# Scaling Encodings to Larger Alphabets for Peptide-Based Data Storage

Nils Hoppe[1], Gerd Specht[2], and Lena Wiese[1,2]

[1] Institute of Computer Science, Goethe-University Frankfurt, Germany
[2] Fraunhofer Institute for Toxicology and Experimental Medicine (ITEM), Hanover, Germany

**Abstract.** Peptide-based data storage is a promising alternative to conventional storage technologies, offering high density and longevity. However, the storage channel introduces insertion, deletion, and substitution errors, and must avoid problematic secondary structures during synthesis. This paper explores error-correcting coding techniques for peptide-based storage through simulation. We developed and tested two models: an LDPC-based model using simple and composite mappings, and a HEDGES-based model concatenated with Reed-Solomon codes. Our simulations demonstrate that the HEDGES-based model effectively corrects all types of indel errors at rates up to 2%, achieving code rates between 0.27 and 0.53. The LDPC-based model with a code rate of 0.5 reduces homopolymers but only corrects substitution errors. Our findings suggest that HEDGES codes are applicable to peptide-based data storage systems and outperform LDPC-based approaches for this channel.

**Keywords:** Peptide-based data storage · Error-correcting codes · HEDGES · LDPC · DNA storage

## 1 Introduction

The exponential growth of global data is predicted to reach 150 zettabytes by 2025 [13], which necessitates innovative storage solutions. Conventional technologies such as HDDs and flash storage offer fast access, but limited lifespans of 15–30 years and significant energy consumption [3]. DNA-based data storage has emerged as a promising alternative, offering theoretical densities of 460 exabytes per gram and potential longevity of millions of years [6].

However, DNA storage faces significant challenges: high synthesis costs (millions of dollars per terabyte), slow throughput, and complex error patterns including insertions, deletions, and substitutions (indels) [5, 7]. Additionally, biochemical constraints such as avoiding homopolymers and maintaining appropriate GC content must be considered during encoding [8].

Peptide-based data storage extends this concept to amino acid sequences. Peptides offer a larger alphabet (20+ amino acids vs. 4 nucleotides), potentially higher information density (25 exabytes per gram) [10], and the possibility of

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29912-3_28

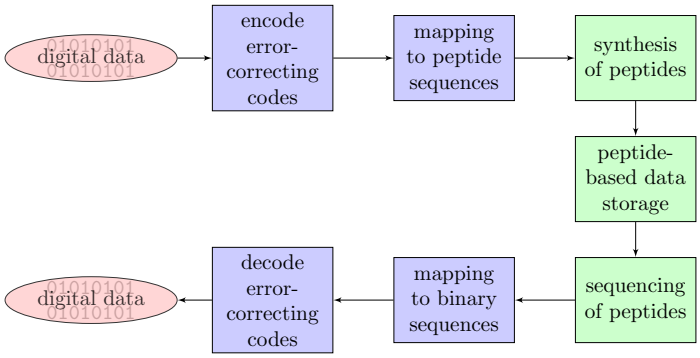


**Fig. 1.** Overview of a Peptide-based data storage pipeline. The red boxes symbolize the start- and endpoints of the encoding and decoding process. The blue fields represent the encoding and decoding steps that are discussed in this paper.

using inexpensive commodity chemicals [4]. Recent work has demonstrated successful peptide synthesis and sequencing for data storage using various coding schemes [11, 14, 15].

Despite these advances, the error-correcting capabilities of different coding approaches for peptide-based storage remain underexplored. The channel capacity for channels with multiple insertions or deletions is unknown [9], making experimental evaluation essential.

We present two simulation models for peptide-based data storage: (1) an LDPC-based model with simple and composite amino acid mappings, and (2) a HEDGES-based model concatenated with Reed-Solomon codes. The applicability of these DNA storage encoding schemes to peptide-based systems with larger alphabets has not been empirically validated. Our contribution is threefold: we provide the first implementation and evaluation of HEDGES codes over an 8-symbol peptide alphabet, we quantify the code rate-reliability trade-off across HEDGES patterns under realistic error conditions, and we demonstrate that composite amino acid mapping, despite its theoretical density advantages, is unsuitable for noisy channels due to error amplification. These results establish a baseline for coding design in peptide-based storage systems.

## 2 Background

### 2.1 Peptide-Based Storage Channel

The peptide storage process (Fig. 1) involves: (1) encoding binary data with error-correcting codes, (2) mapping to amino acid sequences, (3) peptide synthesis, (4) storage, (5) sequencing via mass spectrometry, (6) mapping back to binary, and (7) decoding.

The channel introduces three error types: *insertions* (additional amino acids), *deletions* (lost amino acids), and *substitutions* (replaced amino acids). With an 8-symbol alphabet where each symbol represents 3 bits, a single deletion removes

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29912-3_28

3 bits from the decoded sequence, complicating error correction compared to conventional bit-flip channels.

### 2.2 Mapping Strategies

We consider two mapping approaches. *Simple mapping* assigns each amino acid to a fixed-length bit string (see below). *Composite mapping* [2, 15] uses mixtures of amino acids as symbols; with 20 amino acids, $\binom{20}{3} = 1140$ composite symbols of length 3 exist, allowing 10-bit encoding per symbol.

$$\texttt{000} \mapsto S,\ \texttt{001} \mapsto T,\ \texttt{010} \mapsto E,\ \texttt{011} \mapsto Y$$
$$\texttt{100} \mapsto A,\ \texttt{101} \mapsto V,\ \texttt{110} \mapsto L,\ \texttt{111} \mapsto F.$$

Homopolymers (repeated identical amino acids) increase synthesis errors [1]. Longer bit sequences per symbol naturally reduce homopolymer probability to $(1/2^l)^2$ for runs of length 2, where $l$ is the bit sequence length [15].

## 3 Methods

### 3.1 LDPC-Based Model

Our first model uses a regular Low-Density Parity-Check (LDPC) code as the outer code, following successful applications in DNA storage [11]. LDPC codes are linear block codes with sparse parity-check matrices. Decoding uses belief propagation, an iterative message-passing algorithm: in each iteration, variable nodes (representing coded bits) and check nodes (representing parity constraints) exchange probability estimates about bit values. Over successive iterations, these estimates converge, allowing the decoder to identify and correct erroneous bits. The sparsity of the parity-check matrix keeps the number of messages per iteration low, making decoding efficient even for long codewords. We use code rate $r \approx 0.5$ with codeword lengths of 100–1000 bits.

The inner code consists of the mapping functions (simple or composite) that translate between bit sequences and amino acids. While these mappings lack error-correcting properties, they enable evaluation of homopolymer avoidance and provide a baseline for substitution-only error correction.

### 3.2 HEDGES-Based Model

Our second model concatenates a Reed-Solomon RS(255,223) outer code with the HEDGES (Hash Encoded, Decoded with Greedy Exhaustive Search) inner code [12]. The RS code corrects up to 16 symbol errors per block with code rate 0.8745. HEDGES is a convolutional code that generates a pseudorandom hash value which is computed from the strand ID, the current bit index, and several

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29912-3_28

preceding message bits for the encoding process. The message bits shift this hash value to produce an output symbol:

$$C_i = F(S_i, I_i, V_i) + v_i \mod 8 \quad (1)$$

where $C_i \in \{0, \ldots, 7\}$ corresponds to amino acids $\{\mathrm{S}, \mathrm{T}, \mathrm{E}, \mathrm{Y}, \mathrm{A}, \mathrm{V}, \mathrm{L}, \mathrm{F}\}$, $F$ is the hash function, and $v_i$ represents input bits determined by the *pattern* parameter.

Because the hash depends on the strand ID, bit position, and previous bits, any single decoding error (substitution, deletion or insertion) corrupts all subsequent hash predictions. During decoding, HEDGES exploits this property: it maintains a heap of candidate bit sequences scored by how well their predicted symbols match the received sequence. Correct hypotheses accumulate rewards, while incorrect ones rapidly accumulate penalties from hash mismatches and are pruned. This A* search process either recovers the original bits, converting any indels back into synchronization, or signals failure, which the outer RS code treats as an erasure. The pattern controls the code rate: pattern $[1, 1, \ldots]$ yields rate 1/3 (one amino acid per bit), while $[3, 2, 1, 2]$ yields rate 0.66.

### 3.3 Error Simulation

We simulate independent insertions, deletions, and substitutions at configurable rates. Peptide sequences are stored in FASTA format with metadata. For composite mapping, substitutions replace entire composite symbols. We tested error rates from 0.4% to 2%.

## 4 Results

### 4.1 LDPC-Based Model

Table 1 shows results for the simple mapping (16-symbol alphabet) with 1% substitution errors. Longer LDPC codewords improve reliability, achieving 100% success for codewords $\geq 250$ bits on small files. At 2% error rate, performance degrades significantly, with the 591-byte file achieving only 80% success even with 1000-bit codewords.

**Table 1.** LDPC-based model performance (100 runs, $p_{sub} = 0.01$).

| LDPC Codeword Length | 100 | 250 | 500 | 1000 |
|---|---|---|---|---|
| Success (%) - 47 bytes | 64 | 100 | 100 | 100 |
| Success (%) - 591 bytes | 0 | 82 | 99 | 100 |

Composite mapping performed poorly, achieving only 2% success on the 591-byte file at 1% error rate with 1000-bit codewords. This is because a single composite symbol error causes up to 7 bit flips, overwhelming the outer decoder.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29912-3_28

Regarding homopolymers, simple mapping produced only short runs (length $\leq 2$) except for padding-related sequences. Composite mapping eliminated homopolymers entirely.

### 4.2 HEDGES-Based Model

Table 2 shows results for isolated error types on the 591-byte file. All three patterns achieved 100% success at 1% error rates for each error type individually, with decoding times under 70 seconds.

Table 3 presents results for combined error sources, with one error type at 1% and others at 0.4%. Patterns 1 and 2 maintained $\geq 98\%$ success, while pattern 3 achieved $\geq 90\%$ success across all configurations.

**Table 2.** HEDGES-based model: isolated errors (100 runs, 591 bytes).

| Error Type (1%) | Pattern 1 ($r = 0.33$) | Pattern 2 ($r = 0.53$) | Pattern 3 ($r = 0.66$) |
|---|---|---|---|
| Insertion only | 100%, 67.7s | 100%, 55.4s | 100%, 52.2s |
| Deletion only | 100%, 54.3s | 100%, 46.0s | 100%, 63.8s |
| Substitution only | 100%, 51.3s | 100%, 41.5s | 100%, 44.8s |
| Peptide length | 32 AA | 18 AA | 16 AA |

The overall code rates (including outer RS code) are 0.27, 0.43, and 0.53 for patterns 1, 2, and 3 respectively. This translates to approximately 0.8, 1.3, and 1.6 bits utilized per amino acid out of the theoretical maximum of 3 bits.

Homopolymers in the HEDGES model occurred but remained manageable, with maximum lengths of 6 for only two amino acids and most staying below 4.

## 5 Discussion

Our results suggest that the HEDGES-based model offers notable advantages over the LDPC-based approach for peptide-based data storage. While the LDPC-based model only corrects substitution errors, the HEDGES-based model appears capable of handling all indel error types at realistic rates ($\leq 2\%$).

Regarding mapping strategies, our findings indicate that composite mapping, despite its density advantages, may be unsuitable for noisy channels due to error amplification. Potential mitigation strategies include replacing the bit-level LDPC code with a symbol-level Reed-Solomon code, which would treat each composite symbol error as a single correctable unit regardless of internal bit damage. However, neither symbol-level codes nor bit interleaving address synchronization loss from indel errors. In contrast, simple mapping helps prevent homopolymers while maintaining compatibility with outer codes. Notably,

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29912-3_28

**Table 3.** HEDGES-based model: combined errors (100 runs, 591 bytes).

| Error Configuration | Pattern 1 | Pattern 2 | Pattern 3 |
|---|---|---|---|
| $p_{ins} = 1\%, p_{del} = p_{sub} = 0.4\%$ | 99% | 100% | 93% |
| $p_{del} = 1\%, p_{ins} = p_{sub} = 0.4\%$ | 98% | 99% | 95% |
| $p_{sub} = 1\%, p_{ins} = p_{del} = 0.4\%$ | 100% | 100% | 90% |

HEDGES codes, originally designed for DNA storage with 4 nucleotides, appear to extend well to the 8-symbol peptide alphabet. A trade-off between code rate and error correction capability can be observed in our results: pattern 1 (r = 0.33) tends to offer higher reliability, while pattern 3 (r = 0.66) maximizes information density at somewhat reduced reliability.

The HEDGES-based model also shows decoding times approximately 10× faster than the LDPC-based model for comparable configurations. Specifically, the A* search complexity per strand is primarily driven by error rate rather than file size, as higher error rates increase heap branching up to the configurable $H_{\text{limit}}$ budget. This is reflected in Table 2, where decoding times vary more across error types than across patterns despite differing strand counts. For practical deployments at scale, reducing per-strand decoding time through optimized implementations or tighter $H_{\text{limit}}$ bounds would be necessary.

Regarding scalability, the HEDGES-RS architecture encodes larger files by increasing the number of independently decoded strands rather than strand length. Decoding is therefore pleasingly parallel. Press et al. [12] predicted error-free recovery at exabyte scale for DNA using the same concatenated design; the extension to an 8-symbol alphabet does not alter this architectural property. However, decoding throughput remains a practical concern. Sequential decoding times of up to 70 seconds for 591 bytes reflect both the computational cost of the heap-based search and the use of a Python implementation, compared to the C++ implementation of the original work. Scaling to larger payloads will require parallel execution and optimized implementations.

The simulations assume independent errors, which may not fully reflect realistic synthesis and sequencing error patterns. Solid-phase peptide synthesis exhibits position-dependent error rates that increase with sequence length because of cumulative coupling inefficiency, and certain amino acid pairs are more prone to correlated side reactions [14]. Similarly, MS/MS sequencing accuracy varies with peptide composition and fragmentation patterns. These correlations may reduce the effective error-correcting capability of both models, particularly for HEDGES, whose decoder assumes uniform error probability along the strand. Additionally, the HEDGES biofilter for constraint enforcement was not implemented. Future work could explore correlated error models based on experimental data, larger alphabets exploiting larger ranges of amino acids, LDPC-marker code combinations for indel correction, and experimental validation with actual peptide synthesis.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29912-3_28

In conclusion, HEDGES codes show promise for peptide-based data storage, as they can transform indel errors into correctable substitution-like errors for the outer decoder. With appropriate parameter selection, this approach may achieve reliable storage at code rates suitable for practical applications.

**Acknowledgments.** This work was conducted within the framework of the Fraunhofer PREPARE program. The authors wish to express their gratitude for the financial support provided, which enabled the realization of this research.

**Disclosure of Interests.** The authors declare no competing interests.

## References

1. Akash, A., Bencurova, E., Dandekar, T.: How to make DNA data storage more applicable. Trends in Biotechnology **42**(1), 17–30 (2024). https://doi.org/10.1016/j.tibtech.2023.07.006
2. Anavy, L., Vaknin, I., Atar, O., Amit, R., Yakhini, Z.: Data storage in DNA with fewer synthesis cycles using composite dna letters. Nature Biotechnology **37** (2019). https://doi.org/10.1038/s41587-019-0240-x
3. Bornholt, J., Lopez, R., Carmean, D.M., Ceze, L., Seelig, G., Strauss, K.: A DNA-based archival storage system. SIGARCH Comput. Archit. News **44**(2), 637–649 (2016). https://doi.org/10.1145/2980024.2872397
4. Cafferty, B.J., Ten, A.S., Fink, M.J., Morey, S., Preston, D.J., Mrksich, M., Whitesides, G.M.: Storage of information using small organic molecules. ACS Central Science **5**(5), 911–916 (2019). https://doi.org/10.1021/acscentsci.9b00210
5. Ceze, L., Nivala, J., Strauss, K.: Molecular digital data storage using DNA. Nature Reviews Genetics **20**(8), 456–466 (2019). https://doi.org/10.1038/s41576-019-0125-3
6. Dong, Y., Sun, F., Ping, Z., Ouyang, Q., Quian, L.: DNA storage: research landscape and future prospects. National Science Review **7**, 1092–1107 (2020). https://doi.org/10.1093/nsr/nwaa007
7. Doricchi, A., Platnich, C.M., Gimpel, A., Horn, F., Earle, M., Lanzavecchia, G., Cortajarena, A.L., Liz-Marzán, L.M., Liu, N., Heckel, R., Grass, R.N., Krahne, R., Keyser, U.F., Garoli, D.: Emerging approaches to DNA data storage: Challenges and prospects. ACS Nano **16**(11), 17552–17571 (2022). https://doi.org/10.1021/acsnano.2c06748, pMID: 36256971
8. Heckel, R., Mikutis, G., Grass, R.N.: A characterization of the DNA data storage channel. Scientific Reports **9**, 9663 (2019). https://doi.org/10.1038/s41598-019-45832-6
9. Lenz, A., Siegel, P.H., Wachter-Zeh, A., Yaakobi, E.: The noisy drawing channel: Reliable data storage in DNA sequences. IEEE Transactions on Information Theory **69**(5), 2757–2778 (2023). https://doi.org/10.1109/TIT.2022.3231752
10. Luo, B., Gao, S., Wu, M., Dong, X., Deng, X., Hu, H.: High-capacity information storage using peptide-encapsulated hydrogels for long-term data preservation. Communication Materials **6**, 183 (2025). https://doi.org/10.1038/s43246-025-00915-y
11. Ng, C.C.A., Tam, W.M., Yin, H., Wu, Q., So, P.K., Wong, M.Y.M., Lau, F.C.M., Yao, Z.P.: Data storage using peptide sequences. Nature Communications **12**(1) (2021). https://doi.org/10.1038/s41467-021-24496-9

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29912-3_28

12. Press, W., Hawkins, J., Jones, S., Schaub, J., Finkelstein, I.: Hedges error-correcting code for dna storage corrects indels and allows sequence constraints. Proceedings of the National Academy of Sciences **117**, 202004821 (07 2020). https://doi.org/10.1073/pnas.2004821117
13. Reinsel, D., Gantz, J., Rydning, J.: The digitization of the world. White paper, International Data Corporation (2018)
14. Ren, Y., Zhang, Y., Liu, Y., Wu, Q., Hu, H.G., Li, J., Fan, C., Chen, D., Liu, K., Zhang, H.: Highly reliable and efficient encoding systems for hexadecimal polypeptide-based data storage. Fundamental Research **3**(2), 298–304 (2021). https://doi.org/10.1016/j.fmre.2021.11.030
15. Zhang, A., Wang, L., Zhai, X., Xiao, Y., Wu, Y., Zhao, Y., Liu, K., Zheng, J.S., Chen, D.: Composite mapping for peptide-based data storage with higher coding density and fewer synthesis cycles. Advanced Science **12**(27) (2025). https://doi.org/10.1002/advs.202503790

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29912-3_28