughly, a homomorphic encryption scheme enables (certain) computations to be performed on encrypted data, yielding a ciphertext containing the encrypted result. In the case of Paillier encryption, the computation that can be performed is (modular) addition. Fix a private key $n = pq$. Then the Paillier scheme has the property that multiplying an encryption of m_1 and an encryption of m_2 (done $\pmod{n^2}$) results in an encryption of $m_1 + m_2 \mod n$; this is because

$$((1+n)^{m_1} r_1^n)((1+n)^{m_2} r_2^n) \equiv (1+n)^{(m_1+m_2 \mod n)} (r_1 r_2)^n \pmod{n^2}.$$

A nice feature of Paillier encryption is that it is homomorphic over a large additive group $\mathbb{Z}_n$. To see an application of this, consider the following distributed voting scheme, where voters can vote "no" or "yes" and the goal is to tabulate the number of "yes" votes:

1. A voting authority generates a public key n for the Paillier encryption scheme and publicizes n.
2. Let 0 stand for a "no" and let 1 stand for a "yes". Each voter casts their vote by encrypting it. That is, voter i casts her vote v_i by computing $c_i = (1+n)^{v_i}(r_i)n \mod n^2$ for a uniform $r_i \in \mathbb{Z}_n^*$.
3. Each voter broadcasts their vote c_i. These votes are then publicly aggregated by computing $c_{total} = \prod_{i=1}^{\ell} c_i \mod n^2$.
4. The authority is given c_{total}. By decrypting it, the authority obtains the vote total $v_{total} = \sum_{i=1}^{\ell} v_i \mod n$.

If ℓ is small (so that $v_{total} < n$), there is no wrap-around modulo n and $v_{total} = \sum_{i=1}^{\ell} v_i$.


THE
END IS
NEAR

Proposition

Let $n = pq$ the product of two primes of equal length. Then

1. $(n, \Phi(n)) = 1$; Φ
2. $\forall a \geq 0\ (1+n)^a \equiv 1 + an \pmod{n^2}$. As a consequence, $ord(1+n)$ in $\mathbb{Z}^*_{n^2}$ is n. That is, $(1+n)^n \equiv 1 \pmod{n^2}$, and $1 \leq a < n \implies (1+n)^a \not\equiv 1 \pmod{n^2}$.
3. $\mathbb{Z}_n \times \mathbb{Z}^*_n$ is isomorphic to $\mathbb{Z}^*_{n^2}$ being the isomorphism

$$\begin{array}{rcl} f : \mathbb{Z}_n \times \mathbb{Z}^*_n & \longrightarrow & \mathbb{Z}^*_{n^2} \\ (a, b) & \longmapsto & (1+n)^a b^n \mod n^2. \end{array}$$

We will write $x \leftrightarrow (a, b)$ if $f(a, b) = x$.

Proof:

1. $(n, \Phi(n)) = 1$
   We know that $\Phi(n) = (p-1)(q-1)$. why? Assume $p > q$, thus $p > p-1 > q-1$. It is clear that $(p, \Phi(n)) = 1$ and $(q, q-1) = 1$. If $(q, p-1) \neq 1$ then $(q, p-1) = q$, since q is prime. But then $(p-1)/q \geq 2$ which contradicts the assumption that p and q have binary representations of the same size. Thus
   $$\left(n, \Phi(n)\right) = 1.$$
2. $\forall a\ a \geq 0 \implies (1+n)^a \equiv (1+an) \pmod{n^2}$, thus $ord(1+n)$ in $\mathbb{Z}^*_{n^2}$ is n. why?
3. The group $\mathbb{Z}_n \times \mathbb{Z}^*_n$ is isomorphic to the group $\mathbb{Z}^*_{n^2}$ with isomorphism
   $$\begin{aligned} f : \mathbb{Z}_n \times \mathbb{Z}^*_n &\longrightarrow \mathbb{Z}^*_{n^2} \\ (a, b) &\longmapsto (1+n)^a b^n \mod n^2 \end{aligned}$$
   why?

back

Proposition

Let p be a prime and a a positive integer, then

$$\Phi(p^a) = p^a - p^{a-1}.$$

Proof: All the non-coprimes with p and its powers are: $p, 2p, \ldots, p^{a-1}p$, i.e., the p^{a-1} multiples of p. Thus the number of coprimes with p^a is

$$p^a - p^{a-1} = \Phi(p^a).$$

□

Proposition

Let a, b be such that $(a, b) = 1$, then

$$\Phi(ab) = \Phi(a)\Phi(b).$$

Proof: Let $m < ab \wedge (m, ab) = 1$. It is easy to see that

$$(m, a) = 1 \wedge (m, b) = 1 \implies (m \mod a, a) = 1 \wedge (m \mod b, b) = 1.$$

Thus, for each integer coprime with ab we have a pair of integers coprime with a and b, respectively. Thus, $\Phi(ab) \leq \Phi(a)\Phi(b)$.
On the other hand, consider a pair (r, s) with $(r, a) = 1 \wedge (s, b) = 1$. As $(a, b) = 1$, by CRT CRT? then exists $m > 0$ s.t. $m \equiv r \pmod{a} \wedge m \equiv s \pmod{b}$. Thus, for each $m \in \{1, 2, \ldots, ab\}$ we get a different pair (r, s), and thus $\Phi(ab) \geq \Phi(a)\Phi(b)$. □

Proposition

Let $p_1, p_2, \ldots, p_k$ be primes and $a_1, a_2, \ldots, a_k$ positive integers. Then,

$$\Phi\left(p_1^{a_1} p_2^{a_2} \cdots p_k^{a_k}\right) = \left(p_1^{a_1} - p_1^{a_1 - 1}\right)\left(p_2^{a_2} - p_2^{a_2 - 1}\right) \cdots \left(p_k^{a_k} - p^{a_k - 1}\right).$$

Proof: (It is straightforward by induction on k.

Corollary

If $n = pq$ with p and q primes

$$\Phi(n) = (p-1)(q-1).$$

back

Theorem (Chinese Remainder Theorem)

Let $n = pq$ where $p, q > 1 \wedge (p, q) = 1$. Then

$$\mathbb{Z}_n \simeq \mathbb{Z}_p \times \mathbb{Z}_q \text{ and } \mathbb{Z}_n^* \simeq \mathbb{Z}_p^* \times \mathbb{Z}_q^*.$$

Moreover, let f be the function mapping elements $x \in \{0, \ldots, n-1\}$ to pairs (x_p, x_q) with $x_p \in \{0, \ldots, p-1\}$ and $x_q \in \{0, \ldots, q-1\}$ defined by

$$f(x) = (x \mod p, x \mod q).$$

Then, f is a n isomorphism from $\mathbb{Z}_n$ to $\mathbb{Z}_p \times \mathbb{Z}_q$, and the restriction of f to $\mathbb{Z}_n^*$ is an isomorphism from $\mathbb{Z}_n^*$ to $\mathbb{Z}_p^* \times \mathbb{Z}_q^*$.

Proof: It is clear that $\forall x \in \mathbb{Z}_n, f(x) = (x_p, x_q)$ with $x_p \in \mathbb{Z}_p$ and $x_q \in \mathbb{Z}_q$. We need to prove that if $x \in \mathbb{Z}_n^*$ then $(x_p, x_q) \in \mathbb{Z}_p^* \times \mathbb{Z}_q^*$. If $x_p \notin \mathbb{Z}_p^*$ then this means that $(x \mod p, p) \neq 1$. But then $(x, p) \neq 1$ and thus $(x, n) \neq 1$ contradicting the fact that $x \in \mathbb{Z}_n^*$.

Let us see that f is one-to-one. Say $f(x) = (x_p, x_q) = f(x')$. Then $x \equiv x_p \equiv x' \pmod{p}$ and $x \equiv x_q \equiv x' \pmod{q}$. This implies that $x - x'$ is divisible both by p and q. As $(p, q) = 1$ then $x - x'$ must be divisible by n which implies that $x \equiv x' \pmod{n}$, which meand $x = x'$. Thus f is one-to-one. Since $|\mathbb{Z}_n| = n = pq = |\mathbb{Z}_p \times \mathbb{Z}_q|$, f is bijective.
That f preserves the operation of the group, let us denote by $\boxplus$ the operation in $\mathbb{Z}_p \times \mathbb{Z}_q$. Then we need to show that

$$f(a+b) = f(a) \boxplus f(b).$$

$$\begin{aligned} f(a+b) &= \big((a+b) \mod p, (a+b) \mod q\big) \\ &= (a \mod p, a \mod q) \boxplus (b \mod p, b \mod q) \\ &= f(a) \boxplus f(b). \end{aligned}$$

□ back

Definition (Euler's Φ function)

Let n be an integer. Then

$$\Phi(n) = \left|\{k | 1 \leq k < n \wedge (n,k) = 1\}\right|.$$

back

Proposition

For all $a \geq 0$, $(1+n)^a \equiv (1+an) \pmod{n^2}$, and thus $ord(1+n)$ in $\mathbb{Z}^*_{n^2}$ is n.

Proof: By the binomial expansion

$$(1+n)^a = \sum_{i=0}^{a} \binom{a}{i} n^i \equiv 1 + an \pmod{n^2}.$$

The smallest nonzero a s.t. $(1+n)^a \equiv 1 \pmod{n^2}$ is, therefore, $a = n$. □ back

Proposition

The group $\mathbb{Z}_n \times \mathbb{Z}_n^*$ is isomorphic to the group $\mathbb{Z}_{n^2}^*$ with isomorphism

$$\begin{array}{rcl} f : \mathbb{Z}_n \times \mathbb{Z}_n^* & \longrightarrow & \mathbb{Z}_{n^2}^* \\ (a, b) & \longmapsto & (1+n)^a b^n \mod n^2 \end{array}$$

Proof: Note that $(1+n)^a b^n$ does not have a factor in common with n^2 since $(1+n, n^2) = 1$ and $(b, n^2) = 1$ (because $b \in \mathbb{Z}_n^*$). Thus, $((1+n)^a b^n \mod n^2) \in \mathbb{Z}_{n^2}^*$.
Now we prove that f is an isomorphism. First we prove that is a bijection. Since

$$|\mathbb{Z}_{n^2}^*| = \Phi(n^2) = p(p-1)q(q-1) = pq(p-1)(q-1) = |\mathbb{Z}_n \times \mathbb{Z}_n^*|$$

it suffices to show that f is one-to-one. Say $a_1, a_2 \in \mathbb{Z}_n$ and $b_1, b_2 \in \mathbb{Z}_n^*$ are s.t. $f(a_1, b_1) = f(a_2, b_2)$

Then

$$(1+n)^{a_1-a_2}(b_1/b_2)^n \equiv 1 \pmod{n^2} \tag{1}$$

(as $b_1, b_2 \in \mathbb{Z}_n^*$ their inverses belong to $\mathbb{Z}_n^*$ too.)
Raising both sides to the power $\Phi(n)$, and using the fact that the order of $\mathbb{Z}_{n^2}^*$ is $\Phi(n^2) = n\Phi(n)$ we get

$$(1+n)^{(a_1-a_2)\Phi(n)}(b_1/b_2)^{n\Phi(n)} \equiv 1 \pmod{n^2} \implies (1+n)^{(a_1-a_2)\Phi(n)} \equiv 1 \pmod{n^2}$$

(because $n\Phi(n) = \Phi(n^2) \implies (b_1/b_2)^{\Phi(n^2)} \equiv 1 \pmod{n^2}$), as we already proved $(1+n)$ has order n modulo n^2,

$$(a_1 - a_2)\Phi(n) \equiv 0 \pmod{n}$$ why?

and so $n \mid (a_1 - a_2)\Phi(n)$. Since $\big(\Phi(n), n\big) = 1$, it follows that $n \mid (a_1 - a_2)$. Since $a_1, a_2 \in \mathbb{Z}_n$, then $a_1 = a_2$. Making $a_1 = a_2$ in (1), we have $b_1 \equiv b_2 \pmod{n^2}$.
This implies $b_1 \equiv b_2 \pmod{n}$ since $b_1, b_2 \in \mathbb{Z}_n^*$. Since $\big(n, \Phi(n)\big) = 1$, exponentiation to the power n is a bijection in $\mathbb{Z}_n^*$ why? This means that $b_1 \equiv b_2 \pmod{n}$, but since $b_1, b_2 \in \mathbb{Z}_n^*$ we have that $b_1 = b_2$. Hence f is one-to-one and, consequently, a bijection.

To show that f is an isomorphism, we show that

$$f(a_1, b_1)f(a_2, b_2) = f(a_1 + a_2, b_1 b_2).$$

Note that the multiplication on the left takes place modulo n^2 while addition/multiplication on the right takes place modulo n. We have

$$\begin{aligned} f(a_1, b_1)f(a_2, b_2) &\equiv ((1+n)^{a_1} b_1^n)((1+n)^{a_2} b_2^n) \pmod{n^2} \\ &\equiv (1+n)^{a_1+a_2}(b_1 b_2)^n \pmod{n^2}. \end{aligned}$$

Since $(1+n)$ has order n modulo n^2, we can write

$$f(a_1, b_1)f(a_2, b_2) \equiv (1+n)^{(a_1+a_2 \mod n)}(b_1 b_2)^n \pmod{n^2}.$$

Let $b_1 b_2 = r + \gamma n$, with $1 \leq r < n$ (cannot be 0 since $b_1, b_2 \in \mathbb{Z}_n^*$). Note that $r = b_1 b_2 \mod n$.

Thus we have

$$
\begin{aligned}
(b_1 b_2)^n &\equiv (r+\gamma n)^n \pmod{n^2} \\
&\equiv \sum_{k=0}^{n} \binom{n}{k} r^{n-k} (\gamma n)^k \pmod{n^2} \\
&\equiv r^n + n r^{n-1} (\gamma n) \pmod{n^2} \\
&\equiv r^n \pmod{n^2} \\
&\equiv (b_1 b_2 \mod n)^n \pmod{n^2}.
\end{aligned}
$$

Thus

$$
\begin{aligned}
f(a_1, b_1) f(a_2, b_2) &= ((1+n)^{(a_1+a_2 \mod n)} (b_1 b_2 \mod n)^n) \mod n^2 \\
&= f(a_1 + a_2, b_1 b_2).
\end{aligned}
$$

□ back

Proposition

Let G be a finite group and $g \in G$, an element of order i. Then,

$$g^x = g^y \iff x \equiv y \pmod{i}.$$

Proof: If $x = y$ then $(x \mod i) = (y \mod i)$ and $g^x = g^{(x \mod i)} = g^{(y \mod i)} = g^y$.
On the other direction, let $g^x = g^y$. Then $1 = g^{x-y} = g^{(x-y \mod i)}$. Since $(x - y \mod i) < i$ and i is the order of g, then $(x - y \mod i) = 0$. $\square$ back

Theorem

Let G be a finite abelian group with $m = |G|$ the order of the group. Then for any $g \in G$, $g^m = 1$.

Proof: Fix an arbitrary $g \in G$, and let $g_1, \ldots, g_m$ be the elements of G. We claim that

$$g_1 \cdot g_2 \cdots g_m = (gg_1)(gg_2) \cdots (gg_m).$$

To see this note that $gg_i = gg_j$ implies $g_i = g_j$, thus each element in parenthesis of the right-hand are distinct. Because there are exactly m elements in both sides of the equality they are just a permutation of eachother. Thus

$$g_1 \cdot g_2 \cdots g_m = g^m \cdot (g_1 \cdot g_2 \cdots g_m),$$

which implies that $g^m = 1$. □

Corollary

Let G be a finite group with $m = |G| > 1$. Then for any $g \in G$ and any integer x, we have $g^x = g^{(x \mod m)}$.

Proof: Say $x = qm + r$, where q, r are integers and $r = (x \mod m)$. Then

$$g^x = g^{qm+r} = g^{qm} g^r = g^r. \quad \square$$

Corollary

Let G be a finite group with $m = |G| > 1$. Let $e > 0$, and define the function $f_e : G \to G$ by $f_e(g) = g^e$. If $(e, m) = 1$ the f_e is a bijection. Moreover if $d = e^{-1} \mod m$ the f_d is the inverse of f_e.

Proof: Since G is finite, the second part implies the first; thus, we only need to show $f_d = f_e^{-1}$. This is true because for every $g \in G$

$$f_d(f_e(g)) = f_d(g^e) = g^{de} = g^{(de \mod m)} = g^1 = g. \quad \square$$

back