A Delivery-chain renderer specification

We summarize the released renderer at four levels: family-level sampling policy, template inventory, fixed parameter pools, and waveform-level operator realization. Unless noted otherwise, the appendix tables are authoritative for the exact configuration used to build the benchmark package in this paper. The operator summary in the main text is intentionally scenario-level; when it is less specific than the appendix, the appendix should be read as the exact release description.

A.1 Family-level rendering logic

For each parent utterance, the renderer emits one direct control and then allocates a fixed nominal rendering budget to each non-direct family. Within a family, it samples concrete templates, instantiates the template-level operators from fixed parameter pools, and realizes the resulting waveform-domain chains. Each non-direct family attempts to contribute up to four rendered children before post-render validation. Template identifiers are deterministically shuffled per parent and sampled without replacement. For paired templates, one canonical member of the pair is instantiated first and its realized operators are then reordered to form the paired counterpart, so that the two children differ only in local order while sharing the same sampled parameters. Candidate outputs that fail the final release checks are dropped, so the released counts are slightly below the nominal family-level rendering budget.

A further detail concerns re-encoding. In same mode, the reencode operator is forced to use the most recent codec already realized earlier in the chain. In cross mode, it is forced to differ from that codec and is sampled from {AAC, Opus}. If no earlier codec exists, the renderer falls back to a family-specific default: AAC for simulated replay and Opus for the relevant hybrid templates. In addition, whenever the realized codec is GSM, the encoding sample rate is fixed to 8 kHz and decoding also proceeds from 8 kHz.

A.2 Template inventory and parameter pools

The configuration used here contains 33 templates, which realize 26 distinct ordered operator sequences because several templates intentionally keep the same operator multiset while changing only local order. Table 8 gives the exact family–template–operator inventory used in the release. In this inventory, call_path appears only in the three telephony session templates (session_nb_mulaw, session_nb_gsm, and session_wb_opus), whereas reencode appears in platform-like, simulated replay, and hybrid, but not telephony.

Table 9 summarizes only the pool ranges and export contract. The exact per-family realizations are already determined by the template inventory and the operator descriptions below.

A.3 Operator realization

All operators are realized as waveform-domain transforms rather than symbolic metadata tags, and every realization is recorded in the manifest trace.

Resampling The resampling operator uses ffmpeg with the SoX resampler at precision 28. One-way modes include 16k→8k, 16k→24k, 16k→32k, 8k→16k, 24k→16k, and 32k→16k. Round-trip modes are implemented as two successive resampling steps with an intermediate temporary waveform.

Band-limiting The band-limit operator is implemented with ffmpeg filtering. The narrowband profile applies a second-order high-pass filter at 250 Hz and a second-order low-pass filter at 3400 Hz, followed by companding with 0.01 s attack and 0.15 s decay. The wideband profile applies a second-order high-pass filter at 50 Hz and a second-order low-pass filter at 7000 Hz. To avoid terminology drift, we use narrowband (250–3400 Hz) and wideband (50–7000 Hz) consistently throughout the appendix and treat these implementation-level cutoffs as authoritative for the released benchmark.

Codec and re-encoding The codec and re-encode operators both perform true encode–decode round trips through ffmpeg. The supported codecs are AAC, Opus, GSM, μ-law PCM, and A-law PCM. After encoding to the corresponding intermediate container, the audio is decoded back to mono PCM WAV. The only difference between these two operators is how the target codec is chosen: codec follows the template specification directly, while reencode follows the context-aware same/cross rule described above.

Packet loss The packet-loss operator works on 20 ms waveform frames. Let p denote the target loss rate after clamping to at most 0.95 and let b denote the requested average burst length in frames. The underlying two-state good/bad process uses

$$P(\text{bad} \to \text{good}) = \min(1, 1/b) \tag{1}$$

and

$$P(\text{good} \to \text{bad}) = \min\left(1, \frac{p\,P(\text{bad} \to \text{good})}{1-p}\right). \tag{2}$$

When a frame enters the bad state, one of three concealment strategies is applied: repeat-fade, interpolation, or noise fill. The processed frames are then concatenated back into a waveform.

Room impulse responses The RIR operator first attempts room simulation with Pyroomacoustics. For a requested room preset and RT60, it computes an absorption coefficient by inverse Sabine, samples valid source and microphone positions, generates a room impulse response, truncates the late tail, and convolves it with the waveform. If Pyroomacoustics is unavailable or fails, the renderer falls back to a synthetic RIR generator based on a direct path, exponentially decaying random reflections, and a low-level noise floor. The reverberant output is peak-normalized after convolution.

Additive noise The noise operator adds synthetic noise at a target SNR. White noise uses i.i.d. Gaussian samples; pink noise uses $1/\sqrt{f}$ spectral shaping; brown noise is obtained by cumulative summation; hiss emphasizes high frequencies; hum combines 50, 100, and 150 Hz tonal components with low-level white noise; and babble is synthesized by summing six randomly shifted and scaled noise streams. For babble, hum, and hiss, a non-stationary amplitude envelope is applied before SNR calibration.

Call paths The call-path operator is a composite transform with a fixed internal pipeline: profile-specific band filtering and resampling to the target sample rate, codec round trip, burst packet loss with the same implementation as above, frame-wise jitter-buffer simulation, and AGC or limiting. The wideband profile uses the same 50–7000 Hz passband as the standalone band-limit operator. The narrowband profile uses the same 250 Hz and 3400 Hz cutoffs but a different compand curve. Jitter is implemented by displacing each frame by a uniformly sampled offset in $[-J, J]$, overlap-averaging the displaced frames, and trimming or padding the result back to the original length.

Table 7: Nominal family-level rendering budget in the released renderer before post-render validation.

Family	Templates	Nominal max	Sampling rule
Direct	1	1	Emit one identity control.
Platform-like	6	4	Attempt up to four samples without replacement; retain only outputs that pass post-render validation.
Telephony	10	4	Attempt up to four samples without replacement; retain only outputs that pass post-render validation.
Simulated replay	7	4	One paired group may be emitted first; otherwise sample without replacement; retain only outputs that pass post-render validation.
Hybrid	9	4	One paired group may be emitted first; otherwise sample without replacement; retain only outputs that pass post-render validation.

Table 8: Template inventory as ordered operator blueprints before parameter instantiation.

Family	Template ID	Ordered operator sequence
Direct	direct_clean	Identity chain with no operator.
Platform-like	aac_single	codec
	opus_single	codec
	aac_reencode	codec → reencode
	opus_reencode	codec → reencode
	aac_resample_reencode	codec → resample → reencode
	resample_opus	resample → codec
Telephony	nb_mulaw	bandlimit → codec
	nb_gsm	bandlimit → codec
	wb_opus	bandlimit → codec
	nb_mulaw_plr	bandlimit → codec → packet_loss
	nb_resample_mulaw_plr	resample → bandlimit → codec → packet_loss
	wb_resample_opus_plr	resample → bandlimit → codec → packet_loss
	wb_opus_resample_return	bandlimit → codec → resample
	session_nb_mulaw	call_path
	session_nb_gsm	call_path
	session_wb_opus	call_path
Simulated replay	rir_only	rir
	rir_noise	rir → noise
	noise_rir	noise → rir
	rir_reencode	rir → reencode
	reencode_rir	reencode → rir
	rir_noise_resample	rir → noise → resample
	resample_rir_reencode	resample → rir → reencode
Hybrid	opus_plr_rir	codec → packet_loss → rir
	rir_aac	rir → codec
	aac_rir	codec → rir
	bandlimit_codec_rir	bandlimit → codec → rir
	rir_bandlimit_codec	rir → bandlimit → codec
	rir_reencode_plr	rir → reencode → packet_loss
	reencode_rir_plr	reencode → rir → packet_loss
	resample_codec_rir	resample → codec → rir
	bandlimit_resample_codec	bandlimit → resample → codec

B Released metadata & protocol reconstruction

We now turn from waveform generation to how the packaged benchmark is encoded. The release contains one global metadata CSV and split-specific metadata CSV files. Each row describes one waveform that survives the final validation and export. The same packaging step also writes a dataset summary with speaker-disjointness checks, duplicate-ID and duplicate-path checks, dropped-row logs, and parent-coverage summaries. These files form the executable interface for reconstructing the benchmark protocols.

Table 10 summarizes the protocol-critical field groups. In addition to standard identifiers and provenance, the metadata records the ordered operator sequence, the order-invariant operator multiset, realized delivery parameters such as codec, bitrate, packet loss, bandwidth mode, SNR, RT60, room dimensions, source–microphone distance, and the instantiation seed.

The structural groups are derived deterministically from the metadata rather than manually enumerated. Operator-substitution groups fix the parent utterance, binary label, delivery family, and local chain context and require exactly one operator identity change at one chain position. Parameter-perturbation groups fix operator identity and order and vary exactly one parameter axis. Order-swap groups fix the realized operator multiset and differ by one

Table 9: Fixed parameter pools and export constraints in the released configuration.

Group	Released range or set	Used by
Codec bitrates	AAC {24, 32, 48}; Opus {16, 24, 32}; telephony Opus {16, 24}; re-encode {24, 32} kbps	codec families
Resample modes	Family-specific one-way and round-trip 8/16/24/32 kHz conversions	resample
Re-encode codec set	AAC or Opus, with same/cross selection	re-encode
Packet-loss settings	Loss {1, 3, 5, 10}%; burst {2, 3, 5} frames; three concealment modes	packet loss, call path
Call-path settings	Jitter {0, 8, 16} ms; AGC {mild, telephony}	call path
Noise settings	Six synthetic noise types; SNR {30, 20, 15, 10} dB	noise
RIR settings	RT60 {0.2, 0.4, 0.6, 0.8} s; distance {0.5, 1.0, 2.0, 3.0} m; three room presets	RIR
Export contract	Mono 16 kHz PCM WAV; duration checked in [1, 30] s before export	all outputs

Table 10: Protocol-critical released metadata groups.

Field group	Why it matters
Identity and paths	Links each delivered child to its clean parent and exported waveform.
Labels and provenance	Supports standard train/dev/test use, filtering, and speaker-disjoint checks.
Generator metadata	Enables provenance-aware analysis by generator family or system.
Delivery-chain metadata	Encodes the realized family, template, operator order, and realized parameters.
Matched-pair annotations	Reconstructs operator substitution, parameter perturbation, and order swap tasks.
Lineage/path annotations	Reconstructs delivery lineages and graph-distance based analyses.
Audio-format checks	Verifies that released files satisfy the final export contract.

adjacent transposition. Path/lineage groups fix the parent utterance, binary label, and delivery family, then derive a shortest-signature reference node and graph-distance annotations from the observed realized chains. The exact string form of the group identifiers is implementation-defined, but the grouping semantics are fixed by these rules.

C Matched-parent preservation validation

We validate the benchmark's matched-parent assumption: delivery rendering should alter surface acoustics without materially changing utterance content or speaker identity, so that the clean parent remains a valid reference for controlled comparison. The analysis runs from the packaged Stage-5 metadata and, by default, scans the exported test split. For each row it resolves the child waveform, its paired clean parent, and the canonical transcript, then writes row-level outputs keyed by sample_id and parent_id.

Each child–parent pair yields three preservation signals and two bookkeeping outputs. The script computes parent and child ASR hypotheses and measures transcript preservation against the canonical transcript using WER and CER, reporting ΔWER and ΔCER as child-minus-parent drift. It also embeds the parent and child waveforms with a fixed speaker backend and reports cosine similarity, while the waveform itself provides lightweight acoustic drift measures such as duration ratio, RMS change, and peak amplitude. In addition, the export records status and error fields and emits both row-level and aggregated summaries.

The text normalization and error-rate computation follow the released implementation. For English, the reference and hypothesis are lowercased, punctuation is stripped, and whitespace is collapsed before tokenization. For Chinese, the normalization keeps CJK and alphanumeric characters, and WER falls back to character-level tokenization when no whitespace segmentation is present. CER is always computed on the normalized character sequence. The row-level export retains both raw and normalized hypotheses so the preservation statistics can be recomputed exactly.

The validation backends are fixed. ASR uses the default transformers_asr backend with a Hugging Face automatic-speech-recognition pipeline instantiated from openai/whisper-large-v3-turbo, chunk length 15 s, batch size 16, generation task transcribe, and half-precision model loading. Speaker preservation uses the default transformers_speaker backend, which loads a wavlm-base-sv x-vector checkpoint through AutoFeatureExtractor and AutoModelForAudioXVector, embeds parent and child audio in batches of 16, ℓ_2-normalizes the embeddings, and computes cosine similarity. These backends are held fixed across all families and languages.

Overall, the mean child-minus-parent drift is 0.061 in WER and 0.043 in CER, with mean speaker cosine 0.941, mean duration ratio 1.056, and mean absolute RMS drift of 2.334 dB. Platform-like remains closest to the parent, telephony introduces moderate degradation, and simulated replay and hybrid induce the largest shifts. This pattern supports the benchmark design: delivery transforms are strong enough to create a meaningful robustness challenge while still preserving the matched-parent structure required for controlled evaluation.

Table 11 reports only family means because this is the clearest compact summary for the paper. The released preservation outputs additionally provide row-level records, family-language aggregates, and explicit status counts. We therefore aggregate bona fide and spoof rows together in the appendix: the preservation question concerns whether the delivery renderer preserves the parent-reference relationship, which is a property of the transformation pipeline rather than of the detection label. The row-level CSV retains the

Table 11: Matched-parent preservation statistics by delivery family. Entries are family means over successful child–parent comparisons. ΔWER and ΔCER denote child-minus-parent transcript error drift against the reference transcript; lower is better. Speaker cosine is higher-is-better. Duration ratio and |ΔRMS| summarize acoustic drift.

Family	ΔWER	ΔCER	Speaker cos.	Dur. ratio	|ΔRMS|
Direct	0.000	0.000	1.000	1.000	0.000
Platform	0.012	0.011	0.986	1.005	0.245
Telephony	0.059	0.046	0.943	1.001	1.620
SimReplay	0.094	0.063	0.909	1.120	4.247
Hybrid	0.095	0.065	0.910	1.114	3.842

original label field, so median, percentile, or label-conditioned summaries can be reproduced directly from the released outputs without rerendering the dataset.

D Benchmark-specific robustness metrics

Here we define only the new robustness metrics introduced in this work. Standard detection metrics such as EER, AUC, accuracy, and F_1 are omitted for brevity.

D.1 Notation and thresholding

Let each evaluation sample have score $s_i \in \mathbb{R}$ and binary label $y_i \in \{0, 1\}$, where $y_i = 1$ denotes bona fide speech and $y_i = 0$ denotes spoofed speech. Larger scores indicate stronger bona fide evidence. For each benchmark task, let $\mathcal{E}$ denote the full scored evaluation split used for method comparison (in our experiments, the pooled test split for that task), with score set $S_{\mathcal{E}}$ and labels $Y_{\mathcal{E}}$. We define a single reference operating point once per task by

$$\tau_{\text{ref}} = \begin{cases} \tau_{\text{EER}}(S_{\mathcal{E}}, Y_{\mathcal{E}}), & \text{if both classes are present in } \mathcal{E}, \\ 0.5, & \text{otherwise.} \end{cases} \tag{3}$$

The matched-pair and lineage metrics then convert scores to hard decisions using this same fixed threshold:

$$\hat{y}_i = \mathbb{I}[s_i \geq \tau_{\text{ref}}]. \tag{4}$$

Two implementation choices matter. First, τ_{ref} is estimated once from the full evaluation split $\mathcal{E}$ and then reused for all matched-pair sets, delivery families, and lineage analyses within that task; it is not recomputed for individual subsets. Second, the hard-decision robustness metrics are intended as standardized analysis summaries at a common evaluation-defined operating point, not as a claim of deployment-ready threshold calibration. This avoids subgroup-specific retuning and anchors all robustness comparisons to the same decision rule.

Accordingly, the threshold-free metrics (notably EER/AUC in the main experiments and MNSD below) should be read as the primary calibration-independent summaries, while PCR, PJA, C-FFD, R_k, and AURC-chain are decision-level diagnostics at the common reference operating point τ_{ref}. τ_{EER} thus standardizes the binary robustness analyses rather than optimizing each protocol independently.

D.2 Matched-pair metrics

For operator substitution, parameter perturbation, and order swap, evaluation is carried out on the set of valid matched pairs

$$\mathcal{P} = \{(i_m, j_m)\}_{m=1}^{M}, \tag{5}$$

where the two members of each pair share the same ground-truth label and differ only in the targeted local intervention.

Pair consistency rate The pair consistency rate is defined as

$$\text{PCR} = \frac{1}{M} \sum_{m=1}^{M} \mathbb{I}[\hat{y}_{i_m} = \hat{y}_{j_m}], \tag{6}$$

and measures whether the detector preserves its binary decision under the controlled local edit, regardless of whether that decision is correct.

Pair joint accuracy To separate stable-and-correct behavior from stable-but-wrong behavior, the pair joint accuracy is defined by

$$\text{PJA} = \frac{1}{M} \sum_{m=1}^{M} \mathbb{I}[\hat{y}_{i_m} = y_{i_m} \ \wedge \ \hat{y}_{j_m} = y_{j_m}], \tag{7}$$

which requires both members of the pair to be classified correctly.

Mean normalized score drift Because a detector can keep the same hard decision while still showing substantial score-level sensitivity, we also report the mean normalized score drift. Let

$$S = \{s_1, \ldots, s_N\}, \qquad \text{IQR}(S) = Q_{0.75}(S) - Q_{0.25}(S). \tag{8}$$

Define the normalization denominator by

$$D(S) = \begin{cases} \text{IQR}(S), & \text{if IQR}(S) > 10^{-12}, \\ 1, & \text{otherwise.} \end{cases} \tag{9}$$

Then

$$\text{MNSD} = \frac{1}{D(S)} \cdot \frac{1}{M} \sum_{m=1}^{M} |s_{i_m} - s_{j_m}|. \tag{10}$$

This matches the scorer implementation exactly: when the score IQR is numerically zero or extremely small, the denominator is set to 1 rather than to a tiny positive constant.

Symmetric misclassification risk The scorer also records an auxiliary diagnostic called symmetric misclassification risk. Define the sample-level 0–1 loss as

$$\ell_i = \mathbb{I}[\hat{y}_i \neq y_i]. \tag{11}$$

The corresponding pairwise quantity is

$$\text{SMR} = \frac{1}{M} \sum_{m=1}^{M} \frac{\ell_{i_m} + \ell_{j_m}}{2}. \tag{12}$$

SMR summarizes the average misclassification burden within a matched pair and is mainly useful as a diagnostic complement to PCR, PJA, and MNSD.

D.3 Delivery-lineage metrics

A delivery lineage groups all rendered samples that share the same parent utterance, delivery family, and binary label. Each node v in lineage g carries an observed operator-signature sequence $\sigma(v)$. The reference node r_g is the shortest observed signature realization in that lineage.

Lineage graph and node correctness Distances are defined on a lineage graph rather than directly on template identifiers. The

Table 12: Compact summary of the benchmark-specific metrics.

Setting	Metric	What it captures	Better
Matched pairs	PCR	Decision stability	Higher
Matched pairs	PJA	Stable correctness	Higher
Matched pairs	MNSD	Score sensitivity	Lower
Matched pairs	SMR (aux.)	Pairwise error risk	Lower
Lineages	C-FFD	First-failure distance	Higher
Lineages	R_k	Correctness up to distance k	Higher
Lineages	AURC-chain	Overall lineage robustness	Higher

graph nodes are observed operator-signature realizations, and an undirected edge is inserted whenever two realizations differ by exactly one atomic delivery edit: a single parameter perturbation, a single operator substitution, an adjacent order swap, or a single insertion/deletion. Let the shortest-path distance from r_g to node v be $d_g(v)$. Using the same fixed reference threshold from Eq. (3), node correctness is defined by

$$c(v) = \mathbb{I}[\hat{y}(v) = y(v)]. \tag{13}$$

Censored first-failure distance For lineage g, the first-failure distance is

$$\rho_g = \min\{d_g(v) : c(v) = 0\}, \tag{14}$$

whenever at least one misclassified node is observed in that lineage. Some lineages may have no observed failure at all and are therefore right-censored.

Let $D_{\max} = \max_{g\in\mathcal{G}} \max_{v\in g} d_g(v)$ be the largest observed lineage distance in the evaluation set. The censored first-failure distance is then

$$\tilde{\rho}_g = \begin{cases} \rho_g, & \text{if lineage } g \text{ contains} \geq 1 \text{ observed failure,} \\ D_{\max} + 1, & \text{otherwise,} \end{cases} \tag{15}$$

and the reported aggregate is

$$\text{C-FFD} = \frac{1}{|\mathcal{G}|} \sum_{g\in\mathcal{G}} \tilde{\rho}_g. \tag{16}$$

C-FFD increases when correct behavior persists farther from the reference node along the lineage graph.

Robust-at-distance To summarize robustness at progressively larger edit distances, we define

$$R_k = \frac{1}{|\mathcal{G}|} \sum_{g\in\mathcal{G}} \mathbb{I}\big[\forall v \in g,\ d_g(v) \leq k \Rightarrow c(v) = 1\big], \quad k = 0, 1, \ldots, D_{\max}. \tag{17}$$

Hence, R_k is the fraction of lineages that remain entirely correct up to and including graph distance k.

AURC-chain Finally, the area under the robustness curve is defined by

$$\text{AURC-chain} = \frac{1}{D_{\max} + 1} \sum_{k=0}^{D_{\max}} R_k. \tag{18}$$

AURC-chain compresses the full robustness profile into a single scalar and increases when correctness is maintained more consistently under progressively larger delivery edits.