Can Bits Seal Language?

Jia Xu
8AI LLC
xu@8ai.llc

Abstract

We present Bits, an encoder that maps each input token to a random sequence of binary values. By working at the finest representational granularity and compressing the result with one small shared network, Bits reduces the input vocabulary to two symbols, 0 and 1, and removes every input parameter indexed by token identity. We evaluate it on 32 tasks in six modalities: human text, genomic and clinical data, code, system logs, EEG signals, and tabular records. It resists six gradient-inversion attacks better than six encoder baselines and four defenses, with TAG recovering up to $167\times$ fewer tokens. Without any defense, it leaks less under TAG, LAMP, and Inverting Gradients than a subword table protected by DP-SGD, gradient compression, InstaHide, or federated DP. It cuts embedding parameters by up to 99% at competitive accuracy and with negligible computational overhead, and transfers across datasets without re-initialization. Trained from scratch at the Qwen2.5-7B scale, it reaches lower held-out perplexity than an embedding table at every checkpoint of every seed, with a $277\times$ smaller embedding. Code and checkpoints: `https://github.com/bit2vec/bitsseal`, `pip install bitsseal`.

1 Introduction

Modern neural networks encode inputs such as text, genomes, code, signals, and tabular records as dense token embeddings before any learning takes place. Existing encoders [52, 59] allocate these parameters per token, one embedding row per vocabulary entry, and that row is where three costs meet. It leaks: only the rows of the tokens in a batch receive a nonzero gradient, so whoever sees the gradient sees which tokens were there, and FILM, TAG, and DLG [28, 16, 77] read exactly this. It grows with the vocabulary size V. And it belongs to one vocabulary, so a new domain or modality starts from untrained rows.

Byte-based encoders share parameters across tokens and remove the per-token row, but they still read an alphabet of 256 symbols, each with its own row. A token then becomes either a byte sequence several times longer than the subword sequence [68, 70], which costs compute, since the Transformer's cost grows with the square of the sequence length, or a $2{,}048$-entry one-hot input [40], which costs memory. An alphabet of 256 is still far larger than it needs to be.

We go to two. Bits gives each token a fixed random code of $n{=}64$ bits and learns one small network over bits. The embedding table is gone, and the input vocabulary is $\{0, 1\}$. Figure 1a follows a sentence through the encoder: each subword is mapped to its n-bit code, drawn once and never trained (step 3), each bit is looked up in a two-row table and the n results are aggregated into one vector (step 4), and a two-layer network shared by every token compresses that vector into the model's d-dimensional embedding (steps 5 and 6), which the unchanged Transformer body consumes. Only the shared network is trained, none of its parameters belongs to a token, and a training step sums the gradients of every token in the batch into one update. Figure 1 places the encoder in federated training, where privacy matters most: clients encode locally and upload only gradients, the server sums them and broadcasts the weights (Figure 1b), and an attacker on the server side sees weights and gradients but no text (Figure 1c). Three properties follow.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

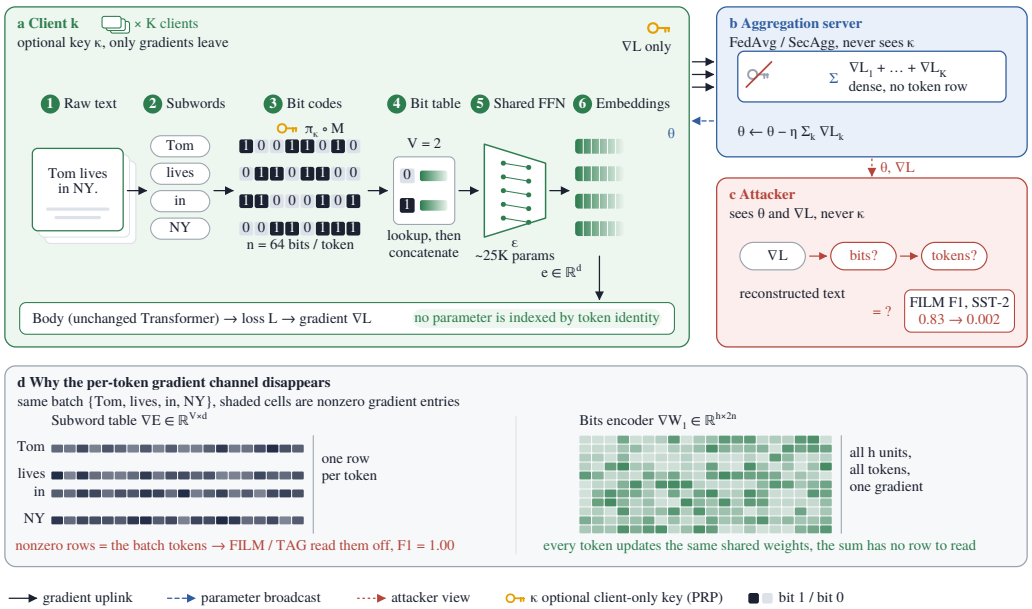

Figure 1: **Bits-encoder pipeline in a federated setting.** (a) A client tokenizes text into subwords, maps each to a 64-bit code, permuted under the client key κ in the keyed variant, looks the bits up in a two-row table, and projects the concatenation through the shared FFN of about 25K parameters. The Transformer body is unchanged and only gradients leave the client. (b) The aggregation server sums dense gradients and never sees κ when one is used. (c) An attacker who sees the weights θ and the gradient $\nabla\mathcal{L}$ recovers no token row, and FILM F1 on SST-2 falls from 0.83 to 0.002. (d) Why the per-token channel disappears: the subword gradient has one nonzero row per batch token, which FILM and TAG read off, while every token of the batch updates the same shared weights of the Bits encoder.

Privacy. No input parameter belongs to a token. Every token passes through the same weights, so the gradient of a batch is a sum of updates to shared parameters, and no row marks which tokens were present (Theorem 1). What an attacker can still extract is not zero, and we measure it: short inputs can be recovered, a single token exactly, and recovery falls toward chance as more tokens share one gradient.

To see why, think of a blender and a bill. The shared network of Figure 1a is the blender: once many fruits are blended into one juice, no single fruit can be recovered, even with the recipe in hand. Figure 1d shows the gradient one batch leaves behind. A subword table leaves one nonzero row per batch token, four tokens, four rows, and FILM and TAG simply read them off. Bits spreads the same four tokens across the same shared weights, and there is no row to read. The batch sum mixes further, and federated aggregation mixes again (Figure 1b). That is the bill: everyone sees the total, no one sees a single order, and the more people share it, the harder any one order is to infer, which is exactly how token recovery falls as sequence length grows. All of this holds against a white-box attacker who knows the codebook, the seed, the architecture, and the weights, the view in Figure 1c, in keeping with Kerckhoffs's principle. The privacy is in the shared representation, not in obscurity.

The key is optional and off by default. Every theorem and experiment uses the unkeyed encoder, except Theorem 2 and the keyed row of Table 5. A client may additionally permute its codes under a secret key κ that the server never sees. This upgrades the guarantee from structural, no gradient row names a token, to cryptographic, recovering even a single token means breaking the cipher.

Size. The encoder has about 25K parameters at d=64 whatever V is, whereas a table has V rows of d values, Vd in all (Theorem 3). The encoder loses nothing in principle (Theorem 4). Sequence length, training time, and inference time are unchanged.

Modality. Nothing in the encoder depends on what a token means or how many there are, so the same architecture reads English subwords, DNA k-mers, C functions, system-log fields, quantized EEG, and serialized clinical records, and the privacy property travels with it.

Experiments confirm all three on 32 tasks in six modalities, with the same model body in every comparison so that only the embedding layer differs (Figure 2). **Privacy.** Under six gradient-inversion

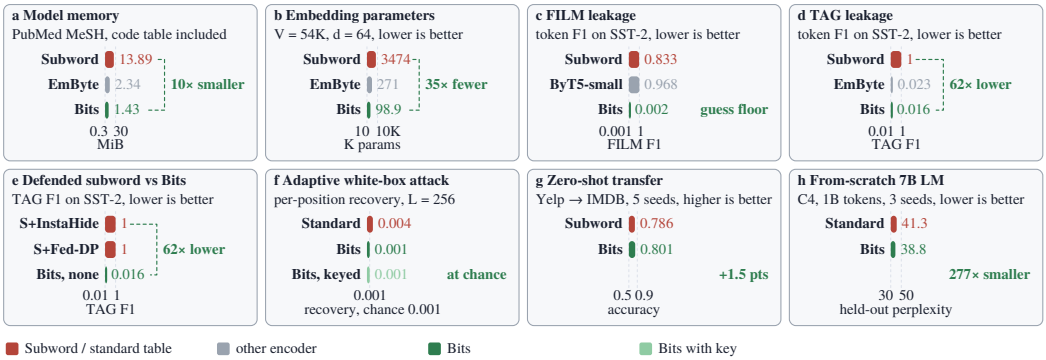

Figure 2: **Headline results.** Same body and recipe in each panel, only the embedding layer differs, and (a) to (f) use log axes. (a) Model memory, code table included, and (b) embedding parameters at $V \approx 54$K (Table 3). (c) FILM and (d) TAG token recovery on SST-2 against the subword table and byte-level encoders (Table 2), where Bits FILM is a guessing floor. (e) TAG on SST-2 for Subword under InstaHide and federated DP vs Bits with none (Table 2). (f) Adaptive white-box attack at L=256 (Table 5). (g) Zero-shot Yelp to IMDB (Table 3). (h) Held-out perplexity of Qwen2.5-7B trained from scratch, three seeds (Table 6).

attacks, Bits cuts TAG token recovery by up to $167\times$ relative to the subword table. FILM recovers the subword table's tokens with F1 up to 1.00, but Bits has no per-token rows for it to read, so it can only guess, about 0.002, the chance level for a 13K vocabulary. With no defense at all, Bits leaks less under TAG, LAMP, and Inverting Gradients than a table protected by DP-SGD, gradient compression, InstaHide, or federated DP. A white-box attacker who knows the codes reaches chance by 256 tokens. Against the table it does not, even at $1{,}024$. **Size.** The embedding layer shrinks by up to 99% and the whole model by up to $14\times$ in parameters and $10\times$ in memory at $V \approx 54$K, at competitive accuracy and with a training step within 0.1 ms of the table's. **Modality and scale.** The TAG gain holds on every text, genomic, code, log, EEG, and tabular task. The encoder transfers zero-shot from Yelp to IMDB with a 1.5-point gain, and, trained from scratch in the Qwen2.5-7B architecture, it reaches lower held-out perplexity than the table at every checkpoint of every seed, with a $277\times$ smaller embedding.

Our contributions are: **(1) A two-symbol encoder with no token-indexed input parameters.** Removing the embedding table removes the per-token gradient row that most gradient-inversion attacks read. **(2) Guarantees for what remains.** Privacy has a structural floor without a key and a cryptographic one with it, memory stays constant in V, and expressivity is not lost in principle (Theorems 1 to 4). **(3) An evaluation from the attacker's side**, from 25K-parameter classifiers to a 7B language model. Attacks run at batch size one, with single and aggregated gradients, with side information, and under six token-to-code maps.

2 Related Work

Tokenization and memory-efficient embeddings. BPE [52] and SentencePiece [38] carry a $V \times d$ table. Byte and character models [12, 68, 46, 70, 63, 54, 48, 35, 73] drop the vocabulary but lengthen the sequence, EmByte [40] keeps the subword length by aggregating the bytes of each subword, and vocabulary scaling goes the other way [58, 30]. Hashing [65, 13, 15], factorization, and quantization shrink the table but keep a per-token row. Hash embeddings [57], discrete codes [10], and deep hash embeddings [36] replace the table with a pool and a code per token, and the pool is still a table of learned rows that a batch touches sparsely. Our SpinHash [67] and random token vectors redrawn in every sequence [31] drop the learned row entirely, but neither studies privacy. Bits keeps our binary encoder and asks what its gradient leaks: a proof (Theorem 1), a measurement under six attacks, and a keyed permutation under which recovery reduces to the cipher (Theorem 2). **Gradient inversion.** FILM [28] and Decepticons [20] read the batch off the nonzero embedding rows. TAG [16], DLG and iDLG [77, 75], Inverting Gradients [25], LAMP [2], DAGER [51], and SPEAR [17] invert the full gradient and project onto those rows. Robbing the Fed [21], Fishing [66], Panning for Gold [11], SEER [24], and TabLeak [61] plant malicious parameters so that even aggregated batches invert, and FL-LLG [62] reads labels from the last layer. All but FL-LLG need a token-indexed input parameter, which Bits removes. Output-side attacks [7, 23, 45] read losses or embeddings and are orthogonal.

Privacy defenses. DP-SGD [1, 41, 69], federated DP [60, 64], PATE [50], and SecAgg [6] bound or hide what the body gradient reveals, and all compose with Bits (§5). Gradient compression [42] keeps the row. InstaHide [33, 32] has been broken [8], heuristic defenses fall to a Bayesian attacker [3], and parameter-efficient tuning lowers recovery without removing the table [74]. All of these act on the body or on the aggregation, while Bits alone acts on the embedding gradient itself. **Multilingual transfer.** Joint vocabularies [18, 47] keep per-row leakage and table cost. Byte models [12, 68] transfer across languages, and tokenizer transfer predicts rows for a new vocabulary [44]. Bits has no row to relearn.

3 Method

The Bits encoder reduces the model's input vocabulary to its minimum, $\mathcal{V}_b = \{0, 1\}$, while keeping the representational capacity of a standard subword Transformer. A standard subword tokenizer, for example BPE [52], splits the input text into m subwords $(w_1, \ldots, w_m)$ drawn from a vocabulary $\mathcal{V}_w$ of size V. Each subword is mapped to a fixed-length binary code, embedded bit by bit, and projected to the Transformer's hidden dimension d by a small feedforward network. Bits is thus a reparameterization of the embedding layer, with the table's job moved into one shared network. Appendix A lists all notation.

Stochastic binary mapping. We define

$$\mathcal{M} : \mathcal{V}_w \to \{0, 1\}^n, \qquad \mathcal{M}(w_i) = \mathbf{b}_i = (b_{i1}, \ldots, b_{in}), \tag{1}$$

so every subword w_i receives a code $\mathbf{b}_i$ of n bits, with $n = 64$ in the main experiments and swept in §5. The codes are drawn once at setup, each bit a fair coin flip, $b_{ij} \sim \mathrm{Bernoulli}(0.5)$, and any code that repeats an earlier one is redrawn (Algorithm 1). Two subwords collide with probability $2^{-n} \approx 5.4 \times 10^{-20}$, so for a vocabulary of 10^8 entries fewer than 0.001 redraws are expected, and after redrawing no two subwords share a code. The map is then fixed and never trained.

Bit-level embedding. Each bit b_{ij} is one-hot encoded as a two-entry vector $\mathbf{o}_{ij} = (1 - b_{ij},\, b_{ij})^\top$, $[1, 0]^\top$ for a 0 and $[0, 1]^\top$ for a 1, which selects one column of the first-layer weight matrix defined below, and the n one-hots are concatenated into the encoder input $\mathbf{c}_i$ of subword w_i, a vector of $2n$ entries, n ones and n zeros:

$$\mathbf{c}_i = [\mathbf{o}_{i1}; \mathbf{o}_{i2}; \ldots; \mathbf{o}_{in}] \in \mathbb{R}^{2n}. \tag{2}$$

Neural compression projector. A two-layer feedforward network with weights $W_1 \in \mathbb{R}^{h \times 2n}$ and $W_2 \in \mathbb{R}^{d \times h}$, biases $\mathbf{u}_1 \in \mathbb{R}^h$ and $\mathbf{u}_2 \in \mathbb{R}^d$, and hidden width $h = 128$ maps $\mathbf{c}_i$ to the embedding $\mathbf{e}_i \in \mathbb{R}^d$ that the Transformer body consumes:

$$\mathbf{e}_i = W_2\, \sigma\big(W_1 \mathbf{c}_i + \mathbf{u}_1\big) + \mathbf{u}_2, \qquad \sigma = \mathrm{ReLU}. \tag{3}$$

At $n = 64$ and $d = 64$ this is $128{\cdot}128 + 128 + 128{\cdot}64 + 64 = 24{,}768$ parameters. In general the encoder has $h(2n + d + 1) + d$ parameters, independent of V, against Vd for a subword table. The body still sees one vector per subword, so a sentence of m subwords occupies m positions rather than the mn positions of a bit-by-bit sequence, and per-layer time stays $\mathcal{O}(m^2 d)$, the same as Subword. Byte-level models read about $c \approx 5$ bytes per subword [53], so their sequence is c times longer, a layer costs c^2 times more, and activations take c times more memory. Table 1 compares Bits with Subword [52], the byte-level encoders CHARFORMER [59] and ByT5 [68], and EmByte [40]. Token identity enters only through which n columns of W_1 the code selects, and token semantics are reconstructed from their sum in the hidden layer. Algorithm 2 gives the forward pass, with dropout p=0.1 on the hidden layer during training.

4 Theory

Replacing the per-token embedding row with a shared bit encoder gives three guarantees at once: a structural privacy floor, a cryptographic upgrade when a key is available, and memory and expressivity bounds. After each theorem we say what it does and does not promise. Proofs are in Appendix C.

Algorithm 1: Binary Code Construction

Input: Vocabulary $\mathcal{V}_w$, code length n, max retries T=1000
Output: Mapping $\mathcal{M} : \mathcal{V}_w \to \{0,1\}^n$ of Eq. 1

$\mathcal{M} \leftarrow \{\}$; $\mathcal{S} \leftarrow \emptyset$; // $\mathcal{S}$ holds the codes assigned so far
foreach $w \in \mathcal{V}_w$ **do**
  $\mathbf{b} \leftarrow (b_1, \ldots, b_n),\ b_j \sim \text{Bernoulli}(0.5)$;
  $r \leftarrow 0$;
  while $\mathbf{b} \in \mathcal{S}$ ***and*** $r < T$ **do**
    $\mathbf{b} \leftarrow (b_1, \ldots, b_n),\ b_j \sim \text{Bernoulli}(0.5)$;
    $r \leftarrow r + 1$;
  $\mathcal{M}[w] \leftarrow \mathbf{b}$; $\mathcal{S} \leftarrow \mathcal{S} \cup \{\mathbf{b}\}$;
return $\mathcal{M}$;

Algorithm 2: Bits Forward Pass (per batch)

Input: Subwords $(w_1, \ldots, w_m)$, map $\mathcal{M}$, weights of Eq. 3
Output: Embeddings $(\mathbf{e}_1, \ldots, \mathbf{e}_m)$ for the body

foreach *subword* $i \in \{1, \ldots, m\}$ **do**
  $\mathbf{b}_i \leftarrow \mathcal{M}[w_i]$; // code of w_i, Eq. 1
  $\mathbf{o}_{ij} \leftarrow (1 - b_{ij},\ b_{ij})$ for $j = 1, \ldots, n$;
  $\mathbf{c}_i \leftarrow [\mathbf{o}_{i1}; \ldots; \mathbf{o}_{in}] \in \mathbb{R}^{2n}$; // Eq. 2
  $\mathbf{z} \leftarrow \text{Dropout}_{0.1}(\text{ReLU}(W_1\mathbf{c}_i + \mathbf{u}_1))$; // hidden
  $\mathbf{e}_i \leftarrow W_2\mathbf{z} + \mathbf{u}_2$; // Eq. 3
return $(\mathbf{e}_1, \ldots, \mathbf{e}_m)$;

Embedding	**Params**	**Seq. len**	**Time per layer**	**Per-token grad.**	**Params at** V=35K	**Size** MB
Subword	$V \cdot d$	m	$m^2 d$	sparse: 1 row	2.24M	8.96
Byte	$V_b \cdot d$	cm	$c^2 m^2 d$	sparse: 1 row/byte	16.4K	0.07
EmByte	$h(2048 + d + 1) + d$	m	$m^2 d$	dense	271K	1.08
Bits	$h(2n + d + 1) + d$	m	$m^2 d$	dense	**25K**	**0.10**

Table 1: **Memory, time, gradient, and parameter complexity** of four embedding paradigms. Byte includes CHARFORMER and ByT5. V is the subword vocabulary size, V_b=256 the byte alphabet, d the embedding dimension, n the bits per token, h the FFN hidden width, m the number of subwords in one input, and c the average number of bytes per subword. EmByte reads up to 8 bytes per subword as $8 \times 256 = 2{,}048$ one-hot inputs. The per-token gradient column says which embedding parameters one token touches: a subword or byte table updates only that token's row, while EmByte and Bits update the whole shared FFN. The last two columns are evaluated at V=35,000, V_b=256, d=64, n=64, and h=128, in fp32.

Setup. From §3: V is the vocabulary size, d the embedding dimension, n=64 the code length, $b : \{1, \ldots, V\} \to \{0,1\}^n$ the public token-to-bit map with $b(t) = \mathcal{M}(w_t)$, and $\epsilon : \{0,1\}^n \to \mathbb{R}^d$ the Bits encoder with parameters $\theta_\epsilon = (W_1, W_2, \mathbf{u}_1, \mathbf{u}_2)$. θ_{body} denotes the body parameters. A batch is a set $\mathcal{B} \subseteq \{1, \ldots, V\}$ of $|\mathcal{B}|$ tokens, ℓ is the per-token loss, and $\mathcal{L}(\mathcal{B}) = \sum_{t \in \mathcal{B}} \ell(t)$ the batch loss, so the encoder gradient is $g = \sum_{t \in \mathcal{B}} \nabla_{\theta_\epsilon} \ell(\epsilon(b(t)))$. The subword baseline $E \in \mathbb{R}^{V \times d}$ has $\nabla_E \mathcal{L}(\mathcal{B})$ supported exactly on the rows in $\mathcal{B}$. H and I denote entropy and mutual information, in bits unless a bound says nats.

Threat model. Following Figure 3, an adversary $\mathcal{A}$ sees $(\theta_\epsilon, \theta_{\text{body}})$, the gradient g, the public map b, and V, and tries to recover $\mathcal{B}$. Three roles hold this view: an honest-but-curious server, a network eavesdropper, and a post-hoc forensic analyst. In the *unkeyed* deployment nothing is secret and Theorem 1 applies. In the *keyed* deployment the map is composed with a cipher under a secret key κ held only by the data owners. None of the three roles has it, and Theorem 2 applies.

Cipher. Let $\Pi = \{\pi_\kappa\}_{\kappa \in \{0,1\}^\lambda}$ be a family of pseudorandom permutations on $\{0,1\}^n$ with a key of λ bits, instantiated by a 64-bit block cipher or a format-preserving cipher [14, 4, 5]. The keyed encoder is $\epsilon_\kappa(t) = \epsilon(\pi_\kappa(b(t)))$, with κ used identically at training and inference and held in an HSM, a TEE, or the FL client cohort. The token-recovery indistinguishability game $\mathsf{TR\text{-}IND}^{\Pi}_{\mathcal{A}}(\lambda)$ is standard: $\mathcal{A}$ submits two disjoint batches $\mathcal{B}_0, \mathcal{B}_1$ of equal shape, labels, and token-repetition pattern, the challenger, which sampled κ before training and trained the model under π_κ, returns the gradient of one chosen at random, and $\mathcal{A}$ guesses which.

Theorem 1 (Privacy floor)**.** *Without the key, the encoder gradient g satisfies:* ***(a)*** No per-token row. $g = \sum_{t \in \mathcal{B}} \nabla_{\theta_\epsilon} \ell(\epsilon(b(t)))$ *is a sum over the batch of terms that share the same parameters: no parameter is indexed by a single token, and the column support of $\nabla_{W_1} \mathcal{L}$ is the bitwise OR of the one-hot batch codes. By contrast,* $\text{supp}(\nabla_E \mathcal{L}) = \mathcal{B}$. ***(b)*** Residual entropy. *If $\mathcal{B}$ is drawn uniformly, then for any unkeyed predictor $\hat{t}$*

$$H(\mathcal{B} \mid g) \ \geq\ \log_2 \binom{V}{|\mathcal{B}|} - I(\mathcal{B}; g), \qquad \Pr[\hat{t}(g) \in \mathcal{B}] \ \leq\ |\mathcal{B}|/V + \sqrt{I(\mathcal{B}; g)/2},\ \textit{I in nats,}$$

versus $H = 0$ and $\Pr = 1$ *for subword.*

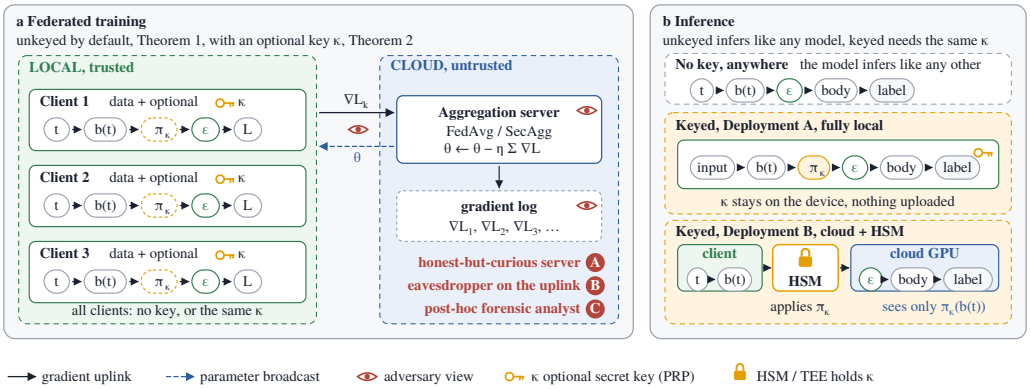

Figure 3: **Threat model.** (a) Federated training, unkeyed by default (Theorem 1): clients upload only gradients, and the server, the eavesdropper on the uplink, and the post-hoc analyst see gradients and public parameters. With the optional key κ (Theorem 2), dashed π_κ, all clients share κ and no adversary holds it. (b) Inference: an unkeyed model infers like any other, anywhere. A keyed model needs the same κ, held in one of two ways. Deployment A keeps κ on the device and runs everything locally. Deployment B keeps κ in an HSM or TEE that applies π_κ, so the cloud GPU sees only the permuted code $\pi_\kappa(b(t))$, never the token or the key.

Part (a) needs no assumption and no key: each token is spread across n bits that it shares with about half the vocabulary, and the shared FFN sums every token into one gradient, so recovering any of them means inverting a mixture that grows more underdetermined with each token added.

Part (b) is conditional entropy plus Pinsker's inequality. Its content is $I(\mathcal{B}; g)$, which we bound empirically through recovery (Table 5). Before seeing g, a batch of 32 tokens from $V{=}50{,}000$ carries $\log_2 \binom{50{,}000}{32} \approx 382$ bits of uncertainty, and a blind guess hits a batch token with probability $|\mathcal{B}|/V = 6.4{\times}10^{-4}$. Seeing g removes at most I of those bits and lifts the hit rate by at most $\sqrt{I/2}$, so a small I leaves the attacker near a guess. A subword table gives away all 382 bits and a hit rate of 1. Unkeyed Bits has no closed form for $I(\mathcal{B}; g)$: the codes are public, so an attacker can push candidate tokens through the encoder and compare gradients. Without the key we claim only that no gradient row names a token and that what remains recoverable falls as more tokens share one gradient. Table 5 measures this: short inputs are recovered, a single token exactly, and recovery reaches chance by 256 tokens. With the key, Theorem 2 takes over.

Theorem 2 (Cipher upgrade)**.** *For every PPT $\mathcal{A}$ in the single-challenge game $\mathsf{TR\text{-}IND}^{\Pi}_{\mathcal{A}}(\lambda)$, there is a PPT distinguisher $\mathcal{D}$ with $\mathsf{Adv}^{\mathsf{TR\text{-}IND}}_{\mathcal{A}}(\lambda) \le \mathsf{Adv}^{\mathsf{PRP}}_{\mathcal{D}}(\lambda)$.*

In words, an attacker who submits two batches and receives the keyed gradient of one cannot tell which, unless it can tell the cipher from a random shuffle, about 2^{128} operations for a 128-bit key. Three limits apply. The guarantee is computational: unbounded time exhausts the key, so we make no claim about $I(\mathcal{B}; g)$ for the keyed encoder. It covers one challenge on the encoder gradient: a fixed key maps a token to the same code every time, so repeated queries reveal which codes recur, which is why the game fixes the repetition pattern (Appendix D.1). Output-side attacks such as membership inference, property inference, and model inversion read predictions and losses and bypass π_κ entirely, so we close them with DP-SGD on the body. Key handling is in Appendix D.2.

Theorem 3 (Memory efficiency)**.** *$|\theta_\epsilon|$ is constant in V. The embedding compression ratio $R(V) = Vd/|\theta_\epsilon| = \Theta(V)$.*

At $d{=}64$, $|\theta_\epsilon| = 24{,}768$ whatever V is, while a subword table grows with V: $90\times$ larger at $V{=}35{,}000$, $140\times$ at $V{=}54{,}000$, and a measured $277\times$ for Qwen2.5-7B with $V{=}152$K and $d{=}3{,}584$ (Appendix F.4).

Theorem 4 (Expressivity equivalence)**.** *For any subword table $E \in \mathbb{R}^{V\times d}$, there is a two-layer ReLU encoder ϵ of hidden width $h \ge V$ with $\|\epsilon(b(t)) - E[t]\|_\infty \le \tau$ for every t and every $\tau > 0$.*

This is a worst-case existence result, not a width requirement. Real tables are far from arbitrary, and the width that works is far below the bound (Table 4).

Task	Baseline / Defense / Encoder	FILM		TAG		LLG		LAMP		Inv. Grad.		iDLG	
		S	B	S	B	S	B	S	B	S	B	S	B
MedMCQA	S	1.000	**0.011**	1.000	**0.029**	0.312	**0.295**	0.173	**0.153**	0.056	**0.044**	0.045	**0.027**
PubMedQA	S	1.000	**0.028**	1.000	**0.108**	**0.369**	0.449	0.051	**0.049**	0.145	**0.140**	0.118	**0.050**
Yelp†	S	0.549	**0.001**	1.000	**0.024**	0.461	**0.499**	0.065	**0.064**	0.043	**0.037**	0.043	**0.036**
Defenses	S	0.833	**0.002**	1.000	**0.016**	0.457	**0.489**	0.300	**0.241**	0.017	**0.013**	0.013	**0.011**
	S+DP-SGD $\varepsilon{=}\infty$	0.168	**0.015**	1.000	**0.100**	**0.497**	0.509	0.311	**0.230**	0.166	**0.110**	0.162	**0.116**
	S+DP-SGD $\varepsilon{=}16$	0.434	**0.015**	1.000	**0.101**	0.528	**0.503**	0.311	**0.230**	0.152	**0.144**	0.162	**0.155**
	S+DP-SGD $\varepsilon{=}8$	0.398	**0.015**	1.000	**0.094**	0.528	**0.503**	0.311	**0.230**	0.154	**0.149**	0.162	**0.155**
	S+DP-SGD $\varepsilon{=}4$	0.297	**0.015**	1.000	**0.098**	0.528	**0.503**	0.311	**0.230**	0.154	**0.147**	0.161	**0.154**
	S+DP-SGD $\varepsilon{=}1$	0.282	**0.015**	1.000	**0.102**	0.528	**0.503**	0.311	**0.230**	0.153	**0.146**	0.161	**0.154**
	S+Top-K no def. $K{=}1.0$	0.532	**0.000**	1.000	**0.014**	0.516	**0.494**	0.300	**0.241**	0.025	**0.018**	0.024	**0.022**
	S+Top-K 10% $K{=}0.1$	0.538	**0.000**	1.000	**0.016**	0.516	**0.494**	0.300	**0.241**	0.023	**0.020**	**0.019**	0.022
	S+Top-K 1% $K{=}0.01$	0.553	**0.001**	1.000	**0.010**	0.516	**0.494**	0.300	**0.241**	**0.020**	0.021	0.020	**0.018**
	S+InstaHide $K{=}2$	1.000	**0.002**	1.000	**0.015**	**0.534**	0.541	0.311	**0.230**	0.017	**0.013**	0.011	**0.010**
	S+InstaHide $K{=}4$	1.000	**0.002**	1.000	**0.010**	**0.534**	0.603	0.311	**0.230**	0.021	**0.016**	**0.011**	0.018
	S+InstaHide $K{=}8$	1.000	**0.002**	1.000	**0.012**	**0.534**	0.597	0.311	**0.230**	0.020	**0.014**	0.016	**0.014**
	S+Fed-DP $\varepsilon{=}8$	1.000	**0.002**	1.000	**0.008**	**0.522**	0.603	0.311	**0.230**	0.025	**0.007**	**0.007**	0.014
	S+Fed-DP $\varepsilon{=}4$	1.000	**0.002**	1.000	**0.010**	**0.522**	0.603	0.311	**0.230**	0.023	**0.009**	**0.009**	0.012
	S+Fed-DP $\varepsilon{=}1$	1.000	**0.002**	1.000	**0.007**	**0.522**	0.603	0.311	**0.230**	0.024	**0.010**	**0.010**	0.013
	HashEmb	0.991		**0.008**		0.566		0.996		0.358		0.648	
	CHARFORMER	0.968		0.144		0.534		0.984		0.320		0.648	
	ByT5-small	0.968		0.132		0.534		0.984		0.303		0.648	
	MegaByte	0.968		0.085		0.431		0.984		0.358		0.648	
	EmByte	**0.002**		0.023		0.561		**0.241**		0.017		0.017	
	Bits		**0.002**		0.016		**0.489**		**0.241**		**0.013**		**0.011**

Table 2: **Privacy attacks, Subword S vs Bits B.** Token-recovery F1, lower is better, except LLG, where closer to chance is better, 0.5 on SST-2. Top: six attacks on three datasets. Middle: Subword under four published defenses next to Bits trained under the same defense. DP-SGD rows measure FILM, TAG, Inverting Gradients, and iDLG on CIViC, Top-K rows on Devign, all LLG and LAMP values and the InstaHide and Fed-DP rows on SST-2, whose FILM implementation saturates even undefended. Bottom: other encoders on SST-2 without defense. †TAG on IMDB, LLG averaged over Yelp and IMDB.

5 Experiments

Setting. Bits uses $n{=}64$ bits and a $128 \to h \to d$ encoder, with $h{=}512$ on the text tasks of Table 3 and $h{=}128$ elsewhere. **Baselines.** We compare Bits with six encoders, Subword [52], a $V \times d$ table, EmByte [40], with a $2048 \to 128 \to d$ encoder, ByT5-small [68], MegaByte [70], Hash Embeddings [57], and CHARFORMER [59], and with Subword under four defenses, DP-SGD [1], Top-K gradient compression [42], InstaHide [33, 32], and Federated DP [64], configured as in Appendix E.2 for encoders, E.9 for defenses, and E.8 for attacks. **Predictors.** Every comparison swaps only the embedding layer and keeps the predictor body: Qwen2.5 for language modeling, a Transformer encoder-decoder for translation, and a small classifier body for the classification tasks (Table 4). Classifiers use $d{=}64$ and Adam at $5{\cdot}10^{-4}$, and their accuracies are means over seeds 42 to 46 with the data split fixed at seed 42 (Appendix E.6). **Tasks.** There are 32 tasks, including 26 classification tasks (Table 3), CIViC for ablations, the zero-shot transfer Yelp→IMDB, two translation tasks, IWSLT14 and WMT14, and two language-modeling tasks, Qwen2.5-1.5B and 7B (Appendix E.4). **Evaluation criteria.** Privacy uses token-recovery F1 under the six attacks and per-position recovery against chance under the adaptive attack, lower is better. Utility uses top-1 accuracy, BLEU, and held-out perplexity. Size uses parameter count and time uses step time. **Hardware.** Most runs use one H100, the adaptive and defense attacks an RTX 5090, and the 7B runs an 8×B200 node (Appendix E.1). Bold marks the better value of the systems compared.

Privacy. Table 2 runs the six attacks on MedMCQA, PubMedQA, and Yelp/IMDB, with SST-2, CIViC, and Devign carrying the defense rows. FILM reads the per-token rows of the embedding gradient. Bits has no such rows, so against Bits FILM can only guess, and its Bits values in Tables 2 and 3 mark that guessing floor, not a measured recovery. Three findings follow from the other attacks. First, Bits with no defense leaks less than a subword table with one. Under DP-SGD at $\varepsilon \in \{\infty, 16, 8, 4, 1\}$ and $\delta{=}10^{-5}$, Top-K keeping $K \in \{1.0, 0.1, 0.01\}$ of the gradient, InstaHide mixing $K \in \{2, 4, 8\}$ inputs, or federated DP at $\varepsilon \in \{8, 4, 1\}$, Subword still leaks more than

	Dataset	V	Acc / metric S	Acc / metric B	Params K S	Params K B	FILM ↓ S	FILM ↓ B	TAG ↓ S	TAG ↓ B	iDLG ↓ S	iDLG ↓ B
Clinical NLP	PubMedQA°	7,389	0.587	0.589	639	**265**	1.000	**0.029**	1.000	**0.111**	0.118	**0.049**
	ADE	10,642	0.892	0.892	847	**265**	1.000	**0.002**	1.000	**0.021**	0.031	**0.024**
	MedMCQA	261	**0.290**	0.285	**68**	150	1.000	**0.012**	1.000	**0.030**	0.045	**0.042**
	PubMed MeSH	54,286	0.458	**0.462**	3,642	**267**	1.000	**0.139**	1.000	**0.015**	0.029	0.029
	Med abstracts	25,293	0.547	**0.556**	1,785	**265**	1.000	**0.003**	1.000	**0.031**	0.037	**0.030**
Sentiment	IMDB	21,742	0.860	**0.869**	1,558	**265**	0.474	**0.000**	1.000	**0.025**	0.031	**0.021**
	SST-2	9,113	0.811	**0.814**	5,838	**3,703**	0.833	**0.001**	1.000	**0.031**	0.013	**0.012**
Domain class.	Education	5,698	0.285	**0.337**	531	**265**	1.000	**0.008**	1.000	**0.038**	0.050	**0.041**
	Finance	7,571	0.793	**0.804**	651	**265**	1.000	**0.003**	1.000	**0.020**	0.031	**0.017**
	Legal	13,252	**0.804**	0.803	1,027	**278**	1.000	**0.000**	1.000	**0.029**	0.062	**0.046**
	Mental	13,311	**0.528**	0.522	1,021	**268**	1.000	**0.449**	1.000	**0.008**	**0.031**	0.040
	Contract	3,794	0.935	**0.938**	409	**265**	1.000	**0.010**	1.000	**0.056**	**0.083**	0.097
	ECtHR	25,031	0.364	**0.371**	1,769	**266**	1.000	**0.002**	1.000	**0.027**	**0.031**	0.032
Intent	Banking77	1,428	**0.749**	0.731	**267**	275	1.000	**0.008**	1.000	**0.070**	**0.122**	0.130
	CLINC150	2,904	**0.739**	0.732	371	**284**	1.000	**0.005**	1.000	**0.028**	**0.052**	0.067
	MASSIVE	12,907	**0.761**	0.741	1,000	**273**	1.000	**0.290**	1.000	**0.007**	0.039	**0.021**
Genomic Bench.	GB human-promoters	4,098	**0.847**	0.800	428	**191**	1.000	**0.231**	1.000	**0.357**	0.293	**0.144**
	GB human-enhancers	4,098	**0.666**	0.635	428	**191**	1.000	**0.213**	1.000	**0.347**	0.198	**0.116**
	GB human-OCR	4,099	**0.736**	0.713	428	**191**	1.000	**0.217**	1.000	**0.356**	0.262	**0.219**
	GB drosophila-enh.	4,098	**0.589**	0.561	428	**191**	1.000	**0.253**	1.000	**0.364**	0.147	**0.047**
	GB coding-vs-intergen	4,098	**0.903**	0.891	428	**191**	1.000	**0.230**	1.000	**0.369**	0.379	**0.313**
Text	Emotion	7,401	0.903	**0.925**	640	**266**	1.000	**0.005**	1.000	**0.013**	**0.034**	0.035
Code/ Sig.	Devign code	26,491	**0.567**	0.546	1,696	**25**	1.000	**0.342**	1.000	**0.006**	**0.008**	0.014
	Linux syslog	2,942	**0.988**	0.987	189	**25**	0.604	**0.040**	1.000	**0.065**	0.076	**0.062**
	Sleep-EDF EEG°	258	0.672	0.672	**17**	25	0.983	**0.884**	0.953	**0.773**	0.560	**0.393**
	UCI Diabetes 130°	715	0.884	0.884	46	**25**	0.403	**0.201**	1.000	**0.195**	**0.290**	0.301
Transfer	Yelp→IMDB 0-shot	52,204	0.786	**0.801**	7,604	**1,054**	0.356	**0.000**	1.000	**0.028**	0.029	**0.019**

Table 3: **Accuracy, size, and privacy, Subword S vs Bits B.** Accuracy is the mean over seeds 42 to 46, std in Table 8, syslog one seed. Params in K. B uses h=512 on text rows. Body: a 2-layer BiLSTM for text and genomic, mean-pool A for code, log, signal, and tabular. °Both encoders score at the majority-class share. MedMCQA uses a shared clinical encoder over 261 bytes, so S is a byte table, and is scored on the validation split. Yelp→IMDB uses a d=128 LSTM.

Body	Acc	Par.	FILM↓	TAG↓
A: mean pool	0.430	**25**	0.015	0.098
B: BiLSTM	0.804	50	**0.015**	0.099
C: x-attn	0.781	42	0.016	0.093
D: x-attn+MLP	**0.807**	75	0.015	0.097
E: 2-layer trf.	0.805	125	0.015	0.093
F: 1-layer trf.	0.756	75	0.016	0.094
G: 4-layer trf.	0.767	225	0.016	0.096

Width h	Acc	Par.	FILM↓	TAG↓
32	0.849	**56**	0.016	**0.092**
64	0.818	63	**0.014**	0.105
128	**0.866**	75	0.016	0.095
256	0.849	100	0.015	0.095
512	0.790	149	0.017	0.088
1024	0.814	248	0.016	0.090
2048	0.856	446	0.015	0.097

Code	Acc	Par.	FILM↓	TAG↓
Standard	0.872±.014	401	0.145	1.000
Random	0.872±.009	**198**	0.016	**0.095**
MD5 hash	0.866±.025	**198**	**0.015**	**0.095**
Surface	0.874±.007	**198**	**0.015**	0.099
Learned	**0.885**±.008	1,534	**0.015**	0.097
Gray	0.430±.000	**198**	**0.015**	0.103
Hierarch.	0.430±.000	**198**	**0.015**	0.096

Table 4: **Body / width / code ablation on Bits, CIViC,** V=5,474**.** Left: body architectures at h=128, n=64, 4 epochs, defined in Appendix E.3, with the same bodies under a subword table in Appendix F.2. Body D is the default for the CIViC ablations and Body A for the attack cells of Table 3. Middle: FFN width h, body D, n=64, with h=128 the default. Right: token-to-code maps, Random is Bits, body D, n=256, h=256, one shared FFN, mean ± std over five seeds, code definitions in Appendix F.2.1. Par. is the total model parameter count in K. Body rows are 4-epoch runs and width rows 6-epoch runs of the same model, hence 0.807 vs 0.866 for the default.

undefended Bits in every TAG, LAMP, and Inverting Gradients cell, with TAG at 1.00 throughout. Second, the gain holds across a wide range of attacks and datasets. With no defense on either side, Bits ties or beats Subword in 19 of the 20 non-FILM cells: TAG falls by 9.3 to 62× from a saturated 1.00, Inverting Gradients by 3 to 24%, and FL-LLG stays within 0.12 of chance on every task. Third, no other encoder closes the channel on its own. Hash Embeddings keep a $V \times K$ importance matrix over K hash functions and leak FILM 0.991. CHARFORMER, ByT5-small, and MegaByte read 256 bytes that nearly all appear in every input and leak 0.968. EmByte, which also has no per-token row, is the one encoder that leaves FILM at the same guessing floor, and it does so with 11× the encoder parameters, 271K against 25K.

Sequence length L	**1**	**2**	**4**	**8**	**16**	**32**	**64**	**128**	**256**	**512**	**1024**
Standard table	1.000	0.460	0.275	0.128	0.064	0.036	0.018	0.009	0.004	0.003	0.002
Bits	1.000	**0.400**	**0.265**	**0.110**	0.064	**0.021**	**0.008**	**0.005**	**0.001**	**0.001**	**0.001**
Bits, keyed	0.000	0.000	0.000	0.000	0.001	0.001	0.001	0.001	0.001	0.001	0.001

Table 5: **Adaptive full-gradient attack by sequence length.** Batch size 1. The attacker knows codes, seed, and weights. Keyed: the attacker is given a codebook drawn without the key. Per-position exact recovery, chance 0.001, 50 sequences per cell (Appendix F.1.1).

Tokens	**100M**	**200M**	**300M**	**400M**	**500M**	**600M**	**700M**	**800M**	**900M**	**1B**
Standard	111.6±3.4	80.0±2.4	70.1±3.1	63.0±2.8	57.9±2.4	52.2±2.1	46.9±1.8	43.3±1.5	41.6±1.4	41.3±1.4
Bits	**97.7±0.8**	**73.2±0.7**	**63.4±0.7**	**59.9±2.8**	**54.2±0.9**	**50.2±0.9**	**44.8±0.3**	**41.1±0.2**	**39.2±0.4**	**38.8±0.4**
Gap	14.0	6.8	6.8	3.1	3.7	2.0	2.1	2.2	2.4	2.5

	Embedding params	**Total params**	**FILM recovery**		**TAG recovery**	
			L=4	L=256	L=4	L=256
Standard	545M	7.62B	0.75	0.996	0.75	0.996
Bits	1.97M	7.07B	0.000	0.0013	0.000	0.0013
Ratio	277× smaller	7% smaller	chance	0.00003	0.0017	chance	0.00003	0.0017

Table 6: **From-scratch Qwen2.5-7B on C4**, held-out perplexity at every 100M-token checkpoint of a 1B-token run, mean ± std over seeds 42, 43, 44 per arm, only the input embedding swapped. Cosine schedule over the full 1B tokens, so checkpoints are not comparable to the 100M-horizon runs in §5. Below: embedding and total parameters, and FILM and TAG token recovery at batch size 1 for sequence lengths L=4 and L=256, where chance is L/V with V=152,064. FILM and TAG agree to three digits because both read the same set of nonzero embedding rows. Curves in Figure 4, setup and the 1.5B attack rows in Appendix F.4.

Accuracy. Table 3 swaps only the embedding on 26 classification tasks, so every accuracy difference is the encoder's. On PubMedQA, Sleep-EDF, and UCI Diabetes 130 both encoders only match the majority-class share, so accuracy cannot separate them. On the other 23, Bits trails the subword table by 0.4 points on average. On the 16 text tasks it leads by 0.4 and is ahead on 10, and on the genomic, code, and log tasks it trails by 2.3. On translation it trails by 0.3 BLEU on IWSLT14 de→en and 1.1 on WMT14 en→de, with a 198× smaller input embedding (Appendix E.7).

Efficiency. Bits is the smaller model in 23 of the 26 classification rows of Table 3, all but Sleep-EDF at V=258, Banking77 at V=1,428, and the shared-encoder MedMCQA row, and the gap grows with V: 1.3× over Subword at $V \approx$ 2,904, 14× at $V \approx$ 54K, where the table is 95% of all parameters and the embedding layer shrinks by 97%. EmByte, which still trains a per-byte table, is 1.9× larger than Bits on MASSIVE. Speed is unchanged at h=128 (protocol in Appendix E.10): an SST-2 federated client step at batch 32 takes 1.39 ms for Bits, 1.30 for Subword, and 1.53 for EmByte, and a 4-layer MT step at V=8K, d=128, batch 16 takes 10.58, 11.23, and 10.84 ms. These times read an unpacked code table, and unpacking a bit-packed one adds about 0.27 ms per encoder step (Appendix E.10). The MASSIVE on-device models hold 1.09 MB of parameters for Bits, plus a 103 KB bit-packed code table, against 4.00 MB for Subword and 2.04 for EmByte.

Transfer. Zero-shot from Yelp to IMDB, Bits reaches 0.801 against 0.786 over five seeds, winning on every seed, a 1.5-point gain larger than its 0.5 points on Yelp itself. Both encoders share the Yelp vocabulary and map unseen IMDB tokens to the unknown token, so the gain is not from coverage. The checkpoint is chosen on Yelp test accuracy. The model is 7.2× smaller, and FILM falls from 0.356 on Subword to chance (Table 3).

Ablations. Each varies one factor on CIViC, all else fixed (Tables 4 and 14). **Body.** The body sets accuracy, the encoder sets privacy. Table 3 uses a 2-layer BiLSTM for the text and genomic rows and the mean-pool Body A for attacks. On CIViC, mean pooling is worst for both encoders at 0.430, and Body D, cross-attention with an MLP residual, is best for Bits at 0.807 with 5× fewer parameters than Subword. Across bodies C to G (Appendix F.2), Bits trails Subword by 0.058 to 0.110, yet keeps FILM near 0.015 and TAG near 0.10, while Subword reaches FILM 0.134 to 0.817 and TAG 1.00. **Width.** It barely matters: h=32 and 128 reach 0.849 and 0.866 with 56K and 75K parameters, h=2,048 costs six times more for no gain, and all lie far below the $h \geq V$=5,474 of Theorem 4. **Code map.** It need not be random, but it must fill the bit space. Random, MD5 as in SpinHash [67], character-surface, and learned codes all match the subword table at 0.866 to 0.885, with TAG

near 0.10. Gray and hierarchical codes vary on only 13 of 256 positions and collapse to 0.430. Random codes fill the space with no codebook or extra parameters. **Code length.** Collisions hurt below $\lceil \log_2 V \rceil$ (0.595 at n=8), and accuracy plateaus from $n \approx 256$, so $n \geq \max(\lceil \log_2 V \rceil, 256)$. Training is stable, as the 7B loss curves show (Figure 4): every token passes through the same FFN, so nearly every parameter gets a gradient each step. The codes are just addresses; the FFN learns their meaning and pulls similar tokens together (Appendix F.3).

Adaptive and federated attacks. The six attacks above target an embedding table, which Bits removes. To rule out that they are the wrong attacks, we build a dedicated attack, the adaptive full-gradient attack, aimed at Bits only, with results in Table 5: an attacker who knows the codebook, the seed, and every weight optimizes a candidate sequence through the shared FFN to match the full gradient at batch size 1 (Appendix F.1.1). Per position, it does no better on Bits than on the subword table at every length from 2 to $1{,}024$ tokens and reaches chance, $1/V$=0.001 on the V=$1{,}000$ proxy, at 256 tokens, where the subword table is still $4\times$ above it. On token sets, which ignore position, Bits is above the subword table below 64 tokens and below it from 64 on (Table 9). It recovers short inputs, a one-token input exactly, as from any embedding, since that gradient is a function of the token alone, and 0.400 of positions at two tokens. The key closes even that, leaving the attacker at or below chance at every length (Table 5). Bits stays at or below the subword table for batch sizes 1 to 32 except 4 and under DLG, iDLG, Inverting Gradients, TAG, and LAMP (Tables 10 and 11), and side information does not help: a malicious server, 0 to 75% known text, or 8 federated rounds leave Bits at or below 0.016, against 0.072 to 1.000 for the subword table (Table 12).

Scaling to LLMs. We train Qwen2.5-1.5B and 7B from scratch on C4 with a Bits encoder of n=256 and h=512 (Table 6, Appendix F.4). At 7B, over three seeds and 1B tokens, Bits leads at every 100M-token checkpoint of every seed and ends at $38.8{\pm}0.4$ vs $41.3{\pm}1.4$ perplexity. The gap shrinks from 14.0 at 100M tokens and holds at 2.0 to 2.5 from 600M on (Figure 4). Single-seed 100M-token runs agree at both scales, 91.2 vs 108.8 at 1.5B and 85.9 vs 96.3 at 7B, with a 254 to $277\times$ smaller embedding. FILM and TAG recover 75 to 99.6% of tokens from the subword table and stay at chance for Bits (Table 16). At 0.014 to 0.14 tokens per parameter, these runs measure sample efficiency under a matched budget, not converged performance (§7). All 66 7B checkpoints, ten per run plus the last, are at `https://huggingface.co/bit2vec/bitsseal`.

6 Conclusion

Bits removes the embedding table, and with it the per-token gradient rows that inversion attacks read. What remains recoverable shrinks as more tokens share one gradient, and with an optional key even a single token is as hard to recover as the cipher is to break. Across six modalities it cuts gradient-inversion token recovery by up to $167\times$ relative to a subword table, at comparable accuracy and with an embedding that no longer grows with the vocabulary. Its gains are largest where a subword table starts from untrained rows and a byte model pays a longer sequence: federated learning, edge deployment, and transfer to new datasets. Collapsing the input vocabulary to two symbols is therefore not only a compression. It improves privacy, memory, and transfer at once.

7 Limitations

Bits closes the embedding-gradient channel only and gives no differential-privacy guarantee. Body gradients, outputs, and losses still leak and need DP-SGD, and our language models keep a per-token output layer. A code-aware attacker still recovers short inputs, a single token exactly (Table 5). The key removes this by Theorem 2, but we tested the keyed encoder against this attacker only, not against the six attacks of Table 2. DAGER and SPEAR [51, 17] were not run. Our claims cover classification, translation, and language modeling, not regression. The 7B runs cover 1B tokens, $140\times$ below the compute-optimal 20 tokens per parameter of Chinchilla [29], and the gap narrows from 14.0 to between 2.0 and 2.5, so whether the advantage holds at full scale remains open.

Acknowledgments and Disclosure of Funding

This work received no external funding. The author has filed provisional patent applications.

References

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 308–318, 2016.

[2] Mislav Balunović, Dimitar I. Dimitrov, Nikola Jovanović, and Martin Vechev. LAMP: Extracting text from gradients with language model priors. In *Advances in Neural Information Processing Systems*, volume 35, pages 7641–7654, 2022.

[3] Mislav Balunović, Dimitar I. Dimitrov, Robin Staab, and Martin Vechev. Bayesian framework for gradient leakage. In *International Conference on Learning Representations*, 2022.

[4] Mihir Bellare, Thomas Ristenpart, Phillip Rogaway, and Till Stegers. Format-preserving encryption. In *Selected Areas in Cryptography (SAC 2009)*, volume 5867 of *Lecture Notes in Computer Science*, pages 295–312, 2009.

[5] John Black and Phillip Rogaway. Ciphers with arbitrary finite domains. In *Topics in Cryptology – CT-RSA 2002*, volume 2271 of *Lecture Notes in Computer Science*, pages 114–130, 2002.

[6] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 1175–1191, 2017.

[7] Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramèr. Membership inference attacks from first principles. In *IEEE Symposium on Security and Privacy*, pages 1897–1914, 2022.

[8] Nicholas Carlini, Samuel Deng, Sanjam Garg, Somesh Jha, Saeed Mahloujifar, Mohammad Mahmoody, Abhradeep Thakurta, and Florian Tramèr. Is private learning possible with instance encoding? In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 410–427, 2021.

[9] Iñigo Casanueva, Tadas Temčinas, Daniela Gerz, Matthew Henderson, and Ivan Vulić. Efficient intent detection with dual sentence encoders. In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*, pages 38–45, 2020.

[10] Ting Chen, Martin Renqiang Min, and Yizhou Sun. Learning K-way D-dimensional discrete codes for compact embedding representations. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 854–863. PMLR, 2018.

[11] Hong-Min Chu, Jonas Geiping, Liam H. Fowl, Micah Goldblum, and Tom Goldstein. Panning for gold in federated learning: Targeted text extraction under arbitrarily large-scale aggregation. In *International Conference on Learning Representations*, 2023.

[12] Jonathan H Clark, Dan Garrette, Iulia Turc, and John Wieting. CANINE: Pre-training an efficient tokenization-free encoder for language representation. *Transactions of the Association for Computational Linguistics*, 10:73–91, 2022.

[13] Benjamin Coleman, Wang-Cheng Kang, Matthew Fahrbach, Ruoxi Wang, Lichan Hong, Ed H. Chi, and Derek Zhiyuan Cheng. Unified embedding: Battle-tested feature representations for web-scale ML systems. In *Advances in Neural Information Processing Systems*, volume 36, pages 56234–56255, 2023.

[14] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES – The Advanced Encryption Standard*. Springer, 2002.

[15] Björn Deiseroth, Manuel Brack, Patrick Schramowski, Kristian Kersting, and Samuel Weinbach. T-FREE: Subword tokenizer-free generative LLMs via sparse representations for memory-efficient embeddings. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 21829–21851, 2024.

[16] Jieren Deng, Yijue Wang, Ji Li, Chenghong Wang, Chao Shang, Hang Liu, Sanguthevar Rajasekaran, and Caiwen Ding. TAG: Gradient attack on transformer-based language models. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3600–3610, 2021.

[17] Dimitar I. Dimitrov, Maximilian Baader, Mark Niklas Müller, and Martin Vechev. SPEAR: Exact gradient inversion of batches in federated learning. In *Advances in Neural Information Processing Systems*, volume 37, pages 106768–106799, 2024.

[18] Angela Fan, Shruti Bhosale, Holger Schwenk, Zhiyi Ma, Ahmed El-Kishky, Siddharth Goyal, Mandeep Baines, Onur Celebi, Guillaume Wenzek, Vishrav Chaudhary, Naman Goyal, Tom Birch, Vitaliy Liptchinsky, Sergey Edunov, Michael Auli, and Armand Joulin. Beyond English-centric multilingual machine translation. *Journal of Machine Learning Research*, 22(107):1–48, 2021.

[19] Jack FitzGerald, Christopher Hench, Charith Peris, Scott Mackie, Kay Rottmann, Ana Sanchez, Aaron Nash, Liam Urbach, Vishesh Kakarala, Richa Singh, Swetha Ranganath, Laurie Crist, Misha Britan, Wouter Leeuwis, Gokhan Tur, and Prem Natarajan. MASSIVE: A 1M-example multilingual natural language understanding dataset with 51 typologically-diverse languages. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4277–4302, 2023.

[20] Liam Fowl, Jonas Geiping, Steven Reich, Yuxin Wen, Wojciech Czaja, Micah Goldblum, and Tom Goldstein. Decepticons: Corrupted transformers breach privacy in federated learning for language models. In *International Conference on Learning Representations*, 2023.

[21] Liam H Fowl, Jonas Geiping, Wojciech Czaja, Micah Goldblum, and Tom Goldstein. Robbing the fed: Directly obtaining private data in federated learning with modified models. In *International Conference on Learning Representations*, 2022.

[22] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1322–1333, 2015.

[23] Karan Ganju, Qi Wang, Wei Yang, Carl A Gunter, and Nikita Borisov. Property inference attacks on fully connected neural networks using permutation invariant representations. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 619–633, 2018.

[24] Kostadin Garov, Dimitar I. Dimitrov, Nikola Jovanović, and Martin Vechev. Hiding in plain sight: Disguising data stealing attacks in federated learning. In *International Conference on Learning Representations*, 2024.

[25] Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients – how easy is it to break privacy in federated learning? In *Advances in Neural Information Processing Systems*, volume 33, pages 16937–16947, 2020.

[26] Katarína Grešová, Vlastimil Martinek, David Čechák, Petr Šimeček, and Panagiotis Alexiou. Genomic benchmarks: A collection of datasets for genomic sequence classification. *BMC Genomic Data*, 24(1):25, 2023.

[27] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. BadNets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

[28] Samyak Gupta, Yangsibo Huang, Zexuan Zhong, Tianyu Gao, Kai Li, and Danqi Chen. Recovering private text in federated learning of language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 8130–8143, 2022.

[29] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Thomas Hennigan, Eric Noland, Katherine Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karén Simonyan, Erich Elsen, Oriol Vinyals, Jack Rae, and Laurent Sifre. An empirical analysis of compute-optimal large language model training. In *Advances in Neural Information Processing Systems*, volume 35, pages 30016–30030, 2022.

[30] Hongzhi Huang, Defa Zhu, Banggu Wu, Yutao Zeng, Ya Wang, Qiyang Min, and Xun Zhou. Over-tokenized transformer: Vocabulary is generally worth scaling. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 26261–26282, 2025.

[31] Qian Huang, Eric Zelikman, Sarah Li Chen, Yuhuai Wu, Gregory Valiant, and Percy Liang. Lex-invariant language models. In *Advances in Neural Information Processing Systems*, volume 36, pages 23990–24012, 2023.

[32] Yangsibo Huang, Zhao Song, Danqi Chen, Kai Li, and Sanjeev Arora. TextHide: Tackling data privacy in language understanding tasks. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1368–1382, 2020.

[33] Yangsibo Huang, Zhao Song, Kai Li, and Sanjeev Arora. InstaHide: Instance-hiding schemes for private distributed learning. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 4507–4518, 2020.

[34] Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. PubMedQA: A dataset for biomedical research question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2567–2577, 2019.

[35] Julie Kallini, Shikhar Murty, Christopher D. Manning, Christopher Potts, and Róbert Csordás. MrT5: Dynamic token merging for efficient byte-level language models. In *International Conference on Learning Representations*, 2025.

[36] Wang-Cheng Kang, Derek Zhiyuan Cheng, Tiansheng Yao, Xinyang Yi, Ting Chen, Lichan Hong, and Ed H. Chi. Learning to embed categorical features without embedding tables for recommendation. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 840–850, 2021.

[37] Bob Kemp, Aeilko H Zwinderman, Bert Tuk, Hilbert A C Kamphuisen, and Josefien J. L. Oberye. Analysis of a sleep-dependent neuronal feedback loop: The slow-wave microcontinuity of the EEG. *IEEE Transactions on Biomedical Engineering*, 47(9):1185–1194, 2000.

[38] Taku Kudo and John Richardson. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, 2018.

[39] Stefan Larson, Anish Mahendran, Joseph J. Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K. Kummerfeld, Kevin Leach, Michael A. Laurenzano, Lingjia Tang, and Jason Mars. An evaluation dataset for intent classification and out-of-scope prediction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1311–1316, 2019.

[40] Shenglan Li, Jia Xu, and Mengjiao Zhang. EmByte: Decomposition and compression learning for small yet private NLP. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 7182–7201, 2025.

[41] Xuechen Li, Florian Tramèr, Percy Liang, and Tatsunori Hashimoto. Large language models can be strong differentially private learners. In *International Conference on Learning Representations*, 2022.

[42] Yujun Lin, Song Han, Huizi Mao, Yu Wang, and William Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. In *International Conference on Learning Representations*, 2018.

[43] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[44] Benjamin Minixhofer, Edoardo Maria Ponti, and Ivan Vulić. Zero-shot tokenizer transfer. In *Advances in Neural Information Processing Systems*, volume 37, pages 46791–46818, 2024.

[45] John X. Morris, Volodymyr Kuleshov, Vitaly Shmatikov, and Alexander M. Rush. Text embeddings reveal (almost) as much as text. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12448–12460, 2023.

[46] Piotr Nawrot, Szymon Tworkowski, Michał Tyrolski, Łukasz Kaiser, Yuhuai Wu, Christian Szegedy, and Henryk Michalewski. Hierarchical transformers are more efficient language models. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1559–1571, 2022.

[47] NLLB Team, Marta R. Costa-jussà, James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, et al. No language left behind: Scaling human-centered machine translation. *arXiv preprint arXiv:2207.04672*, 2022.

[48] Artidoro Pagnoni, Ramakanth Pasunuru, Pedro Rodriguez, John Nguyen, Benjamin Muller, Margaret Li, Chunting Zhou, Lili Yu, Jason E. Weston, Luke Zettlemoyer, Gargi Ghosh, Mike Lewis, Ari Holtzman, and Srini Iyer. Byte latent transformer: Patches scale better than tokens. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9238–9258, 2025.

[49] Ankit Pal, Logesh Kumar Umapathi, and Malaikannan Sankarasubbu. MedMCQA: A large-scale multi-subject multi-choice dataset for medical domain question answering. In *Proceedings of the Conference on Health, Inference, and Learning*, volume 174 of *Proceedings of Machine Learning Research*, pages 248–260, 2022.

[50] Nicolas Papernot, Shuang Song, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Úlfar Erlingsson. Scalable private learning with PATE. In *International Conference on Learning Representations*, 2018.

[51] Ivo Petrov, Dimitar I. Dimitrov, Maximilian Baader, Mark Niklas Müller, and Martin Vechev. DAGER: Exact gradient inversion for large language models. In *Advances in Neural Information Processing Systems*, volume 37, pages 87801–87830, 2024.

[52] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, 2016.

[53] Uri Shaham and Omer Levy. Neural machine translation without embeddings. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 181–186, 2021.

[54] Kevin Slagle. SpaceByte: Towards deleting tokenization from large language modeling. In *Advances in Neural Information Processing Systems*, volume 37, pages 124925–124950, 2024.

[55] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.

[56] Beata Strack, Jonathan P DeShazo, Chris Gennings, Juan L Olmo, Sebastian Ventura, Krzysztof J Cios, and John N Clore. Impact of HbA1c measurement on hospital readmission rates: Analysis of 70,000 clinical database patient records. *BioMed Research International*, 2014:781670, 2014.

[57] Dan Tito Svenstrup, Jonas Hansen, and Ole Winther. Hash embeddings for efficient word representations. In *Advances in Neural Information Processing Systems*, volume 30, pages 4928–4936, 2017.

[58] Chaofan Tao, Qian Liu, Longxu Dou, Niklas Muennighoff, Zhongwei Wan, Ping Luo, Min Lin, and Ngai Wong. Scaling laws with vocabulary: Larger models deserve larger vocabularies. In *Advances in Neural Information Processing Systems*, volume 37, pages 114147–114179, 2024.

[59] Yi Tay, Vinh Q. Tran, Sebastian Ruder, Jai Gupta, Hyung Won Chung, Dara Bahri, Zhen Qin, Simon Baumgartner, Cong Yu, and Donald Metzler. Charformer: Fast character transformers via gradient-based subword tokenization. In *International Conference on Learning Representations*, 2022.

[60] Stacey Truex, Nathalie Baracaldo, Ali Anwar, Thomas Steinke, Heiko Ludwig, Rui Zhang, and Yi Zhou. A hybrid approach to privacy-preserving federated learning. In *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security (AISec)*, pages 1–11, 2019.

[61] Mark Vero, Mislav Balunović, Dimitar Iliev Dimitrov, and Martin Vechev. TabLeak: Tabular data leakage in federated learning. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 35051–35083, 2023.

[62] Aidmar Wainakh, Fabrizio Ventola, Till Müßig, Jens Keim, Carlos García Cordero, Ephraim Zimmer, Tim Grube, Kristian Kersting, and Max Mühlhäuser. User-level label leakage from gradients in federated learning. *Proceedings on Privacy Enhancing Technologies*, 2022(2):227–244, 2022.

[63] Junxiong Wang, Tushaar Gangavarapu, Jing Nathan Yan, and Alexander M. Rush. MambaByte: Token-free selective state space model. In *First Conference on Language Modeling*, 2024.

[64] Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H. Yang, Farhad Farokhi, Shi Jin, Tony Q. S. Quek, and H. Vincent Poor. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security*, 15:3454–3469, 2020.

[65] Kilian Q Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. Feature hashing for large scale multitask learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1113–1120, 2009.

[66] Yuxin Wen, Jonas Geiping, Liam Fowl, Micah Goldblum, and Tom Goldstein. Fishing for user data in large-batch federated learning via gradient magnification. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 23668–23684, 2022.

[67] Jia Xu, Jinhua Wang, and Constantin F. Aliferis. SpinHash: An O(1) encoder for any vocabulary. In *Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing*, pages 13165–13180, 2026.

[68] Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. ByT5: Towards a token-free future with pre-trained byte-to-byte models. *Transactions of the Association for Computational Linguistics*, 10:291–306, 2022.

[69] Da Yu, Saurabh Naik, Arturs Backurs, Sivakanth Gopi, Huseyin A. Inan, Gautam Kamath, Janardhan Kulkarni, Yin Tat Lee, Andre Manoel, Lukas Wutschitz, Sergey Yekhanin, and Huishuai Zhang. Differentially private fine-tuning of language models. In *International Conference on Learning Representations*, 2022.

[70] Lili Yu, Daniel Simig, Colin Flaherty, Armen Aghajanyan, Luke Zettlemoyer, and Mike Lewis. MEGABYTE: Predicting million-byte sequences with multiscale transformers. In *Advances in Neural Information Processing Systems*, volume 36, pages 78808–78823, 2023.

[71] Chulhee Yun, Suvrit Sra, and Ali Jadbabaie. Small ReLU networks are powerful memorizers: A tight analysis of memorization capacity. In *Advances in Neural Information Processing Systems*, volume 32, pages 15558–15569, 2019.

[72] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*, 2017.

[73] Mengjiao Zhang and Jia Xu. Byte-based multilingual NMT for endangered languages. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 4407–4417, Gyeongju, Republic of Korea, October 2022. International Committee on Computational Linguistics.

[74] Zhuo Zhang, Yuanhang Yang, Yong Dai, Qifan Wang, Yue Yu, Lizhen Qu, and Zenglin Xu. FedPETuning: When federated learning meets the parameter-efficient tuning methods of pre-trained language models. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 9963–9977, 2023.

[75] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. iDLG: Improved deep leakage from gradients. *arXiv preprint arXiv:2001.02610*, 2020.

[76] Yaqin Zhou, Shangqing Liu, Jingkai Siow, Xiaoning Du, and Yang Liu. Devign: Effective vulnerability identification by learning comprehensive program semantics via graph neural networks. In *Advances in Neural Information Processing Systems*, volume 32, pages 10197–10207, 2019.

[77] Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. In *Advances in Neural Information Processing Systems*, volume 32, pages 14774–14784, 2019.

Appendix

A Notation

Notation. Symbols in order of first use. Bold lower-case letters are vectors, calligraphic letters are sets or maps.

Symbol	Meaning
Encoder (§3)	
$\mathcal{V}_w$, V	subword vocabulary and its size, V from 258 to 54,286 in Table 3
$\mathcal{V}_b$, V_b	input alphabet of a byte-based encoder and its size, 256 for bytes, 2 for Bits
w_i, m	the i-th subword of an input and the number of subwords
t	a token id, $t \in \{1, \ldots, V\}$
d	embedding dimension, d=64 for classification
n	code length in bits, n=64 unless noted
$\mathcal{M}$, $\mathbf{b}_i$, b_{ij}	token-to-code map, the code of w_i, and its j-th bit
$b(t)$	the code of token t, $b(t) = \mathcal{M}(w_t) \in \{0,1\}^n$ (§4)
$\mathbf{o}_{ij}$	one-hot of bit b_{ij}, $(1 - b_{ij}, b_{ij})^\top$
$\mathbf{c}_i$	encoder input of w_i, the n one-hots stacked, $\mathbf{c}_i \in \mathbb{R}^{2n}$
$W_1, \mathbf{u}_1, W_2, \mathbf{u}_2$	encoder weights and biases, $W_1 \in \mathbb{R}^{h \times 2n}$, $W_2 \in \mathbb{R}^{d \times h}$
h	encoder hidden width, h=128 default, 512 on text tasks
σ, p	ReLU and the dropout rate on the hidden layer, p=0.1 by default
$\mathbf{z}$, $\mathbf{e}_i$	hidden activation and the output embedding of w_i, $\mathbf{e}_i \in \mathbb{R}^d$
ϵ	the Bits encoder as a map $\{0,1\}^n \to \mathbb{R}^d$
θ_ϵ, θ_{body}	encoder and body parameters, $\lvert\theta_\epsilon\rvert = h(2n + d + 1) + d = 24{,}768$ at default sizes
$\mathcal{S}$, r, T	in Algorithm 1: codes assigned so far, redraw counter, redraw budget T=1000
c	bytes per subword, $c \approx 5$ (Table 1)
Theory (§4)	
E, $E[t]$	subword embedding table $E \in \mathbb{R}^{V \times d}$ and its row for token t
$\mathcal{B}$, $\lvert\mathcal{B}\rvert$	a batch of token ids and its size, written s in Appendix C
ℓ, $\mathcal{L}$	per-token loss and batch loss $\sum_{t \in \mathcal{B}} \ell(t)$
g	encoder gradient of a batch, $\sum_{t \in \mathcal{B}} \nabla_{\theta_\epsilon} \ell(\epsilon(b(t)))$
supp	support, the set of nonzero entries of a gradient
H, I	entropy and mutual information, in bits unless a bound says nats
$\hat{t}$	an attacker's predictor of one batch token from g
$\mathcal{A}$	the adversary: server, eavesdropper, or post-hoc analyst
κ, λ	the optional secret key and its length in bits, $\kappa \in \{0,1\}^\lambda$, λ=128
Π, π_κ	a family of pseudorandom permutations of $\{0,1\}^n$ and the member selected by κ
ϵ_κ	keyed encoder, $\epsilon_\kappa(t) = \epsilon(\pi_\kappa(b(t)))$
TR-IND, Adv, $\mathcal{D}$	token-recovery game, attacker advantage in it, and the PRP distinguisher built from $\mathcal{A}$
ρ	a truly random permutation of $\{0,1\}^n$ in the proof of Theorem 2
$R(V)$	embedding compression ratio $Vd/\lvert\theta_\epsilon\rvert$
τ	approximation tolerance in Theorem 4
Experiments (§5)	
L	sequence length in the adaptive attack, tokens per batch at batch size 1
K	Top-K kept fraction, InstaHide mixing count, or Hash Embeddings hash count, as labeled
ε, δ	DP-SGD and Federated DP privacy parameters, distinct from the encoder ϵ
k	k-mer length of genomic tokens

B Encoder Implementation

- The one-hot encoding of bit values is implemented as a fixed, non-learnable, embedding layer with weights $\mathbf{I}_2 = [[1,0],[0,1]]^\top$.
- The FFN uses a 2-layer architecture: $\mathbb{R}^{2n} \to \mathbb{R}^h \to \mathbb{R}^d$ with ReLU activation and dropout after the hidden layer, 0.1 by default and 0.4 in the BiLSTM and sentiment LSTM runs. Default h=128.
- Batch processing: all operations are parallelized across the batch and sequence dimensions. No per-token loop.
- Bit codes are precomputed once and stored as a lookup table of shape (V, n), one `int64` per bit in the trained code. Lookup is a single `torch.embedding` call. For storage the table can be bit-packed as `uint8`, see Memory accounting below.
- The codes are sampled by Algorithm 1 with retry budget T=1000 to enforce uniqueness. In all experiments the budget was sufficient.

Bytes per subword	Byte vocabulary size V_b		
	64	128	256
4	0.79M	1.05M	1.57M
8	1.05M	1.57M	2.62M
16	1.57M	2.62M	4.72M

Table 7: **Embedding parameter count as a function of bytes per subword and byte vocabulary V_b.** Values are for an EmByte-style aggregator with FFN hidden width 1024. Total embedding params scale with bytes per subword times $V_b d$. Bits replaces the V_b axis with the binary minimum $V_b{=}2$, so the V_b contribution disappears and the parameter count drops to $h(2n + d + 1) + d$ for hidden width h. Even at $V_b{=}256$ and 4 bytes per subword the EmByte-style configuration uses 1.57M parameters. Bits at $n{=}64$, $d{=}64$ uses 24,768, a $63\times$ reduction.

- During backpropagation, only the FFN parameters θ_ϵ receive gradients. The bit map b is frozen.

Memory accounting. Computed for $V{=}50$K subwords with $n{=}64$ bits, in decimal MB. Stored bit-packed as `uint8`, eight bits per byte, the mapping table takes 400,000 bytes, 400 KB. The FFN, $128 \to 128 \to 64$, with biases, has 24,768 parameters, 99 KB in `float32`. Total embedding memory is 0.50 MB, compared with 12.8 MB for a standard $50{,}000 \times 64$ subword table in `float32`, a 96% reduction. Unpacking reproduces the stored codes exactly, so every result in the paper, all trained with one `int64` per bit, is unchanged. Stored that way the table would take 25.6 MB, more than the subword table. Measured on an RTX 5090, unpacking adds about 0.27 ms to an encoder step at batch 32×256, 1.65 against 1.38 ms. The trainable embedding parameters alone shrink by 99%.

C Proofs of the Core Theorems

This section gives proof sketches and full proofs for the theorems in §4: privacy (Theorems 1 and 2), memory (Theorem 3), and expressivity (Theorem 4). The TR-IND game definition, threat model, and detailed cryptographic upgrade are in Appendix C.4. Each theorem here is stated in the form it appears in the main text and proved in two passes: a proof sketch that names the load-bearing step, and a full proof that fills in the calculations.

C.1 Setup

Let V be the vocabulary, $b : V \to \{0,1\}^n$ the public injective bit map (default $n{=}64$, sampled by Algorithm 1 with each bit $\mathrm{Bernoulli}(0.5)$, hence expected weight 32), and $\epsilon : \{0,1\}^n \to \mathbb{R}^d$ the Bits encoder. We write $\epsilon(t)$ for $\epsilon(b(t))$ when b is fixed. The encoder is a two-layer ReLU FFN

$$\epsilon(b) \;=\; W_2\,\sigma(W_1\,\mathrm{onehot}(b) + \mathbf{u}_1) + \mathbf{u}_2, \qquad W_1 \in \mathbb{R}^{h\times 2n},\ \mathbf{u}_1 \in \mathbb{R}^h,\ W_2 \in \mathbb{R}^{d\times h},\ \mathbf{u}_2 \in \mathbb{R}^d,$$

where $\mathrm{onehot}(b) \in \{0,1\}^{2n}$ expands each bit b_i into the interleaved pair $(1 - b_i,\ b_i)$ at positions $(2i-1, 2i)$, as in Algorithm 2 of §3. Body parameters θ_{body} and final classifier weights are everything outside $\theta_\epsilon \triangleq (W_1, W_2, \mathbf{u}_1, \mathbf{u}_2)$. For a mini-batch $\mathcal{B} = \{t_1, \ldots, t_s\}$ with labels $\{y_1, \ldots, y_s\}$, the training loss is $\mathcal{L}(\mathcal{B}) = \sum_i \ell(\epsilon(t_i), \theta_{\mathrm{body}}, y_i)$ and the encoder gradient is $g \triangleq \nabla_{\theta_\epsilon}\mathcal{L}(\mathcal{B}) = \sum_i \nabla_{\theta_\epsilon}\,\ell_i$. The Subword baseline replaces ϵ with $E \in \mathbb{R}^{V\times d}$, so $\nabla_E\mathcal{L}$ has support exactly on the rows indexed by $\mathcal{B}$, one row per token.

The keyed variant is $\epsilon_\kappa(t) \triangleq \epsilon(\pi_\kappa(b(t)))$ where $\Pi = \{\pi_\kappa\}$ is a PRP family on $\{0,1\}^n$ keyed by $\kappa \in \{0,1\}^\lambda$, a 64-bit block cipher or a format-preserving cipher. The adversary $\mathcal{A}$ observes the public tuple $(\theta_\epsilon, \theta_{\mathrm{body}}, g, b, V)$ and never κ.

C.2 Theorem 1: Privacy Floor

Theorem 5 (Privacy floor, restated)**.** *Let $\mathcal{B} \subseteq \{1, \ldots, V\}$ with $|\mathcal{B}| = s$, we write s for the batch size throughout this appendix. Without the key:* ***(a)*** *The column-support of $\nabla_{W_1}\mathcal{L}$ equals the bit-wise OR $\bigvee_{t\in\mathcal{B}} \mathrm{onehot}(b(t))$. No parameter of θ_ϵ is indexed by a single token.* ***(b)*** *If $\mathcal{B}$ is drawn uniformly from $\binom{V}{|\mathcal{B}|}$, then for any predictor $\hat{t}$ $H(\mathcal{B} \mid g) \geq \log_2 \binom{V}{|\mathcal{B}|} - I(\mathcal{B}; g)$ and $\Pr[\hat{t}(g) \in \mathcal{B}] \leq |\mathcal{B}|/V + \sqrt{I(\mathcal{B}; g)/2}$, with I in nats in the second bound. For the keyed encoder ϵ_κ we make no*

information-theoretic claim on $I(\mathcal{B}; g)$, since an unbounded adversary can exhaust the λ-bit key. The keyed guarantee is the computational statement of Theorem 2.

This is the statement of Theorem 1 in §4, with the same $I(\mathcal{B}; g)$ remainder. Without the key the remainder has no analytic bound and is estimated per attack, Appendix F.1.1. With the key the computational guarantee of Theorem 2 applies instead, while the structural part (a) and the entropy bound under a uniform prior hold as stated. Below we prove (a) unconditionally, and (b) under the uniform prior.

Proof of (a). By the chain rule on the FFN $\epsilon(x) = W_2\sigma(W_1 x + \mathbf{u}_1) + \mathbf{u}_2$, the per-token first-layer gradient factors as

$$\nabla_{W_1}\ell_t \;=\; \underbrace{\operatorname{diag}\big(\sigma'(W_1\,\mathrm{onehot}(b(t)) + \mathbf{u}_1)\big)\, W_2^\top\, \nabla_{\epsilon(b(t))}\ell_t}_{\mathbf{v}_t \in \mathbb{R}^h}\; \mathrm{onehot}(b(t))^\top \;\in\; \mathbb{R}^{h\times 2n}.$$

Summing over $t \in \mathcal{B}$ gives

$$\nabla_{W_1}\mathcal{L} \;=\; \sum_{t\in\mathcal{B}} \mathbf{v}_t\, \mathrm{onehot}(b(t))^\top. \tag{$\star$}$$

The right factor of each summand, $\mathrm{onehot}(b(t))$, has exactly one nonzero entry in each coordinate pair $(2i-1, 2i)$: the second slot $2i$ when $b(t)_i = 1$ and the first slot $2i-1$ when $b(t)_i = 0$. Its ℓ_1 norm is therefore always n, and what varies with t is the pattern of which slot carries the 1 in each pair. The union over $t \in \mathcal{B}$ of these patterns gives column support $\bigcup_t \mathrm{supp}\big(\mathrm{onehot}(b(t))\big) = \bigvee_t \mathrm{onehot}(b(t))$: column $2i$ is in the support iff some batch token has $b_i = 1$, and column $2i-1$ iff some batch token has $b_i = 0$.

Each row of $\nabla_{W_1}\mathcal{L}$ is a linear combination over *every* token in the batch, so no parameter belongs to one token. For small batches the combination can still be inverted by an attacker who knows the codes, which is exactly what the adaptive attack of Table 5 measures: this is the residual of part (b), not a violation of (a).

The Subword baseline contrasts: $\nabla_E\mathcal{L} \in \mathbb{R}^{V\times d}$ has $\nabla_{E[t]}\mathcal{L} = \nabla_{E[t]}\ell_t \neq \mathbf{0}$ if and only if $t \in \mathcal{B}$, so $\mathrm{supp}(\nabla_E\mathcal{L}) = \mathcal{B}$ exactly. A single observed nonzero row identifies its token. □

Proof of (b). We use only the definition of mutual information and Pinsker's inequality. Let $\mathcal{B}$ be uniform on $\binom{V}{|\mathcal{B}|}$. By definition of conditional entropy,

$$H(\mathcal{B} \mid g) \;=\; H(\mathcal{B}) - I(\mathcal{B}; g) \;=\; \log_2 \binom{V}{|\mathcal{B}|} \;-\; I(\mathcal{B}; g).$$

This gives the entropy bound. For the per-token bound, fix any predictor $\hat{t} : G \to V$ and write $A_g = \{\mathcal{B} : \hat{t}(g) \in \mathcal{B}\}$. Under the prior, $P_{\mathcal{B}}(A_g) = s/V$ for every g, so

$$\begin{aligned}
\Pr[\hat{t}(g) \in \mathcal{B}] &= \frac{s}{V} + \mathbb{E}_g\big[P_{\mathcal{B}|g}(A_g) - P_{\mathcal{B}}(A_g)\big] \\
&\le \frac{s}{V} + \mathbb{E}_g \sqrt{\tfrac{1}{2}\,\mathrm{KL}(P_{\mathcal{B}|g} \,\|\, P_{\mathcal{B}})} \quad \text{(total variation, then Pinsker)} \\
&\le \frac{s}{V} + \sqrt{\tfrac{1}{2}\,\mathbb{E}_g\,\mathrm{KL}(P_{\mathcal{B}|g} \,\|\, P_{\mathcal{B}})} \;=\; \frac{s}{V} + \sqrt{\tfrac{1}{2}\,I(\mathcal{B}; g)} \quad \text{(Jensen; } I \text{ in nats).}
\end{aligned}$$

If I is measured in bits, the last term reads $\sqrt{I(\mathcal{B}; g)\ln 2/2}$. The keyed encoder is handled by Theorem 2 in the computational model. We do not claim that $I(\mathcal{B}; g_\kappa)$ is small in the information-theoretic sense. Without the key, the bound holds with the $I(\mathcal{B}; g)$ term explicit. The exploitable part of this term is estimated from below by attack success rates, which are TAG at most 0.108 on the undefended rows of Table 2, and by the adaptive attack of Table 5.

For the Subword counterexample: the predictor $\hat{t}(g) = \arg\max_t \|\nabla_{E[t]}\mathcal{L}\|$ achieves $\Pr[\hat{t} \in \mathcal{B}] = 1$ since exactly the rows $t \in \mathcal{B}$ have nonzero gradient. Hence $I(\mathcal{B}; \nabla_E\mathcal{L}) = \log_2 \binom{V}{|\mathcal{B}|}$ and $H(\mathcal{B} \mid \nabla_E\mathcal{L}) = 0$. □

C.3 Auxiliary Lemmas Used in Theorem 1(b)

Lemma 1 (Multiset factorization)**.** *Let $\mathcal{B} = (t_1, \ldots, t_s)$ be an ordered batch. The encoder gradient $g(\mathcal{B}) = \sum_{i=1}^{s} \nabla_{\theta_\epsilon} \ell\big(\epsilon(b(t_i)), y_i\big)$ factors as $g(\mathcal{B}) = G\big(\mathrm{mset}(\mathcal{B})\big)$ where $\mathrm{mset} : V^{|\mathcal{B}|} \to \binom{V}{|\mathcal{B}|}_{\mathrm{multi}}$ is the ordered-to-multiset map, and G is well-defined on multisets of size s.*

Proof. The summation in g is over the index i, but the summand $\nabla_{\theta_\epsilon} \ell\big(\epsilon(b(t_i)), y_i\big)$ is a function only of the pair (t_i, y_i) and of the parameter vector θ_ϵ at which the gradient is evaluated, which is the same for every summand. Hence the sum is invariant under any permutation $\pi \in S_s$ of $(t_1, \ldots, t_s)$, with the corresponding labels permuted, and the induced map on multisets is well-defined. □

Lemma 2 (Indistinguishability under PRP)**.** *Let $\mathcal{B} \sim \mathrm{Unif}\big(\binom{V}{|\mathcal{B}|}\big)$, with codes $\{b(t)\}$ pairwise distinct, Algorithm 1 ensures this. Under the keyed encoder ϵ_κ of Theorem 2, for any PPT distinguisher $\mathcal{D}$,*

$$\big|\Pr[\mathcal{D}(g_\kappa(\mathcal{B})) = 1] - \Pr[\mathcal{D}(g_\kappa(\mathcal{B}')) = 1]\big| \;\le\; \mathrm{negl}(\lambda),$$

for any two label-symmetric batches $\mathcal{B}, \mathcal{B}'$ of size s with equal shape and token-repetition pattern.

Proof. By Theorem 2, distinguishing $g_\kappa(\mathcal{B})$ from $g_\kappa(\mathcal{B}')$ implies a PRP distinguisher with the same advantage. Since the PRP advantage is negligible by assumption, so is the distinguishing advantage on g_κ. □

Use in Theorem 1. Lemma 1, unconditional, collapses the relevant adversary state space from $V^{|\mathcal{B}|}$ ordered tuples to $\binom{V}{|\mathcal{B}|}_{\mathrm{multi}}$ multisets. Lemma 2, conditional on PRP, makes the multiset itself computationally hidden under the keyed encoder. This is a computational statement and does not bound $I(\mathcal{B}; g_\kappa)$.

C.4 Theorem 2: Cryptographic Privacy under PRP

Theorem 6 (Cryptographic Privacy, TR-IND under PRP, restated)**.** *Let $\Pi = \{\pi_\kappa\}_{\kappa \in \{0,1\}^\lambda}$ be a PRP family on $\{0,1\}^n$. Define the keyed encoder $\epsilon_\kappa(t) = \epsilon(\pi_\kappa(b(t)))$. For every PPT adversary $\mathcal{A}$ in the single-challenge TR-IND game whose two batches have equal shape, labels, and token-repetition pattern, there exists a PPT distinguisher $\mathcal{D}$ with*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{TR\text{-}IND}}(\lambda) \;\le\; \mathsf{Adv}_{\mathcal{D}}^{\mathsf{PRP}}(\lambda).$$

No negligible term is needed for this construction.

Proof sketch. A two-step hybrid: H_0 is the real game with π_κ. H_1 replaces π_κ with a truly random permutation ρ. Step $H_0 \to H_1$ is bounded by $\mathsf{Adv}_{\mathcal{D}}^{\mathsf{PRP}}$ via a standard reduction. In H_1 the advantage is exactly zero: a permutation preserves the distinctness of the codes, so the image of any batch of s distinct codes is a uniformly random s-tuple of distinct elements of $\{0,1\}^n$, independent of which batch was chosen. □

Full proof. We define the hybrids and bound the adversary's distinguishing advantage in each transition.

Hybrid H_0, real game. The challenger samples the key $\kappa \xleftarrow{\$} \{0,1\}^\lambda$ before training and trains θ_ϵ under π_κ. It then samples the challenge bit $c \xleftarrow{\$} \{0,1\}$ and receives $\mathcal{A}$-chosen label-symmetric $(\mathcal{B}_0, \mathcal{B}_1)$. Returns $g(\mathcal{B}_c)$ computed using π_κ. $\mathcal{A}$'s advantage in H_0 is the TR-IND advantage.

Hybrid H_1, random permutation. Replace π_κ, in training and at the challenge, with $\rho \xleftarrow{\$} S_{2^n}$, the symmetric group on $\{0,1\}^n$. All other steps unchanged.

Step $H_0 \to H_1$. We construct a PPT PRP distinguisher $\mathcal{D}$ as follows: $\mathcal{D}$ has oracle access to a permutation O on $\{0,1\}^n$, either π_κ or ρ. $\mathcal{D}$ simulates the TR-IND challenger using O in place of π_κ, including the training run, which only queries O on vocabulary codes: samples $c \xleftarrow{\$} \{0,1\}$, requests $\mathcal{A}$'s batches, computes $g(\mathcal{B}_c)$ using O, and runs $\mathcal{A}$ on the result. $\mathcal{D}$ outputs 1 iff $\mathcal{A}$ outputs

$\hat{c} = c$. By construction, $\Pr[\mathcal{D} = 1 \mid O = \pi_\kappa] = \Pr[H_0]$ and $\Pr[\mathcal{D} = 1 \mid O = \rho] = \Pr[H_1]$. Hence $|\Pr[H_0] - \Pr[H_1]| \le \mathsf{Adv}^{\mathsf{PRP}}_{\mathcal{D}}(\lambda)$.

Step H_1 analysis. In H_1, ρ is uniformly random, so $\rho(b(t_1)), \ldots, \rho(b(t_s))$ for $\mathcal{B} = \{t_1, \ldots, t_s\}$ is a uniformly random ordered tuple of $|\mathcal{B}|$ *distinct* elements of $\{0,1\}^n$. The $b(t_i)$ are distinct by Algorithm 1, and ρ is a bijection, so the images $\rho(b(t_i))$ are distinct as well: distinct tokens cannot collide after the PRP, and no birthday term arises. Repeated tokens within a batch repeat their hidden code. By the equal-repetition assumption this pattern is the same for $\mathcal{B}_0$ and $\mathcal{B}_1$, so it carries no information about c.

The joint distribution $(\rho(b(t_i)))_{i=1}^{s}$ is the same for $\mathcal{B}_0$ and $\mathcal{B}_1$ *up to relabeling*: any permutation that maps $\mathcal{B}_0$ to $\mathcal{B}_1$, such a bijection exists by the equal-shape and equal-repetition assumptions, composed with ρ is again a uniformly random permutation, so the image tuple has the same law under $c=0$ and $c=1$. Combined with the label-symmetry of the loss term (the per-token labels match by hypothesis), the gradient $g(\mathcal{B}_c)$ has identical distribution under $c=0$ and $c=1$.

Hence in H_1,

$$\big|\Pr[\hat{c} = c \mid H_1] - 1/2\big| \;=\; 0.$$

Combining. $\mathsf{Adv}^{\mathsf{TR\text{-}IND}}_{\mathcal{A}} = |\Pr[H_0] - 1/2| \le |\Pr[H_0] - \Pr[H_1]| + |\Pr[H_1] - 1/2| \le \mathsf{Adv}^{\mathsf{PRP}}_{\mathcal{D}}(\lambda)$, so the TR-IND advantage is at most the PRP advantage. □

The Subword baseline does not admit any such reduction: π_κ has no place to enter, and the per-token row $\nabla_{E[t]}\mathcal{L}$ makes distinguishability information-theoretically unavoidable. The deployment settings are in Figure 3. The formal claim covers the encoder gradient only, for a single challenge on batches of equal shape, labels, and repetition pattern.

C.5 Theorem 3: Memory $O(1)$ in V

Theorem 7 (Memory, $|\theta_\epsilon| = O(1)$ in V, restated)**.** *At default h=128, d=64, n=64, $|\theta_\epsilon| = 2nh + h + hd + d = 24{,}768$, independent of V. The Subword baseline has $|\theta_E| = V\,d$ parameters. The compression ratio $V\,d/|\theta_\epsilon|$ is monotonically increasing in V and unbounded. The cross-over V_{cross} at which Subword first exceeds Bits is $V_{\text{cross}} = |\theta_\epsilon|/d \approx 387$ at d=64.*

Proof sketch. Counting parameters of the FFN gives a V-independent constant. Counting parameters of the Subword table gives $\Theta(V)$. The ratio is $\Theta(V)$, monotonic in V. □

Full proof. The Bits encoder $\epsilon : \{0,1\}^n \to \mathbb{R}^d$ is a two-layer FFN with hidden width h and ReLU activation on the one-hot input $\mathrm{onehot}(b) \in \{0,1\}^{2n}$:

$$\epsilon(b) = W_2\,\sigma(W_1\,\mathrm{onehot}(b) + \mathbf{u}_1) + \mathbf{u}_2.$$

The parameter count is the sum of layer-wise counts:

$$|\theta_\epsilon| = \underbrace{2nh}_{W_1} + \underbrace{h}_{\mathbf{u}_1} + \underbrace{hd}_{W_2} + \underbrace{d}_{\mathbf{u}_2} = h(2n + d + 1) + d.$$

At the defaults $(n, h, d) = (64, 128, 64)$ this evaluates to $128(128+64+1)+64 = 128 \cdot 193 + 64 = 24{,}704 + 64 = 24{,}768$, which is the count of the implementation, $2n \to 128 \to d$, and the constant quoted in §4. It is $O(1)$ in V.

The Subword table has $|\theta_E| = V\,d$ parameters. The compression ratio

$$R(V) \;\triangleq\; \frac{|\theta_E|}{|\theta_\epsilon|} \;=\; \frac{V\,d}{h(2n + d + 1) + d} \;=\; \Theta(V),$$

strictly increasing in V and unbounded. The crossover V_{cross} at which the Subword table first exceeds the Bits encoder solves $V_{\text{cross}}\,d = h(2n + d + 1) + d$, giving $V_{\text{cross}} = [h(2n + d + 1) + d]/d = 24{,}768/64 = 387$ at d=64. Concrete compression: $R(35{,}000) \approx 90$, $R(54{,}000) \approx 140$, and a measured 277 for Qwen2.5-7B (Table 6). □

C.6 Theorem 4: Expressivity at Width V

Theorem 8 (Expressivity equivalence, restated)**.** *For any subword table $E \in \mathbb{R}^{V\times d}$, there is a two-layer ReLU encoder ϵ with weights $(W_1, \mathbf{u}_1, W_2, \mathbf{u}_2)$ of hidden width $h \geq V$ with $\|\epsilon(b(t)) - E[t]\|_\infty \leq \tau$ for every t and every $\tau > 0$.*

Proof sketch. The bit codes $\{b(t) : t \in V\}$ are V distinct points in $\{0,1\}^n$. By the finite-set memorization lemma for two-layer ReLU networks [72, 71], a width-V ReLU network can interpolate any function on a V-point set. Choosing the target function to be $b(t) \mapsto E[t]$ gives the desired weights. □

Full proof. We construct $(W_1, \mathbf{u}_1, W_2, \mathbf{u}_2)$ explicitly.

Notation. For a bit pattern $b \in \{0,1\}^n$, let $\mathrm{onehot}(b) \in \{0,1\}^{2n}$ be the doubled-coordinate encoding defined by $\mathrm{onehot}(b)_{2i-1} = 1 - b_i$ and $\mathrm{onehot}(b)_{2i} = b_i$ for $i = 1, \ldots, n$. Note $\|\mathrm{onehot}(b)\|_1 = n$ for every b, and

$$\mathrm{onehot}(b)^\top \mathrm{onehot}(b') \;=\; \sum_{i=1}^{n} \mathbf{1}\{b_i = b'_i\} \;=\; n - d_{\mathrm{H}}(b, b'),$$

where d_{H} denotes Hamming distance.

Construction. Index hidden units by $t \in \{1, \ldots, V\}$. For each t, set the t-th row of W_1 to $a_t = \mathrm{onehot}(b(t))^\top$ and the t-th entry of $\mathbf{u}_1$ to $-\beta_t = -(n - 1/2)$, so that

$$a_t^\top \mathrm{onehot}(b(t)) - \beta_t \;=\; n - (n - 1/2) \;=\; 1/2 \;>\; 0,$$

$$a_t^\top \mathrm{onehot}(b(t')) - \beta_t \;=\; n - d_{\mathrm{H}}(b(t), b(t')) - (n - 1/2) \;=\; 1/2 - d_{\mathrm{H}}(b(t), b(t')) \;\leq\; -1/2 \;<\; 0,$$

for any $t' \neq t$, since the codes are pairwise distinct so $d_{\mathrm{H}} \geq 1$. Hence the ReLU activation gives

$$\sigma\big(W_1 \,\mathrm{onehot}(b(t)) + \mathbf{u}_1\big)_{t'} \;=\; \begin{cases} 1/2, & t' = t, \\ 0, & t' \neq t, \end{cases}$$

i.e. the hidden layer activates exactly the t-th unit and produces value $1/2$.

Set the t-th column of W_2 to $2\,E[t] \in \mathbb{R}^d$ and $\mathbf{u}_2 = \mathbf{0}$. Then for every t,

$$\epsilon(b(t)) \;=\; W_2\, \sigma\big(W_1 \,\mathrm{onehot}(b(t)) + \mathbf{u}_1\big) + \mathbf{u}_2 \;=\; 2\,E[t] \cdot \frac{1}{2} \;=\; E[t].$$

Conclusion. The Bits encoder with hidden width $h{=}V$ $\big(W_1 \in \mathbb{R}^{V\times 2n}, W_2 \in \mathbb{R}^{d\times V}\big)$ realizes any Subword table $E \in \mathbb{R}^{V\times d}$ exactly: $\epsilon(b(t)) = E[t]$ for all t, so $\|\epsilon(b(t)) - E[t]\|_\infty = 0 \leq \tau$ for every $\tau > 0$. For $h > V$, the extra hidden units get zero weights and zero bias and do not change the output.

Optimization question. The construction is loose in two senses: (i) width $h{=}V$ is sufficient but not always necessary, and (ii) the proof shows expressivity, not optimizability under SGD. The empirical optimization gap is addressed in the body-width ablation: $h{=}128 \ll V$ already attains 0.866 test accuracy on CIViC where $V{=}5{,}474$, Table 4. □

D Keyed Variant

The TR-IND game, its proof, and the three limits of the keyed guarantee are in Appendix C.4 and §4. This section covers the two practical questions that the game leaves open: what a fixed key reveals through repetition, and how the key is handled.

D.1 Frequency Analysis

A natural concern about Bits-with-key is whether the public token-to-bit map b admits a classical cryptanalytic attack via byte- or bit-frequency statistics. The answer is no, for two reasons.

(i) In federated training the attacker observes gradients, not codes. In Deployment A of Figure 3 the adversary sees the encoder gradient, the public parameters, and the public map b, but never the keyed codes $\pi_\kappa(b(t))$ of individual victim tokens. Deployment B is different: the cloud GPU receives $\pi_\kappa(b(t))$ at every inference, and because a fixed key maps a token to the same code every time, classical frequency analysis across queries applies there. Theorem 2 covers Deployment B for a single challenge only, and a deployment that must hide token frequencies from the cloud needs a per-session key or padding, which we do not evaluate.

(ii) Even granting the adversary perfect bit-frequency information, plaintext recovery is combinatorial. Suppose, contrary to (i), the adversary somehow learns the empirical distribution of bit values in the victim batch. Recovering plaintext tokens then requires (a) inverting π_κ from a frequency profile, which is hard under PRP security, and (b) guessing the ordering of recovered bit codes within each token's 64-bit pattern. The number of candidate orderings is $\binom{64}{w}$ for a code of weight w, which is on the order of 10^{18} for $w = 32$. The combinatorial bag-of-bits reconstruction is therefore harder than a single-character substitution cipher despite the alphabet collapsing to $\{0, 1\}$.

D.2 Key Management

Every privacy number in this paper except the keyed row of Table 5 is measured on the unkeyed encoder. The keyed encoder ϵ_κ of Theorem 2 is an optional extension for deployments that can hold secret material, and its guarantee is the reduction to the PRP. Empirically, an attacker without the key recovers nothing above chance at any sequence length, Table 5. A deployment would handle the key as standard secret-key material: *generation* from a cryptographically secure random source as a 128-bit PRP key or a seed for a key-derivation function. *Storage* inside the trust boundary, on the client or in a KMS, HSM, or TEE, never at the aggregation server. *Distribution* only among the trusted training participants, for example the institutions of a cross-silo federation. *Rotation* by re-drawing the key between rounds, which changes every encoder input, so the encoder must be retrained or its input layer re-mapped to the new codes. *Revocation*, compromise, and hardware failure are handled by the same re-draw, after which the deployment falls back to the unkeyed guarantee of Theorem 1. Keying affects only reconstruction attacks that must map a candidate token to its code. FILM reads only which embedding rows have nonzero gradient and never uses the code map, so under FILM keyed and unkeyed Bits are expected to coincide. The keyed encoder was tested only against the adaptive attack, §7. A post-quantum deployment would use a 256-bit key, and the reduction of Theorem 2 is unchanged. At unkeyed inference, sending $b(t)$ to a server reveals the token exactly as sending the token would, since the codes are public: the unkeyed guarantee of Theorem 1 is about training gradients, and only the keyed deployments of Figure 3b change what a server learns at inference.

E Experimental Setup

E.1 Compute, Software, and Reproducibility

Hardware. Most main-paper experiments run on a single NVIDIA H100, 80 GB, GPU on a rented cloud instance. Longer multi-seed jobs and the adaptive and defense attacks run on an NVIDIA RTX 5090, 32 GB, workstation. The three-seed 1B-token Qwen2.5-7B runs use a rented 8×B200 node with FSDP, four GPUs per arm. The WMT14 runs use a 4-GPU RTX 5090 node. A DGX Spark workstation hosts the federated-learning sweeps. Earlier runs used a Tesla V100, 32 GB.

Software. PyTorch 2.0.1 with CUDA 11.8 for the V100-era runs. PyTorch 2.4.0 with CUDA 12.4 for the H100 runs. PyTorch 2.7 or later with CUDA 12.8 or later for the RTX 5090 and B200 runs. Sacrebleu 2.6.0 for BLEU/chrF/TER scoring. Opacus 1.4.0 for DP-SGD. HuggingFace `transformers` 4.41.0 and `datasets` 2.19.0. `pyedflib` 0.1.36, Sleep-EDF, and `ucimlrepo` 0.0.7, UCI Diabetes 130.

Random seeds. Default seed is 42 for `random`, `numpy.random`, `torch.manual_seed`, and `torch.cuda.manual_seed`. Multi-seed experiments use seeds $\{42, 43, 44, 45, 46\}$, reported as mean $\pm$ std.

Determinism. CUBLAS workspace is fixed, `CUBLAS_WORKSPACE_CONFIG=:4096:8`, and `torch.use_deterministic_algorithms(True)` is enabled where the operator graph supports it. Some operations, for example `torch.bmm` on bf16, fall back to non-deterministic kernels, but seed-fixed run-to-run variance is below ± 0.002 accuracy on all classification tasks.

Code and data. All scripts live under `software/`. Raw outputs under `results/`. Runner scripts and result JSONs are listed in the released repository.

E.2 Encoder Baseline Configurations

Subword. A standard $V \times d$ lookup table with no projection. The tokenizer is `basic_english`, TorchText, for sentiment, BPE, SentencePiece, 32K merges, joint vocab, for MT, GPT-2 BPE for WikiText-103, and dataset-specific subword vocabularies, built with `min_freq`$= 2$, for the rest. d=64 classification, d=512 MT. Initialized $\mathcal{N}(0, 0.02^2)$, no padding-token zeroing.

EmByte. A two-stage byte encoder $b(\cdot) \to \mathbb{R}^{V_b \times h_{\text{byte}}} \to \mathbb{R}^d$ with V_b=256, h_{byte}=128, FFN hidden 128, ReLU, dropout 0.1. The byte embedding $E_b \in \mathbb{R}^{V_b \times h_{\text{byte}}}$ is trained jointly. Default d=64.

Bits. As defined in §3: $b : V \to \{0,1\}^n$, n=64, expected weight 32, sampled by Algorithm 1. One-hot expand to $\{0,1\}^{2n}$. FFN $2n \to h \to d$ with h=512 on text tasks and h=128 on genomic, code, log, signal, and tabular tasks, ReLU, dropout 0.1. The bit map is frozen. Only the FFN trains.

Hash Embeddings [57]. Bucketed embedding with 10,000 shared rows, K=2 hashes per token, and a learned $V \times K$ importance matrix. d=64.

ByT5-small [68]. The official `google/byt5-small` checkpoint, $\approx$ 300M parameters. Encoder-decoder. For SST-2 we use only the encoder + a linear head, 3 epochs, lr $5 \cdot 10^{-5}$.

CHARFORMER [59]. A character-level encoder using the gradient-based subword tokenization of [59]. We use a 4-layer encoder with d=256, GBST kernel $\{1, 2, 3\}$, and a linear classification head.

MegaByte [70]. A multi-scale byte transformer with a 4M-parameter local model and a 6.4M-parameter global model, patch size 8. Used as an autoregressive language model on Devign C-code and PubMedQA contexts.

E.3 Body Architectures

We compare seven bodies in the ablation (Table 4). Body D is the default for the CIViC ablations. The text and genomic accuracy rows of Table 3 use a 2-layer bidirectional LSTM with hidden size 64, Body A serves the code, log, signal, and tabular rows and all attack cells, and SST-2, IMDB, and Yelp→IMDB use the LSTM models of Appendix E.6. Body A is used for all attack cells.

Body A: Mean pool. $\frac{1}{|\mathcal{B}|} \sum_t \epsilon(t) \to \text{Linear}(d, n_{\text{cls}})$. No attention.

Body B: BiLSTM. A single bidirectional LSTM layer with hidden size 32 per direction, 64-dimensional output, over the encoder output, followed by a linear classifier.

Body C: Cross-attention. A learnable query of size $1 \times d$ attends to the encoder output $|\mathcal{B}| \times d$ with a single attention head. The attended vector is fed to a linear classifier. d=64, dropout 0.1.

Body D, ablation default: Cross-attention + MLP residual. Body C, then a 2-layer MLP $d \to 2d \to d$ with residual and LayerNorm. Final linear classifier. The MLP residual contributes $\approx$ 33K extra parameters at d=64.

Bodies E, F, G: transformer. Encoder layers 2 for E, 1 for F, and 4 for G, with 4 heads, d=64, FFN 256, dropout 0.1, sinusoidal positional encoding, `[CLS]` pooled.

E.4 Datasets and Evaluation Metrics

Per-dataset V ranges from 258 to 54,286. All datasets are downloaded once and cached locally. We use the following splits and metrics.

Sentiment analysis. *IMDB* [43]: 25,000 train / 25,000 test, 25% of train held out as validation. *SST-2* [55]: 67,349 / 872 / 1,821 train / val / test. *Yelp polarity*: 560,000 / 38,000 train / test, 10% of train held out. Metric: top-1 accuracy.

Clinical NLP. *PubMedQA* [34], *MedMCQA* [49], *ADE*, *PubMed MeSH*, *Med abstracts*. All loaded via `datasets` with the default train/val/test splits. MedMCQA uses one shared encoder trained jointly on four clinical tasks over a 261-symbol byte vocabulary, so its subword baseline is a byte table. It is scored on the official validation split, since the test labels are not public. Metric: top-1 accuracy.

Domain classification. *Education*, *Finance*, *Legal* from CUAD, *Mental* from Mental Health Status, *Contract*, *ECtHR*. Vocabulary is built per dataset, `min_freq`$= 2$. Metric: top-1 accuracy. Long documents are truncated to ≤ 256 tokens.

Intent classification. *Banking77* [9], 77 intents, *CLINC150* [39], 150 intents, in-scope only, *MASSIVE* [19], 60 intents, English only. Metric: top-1 accuracy.

Genomic Benchmarks. Five tasks from Genomic Benchmarks [26]: human-promoters, human-enhancers, human-OCR, drosophila-enhancers, coding-vs-intergenomic. Each is a binary classification on DNA k-mer-tokenized sequences with $V \approx 4{,}098$. Metric: top-1 accuracy. We additionally include the *Emotion* text-classification task as a non-genomic control row.

Code / Logs / Signal / Tabular. *Devign* [76], binary code-vulnerability detection on C functions, $V{\approx}26{,}491$. *Linux syslog*, top-5 service classification on $\sim$ 20,000 system-log lines from `/var/log/syslog,auth.log,kern.log`, $V{\approx}2{,}942$. *Sleep-EDF EEG* [37], 30-second EEG epochs from 10 subjects, 5-class sleep stage, signal quantized to 64 tokens of 256 levels each, $V{=}258$. *UCI Diabetes 130* [56], binary readmission within 30 days, tabular features serialized to `key=value` tokens, $V{=}715$. Metrics: top-1 accuracy throughout.

High-resource MT. *IWSLT14* de-en, 160K training pairs, validation on a held-out 5% of training, test on dev2010 and tst2010 to tst2012. *WMT14* en-de, 4.5M training pairs, newstest2013 validation, newstest2014 test. Both use a joint BPE vocabulary of 32K. BLEU and privacy for both come from the runs described in Appendix E.7.

Cross-dataset transfer. *Yelp* $\rightarrow$ *IMDB*: train on the first 100K Yelp polarity training reviews, evaluate zero-shot on IMDB test. Vocabulary built from Yelp. OOV tokens map to `[UNK]` for Subword. Metric: top-1 accuracy.

E.5 Preprocessing Details

Tokenization. For Subword: `basic_english` tokenizer for sentiment, BPE for MT, dataset-specific subword vocabulary, `min_freq`$= 2$, for the rest. For Bits: the same tokenizer and vocabulary as Subword, each token id mapped to its random code, Algorithm 1. For EmByte: the token's first eight UTF-8 bytes, $V_b{=}256$, one-hot encoded. The Yelp$\rightarrow$IMDB row builds one vocabulary on Yelp, so unseen IMDB tokens map to `[UNK]` for every encoder. Sleep-EDF and UCI Diabetes use synthetic token sequences, signal quantization and `key=value` respectively, before the byte step.

Sentence-length filtering. IWSLT14 training corpus: drop sentences exceeding 800 bytes. Sentiment: max sequence length 256.

Vocabulary thresholds. `min_freq`$= 2$ for IMDB and Yelp, giving about 22K types on IMDB, and 5 for SST-2, giving 9,113.

Padding and special tokens. `[PAD]` index 0 for all encoders. `[BOS]`/`[EOS]` added for MT. `[CLS]` added only for the transformer bodies E, F, and G. `[UNK]` for out-of-vocabulary tokens under every encoder.

E.6 Training Hyperparameters by Task Family

All hyperparameters below are the same across encoders unless otherwise noted, isolating the encoder as the only experimental factor.

Classification, sentiment, intent, domain, clinical, code/logs/signal/tabular. Adam, lr $5 \cdot 10^{-4}$, β_1=0.9, β_2=0.999, weight decay 0, dropout 0.4 in the body and in the encoder FFN, no learning-rate schedule. The other clinical tasks and the domain, intent, genomic, and emotion tasks train 20 epochs at batch size 64 and max sequence length 256, 128 for intent, and report test accuracy at the best-validation epoch. SST-2 and IMDB use the sentiment LSTMs below, 50 epochs at batch size 512 and 64 respectively. Devign, Sleep-EDF, and Diabetes train 4 epochs at batch size 16, and the shared MedMCQA encoder 4 epochs at batch size 32, scored after the last epoch.

Sentiment LSTMs. *SST-2*: a 2-layer LSTM with embedding 256, hidden size 300, dropout 0.4, Adam at lr $5 \cdot 10^{-4}$, batch size 512, 50 epochs. *IMDB*: a 2-layer LSTM with embedding 64, hidden size 64, dropout 0.4, Adam at lr $5 \cdot 10^{-4}$, batch size 64, 50 epochs, max length 256. *Yelp→IMDB*: an LSTM with embedding 128, hidden size 256, 30 epochs at batch size 32, AdamW at lr $5 \cdot 10^{-4}$ with weight decay 10^{-5}, gradient clipping at 1.0, 500 warm-up steps, 100K Yelp reviews, max length 256, and a Bits encoder of width h=512.

Seeds. Each classification accuracy is the mean over seeds 42 to 46. The data split stays fixed at seed 42, and the seed sets initialization, the Bits code draw, and batch order. Table 8 gives the standard deviations. Linux syslog has one seed, since its data are the system logs of the machine it first ran on. The Yelp→IMDB pretraining checkpoint is the epoch with the best Yelp test accuracy.

Task	S	B	Task	S	B
PubMedQA	0.587±0.012	0.589±0.006	Banking77	0.749±0.019	0.731±0.014
ADE	0.892±0.007	0.892±0.007	CLINC150	0.739±0.008	0.732±0.010
MedMCQA	0.290±0.015	0.285±0.015	MASSIVE	0.761±0.002	0.741±0.005
PubMed MeSH	0.458±0.009	0.462±0.002	GB human-promoters	0.847±0.019	0.800±0.015
Med abstracts	0.547±0.007	0.556±0.012	GB human-enhancers	0.666±0.007	0.635±0.020
IMDB	0.860±0.003	0.869±0.006	GB human-OCR	0.736±0.006	0.713±0.004
SST-2	0.811±0.008	0.814±0.007	GB drosophila-enh.	0.589±0.009	0.561±0.009
Education	0.285±0.011	0.337±0.021	GB coding-vs-intergen	0.903±0.002	0.891±0.006
Finance	0.793±0.009	0.804±0.005	Emotion	0.903±0.009	0.925±0.005
Legal	0.804±0.003	0.803±0.004	Devign code	0.567±0.009	0.546±0.006
Mental	0.528±0.006	0.522±0.005	Sleep-EDF EEG	0.672±0.000	0.672±0.000
Contract	0.935±0.008	0.938±0.002	UCI Diabetes 130	0.884±0.000	0.884±0.000
ECtHR	0.364±0.012	0.371±0.028	Yelp→IMDB 0-shot	0.786±0.008	0.801±0.003

Table 8: Accuracy over seeds 42 to 46, mean ± std, Subword S vs Bits B, the means of Table 3, with B at h=512 on text tasks and h=128 elsewhere. The data split is fixed at seed 42, and the seed sets initialization, the Bits code draw, and batch order. Linux syslog is left out because its data are the system logs of the machine it first ran on.

High-resource MT. See §E.7.

Federated learning. 20 clients, IID uniform split. Client participation $c \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$. 50 communication rounds, 1 local epoch per round, batch size 32, FedAvg weighted by dataset size. Local optimizer Adam at lr $5 \cdot 10^{-4}$. Max sequence length 256.

E.7 MT Protocol

Metrics. BLEU computed by sacrebleu with the default `13a` tokenizer, on `newstest2014` for WMT14 and on the concatenation of dev2010 and tst2010 to tst2012 for IWSLT14.

Models and training. The IWSLT14 BLEU and FILM numbers come from a pair of runs, $6{+}6$ layers, $d{=}512$, 4 heads, FFN 1024, dropout 0.2, joint BPE 32K, Adam $5{\cdot}10^{-4}$ with 4000 warm-up steps, label smoothing 0.1, early stopping on validation loss with patience 5, batches of $4{,}096$ tokens with 2 accumulation steps, greedy decoding, one shared input embedding for encoder and decoder and an untied output layer. Subword has 16.4M embedding and 64.6M total parameters, Bits 82.6K and 48.3M. FILM is the attack of Appendix E.8, which reads the nonzero rows of the embedding gradient, applied to the shared input embedding and scored against the set of source and target tokens in the batch. Subword stops at epoch 16 with its best at 11. Bits, run with the epoch cap raised from 40 to 100, stops at 45 with its best at 40, the checkpoint reported. Subword reaches 30.0 BLEU and Bits 29.7. At batch size 16 FILM is 0.962 for Subword, and for Bits, which has no per-token rows, it falls to a guessing floor of 0.001. The WMT14 BLEU and FILM numbers come from a second pair of runs on the 3.87M training pairs left after filtering, Transformer base with $6{+}6$ layers, $d{=}512$, 8 heads, FFN 2048, dropout 0.1, joint BPE 32K, Adam $7{\cdot}10^{-4}$ with 4000 warm-up steps, label smoothing 0.1, 4 GPUs with $8{,}192$ tokens each, greedy decoding, and the same shared input embedding, except that the Bits encoder output is rescaled by a fixed constant at initialization to match the variance of a subword table. Subword has 16.4M embedding and 77.2M total parameters, Bits 82.6K and 60.9M. Training resumes from saved weights with a fresh optimizer twice: epochs 9 to 16 run on one GPU with 4 accumulation steps, the same tokens per step, and later epochs on 4 GPUs again. Early stopping uses patience 3 on validation loss. Subword converges, best at epoch 15, and stops at 18. Bits is still improving when the 30-epoch cap stops it, so its gap is an upper bound. A further resume from epoch 30 was unstable and did not improve on it. Subword reaches 23.1 BLEU and Bits 22.0. FILM is 0.76 for Subword at batch size 16 and 0.98 at batch size 1, and for Bits it falls to a guessing floor of 0.000 at both. TAG and iDLG were not run on MT.

E.8 Privacy Attack Details

All attacks are run on a small classifier with the mean-pool Body A of Table 4, $d{=}64$, so that the embedding is the only varying factor. Each attack is repeated on 12 random batches of size $|\mathcal{B}|{=}16$, except where noted, and the average is reported.

FILM [28]. Recovers the bag of input tokens from the nonzero rows of the embedding gradient. On Bits, which has no per-token rows, the attack has nothing to read and returns a guess, which is the guessing floor we report. Configuration: 1000 iterations, lr 0.01, beam-search width 10, 7K negative subwords mixed with the gold subwords. Metric: token F1.

TAG [16]. Gradient-matching inversion for transformer language models. Configuration: 300 iterations, lr 0.05, ℓ_2 + ℓ_1 matching loss with weight $\alpha{=}0.01$ on the ℓ_1 term. Metric: token F1.

FL-LLG [62]. Reads the sign of the final-layer bias gradient to predict the per-sample class label. Metric: ASR, closer to chance $1/n_{\text{classes}}$ is better, where chance $= 0.5$ for SST-2 / Yelp / IMDB binary tasks, 0.25 for MedMCQA, 0.333 for PubMedQA.

LAMP [2]. Token-recovery attack that combines gradient matching with an LM prior. Configuration: GPT-2, `gpt2`, 124M, prior, 2000 iterations, lr 0.01, prior weight 0.1. We use the official Balunović implementation, SST-2 defenses panel. Metric: token F1.

Inverting Gradients [25]. Optimization-based input recovery. Configuration: 300 iterations of Adam at lr 0.1 with cosine-similarity loss. Metric: token F1.

iDLG [75]. Improved DLG that recovers the class label from the sign of the cross-entropy gradient at the final layer, then runs DLG with the recovered label fixed. Configuration: 100 iterations, lr 0.1. Metric: token F1.

Attacks not reported. Decepticons [20], Property Inference [23], BadNets [27], Model Inversion [22], and LiRA [7] are not part of this evaluation.

E.9 Defense Configurations

DP-SGD [1]. Implemented via Opacus 1.4.0. Configuration: δ=10^{-5}, target $\varepsilon \in \{\infty, 16, 8, 4, 1\}$, per-sample gradient clip C=1.0, microbatch size 1. Privacy accountant: RDP. ε=∞ denotes training without noise or clipping. Batch size 16.

Top-K Gradient Compression [42]. Mask all but the top-K entries, by absolute value, of every gradient tensor before transmission. $K \in \{1.0, 0.1, 0.01\}$, that is 100%, 10%, 1% retained. Implemented as a hook on the optimizer step.

InstaHide [33]. Linearly mix K training samples with random sign flips before each forward pass. $K \in \{2, 4, 8\}$. Mixing coefficients are sampled from the uniform simplex.

Federated DP [64]. Server-side Gaussian noise injection on the aggregated gradient with δ=10^{-5} and $\varepsilon \in \{8, 4, 1\}$, computed assuming 20 clients with full participation.

Other. Secure Aggregation [6] and PATE [50] are orthogonal stack components that compose with Bits. We discuss but do not benchmark them.

E.10 Memory and Time Measurement Protocol

Parameters in K and Total MB. *Parameters* is the `numel()` count over all `nn.Module` parameters, encoder + body + head. *Total MB* in Table 3 is parameters $\times 4$ bytes / 1,048,576, that is MiB, while the memory accounting of Appendix B uses decimal MB. The bit map is excluded from Total MB since it is a frozen, non-parameter lookup table. Figure 2a and the MASSIVE on-device memory in §5 include the bit-packed code table.

Per-step latency. Wall-clock time of one full forward+backward+optimizer step under a `torch.cuda.synchronize()` fence, averaged over 100 warm steps after 10 discarded warm-up steps. Memory use is read from `torch.cuda.max_memory_allocated()`.

FedAvg client step. Single-client step on SST-2 with batch 32, max sequence length 256, lr $5{\cdot}10^{-4}$, dropout 0.4. Reported in milliseconds.

MT step. 4-layer encoder-decoder, d=128, FFN 512, 4 heads, V=8K, batch 16, max sequence length 128. Reported in milliseconds.

F Additional Results

This section reports results beyond the main tables, grouped by the claim they support. Unless stated otherwise, classification runs use CIViC, Body D, V=5,474, test set of 709 examples, and all privacy attacks are run against the *unkeyed* encoder under the Kerckhoffs threat model: the attacker is given the codebook, the seed, the encoder weights, and the body weights. The keyed variant of Theorem 2 is evaluated only under the adaptive attack, Table 5.

F.1 Privacy

F.1.1 Adaptive Full-Gradient Attack at Batch Size One

The attacker optimizes a soft token sequence $A \in \mathbb{R}^{L \times V}$ so that the gradient it induces matches the full observed gradient of a single sequence, batch size 1. For Bits the candidate embedding is routed through the real encoder, $\epsilon(A\,C)$ with C the known codebook, so the attack targets the shared-FFN gradient rather than embedding rows. Setup: V=1,000 proxy vocabulary, d=64 mean-pool body, Adam with 600 steps at learning rate 0.1, 50 target sequences per point, two metrics: per-position exact recovery, the fraction of positions whose recovered token equals the true token, chance $1/V$=0.001, and token-set recovery $|\hat{\mathcal{B}} \cap \mathcal{B}|/L$, chance L/V, since a random set of L tokens already hits L/V of the targets. The keyed row gives the attacker a codebook drawn with a different key, which is what an attacker without κ can do under Theorem 2.

Sequence length L	**1**	**2**	**4**	**8**	**16**	**32**	**64**	**128**	**256**	**512**	**1024**
Token-set recovery (chance $\min(1, L/V)$*)*											
Chance	0.001	0.002	0.004	0.008	0.016	0.032	0.064	0.128	0.256	0.512	1.000
Standard table	1.000	1.000	**0.900**	**0.816**	**0.768**	0.714	0.708	0.813	0.943	0.985	0.981
Bits	1.000	1.000	0.995	0.983	0.881	**0.613**	**0.391**	**0.320**	**0.314**	**0.339**	**0.393**
Bits, keyed	0.000	0.000	0.000	0.003	0.020	0.031	0.059	0.119	0.182	0.258	0.345

Table 9: Adaptive attack, token-set recovery, batch size 1, 50 sequences per cell. Per-position recovery is in Table 5. Token-set recovery is uninformative once L approaches V, since the chance set already covers the targets. All cells use 50 sequences each.

Batch size, at L=8	**1**	**2**	**4**	**8**	**16**	**32**
Standard table	0.143	0.128	0.055	0.053	0.016	0.007
Bits	0.143	0.112	0.065	0.013	0.010	0.002

Table 10: Adaptive attack versus batch size at sequence length 8. Recovery falls with aggregation under both encoders. Bits is comparable to the standard table at small batches and lower from batch 8 on. This is a separate run with 150 attacker iterations and 48 targets at batch 1, falling to 10 from batch 8, so batch 1 differs from Table 5.

F.1.2 Attack Suite at Sequence Length 1 and 256

Same proxy setup as Appendix F.1.1. Each cell gives recovery at L=1 then at L=256.

Attack	**Standard table**	Bits
DLG, full-gradient matching	$1.00 \to 0.006$	$1.00 \to 0.002$
iDLG, known labels	$1.00 \to 0.006$	$1.00 \to 0.002$
Inverting Gradients	$1.00 \to 0.006$	$1.00 \to 0.002$
TAG	$1.00 \to 0.005$	$1.00 \to 0.002$
LAMP, language prior	$1.00 \to 0.008$	$1.00 \to 0.002$
FILM, row support	$1.00 \to 0.88$	$0.000 \to 0.23$

Table 11: Six attacks at batch size 1, unkeyed. An attacker without the key cannot map candidate embeddings to codes, so the keyed encoder is expected to sit at chance under the first five. This is not measured, §7. Bits is at or below the standard table in every cell. The FILM residual of 0.23 for Bits is a small-vocabulary artifact of the V=1,000 proxy. At $V \approx 152$K, Table 16, FILM recovery for Bits is at chance.

F.1.3 Federated Attacks with Side Information

Run on CIViC with Body D. *Malicious server* follows the trap-weight construction [21] in which the server plants weights that route individual tokens to separate neurons. *Known text* gives the attacker a fixed fraction of the private sequence. *Multi-round* lets the attacker observe the client's gradient across 1 to 8 federated rounds and aggregate the recovered sets.

F.2 Ablations

Table 13 gives the subword-table counterpart of the body block of Table 4: same bodies, same CIViC split, 4 epochs, seed 42.

F.2.1 Encoding Scheme Ablation

Every row except Standard uses the same shared two-layer FFN, n=256, h=256. Only the token-to-code map changes. Random draws each code uniformly, Algorithm 1. MD5 hashes the seeded token index and chains the digest, h_{k+1}=MD5(h_k), until n bits are available, so no codebook is stored. Surface encodes the token's characters. Learned trains the code table jointly with the FFN. Gray and Hierarchical assign consecutive Gray codes and a tree-structured prefix code to token ids, which leaves all but $\lceil \log_2 V \rceil$=13 of the 256 bit positions constant across the vocabulary. Accuracy is mean $\pm$ std over five seeds, 42 to 46. FILM and TAG are token F1 without a std, lower is better.

Attack	Setting	Standard table	Bits
Malicious server	trap reconstruction	0.072	0.003
Known text	0% known	1.000	0.016
	25%	1.000	0.013
	50%	1.000	0.006
	75%	1.000	0.001
Multi-round	1 round	1.000	0.013
	2	1.000	0.013
	4	1.000	0.014
	8	1.000	0.014

Table 12: Federated attacks with side information on CIViC, body D.

Body, subword table	Acc	Par.	FILM↓	TAG↓
A	0.430	351	0.817	1.000
C	0.860	367	0.150	1.000
D	0.877	401	0.174	1.000
E	0.863	451	0.461	1.000
F	0.866	401	0.134	1.000
G	0.872	551	0.454	1.000

Table 13: Body ablation with the subword table on CIViC. Par. is the total parameter count in K.

F.2.2 Code-Length Sweep

Body D, h=256, three seeds. The injective threshold is $\lceil \log_2 V \rceil$=13 bits. At n=8, at least 95% of tokens share a code with another token.

n	Test acc.	n	Test acc.
8	0.595 ± 0.076	256	0.858 ± 0.007
16	0.736 ± 0.035	512	0.869 ± 0.024
32	0.815 ± 0.051	1024	0.862 ± 0.005
64	0.804 ± 0.034	2048	0.876 ± 0.015
128	0.833 ± 0.017		

Table 14: Accuracy versus code length n on CIViC, body D, h=256, three seeds. The main experiments use n=64, fixed before this sweep.

F.3 Embedding Geometry

Table 15 reports cosine similarities between Bits-derived subword embeddings on IMDB.

F.4 Scaling to Billion-Parameter Language Models

We train decoder language models from scratch with the Qwen2.5-1.5B and Qwen2.5-7B architectures on C4, swapping only the input embedding, standard table versus a Bits encoder with n=256, h=512 that takes the raw bit vector as input, $n \to h \to d$, so 1.97M parameters at 7B and 0.92M at 1.5B, and keeping everything else identical: bf16, AdamW at learning rate $3 \cdot 10^{-4}$, 2% warmup, cosine decay, gradient clipping at 1.0, 32,768 tokens per step, seed 42 for the single-seed 100M-token runs reported in §5 and seeds 42, 43, 44 for the 1B-token 7B runs of Table 6. Both arms are evaluated on the same held-out C4 validation set of 2.048M tokens.

The per-checkpoint three-seed results are in Table 6 in the main text. The training curves follow. All 66 bf16 checkpoints of the three-seed 7B runs, ten per run at every 100M tokens plus the final step, both arms, the training and evaluation logs, the held-out set, and the scripts are released at `https://huggingface.co/bit2vec/bitsseal` under `qwen2.5-7b_c4_1B/`.

	good	**great**	**funny**	**bad**	**worse**	**boring**
good	1.00	**0.63**	**0.49**	-0.58	-0.61	-0.58
great	0.63	1.00	**0.40**	-0.53	-0.33	-0.38
funny	0.49	0.40	1.00	-0.72	-0.61	-0.60
bad	-0.58	-0.53	-0.72	1.00	**0.72**	**0.85**
worse	-0.61	-0.33	-0.61	0.72	1.00	**0.88**
boring	-0.58	-0.38	-0.60	0.85	0.88	1.00

Table 15: **Cosine similarity between Bits-derived subword embeddings on IMDB sentiment.** Despite the random bit code per token, the trained Bits FFN learns embedding geometry consistent with task semantics: positive-positive pairs, good / great / funny, have positive similarity in $[0.40, 0.63]$, negative-negative pairs, bad / worse / boring, have positive similarity in $[0.72, 0.88]$, and positive-negative pairs are uniformly negative, at most -0.33. The structure refutes a "Bits is just a hash" interpretation: the FFN maps the fixed codes into a semantically meaningful manifold during training.

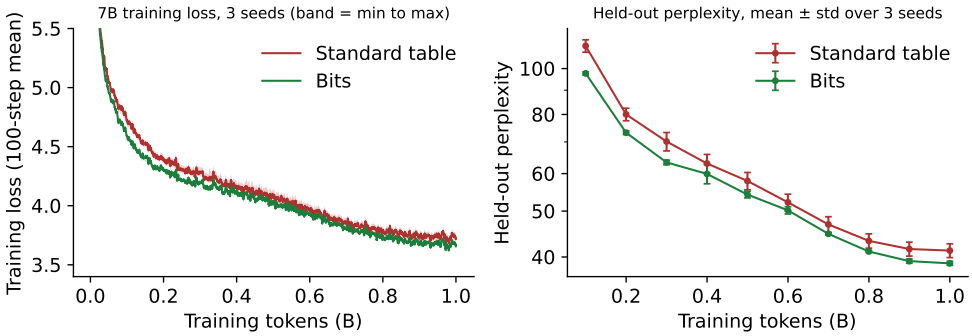

Figure 4: Qwen2.5-7B over 1B tokens, three seeds per arm. Left: training loss, 100-step mean, band is the min to max over seeds. Right: held-out perplexity at each 100M-token checkpoint, mean $\pm$ std.

G Ethical Considerations

We use only public datasets under their licenses. The work targets privacy-preserving FL, on-device inference, and transfer across vocabularies, with no risks beyond standard NLP practice. Large language models assisted with editing the text, literature search, and writing and running experiment scripts. All methods, experiments, results, and claims were designed, checked, and are the responsibility of the author.

Attack	**Model**	**Standard** $L{=}4 \to 256$	Bits $L{=}4 \to 256$
FILM	Qwen2.5-1.5B	$0.75 \to 0.996$	$0.000 \to 0.0012$
TAG	Qwen2.5-1.5B	$0.75 \to 0.996$	$0.000 \to 0.0012$
FILM	Qwen2.5-7B	$0.75 \to 0.996$	$0.000 \to 0.0013$
TAG	Qwen2.5-7B	$0.75 \to 0.996$	$0.000 \to 0.0013$

Table 16: Set-recovery attacks on the trained models, batch size 1, 30 target sequences per point, $V{=}152{,}064$. FILM and TAG agree to three digits because both read the same set of nonzero embedding rows. Chance is L/V, from $2.6{\cdot}10^{-5}$ at $L{=}4$ to $1.7{\cdot}10^{-3}$ at $L{=}256$.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract states (i) up to $167\times$ lower TAG token recovery, Table 3, (ii) up to 99% fewer embedding parameters, Table 3, and (iii) lower held-out perplexity than an embedding table for from-scratch 1.5B and 7B models, Table 6, each supported by the experiments.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The Limitations section, §7, states that Bits closes only the embedding-gradient channel and gives no differential-privacy guarantee, that body gradients, outputs, and losses still leak and need DP-SGD and the language models keep a per-token output layer, that a code-aware attacker still recovers an isolated token, that the keyed encoder was tested only against the adaptive attack, that DAGER and SPEAR were not run, that the claims cover classification, translation, and language modeling but not regression, and that the 7B runs cover 1B tokens, far below compute-optimal.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions, for example independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally. The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g. if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best

judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete, and correct, proof?

   Answer: [Yes]

   Justification: Theorems 1-4 in §4 state their assumptions in full, and Appendix C gives a proof sketch and a complete proof for each, including the uniform-prior and PRP assumptions.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper, regardless of whether the code and data are provided or not,?

   Answer: [Yes]

   Justification: We specify the bits-encoder architecture, subword tokenizer, bit tokenizer, embedding matrix dimensions, aggregation, FFN, and list datasets, splits, model sizes, optimizer, learning rates, batch sizes, training schedules, and evaluation protocols for all six encoder baselines and four defenses and the proposed method.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model, for example in the case of a large language model, releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

(c) If the contribution is a new model, for example a large language model, then there should either be a way to access this model for reproducing the results or a way to reproduce the model, for example with an open-source dataset or instructions for how to construct the dataset.

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way, for example to registered users, but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Code and scripts are released at `https://github.com/bit2vec/bitsseal`, PyPI package `bitsseal`. The from-scratch 7B checkpoints are linked from the same repository. Datasets are public, Appendix E.4.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines, `https://neurips.cc/public/guides/CodeSubmissionPolicy`, for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution, for example for a new open-source benchmark.
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines, `https://neurips.cc/public/guides/CodeSubmissionPolicy`, for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions, if applicable.
   - Providing as much information as possible in supplemental material, appended to the paper, is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details, for example data splits, hyperparameters, how they were chosen, type of optimizer, necessary to understand the results?

   Answer: [Yes]

   Justification: §5 and Appendix E.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: Accuracies in Table 3 are means over seeds 42 to 46 with the data split fixed, and their standard deviations are in Table 8. Linux syslog and the attack values in Tables 2 and 3 use seed 42, except the Bits LLG value on SST-2 in Table 2, a mean over five seeds. MT BLEU is a single run per arm. The code-map block of Table 4 reports mean $\pm$ std over five seeds, Table 6 over three seeds, and the appendix tables state their seed counts.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated, for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions.
- The method for calculating the error bars should be explained, closed form formula, call to a library function, bootstrap, etc.,
- The assumptions made should be given, for example Normally distributed errors.
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range, for example negative error rates.
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources, type of compute workers, memory, time of execution, needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix E.1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper, for example preliminary or failed experiments that didn't make it into the paper.

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Appendix G

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.

- The authors should make sure to preserve anonymity, for example if there is a special consideration due to laws or regulations in their jurisdiction.

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Appendix G

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses, for example disinformation, generating fake profiles, surveillance, fairness considerations, for example deployment of technologies that could make decisions that unfairly impact specific groups, privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from, intentional or unintentional, misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies, for example gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML.

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse, for example pre-trained language models, image generators, or scraped datasets,?

    Answer: [N/A]

    Justification: The released encoders are small classifiers and from-scratch language models trained on public C4 text, which pose no misuse risk beyond that of the public data.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets, for example code, data, models, used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

Justification: Appendix G

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license, for example CC-BY 4.0, should be included for each asset.
- For scraped data from a particular source, for example website, the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset, if it has changed, should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: README and model card in the released repositories.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets, if applicable. You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation, if any,?

    Answer: [N/A]

    Justification: No crowdsourcing, not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board, IRB, approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board, IRB, approvals, or an equivalent approval/review based on the requirements of your country or institution, were obtained?

Answer: [N/A]

Justification: Not applicable.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval, or equivalent, may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity, if applicable, such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: Large language models assisted with editing the text, literature search, and writing and running experiment scripts. They are not a component of the method, and all methods, experiments, results, and claims are the responsibility of the author. See Appendix G.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.