

Master Briefing Document: Advanced Browser Fingerprinting & Stateless Telemetry Engineering

Executive Summary

Stateless browser fingerprinting relies on interrogating client-side runtime environments to synthesize persistent, unique identifiers without storing client-side state (such as cookies, IndexedDB, or LocalStorage). Rather than querying isolated, easily spoofed properties, modern telemetry operates across a multi-layered matrix of mathematical information theory, browser engine architecture, network protocol invariants, hardware-accelerated graphics/compute shaders, digital signal processing (DSP) pipelines, and physical device topologies.

Key Takeaways:

- Entropy Thresholds & Mathematical Uniqueness** : Isolating a unique device among an estimated global internet population of 5.4×10^9 active users requires a minimum joint information entropy of 32.33 bits . Due to skewed feature distributions and architectural covariance, operational tracking engines require $40 \text{ to } 50 \text{ bits}$ of effective entropy. Maintaining a collision rate strictly below 0.1% across $1,000,000$ active devices demands at least 48.84 bits of effective entropy.
- Architectural Execution Invariants** : Browser engines expose deterministic microarchitectural signatures via JIT compilation tiers (V8's Ignition/Sparkplug/Maglev/TurboFan vs. SpiderMonkey's WarpMonkey vs. JSC's FTL/B3), hidden class transitions, and subpixel coordinate quantization. Blink (Chromium) calculates layout geometry using $1/64 \text{ th px}$ fixed-point units (LayoutUnit), whereas Gecko (Firefox) snaps coordinates to $1/60 \text{ th px}$ integer units (AppUnits/nscoord).
- Cleanroom Environment Introspection** : Client-side tampering by anti-detect browsers or browser extensions can be systematically unmasked by evaluating prototype chain placement, verifying native method stringification (native code), detecting internal slot execution failures via structuredClone(), and performing **Cleanroom Context Diffing** against pristine, ephemeral about:blank iframes.
- Transport & Network Layer Synchronization** : The transition to User-Agent Client Hints (UA-CH; RFC 8942 / RFC 9651) introduces explicit server negotiation via Accept-CH and Critical-CH. Spoofed client profiles frequently fail cross-layer invariant checks, such as mismatching HTTP/2 HPACK header frame ordering, violating modern Chromium unreduced User-Agent string rules (where minor/build/patch versions must be 0.0.0), or misaligning Windows 11 kernel build numbers (platformVersion $\geq 13.0.0$) with WebGL graphics backends.
- Hardware-Bound Graphics & Compute Silicon** : Custom GLSL (WebGL) and WGSL (WebGPU) compute shaders evaluate transcendental functions (sin, cos, exp, log), ill-conditioned matrix inversions, and subnormal floating-point numbers directly on physical GPU execution units. Microscopic Unit in the Last Place (ULP) arithmetic drift, Fused Multiply-Add (FMA) behavior, and shared-memory workgroup bandwidth reveal the physical GPU silicon family (e.g., NVIDIA Ampere/Ada vs. Apple Silicon vs. AMD RDNA) regardless of DOM-level string overrides.

6. **Acoustic & Non-Linear Signal Processing Telemetry** : Offline Web Audio processing reveals host CPU microarchitecture (x86_64 vs. ARM64) and compiler SIMD optimizations (AVX2/SSE vs. ARM NEON). Evaluating non-linear state-space equations in DynamicsCompressorNode highlights soft-knee spline calculations and subnormal float decay tails, exposing Flush-to-Zero (FTZ) / Denormals-Are-Zero (DAZ) flags on x86 processors vs. native hardware subnormal processing on ARM64 silicon. Defensive noise injection ("farbling") is unmasked by analyzing second-derivative (Δ^2) high-frequency curvature spikes across adjacent audio frames.

Section 1: Mathematical Foundations & Information Entropy

Client-side device identification models browser sessions as probabilistic partitioning problems across an observation space $\mathbf{\Omega}$. A fingerprinting engine applies a measurement function $\mathbf{\Phi}$ over a client instance u within a population $\mathcal{U} = \{u_1, u_2, \dots, u_N\}$ to construct a composite attribute vector: $\mathbf{\Phi}(u) = \mathbf{x} = \begin{bmatrix} x_1 & x_2 & \dots & x_d \end{bmatrix}^T$ $\mathbf{\Omega} = \mathcal{X}_1 \times \mathcal{X}_2 \times \dots \times \mathcal{X}_d$

Shannon Entropy & Uniqueness Bounds

Each observable feature X_k is a random variable defined over a support set $\text{supp}(X_k) = \{x_{k,1}, \dots, x_{k,m_k}\}$. The information yield or "surprisal" of observing a specific feature value $x_{k,i}$ is: $I(x_{k,i}) = -\log_2 P(x_{k,i})$. High-probability events yield minimal discriminative power, whereas rare hardware anomalies yield substantial identity resolution:

Event Type	Example Observable	$P(x)$	Information Yield $I(x)$
High Probability	Canvas Support = true	0.999	0.0014 bits
Low Probability	Rare Font Metric / Unique GPU Driver	0.00001	16.61 bits

(Low Discriminative Power) || **High Probability** | Canvas Support = true | 0.999 | 0.0014 bits (Low Discriminative Power) || **Low Probability** | Rare Font Metric / Unique GPU Driver | 0.00001 | 16.61 bits (High Discriminative Power) |

The expected uncertainty across a feature distribution is quantified by Shannon Information Entropy $H(X_k) = -\sum_{i=1}^{m_k} P(x_{k,i}) \log_2 P(x_{k,i})$. To uniquely isolate a single device within a population $N = |\mathcal{U}|$, the joint entropy $H(\mathbf{X})$ must satisfy: $H(\mathbf{X}) \geq \log_2(N)$

- For a global internet population of $N \approx 5.4 \times 10^9$ active users, $\log_2(5.4 \times 10^9) \approx 32.33$ bits.
- Because empirical feature distributions are skewed toward common baseline builds (e.g., standard Windows/Chrome configurations on 1080p displays), operational entropy targets require **40 to 50 bits**.

Feature Covariance & Conditional Entropy

In real-world telemetry, individual attributes exhibit deep architectural coupling (e.g., WebGLRenderingContext.RENDERER strongly correlates with navigator.hardwareConcurrency and antialiasing characteristics). Joint entropy is bounded from above by the sum of marginal entropies: $H(X_1, X_2, \dots, X_d) \leq \sum_{k=1}^d H(X_k)$. Equality holds only under strict mutual independence. To evaluate the independent contribution of a newly added feature Y given an existing feature vector $\mathbf{X}$, engines compute Conditional Entropy $H(Y \mid \mathbf{X})$ and Mutual Information

$I(\mathbf{X}; Y) = H(Y \mid \mathbf{X}) = -\sum_{\mathbf{x} \in \Omega} P(\mathbf{x}, y) \log_2 P(y \mid \mathbf{x})$
 $I(\mathbf{X}; Y) = H(Y) - H(Y \mid \mathbf{X}) = H(\mathbf{X}) + H(Y) - H(\mathbf{X}, Y)$
 Collecting redundant features (such as multiple WebGL extension strings hardcoded to the same driver build) collapses marginal information gain toward zero ($I(\text{Ext}_1; \text{Ext}_{2 \dots 20}) \approx H(\text{Ext}_{2 \dots 20})$), increasing bandwidth overhead without enhancing identity resolution.

Anonymity Sets & Collision Bounds

The Anonymity Set $\mathcal{A}(\mathbf{x}) \subseteq \mathcal{U}$ defines the set of devices sharing signature $\mathbf{x}$: $\mathcal{A}(\mathbf{x}) = \{u \in \mathcal{U} \mid \Phi(u) = \mathbf{x}\}$. A fingerprint is strictly unique if $|\mathcal{A}(\mathbf{x})| = 1$. Modeling collisions across n active devices interacting with an observation space of effective entropy H_{eff} (providing $K = 2^{H_{\text{eff}}}$ operational buckets) follows the generalized birthday distribution: $P_{\text{collision}}(n, K) \approx 1 - \exp\left(-\frac{n(n-1)}{2 \cdot 2^{H_{\text{eff}}}}\right) \approx \frac{n^2}{2 \cdot 2^{H_{\text{eff}}}} + 1$. Rearranging yields the effective entropy threshold required to guarantee a collision rate strictly below ϵ : $H_{\text{eff}} \geq 2 \log_2(n) - \log_2(\epsilon) - 1$. Active Devices (n) | Target Collision Threshold (ϵ) | Required Effective Entropy (H_{eff}) || ----- | ----- | ----- || **100,000** | < 0.001 (0.1%) | ~ 42.20 bits || **1,000,000** | < 0.001 (0.1%) | ~ 48.84 bits || **10,000,000** | < 0.0001 (0.01%) | ~ 58.81 bits |

Kullback-Leibler (KL) Divergence for Temporal Drift

Software auto-updates, browser version rollouts, and hardware upgrades cause empirical feature distributions to drift over time. To measure the information loss incurred when approximating a live distribution $Q(X)$ against a reference baseline $P(X)$, risk pipelines calculate Kullback-Leibler (KL) Divergence: $D_{\text{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log_2 \left(\frac{P(x)}{Q(x)} \right)$. Tracking D_{KL} over sliding time windows allows systems to dynamically recalibrate attribute surprisal weights and detect non-standard distributions injected by automated anti-fingerprinting poisoning tools.

Section 2: Browser Architecture & Execution Environment Telemetry

Multi-Process Topology & IPC Latency

Modern browsers isolate untrusted web contexts across discrete operating system processes:

- Browser Process** : Privileged supervisor managing storage coordination, network brokers, UI, and native window lifecycles.
- Renderer Process** : Sandboxed, unprivileged process executing JavaScript engines, DOM parsing, layout, and pre-paint computations. Site Isolation isolates cross-origin iframes (OOPIF) into separate virtual memory address spaces.
- GPU Process** : Centralized process multiplexing drawing commands, driving rasterization engines (Skia, Graphite, WebRender), and communicating directly with hardware graphics drivers via Direct3D, Metal, Vulkan, or OpenGL. Inter-Process Communication (IPC) is managed via platform primitives (Chromium's Mojo IPC,

Gecko's IPDL, or Mach ports on Darwin). Synchronous API readbacks forcing immediate GPU frame buffer extraction (e.g., `getImageData()` or `readPixels()`) trigger mandatory IPC synchronization barriers. The resulting round-trip latency exposes structural variations in IPC queue management, thread scheduling, and sandbox context switching.

JIT Compilation Tiers & Hidden Class Structures

Divergences across major JavaScript execution engines create timing and memory layout profiles:

- **Google V8 (Chromium)** : Executes source code through Ignition (bytecode interpreter) $\rightarrow$ Sparkplug (non-optimizing baseline compiler) $\rightarrow$ Maglev (mid-tier compiler) $\rightarrow$ TurboFan (optimizing JIT using a Sea-of-Nodes intermediate representation). Violating SMI (Small Integer) type assumptions triggers expensive deoptimization bailouts back to Ignition.
- **Mozilla SpiderMonkey (Firefox)** : Utilizes a C++ interpreter, Baseline Interpreter, Baseline Compiler, and WarpMonkey JIT (which translates CacheIR traces off-thread into MIR).
- **JavaScriptCore / JSC (Safari)** : Employs a four-tier pipeline: LLInt (Low-Level Assembly Interpreter), Baseline JIT, DFG (Data Flow Graph) JIT, and FTL (Faster Than Light, built on the B3 backend). Engines model dynamic object structures using internal shape transition models: V8 uses **Maps** , SpiderMonkey uses **Shapes/SharedShapes** , and JavaScriptCore uses **StructureIDs** . Injecting mutating property structures transitions Inline Caches (ICs) from monomorphic to megamorphic lookup states, exposing engine identity through deoptimization latency metrics.

Subpixel Layout Quantization

Layout engines calculate visual bounding boxes using distinct fractional spatial unit representations:

- **Blink LayoutNG (Chromium)** : Stores fractional dimensions using fixed-point arithmetic (LayoutUnit) with 6 fractional bits, representing spatial increments of $\frac{1}{64}$ CSS pixel (0.015625 px).
- **Gecko / Stylo (Firefox)** : Snaps internal spatial geometry to integer **AppUnits** (nscoord), where 1 CSS pixel = 60 AppUnits (0.016666667 px). Querying `getBoundingClientRect()` on deeply nested fractional flexbox hierarchies converts AppUnits back to double-precision floats, leaving distinct $0.016666\dots$ quantization residuals.
- **WebCore Layout (WebKit)** : Maintains direct float/double spatial representations.

Global Scope Introspection & Cleanroom Context Diffing

Traversing the global context (window, globalThis, self) captures exposed WebIDL interfaces across three distinct artifact tiers:

1. **Standard Interfaces** : Constructors defined by ECMAScript/W3C (AudioContext, WebGLRenderingContext, SharedArrayBuffer).
2. **Engine & Vendor Globals** : Non-standard properties injected by browser vendors (chrome.loadTimes, InstallTrigger, safari).

3. **Injected Automation & Extension Artifacts** : Globals introduced by testing frameworks or anti-detect proxies (window.cdc_adoQx080n..., **puppeteer_evaluation_script**, __playwright, __pwInitScripts).

Prototype Verification & Hook Unmasking

Anti-detect tools attempt to spoof properties like navigator.userAgent or navigator.hardwareConcurrency using Object.defineProperty or Proxy wrappers. These shims create detectable structural anomalies:

- **Instance Shadowing** : Native getters reside on prototypes (e.g., Navigator.prototype.userAgent). Defining userAgent directly on the navigator instance exposes direct shadowing.
- **Stringification Checks** : Invoking Function.prototype.toString.call(getter) on native accessors must strictly match the native regex signature `/^function get a-zA-Z0-9_$+\\(\\)s*\\{s*\\[native code\\]s*\\}$/`.
- **Internal Slot Failures** : Passing a Proxy-wrapped object to native methods requiring internal slots (e.g., executing structuredClone(target)) throws a TypeError indicating the object could not be cloned.

Ephemeral Cleanroom Context Diffing

To bypass top-level DOM tampering (window.top), telemetry scripts spawn an ephemeral, unattached about:blank iframe. Unless the spoofing tool intercepts dynamic DOM creation and synchronously patches child contexts prior to execution, the iframe inherits clean, unpolluted C++ native prototypes directly from the browser core. Side-by-side descriptor property diffing between window and iframe.contentWindow reveals altered property values and descriptor discrepancies.

Section 3: Protocol, Network, & User-Agent Client Hints (UA-CH) Forensics

UA-CH Architecture vs. Legacy Headers

Legacy User-Agent HTTP headers passively broadcast 30 to 40 bits of entropy on every outbound request. User-Agent Client Hints (UA-CH; RFC 8942 / RFC 9651) divide client metadata into low-entropy and high-entropy tiers:

- **Low-Entropy Hints** (Sent automatically): Sec-CH-UA (brand and major version), Sec-CH-UA-Mobile (?0 or ?1), Sec-CH-UA-Platform (e.g., "Windows", "macOS").
- **High-Entropy Hints** (Explicitly requested via Accept-CH): Sec-CH-UA-Arch ("x86", "arm"), Sec-CH-UA-Bitness ("64"), Sec-CH-UA-Full-Version-List, Sec-CH-UA-Model, Sec-CH-UA-Platform-Version, Sec-CH-UA-Form-Factors.

Critical-CH & Third-Party Delegation

To eliminate the asynchronous lag of Accept-CH on first-page visits, servers issue Critical-CH. Upon receiving Critical-CH, if the initial request lacked the required tokens, the browser aborts the rendering pass and immediately re-executes the GET request over the existing TLS connection, attaching the demanded high-entropy headers. Third-party iframe access requires explicit parent frame delegation via HTTP response headers:

Permissions-Policy: ch-ua-arch=(self "https://trusted.com").

GREASE Permutations

Browsers inject GREASE (Generate Random Extensions And Sustain Extensibility) tokens into Sec-CH-UA arrays (e.g., "Not/A)Brand";v="8", "Not=A?Brand";v="24") to prevent brittle regex matching. Robust backends process hints using RFC 9651 structured field parsers.

Network Invariants & Discrepancy Matrix

Cross-layer validation correlates passive network frames with client JS telemetry (`navigator.userAgentData.getHighEntropyValues()`):

1. **HTTP/2 HPACK Framing & Order** : Chromium network stacks group headers in deterministic sequences (:method, :authority, :scheme, :path, sec-ch-ua, sec-ch-ua-mobile, sec-ch-ua-platform, upgrade-insecure-requests, user-agent, accept, sec-fetch-*). Firefox does not send Sec-CH-UA headers and places user-agent immediately after :path.
2. **Chromium Legacy Reduction Invariant** : For Chromium v107+, legacy User-Agent strings are frozen to 0.0.0 for minor/build/patch components (e.g., Chrome/128.0.0.0). Transmitting an unreduced string like Chrome/128.0.6613.120 alongside modern Sec-CH-UA headers indicates a manipulated profile.
3. **OS Kernel Version Mapping** : On Windows 11, legacy headers claim Windows NT 10.0 due to backward compatibility. However, high-entropy hints disclose true NT build levels via platformVersion ("15.0.0" corresponds to Windows 11 22H2/23H2, whereas Windows 10 reports versions < 13.0.0).

Section 4: Graphics & Compute Subsystem Profiling (Canvas, WebGL, & WebGPU)

Canvas 2D Vector & Blend Execution

Drawing to a 2D canvas processes commands through vector engines (Google Skia in Chromium, Moz2D in Firefox, Core Graphics in WebKit) down to OS font rasterizers:

- **DirectWrite (Windows)** : Applies subpixel ClearType antialiasing with horizontal grid-fitting (hinting).
- **Core Text / Quartz 2D (macOS)** : Prioritizes design geometry over grid snapping, employing continuous floating-point advance widths and optical dilation.
- **FreeType (Linux/Android)** : Relies on system-level fontconfig rules, evaluating auto-hinting and custom gamma tables.

Vector Tessellation & Non-Linear Compositing

Advanced canvas extraction evaluates geometric path flattening and composite math:

- **Cubic Bézier Subdivision** : $B(t) = (1-t)^3 P_0 + 3(1-t)^2 t P_1 + 3(1-t) t^2 P_2 + t^3 P_3$. Curves are flattened into linear chords until distance falls below a flatness tolerance ϵ . Skia, Core Graphics, and Direct2D enforce different values for ϵ , producing varying boundary pixel coverage.
- **Winding Rules & Subpixel Clipping** : Evaluating overlapping paths using evenodd vs. nonzero winding rules over fractional coordinates (e.g., $x = 14.375$) forces rasterizers to calculate subpixel coverage along acute vertices.

- **Porter-Duff Blend Modes** : Non-separable blend modes (e.g., color-dodge, difference) execute non-linear color math: $B(C_b, C_s) = \min\left(1, \frac{C_b}{1 - C_s}\right)$ Un-premultiplying alpha channels ($\text{RGB} = \text{round}(R' / \alpha)$) introduces floating-point division rounding variations across hardware ALUs.

Anti-Spoofing & Invariant Probes

Privacy tools attempt to poison canvas readbacks:

- **Brave Farbling** : Injects deterministic, origin-seeded pseudo-random noise into the least significant bits (LSB) of color channels within C++ Skia readback routines.
- **Firefox RFP** : Blocks access or returns a white buffer (0xFFFFFFFF).
- **Extension Noise Poisoning** : Injecting random noise on every read violates the mathematical invariant $B(C, C) = |C - C| = 0$. Executing a self-canceling difference blend of a complex shape over itself must evaluate to pure zero (rgba(0,0,0,0)). Non-zero residual deltas ($\Delta > 0$) expose active per-call noise injection.

WebGL State Machine & Deep GPU Forensics

WebGL context probing queries parameters (gl.getParameter) and unmasked strings via WebGL_debug_renderer_info:

- UNMASKED_VENDOR_WEBGL / UNMASKED_RENDERER_WEBGL: Exposes the underlying hardware and translation backend (e.g., ANGLE (NVIDIA, NVIDIA GeForce RTX 3080 Direct3D11 vs_5_0 ps_5_0) vs. Apple M2 Pro vs. Mesa Intel UHD Graphics 620).
- **Hardware Parameter Limits** : Enumerating MAX_TEXTURE_SIZE, MAX_VIEWPORT_DIMS, and ALIASED_LINE_WIDTH_RANGE. Direct3D backends on Windows clamp aliased line widths strictly to 1, 1, whereas native Linux OpenGL drivers expose wide ranges such as 1, 73.
- **Shader Precision Boundaries** : gl.getShaderPrecisionFormat() extracts 18 numerical boundaries across vertex/fragment stages.

Deep GPU Forensics via Custom GLSL Shaders

Deep GPU forensics bypasses declarative state inspection by executing custom GLSL shaders to evaluate silicon-level floating-point arithmetic on the GPU's execution units:

32-bit Single Precision Float Value (IEEE 754)

[Sign: 1b | Exponent: 8b | Mantissa: 23b]

```

    |
    [ GLSL Shader Computation ]
- Transcendental Approximations (sin, cos, exp, log)
- Ill-Conditioned Matrix Inversions (catastrophic cancellation)
- Fused Multiply-Add (FMA) / Subnormal FTZ vs. Preserved

```

[RGBA8 Bit-Packing: v * vec4(1.0, 255.0, 65025.0, 16581375.0)]

[gl.readPixels Readback -> Direct Byte Array Silicon Hash]

1. **Transcendental Approximation Drift** : Functions like $\sin()$ and $\log()$ are evaluated using hardware Special Function Units (SFUs) running piecewise polynomials or CORDIC iterations. ULP variance ($\text{ulp}(x) = 2^{\lfloor \log_2(|x|) \rfloor - 23}$) compounds across iterative operations, producing distinct pixel hashes across GPU architectures.
2. **III-Conditioned Matrix Inversion** : Calculating $\det(A)$ for an ill-conditioned matrix $\kappa(A)$ forces subtractions of large, nearly identical floating-point products, causing catastrophic cancellation that exposes low-order mantissa rounding.
3. **Float Encoding & Readback** : To extract 32-bit floats without specialized extensions, GLSL fragment shaders pack scalar floats into standard 8-bit RGBA channels (RGBA8) using fractional modulus arithmetic, allowing direct host-side memory inspection. Evaluating GLSL execution unmasks software rasterizers (SwiftShader / llvmpipe), which process math via CPU vector units (AVX/SSE) and exhibit strict IEEE-754 profiles that expose headless bot infrastructure.

WebGPU Telemetry & Compute Shader Profiling

WebGPU exposes direct access to modern low-level graphics APIs (Direct3D 12, Metal, Vulkan) via browser backends (Dawn in Chromium, wgpu in Firefox).

Adapter Introspection & WGSL Compute Kernels

Querying `navigator.gpu.requestAdapter()` yields explicit hardware parameters:

- `adapter.info`: Hardware vendor, architecture, device PCI ID, and driver description.
- `adapter.limits`: Structural hardware ceilings, including `maxStorageBufferBindingSize` (128 MB mobile vs. $2 \text{ GB} + \text{desktop}$), `maxComputeWorkgroupStorageSize` (16 KB , 32 KB , or 64 KB shared memory), and `maxComputeInvocationsPerWorkgroup` (256 vs. 1024).
- `adapter.features`: Supported native capabilities (texture-compression-bc, shader-f16, subgroups). Executing parallel compute shaders written in WGSL (WebGPU Shading Language) over a storage buffer maps floating-point execution results directly back to CPU memory via `mapAsync()`:

Anti-detect tools attempting to spoof WebGPU strings are unmasked when raw WGSL storage buffer readbacks return ALU arithmetic hashes that contradict declared adapter descriptors or when native Direct3D 12 buffer alignment rules (`minUniformBufferOffsetAlignment: 256`) appear on profiles claiming to be Safari on macOS.

Section 5: Audio Subsystem & Signal Processing Telemetry

OfflineAudioContext Synthesis

`OfflineAudioContext` provides a headless audio processing graph that renders synthesized PCM audio buffers directly to memory without activating physical speakers or requiring user permissions.

```

[ OscillatorNode (Triangle Wave @ 10 kHz) ]
      |
[ BiquadFilterNode (Resonant Bandpass @ 8 kHz, Q = 12.0) ]
      |
[ GainNode (Dynamic Scaling @ 0.75) ]
      |
[ OfflineAudioContext Destination (PCM Buffer @ 44.1 kHz) ]
      |
[ AudioBuffer.getChannelData(0) -> Float32Array ]
      |
+-----+-----+
|                                     |
[ Sub-Sample Hash ]           [ Statistical Moments Extraction ]
(Murmur3 / SHA-256)         (Variance, Skewness, Kurtosis, ZCR)

```

Sources of Audio Buffer Variance

1. **Wavetable Synthesis Interpolation** : Oscillators generate non-sinusoidal shapes using band-limited Fourier series approximations. Interpolation routines (linear, cubic hermite, or spline) vary across engine codebases.
2. **Compiler Vectorization & SIMD Paths** : Audio processing loops compile to AVX2/SSE instructions on x86_64 versus ARM NEON on ARM64. Fused Multiply-Add (FMA) flags ($a \cdot b + c$) alter single-bit mantissa values, compounding across iterative render quanta (128-sample blocks).
3. **Biquad Filter Recurrence Relations** : Digital biquad filters execute Direct Form I/II difference equations:
$$y_n = \frac{b_0}{a_0}x_n + \frac{b_1}{a_0}x_{n-1} + \frac{b_2}{a_0}x_{n-2} - \frac{a_1}{a_0}y_{n-1} - \frac{a_2}{a_0}y_{n-2}$$
 High resonance ($Q \geq 10.0$) forces each output sample y_n to depend recursively on y_{n-1} and y_{n-2} , propagating floating-point rounding deltas across all subsequent samples.

Acoustic Entropy Classification

Extracting statistical moments (mean, variance, skewness, excess kurtosis) and Zero-Crossing Rates (ZCR) categorizes platform microarchitectures:

Farbling defenses (e.g., Brave's LSB noise injection) disrupt raw string hashes (pcmHash) but fail to alter global statistical moments. Farbling can be detected by evaluating second derivatives ($\Delta^2 = (y_n - y_{n-1}) - (y_{n-1} - y_{n-2})$) across adjacent samples, which reveals high-frequency differential noise that breaks natural DSP curves.

DynamicsCompressor Non-Linear Artifacts

DynamicsCompressorNode introduces time-varying non-linear dynamic range compression, evaluating logarithmic power transformations, soft-knee quadratic splines, and exponential envelope decay ballistics.

```

[ Input Sample x[n] ] ---> [ Decibel Power Converter: 20*log10(|x|
+ eps) ]
                                |
                                [ Soft-Knee Transfer Function F(x_dB) ]
                                |
                                [ Asymmetric Attack/Release Envelope ]
                                [ g_env[n] = g_env[n-1] + alpha *
(g_target
- g_env) ]
                                |
[ Output y[n] ] <----- [ Linear Gain Multiplier: 10^(g_env /
20) ]

```

Decibel Transfer Functions & Asymmetric Ballistics

1. **Soft-Knee Splines** : When input power x_{dB} crosses threshold T within knee width K , gain reduction follows a quadratic transition curve:

$$F(x_{\text{dB}}) = x_{\text{dB}} + \frac{\left(\frac{1}{R} - 1\right)\left(x_{\text{dB}} - T + \frac{K}{2}\right)^2}{2K}$$
2. **Recursive Envelope Integration** : Control gain updates via attack/release time constants τ_A, τ_R :

$$\alpha_A = 1 - e^{-\frac{1}{\tau_A} f_s \cdot \tau_{\text{attack}}}, \quad \alpha_R = 1 - e^{-\frac{1}{\tau_R} f_s \cdot \tau_{\text{release}}}$$

$$g_{\text{env}}[n] = g_{\text{env}}[n-1] + \alpha \left(g_{\text{target}}[n] - g_{\text{env}}[n-1] \right)$$

Subnormal Floating-Point Tail Behavior

When processing high-transient bursts followed by silence, trailing envelope values decay toward zero ($< 1.175494 \times 10^{-38}$):

- **x86_64 Processors** : Operate with Flush-to-Zero (FTZ) and Denormals-Are-Zero (DAZ) flags enabled in the MXCSR register, snapping trailing release samples to true zero abruptly.
- **ARM64 Processors (Apple Silicon / Cortex)** : Evaluate subnormal floating-point numbers directly in hardware with single-cycle throughput, preserving non-zero exponential decay tails across additional render blocks. Mocking DynamicsCompressorNode using a linear GainNode fails because linear gain yields static decay ratios regardless of input energy, breaking calculated decay skewness and kurtosis.

Section 6: Peripheral Topology & Media Device Enumeration

Permission Architecture & Origin-Salted Hashes

The navigator.mediaDevices.enumerateDevices() API exposes client multimedia hardware topology across two permission tiers:

```

[ navigator.mediaDevices.enumerateDevices() Call ]
      |
      +-----+-----+
      |               |
[ Unpermissioned State ]           [ Permissioned State ]
- deviceId: Anonymous / Blank      - deviceId: Stable
HMAC-SHA256 Hash
- label: "" (Sanitized String)    - label: Cleartext
("Logitech Brio 4K")
- kind: audioin/audioout/videoin  - groupId: Hardware
Composite GUID Hash
- Device Counts Exposed           - Extended Capabilities
Accessible

```

Unpermissioned Device Count Tuples

Prior to permission prompts, `enumerateDevices()` returns sanitized `MediaDeviceInfo` arrays. Although label and deviceId strings are stripped, the device count tuple $(N_{\text{audioin}}, N_{\text{audioout}}, N_{\text{videoin}})$ remains accessible. Corporate laptops typically exhibit $(1, 1, 1)$ or $(2, 2, 1)$ signatures, whereas dedicated desktop workstations present expanded topologies like $(3, 5, 2)$.

Origin-Salted Device IDs & Group ID Clustering

When permissions are granted, deviceId strings are derived via a cryptographic HMAC: $\text{deviceId} = \text{HMAC-SHA256}(\text{HardwareGUID} \parallel \text{DriverInterfacePath}, \text{OriginSalt})$. While OriginSalt prevents cross-origin tracking, the generated hash remains persistent within the origin across reboots. The groupId property links distinct media interfaces belonging to the same physical composite housing (e.g., a webcam with an integrated microphone array), forming an adjacency graph of physical hardware structures.

Capability Forensics & Virtual Device Unmasking

Calling `MediaStreamTrack.getCapabilities()` on an active stream exposes hardware driver bounds:

- **Resolution & Frame Rate Stepping** : Physical optical sensors support discrete hardware resolution steps (640×480 , 1280×720 , 1920×1080) and stepped frame rates (15.0 , 24.0 , 30.0 , 60.0 fps).
- **Optomechanical Controls** : Physical webcams expose continuous ranges for focusMode, exposureMode, whiteBalanceMode, zoom, and torch.

Automated scrapers launching Chromium with synthetic media flags

(`--use-fake-device-for-media-stream`) expose predictable indicators: static labels

(fake_device_0, MediaStream Monitor), zero frame delivery jitter, and fixed 440 Hz sine-wave audio tones, allowing identity pipelines to reject automated bot traffic.