Modular Exponentiation

Michael Levin

Computer Science Department, Higher School of Economics

Outline

Fast Modular Exponentiation

Fermat's Little Theorem

Euler's Totient Function

Euler's Theorem

Modular Exponentiation

- The central operation for public key cryptography
- Properties allow fast encryption and decryption for Alice and Bob
- Easy in one direction, hard in reverse — for Eve
- Computing modular exponent quickly
- Key properties for encryption and decryption

Modular Exponentiation

$$c \leftarrow b^e \pmod{m}$$

- How to compute $b^e \bmod m$?

- How to compute $b^e \bmod m$?
- No need to compute the giant number b^e and divide by m: we can start with 1, then multiply by b and immediately take the result modulo m, repeat e times

$b = 7, e = 4, m = 11$:

$$c \leftarrow 1$$

$$c \leftarrow (c \cdot b) \equiv (1 \cdot 7) \equiv 7 \bmod 11 = 7$$

$b = 7, e = 4, m = 11$:

$$c \leftarrow 7$$
$$c \leftarrow (c \cdot b) \equiv (7 \cdot 7) \equiv 49 \bmod 11 = 5$$

$b = 7, e = 4, m = 11$:

$$c \leftarrow 5$$

$$c \leftarrow (c \cdot b) \equiv (5 \cdot 7) \equiv 35 \bmod 11 = 2$$

$b = 7, e = 4, m = 11$:

$$c \leftarrow 2$$
$$c \leftarrow (c \cdot b) \equiv (2 \cdot 7) \equiv 14 \bmod 11 = 3$$

$b = 7, e = 4, m = 11$:

$$c \leftarrow 2$$
$$c \leftarrow (c \cdot b) \equiv (2 \cdot 7) \equiv 14 \bmod 11 = 3$$
$$b^e \bmod m = 7^4 \bmod 11 = 3$$

Straightforward Algorithm

- Start with $c \leftarrow 1$
- Repeat e times: $c \leftarrow c \cdot b \bmod m$

- Just e multiplications

- Just e multiplications
- Fast, right?

- Just e multiplications
- Fast, right?
- What if b, e, m are integers with 1000 digits?

- Just e multiplications
- Fast, right?
- What if b, e, m are integers with 1000 digits?
- Can we do faster?

What if $e = 2^k$?

$b = 7, e = 128, m = 11$:

$$\begin{aligned} 7^1 &\equiv 7 \bmod 11 = 7 \\ 7^2 \equiv 7 \cdot 7 &\equiv 49 \bmod 11 = 5 \end{aligned}$$

$b = 7, e = 128, m = 11$:

$$\begin{aligned} 7^2 &\equiv 7 \cdot 7 \equiv 49 \bmod 11 = 5 \\ 7^4 &\equiv 7^2 \cdot 7^2 \equiv 5 \cdot 5 \equiv 25 \bmod 11 = 3 \end{aligned}$$

$b = 7, e = 128, m = 11$:

$$7^4 \equiv 7^2 \cdot 7^2 \equiv 5 \cdot 5 \equiv 25 \bmod 11 = 3$$
$$7^8 \equiv 7^4 \cdot 7^4 \equiv 3 \cdot 3 \equiv 9 \bmod 11 = 9$$

$b = 7, e = 128, m = 11$:

$$
\begin{aligned}
7^8 &\equiv 7^4 \cdot 7^4 \equiv 3 \cdot 3 \equiv 9 \bmod 11 = 9 \\
7^{16} &\equiv 7^8 \cdot 7^8 \equiv 9 \cdot 9 \equiv 81 \bmod 11 = 4
\end{aligned}
$$

$b = 7, e = 128, m = 11$:

$$\begin{aligned} 7^{16} &\equiv 7^8 \cdot 7^8 \equiv 9 \cdot 9 \equiv 81 \bmod 11 = 4 \\ 7^{32} &\equiv 7^{16} \cdot 7^{16} \equiv 4 \cdot 4 \equiv 16 \bmod 11 = 5 \end{aligned}$$

$b = 7, e = 128, m = 11$:

$$7^{32} \equiv 7^{16} \cdot 7^{16} \equiv 4 \cdot 4 \equiv 16 \bmod 11 = 5$$
$$7^{64} \equiv 7^{32} \cdot 7^{32} \equiv 5 \cdot 5 \equiv 25 \bmod 11 = 3$$

$b = 7, e = 128, m = 11$:

$$7^{64} \equiv 7^{32} \cdot 7^{32} \equiv 5 \cdot 5 \equiv 25 \bmod 11 = 3$$
$$7^{128} \equiv 7^{64} \cdot 7^{64} \equiv 3 \cdot 3 \equiv 9 \bmod 11 = 9$$

- Start with $c \leftarrow b \bmod m$
- Repeat k times: $c \leftarrow c^2 \bmod m$
- In the end, $c = b^{2^k} = b^e \bmod m$

- Just $k = \log_2 2^k = \log_2 e$ multiplications — much faster!

- Just $k = \log_2 2^k = \log_2 e$ multiplications — much faster!
- What if e is not a power of 2?

$$b, b^2, b^4, b^8, \ldots$$

$$b, b^2, b^4, b^8, \ldots$$

$$b^{13}?$$

$$b, b^2, b^4, b^8, \ldots$$

$$b^{13}?$$

$$13 = 8 + 4 + 1$$

$$b, b^2, b^4, b^8, \ldots$$

$$b^{13}?$$

$$13 = 8 + 4 + 1$$

$$b^{13} = b^8 \cdot b^4 \cdot b^1$$

$b = 7, e = 13, m = 11$:

$b = 7, e = 13, m = 11$:

$$e = 13 = 8 + 4 + 1 = 1101_2 = 2^3 + 2^2 + 2^0$$

$b = 7, e = 13, m = 11$:

$$e = 13 = 8 + 4 + 1 = 1101_2 = 2^3 + 2^2 + 2^0$$

$b^{2^0} \bmod m$
$b^{2^1} \bmod m$
$b^{2^2} \bmod m$
$b^{2^3} \bmod m$
$b^{13} \bmod m$

$b = 7, e = 13, m = 11$:

$$e = 13 = 8 + 4 + 1 = 1101_2 = 2^3 + 2^2 + 2^0$$

$b^{2^0} \bmod m$	7
$b^{2^1} \bmod m$	5
$b^{2^2} \bmod m$	3
$b^{2^3} \bmod m$	9
$b^{13} \bmod m$	

$b = 7, e = 13, m = 11$:

$$e = 13 = 8 + 4 + 1 = 1101_2 = 2^3 + 2^2 + 2^0$$

$b^{2^0} \bmod m$	7	$\times$
$b^{2^1} \bmod m$	5	
$b^{2^2} \bmod m$	3	$\times$
$b^{2^3} \bmod m$	9	$\times$
$b^{13} \bmod m$		

$b = 7, e = 13, m = 11$:

$$e = 13 = 8 + 4 + 1 = 1101_2 = 2^3 + 2^2 + 2^0$$

$b^{2^0} \bmod m$	7	$\times$
$b^{2^1} \bmod m$	5	
$b^{2^2} \bmod m$	3	$\times$
$b^{2^3} \bmod m$	9	$\times$
$b^{13} \bmod m$		

$$7 \cdot 3 \cdot 9 \equiv 21 \cdot 9 \equiv 10 \cdot 9 \equiv 90 \equiv 2 \bmod 11$$

$b = 7, e = 13, m = 11$:

$$e = 13 = 8 + 4 + 1 = 1101_2 = 2^3 + 2^2 + 2^0$$

$b^{2^0} \bmod m$	7	$\times$
$b^{2^1} \bmod m$	5	
$b^{2^2} \bmod m$	3	$\times$
$b^{2^3} \bmod m$	9	$\times$
$b^{13} \bmod m$	2	

$$7 \cdot 3 \cdot 9 \equiv 21 \cdot 9 \equiv 10 \cdot 9 \equiv 90 \equiv 2 \bmod 11$$

- Rewrite e in binary form: $e = 1101\dots01_2$
- Compute $b^{2^k} \bmod m$ for all $2^k \leq e$
- Multiply together the results for all 2^k in the binary representation of e
- $\log_2 e$ multiplications to compute all $b^{2^k} \bmod m$
- At most $\log_2 e$ multiplications to compute $b^e \bmod m$ by multiplying all the needed $b^{2^k} \bmod m$
- At most $2\log_2 e$ multiplications in total!

Conclusion

- Modular exponentiation can be computed using at most $2\log_2 e$ multiplications
- Represent e in binary form
- Compute $b^{2^k} \bmod m$ for all $2^k \leq e$ using squaring
- Compute $b^e \bmod m$ as a product of all needed b^{2^k} using binary representation of e

Outline

Fast Modular Exponentiation

Fermat's Little Theorem

Euler's Totient Function

Euler's Theorem

Fermat's Little Theorem

- Key result for cryptography using modular exponentiation
- Key result for fast algorithms testing whether a large integer is prime
- Can be used to make modular exponentiation even faster

Fermat's Little Theorem

Theorem

If prime p doesn't divide integer a, then
$a^{p-1} \equiv 1 \bmod p$.

Proof

- Consider all $p-1$ non-zero remainders modulo p: $1, 2, \ldots, p-1$

Proof

- Consider all $p-1$ non-zero remainders modulo p: $1, 2, \ldots, p-1$
- Multiplying any such remainder by a is invertible since $p \not| a$, so the new remainder is also non-zero

Proof

- Consider all $p-1$ non-zero remainders modulo p: $1, 2, \ldots, p-1$
- Multiplying any such remainder by a is invertible since $p \nmid a$, so the new remainder is also non-zero
- Graph on remainders: edge from r to ar

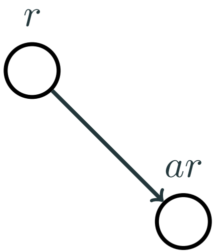
Proof

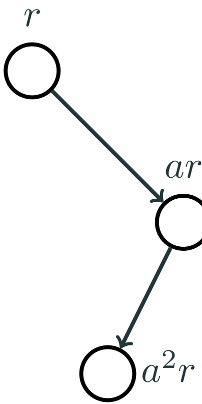
- Consider all $p-1$ non-zero remainders modulo p: $1, 2, \ldots, p-1$
- Multiplying any such remainder by a is invertible since $p \nmid a$, so the new remainder is also non-zero
- Graph on remainders: edge from r to ar
- All incoming and outgoing degrees are 1

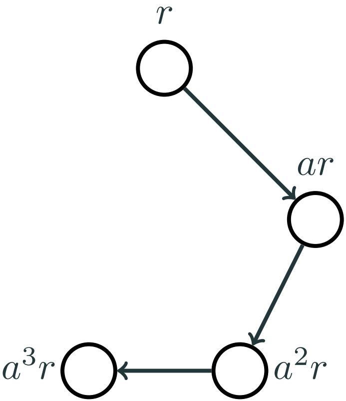
Proof

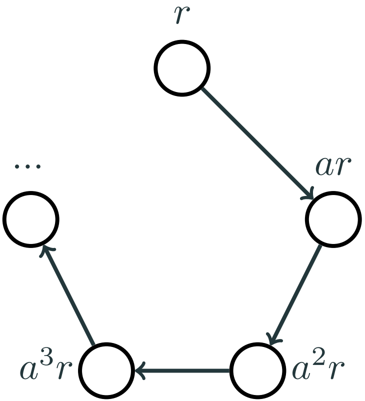
- Consider all $p-1$ non-zero remainders modulo p: $1, 2, \ldots, p-1$
- Multiplying any such remainder by a is invertible since $p \not| a$, so the new remainder is also non-zero
- Graph on remainders: edge from r to ar
- All incoming and outgoing degrees are 1
- What happens if we start with r, multiply it by a, take remainder modulo p, repeat several times?


r

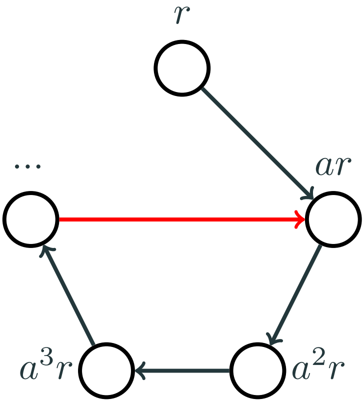

r
ar

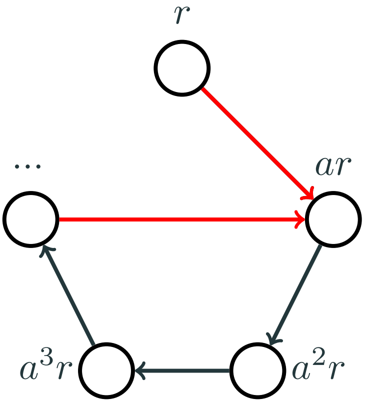

r
ar
a^2r

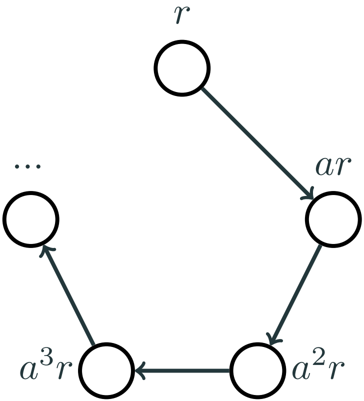

r
ar
a^3r
a^2r

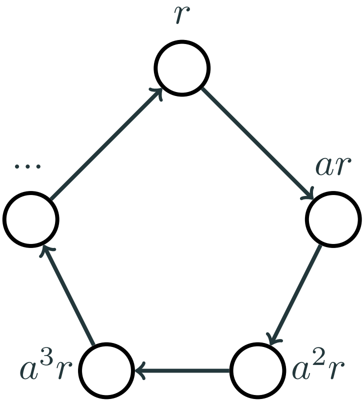


r
ar
a^2r
a^3r
$\cdots$

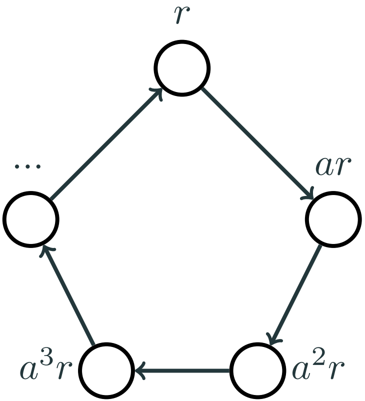

r
ar
$\cdots$
a^2r
a^3r


r
ar
$\cdots$
a^3r
a^2r

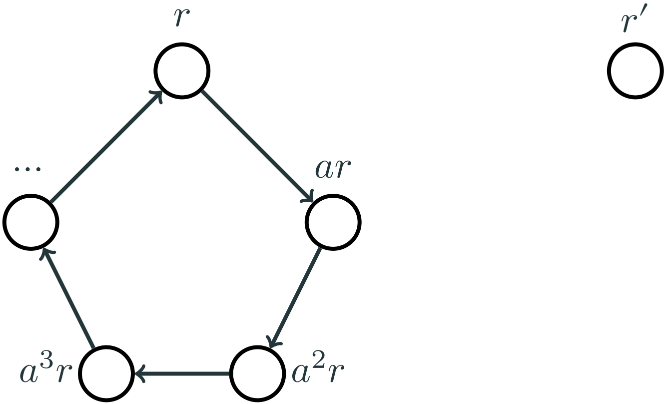

r
ar
a^2r
a^3r
$\cdots$



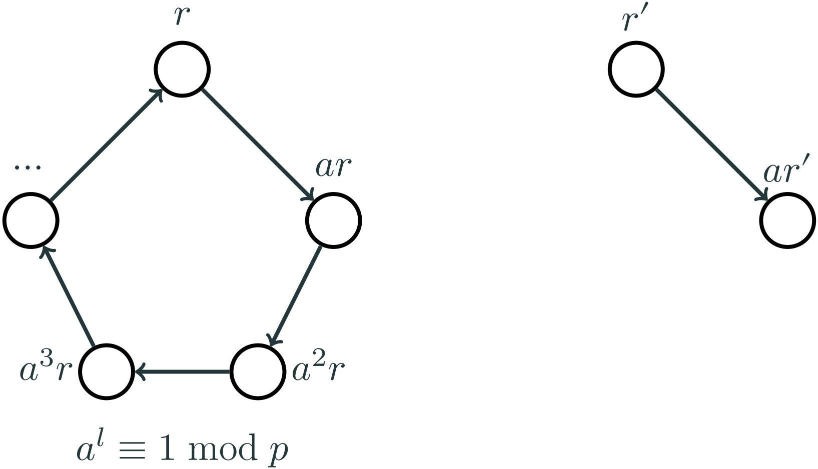
r
ar
a^2r
a^3r
$\cdots$



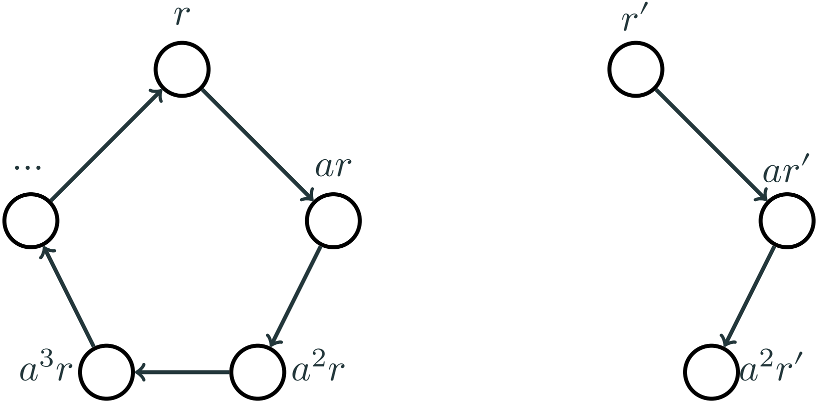
$$a^l \equiv 1 \bmod p$$



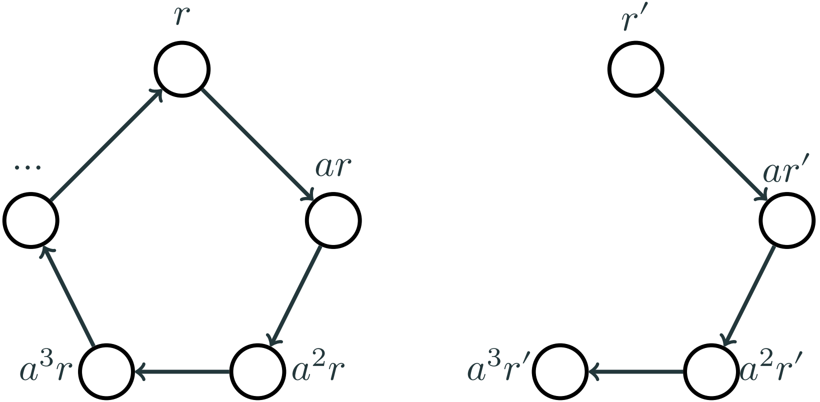
$a^l \equiv 1 \bmod p$



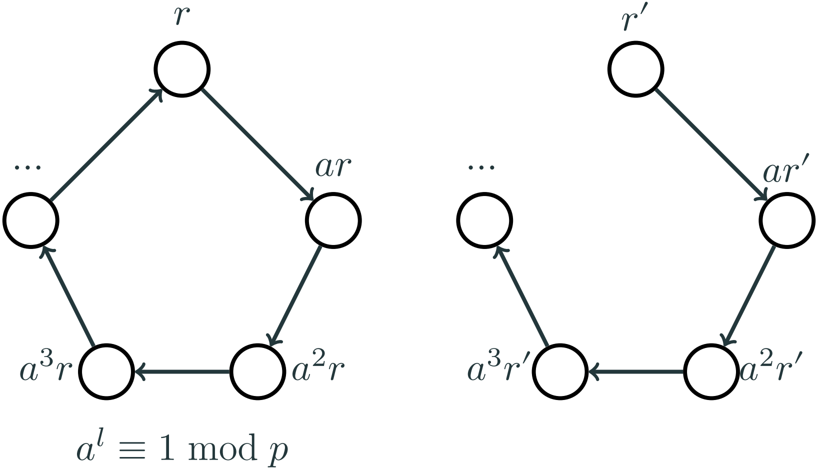
r
ar
a^2r
a^3r
$\dots$
$a^l \equiv 1 \bmod p$
r'
ar'

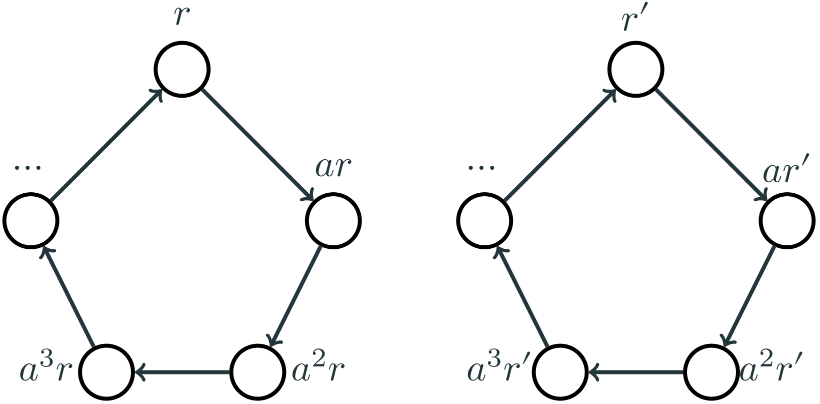


$$a^l \equiv 1 \bmod p$$



$a^l \equiv 1 \bmod p$


r
ar
a^2r
a^3r
$\dots$
$a^l \equiv 1 \bmod p$
r'
ar'
a^2r'
a^3r'
$\dots$



$a^l \equiv 1 \bmod p$ $\qquad$ $a^l \equiv 1 \bmod p$

Proof

- Starting with any r, we get a cycle of some length l

Proof

- Starting with any r, we get a cycle of some length l
- This means $a^l \equiv 1 \bmod p$

Proof

- Starting with any r, we get a cycle of some length l
- This means $a^l \equiv 1 \bmod p$
- Starting with any r' we get a cycle of length l

Proof

- Starting with any r, we get a cycle of some length l
- This means $a^l \equiv 1 \bmod p$
- Starting with any r' we get a cycle of length l
- These cycles don't intersect and contain all $p - 1$ non-zero remainders

Proof

- Starting with any r, we get a cycle of some length l
- This means $a^l \equiv 1 \bmod p$
- Starting with any r' we get a cycle of length l
- These cycles don't intersect and contain all $p-1$ non-zero remainders
- If there are c cycles, then $cl = p - 1$

Proof

- Starting with any r, we get a cycle of some length l
- This means $a^l \equiv 1 \bmod p$
- Starting with any r' we get a cycle of length l
- These cycles don't intersect and contain all $p-1$ non-zero remainders
- If there are c cycles, then $cl = p-1$
- $a^{p-1} = a^{cl} = (a^l)^c \equiv 1^c \equiv 1 \bmod p$ $\square$

Optimizing Modular Exponentiation

- If $p \not| \, a$, $a^{p-1} \equiv 1 \bmod p$
- $a^n \equiv a^{n \bmod (p-1)} \bmod p$
- If $p \mid a$, $a^n \equiv 0 \equiv a^{n \bmod (p-1)} \bmod p$
- For any a and n, $a^n \equiv a^{n \bmod (p-1)} \bmod p$
- Compute only powers up to $p - 1$

Outline

Fast Modular Exponentiation

Fermat's Little Theorem

Euler's Totient Function

Euler's Theorem

Euler's Totient Function

- Key for the RSA encryption
- Easy to compute if factorization of n is known
- No fast algorithms are known to compute if factorization of n is unknown
- No fast algorithms are known for factorization of integers
- Easy to compute with some private information, but no known way to compute without it — this is the key property for cryptography

Euler's Totient Function

Definition

Euler's totient function $\phi(n)$ counts integers between 0 and $n-1$ which are coprime with n

$n = 1$

0 is coprime with $n = 1$, so $\phi(1) = 1$

$n = 2$

0 is not coprime with 2, and 1 is coprime with 2, so $\phi(2) = 1$

$n = 3$

1 and 2 are coprime with 3, so $\phi(3) = 2$

$n = 10$

$1, 3, 7$ and 9 are coprime with 10, so $\phi(10) = 4$

Lemma

If p is prime, $\phi(p) = p - 1$.

Proof

0 is not coprime with p, and $1, 2, 3, \ldots, p-1$ are coprime with p, so $\phi(p) = p - 1$. $\square$

Lemma

If p and q are prime, then $\phi(pq) = (p-1)(q-1)$.

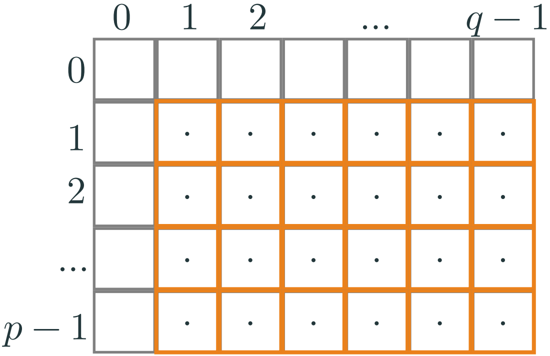
Proof

- Consider all pq remainders modulo pq
- By Chinese Remainder Theorem, each r corresponds to pair (r_p, r_q)
- $\mathrm{GCD}(pq, r) = 1 \Leftrightarrow r_p, r_q \neq 0$

	0	1	2		$\dots$		$q-1$
0							
1							
2							
$\dots$							
$p-1$							

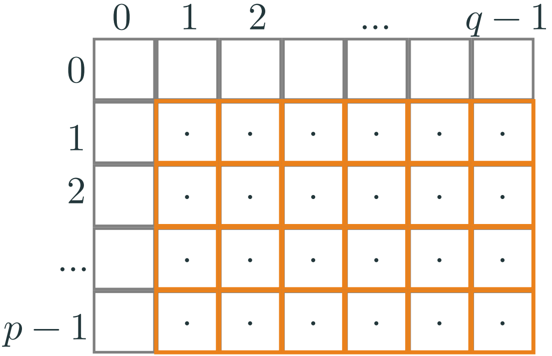
Proof

- Consider all pq remainders modulo pq
- By Chinese Remainder Theorem, each r corresponds to pair (r_p, r_q)
- $\mathrm{GCD}(pq, r) = 1 \Leftrightarrow r_p, r_q \neq 0$



Proof

- Consider all pq remainders modulo pq
- By Chinese Remainder Theorem, each r corresponds to pair (r_p, r_q)
- $\mathrm{GCD}(pq, r) = 1 \Leftrightarrow r_p, r_q \neq 0$



$(p-1)(q-1)$

such (r_p, r_q)

Outline

Fast Modular Exponentiation

Fermat's Little Theorem

Euler's Totient Function

Euler's Theorem

Euler’s Theorem

- Generalization of Fermat’s Little Theorem
- Together with modular exponentiation, is used to encrypt and decrypt in RSA

Euler's Theorem

Theorem

If a is coprime with n, $a^{\phi(n)} \equiv 1 \bmod n$.

Proof

- Very similar to Fermat's Little Theorem
- Consider all $\phi(n)$ remainders modulo n which are coprime with n
- Multiplying by a is invertible, so new remainder is also coprime with n
- Multiplying some r by a many times leads to cycle of length l
- $a^l \equiv 1 \bmod n$

Proof

- $a^l \equiv 1 \bmod n$
- All cycle lengths are the same
- Cycles don't intersect and cover all remainders coprime with n
- If there are c cycles, then $cl = \phi(n)$
- $a^{\phi(n)} = a^{cl} = (a^l)^c \equiv 1^c \equiv 1 \bmod n$ $\square$

Conclusion

- Defined modular exponentiation

Conclusion

- Defined modular exponentiation
- Designed fast algorithm to compute modular exponentiation

Conclusion

- Defined modular exponentiation
- Designed fast algorithm to compute modular exponentiation
- Proved Fermat's Little Theorem

Conclusion

- Defined modular exponentiation
- Designed fast algorithm to compute modular exponentiation
- Proved Fermat's Little Theorem
- Defined Euler's totient function

Conclusion

- Defined modular exponentiation
- Designed fast algorithm to compute modular exponentiation
- Proved Fermat's Little Theorem
- Defined Euler's totient function
- Proved Euler's theorem

Conclusion

- Defined modular exponentiation
- Designed fast algorithm to compute modular exponentiation
- Proved Fermat's Little Theorem
- Defined Euler's totient function
- Proved Euler's theorem
- Next module — learn public key cryptography and break some secret codes yourself!