The RSA Assumption

Lecture 22: Easy powers, hard roots

Lecture overview

Raising a number to a power modulo N is efficient. A carefully chosen exponent makes this operation reversible. The factors of N tell us how to reverse it.

Based on the instructor's Lecture 22 [Maj26].

Abstract and lecture roadmap

We already know how to generate primes and compute large powers efficiently. Today we use those tools to build a candidate **trapdoor permutation**: a public bijection with a secret that makes inversion easy.

1. Work through the example $N = 33$: cube a number, then recover it.
2. Explain why the same calculation works for every valid RSA key.
3. State precisely what an attacker receives and what we assume is hard.
4. See how this gives a starting point for public-key cryptography.

Proof roadmap

Choose an exponent d so that $ed = 1 + k\varphi(N)$. Euler's theorem removes the extra factor $x^{k\varphi(N)}$, leaving just x.

[BS23, §10.3, pp. 403–406]; [HPS14, §§3.1–3.2, pp. 117–125].

What we prove, and what we assume

Theorem: RSA exponentiation is a permutation

Let $N = pq$ for distinct primes p, q. If $e \geq 1$ and $\gcd(e, \varphi(N)) = 1$, then

$$f_{N,e} \colon \mathbb{Z}_N^* \to \mathbb{Z}_N^*, \qquad x \mapsto x^e \bmod N$$

is a bijection. Its inverse is $y \mapsto y^d \bmod N$, where $ed \equiv 1 \pmod{\varphi(N)}$.

RSA assumption: inversion is hard without the trapdoor

For the key distribution we specify, every efficient attacker has negligible probability of recovering a uniformly random input from $(N, e, x^e \bmod N)$.

The first statement has a proof. The second is a computational assumption.

[HPS14, Prop. 3.5, pp. 120–121]; [BS23, Def. 10.5, p. 405].

The domain: numbers we can multiply and divide

For $N > 1$, write

$$\mathbb{Z}_N = \{0, 1, \ldots, N-1\}, \qquad \mathbb{Z}_N^* = \{x \in \mathbb{Z}_N : \gcd(x, N) = 1\}.$$

Every element of $\mathbb{Z}_N^*$ has a multiplicative inverse modulo N.

Counting the domain when $N = pq$

For distinct primes p, q, the nonunits are precisely the multiples of p or q. Counting in $\{0, \ldots, N-1\}$ gives

$$|\mathbb{Z}_N^*| = N - q - p + 1 = (p-1)(q-1) =: \varphi(N).$$

There are q multiples of p, p multiples of q, and one overlap: 0.

All powers in this lecture are reduced modulo N.

[HPS14, §3.1, pp. 117–120].

Our example: $p = 3$, $q = 11$, and $e = 3$

$$N = 3 \cdot 11 = 33, \qquad \varphi(N) = 2 \cdot 10 = 20.$$

The twenty inputs

$$\mathbb{Z}_{33}^* = \{1, 2, 4, 5, 7, 8, 10, 13, 14, 16, \\ 17, 19, 20, 23, 25, 26, 28, 29, 31, 32\}.$$

Choose $e = 3$. Since $\gcd(3, 20) = 1$, the map $x \mapsto x^3 \bmod 33$ will rearrange these twenty numbers.

This tiny example illustrates the arithmetic. The security experiment later uses two large primes of the same bit length.

Example from source slides, pp. 6–13 [Maj26].

Computing the forward map: $13^3 \bmod 33$

Square, then multiply

$$\begin{aligned} 13^2 &= 169 \equiv 4 \pmod{33}, \\ 13^3 &\equiv 13 \cdot 4 \\ &= 52 \equiv 19 \pmod{33}. \end{aligned}$$

The general method

Compute

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

by repeated squaring. Multiply the powers selected by the binary digits of the exponent.

For $e = 3 = 2 + 1$, we need one squaring and one multiplication.

We reduce after every multiplication. Intermediate integers stay small even when the exponent is large.

[HPS14, §1.3.2, pp. 24–26].

Cubing rearranges all twenty inputs

The row x^3 contains every unit exactly once. All entries are modulo 33.

x	1	2	4	5	7	8	10	13	14	16
x^2	1	4	16	25	16	31	1	4	31	25
x^3	1	8	31	26	13	17	10	19	5	4

x	17	19	20	23	25	26	28	29	31	32
x^2	25	31	4	1	31	16	25	16	4	1
x^3	29	28	14	23	16	20	7	2	25	32

For example, $13^3 = 19$, so the unique cube root of 19 is 13 (modulo 33).

Source slides, pp. 7–10 [Maj26]; entries checked by direct modular arithmetic.

The inverse exponent lives modulo $\varphi(N)$

We want an exponent d such that $ed \equiv 1 \pmod{\varphi(N)}$. For our example, solve $3d \equiv 1 \pmod{20}$.

The extended Euclidean algorithm finds d

$$20 = 6 \cdot 3 + 2, \qquad 3 = 1 \cdot 2 + 1.$$

Substitute backward:

$$1 = 3 - (20 - 6 \cdot 3) = 7 \cdot 3 - 20.$$

Thus $d = 7$, because $3 \cdot 7 = 21 = 1 + 20$.

The exponent relation is modulo $20 = \varphi(33)$. The powers themselves are computed modulo 33.

[HPS14, §3.2, Table 3.1, p. 123].

Recovering the input: $19^7 \bmod 33$

Since $7 = 4 + 2 + 1$, compute two squares and multiply.

Build the powers

$$19^2 = 361 \equiv 31 \pmod{33},$$
$$19^4 \equiv 31^2 = 961 \equiv 4 \pmod{33}.$$

Combine them

$$19^7 \equiv 19 \cdot 31 \cdot 4 \equiv 28 \cdot 4 = 112 \equiv 13 \pmod{33}.$$

$$13 \xrightarrow{\text{cube}} 19 \xrightarrow{\text{seventh power}} 13.$$

Knowing d makes taking a cube root just another exponentiation.

The seventh power undoes the cube

Apply the inverse operation to each output of the cube map.

x	1	2	4	5	7	8	10	13	14	16
$y = x^3$	1	8	31	26	13	17	10	19	5	4
y^7	1	2	4	5	7	8	10	13	14	16

x	17	19	20	23	25	26	28	29	31	32
$y = x^3$	29	28	14	23	16	20	7	2	25	32
y^7	17	19	20	23	25	26	28	29	31	32

Keep the input to each operation straight

x^7 is the seventh power of the original input. To undo $y = x^3$, we compute $y^7 = (x^3)^7$.

Source slides, pp. 11–13 [Maj26]; complete power tables appear in the appendix.

The calculation we need to explain

The example works because

$$(x^3)^7 = x^{21} = x \cdot x^{20} \equiv x \pmod{33} \qquad (x \in \mathbb{Z}_{33}^*).$$

Euler's theorem

For every $N > 1$ and $x \in \mathbb{Z}_N^*$,

$$x^{\varphi(N)} \equiv 1 \pmod{N}.$$

The plan for the general proof

First explain why $x^{\varphi(N)} = 1$. Then choose d so that ed differs from 1 by a multiple of $\varphi(N)$.

For $N = pq$, see [HPS14, Thm. 3.1, pp. 118–119]; its lcm exponent also implies this formula.

A short proof of Euler's theorem

List the units as $u_1, \dots, u_t$, where $t = \varphi(N)$, and fix $x \in \mathbb{Z}_N^*$.

Multiplication by x rearranges the units

If $xu_i \equiv xu_j \pmod{N}$, multiply by x^{-1} to obtain $u_i = u_j$. Thus $xu_1, \dots, xu_t$ are the same units in a different order.

Multiply all the entries:

$$(xu_1) \cdots (xu_t) \equiv u_1 \cdots u_t \pmod{N}.$$

The product $u_1 \cdots u_t$ is itself a unit, so we may cancel it:

$$\boxed{x^t = x^{\varphi(N)} \equiv 1 \pmod{N}.}$$

The cancellation is exactly why we work in $\mathbb{Z}_N^*$.

The inverse formula proves bijectivity

Suppose $\gcd(e, \varphi(N)) = 1$. Choose $1 \le d < \varphi(N)$ with

$$ed = 1 + k\varphi(N)$$

for some integer $k \ge 0$.

Both compositions are the identity

For every $x, y \in \mathbb{Z}_N^*$, Euler's theorem gives

$$(x^e)^d = x^{ed} = x\big(x^{\varphi(N)}\big)^k \equiv x \pmod{N},$$

and, by the same calculation, $(y^d)^e \equiv y \pmod{N}$. Hence $x \mapsto x^e$ and $y \mapsto y^d$ are inverse bijections.

An independent check: if $u^e = v^e$, raise both sides to d to get $u = v$. The map is injective on a finite set, so it is bijective.

[HPS14, Prop. 3.5, pp. 120–121, eq. (3.3)].

Two distinctions worth keeping straight

The condition on the exponent matters

Squaring is not a permutation of $\mathbb{Z}_{33}^*$:

$$2^2 \equiv 31^2 \equiv 4 \pmod{33}, \qquad 2 \neq 31.$$

Here $\gcd(2, \varphi(33)) = \gcd(2, 20) \neq 1$.

A multiplicative inverse and an RSA root solve different equations

$$\text{Multiplicative inverse:} \quad yz \equiv 1 \pmod{N},$$

$$\text{RSA root:} \quad z^e \equiv y \pmod{N}.$$

Euclid's algorithm computes the first from (N, y). Our method for the second uses the secret exponent d.

Generating a key: the lecture's distribution

Let $n \geq 3$ be the security parameter and $\mathcal{P}_n = \{p \text{ prime} : 2^{n-1} < p < 2^n\}$.

Key generation $\mathrm{Gen}(1^n)$

1. Sample $p \xleftarrow{\$} \mathcal{P}_n$, then $q \xleftarrow{\$} \mathcal{P}_n \setminus \{p\}$.
2. Set $N = pq$ and $\varphi(N) = (p-1)(q-1)$.
3. Sample $e \xleftarrow{\$} \mathbb{Z}^*_{\varphi(N)}$, represented in $\{1, \ldots, \varphi(N) - 1\}$.
4. Compute $d = e^{-1} \bmod \varphi(N)$ by the extended Euclidean algorithm.

$$\mathsf{pk} = (N, e), \qquad \mathsf{sk} = d.$$

Keep $p, q, \varphi(N)$ secret as well. The modulus N has $2n - 1$ or $2n$ bits.

Random-exponent convention from source pp. 4–5 [Maj26]; compare [BS23, §10.3, p. 403].

The RSA inversion experiment

Challenger

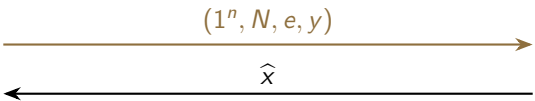
Generate $(N, e, d) \leftarrow \mathsf{Gen}(1^n)$.

Choose $y \xleftarrow{\$} \mathbb{Z}_N^*$. Keep d hidden.

Attacker A

Receive $(1^n, N, e, y)$.

Output a proposed root $\widehat{x}$.



Winning condition

The attacker wins exactly when $\widehat{x} \in \mathbb{Z}_N^*$ and $\widehat{x}^{\,e} \equiv y \pmod{N}$. There is exactly one winning answer.

Compare [BS23, Attack Game 10.3 and Def. 10.5, pp. 404–405].

The assumption is an average-case statement

RSA assumption for Gen

For every probabilistic polynomial-time attacker A,

$$\Pr\left[\begin{array}{c}(N,e,d)\leftarrow \mathsf{Gen}(1^n),\quad y\xleftarrow{\$}\mathbb{Z}_N^*,\\ \widehat{x}\leftarrow A(1^n,N,e,y):\ \widehat{x}\in\mathbb{Z}_N^*,\ \widehat{x}^{\,e}\equiv y \pmod N\end{array}\right]\leq \mathsf{negl}(n).$$

A function $\varepsilon(n)$ is **negligible** if, for every constant $c>0$, $\varepsilon(n)\leq n^{-c}$ for all sufficiently large n.

- The probability includes key generation, the target, and A's coins.
- Easy individual targets, such as $y=1$, are allowed.
- Exponentially small success is a stronger requirement than negligible success.

[BS23, Def. 10.5 and §10.3.2, p. 405].

Why we must specify the key distribution

The convention in these slides

Choose the primes first, then choose a uniformly random invertible exponent e. The RSA assumption concerns this whole random experiment.

The convention in Boneh–Shoup

Fix an odd exponent $e > 2$ first. Then generate distinct primes satisfying

$$\gcd(e, p-1) = \gcd(e, q-1) = 1.$$

This is a different distribution of public keys.

Correctness holds for both conventions. Hardness refers to the chosen distribution.

A valid key need not be hard: $e = 1$ is an easy example.

[BS23, §10.3, algorithm RSAGen, p. 403].

What is easy, and what is assumed hard?

Task	Available information	Method or status
Evaluate $x^e \bmod N$	N, e, x	Repeated squaring
Invert y	N, d, y	Compute $y^d \bmod N$
Compute d	e, p, q	Compute $\varphi(N)$, then use Euclid
Invert random y	N, e, y	Assumed hard for generated keys

Why a large exponent is still manageable

An L-bit exponent needs $O(L)$ modular multiplications. Here $e, d < N < 2^{2n}$, so forward and trapdoor evaluation take polynomial time in n.

[HPS14, §1.3.2, pp. 24–26]; [BS23, §10.3].

Factoring gives a way to invert RSA

An efficient factoring algorithm would break RSA

Given N, e, y, factor $N = pq$, then compute

$$\varphi(N) = (p-1)(q-1), \qquad d = e^{-1} \bmod \varphi(N), \qquad x = y^d \bmod N.$$

The correctness proof shows that $x^e \equiv y \pmod{N}$.

Knowing $\varphi(N)$ also reveals the factors

Since $\varphi(N) = N - p - q + 1$, we know $s = p + q = N + 1 - \varphi(N)$. The roots of $T^2 - sT + N$ are p and q.

These implications do not prove that every RSA inverter can factor N.

[BS23, §10.3, p. 404]; [HPS14, §3.3, pp. 126–128].

A random input gives a random target

Fix a valid public key (N, e) and sample $X \xleftarrow{\$} \mathbb{Z}_N^*$. Put $Y = X^e \bmod N$.

The permutation makes the distributions agree

For every $y \in \mathbb{Z}_N^*$, there is exactly one x with $x^e = y$, so

$$\Pr[Y = y] = \frac{1}{|\mathbb{Z}_N^*|}.$$

Thus we may sample a uniform target directly, or exponentiate a uniform input.

A protocol can now use this fact: choose X and send its image Y.

Sampling a unit requires no factorization: sample $X \in \{0, \ldots, N-1\}$ and retry until $\gcd(X, N) = 1$.

[BS23, §10.3.1, p. 405].

Using RSA to transport a random secret

Alice

Generate (N, e, d).

Publish (N, e) and keep d.

On receiving Y, recover

$$X = Y^d \bmod N.$$

Bob

Receive (N, e).

Choose $X \xleftarrow{\$} \mathbb{Z}_N^*$.

Send

$$Y = X^e \bmod N.$$

A passive observer sees (N, e, Y). Recovering the whole secret X is exactly the RSA inversion task.

This argument concerns recovery of X. A secure encryption key needs an additional guarantee about the bits we derive from X.

[BS23, §10.3.1, p. 405; §11.4, pp. 441–444].

A hidden secret is not automatically a one-time pad

Suppose we try to encrypt an ℓ-bit message by

$$Y = X^e \bmod N, \qquad C = M \oplus \mathrm{bin}_\ell(X), \quad \ell = \lceil \log_2 N \rceil.$$

An attacker can test a candidate message M_0

1. Decode X_0 from the bit string $C \oplus M_0$.
2. Check that $X_0 < N$, $\gcd(X_0, N) = 1$, and $X_0^e \equiv Y \pmod N$.

Exactly the correct message candidate passes: Y has only one root in $\mathbb{Z}_N^*$.

The next step

Use a construction that proves the derived key bits look random to the observer. One-wayness of RSA alone is not a proof for the raw XOR scheme above.

Direct attack; related encryption construction: [BS23, §11.4, pp. 441–444].

The lecture in three equations

Public evaluation

$$y = x^e \bmod N.$$

Anyone with the public key can compute this efficiently.

Secret inversion

$$ed \equiv 1 \pmod{\varphi(N)}, \qquad x = y^d \bmod N.$$

Euler's theorem proves that the two operations undo each other.

Computational hardness

$$\Pr[A(1^n, N, e, y) \text{ finds the root}] \leq \mathrm{negl}(n).$$

This is the RSA assumption for our specified key distribution.

Check your understanding

1. For $N = 33$ and $e = 3$, why does $d = 7$ work? Which modulus belongs in the equation for ed?
2. Why does the collision $2^2 \equiv 31^2 \pmod{33}$ not contradict the permutation theorem?
3. If an attacker can recover X from a random RSA image, what part of our assumption fails?
4. If the attacker can predict one bit of X, does that alone contradict the stated RSA assumption?

Try to answer without looking at the power tables. The distinction between recovering a whole value and hiding useful information about it will matter again.

Answers to the check

1. $3 \cdot 7 = 21 = 1 + 20$, so $ed \equiv 1 \pmod{\varphi(33)}$. Consequently, $(x^3)^7 = x^{21} \equiv x$ for every unit.
2. The permutation theorem requires $\gcd(e, \varphi(N)) = 1$. For squaring, $\gcd(2, 20) = 2$.
3. An efficient algorithm with nonnegligible success over the specified key and target distribution breaks the RSA assumption.
4. Not from the statement alone. RSA inversion asks for the entire preimage. A claim that a particular bit is hard to predict needs a separate argument.

Complete table modulo 33: first ten inputs

$e = 3$, $d = 7$, and $ed = 21 \equiv 1 \pmod{20}$.

x	1	2	4	5	7	8	10	13	14	16
x^2	1	4	16	25	16	31	1	4	31	25
x^4	1	16	25	31	25	4	1	16	4	31
$y = x^3$	1	8	31	26	13	17	10	19	5	4
x^7	1	29	16	14	28	2	10	7	20	25
$y^7 = (x^3)^7$	1	2	4	5	7	8	10	13	14	16

How each row is computed

Square to obtain x^2 and x^4. Multiply to obtain $x^3 = x \cdot x^2$ and $x^7 = x \cdot x^2 \cdot x^4$. The final row applies the seventh power to $y = x^3$.

Complete arithmetic from source pp. 7–13 [Maj26], regrouped for readability.

Complete table modulo 33: remaining inputs

The same calculations continue for the other ten units.

x	17	19	20	23	25	26	28	29	31	32
x^2	25	31	4	1	31	16	25	16	4	1
x^4	31	4	16	1	4	25	31	25	16	1
$y = x^3$	29	28	14	23	16	20	7	2	25	32
x^7	8	13	26	23	31	5	19	17	4	32
$y^7 = (x^3)^7$	17	19	20	23	25	26	28	29	31	32

Three checks

The x^3 row is a permutation of all twenty units. So is the x^7 row, since $\gcd(7, 20) = 1$. The last row agrees with x in every column.

All entries independently checked by modular exponentiation.

Reading guide

The mathematical explanation

Hoffstein–Pipher–Silverman, second edition [HPS14]:

- §1.3.2, pp. 24–26: fast exponentiation.
- §3.1, pp. 117–122: Euler's formula and roots modulo pq.
- Proposition 3.5, pp. 120–121: the unique-root formula.
- §3.2, pp. 123–125: the RSA cryptosystem.

The cryptographic formulation

Boneh–Shoup, version 0.6 [BS23]: §10.3, pp. 403–406, for trapdoor permutations and the RSA assumption; §11.4, pp. 441–444, for encryption from a trapdoor function.

The original construction is due to Rivest, Shamir, and Adleman [RSA78].

References

[BS23] Dan Boneh and Victor Shoup.
A Graduate Course in Applied Cryptography, 2023. Version 0.6, January 2023. URL: https://crypto.stanford.edu/~dabo/cryptobook/BonehShoup_0_6.pdf.

[HPS14] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman.
An Introduction to Mathematical Cryptography. Undergraduate Texts in Mathematics. Springer, New York, 2 edition, 2014. doi:10.1007/978-1-4939-1711-2.

[Maj26] Hemanta K. Maji.
CS355, Lecture 22: RSA Assumption. Instructor's lecture slides, Purdue University, 2026. Source material supplied for this lecture.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman.
A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978. doi:10.1145/359340.359342.

Technical inputs used in the lecture

Extended Euclidean algorithm and fast exponentiation

Given positive integers a, b, the extended Euclidean algorithm finds integers u, v with $ua + vb = \gcd(a, b)$ in polynomial time. Thus it computes an inverse modulo b when $\gcd(a, b) = 1$.

Binary exponentiation computes $x^r \bmod N$ with $O(\log r)$ modular multiplications for $r \geq 2$.

[HPS14, Thm. 1.11, pp. 16–17, Fig. 1.3, p. 19; §1.3.2, pp. 24–26].

Prime sampling convention

The experiment uses ideal uniform sampling of distinct n-bit primes, as in the source lecture. Prime generation is a prerequisite from earlier lectures. A probabilistic implementation must make its primality-testing error negligible.

[BS23, §10.3.2, pp. 405–406].

The only computational hardness assumption

RSA inversion for the stated generator

For every probabilistic polynomial-time A, its probability of producing $x \in \mathbb{Z}_N^*$ with $x^e \equiv y \pmod{N}$ is negligible, where

$$(N, e, d) \leftarrow \mathrm{Gen}(1^n), \qquad y \xleftarrow{\$} \mathbb{Z}_N^*.$$

The generator samples distinct uniform n-bit primes, then a uniform $e \in \mathbb{Z}_{\varphi(N)}^*$.

Relation to the cited textbook

Boneh–Shoup's Attack Game 10.3 and Definition 10.5, pp. 404–405, state RSA hardness for a fixed-exponent generator and targets in $\mathbb{Z}_N$. Our slides explicitly use the instructor's random-exponent generator and the unit domain $\mathbb{Z}_N^*$.

[Maj26, pp. 4–5]; [BS23, §10.3, pp. 403–406].