

One-way Functions

Lecture 14: Easy to compute, hard to invert

Lecture overview

Multiplication is easy. Recovering the prime factors of a large product appears much harder. One-way functions turn this familiar gap into a precise computational assumption.

Definitions, candidate constructions, and proof roadmaps.

Primary source: Lecture 14 [Maj26].

Abstract and lecture roadmap

A one-way function is efficient to evaluate, but every efficient algorithm has only a tiny chance of finding a preimage of the output generated from a sampled input.

1. Specify the inversion experiment and its input distribution.
2. Use factoring to obtain a weak one-way function, then amplify its hardness.
3. Study discrete logarithms, modular squaring, and elliptic curves.
4. See why one-way functions would imply $P \neq NP$.

The recurring proof plan

If an algorithm could invert the proposed function, use it to solve a problem already assumed to be hard.

[BS23, §8.11.1]; [Gol01, Ch. 2].

Main results

Multiplication and amplification

Assume that factoring a product of two independent uniform n -bit primes is hard. Then multiplication of uniform n -bit strings is weakly one-way. A tuple of n^3 independent products is strongly one-way.

A complexity consequence

If $P = NP$, every polynomial-time function on fixed-length bit strings can be inverted efficiently on its image. Consequently, one-way functions imply $P \neq NP$.

The first statement is conditional on factoring hardness. The second is an unconditional implication.

Factoring: [HPS14, §3.6]. General amplification: [Gol01, Thm. 2.3.2].

What counts as an inverse?

For multiplication, the output 77 has the explanations

$$7 \cdot 11 = 77 = 11 \cdot 7.$$

Either valid input pair inverts the output.

Any preimage is enough

Given $z = f(x)$, the task is to find any valid x' satisfying

$$f(x') = z.$$

The task does not require recovering the particular x that was sampled.

Think. For the constant function $f(x) = 0^n$, guessing the original x succeeds with probability 2^{-n} . Yet finding a preimage is immediate.

[BS23, §8.11.1, p. 336].

The inversion experiment

Let n be the security parameter. Specify a domain D_n , an efficient sampler for $X \in D_n$, and an efficiently computable map f_n .

1. The challenger samples X and computes $Z = f_n(X)$.
2. The adversary receives 1^n and Z , and returns X' .
3. It wins exactly when $X' \in D_n$ and $f_n(X') = Z$.

$$\text{Succ}_{f,A}(n) = \Pr[X' \in D_n \text{ and } f_n(X') = f_n(X)].$$

Probability includes the sampling of X and the adversary's random bits.

For indexed candidates, the public parameters are also given to the adversary and their sampling is included in the experiment.

[BS23, §8.11.1].

Efficiency and a tiny success probability

One-wayness

A family (f_n) is one-way if evaluation is polynomial-time and, for every probabilistic polynomial-time adversary A ,

$$\text{Succ}_{f,A}(n) = \text{negl}(n).$$

A function $\varepsilon(n) \geq 0$ is **negligible** if, for every constant $c > 0$, eventually $\varepsilon(n) < n^{-c}$.

Example	Negligible?	Reason
2^{-n}	Yes	Beats every inverse polynomial
n^{-100}	No	Does not beat n^{-101}

[BS23, §2.3.1, p. 28; §8.11.1].

What is known, and what is assumed?

- An unbounded algorithm can enumerate the domain and test each possible preimage.
- One-wayness concerns efficient algorithms and a specified distribution of challenges.
- The candidates in this lecture rely on unproved computational assumptions.

Worst-case difficulty is not enough

A few difficult inputs do not establish one-wayness. The inversion probability must be small when the challenger samples an input in the prescribed way.

The appendix proves that $P = NP$ would rule out one-way functions. The converse implication is not known.

[Gol01, §2.1]; [HPS14, §5.7].

Multiplication of two random primes

Let n denote the number of bits in each prime, and write

$$\mathcal{P}_n = \{p : p \text{ prime}, 2^{n-1} \leq p < 2^n\}.$$

Sample p, q independently and uniformly from $\mathcal{P}_n$.

The prime-product candidate

$$f_n : \mathcal{P}_n \times \mathcal{P}_n \longrightarrow \{0, 1\}^{2n}, \quad f_n(p, q) = pq.$$

The output is the product's binary encoding, padded to $2n$ bits.

Schoolbook multiplication takes $O(n^2)$ bit operations. Inversion asks for the prime factors, in either order.

[HPS14, §3.6]; [BS23, §10.3].

The factoring assumption

For an efficient algorithm B , define

$$\text{Fact}_B(n) = \Pr \left[\begin{array}{l} p', q' \in \mathcal{P}_n, \quad p'q' = pq \\ (p', q') = B(1^n, pq), \quad p, q \xleftarrow{\$} \mathcal{P}_n \end{array} \right].$$

Assumption used in this lecture

For every probabilistic polynomial-time B , the function $\text{Fact}_B(n)$ is negligible.

- The input distribution is part of the assumption.
- $(1, pq)$ is not a valid answer: both output factors must lie in $\mathcal{P}_n$.
- This says nothing about the difficulty of factoring every composite integer.

Background and factoring algorithms: [HPS14, §3.6].

Multiplication on uniform bit strings

Now identify each n -bit string with an integer in $\{0, \dots, 2^n - 1\}$, and set

$$g_n : \{0, 1\}^n \times \{0, 1\}^n \longrightarrow \{0, 1\}^{2n}, \quad g_n(x, y) = xy.$$

Let H be the event that both sampled inputs are n -bit primes.

Probability of the prime-pair event

$$\delta_n := \Pr(H) = \left(\frac{|\mathcal{P}_n|}{2^n} \right)^2 \sim \frac{1}{4(n \ln 2)^2} = \Theta(n^{-2}).$$

Conditioned on H , the challenge has exactly the factoring distribution. The prime number theorem supplies the density estimate.

Prime density: [Sho09, Thm. 5.14, p. 116].

A weak guarantee for uniform inputs

Let S be the event that an efficient algorithm A returns a valid preimage under g_n .

Proof roadmap. Split according to H . Use factoring hardness on H , and the bound 1 elsewhere.

$$\begin{aligned}\Pr(S) &= (1 - \delta_n) \Pr(S \mid \neg H) + \delta_n \Pr(S \mid H) \\ &\leq 1 - \delta_n + \delta_n \text{Fact}_B(n).\end{aligned}$$

Here B runs A on its prime-product challenge.

Why every valid preimage gives the factors

For $n \geq 3$, a product of two n -bit primes exceeds $2^n - 1$. Since both output factors must be below 2^n , the pair $(1, pq)$ is invalid. Unique factorization leaves (p, q) or (q, p) .

The input and output length restrictions are essential.

Weak and strong one-wayness

Strong one-wayness

Every efficient inverter succeeds with negligible probability.

Weak one-wayness

There is a polynomial Q such that every efficient inverter fails with probability at least $1/Q(n)$ for all sufficiently large n .

For multiplication, the previous proof gives, for sufficiently large n ,

$$\Pr(\text{failure}) \geq \delta_n(1 - \text{Fact}_B(n)) \geq \delta_n/2 = \Omega(n^{-2})$$

This proves a weak guarantee. It does not decide whether unamplified g_n is strongly one-way.

A tuple of independent products

Choose a number of coordinates $k = k(n)$ and define

$$G_{n,k}(x_1, y_1, \dots, x_k, y_k) = (x_1 y_1, \dots, x_k y_k).$$

Every input coordinate is a fresh pair of uniform n -bit strings.

What inversion requires

The adversary must return valid n -bit factor pairs for *every* coordinate. One failed coordinate means it loses the experiment.

Proof roadmap

A long tuple probably contains a prime-pair coordinate. An inverter that handles the entire tuple can then be used to factor a random prime product.

General direct-product amplification: [Gol01, Thm. 2.3.2].

The independence calculation we can use

Let H_i mean that the original input pair in coordinate i consists of two n -bit primes. These events are independent, so

$$\Pr(\text{no } H_i) = (1 - \delta_n)^k \leq e^{-k\delta_n}.$$

A tuple usually includes a factoring challenge

If $k = n^3$, then $k\delta_n = \Theta(n)$, and hence

$$\Pr(\text{no } H_i) \leq e^{-\Omega(n)}.$$

An adversary can process all coordinates jointly. Its success events can be correlated. The next step is therefore a reduction to factoring.

See the discussion of direct-product amplification in [Gol01, §2.3.1].

Embedding one factoring challenge

Suppose A tries to invert the entire tuple. Construct a factoring algorithm B as follows.

1. Receive $z = pq$, with independent uniform $p, q \in \mathcal{P}_n$.
2. Choose a uniform position $J \in \{1, \dots, k\}$.
3. Put z in position J . Fill every other position with an independently sampled product $x_i y_i$.
4. Run A . Return its factor pair in position J if valid.

For each fixed $J = i$, the tuple has the original distribution conditioned on H_i .

If k is polynomial in n , then B is efficient whenever A is efficient.

The amplification bound

Let S mean that A inverts the whole tuple, and V_i that its answer in position i is valid. The reduction gives

$$\eta := \text{Fact}_B(n) = \frac{1}{k} \sum_{i=1}^k \Pr(V_i \mid H_i).$$

Split off tuples with no prime-pair coordinate, then use a union bound:

$$\begin{aligned} \Pr(S) &\leq \Pr(\text{no } H_i) + \sum_{i=1}^k \Pr(S \cap H_i) \\ &\leq (1 - \delta_n)^k + \delta_n \sum_{i=1}^k \Pr(V_i \mid H_i) \\ &\leq \boxed{e^{-k\delta_n} + k\delta_n \text{Fact}_B(n).} \end{aligned}$$

Only the prime-pair events H_i were multiplied. No independence of the adversary's successes was used.

Choosing the number of coordinates

Take $k = n^3$. Since $\delta_n = \Theta(n^{-2})$,

$$\Pr(A \text{ inverts } G_{n,n^3}) \leq e^{-\Omega(n)} + O(n) \text{Fact}_B(n) = \text{negl}(n).$$

Evaluation uses k multiplications, hence polynomial time.

The amplified one-way family

The input and output lengths are $2nk = 2n^4$. Negligibility in n is also negligibility in this polynomially related length.

General theorem. Every weak one-way function can be converted into a strong one-way function by a sufficiently long polynomial-size direct product.

General statement: [Gol01, Thm. 2.3.2, §2.3.1]. The multiplication case was proved above.

Discrete logarithms

Let $G = \langle g \rangle$ be a cyclic group of order r , with efficient group operations. Its public description includes g and r .

The exponentiation map

$$f : \mathbb{Z}/r\mathbb{Z} \longrightarrow G, \quad f(x) = g^x.$$

Given g^x , inversion asks for the exponent x modulo r .

Small example. In $\mathbb{F}_{23}^\times$, the element 5 generates the group, and

$$5^0 = 1, \quad 5^1 = 5, \quad 5^2 = 2, \quad 5^3 = 10 \pmod{23}.$$

Thus the discrete logarithm of 10 to the base 5 is 3 modulo 22.

[HPS14, §2.2, pp. 64–66]; [BS23, §10.5].

Why exponentiation is efficient

Binary exponentiation computes g^x using $O(\log r)$ group operations for $0 \leq x < r$.

$$13 = (1101)_2, \quad g^{13} = g^8 g^4 g.$$

Repeated squaring produces g, g^2, g^4, g^8 . Multiply the selected powers.

The discrete-log assumption

Choose a suitable family of groups whose descriptions have size polynomial in n . For uniform $x \in \mathbb{Z}/r\mathbb{Z}$, every efficient algorithm finds x from g^x with negligible probability.

The choice of group matters. In the additive group $\mathbb{Z}/r\mathbb{Z}$ with generator 1, the corresponding map is $x \mapsto x \cdot 1 = x$, which is easy to invert.

[HPS14, §1.3.2; §2.6]; [BS23, §10.5].

Rabin's squaring function

Sample independent uniform n -bit primes p, q , conditioned on $p \neq q$, with $n \geq 3$. Make their product $N = pq$ public. Write $\mathbb{Z}_N^\times$ for the residues relatively prime to N .

The squaring map

$$R_N : \mathbb{Z}_N^\times \longrightarrow \mathbb{Z}_N^\times, \quad R_N(x) = x^2 \bmod N.$$

Sample x uniformly from $\mathbb{Z}_N^\times$. Inversion means finding any square root of $R_N(x)$.

- Evaluation takes one modular multiplication.
- Each output has exactly four roots modulo N .
- Knowing p, q permits efficient root extraction modulo each prime, followed by the Chinese remainder theorem.

[HPS14, §3.1, pp. 117–122]; [Sho09, §§2.4, 12.5].

Square roots and the Chinese remainder theorem

Take $N = 77 = 7 \cdot 11$. Since $20^2 \equiv 15 \pmod{77}$, consider the challenge 15.

The roots modulo 7 are ± 1 , and those modulo 11 are ± 2 . The four sign choices give:

Root modulo 7	Root modulo 11	Root modulo 77
1	2	57
1	-2	64
-1	2	13
-1	-2	20

Two different sign patterns reveal a factor

$$\gcd(20 - 13, 77) = 7.$$

[HPS14, §3.1]; [Sho09, §2.4].

A square-root finder gives a factoring algorithm

Proof roadmap. Choose a root secretly, ask for another root of its square, then compare them with a gcd.

1. Sample $x \xleftarrow{\$} \mathbb{Z}_N^\times$ and give $y = x^2 \bmod N$ to a root finder.
2. If it returns a valid root r , compute $d = \gcd(x - r, N)$.
3. Return d when $1 < d < N$.

Conditioned on y , the hidden x is uniform among its four roots. For any returned root r :

- $x = r$ gives $d = N$, and $x = -r$ gives $d = 1$.
- The other two roots agree with r modulo exactly one prime, so $d = p$ or $d = q$.

Success probability

A root finder with success ε yields factoring success $\varepsilon/2$.

Elliptic curves over finite fields

Let $p > 3$ be prime. Choose $a, b \in \mathbb{F}_p$ satisfying $4a^3 + 27b^2 \neq 0$.

The curve and its identity

$$E(\mathbb{F}_p) = \{(X, Y) \in \mathbb{F}_p^2 : Y^2 = X^3 + aX + b\} \cup \{\mathcal{O}\}.$$

The symbol $\mathcal{O}$ denotes the point at infinity, which serves as the identity for point addition.

An explicit addition rule makes $E(\mathbb{F}_p)$ an abelian group.

- The inverse of (X, Y) is $(X, -Y)$.
- The sum of two points is another point on the curve.
- The appendix gives the geometric picture and the formulas.

[HPS14, §§6.1–6.2]; [BS23, §§15.1–15.2, pp. 615–618].

Discrete logarithms on an elliptic curve

Choose a public point P generating a subgroup of large prime order r .

Scalar multiplication

$$f : \mathbb{Z}/r\mathbb{Z} \longrightarrow \langle P \rangle, \quad f(x) = xP = \underbrace{P + \cdots + P}_{x \text{ copies}}.$$

Doubling and adding evaluates xP in $O(\log r)$ group operations:

$$13P = 8P + 4P + P.$$

For suitable curve families, recovering uniform x from P, xP is assumed to be hard.

This is the discrete-log problem written using additive group notation. If P is sampled as a public parameter, the map can be written $(P, x) \mapsto (P, xP)$.

[HPS14, §6.3, pp. 310–315]; [BS23, §15.2, pp. 617–618].

One-way permutations

Definition

A one-way permutation is a one-way function $f_n : D_n \rightarrow D_n$ that is a bijection. In the bit-string setting, $D_n = \{0, 1\}^n$.

Every output then has exactly one preimage, but finding it is still computationally difficult.

- Squaring on $\mathbb{Z}_N^\times$ is not a permutation: each square has four roots.
- The map $x \mapsto g^x$ is a bijection between $\mathbb{Z}/r\mathbb{Z}$ and G . This alone does not make it a permutation of one common encoded domain.

One-way permutations give a useful first setting for constructions of pseudorandom generators.

RSA as a permutation candidate

Let $N = pq$ for distinct primes. Choose a public exponent e with

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

The RSA map on units

$$F_{N,e} : \mathbb{Z}_N^\times \longrightarrow \mathbb{Z}_N^\times, \quad F_{N,e}(x) = x^e \bmod N.$$

Choose d with $ed \equiv 1 \pmod{\varphi(N)}$. Euler's theorem gives

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

Thus the map is a permutation. Its one-wayness for suitable random public keys is the *RSA assumption*, which does not follow from factoring hardness alone by any known reduction.

[HPS14, §3.2]; [BS23, §10.3, pp. 403–405].

The candidates at a glance

Candidate	Easy direction	Inversion task
Prime multiplication	$(p, q) \mapsto pq$	Find the two prescribed-size prime factors
Exponentiation	$x \mapsto g^x$	Find a discrete logarithm in the chosen group
Rabin squaring	$x \mapsto x^2 \bmod N$	Find any root of a random square modulo N
Elliptic curve	$x \mapsto xP$	Find a discrete logarithm in $\langle P \rangle$
RSA permutation	$x \mapsto x^e \bmod N$	Find the unique preimage in $\mathbb{Z}_N^\times$

For every candidate

Specify the domain, the public parameters, the challenge distribution, and the exact event that counts as successful inversion.

[HPS14, Chs. 2–3, 6]; [BS23, Chs. 10, 15].

Exercises

1. Why must a valid inverse belong to the declared domain? Apply your answer to $(p, q) \mapsto pq$ and the proposed answer $(1, pq)$.
2. Suppose $\delta_n = \Theta(n^{-2})$. For each of $k = n$, $k = n^2$, and $k = n^3$, estimate $\Pr(\text{no prime-pair coordinate})$.
3. Find all square roots of 4 modulo 77. Which roots, compared with the hidden root 2, reveal a factor?
4. For $N = 77$ and $e = 7$, find an inverse exponent d modulo $\varphi(N)$. Verify that decrypting $2^7 \bmod 77$ recovers 2.

Discussion

Why does an efficient solution of every NP decision problem also let us *find* a preimage? The appendix constructs the inverse one bit at a time.

Appendix: why $P = NP$ rules out one-way functions

Let $f_n : \{0, 1\}^n \rightarrow \{0, 1\}^{m(n)}$ be a uniformly polynomial-time computable family, with $m(n)$ polynomially bounded.

Theorem

If $P = NP$, there is a polynomial-time algorithm that, given 1^n and y in the image of f_n , finds x satisfying $f_n(x) = y$.

Proof roadmap

Ask whether a proposed prefix can still be extended to a preimage. Choose one bit that keeps the answer yes, and repeat until all n bits are fixed.

Compare the supplied appendix, pp. 18–23 [Maj26]; [Gol01, §2.1].

The prefix-extension question

For a bit string s of length at most n , consider the question

Does there exist $u \in \{0, 1\}^{n-|s|}$ such that $f_n(su) = y$?

This is an NP decision problem

A witness is the remaining suffix u . To verify it, concatenate s and u , evaluate $f_n(su)$, and compare the result with y .

Both witness length and verification time are polynomial in the input length, which includes the unary parameter 1^n .

If $P = NP$, a deterministic polynomial-time algorithm decides this question. Call it $\text{Extend}(1^n, y, s)$.

An inverse, one bit at a time

```
Invert( $1^n, y$ )  
   $s \leftarrow$  the empty string  
  if Extend( $1^n, y, s$ ) = NO: return failure  
  for  $i = 1, \dots, n$ :  
    if Extend( $1^n, y, s0$ ) = YES:  $s \leftarrow s0$   
    else:  $s \leftarrow s1$   
  return  $s$ 
```

Invariant

The current prefix s has at least one extension $x \in \{0, 1\}^n$ with $f_n(x) = y$.

If no extension begins with $s0$, some extension must begin with $s1$. After n choices, the prefix is itself a full preimage.

Correctness and running time

- If y is in the image, the initial prefix has an extension.
- Each step preserves the existence of an extension.
- At length n , the only possible extension is the string itself. Thus $f_n(s) = y$.

There are $n + 1$ calls to a polynomial-time decision algorithm, on inputs of polynomial length. Hence inversion takes polynomial time.

Consequence

Under $P = NP$, an efficient algorithm inverts every generated output. Its success probability is 1, contradicting one-wayness.

For the textbook bit-string definition, this proves OWFs exist $\implies P \neq NP$.

Connection to the SAT formulation

For a fixed output y , an evaluation circuit for f_n can be encoded by a polynomial-size 3-CNF formula

$$\Phi_y(x, w), \quad f_n(x) = y \iff \exists w \Phi_y(x, w).$$

The auxiliary variables w record internal wire values.

Fixing a prefix of x leaves a satisfiability question about the remaining input bits and the wire variables.

A small example of restriction

$$\begin{aligned} \phi(x) &= (x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3), \\ \phi(x)|_{x_1=0} &= x_2 \vee \neg x_3. \end{aligned}$$

Circuit encoding: [AB07, §6.7, Lemmas 6.31–6.32, p. 112].

Appendix: the geometric addition rule

Over $\mathbb{R}$, consider

$$Y^2 = X^3 - 2X + 4.$$

1. Draw a line through P and Q .
2. Find its third intersection R' with the curve.
3. Reflect R' across the horizontal axis to obtain $R = P + Q$.

For $P = Q$, use the tangent. A vertical line gives $P + (-P) = \mathcal{O}$.

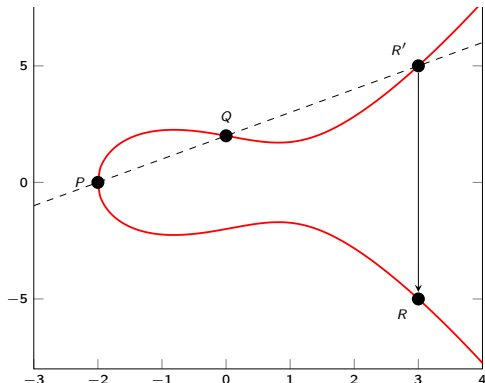


Figure from the supplied Lecture 14, p. 24 [Maj26]. See [HPS14, §6.1].

Addition formulas over a finite field

Let $P = (X_1, Y_1)$ and $Q = (X_2, Y_2)$ lie on $Y^2 = X^3 + aX + b$, over a field of characteristic greater than 3.

If $X_1 \neq X_2$, set

$$\lambda = \frac{Y_2 - Y_1}{X_2 - X_1}, \quad X_3 = \lambda^2 - X_1 - X_2, \quad Y_3 = \lambda(X_1 - X_3) - Y_1.$$

Then $P + Q = (X_3, Y_3)$. For doubling with $Y_1 \neq 0$, use

$$\lambda = \frac{3X_1^2 + a}{2Y_1}, \quad X_2 = X_1.$$

Exceptional cases and efficiency

$P + \mathcal{O} = P$. If $Q = -P$, the sum is $\mathcal{O}$; in particular $2P = \mathcal{O}$ when $Y_1 = 0$. All divisions above mean field inverses, computable efficiently by Euclid's algorithm over $\mathbb{F}_p$.

[HPS14, §6.2, pp. 306–309]; [BS23, §15.1].

Technical inputs: prime density and amplification

Prime number theorem

If $\pi(X)$ counts primes at most X , then $\pi(X) \sim X/\ln X$. Consequently, for $n \geq 3$,

$$|\mathcal{P}_n| = \pi(2^n) - \pi(2^{n-1}) \sim \frac{2^{n-1}}{n \ln 2} \quad (n \rightarrow \infty).$$

Citation: Shoup, *A Computational Introduction to Number Theory and Algebra*, 2nd ed., Theorem 5.14, p. 116 [Sho09].

General weak-to-strong amplification

A polynomial-time function with an inverse-polynomial lower bound on every efficient inverter's failure probability has a polynomial-size direct product that is strongly one-way.

Citation: Goldreich, *Foundations of Cryptography*, Vol. 1, Theorem 2.3.2, §2.3.1, pp. 44–48 [Gol01].

Technical inputs: modular arithmetic

Chinese remainder theorem and roots modulo primes

For distinct odd primes p, q , reduction gives a bijection

$$\mathbb{Z}/(pq)\mathbb{Z} \longrightarrow \mathbb{F}_p \times \mathbb{F}_q.$$

The inverse is efficient. Square roots modulo an odd prime can be found in expected polynomial time. A nonzero square has exactly two roots, so a unit square modulo pq has exactly four roots.

Citations: Shoup, Theorem 2.6, p. 23 (CRT); §12.5.1, pp. 350–351 (square roots) [Sho09]. HPS, §3.1, pp. 117–122 [HPS14].

Euler's theorem

For $x \in \mathbb{Z}_N^\times$, $x^{\varphi(N)} \equiv 1 \pmod{N}$. For distinct primes p, q , $\varphi(pq) = (p-1)(q-1)$.

Citation: HPS, §3.1, pp. 117–122 [HPS14].

Technical inputs: curves and Boolean formulas

The elliptic-curve group law

Over a field of characteristic > 3 , the nonsingular curve $Y^2 = X^3 + aX + b$, together with $\mathcal{O}$, forms an abelian group under the stated addition formulas. Associativity is part of this external theorem.

Citations: HPS, §§6.1–6.2, pp. 299–309 [HPS14]; Boneh–Shoup, §§15.1–15.2, pp. 615–618 [BS23].

Circuit encoding as 3-CNF

A circuit with AND/OR/NOT gates of fan-in at most two has a 3-CNF encoding of linear size, with variables w for its wires. Fixing its output to y gives $f_n(x) = y$ if and only if $\exists w \Phi_y(x, w)$.

Citation: Arora–Barak, January 2007 draft, §6.7, Lemmas 6.31–6.32, p. 112 [AB07].

References I

- [AB07] Sanjeev Arora and Boaz Barak.
Computational complexity: A modern approach, 2007.
January 2007 draft; section and page references refer to this draft.
URL: <https://theory.cs.princeton.edu/complexity/book.pdf>.
- [BS23] Dan Boneh and Victor Shoup.
A Graduate Course in Applied Cryptography.
2023.
Version 0.6, January 2023.
URL: <https://toc.cryptobook.us/book.pdf>.
- [Gol01] Oded Goldreich.
Foundations of Cryptography, Volume 1: Basic Tools.
Cambridge University Press, 2001.
URL: <https://www.wisdom.weizmann.ac.il/~oded/foc-vol1.html>,
doi:10.1017/CB09780511546891.

References II

- [HPS14] **Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman.**
An Introduction to Mathematical Cryptography.
Undergraduate Texts in Mathematics. Springer, New York, second edition, 2014.
doi:10.1007/978-1-4939-1711-2.
- [Maj26] **Hemanta K. Maji.**
Lecture 14: One-way functions.
CS355, Purdue University, Fall semester, 2026.
Instructor's supplied lecture slides, 24 pages.
- [Sho09] **Victor Shoup.**
A Computational Introduction to Number Theory and Algebra.
Cambridge University Press, second edition, 2009.
Online version 2.5.
URL: https://www.shoup.net/ntb/ntb-v2_5.pdf.