Verifiable Random Functions (Micali Rabin Vadhan, FOCS 1999): A Worked Summary with Constructions and Proof Ideas

Abstract

A verifiable random function (VRF) is a function keyed by a secret key that outputs, on any input x, a value y together with a non-interactive proof π that y is correct, such that y remains indistinguishable from random at any input for which no proof has been released. Micali Rabin Vadhan (FOCS 1999) formalize VRFs with a strong *unique provability* property and construct VRFs from an RSA root-extraction hardness assumption. The construction proceeds in three conceptual steps: (1) build a weaker primitive called a verifiable unpredictable function (VUF) from RSA; (2) lift VUFs to VRFs using a Goldreich Levin (GL) hardcore predicate idea; (3) extend fixed-length VRFs to all inputs $\{0,1\}^*$ via a tree-based composition with prefix-free encoding.

Contents

1 Roadmap (What is built and why) 2

2 Notation and background primitives 2

3 Primitives: VRFs and VUFs 3
- 3.1 VRFs (syntax and properties) . . . 3
- 3.2 Residual pseudorandomness (the VRF security game) . . . 3
- 3.3 VUFs: verifiable unpredictability . . . 3

4 Hardness assumption used by the construction 4
- 4.1 RSA root-extraction with random large prime exponent . . . 4

5 Construction 1: RSA-based VUF (value and proof) 4
- 5.1 High-level idea . . . 4
- 5.2 Prime indexing: $x \mapsto p_x$ . . . 4
- 5.3 Algorithms . . . 4
- 5.4 A compact correctness/uniqueness picture . . . 5
- 5.5 Core properties and their proofs . . . 5
- 5.6 Residual unpredictability: the reduction structure . . . 6

6 Construction 2: Lifting VUFs to VRFs via Goldreich Levin 6
- 6.1 Construction (bit-valued VRF) . . . 6
- 6.2 Why this yields pseudorandomness (proof idea) . . . 6

7 Construction 3: Extending the input domain to $\{0,1\}^*$ **7**
7.1 Tree-based composition . . . 7
7.2 Proof structure . . . 7
7.3 Prefix-free encoding . . . 8
7.4 Security idea (what must be shown) . . . 8

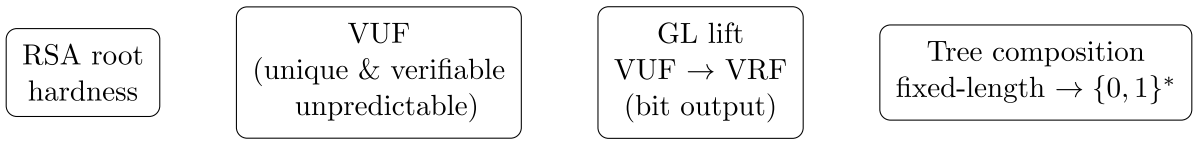
8 Summary tables (what each step guarantees) **8**
8.1 Primitive checklist . . . 8
8.2 Construction and cost overview . . . 9

9 End-to-end statement (what you obtain) **9**

10 Practical reading notes (what to remember) **9**

1 Roadmap (What is built and why)

The paper's approach can be read as a pipeline of reductions:


RSA root hardness
VUF (unique & verifiable unpredictable)
GL lift VUF → VRF (bit output)
Tree composition fixed-length → $\{0,1\}^*$

The remainder of this document:

- defines the primitives and security games precisely enough to reason about them,
- gives the concrete RSA-based VUF (including the proof/verification mechanism),
- explains the GL lift and why it yields pseudorandomness from unpredictability (with the key reduction idea),
- gives the domain-extension construction and its security intuition,
- summarizes the overall theorem and typical costs (proof size, verification work).

2 Notation and background primitives

Efficient algorithms and negligible functions. All algorithms are probabilistic polynomial-time (PPT) in the security parameter k. A function $\mathsf{negl} : \mathbb{N} \to R_{\geq 0}$ is negligible if $\forall$ polynomials p there exists k_0 such that for $k \geq k_0$, $\mathsf{negl}(k) < 1/p(k)$.

Groups modulo an RSA modulus. Let $m = pq$ be an RSA modulus with distinct odd primes p, q. Let $\mathbb{Z}_m$ denote integers modulo m and $\mathbb{Z}_m^*$ its multiplicative group of units. Euler's totient is $\varphi(m) = (p-1)(q-1)$.

Lemma 1 (Permutation by exponentiation)**.** *If* $\gcd(e, \varphi(m)) = 1$*, the map* $\phi_e : \mathbb{Z}_m^* \to \mathbb{Z}_m^*$ *defined by* $\phi_e(z) = z^e \bmod m$ *is a permutation.*

Proof. Because $\gcd(e, \varphi(m)) = 1$, there exists d with $ed \equiv 1 \pmod{\varphi(m)}$. Then for all $z \in \mathbb{Z}_m^*$, $(z^e)^d = z^{ed} \equiv z \pmod m$ by Euler's theorem, so ϕ_d is the inverse of ϕ_e. □

Inner product over $\mathbb{F}_2$. For $a, r \in \{0,1\}^b$,

$$\langle a, r \rangle \;=\; \sum_{i=1}^{b} a_i r_i \pmod 2.$$

3 Primitives: VRFs and VUFs

3.1 VRFs (syntax and properties)

A VRF is a triple $(\mathsf{G}, \mathsf{F}, \mathsf{V})$:

- $\mathsf{G}(1^k) \to (\mathsf{PK}, \mathsf{SK})$.
- $\mathsf{F}(\mathsf{SK}, x) \to (y, \pi)$ where y is the function value and π is a proof.
- $\mathsf{V}(\mathsf{PK}, x, y, \pi) \in \{0,1\}$ is the public verification algorithm.

Definition 1 (Correctness and complete provability)**.** *For all x in the input domain, if* $(\mathsf{PK}, \mathsf{SK}) \leftarrow \mathsf{G}(1^k)$ *and* $(y, \pi) \leftarrow \mathsf{F}(\mathsf{SK}, x)$*, then*

$$\Pr[\mathsf{V}(\mathsf{PK}, x, y, \pi) = 1] \geq 1 - \mathsf{negl}(k).$$

Definition 2 (Unique provability)**.** *For any public key* PK *(even adversarially chosen) and any input x, there do not exist two distinct values $y \neq y'$ with proofs π, π' such that* $\mathsf{V}(\mathsf{PK}, x, y, \pi) = \mathsf{V}(\mathsf{PK}, x, y', \pi') = 1$*, except with negligible probability.*

3.2 Residual pseudorandomness (the VRF security game)

Informally: even after seeing many (y, π) pairs at chosen inputs, the value at a fresh input looks random.

Definition 3 (Residual pseudorandomness game)**.** *Let $\mathcal{O}_{\mathsf{SK}}(x)$ return $(y, \pi) \leftarrow \mathsf{F}(\mathsf{SK}, x)$. Adversary $\mathcal{A}$ gets* PK *and oracle access to $\mathcal{O}_{\mathsf{SK}}$, outputs a fresh $x^\star$ (never queried), then receives either the real $y^\star$ or a uniform string of the same length; it may continue querying on inputs $\neq x^\star$ and must guess which case it is in. The advantage is*

$$\mathsf{Adv}^{\mathsf{vrf}}_{\mathcal{A}}(k) = \left|\Pr[\mathcal{A} \textit{ guesses correctly}] - \tfrac{1}{2}\right|.$$

A VRF is secure if this advantage is negligible for all PPT *adversaries.*

3.3 VUFs: verifiable unpredictability

The paper introduces a weaker primitive: a *verifiable unpredictable function* (VUF). The syntax, correctness, and unique provability are the same; the security goal is *unpredictability* rather than pseudorandomness.

Definition 4 (Residual unpredictability (VUF security))**.** *Given* PK *and oracle access to* $\mathcal{O}_{\mathsf{SK}}(x) = (y, \pi)$*, no* PPT *adversary can output a fresh input $x^\star$ and a pair $(y^\star, \pi^\star)$ such that* $\mathsf{V}(\mathsf{PK}, x^\star, y^\star, \pi^\star) = 1$*, except with negligible probability.*

Remark 1. *A VUF is closely aligned with a* unique *signature scheme where the signature is the proof and the message is the input. The uniqueness condition matches unique provability.*

4 Hardness assumption used by the construction

4.1 RSA root-extraction with random large prime exponent

The core assumption is that extracting p-th roots modulo an RSA modulus is hard when p is a random large prime.

Definition 5 (RSA root hardness (informal))**.** *Let m be an RSA modulus of size $\approx k$ bits, let $x \leftarrow \mathbb{Z}_m^*$ be uniform, and let p be a random $(k+1)$-bit prime (hence $p > m$). Given (m, x, p), it is hard for any $s(k)$-time adversary to find y such that*

$$y^p \equiv x \pmod{m}.$$

Why $p > m$ is useful for VRFs. Because $p > m > \varphi(m)$ and p is prime, we have $\gcd(p, \varphi(m)) = 1$, so the p-th root in $\mathbb{Z}_m^*$ (if it exists) is *unique*. This uniqueness is exactly what the construction exploits to satisfy unique provability.

5 Construction 1: RSA-based VUF (value and proof)

5.1 High-level idea

Fix a public element $r \in \mathbb{Z}_m^*$. For each input x, associate a (publicly computable) prime exponent p_x. Define the function value to be the unique p_x-th root of r:

$$v_x := r^{1/p_x} \pmod{m}.$$

The proof is simply the witness v_x itself, and verification checks $v_x^{p_x} \equiv r \pmod{m}$.

5.2 Prime indexing: $x \mapsto p_x$

The construction needs a public map sending each x to a (nearly always distinct) prime p_x of about $(k+1)$ bits. The paper uses a *prime-sequence generator* (based on limited-independence polynomials and primality testing coins) to ensure that with overwhelming probability, all p_x are prime and distinct over the intended domain.

For this summary, treat the following as a black box:

> **Prime Indexer $\mathsf{Prime}(x)$:** a public deterministic algorithm (given some public seed) that outputs a $(k+1)$-bit prime p_x for each input x, and outputs distinct primes for distinct x except with negligible probability.

5.3 Algorithms

Key generation.

$$\mathsf{G}(1^k): \quad \text{pick RSA modulus } m = pq; \text{ pick } r \leftarrow \mathbb{Z}_m^*; \text{ publish seed for } \mathsf{Prime}(\cdot).$$

Public key:

$$\mathsf{PK} = (m, r, \text{seed}), \qquad \mathsf{SK} = (\mathsf{PK}, \varphi(m)).$$

Evaluation (value + proof). On input x:

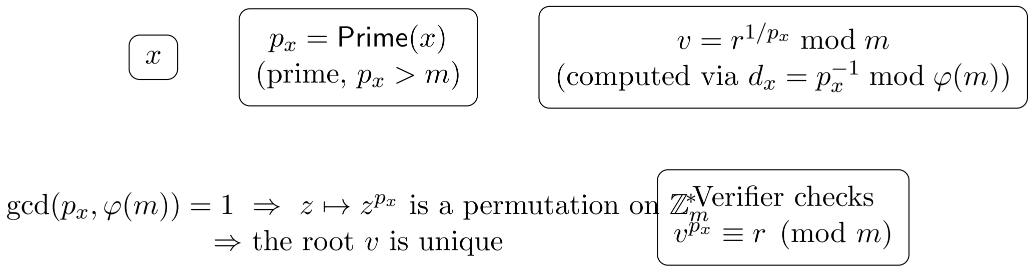
$$p_x := \mathsf{Prime}(x), \qquad d_x := p_x^{-1} \bmod \varphi(m), \qquad v := r^{d_x} \bmod m, \qquad \pi := v.$$

Output (v, π) (the proof is the same element).

Verification. On (x, v, π) (with $\pi = v$):

1. Compute $p_x := \mathsf{Prime}(x)$; verify p_x is prime (with fresh randomness) and enforce $p_x > m$.
2. Check $v \in \mathbb{Z}_m^*$ and
$$v^{p_x} \equiv r \pmod{m}.$$
3. Accept iff all checks pass.

5.4 A compact correctness/uniqueness picture



x
$p_x = \mathsf{Prime}(x)$ (prime, $p_x > m$)
$v = r^{1/p_x} \bmod m$ (computed via $d_x = p_x^{-1} \bmod \varphi(m)$)
$\gcd(p_x, \varphi(m)) = 1 \Rightarrow z \mapsto z^{p_x}$ is a permutation on $\mathbb{Z}_m^*$
$\Rightarrow$ the root v is unique
Verifier checks $v^{p_x} \equiv r \pmod{m}$

Figure 1: RSA-based VUF: the "proof" is the root v itself; verification is one modular exponentiation.

5.5 Core properties and their proofs

Lemma 2 (Completeness)**.** *If $(v, \pi) \leftarrow \mathsf{F}(\mathsf{SK}, x)$ then $\mathsf{V}(\mathsf{PK}, x, v, \pi) = 1$ except with negligible probability.*

Proof. Let $p = p_x$ and $d = d_x$ with $pd \equiv 1 \pmod{\varphi(m)}$, so $pd = 1 + t\varphi(m)$ for some integer t. Then $v = r^d$ satisfies

$$v^p \equiv (r^d)^p = r^{pd} = r^{1+t\varphi(m)} \equiv r \cdot (r^{\varphi(m)})^t \equiv r \pmod{m},$$

since $r \in \mathbb{Z}_m^*$ implies $r^{\varphi(m)} \equiv 1 \pmod{m}$. □

Lemma 3 (Unique provability)**.** *For any PK and input x, there is at most one $v \in \mathbb{Z}_m^*$ such that $v^{p_x} \equiv r \pmod{m}$ (except with negligible probability due to primality-test error).*

Proof. Verification enforces that p_x is prime and $p_x > m$, hence $p_x \nmid \varphi(m)$ because $\varphi(m) < m$. Thus $\gcd(p_x, \varphi(m)) = 1$ and exponentiation by p_x is a permutation on $\mathbb{Z}_m^*$. A permutation has at most one preimage for r, so at most one v can satisfy $v^{p_x} \equiv r$. □

5.6 Residual unpredictability: the reduction structure

The key security statement: producing a correct (v, π) for a fresh $x^\star$ is as hard as extracting a root with a fresh exponent. The paper's reduction uses two central ingredients:

1. **Programming one exponent:** choose a target $x^\star$ and construct the public seed so that $p_{x^\star} = p$ where p is the RSA challenge exponent (while keeping the seed distribution close to honest).

2. **Algebraic simulation:** set $r := u^E \bmod m$ where u comes from the RSA challenge and $E = \prod_{x \neq x^\star} p_x$. Then for any $x \neq x^\star$ one can answer the oracle query by outputting

$$v_x := u^{E/p_x} \bmod m,$$

since $(v_x)^{p_x} = u^E = r$.

If the adversary outputs a valid $v^\star$ for $x^\star$, then $(v^\star)^p = r = u^E$. Using Bézout coefficients α, β with $\alpha E + \beta p = 1$, one extracts $u^{1/p}$ as

$$u^{1/p} \equiv (v^\star)^\beta \cdot u^\alpha \pmod{m}.$$

Theorem 1 (RSA-based VUF security (informal))**.** *Assuming RSA root-extraction is hard for random large primes, the above construction is a secure VUF: any* PPT *adversary that forges a valid value at a fresh input with non-negligible probability yields an RSA-root inverter with non-negligible probability (up to the standard "guess the challenge input" factor when selecting $x^\star$).*

6 Construction 2: Lifting VUFs to VRFs via Goldreich Levin

6.1 Construction (bit-valued VRF)

Let the VUF output be encoded as a b-bit string $v(x) \in \{0,1\}^b$. Sample a public random $r \in \{0,1\}^b$ and define the derived output bit:

$$y(x) := \langle v(x), r \rangle \bmod 2.$$

Evaluation returns a proof that reveals $v(x)$ and proves it is correct:

$$\mathsf{F}'(\mathsf{SK}, x): \ (v, \pi) \leftarrow \mathsf{F}(\mathsf{SK}, x), \quad y = \langle v, r \rangle, \quad \pi' = (v, \pi).$$

Verification checks both:

$$\mathsf{V}'(\mathsf{PK}', x, y, \pi') = 1 \iff \Big(\mathsf{V}(\mathsf{PK}, x, v, \pi) = 1\Big) \wedge \Big(y = \langle v, r \rangle\Big),$$

where $\mathsf{PK}' = (\mathsf{PK}, r)$ and $\pi' = (v, \pi)$.

6.2 Why this yields pseudorandomness (proof idea)

Goldreich Levin (GL) says: if you can predict $\langle w, r \rangle$ for random r with noticeable advantage, you can reconstruct w with non-negligible probability (using $\mathsf{poly}(k)$ queries to the predictor).

Here, the role of w is played by $v(x^\star)$ at a fresh input $x^\star$. So, if an adversary distinguishes $y(x^\star)$ from uniform, one can convert it into a predictor for $\langle v(x^\star), r \rangle$, then apply GL reconstruction to recover $v(x^\star)$ itself. But recovering $v(x^\star)$ (together with its proof) breaks the VUF unpredictability.

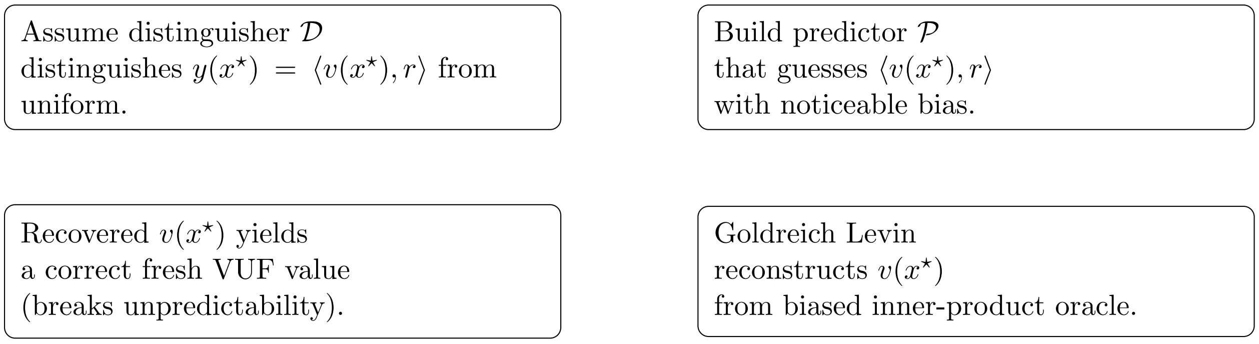

Assume distinguisher $\mathcal{D}$ distinguishes $y(x^\star) = \langle v(x^\star), r\rangle$ from uniform.
Build predictor $\mathcal{P}$ that guesses $\langle v(x^\star), r\rangle$ with noticeable bias.
Recovered $v(x^\star)$ yields a correct fresh VUF value (breaks unpredictability).
Goldreich Levin reconstructs $v(x^\star)$ from biased inner-product oracle.

Figure 2: Security flow for the GL lift: distinguisher $\Rightarrow$ predictor $\Rightarrow$ GL reconstruction $\Rightarrow$ VUF break.

The technical nuisance addressed in the paper: the adversary chooses $x^\star$ adaptively, while GL wants a *fixed* $w = v(x^\star)$. The reduction handles this by choosing a random $x^\star$ and hoping it matches the adversary's exam point (the familiar $2^{-|x|}$ loss), motivating an initial "short-input" regime; the next construction removes this restriction.

Theorem 2 (VUF $\Rightarrow$ VRF (informal))**.** *If* $(\mathsf{G}, \mathsf{F}, \mathsf{V})$ *is a secure VUF with unique provability, then the GL-derived* $(\mathsf{G}', \mathsf{F}', \mathsf{V}')$ *is a (bit-valued) VRF for an appropriate parameter regime (with polynomial overhead and the standard challenge-point guessing loss).*

7 Construction 3: Extending the input domain to $\{0,1\}^*$

7.1 Tree-based composition

Assume a base VRF (or verifiable pseudorandom predicate) that can be applied iteratively. The idea is to label a binary tree so that each node label deterministically defines its children via the base VRF.

Conceptually:

- Root has label $L(\epsilon)$.
- For a node with label $L(w)$, define

  $$L(w0) := f(L(w), 0), \qquad L(w1) := f(L(w), 1),$$

  where f is a base VRF/predicate-to-string variant.
- For $x = b_1 \cdots b_t$, the output is $L(x)$, the label at the end of the path.

7.2 Proof structure

A proof for $x = b_1 \cdots b_t$ contains the intermediate labels and per-edge proofs:

$$\big(L(b_1), \pi_1\big),\ \big(L(b_1 b_2), \pi_2\big),\ \ldots,\ \big(L(b_1 \cdots b_t), \pi_t\big),$$

where π_i proves that $L(b_1 \cdots b_i)$ was computed correctly from $L(b_1 \cdots b_{i-1})$ and bit b_i.

7.3 Prefix-free encoding

To prevent revealing values on prefixes of future challenges, the input x is first mapped to a prefix-free encoding $\widehat{x}$ (so no $\widehat{x}$ is a prefix of $\widehat{x}'$ for $x \neq x'$). The tree walk is done on $\widehat{x}$ instead of x.

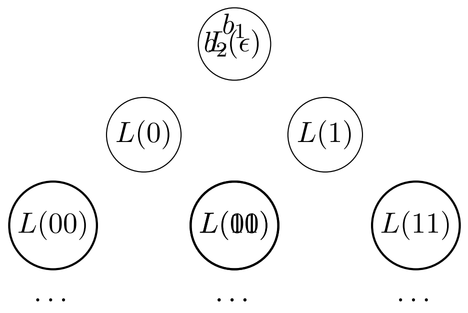


Figure 3: Domain extension: compute $L(\widehat{x})$ by walking the tree. Proofs certify each edge transition.

7.4 Security idea (what must be shown)

Fix an adversary that sees many path proofs and then challenges on a fresh $\widehat{x}$. Two complementary arguments are used:

- **Conditioned-on-no-collisions:** if no label repeats among nodes revealed so far, then the challenge label is distributed like a fresh base-VRF output at an unseen point, hence pseudorandom.
- **Collisions are unlikely (or exploitable):** if label repetitions happen with noticeable probability, one can leverage the first collision to predict a new label and contradict the base VRF security (label lengths are chosen to push collision probability down, and any non-negligible collision probability yields a distinguisher/predictor).

Theorem 3 (Fixed-length $\Rightarrow$ unrestricted-length VRF (informal))**.** *Given a secure fixed-length VRF with sufficiently long labels (output length), the tree construction (with prefix-free encoding) yields a secure VRF on* $\{0,1\}^*$*, with proof size and verification time linear in* $|\widehat{x}|$*.*

8 Summary tables (what each step guarantees)

8.1 Primitive checklist

Primitive	**Verifiable?**	**Unique proof?**	**Security goal**
VUF	yes	yes	unpredictability at fresh points
VRF	yes	yes	pseudorandomness at fresh points
Tree-extended VRF	yes	yes	pseudorandomness on $\{0,1\}^*$

Table 1: Conceptual distinction: VUF vs. VRF.

8.2 Construction and cost overview

Step	What is built	Main cost
RSA $\Rightarrow$ VUF	$v_x = r^{1/p_x} \bmod m$ with proof $\pi = v_x$	One exponentiation for verify; indexer computes p_x
GL lift	Bit output $y = \langle v, r\rangle$ with proof revealing v	Proof includes v (and VUF proof), plus GL reduction overhead
Tree extension	VRF on $\{0,1\}^*$ via path labels and proofs	Proof size $\Theta(\|\widehat{x}\|)$; verify $\Theta(\|\widehat{x}\|)$

Table 2: Each reduction adds structure (and typically proof length) while preserving verifiability and uniqueness.

9 End-to-end statement (what you obtain)

Theorem 4 (End-to-end outcome (informal))**.** *Assuming RSA root-extraction is hard for random large prime exponents, there exists a VRF family on inputs* $\{0,1\}^*$ *that outputs at least one pseudorandom bit together with a publicly verifiable, uniquely valid proof of correctness. The construction is explicit: RSA-based VUF* $\Rightarrow$ *GL-derived verifiable pseudorandom predicate* $\Rightarrow$ *tree-based domain extension.*

10 Practical reading notes (what to remember)

- **The proof is not a separate object in the RSA VUF:** the witness v_x *is* the proof.
- **Uniqueness is structural, not heuristic:** enforcing p_x prime and $p_x > m$ makes exponentiation a permutation.
- **Security reductions have two recurring motifs:** (i) *program a special input* and pay a "guess the challenge" loss, (ii) *algebraic simulation* that answers all other queries consistently without knowing secret structure.
- **Domain extension trades interaction for proof length:** proofs grow linearly with input length.