

Pseudorandom Functions

Lecture 18: A short key for a huge table

Lecture overview

A pseudorandom function lets us compute any entry of a seemingly random table from one short secret key. We use it for encryption, construct it from a pseudorandom generator, and examine an algebraic alternative.

Random functions, the GGM tree, and the Naor–Reingold construction.

Based on the instructor's Lecture 18 [Maj26].

Abstract and lecture roadmap

Last lecture, a secret random function supplied one pad per input. Its table was enormous. Today, Alice and Bob keep only a short key and compute the entry they need.

1. Define what it means to look like a random function.
2. Replace the random table in our encryption scheme.
3. Build a function by following a path through a PRG tree.
4. Compare this construction with the Naor–Reingold PRF.

Proof roadmap

Each proof replaces a pseudorandom object by a genuinely random one. Once that replacement is made, independence does the remaining work.

[BS23, §§4.4, 4.6, 5.4.1]; [GGM86, NR04].

Main results

Throughout, k is the security parameter. Input and output lengths, and the number of queries, are polynomially bounded in k .

PRFs give stateless encryption

If $F_K: \{0, 1\}^m \rightarrow \{0, 1\}^n$ is a secure PRF, then

$$\text{Enc}_K(M) = (M \oplus F_K(R), R), \quad R \xleftarrow{\$} \{0, 1\}^m,$$

is secure against chosen-plaintext attacks whenever nonce collisions are negligible. Taking $m = k$ suffices for polynomially many encryptions.

PRGs give PRFs: the GGM theorem

A secure generator $H: \{0, 1\}^k \rightarrow \{0, 1\}^{2k}$ gives a secure PRF on m -bit inputs. One evaluation follows m branches of a binary tree. An output map adjusts the final length to n bits.

[BS23, Thms. 4.10 and 5.2]; [GGM86].

What the key has to replace

	Random function f	Pseudorandom function F_K
Stored secret	$n2^m$ table bits	A k -bit key K
Evaluation at x	Look up row x	Run a public algorithm on (K, x)
Security	Exact randomness	Indistinguishable to efficient observers

Notation for today

m : input length. n : output or message-block length.
 k : key length. q : number of queries or encryptions.
 K : a secret key. M : an n -bit message block.

The family has at most 2^k functions, compared with 2^{n2^m} possible tables.

[BS23, §4.4.1 and Def. 4.2, pp. 132–133].

Two experiments with the same interface

An observer chooses an input, receives an output, and may choose its next input using everything it has seen.

Real experiment

Sample $K \xleftarrow{\$} \{0, 1\}^k$ once.

On query $x \in \{0, 1\}^m$, return
 $F_K(x)$.

Ideal experiment

Sample a uniform function $f: \{0, 1\}^m \rightarrow \{0, 1\}^n$ once.

On query x , return
 $f(x)$.

In both experiments, querying the same input again returns the same answer. Only a *new* input exposes a fresh ideal table entry.

[BS23, Attack Game 4.2 and §4.4.2, pp. 132–134].

Pseudorandom means indistinguishable under queries

Write A^O for an algorithm that may ask an oracle O for function values and then outputs a bit.

PRF advantage

$$\text{Adv}_F^{\text{prf}}(A) = \left| \Pr_K[A^{F_K} = 1] - \Pr_f[A^f = 1] \right|.$$

The family is a **secure PRF** if this advantage is negligible in k for every probabilistic polynomial-time observer A .

- The evaluation algorithm $F_K(x)$ must be efficient.
- Queries may be adaptive: later inputs can depend on earlier answers.
- The observer knows the algorithm, but does not know K .

A negligible quantity is eventually smaller than k^{-c} for every fixed $c > 0$.

[BS23, Def. 4.2, p. 133].

Uniform individual outputs are not enough

For a simple example, let $m = n = k$ and define $F_K(x) = K \oplus x$ with uniform K .

At every fixed input x , the value $F_K(x)$ is exactly uniform. But two queries reveal a rigid relation:

$$F_K(x) \oplus F_K(y) = x \oplus y.$$

A two-query distinguisher

Query distinct inputs x, y . Output 1 exactly when the relation above holds.

- In the real experiment, it always holds.
- For a random function, it holds with probability 2^{-k} .

The distinguishing advantage is $1 - 2^{-k}$.

The PRF definition controls the whole observed collection of values, including their relationships.

Definition and oracle viewpoint: [BS23, §4.4].

Encryption using a pseudorandom function

Key generation

$$\text{Gen}() : K \xleftarrow{\$} \{0, 1\}^k.$$

Only the k -bit key is secret.

Encrypt $M \in \{0, 1\}^n$

Sample $R \xleftarrow{\$} \{0, 1\}^m$. Set

$$C = M \oplus F_K(R).$$

Return (C, R) .

Decrypt (C, R)

Return

$$\text{Dec}_K(C, R) = C \oplus F_K(R).$$

The public nonce selects the pad.

Each call samples an independent nonce. Alice and Bob keep no persistent stream position.

Source slides, p. 3 [Maj26]; [BS23, Ex. 5.2, pp. 183–184].

Correctness is exact; secrecy uses pseudorandomness

Correctness

For every key, message, and nonce,

$$\text{Dec}_K(M \oplus F_K(R), R) = M \oplus F_K(R) \oplus F_K(R) = M.$$

Secrecy roadmap. Replace F_K by a random function f . At distinct nonces, the values $f(R)$ are independent uniform pads. Apply Lecture 17's collision bound.

Repeated nonces still expose a relation

If $R_i = R_j$, then $C_i \oplus C_j = M_i \oplus M_j$. A short key does not change this calculation.

This scheme supplies confidentiality against chosen-plaintext attacks. Modifying C also modifies the recovered message, so authentication requires an additional mechanism.

[BS23, Thm. 5.2, pp. 184–187; §6.1].

The quantitative encryption guarantee

Fix a hidden bit b . On request $(M_{i,0}, M_{i,1})$, return an encryption of $M_{i,b}$. The observer may choose later pairs adaptively. Put

$$\text{Adv}^{\text{enc}}(A) = |\Pr[A \text{ outputs } 1 \mid b = 1] - \Pr[A \text{ outputs } 1 \mid b = 0]|.$$

PRF error plus collision error

For suitable oracle distinguishers B_0, B_1 making at most q queries,

$$\text{Adv}^{\text{enc}}(A) \leq \text{Adv}_F^{\text{prf}}(B_0) + \text{Adv}_F^{\text{prf}}(B_1) + \frac{\binom{q}{2}}{2^m}.$$

If both PRF advantages are at most ε , the bound is $2\varepsilon + \binom{q}{2}/2^m$.

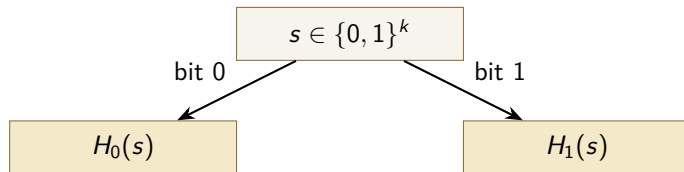
q counts all encrypted blocks under this key. With $m = k$, polynomial q gives a negligible collision term. The appendix gives the reduction with this advantage convention.

Compare [BS23, Thm. 5.2, Eq. (5.11)]; birthday background: [HPS14, §5.4].

A PRG supplies the two children of a seed

Let $H: \{0, 1\}^k \rightarrow \{0, 1\}^{2k}$ be a secure pseudorandom generator. Split its output into two k -bit blocks:

$$H(s) = H_0(s) \parallel H_1(s).$$



What security of H says

For uniform s , the *pair* $(H_0(s), H_1(s))$ is computationally indistinguishable from two independent uniform k -bit strings.

[BS23, §4.6, pp. 147–151]; [GGM86].

An input names a path through the tree

Use K as the root label. Read $x = x_1 \dots x_m$ from left to right. At bit x_i , move to child x_i .

The k -bit leaf label

$$s_0 = K, \quad s_i = H_{x_i}(s_{i-1}) \quad (1 \leq i \leq m), \quad T_K(x) = s_m.$$

Equivalently,

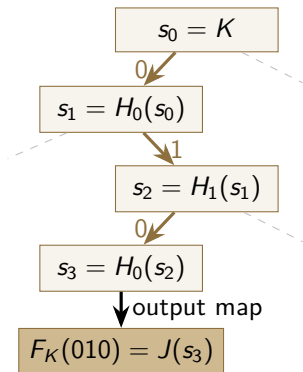
$$T_K(x) = H_{x_m}(\dots H_{x_2}(H_{x_1}(K)) \dots).$$

The full tree has 2^m leaves, but a single evaluation computes only one path. Different inputs can share part of that path.

All inputs in this construction have the same fixed length m .

Source slides, pp. 4–5 [Maj26]; [BS23, Thm. 4.10].

Worked path: input 010



Read the bits in order

0: take the left child.

1: take the right child.

0: take the left child.

Every intermediate label has k bits.

The map J converts the final label to an n -bit output.

The dashed branches need no further evaluation.

The input 010 is the example in the source slides, p. 5 [Maj26].

Adjusting the output length

Choose an efficient deterministic map $J: \{0, 1\}^k \rightarrow \{0, 1\}^n$ such that

$$J(U_k) \approx_c U_n,$$

where U_j is a uniform j -bit string and $\approx_c$ denotes computational indistinguishability.

Desired length	Choice of J	Distribution on a uniform seed
$n < k$	First n bits	Exactly uniform
$n = k$	Identity	Exactly uniform
$n > k$	A secure length- n PRG	Pseudorandom

The final function

$$F_K(x) = J(T_K(x)) = J(H_{x_m}(\cdots H_{x_1}(K) \cdots)).$$

GGM tree: [BS23, §4.6]. Polynomial output extension: [BS23, §3.4].

GGM evaluation: pseudocode and cost

Evaluate $F_K(x)$

- 1 Parse $x = x_1 \dots x_m$ and set $s \leftarrow K$.
- 2 For $i = 1, \dots, m$:
 Compute $(y_0, y_1) \leftarrow H(s)$, with k bits per half.
 Set $s \leftarrow y_{x_i}$.
- 3 Return $J(s)$.

- One evaluation uses m calls to H and one call to J .
- The key has k bits. The intermediate seed also has k bits.
- Repeating an input gives the same path and the same answer.

Every evaluation starts at K ; the key stays fixed.

Source slides, p. 7 [Maj26]; [BS23, §4.6].

Shared prefixes explain the proof challenge

Consider queries 010 and 011. They have the same first two bits:

$$s_{01} = H_1(H_0(K)), \quad T_K(010) = H_0(s_{01}), \quad T_K(011) = H_1(s_{01}).$$

One expansion produces both answers

The two leaf labels form the pair $H(s_{01})$. They come from one PRG call, so the proof must treat them together.

Roadmap. Replace tree levels from the root downward. When a node's label has become uniform, replace its pair of children by two independent uniform strings. Preserve the labels already assigned to each tree position.

Generate only queried paths. With q queries, at most qm internal-node expansions are encountered.

Hybrid proof of [BS23, Thm. 4.10, pp. 148–151].

Why the GGM tree looks like a random function

1. **Replace one level at a time.** Each adjacent comparison uses the PRG guarantee on a uniform seed. At most q reached nodes occur at a level.
2. **Reach the leaves.** After m levels, distinct input strings have independent uniform leaf labels. Repeated inputs reuse the same label.
3. **Apply the output map.** Replace J of each new uniform leaf by an independent uniform n -bit output. There are at most q such replacements.

The quantitative bound

If the relevant single-sample advantages for H and J are at most ε_H and ε_J , then

$$\text{Adv}_F^{\text{prf}}(A) \leq qm\varepsilon_H + q\varepsilon_J.$$

The reductions include the time needed to simulate the observer and its tree.

Tree bound: [BS23, Thm. 4.10, p. 148]. Output-map step: standard q -term hybrid.

Where the pseudorandom generator comes from

Let k be even and let $f: \{0, 1\}^{k/2} \rightarrow \{0, 1\}^{k/2}$ be an efficiently computable one-way permutation.

For $r, z \in \{0, 1\}^{k/2}$, define the inner-product bit

$$h(r, z) = \langle r, z \rangle = \sum_{j=1}^{k/2} r_j z_j \pmod{2}.$$

Goldreich–Levin supplies the hidden bit

For independent uniform r, z , the bit $h(r, z)$ is computationally indistinguishable from a fresh uniform bit even when $(r, f(z))$ is given.

The state update $P(r, z) = (r, f(z))$ is itself a permutation. The vector r stays fixed while z advances through iterations of f .

[GL89]; sequential generation: [BS23, §3.4.2].

The iterative generator in the source slides

For a polynomially bounded output length t , define $\text{Stream}_t: \{0,1\}^k \rightarrow \{0,1\}^t$ as follows.

Input: a seed $s = (r, z)$ with two $k/2$ -bit halves

- 1 Initialize an empty list of bits.
- 2 For $i = 1, \dots, t$:
 Append $\langle r, z \rangle$ to the list.
 Set $z \leftarrow f(z)$.
- 3 Return the list.

The output is

$$\langle r, z_0 \rangle, \langle r, f(z_0) \rangle, \dots, \langle r, f^{t-1}(z_0) \rangle.$$

For uniform s , the output is pseudorandom. When $t > k$, it is a PRG with stretch. See the appendix for the proof.

Source slides, p. 6 [Maj26]; [GL89]; [BS23, Thm. 3.3, p. 59].

Putting the constructions together

Component	Choice
Underlying assumption	A one-way permutation f on $k/2$ bits
Length-doubling PRG	$H(s) = \text{Stream}_{2k}(s)$
Output map	$J(s) = \text{Stream}_n(s)$, or truncation if $n \leq k$
PRF	$F_K(x) = J(H_{x_m}(\cdots H_{x_1}(K)\cdots))$
Encryption	$(M \oplus F_K(R), R)$ with fresh uniform R

What this establishes

One-way permutations suffice to construct the PRFs and encryption scheme in this lecture. GGM itself only requires a secure length-doubling PRG.

[GL89, GGM86]; [BS23, §§3.4.2, 4.6, 5.4.1].

An algebraic alternative: Naor–Reingold

Let $\mathbb{G} = \langle g \rangle$ be a cyclic group of prime order ℓ , with efficient group operations. Assume DDH in this group.

A concrete group setting

Take primes p, ℓ with $\ell \mid p - 1$, and choose $g \in \mathbb{F}_p^*$ of order ℓ . Then $\mathbb{G}$ is the subgroup generated by g .

Choose the secret key

$$a = (a_0, a_1, \dots, a_m) \xleftarrow{\$} \mathbb{F}_\ell^{m+1}.$$

For $x = x_1 \dots x_m \in \{0, 1\}^m$, define

$$F_a(x) = g^{a_0 \prod_{i=1}^m a_i^{x_i}} \in \mathbb{G}.$$

Use exponents modulo ℓ , with $a_i^0 = 1$ also when $a_i = 0$.

[BS23, Ex. 10.4, p. 424]; [NR04]. Groups and DH: [HPS14, §2.3].

A small Naor–Reingold evaluation

Use $p = 23$, $\ell = 11$, and $g = 2$, which has order 11 modulo 23. Take $m = 3$ and $a = (3, 2, 5, 7)$.

Input x	Exponent $e(x) \pmod{11}$	Output $2^{e(x)} \pmod{23}$
000	3	8
010	$3 \cdot 5 = 15 \equiv 4$	16
101	$3 \cdot 2 \cdot 7 = 42 \equiv 9$	6
111	$3 \cdot 2 \cdot 5 \cdot 7 = 210 \equiv 1$	2

What each input bit does

If $x_i = 1$, include the factor a_i . If $x_i = 0$, omit it. Always include a_0 .

These tiny parameters illustrate arithmetic only.

Evaluation formula: [NR04].

The assumption and the correct output space

Decisional Diffie–Hellman (DDH)

For independent uniform $u, v, w \in \mathbb{F}_\ell$, an efficient observer cannot distinguish

$$(g, g^u, g^v, g^{uv}) \quad \text{from} \quad (g, g^u, g^v, g^w).$$

The assumption concerns the chosen family of groups as the security parameter grows.

Naor–Reingold theorem (imported)

Under DDH in $\mathbb{G}$, the family $F_a: \{0, 1\}^m \rightarrow \mathbb{G}$ is a secure PRF for polynomially bounded m . The ideal comparison is a uniformly random function with values in $\mathbb{G}$.

In the finite-field setting, all outputs satisfy $y^\ell = 1$. A uniform element of $\mathbb{F}_p^*$ satisfies this with probability $\ell/(p-1)$. Thus the ambient group is not the ideal output space when $\mathbb{G}$ is proper.

[BS23, Ex. 10.4]; [NR04, Thm. 4.1, pp. 245–246].

Why this evaluation offers more parallelism

GGM

At step i , compute

$$s_i = H_{x_i}(s_{i-1}).$$

Each update needs the preceding result. A path has m successive steps.

Naor–Reingold

Select the factors $a_i^{x_i}$ independently. Multiply them using a balanced product tree, then compute

$$g^{a_0} \prod_i a_i^{x_i}.$$

The factor product has $O(\log m)$ layers of field multiplications.

The $O(\log m)$ count is for the factor product. Modular arithmetic and exponentiation also contribute to the total cost.

Source slides, p. 8 [Maj26]; [NR04].

The same formula in an additive group

Let P generate a prime-order subgroup $\mathbb{G}$ of an elliptic-curve group. Assume DDH in this subgroup, whose order is ℓ .

Replace exponentiation by scalar multiplication

$$F_a(x) = \left(a_0 \prod_{i=1}^m a_i^{x_i} \right) P.$$

Compute the scalar in $\mathbb{F}_\ell$, then multiply the point P by that scalar.

For the input 010, the output is $(a_0 a_2)P$. The key and selected-factor rule are the same as in multiplicative notation.

The ideal function now returns independent uniform points of $\mathbb{G}$ at new inputs. Its encoded points need not be uniform bit strings.

Source slides, p. 8 [Maj26]; [NR04]; [HPS14, §§6.2–6.4].

Comparing the two constructions

	GGM	Naor–Reingold
Starting assumption	A secure PRG	DDH in a prime-order group
Secret key	One k -bit seed	$m + 1$ field elements
Input	An m -bit path	An m -bit selection of factors
Evaluation	Successive PRG calls	Factor product and exponentiation
Output	n bits after J	One element of $\mathbb{G}$

For the XOR encryption scheme

We need pseudorandom *bit strings*. A group-element encoding need not be uniform as a bit string. Using Naor–Reingold there requires a separately justified conversion or a group-based encryption scheme.

[GGM86, NR04]; [BS23, §4.6; Ex. 10.4].

Questions to take away

1. Why does the ideal oracle remember its answer to a repeated input?
2. Why is $K \oplus x$ a poor PRF even though every individual output is uniform?
3. Which GGM computations do inputs 010 and 011 share?
4. What are the two error terms in the encryption security proof?
5. Why must the Naor–Reingold ideal function take values in $\mathbb{G}$?

The construction principle

A short seed can support an enormous collection of evaluations. Security asks whether their joint behavior can be distinguished from one consistent random table.

Appendix: the encryption reduction

Let p_b be the probability that A outputs 1 in real experiment b , and $\tilde{p}_b$ the probability with a uniform function f instead.

The reduction B_b . Given oracle O , answer the i th request by choosing $R_i \xleftarrow{\$} \{0, 1\}^m$ and returning $(M_{i,b} \oplus O(R_i), R_i)$. Output whatever A outputs. Hence

$$|p_b - \tilde{p}_b| = \text{Adv}_F^{\text{prf}}(B_b).$$

The ideal comparison. Until a nonce repeats, lazy sampling supplies a fresh uniform pad. Couple the two ideal experiments using the same nonces, observer coins, and masked ciphertexts until that event. Their views then agree.

Triangle inequality

$$\begin{aligned} |p_1 - p_0| &\leq |p_1 - \tilde{p}_1| + |\tilde{p}_1 - \tilde{p}_0| + |\tilde{p}_0 - p_0| \\ &\leq \text{Adv}_F^{\text{prf}}(B_1) + \text{Adv}_F^{\text{prf}}(B_0) + \binom{q}{2} 2^{-m}. \end{aligned}$$

[BS23, Thm. 5.2, pp. 184–187].

Appendix: how to read the GGM hybrids

Let $\mathcal{H}_j$ use independent uniform labels at depth j and the ordinary H -tree below. Generate only queried paths.

- $\mathcal{H}_0$ is the real tree: one uniform root K .
- $\mathcal{H}_m$ assigns independent uniform labels directly to leaves.
- To compare $\mathcal{H}_j$ and $\mathcal{H}_{j+1}$, replace $H(U_k)$ by U_{2k} at each reached depth- j position.

At each depth, replace at most q positions in order of first encounter. Store each pair for consistent replies on later visits. This costs at most $q\varepsilon_H$ per level.

Summing the comparisons

$$|\Pr[A^{\mathcal{H}_0} = 1] - \Pr[A^{\mathcal{H}_m} = 1]| \leq mq\varepsilon_H.$$

After applying J at the leaves, at most q additional hybrids cost $q\varepsilon_J$.

Full adaptive reduction: [BS23, Thm. 4.10, pp. 148–151].

Appendix: why the iterative generator works

Set $P(r, z) = (r, f(z))$ and $h(r, z) = \langle r, z \rangle$. Goldreich–Levin gives the one-bit extension

$$(P(U_k), h(U_k)) \approx_c (U_k, U_1).$$

Roadmap. Prove pseudorandomness even when the final state is included. Each step extends the state and keeps the already generated bits.

For a uniform initial state S , consider

$$(P^t(S), h(S), h(P(S)), \dots, h(P^{t-1}(S))).$$

An induction on t compares the preceding state-and-bits distribution with independent uniform strings. Apply the efficient update to both sides, then use the one-bit extension on the uniform state.

Polynomially many extensions

Each step adds at most one hard-core distinguishing error. Thus the final state together with the t bits is pseudorandom for polynomial t . Discarding the final state gives $\text{Stream}_t(S)$.

Hard-core input: [GL89]. Sequential extension: [BS23, Thm. 3.3, p. 59].

Appendix: the scope of the assumptions

Polynomial lengths and fixed input size

Take m, n, q, t polynomially bounded in the security parameter, so polynomial hybrid losses preserve negligible error. GGM here uses a fixed input length m .

The group family matters

Assume DDH in the actual group $\mathbb{G}$ used by Naor–Reingold. Prime order alone does not establish DDH. The order grows with the security parameter.

Uniform keys in $\mathbb{F}_\ell^{m+1}$ may contain zero. Under the convention $a_i^0 = 1$, the function remains defined. The probability of any zero key coordinate is at most $(m+1)/\ell$, negligible in the usual exponentially large-order setting.

[BS23, §4.6.1; Ex. 10.4]; [NR04].

Boneh–Shoup, version 0.6 (January 2023)

- Attack Game 4.2 and Definition 4.2, pp. 132–133: PRF security.
- §4.6, Theorem 4.10, pp. 147–151: the GGM tree construction.
- Example 5.2 and Theorem 5.2, pp. 183–187: PRF encryption.
- Theorem 3.3, p. 59: sequential PRG extension.
- Exercise 10.4, p. 424: the Naor–Reingold PRF.

Hoffstein–Pipher–Silverman, second edition (2014)

§2.3: Diffie–Hellman key exchange. §5.4: birthday estimates.

§§6.2–6.4: elliptic-curve groups and cryptography.

For the PRF constructions themselves, use Boneh–Shoup and the original papers.

[BS23, HPS14]. Page numbers refer to printed pages in the stated editions.

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.
- [GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali.
How to construct random functions.
Journal of the ACM, 33(4):792–807, 1986.
doi:10.1145/6490.6503.
- [GL89] Oded Goldreich and Leonid A. Levin.
A hard-core predicate for all one-way functions.
In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, pages 25–32. ACM, 1989.
doi:10.1145/73007.73010.

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 18: Pseudorandom functions.
CS355, Purdue University, Fall semester, 2026.
Instructor's supplied lecture slides, 8 pages.
- [NR04] Moni Naor and Omer Reingold.
Number-theoretic constructions of efficient pseudo-random functions.
Journal of the ACM, 51(2):231–262, 2004.
doi:10.1145/972639.972643.

Technical inputs: pseudorandom generators

Goldreich–Levin hard-core theorem

If $f: \{0,1\}^d \rightarrow \{0,1\}^d$ is a one-way permutation, then for independent uniform $X, R \in \{0,1\}^d$ and $B \in \{0,1\}$,

$$(f(X), R, \langle R, X \rangle) \approx_c (f(X), R, B).$$

Source: Goldreich–Levin, *A Hard-Core Predicate for All One-Way Functions*, STOC 1989, §3, Theorem 1, p. 28 [GL89]. This is the permutation specialization of their hard-core theorem.

Sequential extension theorem

A secure generator with one extra output bit can be iterated to obtain polynomial stretch. The distinguishing loss is at most the number of extensions times the relevant one-step advantage.

Source: Boneh–Shoup, v0.6, §3.4.2, Theorem 3.3, printed p. 59 [BS23]. Applied here to $(r, z) \mapsto (r, f(z), \langle r, z \rangle)$.

Technical inputs: the PRF theorems

GGM tree theorem

For a secure length-doubling PRG, the depth- m tree defines a secure PRF from $\{0, 1\}^m$ to $\{0, 1\}^k$. For q adaptive queries the reduction loses a factor at most mq , allowing the stated simulation time.

Source: Boneh–Shoup, v0.6, Theorem 4.10, p. 148, with proof pp. 148–151 [BS23]; original construction [GGM86].

Naor–Reingold theorem

Under DDH in an efficiently represented prime-order group $\mathbb{G} = \langle g \rangle$, the functions $F_a(x) = g^{a_0 \prod_i a_i^{x_i}}$, for uniform $a \in \mathbb{F}_\ell^{m+1}$ and polynomial m , form a secure PRF with range $\mathbb{G}$.

Source: Naor–Reingold, *Number-Theoretic Constructions of Efficient Pseudo-Random Functions*, JACM 51(2), 2004, Construction 4.1 and Theorem 4.1, pp. 245–246 [NR04]; see also Boneh–Shoup, Exercise 10.4, p. 424 [BS23].