

Pseudorandom Generators

Lecture 15: From one hidden bit to a long string

Lecture overview

A short random seed can produce a much longer string that looks random to every efficient observer. The starting point is surprisingly small: one extra bit.

One-way permutations, the Goldreich–Levin bit, and iteration.

Primary source: Lecture 15 [Maj26].

Abstract and lecture roadmap

A pseudorandom generator expands a uniformly random seed into a longer string. Its output need not be statistically close to uniform. The requirement is that efficient algorithms cannot tell the difference.

1. Define what it means for a string to look random.
2. Use a one-way permutation and a hard-core bit to gain one bit.
3. Repeatedly apply a one-bit generator to obtain polynomial stretch.
4. Use the resulting string to encrypt a message longer than the key.

Proof roadmap

Prove security for one step. Then replace the steps one at a time and add the distinguishing gaps.

[BS23, §§3.1–3.4]; motivation: [HPS14, §1.7.5].

Main results

One extra bit from a one-way permutation

Let $f_n : \{0, 1\}^n \rightarrow \{0, 1\}^n$ be a one-way permutation. Then

$$G(r, x) = (r, f_n(x), \langle r, x \rangle)$$

is a pseudorandom generator from $2n$ bits to $2n + 1$ bits.

One extra bit is enough

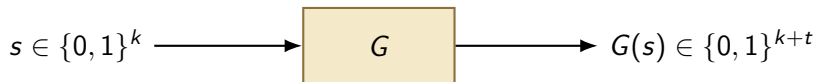
Every one-bit PRG on k -bit seeds yields a PRG with output length $k + t(k)$, for every positive, efficiently computable polynomial stretch $t(k)$.

Goldreich–Levin proves the first result. A hybrid argument proves the second.

[GL89]; iteration: [BS23, Thm. 3.3, pp. 59–61].

A short seed, a long string

Suppose Alice and Bob already share a uniformly random k -bit secret. They would like to use it to hide a much longer message.



The computational goal

No efficient test should distinguish $G(s)$ from a uniformly random string of the same length with a noticeable advantage.

The generator is public. The seed supplies the secret randomness.

[BS23, §§3.1–3.2]; [HPS14, §1.7.5, pp. 44–45].

The pseudorandomness experiment

Write U_j for a uniformly random j -bit string. A test D receives either

$$G(U_k) \quad \text{or} \quad U_{k+t(k)}$$

and outputs one bit. Define its distinguishing advantage by

$$\text{Adv}_{G,D}(k) = \left| \Pr[D(1^k, G(U_k)) = 1] - \Pr[D(1^k, U_{k+t(k)}) = 1] \right|.$$

Definition: pseudorandom generator (PRG)

A PRG is a deterministic polynomial-time family $G_k : \{0, 1\}^k \rightarrow \{0, 1\}^{k+t(k)}$ with $t(k) > 0$ and negligible advantage for every probabilistic polynomial-time test D .

Negligible means smaller than k^{-c} eventually, for every fixed $c > 0$. We suppress the public security parameter in later formulas.

[BS23, Def. 3.1, §3.1.1, pp. 46–47].

The output is statistically far from uniform

A deterministic map $G : \{0, 1\}^k \rightarrow \{0, 1\}^{k+t}$ has at most 2^k possible outputs. Let $S = \text{Im}(G)$.

$$\Pr[G(U_k) \in S] = 1, \quad \Pr[U_{k+t} \in S] = \frac{|S|}{2^{k+t}} \leq 2^{-t}.$$

A counting bound

The total variation distance satisfies

$$\text{TV}(G(U_k), U_{k+t}) \geq 1 - 2^{-t}.$$

Even one extra bit gives distance at least $1/2$.

An unbounded observer can enumerate every seed and test membership in S . Pseudorandomness says that efficient observers cannot exploit such a gap.

Compare [BS23, §3.1.1, p. 47]; source slides, pp. 3–4 [Maj26].

A balanced bit can still be easy to recognize

Consider these tempting expansions of a uniform k -bit seed s .

Proposed output	Efficient test	Advantage
$(s, 0)$	Is the last bit zero?	$1/2$
(s, s_1)	Does the last bit equal the first?	$1/2$
$(s, s_1 \oplus \dots \oplus s_k)$	Is the last bit the parity of the first k ?	$1/2$

The whole output matters

In the last two examples, each individual bit is uniform. A simple relation among the bits still reveals the distribution.

The extra bit must remain hidden even when the rest of the output is visible.

A permutation preserves the uniform distribution

Recall that a one-way permutation $f_n : \{0, 1\}^n \rightarrow \{0, 1\}^n$ is a bijection that is efficient to evaluate and hard to invert on a uniform input.

Uniformity needs only bijectivity

For every $y \in \{0, 1\}^n$,

$$\Pr[f_n(U_n) = y] = \Pr[U_n = f_n^{-1}(y)] = 2^{-n}.$$

Thus $f_n(U_n)$ is exactly uniform.

If R, X are independent and uniform in $\{0, 1\}^n$, then

$$(R, f_n(X)) \text{ is uniform in } \{0, 1\}^{2n}.$$

Append one bit that stays hidden from efficient observers given this uniform prefix.

Source slides, p. 6 [Maj26]; one-wayness: [BS23, §8.11.1].

The Goldreich–Levin bit

For $r, x \in \{0, 1\}^n$, define their inner product modulo 2 by

$$\langle r, x \rangle = r_1x_1 \oplus r_2x_2 \oplus \cdots \oplus r_nx_n.$$

It is the parity of the coordinates of x selected by r .

Example

For $r = (1, 0, 1, 0)$ and $x = (1, 1, 0, 1)$,

$$\langle r, x \rangle = 1 \cdot 1 \oplus 0 \cdot 1 \oplus 1 \cdot 0 \oplus 0 \cdot 1 = 1.$$

Computing the bit is easy given r and x . The question is whether it remains predictable given only r and $f_n(x)$.

[GL89].

The Goldreich–Levin theorem

Technical input: a hard-core predicate

Let $f_n : \{0, 1\}^n \rightarrow \{0, 1\}^n$ be a one-way function. If R, X are independent uniform n -bit strings, then every efficient predictor A satisfies

$$\Pr[A(R, f_n(X)) = \langle R, X \rangle] \leq \frac{1}{2} + \text{negl}(n).$$

The probability also includes the predictor's random bits.

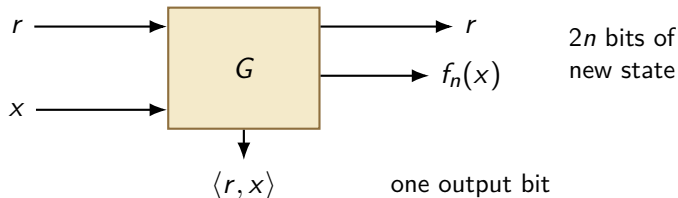
- Guessing a fair bit always succeeds with probability $1/2$.
- A useful predictor must improve on $1/2$ by a nonnegligible amount.
- The theorem averages over R and X . It does not make a claim for every fixed r .

We use this theorem without proof.

One extra bit from a one-way permutation

Assume now that f_n is a one-way permutation. Define

$$G : \{0, 1\}^{2n} \rightarrow \{0, 1\}^{2n+1}, \quad G(r, x) = (r, f_n(x), \langle r, x \rangle).$$



Length accounting

The seed contains both r and x : $2n$ bits in, $2n + 1$ bits out.

Source slides, pp. 9–10 [Maj26]; [GL89].

Why does the extra bit look random?

Let R, X be independent uniform strings and let C be an independent fair bit.

Step 1: replace the hard-core bit

For every efficient test,

$$(R, f_n(X), \langle R, X \rangle) \approx_c (R, f_n(X), C).$$

A distinguisher would predict the hard-core bit. The appendix proves this implication.

Step 2: identify the distribution

The permutation makes $(R, f_n(X))$ uniform. The independent fair bit C makes the full tuple exactly U_{2n+1} .

Evaluation is efficient, so all parts of the PRG definition now hold.

Hard-core theorem: [GL89].

Where the permutation hypothesis enters

Goldreich–Levin applies to every one-way function. The simple PRG formula also needs the visible prefix to be uniform.

Why a general one-way function does not suffice here

If $h : \{0, 1\}^{n-1} \rightarrow \{0, 1\}^{n-1}$ is one-way, then

$$f(x_1, z) = (0, h(z))$$

is one-way, but its first output bit is always zero. Appending a hard-core bit cannot hide that visible defect.

A broader theorem

One-way functions imply PRGs, via a more involved construction.

General theorem: Håstad–Impagliazzo–Levin–Luby [HILL99].

A one-bit generator as a state update

From now on, let k be the seed length of any one-bit PRG

$$G : \{0, 1\}^k \rightarrow \{0, 1\}^{k+1}.$$

Split its output into a new k -bit state and one bit:

$$G(s_i) = (s_{i+1}, b_i).$$

The iteration

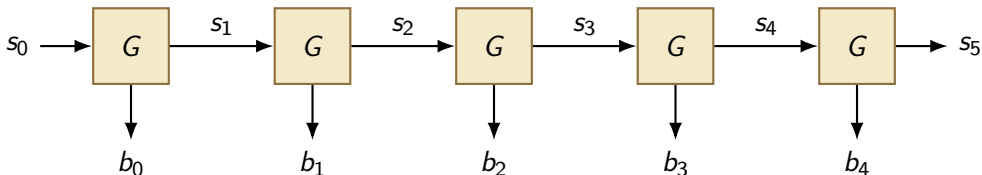
Start with $s_0 = s$. For $i = 0, \dots, t - 1$, compute $(s_{i+1}, b_i) = G(s_i)$, and return

$$G^{[t]}(s) = (s_t, b_0, b_1, \dots, b_{t-1}).$$

The final state contributes k bits. The emitted bits contribute t more. The total output length is $k + t$.

Sequential construction: [BS23, §3.4.2, pp. 59–61].

Five rounds of expansion



The complete output

$$G^{[5]}(s_0) = (s_5, b_0, b_1, b_2, b_3, b_4) \in \{0, 1\}^{k+5}.$$

The state stays k bits long throughout the computation. Each round contributes exactly one additional bit to the final output.

Compare the five-round diagram in the supplied slides, p. 12 [Maj26].

The stretch theorem and its proof plan

Theorem

If $G : \{0, 1\}^k \rightarrow \{0, 1\}^{k+1}$ is a PRG and $t = t(k) > 0$ is polynomially bounded and efficiently computable, then $G^{[t]}$ is a PRG from k bits to $k + t$ bits.

1. Start with the actual t -round output.
2. Replace the first transition by a fresh uniform state and bit.
3. Do the same with the next transition, until every output bit is uniform.
4. Show that each replacement is just one use of the PRG assumption.

We do not assume that the actual intermediate states are independent of previously emitted bits. The proof handles their dependence.

Compare [BS23, Thm. 3.3, pp. 59–61].

The intermediate distributions

For $0 \leq j \leq t$, define a distribution H_j by the following experiment.

1. Sample $b_0, \dots, b_{j-1}$ as independent fair bits.
2. Independently sample a uniform state $s_j \in \{0, 1\}^k$.
3. Run $t - j$ real transitions and return $(s_t, b_0, \dots, b_{t-1})$.

Distribution	Earlier bits	Remaining computation
H_0	None	All t real transitions
H_j	j independent fair bits	$t - j$ real transitions
H_t	t independent fair bits	Independent uniform final state

Thus $H_0 = G^{[t]}(U_k)$ and $H_t = U_{k+t}$.

One replacement is a one-step PRG test

Suppose D tries to distinguish H_j from H_{j+1} . Build a test B_j that receives a $(k+1)$ -bit challenge (z, c) .

1. Sample the earlier bits $b_0, \dots, b_{j-1}$ independently.
2. Set $s_{j+1} = z$ and $b_j = c$.
3. Run the later transitions starting from z .
4. Give the assembled output to D and return its decision.

Two exact simulations

If $(z, c) \sim G(U_k)$, the assembled distribution is H_j .

If $(z, c) \sim U_{k+1}$, the assembled distribution is H_{j+1} .

The reduction only evaluates G forward. It never needs to recover an earlier state.

Adding the gaps proves security

Write $p_j = \Pr[D(H_j) = 1]$. A single test B samples j uniformly from $\{0, \dots, t-1\}$ and runs the preceding simulation.

$$\begin{aligned} & \Pr[B(G(U_k)) = 1] - \Pr[B(U_{k+1}) = 1] \\ &= \frac{1}{t} \sum_{j=0}^{t-1} (p_j - p_{j+1}) = \frac{p_0 - p_t}{t}. \end{aligned}$$

The security loss is a factor of t

$$\text{Adv}_{G^{[t]}, D}(k) = t \text{Adv}_{G, B}(k).$$

The test B takes the time of D plus at most t evaluations of G .

A polynomial factor preserves negligible advantage. This completes the proof.

The same linear loss appears in [BS23, Thm. 3.3].

Returning to the permutation construction

For $G(r, x) = (r, f_n(x), \langle r, x \rangle)$, the state and emitted bit at round i are

$$s_i = (r, f_n^{\circ i}(x)), \quad b_i = \langle r, f_n^{\circ i}(x) \rangle,$$

where $f_n^{\circ 0}(x) = x$ and $f_n^{\circ i}$ means i repeated applications.

After five rounds

$$G^{[5]}(r, x) = (r, f_n^{\circ 5}(x), \langle r, x \rangle, \langle r, f_n(x) \rangle, \langle r, f_n^{\circ 2}(x) \rangle, \langle r, f_n^{\circ 3}(x) \rangle, \langle r, f_n^{\circ 4}(x) \rangle).$$

This is a $(2n + 5)$ -bit output from a $2n$ -bit seed.

The same r is used in every round. The hybrid proof establishes the pseudorandomness of the whole tuple, including its final state.

Source construction, pp. 11–12 [Maj26].

Length doubling and longer output

A length-doubling generator

For a k -bit state, choose $t = k$. Then

$$G^{[k]} : \{0, 1\}^k \longrightarrow \{0, 1\}^{2k}.$$

In the Goldreich–Levin construction, $k = 2n$. Thus $2n$ rounds expand a $2n$ -bit seed into $4n$ bits.

Any desired polynomial output length

For $L(k) > k$, choose $t = L(k) - k$. The resulting generator has output length $L(k)$ and takes t iterations.

One may also keep only the emitted bits: any test of those bits becomes a test of the full output by discarding the final state.

A message longer than the key

Let $P : \{0, 1\}^k \rightarrow \{0, 1\}^L$ be a PRG, with $L > k$. Alice and Bob share $K \xleftarrow{\$} \{0, 1\}^k$.

Encryption

For $m \in \{0, 1\}^L$, send

$$c = m \oplus P(K).$$

The key has k bits. The message and ciphertext have L bits.

Decryption

Compute

$$c \oplus P(K) = m.$$

Both parties can reproduce the same pad from their shared seed.

Security roadmap. Replace $P(K)$ by an independent uniform L -bit pad. This gives the ordinary one-time pad, which hides the message perfectly.

This construction is for one encryption under a fresh key.

[BS23, §3.2, Thm. 3.1, pp. 48–51]; [HPS14, §1.7.5].

The encryption proof in one reduction

An efficient observer chooses L -bit messages m_0, m_1 , sees an encryption of m_β for a fair hidden bit β , and guesses β .

A test of a candidate pad w

Choose a fair β , give $m_\beta \oplus w$ to the observer, and accept if it guesses β .

- If $w = P(K)$, this is the real encryption experiment.
- For uniform w , the ciphertext is independent of β . Success is exactly $1/2$.

Conclusion

The advantage over $1/2$ equals a PRG distinguishing advantage, hence is negligible.

[BS23, Thm. 3.1, pp. 49–51].

What the construction guarantees

The pad is used only once

Reusing the same key in this construction reuses the same pad:

$$c_1 \oplus c_2 = (m_1 \oplus P(K)) \oplus (m_2 \oplus P(K)) = m_1 \oplus m_2.$$

Protecting multiple messages requires an additional construction.

The adversary model is part of the assumption

The proofs here concern classical probabilistic polynomial-time algorithms. A quantum security claim requires assumptions and reductions valid against quantum algorithms.

Changing the underlying hard problem can change the security of an instantiation. The construction alone does not make a broken assumption secure.

Pad reuse: [BS23, §3.2.1].

The lecture in four statements

1. **Pseudorandomness is computational.** A PRG's output can be statistically far from uniform while fooling every efficient test.
2. **Goldreich–Levin supplies the hidden bit.** The pair $(r, f_n(x))$ hides $\langle r, x \rangle$ from efficient predictors.
3. **Bijectivity supplies the uniform prefix.** Together, these give a one-bit PRG with a $2n$ -bit seed.
4. **Hybrids justify repeated use.** Polynomially many iterations lose only a polynomial factor in security.

The resulting application

A uniformly random short key can hide one longer message against efficient observers.

Questions to check your understanding

1. Why does (s, s_1) fail to be pseudorandom even though each coordinate is a fair bit?
2. Goldreich–Levin permits $r = 0^n$, for which the inner product is always zero. Why is there no contradiction?
3. What is the output length after t rounds of the permutation construction? What must be included to obtain that length?
4. Why does the hybrid reduction sample the earlier bits itself rather than try to reconstruct earlier states?
5. Why does the stretch proof require a polynomial bound on t ? What goes wrong with $t = 2^k$?
6. If the same pseudorandom pad encrypts two messages and one message is known, how can the other be recovered?

Appendix: a distinguisher gives a predictor

Let $Y = (R, f_n(X))$ and $B = \langle R, X \rangle$. For a test D , put

$$\delta = \Pr[D(Y, B) = 1] - \Pr[D(Y, C) = 1],$$

where C is an independent fair bit.

Predicting B from Y

Choose $c \xleftarrow{\$} \{0, 1\}$. Run $D(Y, c)$. If it accepts, output c ; otherwise, output $1 - c$.

For fixed $(Y, B) = (y, b)$, write $q_c(y) = \Pr[D(y, c) = 1]$. The predictor succeeds with probability

$$\frac{1}{2}q_b(y) + \frac{1}{2}(1 - q_{1-b}(y)) = \frac{1}{2} + q_b(y) - \frac{q_0(y) + q_1(y)}{2}.$$

Averaging gives success probability $1/2 + \delta$.

Appendix: finishing the hard-core reduction

If $\delta > 0$, the preceding predictor has advantage δ . If $\delta < 0$, complement its output to obtain advantage $-\delta$.

Goldreich–Levin bounds the success of both fixed efficient predictors, so $|\delta|$ is negligible. Consequently,

$$(Y, B) \approx_c (Y, C).$$

What this argument proves

A bit that no efficient algorithm can predict with nonnegligible advantage is computationally indistinguishable from a fresh fair bit, even with the side information Y present.

When f_n is a permutation, (Y, C) is exactly uniform. This completes the detailed proof of the one-bit generator.

The only external ingredient is the hard-core theorem [GL89].

Technical input: Goldreich–Levin

Hard-core predicate theorem used in the proof

Let (f_n) be a one-way family on uniformly sampled n -bit inputs. For independent $R, X \xleftarrow{\$} \{0, 1\}^n$, every probabilistic polynomial-time algorithm A satisfies

$$\Pr[A(1^n, R, f_n(X)) = \langle R, X \rangle] \leq \frac{1}{2} + \text{negl}(n).$$

Citation: O. Goldreich and L. A. Levin, *A Hard-Core Predicate for All One-Way Functions*, STOC 1989, pp. 25–32 [GL89]. A precise restatement is Håstad et al., §4.1, Proposition 4.1, p. 1376 [HILL99].

What is assumed and what is proved here

We assume a one-way permutation and use Goldreich–Levin. The one-bit construction, polynomial expansion, and one-message encryption guarantee are proved in these slides.

Further reading and the general existence theorem

External theorem mentioned for perspective

If one-way functions exist, then pseudorandom generators exist. This statement does not require one-way permutations and is not used in our proofs.

Citation: J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby, *A Pseudorandom Generator from Any One-way Function*, SIAM J. Comput. 28(4), 1364–1396 (1999), Theorem 6.3, p. 1387 [HILL99].

Reading guide

Boneh–Shoup: Definition 3.1, pp. 46–47 (PRGs); Theorem 3.1, pp. 49–51 (stream encryption); Theorem 3.3, pp. 59–61 (sequential expansion) [BS23].

Hoffstein–Pipher–Silverman: §1.7.5, pp. 44–45 (random bit sequences and symmetric ciphers); §8.2, pp. 474–477 (pseudorandom number generation) [HPS14].

References I

- [BS23] Dan Boneh and Victor Shoup.
A Graduate Course in Applied Cryptography.
Author-published online textbook, 2023.
Version 0.6, January 2023.
URL: https://crypto.stanford.edu/~dabo/cryptobook/BonehShoup_0_6.pdf.
- [GL89] Oded Goldreich and Leonid A. Levin.
A hard-core predicate for all one-way functions.
In *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing*, pages 25–32.
ACM, 1989.
URL: <https://www.wisdom.weizmann.ac.il/~oded/X/gl.pdf>, doi:10.1145/73007.73010.
- [HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby.
A pseudorandom generator from any one-way function.
SIAM Journal on Computing, 28(4):1364–1396, 1999.
URL: <https://www.cs.bu.edu/fac/lnd/pdf/hill.pdf>, doi:10.1137/S0097539793244708.

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 15: Pseudo-random generators.
CS355, Purdue University, Fall semester, 2026.
Instructor's supplied lecture slides, 14 pages.