# FoldMark: Safeguarding Protein Structure Generative Models with Distributional and Evolutionary Watermarking

Zaixi Zhang[1], Ruofan Jin[1], Guangxue Xu[2], Xiaotong Wang[2], Le Cong[2*], Mengdi Wang[1*]

[1]Princeton University, Princeton, NJ, USA. [2]Stanford University, Stanford, CA, USA.

[*]Corresponding authors. Emails: lecong@stanford.edu, mengdiw@princeton.edu

**Proteins are the principal architects of life, fueling advances in bioengineering, drug discovery, and synthetic biology. The integration of generative AI with computational protein science has revolutionized protein design while also posing dual-use risks, such as enabling the creation of pandemic-capable proteins that require strong biosecurity safeguards. Here, we introduce FoldMark, a first-of-its-kind watermarking strategy leveraging distributional and evolutionary principles tailored for protein generative models, balancing watermark capacity and structural quality. FoldMark achieves over 95% watermark bit accuracy at 32 bits with minimal impact on structural quality (>0.9 scTM scores) for leading models including AlphaFold3, ESMFold, RFDiffusion, and RFDiffusionAA. For user tracing, FoldMark can successfully trace up to *1 million* users. To validate FoldMark in wet lab, we applied it to structure-based design of EGFP and CRISPR-Cas13, showing wildtype-level function (98% fluorescence, 95% editing efficiency) and >90% watermark detection, demonstrating its practical utility for safeguarding AI-driven protein research.**

# Introduction

Proteins are life's essential building blocks and the basis of all living organisms. Understanding their structure is key to uncovering the mechanisms behind their function. With the advancement of generative AI (1), protein structure generative models have revolutionized both protein structure prediction (2, 3) and de novo protein design (4–6), opening up a wide range of applications in bioengineering and drug discovery. For example, AlphaFold2 (2) made a breakthrough by accurately predicting protein structures from amino acid sequences at near-experimental accuracy, solving a decades-old challenge in biology. Its successor, AlphaFold3 (3), further improved on this by enhancing the ability to model more complex protein interactions and assemblies. Meanwhile, RFDiffusion (4) and Chroma (5) introduced diffusion-based generative models that enable the creation of novel protein structures with desired properties and functions, pushing the boundaries of de novo protein design. In recognition of the profound impact of these models, the *2024 Nobel Prize in Chemistry* was awarded to David Baker "*for computational protein design*" and to Demis Hassabis and John M. Jumper "*for protein structure prediction*" (7).

However, the rapid development of protein generative models (PGMs) and the lack of corresponding regulations lead to **biosecurity** and **Intellectual Property (IP) infringement** concerns (8–10). **First**, the unregularized yet powerful protein generative models are vulnerable to misuse and cause bio-security/safety concerns (9, 10) (Figure. 1 A). For example, protein generative models can be used to design new proteins with harmful properties, such as pathogens (11), toxins (12), or viruses (13) that can be used as bioweapons. Recent community discussions call for *built-in biosecurity safeguards for generative AI tools* to mitigate these dangers (14, 15). **Second**, the ease of model sharing brings up IP infringement concerns, including the risk of unauthorized use of generated structures and redistribution of pretrained models for profit, which could undermine the interests of the original creators (16–18). For instance, the latest AlphaFold3 Server explicitly specifies in its Terms of Use that users *must not use AlphaFold Server or its outputs to train machine learning models"* or employ it *in connection with any commercial activities"* (19) (Figure 1B). Therefore, there is an urgent need for a reliable tool to track and audit the use of protein generative models (14, 15).

Similar problems have occurred in text and image generation. For instance, Large language

models (LLMs) such as ChatGPT can be used to create fake news and to cheat on academic writing (20–22). The latest text-to-image models, such as Stable Diffusion (23) and DALL·E 3 (24) enable users to create photo-realistic images like deepfakes (25) for illegal purposes. As a result, there is growing consensus that the ability to detect, track, and audit the use of AI-generated content is essential for harm reduction and regulation (21,26). Recently, the watermark has become one of the most promising protection strategies, which embeds hidden patterns in the generated content and is imperceptible to humans, while making the embedded information algorithmically identifiable. Although watermarks have been applied for Language models (27–32) (e.g., SynthID-Text (31)) and text-to-image models (33–36) (e.g., AquaLoRA (35)), extending these methods to protein structure data presents unique challenges. Unlike text and images, protein structures are highly sensitive to minute changes (2, 6, 37), and embedding watermarks without disrupting the biological functionality or stability of the proteins is a complex task. Moreover, protein structures exhibit complex geometrical symmetries, making traditional watermarking methods less effective due to the requirement for equivariance (4, 38).

In this paper, as a proof of concept, we propose FoldMark, a generalized watermarking method for protein generative models, e.g., AlphaFold3 (3), ESMFold (39), RFDiffusion (4), and RFDiffusionAA (40). FoldMark builds upon pretrained protein generative models and generally has two training stages (Figure. 1 C and D). In the first stage, FoldMark pretrains a watermark encoder and decoder to subtly modify structures, embedding watermark code across all residues (i.e., *distributional*), guided by *evolutionary* signals (e.g., genetic search, protein language model, structure templates) that minimize noise in conserved regions and exploit flexible regions for enhanced capacity, ensuring retrievability without compromising structural quality. In the second stage, we use watermark-conditioned LoRA (41), which flexibly encodes the given watermark code and merges it into the original model weights, without changing or adding extra model architecture. The protein generative model is fine-tuned using three objectives: message retrieval loss, consistency loss, and structure auxiliary loss. The message retrieval loss ensures that watermarks are effectively embedded within the generated structures, enabling the reliable recovery of the embedded codes. The consistency loss promotes minimal impact of watermarking on the quality of the generated structures. Additionally, auxiliary losses, such as pairwise distance losses inherited by AlphaFold3 (3) and RFDiffusion (4), are incorporated to further enhance structural quality. In applications, we

demonstrate FoldMark's strong capability for both generated-structure detection and user tracing (Figure. 1 E). The model owner can assert ownership over generated structures or trace the user responsible for producing a potentially pathogenic protein, thereby safeguarding intellectual property rights and enhancing biosecurity.

In experiments, we first observe that FoldMark achieves nearly 100% bit accuracy on watermark code recovery from encoded protein structures with minimal influence on structural validity (measured by scRMSD, scTM, and RMSD), which means FoldMark can reliably embed and retrieve watermarks (up to 32 bits) with minimal structural deviation (Figure. 2 and 3). In two fundatmental applications: watermark detection and user tracing. In detection, it reliably verifies if a structure originates from a specific model (around 99% TPR @ FPR=$10^{-5}$). In tracing, it accurately traces the exact user by matching watermarks (up to $10^6$ users). In another advanced application to prohibit the unauthorized use of generated data for third-party model training (e.g., compliance with AlphaFold3's terms), FoldMark can reliably detect the violation of terms, even though only 30% of the datasets are from AlphaFold3 and precautions to evade detections (e.g., early stop of fine-tuning) are used (Figure. 5). Detailed analysis reveals the correlation between watermark confidence of FoldMark and the structural confidence (e.g., pLDDT), demonstrating that FoldMark adaptively embeds watermarks to balance retrieval accuracy and structural integrity (Figure. 4). Finally, we apply FoldMark to the pipeline of EGFP and Cas13 structure-based redesign (Figure. 6): the redesigned proteins maintain wildtype-equivalent functionality in wet experiments (avg. 98% fluorescence and avg. 95% editing efficiency retained, respectively) with over 90% watermark detection success rates, showing FoldMark's practical utility in ensuring secure and ethical protein engineering.

In summary, by integrating distributional and evolutionary design principles for protein structures with state-of-the-art generative AI techniques, FoldMark offers an innovative solution for watermarking protein generative models. FoldMark is poised to play a pivotal role in future biosecurity safeguard frameworks for emerging biological foundation models, helping to ensure both safety and continued innovation in the era of AI-driven scientific research.

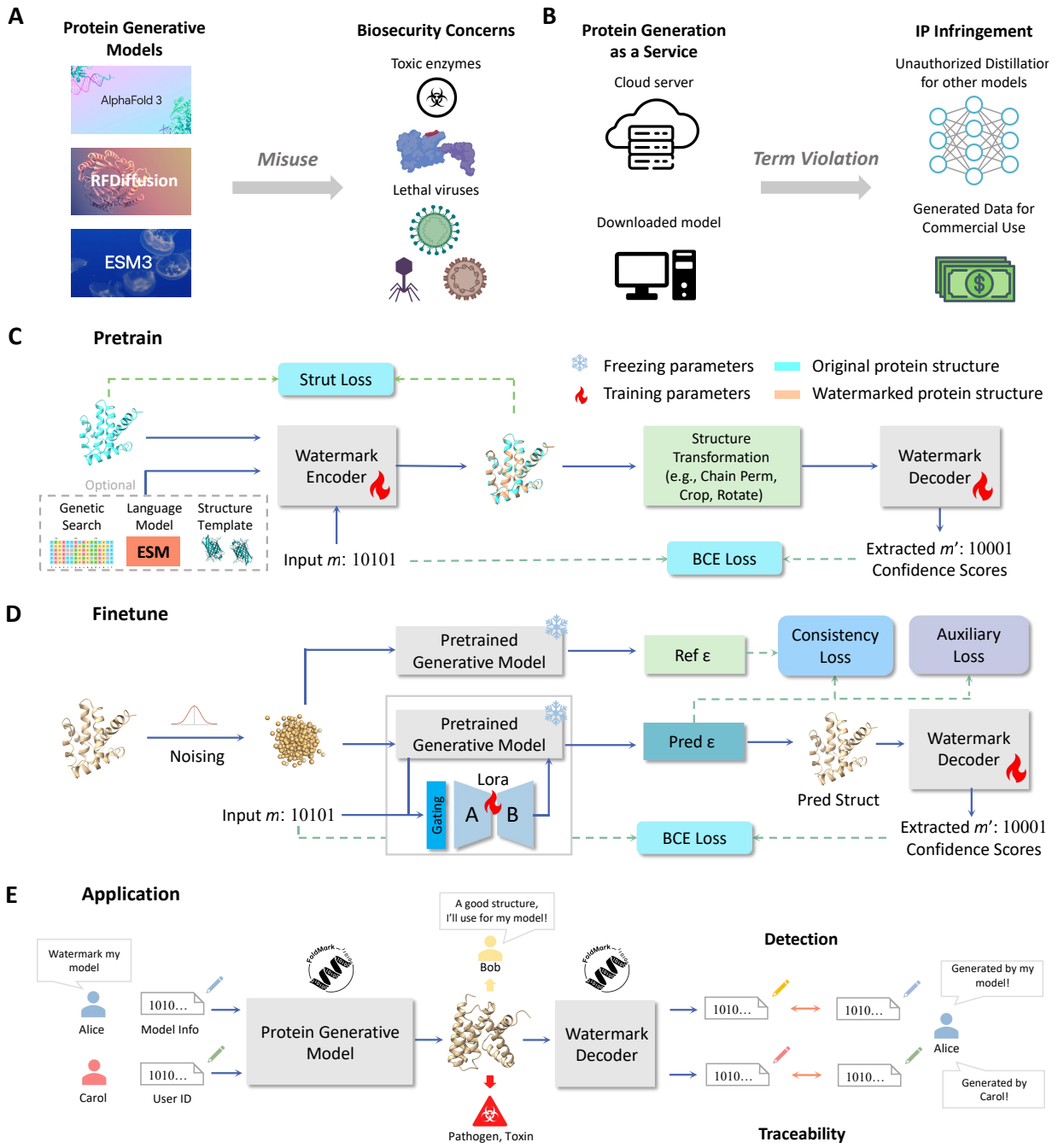



**Figure 1**: **FoldMark: Distributional and Evolutionary Watermark for Protein Generative Models (PGMs).** (**A** and **B**) Powerful protein generative models, such as AlphaFold3, RFDiffusion, and ESM3, bring increasing biosecurity and IP infringement concerns (14, 15). (**A**) PGMs may be misused for synthesizing toxic enzymes or lethal viruses. (**B**) Protein generative models like AlphaFold3 Server are evolving into 'Protein Generation as a Service' (PGaaS). AlphaFold3 Server has terms that users "*must not use AlphaFold Server or its outputs to train machine learning models*" or "*in connection with any commercial activities*" (19). (Caption continues on the next page.)

**Figure 1 (continued).** However, users may breach terms via unauthorized distillation for unauthorized commercial use. (**C** and **D**) FoldMark provides built-in protection for PGMs by embedding watermarks during protein structure generation for reliable tracking. The implementation of FoldMark on PGMs takes a two-stage process. (**C**) FoldMark pretraining has two stages named Encoder and Decoder. The FoldMark Encoder learns to embed watermark codes by subtly modifying protein structures. It also accepts a wide variety of optional input features, such as genetic search, protein language model embeddings, and structural templates for better performance. FoldMark Decoder learns to recover the watermark based on the watermark structures. (**D**) FoldMark fine-tuning: The pretrained PGM is fine-tuned with watermark-conditioned LoRA. (**E**) Applying FoldMark for generated-structure detection and user tracing. The model owner, Alice releases the pretrained model or deploys it on public platforms. Bob, who downloads Alice's model and code, generates protein structures and falsely claims ownership. Carol registers as a user on the server and utilizes the API to generate protein structures for pathogen and toxin synthesis. FoldMark allows Alice to embed watermarks in generated protein structures, which can then be extracted to detect and identify unauthorized or illegal use, thus safeguarding biosecurity and intellectual property rights.

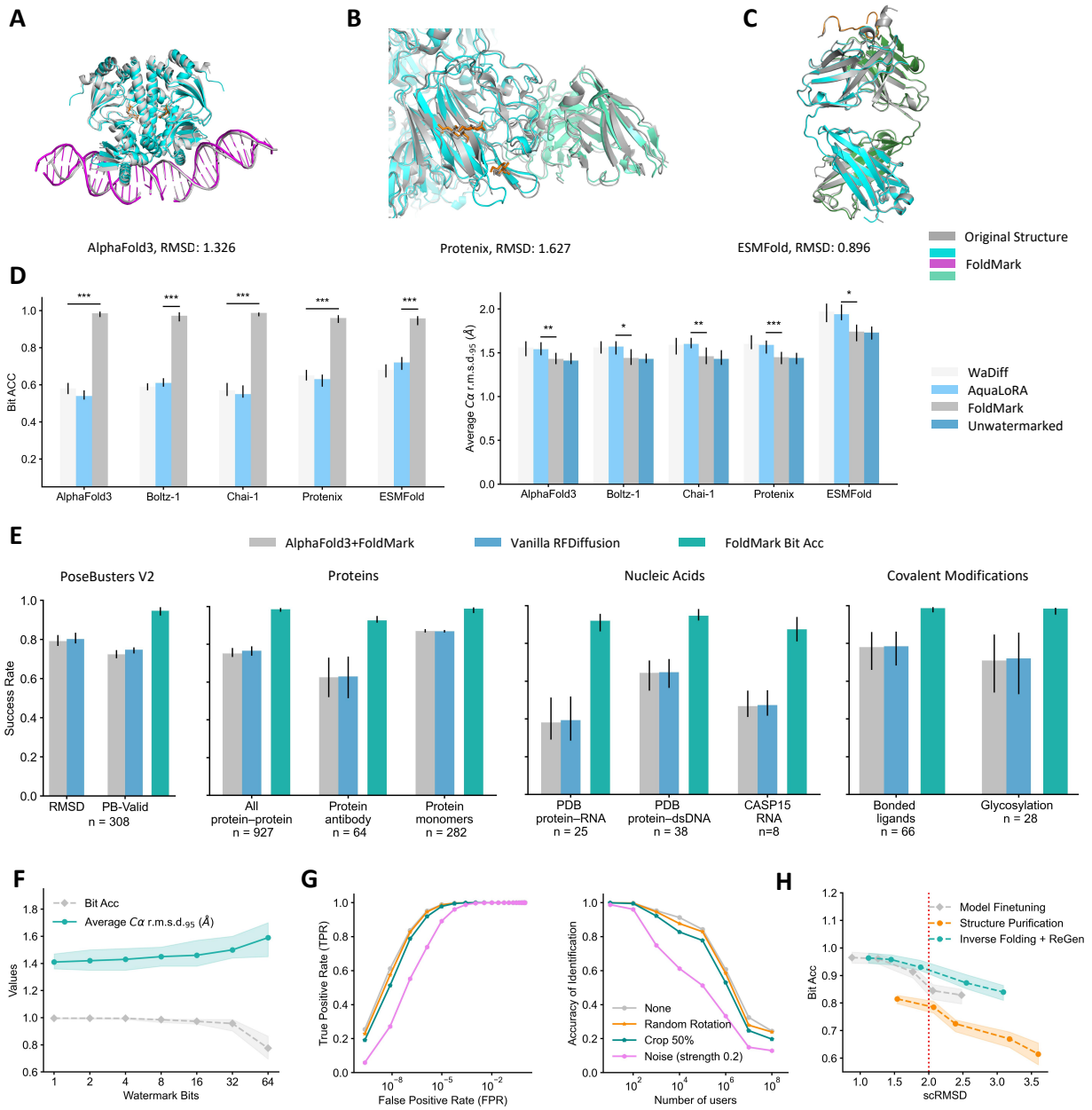



**Figure 2**: **FoldMark Successfully Watermark Structure Prediction Models with Minimal Influence on Structure Quality.** (**A** to **C**) Example structures watermarked by FoldMark, RMSD is reported between the watermarked and the original predicted structure. (**A**) Bacterial CRP/FNR family transcriptional regulator protein bound to DNA and cGMP (PDB 7PZB; Watermarked AlphaFold3). (**B**) Human coronavirus OC43 spike protein (PDB 7PNM; Watermarked Protenix). (**C**) Mesothelin C-terminal peptide bound to the monoclonal antibody 15B6 (PDB 7U8C; Watermarked ESMFold). (**D**) Benchmarking FoldMark with adapted watermark methods from image domain on single protein chains (number of single protein chains = 500). (Caption continues on the next page.)

**Figure 2 (continued).** We measure Bit ACC with 16 watermark bits. The structural validity is quantified by computing C$\alpha$ r.m.s.d. at 95% coverage. The means and 95% confidence intervals with 10,000 bootstrap resamplings are shown. Significance levels were calculated using two-sided Wilcoxon signed-rank tests between FoldMark and the second-best watermark method; $^{***}P < 0.001$, $^{**}P < 0.01$, $^{*}P < 0.05$. Exact $P$-values (from left to right) are as follows: $1.46 \times 10^{-80}$, $6.97 \times 10^{-61}$, $1.33 \times 10^{-67}$, $2.41 \times 10^{-48}$, $3.24 \times 10^{-20}$, $9.73 \times 10^{-3}$, $2.11 \times 10^{-2}$, $1.05 \times 10^{-2}$, $1.46 \times 10^{-6}$, and $4.48 \times 10^{-2}$. (**E**) The performance of watermarked AlphaFold3 with FoldMark on different data categories. Metrics are as follows: percentage of pocket-aligned ligand r.m.s.d. < 2 Å for ligands and covalent modifications; LDDT for nucleic acid and protein monomers; interface LDDT for protein–nucleic acid complexes; and percentage DockQ > 0.23 for protein–protein and protein–antibody interfaces. Sampling and ranking details follow AlphaFold3. For ligands, $n$ indicates the number of targets; for nucleic acids, $n$ indicates the number of structures; for the others, $n$ indicates the number of clusters. The bar height indicates the mean; error bars indicate exact binomial distribution 95% confidence intervals for PoseBusters and by 10,000 bootstrap resamples for all others. (**F**) Evaluating AlphaFold3 + FoldMark with different watermark bits. The shadows indicate the 95% confidence intervals obtained by 10,000 bootstrap resampling. (**G**) FoldMark detection and user identification results with AlphaFold3. The left shows TPR/FPR curve of the detection under different transformations. The right is the accuracy of identification with different numbers of users (FPR = $10^{-3}$). None: no post-processing; Random Rotation: randomly rotated the generated structure; Crop 50%: randomly crop 50% of the continuous protein chains; Noise (strength 0.2): add Gaussian noise with 0.2 strength to coordinates. (**H**) Robustness of FoldMark under different adaptive attacks. The red dotted line indicates scRMSD=2.0 and structures with scRMSD under 2.0 are generally regarded as good quality. The shadows indicate the 95% confidence intervals obtained by 10,000 bootstrap resampling.

# Results

## Distributional and Evolutionary Watermark

The rapid advancement of generative AI for protein design raises critical ethical challenges, including biosecurity risks of creating pathogenic proteins (Figure. 1 A) and IP infringement concerns due to unauthorized use of generated structures (Figure. 1 B), prompting the development of FoldMark as a protein watermarking solution to ensure traceability and IP protection. As a generalized watermarking strategy tailored for protein structure generative models, FoldMark consists of two main stages (Figure. 1 C and D).

First, FoldMark pretrains a watermark encoder and decoder to subtly modify protein structures, embedding user-specific watermark code that can be reliably retrieved. FoldMark leverages **evolutionary information** for enhanced watermarking: The FoldMark encoder also supports optional input features like genetic search-derived multiple sequence alignments (MSAs) (42), protein language model embeddings from ESM-3B (39), and structural templates from UniRef100 (43) and Protein Data Bank (PDB) (44) databases. These features capture co-evolutionary signals and species-specific nuances, guiding watermark embedding: *conserved regions receive minimal noise to preserve functional integrity, while intrinsically disordered regions tolerate higher noise*, leveraging evolutionary knowledge for robust and context-aware watermarking. The watermark code $m$ is a binary string that is encoded into the structure in a **distributional manner**—across all residues or tokens (with the length of $m$ referred to as the number of watermark bits). This approach enhances robustness against cropping and other structural modifications that might otherwise compromise complete watermark retrieval. The watermark decoder take the watermarked structure as input, predicts the watermark code/confidence based on residue/token representations, and take a weighted sum to get the final extracted watermark $m'$. In the pertaining, the structure loss and watermark binary-cross entropy (BCE) loss are optimized, encouraging small structure deviations and high watermark retrieval accuracy. To enhance the robustness of watermark decoding, the watermarked structure undergoes structure transformation (e.g., symmetric chain permutation, cropping, and random rotation) before feeding into watermark decoder.

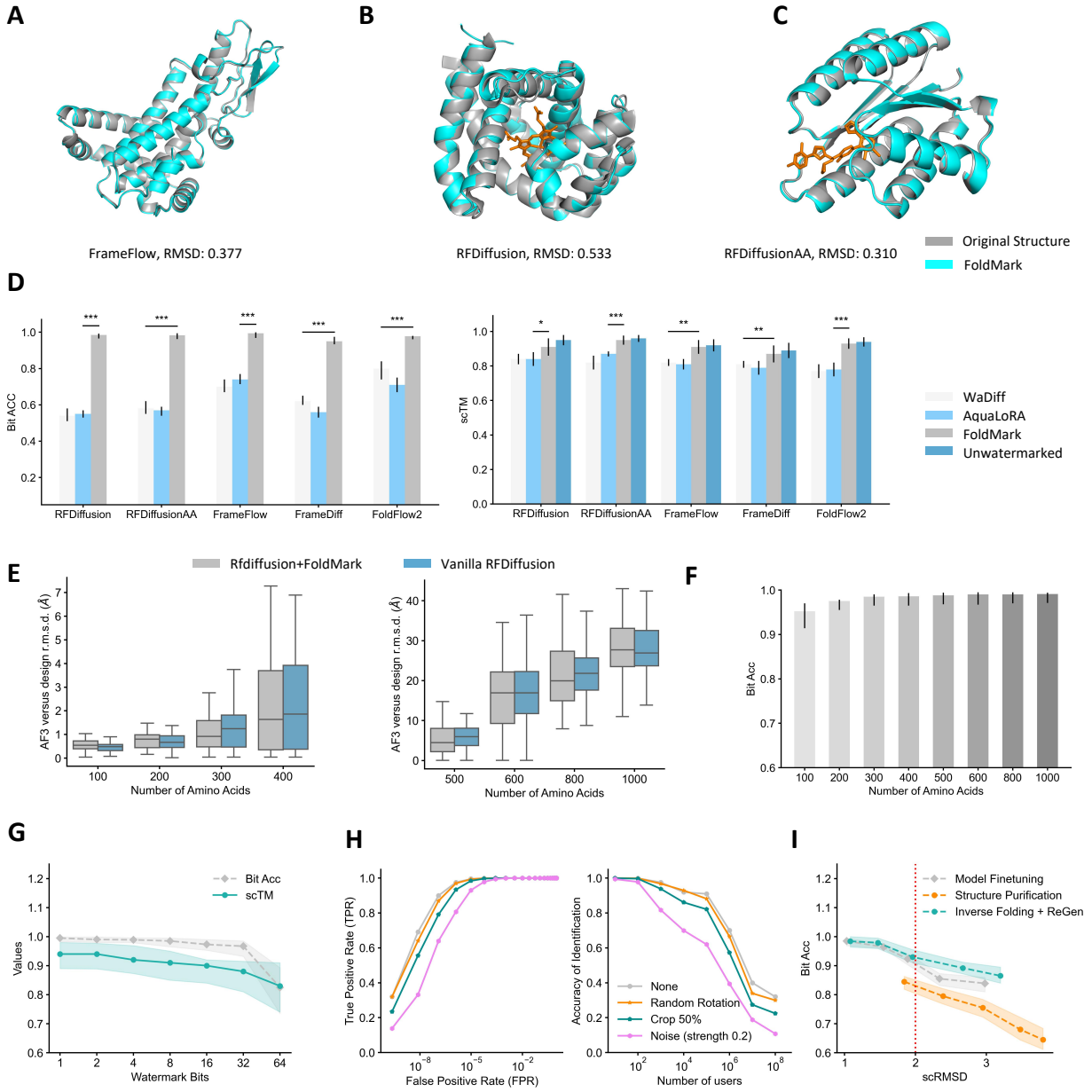



**Figure 3**: **FoldMark Successfully Watermark *de novo* Structure Generation Models.** (**A** to **C**) Example structures watermarked by FoldMark, RMSD is reported between the watermarked and the original generated structure. (**A**) A 256-length protein generated by FrameFlow/ FoldMark+FrameFlow (RMSD:0.377). (**B**) Generated binder to the ligand heme from PDB 8vc8 by RFDiffusion/ FoldMark+RFDiffusion (RMSD: 0.533). (**C**) Generated binder to the ligand OQO from PDB 7v11 by RFDiffusionAA/ FoldMark+RFDiffusionAA (RMSD: 0.310). (**D**) Benchmarking FoldMark with adapted watermark methods from image domain on generating 300-length proteins (number of generated proteins = 100). We measure Bit ACC with 16 watermark bits. The structural validity is quantified by computing scTM. (Caption continues on the next page.)

**Figure 3 (continued).** The means and 95% confidence intervals with 10,000 bootstrap resamplings are shown. Significance levels were calculated using two-sided Wilcoxon signed-rank tests between FoldMark and the second-best watermark method; $***P < 0.001$, $**P < 0.01$, $*P < 0.05$. Exact $P$-values (from left to right) are as follows: $3.89 \times 10^{-18}$, $6.91 \times 10^{-18}$, $1.70 \times 10^{-16}$, $3.90 \times 10^{-18}$, $1.50 \times 10^{-9}$, $3.68 \times 10^{-2}$, $5.55 \times 10^{-7}$, $1.09 \times 10^{-3}$, $1.27 \times 10^{-3}$, and $8.96 \times 10^{-8}$. (**E**) Watermarked RFDiffusion by FoldMark and the vanilla RFDiffusion have similar generation quality (scRMSD) across different lengths. 100 structures are generated at each length. Each box represents the interquartile range (IQR), with the bottom edge marking the 25th percentile (Q1) and the top edge indicating the 75th percentile (Q3). The median (50th percentile) is shown as a horizontal line within each box. Whiskers extend to the minimum and maximum values within 1.5 times the IQR from Q1 and Q3, respectively. (**F**) The relationship between watermark capability and the number of amino acids of the generated structure. The watermark bits is set to 16. (**G**) Evaluating RFDiffusion + FoldMark with different watermark bits. The shadows indicate the 95% confidence intervals obtained by 10,000 bootstrap resampling. (**H**) FoldMark detection and user identification results with RFDiffusion. The left shows TPR/FPR curve of the detection under different transformations. The right is the accuracy of identification with different numbers of users (FPR=$10^{-3}$). (**I**) Robustness of FoldMark under different adaptive attacks. The red dotted line indicates scRMSD=2.0 and structures with scRMSD under 2.0 are generally regarded as good. The shadows indicate the 95% confidence intervals obtained by 10,000 bootstrap resampling.

Second, protein generative models are fine-tuned with watermark-conditioned parameter efficient fine-tuning to preserve generation quality while learning to generate watermarked structures with high recovery rates. As most state-of-the-art protein generative models adopt diffusion-based architecture such as RFDiffusion (4) and AlphaFold3 (3), Figure. 1 D uses diffusion model fine-tuning as one example, which can be generalized to non-diffusion architectures such as FrameFlow (45) and ESMFold (39). During fine-tuning, the watermarked structure is first noised to $t \sim \text{Uniform}[0, 1]$, initiating a controlled corruption process. The fine-tuned PGM then learns to reverse this noise, iteratively refining the structure by predicting the original watermarked structure from the noised structure. To preserve generation quality and avoid overfitting, a consistency loss is added by comparing the predicted noise (Pred $\epsilon$) and the reference noise from the frozen pre-

trained PGM (Ref $\epsilon$). Note that a watermark-conditioned LoRA (41) is employed during fine-tuning to efficiently embed watermark information into protein structures while reducing computational costs. The approach leverages a low-rank adaptation (LoRA) update, formulated as The approach uses a low-rank update $\Delta\mathbf{W}(m) = \mathbf{G}(m) \odot (\mathbf{A} \times \mathbf{B})$, where the gating vector $\mathbf{G}$, derived from a binary watermark code $m$ via a linear transformation, modulates the LoRA weight update applied to all linear and attention layers in structural prediction modules of PGM. This update is added to the original model weights as $\mathbf{W}_{\text{watermarked}} = \mathbf{W} + \alpha\Delta\mathbf{W}(m)$, with $\alpha$ scaling the watermark's influence, enabling flexible and efficient watermark integration (more details in Methods). Moreover, in finetuning, auxiliary losses such as pairwise distance loss from FrameDiff (46) penalizing atomic clashes are considered to further encourage structural quality. Finally, the watermark decoder takes the reconstructed structure based on the predicted noise to predict the watermark code and computes the BCE Loss similar to FoldMark Pretraining.

After fine-tuning, the protein generative model gains the ability to produce structures embedded with a designated watermark. When deployed on platforms like the AlphaFold3 server, these structures can be encoded with user ID or timestamp information for reliable traceability. The watermark-conditioned LoRA weights can also be merged into the pretrained weights for open-source distribution, and the generated structures can encode model name or version information for watermark detection.

## Watermarking State-of-the-art Protein Generative Models

In Figure. 2 and 3, we showed experiments of watermarking unconditional protein structure generative models (i.e., RFDiffusion (4), RFDiffusion-AA (40), FoldFlow (47), FrameDiff (46), and FrameFlow (45)) and protein structure prediction models (i.e., AlphaFold3 (3), Protenix (48), Boltz-1 (49), Chai-1 (50), and ESMFold (39)). Figure. 2 A to C and 3 A to C shows the examples of the generated structures by original PGMs and the watermarked PGMs, demonstrating small structure modifications for watermarking. Furthermore, Figure. 2 D and Figure. 3 D uses watermark bit accuracy (Bit ACC) with 16 watermark bits to benchmark FoldMark with adapted watermark methods from the image domain (WaDiff and AquaLoRA). With customized geometric data augmentation, powerful structure encoder/decoder, optimized fine-tuning strategy, and evo-

lutionary/template supporting information, FoldMark overperforms baseline methods significantly ($p<0.001$ with two-sided Wilcoxon signed-rank tests), consistently achieving over 95% Bit ACC across different PGMs. We also quantify the structural validity with C$\alpha$ r.m.s.d. at 95% coverage and scTM following AlphaFold2 (2) and FrameDiff (46). FoldMark performs better than baseline methods while keeping comparable structural quality. More comparisons of FoldMark with baseline methods are listed in Methods. In Figures 2 E and 3 E, we stratify the performance of FoldMark combined with AlphaFold3 and RFDiffusion across different biomolecule categories (protein-ligand, protein-protein, nucleic acids, and covalent modifications) or varying sequence lengths (100 to 1,000 amino acids). We observe that FoldMark consistently achieves high predictive accuracy for most interactions and shorter sequences (e.g., $< 500$ amino acids), while structural quality tends to decline for nucleic acid complexes and longer sequences, due to the increased structural complexity. Note that some underrepresented data categories (e.g., protein-antibody) may exhibit lower performance, representing a limitation that could be addressed in future work through expanded training datasets or refined modeling approaches. In Figures 2 F and 3 G, we vary the watermark bit length from 1 to 64 and evaluate its impact on watermark bit accuracy (Bit Acc) and structural validity, measured as $C_\alpha$ root-mean-square deviation (r.m.s.d.) at 95% coverage and self-consistency TM-score (scTM). In most cases with fewer than 32 bits, FoldMark achieves nearly 100% bit accuracy in recovering watermark codes from encoded protein structures, with minimal impact on structural validity ($< 1.5$ r.m.s.d. and $> 0.9$ scTM). Figure 3 H explores watermark capacity as a function of protein sequence length, with a fixed watermark bit length of 16. Using distributional watermark codes, FoldMark demonstrates a steady increase in Bit Acc for longer proteins, attributable to the expanded data space available for watermark embedding. Overall, FoldMark provides a robust and effective framework for protecting protein generative models while preserving structural integrity.

## Robustness against Post-processing and Adaptive Attacks

In real applications, the user may unintentionally use post-processing or intentionally take adaptive attacks to bypass the safeguarding of FoldMark. Here, we consider three common post-processing methods and three adaptive attacks in Figure. 2 G and H and Figure. 3 H and I. The post-processing methods include (1) Random Rotation: randomly rotated the generated structure; (2) Crop 50%:

randomly crop 50% of the continuous protein chains; (3) Noise (strength 0.2): add Gaussian noise with 0.2 strength to coordinates. Adaptive attacks include Structure Purification, Model Fine-Tuning, and Inverse Folding+ReGen. (1) Structure Purification is inspired by DiffPure (51), which uses a clean-version PGM to first noise the watermarked structure (e.g., 20 steps) and then denoise to obtain the new structure; (2) Model Fine-Tuning tries to fine-tune the watermarked model using clean protein data to erase the watermark in PGMs; (3) Inverse Folding + Regen means first use inverse folding models such as ProteinMPNN (52) to obtain the protein sequences and then use watermarked PGMs to regenerate the structure based on the derived sequence. Our analysis reveals that FoldMark exhibits robust resistance to both standard transformations and these adaptive attacks. Its resilience to cropping, translation, and rotation stems from its distributional watermark scheme, where watermark information is encoded into each residue rather than localized in specific regions. For instance, cropping only partially reduces the signal without fully erasing it, as the remaining residues retain their encoded information. In adaptive attacks, Model Fine-Tuning and Inverse Folding + Regeneration are less successful. Fine-tuning requires substantial clean data and epochs to overwrite the watermark, yet FoldMark's pretraining on augmented, watermarked structures embeds the watermark deeply into its parameter space, resisting casual erasure. In Inverse Folding + Regeneration, the watermark's structural features shape the inferred sequence with latent biases, which the watermarked PGM amplifies during regeneration. The PGM's pretrained priors favor watermarked structures, reinstating the signal even from an apparently clean sequence. Adding noise to a watermarked structure and Structure Purification, can partially disrupt the watermark by introducing randomness that obscures the watermark signal. However, this comes at a steep cost: excessive noise could significantly degrade structural quality, often increasing $C_\alpha$ r.m.s.d. beyond acceptable thresholds (e.g., >2 Å). This trade-off limits the attack's practicality, as the resulting structures become biologically implausible.

To sum up, FoldMark's integrated design—combining a distributional watermark scheme with strategic data augmentation—underpins its resistance to data transformation and adaptive attacks.

## Applications in Watermark Detection and User Tracing

As shown in Figure 1 E, two fundamental applications of FoldMark: watermark detection and user tracing are considered. Consider a scenario with Alice, the model owner, who trains and deploys a pretrained protein generative model (PGM) with inference code on a platform. Bob, an unauthorized user, downloads Alice's model to generate protein structures, claiming ownership. Carol, a registered user, generates structures via the platform's API. In watermark detection, extracting Alice's watermark from a structure confirms her copyright and flags it as artificially generated. In user tracing, Alice assigns unique watermarks to each user (e.g., Carol), enabling identification by matching extracted watermarks against a database, thus tracing infringement sources. Following previous works (33, 36), we formulate the tasks with rigorous statistical tests:

### FoldMark for Detection

Alice embeds a $k$-bit binary code $m \in \{0,1\}^k$ into generated structures. The watermark decoder extracts a message $m'$ from a structure $x$ and compares it to $m$. Detection hinges on the number of matching bits, $M(m, m')$, with a threshold $\tau \in \{0, \ldots, k\}$:

$$M(m, m') \geq \tau, \tag{1}$$

flagging the structure as Alice's if satisfied. Formally, we test $H_1$: "Structure was generated by Alice's model" against $H_0$: "Structure was not generated by Alice's model." Under $H_0$ (unwatermarked structures), extracted bits $m'_1, \ldots, m'_k$ are independent and identically distributed (i.i.d.) Bernoulli random variables with parameter 0.5, so $M(m, m')$ follows a binomial distribution with parameters $(k, 0.5)$. The True Positive Rate (TPR) is the probability that the watermarked structure is successfully detected; The False Positive Rate (FPR) is the probability that $M(m, m')$ exceeds the threshold $\tau$ for unwatermarked structures. It is obtained from the cumulative distribution function (CDF) of the binomial distribution and can be expressed using the regularized incomplete beta function $I_x(a; b)$:

$$\mathrm{FPR}(\tau) = \mathbb{P}(M > \tau \mid H_0) = I_{1/2}(\tau + 1, k - \tau). \tag{2}$$

In experiments, Equation 2 is used inversely: we find the threshold $\tau$ to achieve a required FPR for detection. We conduct detection experiments with AlphaFold3 and RFDiffusion, generating

5,000 watermarked and unwatermarked structures respectively for each model, and use FoldMark for detection. Figure. 2 G and Figure. 3 H shows the trade-off between TPR/FPR under different data transformations. We can observe that FoldMark successfully achieves over 90% TPR at FPR equals around $10^{-5}$, demonstrating robust detection performance even under transformations such as rotation, cropping, and noise addition.

**FoldMark for Tracing**

For tracing, Alice assigns each user $i$ (where $i = 1, \ldots, N$) a unique $k$-bit signature $m^{(i)} \in \{0, 1\}^k$, embedded in their model instance. This allows tracking misuse—e.g., identifying a specific user generating viral structures. The extractor compares $m'$ from a structure against $(m^{(1)}, \ldots, m^{(N)})$, testing $N$ hypotheses. If all are rejected, the structure is deemed untraceable; otherwise, it's attributed to $\text{argmax}_{i=1..N} M(m', m^{(i)})$. Regarding the tracing task, false positives are more likely since there are $N$ tests. The global False Positive Rate (FPR) at a given threshold $\tau$ is:

$$\text{FPR}(\tau, N) = 1 - (1 - \text{FPR}(\tau))^N \approx N \times \text{FPR}(\tau). \tag{3}$$

In experiments (Figure. 2 G and Figure. 3 H), we vary the number of users, and calculate the accuracy of identification (the ratio of watermarked structures successfully being traced to the right user). We use AlphaFold3 and RFDiffusion for experiments. Each user generate 5 structures. For scenarios with over 10,000 users, we subsampled part of the users for testing following previous works (33, 36). We observe that FoldMark identify a user among $10^4$ users with 95% accuracy when the structure is not modified. With noising and up to 1 million users, the accuracy may drop to around 40%. However, this may still be promising: if a user generates 3 structures, he will be identified 80% of the time.

## Distributional and Evolutionary Watermark Scheme

FoldMark introduces a novel distributional and evolutionary watermarking scheme, embedding watermark information across all residues of a protein structure, and leveraging evolutionary priors such as MSA and structural to guide the process. In Figure. 4, we conduct a detailed analysis. In Figure. 4 A and B, we compare the structural confidence pLDDT from PGMs and the normalized

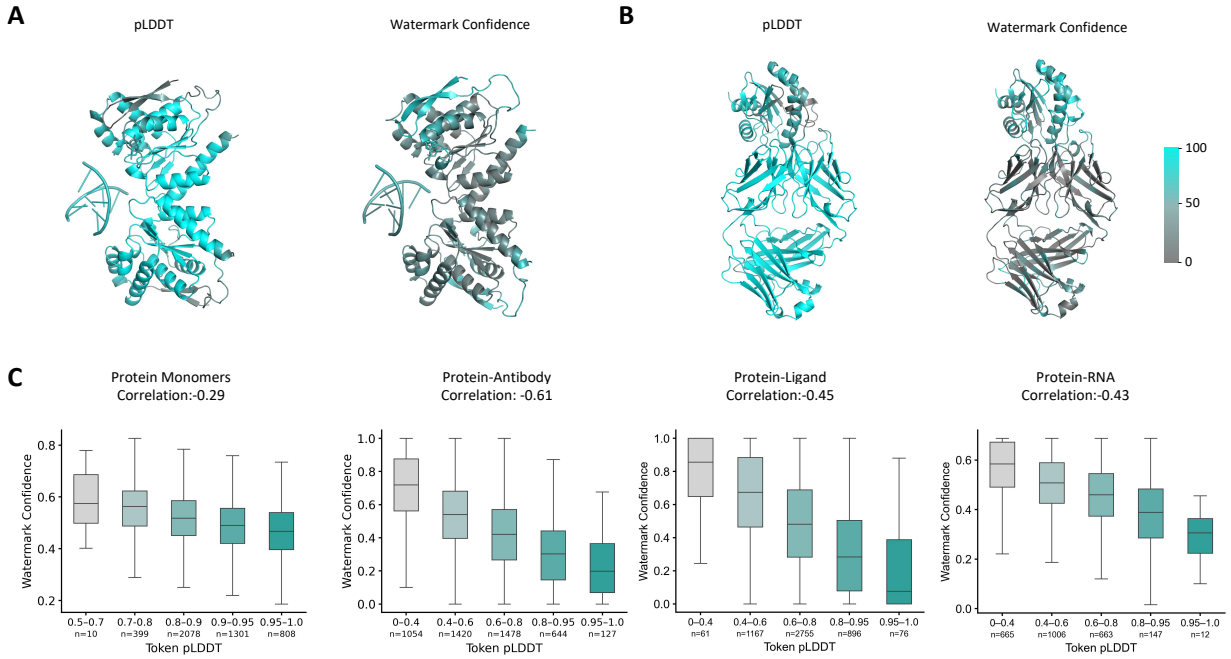



**Figure 4**: **Relationship of Watermark Confidence with Structure Confidence**. (**A** and **B**) The predicted Local Distance Difference Test (pLDDT) scores from the watermarked structure prediction model and the normalized watermark confidence scores from FoldMark decoder. (**A**) Predicted Tir-dsDNA complex (PDB 7X5M; Watermarked AlphaFold3). (**B**) Predicted structure of HIV-1 MPER scaffold in complex with antibody Fab Ab45.2 (PDB 9BDI; Watermarked Protenix). (**C**) The relationship of watermark confidence with token pLDDT. Protein monomers, protein-antibody interfaces, protein-ligand interfaces, and protein-RNA interfaces are selected for evaluation. 50 structures from the first three categories and 25 structures from protein-RNA are selected (100 tokens are randomly sampled in each structure). *n* values report the number of evaluated tokens in each band. The Pearson Correlations are -0.29, -0.61, -0.45, and -0.43 respectively. Each box represents the interquartile range (IQR), with the bottom edge marking the 25th percentile (Q1) and the top edge indicating the 75th percentile (Q3). The median (50th percentile) is shown as a horizontal line within each box. Whiskers extend to the minimum and maximum values within 1.5 times the IQR from Q1 and Q3, respectively.

watermark confidence from FoldMark for the predicted Tir-dsDNA complex (PDB 7X5M) and HIV-1 MPER scaffold in complex with antibody Fab Ab45.2 (PDB 9BDI). For 7X5M, the plDDT is high in the core domains of Tir, such as helical regions involved in membrane anchoring and effector

binding, which are evolutionarily conserved and critical regions for function (53). Conversely, the plDDT is low for flexible loops or linkers connecting Tir's structured domains and the N- or C-termini of Tir. These regions are often disordered or lack strong evolutionary constraints, leading to low pLDDT. The watermark confidence from Folamark highlights the low-plDDT regions such as loop regions and assigns low weights to rigid helical regions. Similarly, the plDDT is high in the conserved MPER helical core and low in antibody CDR loops for 9BDI (54). The watermark confidence has the highest scores for the interface between the Fab and MPER including the CDR loops.

Therefore, FoldMark is an evolutionary information-driven watermarking framework that respects protein context. By leveraging evolutionary signals, it embeds watermarks with precision: conserved regions receive minimal noise to preserve structural integrity, ensuring biological fidelity, while flexible or intrinsically disordered regions tolerate higher noise, increasing watermark capacity and robustness. This adaptive strategy optimizes both detectability and compatibility with protein function. Figure.4 C further explores this across protein monomers, antibody interfaces, ligand interfaces, and RNA interfaces, sampling 100 tokens per structure. The negative Pearson correlations ($-0.29$, $-0.61$, $-0.45$, $-0.43$) suggest an inverse relationship between watermark confidence and structural confidence (pLDDT), indicating that FoldMark's distributional watermark subtly modulates less confident, disordered regions more heavily, aligning with evolutionary variability while maintaining detectability.

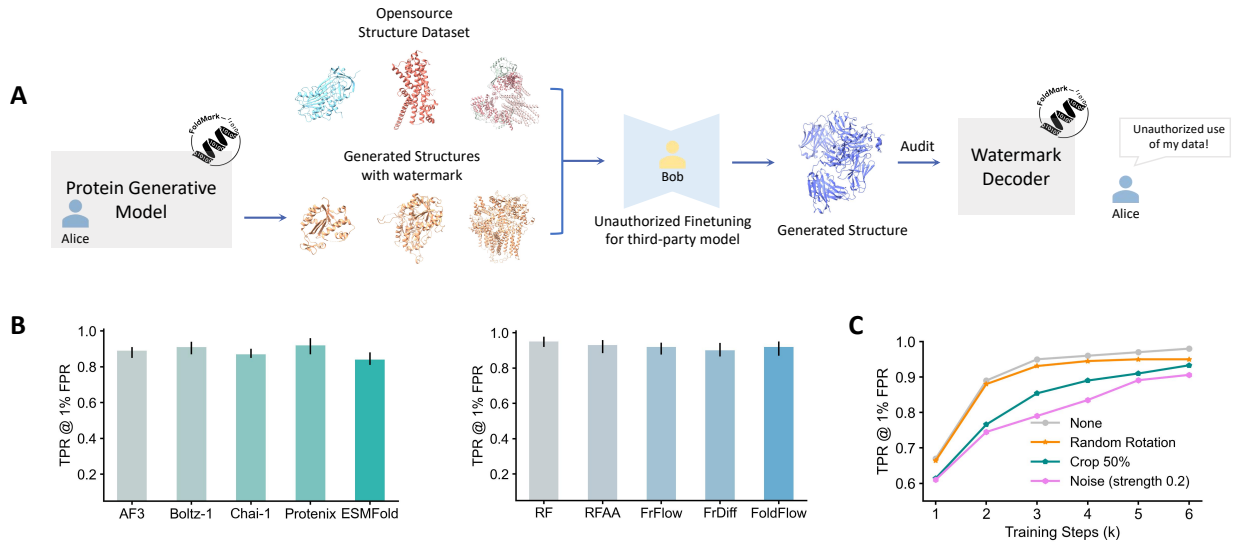



**Figure 5**: **FoldMark reliably detects unauthorized use of watermarked data in third-party models**. (**A**) FoldMark's audit pipeline detects when watermarked protein structures are misused in training third-party models. For example, an external party—Bob—might use Alice's model (e.g., AlphaFold3) to generate novel, high-quality protein structures. By combining these watermarked structures with open-source data to train his own model, Bob inadvertently incorporates Alice's watermark into the new outputs. FoldMark can then audit the generated structures, recover the embedded watermark, and thereby reveal that unauthorized watermarked data was used in training. (**B**) Detection True Positive Rate (TPR) across different models at a False Positive Rate (FPR) of just 1%. Each model generates 1,000 samples independently, with the mean and 95% confidence intervals reported. AF3: AlphaFold3, RF: RFDiffusion, RFAA: RFDiffusionAA, FrFlow: FrameFlow, FrDiff: FrameDiff. (**C**) TPR @ 1% FPR as a function of training steps in third-party models under different data transformation conditions. None: no post-processing; Random Rotation: randomly rotated the generated structure; Crop 50%: randomly crop 50% of the continuous protein chains; Noise (strength 0.2): add Gaussian noise with 0.2 strength to coordinates. AlphaFold3 is used for evaluation.

## FoldMark Detects Unauthorized Model Training with Watermarked Data

With recent advances in protein AI, protein generation as a service (PGaaS) has transformed computational biology. Yet, the availability of such services raises serious IP compliance concerns. For instance, AlphaFold3 prohibits using its data for commercial purposes or for training third-party models — a restriction that is hard to enforce.

FoldMark addresses this challenge and provides a robust audit pipeline enabling the enforcement of commercial usage restrictions, as illustrated in Figure 5 A. When the data is misused, for example, if an external party combines watermarked AlphaFold3-generated structures with open-source data to fine-tune their own model—FoldMark can recover the embedded watermark, thereby revealing unauthorized use. FoldMark effectively detects watermarks in the outputs of these models, enabling robust auditing. We simulated the auditing process by fine-tuning third-party models with a mixed dataset consisting of 30% watermarked structures and 70% original training data. In Figure. 5 B, FoldMark shows robust detection performance across various protein generative models—including AlphaFold3, Chai-1, RFDiffusionAA, and FrameFlow—achieving high True Positive Rates (TPR) at a False Positive Rate (FPR) of just 1%. Furthermore, Figure. 5 C reveals that the TPR steadily increases with additional fine-tuning steps, likely because extended fine-tuning amplifies the watermark signal's persistence in the model's output distribution, thereby enhancing FoldMark's sensitivity.

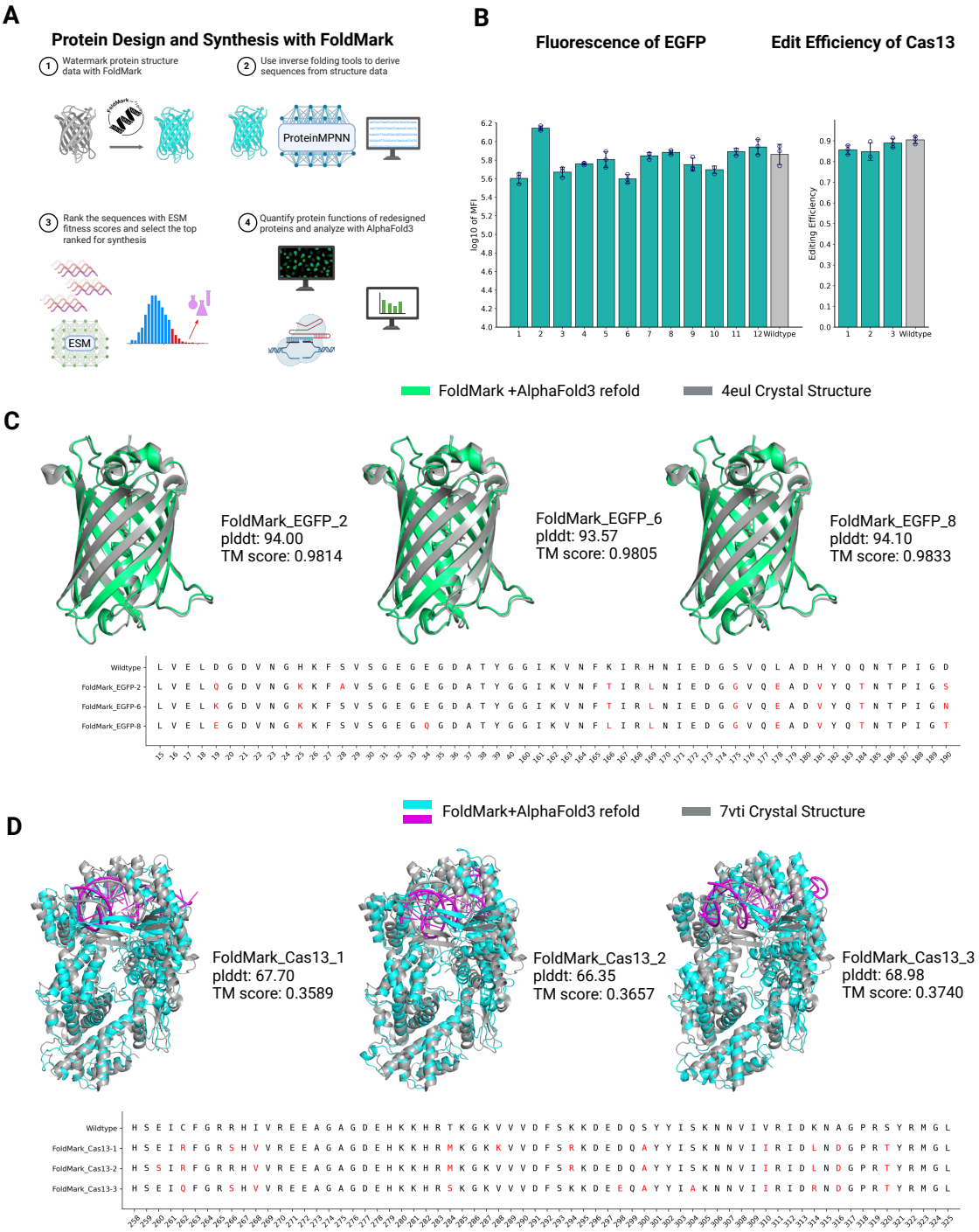



**Figure 6**: **EGFP and Cas13 redesign with FoldMark**. (**A**) The pipeline of protein design and synthesis with FoldMark: (1) The original structure is watermarked with FoldMark; (2) The protein sequences are derived with inverse folding tools; (3) The designed sequences are ranked with ESM fitness scores and the top ones are synthesized; (4) The functions of the synthesized proteins are quantified and analyzed. (Caption continues on the next page.)

**Figure 6 (continued).** (**B**) Experimental results of the designed sequences with FoldMark. Based on Kolmogorov-Smirnov (KS) tests, 11 out of 12 redesigned EGFP variants and all 3 redesigned Cas13 variants exhibit distributions of fluorescence intensity or editing efficiency statistically indistinguishable from their wildtype counterparts ($p > 0.05$). (**C**) predicted structure of FoldMark redesigned EGFP aligned with the crystal structure of 4eul. (**D**) predicted structure of FoldMark redesigned Cas13 aligned with the crystal structure of 7vti. The plddt and TM scores are presented. The mutation points in the sequences are marked in red.

## Applying FoldMark to EGFP and Cas13 Structure-based Redesign Pipeline

To demonstrate the effectiveness of FoldMark in real-world applications, FoldMark is applied to the structure-based redesign of EGFP (55) (pdb id 4eul) and Cas13 (56) (pdb id 7vti), shown in Figure.6 A. The process begins with watermarking the original protein structures using FoldMark (watermarking the loop regions of EGFP and the lid domain (57) of Cas19), followed by deriving novel protein sequences through inverse folding tools (ProteinMPNN (52)). These sequences are ranked based on ESM fitness scores (39), and the top candidates are synthesized for experimental testing. Wet-lab validation demonstrated that the functionality of nearly all redesigned EGFP and Cas13 proteins is equivalent to their wildtype counterparts (Figure 6 B). The redesigned EGFP variants retained an average of 98% fluorescence intensity, while the redesigned Cas13 variants maintained an average of 95% editing efficiency, indicating no significant deviation from wildtype performance. Moreover, Kolmogorov-Smirnov (KS) tests revealed that the fluorescence intensity distributions of 11 out of 12 redesigned EGFP variants, and the editing efficiency distributions of all 3 redesigned Cas13 variants, are statistically indistinguishable from their respective wildtype controls ($p > 0.05$). Meanwhile, the watermark detection success rates exceeding 90% (91.7% for EGFP and 100% for Cas13). Finally, in Figure. 6 C and D, Structural analyses by AlphaFold3 (3), including alignment of redesigned sequences with wildtype, verify that the structural quality is preserved while embedding the watermark by FoldMark. These results demonstrate that **FoldMark can reliably embed watermarks without compromising the original protein functions and structures in practical applications**.

# Discussion

This paper establishes the feasibility of watermarking protein generative models and their outputs through FoldMark, a novel method leveraging distributional and evolutionary principles. By pretraining a watermark encoder-decoder to embed user-specific data across residues—guided by evolutionary signals that balance noise in conserved and flexible regions—followed by watermark-conditioned fine-tuning, FoldMark preserves structural quality while enabling robust authentication and tracking. Extensive tests on models like AlphaFold3, ESMFold, RFDiffusion, and RFDiffusionAA demonstrate near-100% watermark recovery with minimal impact on structure integrity, even under post-processing and adaptive attacks. FoldMark's adaptive embedding, aligned with structural confidence (e.g., pLDDT), further bolsters its resilience, offering a powerful solution to ethical challenges such as IP infringement and unauthorized use in protein design.

There are a few limitations that we would like to address in the future. First, advanced users may leverage protein generative models beyond de novo design, employing them for tasks such as iterative structure editing, functional optimization (e.g., enhancing binding affinity or enzymatic activity), or motif scaffolding to integrate specific structural elements into larger frameworks. At present, FoldMark's watermarking technique is primarily tailored to simple structure generation and does not fully account for these complex, multi-step modifications, which may alter or obscure the embedded watermark through successive transformations. This limitation reduces its effectiveness in advanced usage scenarios where proteins undergo significant redesign or iterative refinement, as the watermark's detectability could weaken with each modification. To address this, future iterations could incorporate adaptive watermarking that takes these advanced protein editing/transformations into consideration.

Second, FoldMark will be optimized for scalability to support deployment on platforms tracing millions of users while maintaining efficiency and a lightweight design. By implementing optimized watermark embedding techniques, such as sparse or quantized watermarks, FoldMark minimizes computational overhead during protein structure generation, ensuring rapid performance even under high demand. Scalable decoding infrastructure, leveraging distributed computing, enables simultaneous processing of multiple extraction requests, avoiding bottlenecks in user tracing. Lightweight authentication protocols, like hash-based checks or partial decoding, streamline verification without

compromising reliability. Collaborating with biotech firms and cloud providers further enhances FoldMark's scalability by integrating it into existing, robust infrastructures.

Finally, while FoldMark provides a robust foundation for watermarking protein generative models, it is only one piece of a broader, systematic safeguarding framework necessary to address the ethical and safety challenges in this domain (14). Future efforts will focus on developing a multi-aspect approach that integrates FoldMark with additional techniques to ensure responsible use. For example, alignment techniques, such as reinforcement learning from human feedback (RLHF) (58) can align model outputs with ethical guidelines, mitigating risks like the generation of harmful protein designs (e.g., toxins or viral structures) or copyrighted sequences. Complementing these, training-free methods—such as guided sampling or real-time detection algorithms—offer lightweight solutions to monitor and refine generated outputs without requiring extensive model retraining (59). This comprehensive framework will combine authentication (via FoldMark), ethical enforcement (via alignment), and runtime oversight (via training-free approaches), creating a cohesive system that not only tracks and verifies protein designs but also prevents misuse.

# Methods

Figure 1 provide an overview of FoldMark. Inspired by previous works (33–35), FoldMark consists of two main stages: Watermark Encoder/Decoder Pretraining and Protein Generative Model Fine-Tuning. The pretraining stage enables the watermark encoder and decoder to learn how to embed watermark information into the structure space and accurately extract it. The fine-tuning stage equips pretrained protein generative models with watermarking capabilities while preserving their original generative performance. FoldMark is a versatile method that can be applied to various mainstream protein structure generative models. We use a diffusion-based model as an example and present the details of FoldMark below.

## Watermark Encoder/Decoder Pretraining

We first train a watermark encoder $\mathcal{W}$ and decoder $\mathcal{D}$ such that $\mathcal{D}$ can correctly retrieve the watermark message $m$ embedded by $\mathcal{W}$:

$$\mathcal{L}_{Pretrain} = \mathbb{E}_{x,m,f}[\mathcal{L}_{BCE}(\mathcal{D}(f(\mathcal{W}(x,m))), m) + \gamma \|\mathcal{W}(x,m) - x\|_2], \tag{4}$$

where $x$ represents the protein structure data and $m$ denotes the string of binary watermark code. $\gamma > 0$ is a hyperparameter to control the strength of structure adjustment for watermarking. $f$ represents a randomly selected structure transformation as data augmentation. The pool of data augmentation includes random rotation/translation, adding Gaussian noise to protein coordinates, chain permutation, and randomly cropping the protein structure. $\mathcal{L}_{BCE}(\mathcal{D}(f(\mathcal{W}(x,m))), m)$ and $\|\mathcal{W}(x,m) - x\|_2$ correspond to the CE Loss and Struct Loss in Figure 1 respectively.

For the watermark encoder/decoder architecture, we have equivariant and non-equivariant versions inspired by prior works (2, 3). The pseudo-algorithms of the water encoder/decoder are shown in the supplementary (Algorithm 1, 2, 3, and 4).

## Optional Evolutionary and Template Input Features

FoldMark Encoder accepts a wide variety of optional input features, such as genetic search, protein language model embeddings, and structural templates for better performance. More details of the use of optional input features are included in the Supplementary.

**MSA Module** Following Protenix (48), we search multiple sequence alignments (MSAs) using MMseqs2 (42) and ColabFold (60) MSA pipeline, and use MSAs from the Uniref100 (43) database for pairing (species are identified with taxonomy IDs). Retrieved MSAs results are cropped to the maximum allowed sequence length. If no hits are found, a length-1 MSA containing only the query sequence is returned.

The MSA module adopted from AlphaFold3 processes MSAs by embedding a randomly sampled subset of sequences into a representation for each token, updated per recycling iteration. It comprises 4 homogeneous blocks that iteratively refine the pair representation $z_{ij}$ alongside the MSA, using a network structure similar to the Pairformer Stack, where MSA rows are processed with row-wise attention projected solely from $z_{ij}$, omitting key-query attention to reduce computational

overhead. This design ensures all MSA-derived information flows through the pair representation, enhancing its role as the informational backbone, with updates applied via Triangular Multiplicative Update, Triangular Self-Attention, and SwiGLU-based transition blocks.

**ESM Module** By training on extensive datasets of protein sequences, protein language models such as Evolutionary Scale Modeling (ESM) are capable of capturing co-evolutionary information inherent in protein sequences. Compared with MSA, ESM eliminates the need for time-consuming sequence retrieval processes and effectively processes sequences with limited homologous data. In FoldMark, we use ESM-3B to obtain the per-residue representations for protein chains following Chai-1 (50) and add to the initialized single representations.

**Template Module** Following AlphaFold3 (3), MSAs derived from UniRef100 are converted into profile hidden Markov models using hmmbuild. These models undergo hmmsearch against a PDB sequence database with stringent parameters (–noali –F 0.1 –E 100 thresholds). Structural eligibility requires minimum length (≥10 residues) and query coverage (>10%), while excluding sequences with >95% identity to the target. Template coordinates are extracted from mmCIF files using sequence alignment validation through Kalign v0.2.0. The system prioritizes templates by e-value ranking, retaining top 20 candidates. During modeling, maximum 4 templates are utilized through randomized selection in training ($k \sim \min(\text{Uniform}[0, n], 4)$) versus deterministic top-4 selection in inference.

In FoldMark Encoder, template embedding integrates all raw template features into a pair representation, which is then processed with existing pair representation $z_{ij}$. This enables model to focus on specific regions of the template, guided by its current understanding of the structure.

## Watermark Confidence and Weighted Prediction

In the FoldMark decoder, we extract both a confidence score and a weighted watermark code for robust watermark prediction. The confidence score denoted as $p_{\text{confidence}} = \text{Linear}(a_i)$, is computed by applying a linear transformation to the obtained token representations $a_i$. Concurrently, a per-token watermark code $m_i = \text{Sigmoid}(\text{Linear}(a_i))$ is derived through a separate linear transformation followed by a sigmoid activation, ensuring a bounded output between 0 and 1. The final prediction,

$m_{\text{pred}} = \sum_i \text{Softmax}(p_{\text{confidence}})_i \cdot m_i$, combines these by weighting each $m_i$ with the corresponding softmax-normalized confidence scores, effectively producing a weighted average that prioritizes contributions based on their predicted reliability. This mechanism enhances the decoder's ability to integrate watermark information with confidence-driven precision, improving the robustness of the generated outputs.

## Protein Generative Model Fine-Tuning

Instead of fine-tuning all the parameters of the generative model, we selectively fine-tune part of the protein generative model with LoRA and the watermark decoder as shown in Figure 1 D. The other parameters, such as the reference pretrained model are kept unchanged. We discuss the details of watermark module in the next subsection.

Here, we take the diffusion-based protein generative model (e.g., AlphaFold3 (3) and RFDiffusion (4)) as an example to construct the fine-tuning loss. The diffusion model typically involves two critical components known as the forward and backward process, where the forward process gradually noises the original protein structure $x_0$ into $x_t$ for $t \in \{1, \cdots, T\}$ and the model learns to predict the original structure $\epsilon_\theta(x_t)$ based on $x_t$. There are two losses in the fine-tuning: the consistency loss for regularization and the message retrieval loss to encourage correct watermark retrieval. In the consistency loss $\mathcal{L}_c$, the prediction of the fine-tuned model is compared with the original pretrained model so that the finetuned model weights will not deviate too much from the original ones. For the watermark retrieval loss $\mathcal{L}_m$, we take a single reverse step with respect to $x_t$ to obtain $\hat{x}_t = (x_t - \sqrt{1-\bar{\alpha}_t}\epsilon_\theta(x_t))/\sqrt{\bar{\alpha}_t}$, and then feed it into the decoder to predict the watermark code. In sum, we incorporate both optimization objectives above and formulate the consistency-preserving fine-tuning loss as:

$$\mathcal{L}_{Finetune} = \mathbb{E}_{x,t,m}[\mathcal{L}_c(\epsilon_\theta(\hat{x}_t), \epsilon_{\theta_{ref}}(x_t)) + \eta \cdot \frac{t-T}{T}\mathcal{L}_m(\mathcal{D}(\hat{x}_t), m)], \tag{5}$$

where $\eta$ controls the trade-off between consistency loss $\mathcal{L}_c$ and watermark retrieval loss $\mathcal{L}_m$. We place an additional weight $\frac{t-T}{T}$ for the retrieval loss because the generated structure contains more information of watermark as $t \to 0$ and we observe better performance in experiments.

## Watermark-conditioned LoRA

Inspired by previous works in image domains (e.g., AquaLoRA (35) and EA-LoRA (61)), we use watermark-conditioned LoRA to save the computation costs of fine-tuning and flexibly embed watermark information in the generation process. The computation formula for Watermark-conditioned LoRA in FoldMark can be expressed as:

$$\Delta \mathbf{W}(m) = \mathbf{G}(m) \odot (\mathbf{A} \times \mathbf{B}), \tag{6}$$

where $\mathbf{A} \in \mathbb{R}^{n \times r}$ and $\mathbf{B} \in \mathbb{R}^{r \times m}$ are the low-rank matrices, and $\mathbf{G} \in \mathbb{R}^{n}$ is the gating vector derived from the watermark code. The operator $\odot$ denotes element-wise multiplication, where $\mathbf{G}$ modulates the rows of $\mathbf{A} \times \mathbf{B}$. This formulation maintains efficiency while allowing flexible incorporation of watermark information.

To input the watermark information into the fine-tuned model, we utilize an adapter layer that converts a watermark code of length $l$ into a gating vector $\mathbf{G}$. Specifically, the watermark code $m = \{b_0, b_1, \ldots, b_l\}$ is passed through a linear transformation defined as:

$$\mathbf{G}(m) = \mathbf{W}_g \cdot m + \mathbf{b}_g, \tag{7}$$

where $\mathbf{W}_g \in \mathbb{R}^{n \times l}$ is the linear transformation matrix, and $\mathbf{b}_g \in \mathbb{R}^{n}$ is the bias vector. Here, $b_i \in \{0, 1\}$ represents the binary state of the $i$-th bit in the watermark code. The gating vector $\mathbf{G}$ modulates the LoRA weight updates by scaling the rows of the low-rank update $\mathbf{A} \times \mathbf{B}$.

During the generation process, when embedding a watermark into the model, we compute the gating vector $\mathbf{G}$ based on the watermark code. The resulting LoRA weight update $\Delta\mathbf{W}$ is added to the original model weights $\mathbf{W}$ to produce the watermarked model weights $\mathbf{W}_{\text{watermarked}}$:

$$\mathbf{W}_{\text{watermarked}} = \mathbf{W} + \alpha \Delta \mathbf{W}(m), \tag{8}$$

where $\alpha$ is a scaling factor controlling the impact of the watermark on the model weights. FoldMark applies LoRA to all linear and attention layers in structural prediction modules. In contrast, AquaLoRA applies LoRA to linear and convolutional layers in U-Net.

## Training Dataset

**Dataset for Sturcture Prediction Models** For training of watermarked biomolecule structure prediction models (AlphaFold3, Protenix, Boltz-1, and Chai-1), we use all PDB (44) structures

released before 2021-09-30 (same training cut-off date as AlphaFold3 (3)) and with a resolution of at least 9 Åfollowing (3, 48). We parse the Biological Assembly 1 from these structures using their mmCIF file. We use the reference sequence for each polymer chain and align it to the residues available in the structure. For ligands, we use the Chemical Component Dictionary (CCD) dictionary to create conformers and match atoms from the structure. We remove atoms under the following conditions: (1) the ligand is covalently bound, and (2) the atom does not appear in the PDB structure. Finally, we follow the same data-cleaning process as AlphaFold3, which includes the ligand exclusion list, the minimum number of resolved residues, and the removal of clashing chains. The total number of PDB structures is around 220k. For the training of watermarked ESMFold, we only use the protein monomer subset.

**Dataset for Unconditional Structure Generative Models** We trained watermark encoders/decoders and fine-tuned protein generative models using the monomers from the PDB (44) dataset, focusing on proteins ranging in length from 60 to 512 residues with a resolution better than 5 Å. This initial dataset consisted of 23,913 proteins. Following previous work (46), we refined data by applying an additional filter to include only proteins with high secondary structure content. For each monomer, we used Dictionary of Secondary Structure of Proteins (DSSP) (62) to analyze secondary structures, excluding those with over 50% loops. This filtering process resulted in 20,312 proteins.

## Evaluation Dataset

**PoseBusters V2** We utilize the PoseBusters Benchmark Set Version 2 (PoseBusters V2) (63) to evaluate protein–ligand interactions. This dataset comprises 308 structures. To ensure a fair comparison with AlphaFold3, we adopt their methodology (3) to exclude chains that clash with the ligand of interest, specifically in the following PDB entries: 7O1T, 7PUV, 7SCW, 7WJB, 7ZXV, and 8AIE. Similarly, for PDB entry 8F4J, we retain the protein chains within 20 Å of the ligand.

**Low Homology Recent PDB Set** We adhere to AlphaFold3 (3), Protenix (48), and Chai-1 (50) to compile the Low Homology Recent PDB set. Homology is defined as sequence identity to sequences in the training set and is measured by template search. Our evaluation set is constructed from a temporal split of PDB entries released between May 1, 2022, and January 12, 2023. All these

structures were released after the data in the training sets. We additionally restricted non-Nuclear Magnetic Resonance (NMR) structures to those with a resolution better than 4.5 Å. We restricted to examples that had 2048 or fewer tokens and less than 20 chains. Moreover, we removed monomers and interfaces with substantial homology to the training set:

- Monomers with 40% or greater sequence identity to the training set are removed.
- Polymer-polymer interfaces where both polymers have greater than 40% sequence identity to two chains in the same complex in the training set are filtered out.
- Polymer-peptide interfaces where the non-peptide entity has 40% or greater sequence identity to the training set are removed.

We additionally cluster the above low-homology PDB set. Individual polymer chains were clustered at 40% sequence identity for proteins with more than 9 residues, and 100% sequence identity for proteins with 9 or fewer residues and for all nucleic acids.

To study antibody-antigen interface prediction, we filter the low-homology recent PDB set to complexes that contain at least one protein–protein interface where one of the protein chains is in one of the two largest PDB chain clusters (these clusters are representative of antibodies).

We apply quality filters to the bonded ligands and glycosylation dataset (as does PoseBusters): we include only ligands with high-quality experimental data (`ranking_model_fit` $> 0.5$, according to the RCSB (64) structure validation report, that is, X-ray structures with a model quality above the median). As with the PoseBusters set, the bonded ligands and glycosylation datasets are not filtered by homology to the training dataset.

Finally, we obtained 927 protein-protein clusters, 64 protein-antibody clusters, 282 protein monomer clusters, 66 bonded ligand clusters, and 28 Glycosylation clusters. We note that while the above methodology for constructing the low homology recent PDB ret follows the procedure laid out in previous works, we are not able to match exactly the evaluation set used in AlphaFold3 (3). As a result, corresponding experimental values on protein-protein interactions, protein-antibody interactions, and protein-RNA interactions may deviate a little from previous works.

**Filtered Single Chain Set** Based on the clustering of the low-homology PDB set, we select the representative single chain from each cluster, prioritizing chains with a high number of resolved

alpha carbons and lengths between 100 and 800 residues. Following AlphaFold2 (2), the filtered single chain set is used to benchmark protein structure prediction models in a general manner. The final number of selected protein chains is 500.

**Nucleic Acids Evaluation Set** For RNA along data, we extracted 8 publicly available targets from CASP15 (65) for evaluation: R1116/8S95, R1117/8FZA, R1126, R1128/8BTZ, R1136/7ZJ4, R1138/[7PTK/7PTL], R1189/7YR7 and R1190/7YR6.

For Protein-Nucleic Acids datasets for evaluation, we screened PDB IDs listed in Table 14 of the AlphaFold3 supplementary materials to create the nucleic acid evaluation set. We also omitted systems containing both DNA and RNA simultaneously. From there, we retained only structures with either (i) one RNA chain and one protein chain or (ii) two DNA chains (forming double-stranded DNA) and one protein chain. Next, we filtered out systems with non-standard residues in the DNA. The final dataset matched the counts reported in AlphaFold3 Figure 1, consisting of 25 RNA-protein structures and 38 dsDNA-protein structures.

## Metrics

**Watermark Bit Accuracy** Watermark bit accuracy quantifies the proportion of correctly extracted bits from a watermarked structure, serving as a key metric for evaluating watermarking models. To compute this accuracy, compare the extracted binary bit sequence to the original embedded bit sequence and calculate the ratio of matching bits.

**scRMSD and scTM** self-consistency $C_\alpha$-RMSD (scRMSD, lower is better) and TM-score (scTM, higher is better) are used to evaluate the quality of generated structures following previous works (4, 5, 46). They measure whether there exists an amino acid sequence that folds to the generated structure and the consistency of sequence-structure pairs. In this work, we use ProteinMPNN (52) at the temperature of 0.1 to generate 8 sequences for AlphaFold3 (3) to predict structures. The final sequence-structure pair is the original sampled structure along with the ProteinMPNN sequence with minimum scRMSD or highest scTM. Generally, a generated structure is called high-quality or designable if scRMSD $< 2.0$ Å.

**lDDT** To assess folding accuracy for protein and nucleotide monomers, we employ the Local Difference Delta Test (LDDT) (66) computed using specific reference atoms. For proteins, LDDT is calculated on C$\alpha$ atoms with an inclusion radius of 15 Å. For nucleotides, it is evaluated on C1′ atoms with an inclusion radius of 30 Å. To evaluate protein-nucleotide interfaces, we use the interface LDDT (iLDDT), calculated over C$\alpha$ and C1′ atoms within the interface.

**DockQ** We measure the quality of protein-protein interfaces using DockQ (67). If the complex contains multiple identical entities, the optimal assignment of the predicted units to the ground truth units is found by either an exhaustive search over all permutations (for groups up to 8 members) or a simulated annealing optimization (for larger groups) following AlphaFold3 (3).

**C$\alpha$ r.m.s.d. at 95% coverage** We also report accuracies using the r.m.s.d.$_{95}$ (C$\alpha$ r.m.s.d. at 95% coverage). To calculate this, we perform five iterations of the following steps: (1) a least-squares alignment of the predicted structure with the PDB structure, based on the currently selected C$\alpha$ atoms (using all C$\alpha$ atoms in the first iteration), and (2) selection of the 95% of C$\alpha$ atoms with the lowest alignment error. The r.m.s.d. of the atoms selected in the final iteration is defined as the r.m.s.d.$_{95}$. This metric is more robust against apparent errors that may arise from crystal structure artifacts, though in some cases, the excluded 5% of residues may include genuine modeling errors.

## Implementations.

Our FoldMark model is pretrained for 20 epochs and fine-tuned for 10 epochs with Adam (68) optimizer, where the learning rate is 0.0001, and the max batch size is 64. We use the batching strategy from FrameDiff (69) of combining proteins with the same length into the same batch to remove extraneous padding. In the LoRA, the rank is set as 16 in the default setting. $\gamma$ and $\eta$ are set as 2. $\alpha$ is set to 1.0. To minimally influence the generation quality of the pretrained generative models while successfully embedding the watermark, we take an early stop strategy: the training is stopped when the validation watermark bit accuracy is larger than 90%. The evaluation set is constructed by subsampling the training set and the checkpoint frequency is per 500 steps. It takes around 24 hours to finish the whole FoldMark training process (FoldMark pretraining+finetuning) on 1 Tesla H100 80G GPU for AlphaFold3. More hyperparameter settings are listed in Table. S3

and S4. For each PDB entry, we follow AlphaFold3's inference setup (3), generating 25 predictions using 5 model seeds, with each seed producing 5 diffusion samples. The predictions are ranked using confidence scores.

## Watermark Baselines.

To the best of our knowledge, FoldMark is the first watermarking method specifically designed for protein structure generative models. For comparison, we adapted two state-of-the-art watermarking methods originally developed for image generation: WaDiff (34) [1] and AquaLoRA (35) [2]. The adapted baselines use their loss functions, training strategy, and part of the watermark model design. Both baseline models were designed for image diffusion models, such as Stable Diffusion (23). Since most protein generative models are also diffusion-based, we applied default hyperparameters from the original works.

## Implementations of Protein Generative Models

FoldMark is applied for broad state-of-the-art protein structure generative models, including AlphaFold3 (3), Protenix (48), Boltz-1 (49), Chai-1 (50), and ESMFold (39) for structure prediction, and RFDiffusion (4), RFDiffusionAA (40), FrameDiff (46), FrameFlow (45), and FoldFlow2 (47) for *de novo* structure generation. We use their open-source pretrained checkpoints and default hyperparameter settings for implementations. More introductions and implementation details are included in the supplementary.

## Protocols of Wet Experiments

### Watermark Variant Vector Construction

Gene fragments containing watermarked EGFP and truncated Cas13bt3 sequences were synthesized by Twist Biosciences and amplified using Phanta Flash Master Mix. Amplified fragments were gel-purified (NEB) and inserted downstream of the pcDNA3.1 vector backbone by Gibson Assembly using NEBuilder HiFi DNA Assembly Master Mix (NEB). Ligated plasmid products were

[1] https://github.com/rmin2000/WaDiff

[2] https://github.com/Georgefwt/AquaLoRA

transformed into competent Stbl3 Chemically Competent *E. coli* (Zymo Research) and validated by Sanger sequencing (Azenta).

### Mammalian Cell Culture

HEK293T cells (ATCC) were cultured in complete Dulbecco's Modified Eagle's Medium (Gibco, 11965-092) supplemented with 10 % (v/v) fetal bovine serum (Gemini Bio, 100-106-500) and 1 % penicillin-streptomycin (Gibco, 15-140-122) at 37 °C under 5 % $CO_2$.

### Evaluation of Watermarked EGFP Variants

HEK293T cells were seeded into 96-well plates (Corning) at a density of $2.5 \times 10^4$ cells per well 24 hours prior to transfection. On the day of transfection, 50 ng of watermarked EGFP plasmids with guide RNA variants were transfected using PEIMAX (Polysciences, 24765). At 24 hours post-transfection, EGFP expression levels were measured by flow cytometry using a CytoFLEX analyzer (Beckman).

### Evaluation of Watermarked Cas13bt3 Variants

HEK293T cells were seeded into 96-well plates (Corning) at a density of $2.5 \times 10^4$ cells per well 24 hours prior to transfection. On the day of transfection, 10 ng of reporter plasmid (a bicistronic construct consisting of constitutively active mCherry fused with a downstream mutated EGFP harboring a premature stop codon W57* via a self-cleaving T2A peptide), 45 ng of Cas13bt3 guide RNA expressing plasmid (spacer: 5'-GGGTGGGCCAGGGCACGGGCAGCTTGCCGG-3'), and 45 ng of Cas13bt3 watermarked variant expressing plasmids were transfected using PEIMAX (Polysciences, 24765). At 48 hours post-transfection, EGFP and mCherry expression levels were measured by flow cytometry using a CytoFLEX analyzer (Beckman).

# References and Notes


1. C. Stokel-Walker, R. Van Noorden, What ChatGPT and generative AI mean for science. *Nature* **614** (7947), 214–216 (2023).

2. J. Jumper, *et al.*, Highly accurate protein structure prediction with AlphaFold. *Nature* **596** (7873), 583–589 (2021).

3. J. Abramson, *et al.*, Accurate structure prediction of biomolecular interactions with AlphaFold 3. *Nature* **630** (8016), 493–500 (2024).

4. J. L. Watson, *et al.*, De novo design of protein structure and function with RFdiffusion. *Nature* **620** (7976), 1089–1100 (2023).

5. J. B. Ingraham, *et al.*, Illuminating protein space with a programmable generative model. *Nature* **623** (7989), 1070–1078 (2023).

6. Y. Liu, *et al.*, De novo protein design with a denoising diffusion network independent of pretrained structure prediction models. *Nature Methods* **21** (11), 2107–2116 (2024).

7. T. N. Committee, The Nobel Prize in Chemistry 2024 - Press Release (2024), accessed: 2024-10-15.

8. Y. Bengio, S. Mindermann, D. Privitera, International AI Safety Report 2025 (2025).

9. D. Bloomfield, *et al.*, AI and biosecurity: The need for governance. *Science* **385** (6711), 831–833 (2024).

10. D. Baker, G. Church, Protein design meets biosecurity. *Science* **383** (6681), 349–349 (2024).

11. Y. Wang, R. N. Pruitt, T. Nuernberger, Y. Wang, Evasion of plant immunity by microbial pathogens. *Nature Reviews Microbiology* **20** (8), 449–464 (2022).

12. D. Jurėnas, N. Fraikin, F. Goormaghtigh, L. Van Melderen, Biology and evolution of bacterial toxin–antitoxin systems. *Nature Reviews Microbiology* **20** (6), 335–350 (2022).

13. C. W. Tan, *et al.*, A SARS-CoV-2 surrogate virus neutralization test based on antibody-mediated blockage of ACE2–spike protein–protein interaction. *Nature biotechnology* **38** (9), 1073–1078 (2020).

14. M. Wang, *et al.*, A call for built-in biosecurity safeguards for generative AI tools. *Nature Biotechnology* pp. 1–3 (2025).

15. K. Tjandra, Built-in safeguards might stop AI from designing bioweapons (2025), accessed: 2025-05-05.

16. A. Jo, The promise and peril of generative AI. *Nature* **614** (1), 214–216 (2023).

17. Z. Epstein, *et al.*, Art and the science of generative AI. *Science* **380** (6650), 1110–1111 (2023).

18. B. Meskó, E. J. Topol, The imperative for regulatory oversight of large language models (or generative AI) in healthcare. *NPJ digital medicine* **6** (1), 120 (2023).

19. AlphaFold Terms of Use (2024), accessed: 2024-10-15.

20. A. S. Bergman, *et al.*, Guiding the release of safer E2E conversational AI through value sensitive design, in *Proceedings of the 23rd Annual Meeting of the Special Interest Group on Discourse and Dialogue* (Association for Computational Linguistics) (2022).

21. E. Mitchell, Y. Lee, A. Khazatsky, C. D. Manning, C. Finn, Detectgpt: Zero-shot machine-generated text detection using probability curvature, in *International Conference on Machine Learning* (PMLR) (2023), pp. 24950–24962.

22. J. Wu, J. Guo, B. Hooi, Fake News in Sheep's Clothing: Robust Fake News Detection Against LLM-Empowered Style Attacks, in *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (2024), pp. 3367–3378.

23. R. Rombach, A. Blattmann, D. Lorenz, P. Esser, B. Ommer, High-resolution image synthesis with latent diffusion models, in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2022), pp. 10684–10695.

24. J. Betker, *et al.*, Improving image generation with better captions **2** (3), 8 (2023).

25. M. Westerlund, The emergence of deepfake technology: A review. *Technology innovation management review* **9** (11) (2019).

26. T. Zhang, Deepfake generation and detection, a survey. *Multimedia Tools and Applications* **81** (5), 6259–6276 (2022).

27. J. Kirchenbauer, *et al.*, A watermark for large language models, in *International Conference on Machine Learning* (PMLR) (2023), pp. 17061–17084.

28. A. Liu, *et al.*, An unforgeable publicly verifiable watermark for large language models, in *The Twelfth International Conference on Learning Representations* (2023).

29. R. Zhang, S. S. Hussain, P. Neekhara, F. Koushanfar, {REMARK-LLM}: A Robust and Efficient Watermarking Framework for Generative Large Language Models, in *33rd USENIX Security Symposium (USENIX Security 24)* (2024), pp. 1813–1830.

30. A. Liu, *et al.*, A survey of text watermarking in the era of large language models. *ACM Computing Surveys* (2024).

31. S. Dathathri, *et al.*, Scalable watermarking for identifying large language model outputs. *Nature* **634** (8035), 818–823 (2024), doi:10.1038/s41586-024-08025-4.

32. Y. Chen, *et al.*, Enhancing Biosecurity with Watermarked Protein Design. *bioRxiv* pp. 2024–05 (2024).

33. P. Fernandez, G. Couairon, H. Jégou, M. Douze, T. Furon, The stable signature: Rooting watermarks in latent diffusion models, in *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2023), pp. 22466–22477.

34. R. Min, S. Li, H. Chen, M. Cheng, A watermark-conditioned diffusion model for ip protection. *ECCV* (2024).

35. W. Feng, *et al.*, AquaLoRA: Toward White-box Protection for Customized Stable Diffusion Models via Watermark LoRA, in *ICML* (2024).

36. Z. Yang, *et al.*, Gaussian Shading: Provable Performance-Lossless Image Watermarking for Diffusion Models, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2024), pp. 12162–12171.

37. M. Van Kempen, *et al.*, Fast and accurate protein structure search with Foldseek. *Nature biotechnology* **42** (2), 243–246 (2024).

38. W. Lu, *et al.*, DynamicBind: Predicting ligand-specific protein-ligand complex structure with a deep equivariant generative model. *Nature Communications* **15** (1), 1071 (2024).

39. Z. Lin, *et al.*, Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science* **379** (6637), 1123–1130 (2023).

40. R. Krishna, *et al.*, Generalized biomolecular modeling and design with RoseTTAFold All-Atom. *Science* p. eadl2528 (2024).

41. E. J. Hu, *et al.*, Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685* (2021).

42. M. Steinegger, J. Söding, MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nature biotechnology* **35** (11), 1026–1028 (2017).

43. B. E. Suzek, *et al.*, UniRef clusters: a comprehensive and scalable alternative for improving sequence similarity searches. *Bioinformatics* **31** (6), 926–932 (2015).

44. H. M. Berman, *et al.*, The protein data bank. *Nucleic acids research* **28** (1), 235–242 (2000).

45. J. Yim, *et al.*, Fast protein backbone generation with SE (3) flow matching. *arXiv preprint arXiv:2310.05297* (2023).

46. J. Yim, *et al.*, SE (3) diffusion model with application to protein backbone generation. *arXiv preprint arXiv:2302.02277* (2023).

47. A. J. Bose, *et al.*, SE(3)-Stochastic Flow Matching for Protein Backbone Generation, in *The International Conference on Learning Representations (ICLR)* (2024).

48. x. chen, *et al.*, Protenix-Advancing Structure Prediction Through a Comprehensive AlphaFold3 Reproduction. *bioRxiv* pp. 2025–01 (2025).

49. J. Wohlwend, *et al.*, Boltz-1: Democratizing Biomolecular Interaction Modeling. *bioRxiv* pp. 2024–11 (2024).

50. C. D. team, *et al.*, Chai-1: Decoding the molecular interactions of life. *BioRxiv* pp. 2024–10 (2024).

51. W. Nie, *et al.*, Diffusion Models for Adversarial Purification, in *International Conference on Machine Learning* (PMLR) (2022), pp. 16805–16827.

52. J. Dauparas, *et al.*, Robust deep learning–based protein sequence design using ProteinMPNN. *Science* **378** (6615), 49–56 (2022).

53. D. Yu, *et al.*, TIR domains of plant immune receptors are 2, 3-cAMP/cGMP synthetases mediating cell death. *Cell* **185** (13), 2370–2386 (2022).

54. R. Ray, *et al.*, Affinity gaps among B cells in germinal centers drive the selection of MPER precursors. *Nature Immunology* **25** (6), 1083–1096 (2024).

55. J. A. Arpino, P. J. Rizkallah, D. D. Jones, Crystal structure of enhanced green fluorescent protein to 1.35 Å resolution reveals alternative conformations for Glu222 (2012).

56. R. Nakagawa, *et al.*, Structure and engineering of the minimal type VI CRISPR-Cas13bt3. *Molecular cell* **82** (17), 3178–3192 (2022).

57. H. Yang, D. J. Patel, Structures, mechanisms and applications of RNA-centric CRISPR–Cas13. *Nature chemical biology* **20** (6), 673–688 (2024).

58. L. Ouyang, *et al.*, Training language models to follow instructions with human feedback. *Advances in neural information processing systems* **35**, 27730–27744 (2022).

59. Z. Xu, *et al.*, Safedecoding: Defending against jailbreak attacks via safety-aware decoding. *arXiv preprint arXiv:2402.08983* (2024).

60. M. Mirdita, *et al.*, ColabFold: making protein folding accessible to all. *Nature methods* **19** (6), 679–682 (2022).

61. D. Lin, Y. Li, B. Tondi, B. Li, M. Barni, An Efficient Watermarking Method for Latent Diffusion Models via Low-Rank Adaptation. *arXiv preprint arXiv:2410.20202* (2024).

62. W. Kabsch, C. Sander, Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers: Original Research on Biomolecules* **22** (12), 2577–2637 (1983).

63. M. Buttenschoen, G. M. Morris, C. M. Deane, PoseBusters: AI-based docking methods fail to generate physically valid poses or generalise to novel sequences. *Chemical Science* **15** (9), 3130–3139 (2024).

64. P. W. Rose, *et al.*, The RCSB protein data bank: integrative view of protein, gene and 3D structural information. *Nucleic acids research* p. gkw1000 (2016).

65. A. Elofsson, Progress at protein structure prediction, as seen in CASP15. *Current opinion in structural biology* **80**, 102594 (2023).

66. V. Mariani, M. Biasini, A. Barbato, T. Schwede, lDDT: a local superposition-free score for comparing protein structures and models using distance difference tests. *Bioinformatics* **29** (21), 2722–2728 (2013).

67. S. Basu, B. Wallner, DockQ: a quality measure for protein-protein docking models. *PloS one* **11** (8), e0161879 (2016).

68. D. P. Kingma, J. Ba, Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

69. J. Yim, *et al.*, Improved motif-scaffolding with SE (3) flow matching. *arXiv preprint arXiv:2401.04082* (2024).

70. A. Vaswani, *et al.*, Attention is all you need. *Advances in neural information processing systems* **30** (2017).

71. J. L. Ba, J. R. Kiros, G. E. Hinton, Layer normalization. *arXiv preprint arXiv:1607.06450* (2016).

72. T. UniProt Consortium, UniProt: the universal protein knowledgebase. *Nucleic acids research* **46** (5), 2699–2699 (2018).

73. K. Tunyasuvunakool, *et al.*, Highly accurate protein structure prediction for the human proteome. *Nature* **596** (7873), 590–596 (2021).

74. L. Richardson, *et al.*, MGnify: the microbiome sequence data analysis resource in 2023. *Nucleic acids research* **51** (D1), D753–D759 (2023).

75. I. Kalvari, *et al.*, Rfam 14: expanded coverage of metagenomic, viral and microRNA families. *Nucleic acids research* **49** (D1), D192–D200 (2021).

76. RNAcentral 2021: secondary structure integration, improved sequence search and new member databases. *Nucleic acids research* **49** (D1), D212–D220 (2021).

77. E. W. Sayers, *et al.*, Database resources of the National Center for Biotechnology Information in 2023. *Nucleic acids research* **51** (D1), D29 (2022).

78. W. Kabsch, A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography* **32** (5), 922–923 (1976).

79. G. Ahdritz, *et al.*, OpenProteinSet: Training data for structural biology at scale. *Advances in Neural Information Processing Systems* **36**, 4597–4609 (2023).

80. P. Fernandez, A. Chaffin, K. Tit, V. Chappelier, T. Furon, Three bricks to consolidate watermarks for large language models, in *2023 IEEE International Workshop on Information Forensics and Security (WIFS)* (IEEE) (2023), pp. 1–6.

81. Y. Lu, A. Liu, D. Yu, J. Li, I. King, An entropy-based text watermarking detection method. *arXiv preprint arXiv:2403.13485* (2024).

82. J. Kirchenbauer, *et al.*, On the reliability of watermarks for large language models. *arXiv preprint arXiv:2306.04634* (2023).

83. Y. Liu, Z. Li, M. Backes, Y. Shen, Y. Zhang, Watermarking diffusion model. *arXiv preprint arXiv:2305.12502* (2023).

# Acknowledgements

We thank the authors of WaDiff (34) and AquaLoRA (35) for discussions, guidance, feedback, and assistance. We thank Professor Qi Liu from University of Science and Technology of China for discussion and establishing connections.

**Competing interests:** The authors declare no competing interests.

**Author contributions:** Z.X.Z., and M.D.W. designed the research, Z.X.Z., R.F.J., G.X.X., and X.T.W., conducted the experiments, Z.X.Z., G.X.X., L.C., and M.D.W. analyzed the results. Z.X.Z., R.F.J., L.C., and M.D.W. wrote the manuscript. All authors reviewed the manuscript.

**Data an materials availability:** The data and source code of this study is freely available at GitHub (`https://github.com/zaixizhang/FoldMark`) to allow for replication of the results of this study. The HuggingFace demo website for FoldMark is at `https://huggingface.co/spaces/Zaixi/FoldMark`.

## Supplementary materials

Materials and Methods

Supplementary Text

Figs. S1 to S5

Tables S1 to S4

# Supplementary Materials for FoldMark

Zaixi Zhang, Ruofan Jin, Guangxue Xu, Xiaotong Wang,

Le Cong*, Mengdi Wang*

*Corresponding author. Emails: lecong@stanford.edu, mengdiw@princeton.edu

**This PDF file includes:**

Materials and Methods

Supplementary Text

Figures S1 to S5

Tables S1 to S4

## Materials and Methods

### Watermark Encoder-Decoder Architecture

**Euivariant Architecture** In the FoldMark Equivariant Encoder/Decoder, we adopt a neural network architecture modified from the FrameDiff (46). This architecture consists of Invariant Point Attention from AlphaFold2 (2) and transformer blocks (70). Here, for ease of expression, we use superscripts to refer to the network layer and subscripts to indexes or variables. In the complex structure, each residue/ligand atom is represented by one embedding $h \in \mathbb{R}^{D_x}$ and a frame $T \in SE(3)$. Overall, at the $\ell$-th layer of the network, $s^\ell = [s_1^\ell, \ldots, s_N^\ell] \in \mathbb{R}^{N \times D_x}$ are all the node embeddings where $s_i^\ell$ is the embedding for the $i$-th node and $N$ is the total number of nodes; $T^\ell = [T_1^\ell, \ldots, T_N^\ell] \in SE(3)^N$ is the frames of every node at the $\ell$-th layer; $\boldsymbol{z}^\ell \in \mathbb{R}^{N \times N \times D_z}$ are edge embeddings with $z_{ij}^\ell$ being the embedding of the edge between residues $i$ and $j$. In the following paragraphs, we introduce the details of feature initialization, node/edge update, backbone update, and sidechain torsion angle update for watermark encoding and decoding. The pseudo algorithms of Equivariant FoldMark Encoder and Decoder are shown in Algorithm 1 and 2.

**Feature Initialization.** In Equivariant Encoder/Decoder, the node embeddings are first initialized with residue indices while edge embeddings additionally get relative sequence distances. Initial embeddings at layer 0 for residues $i, j$ are obtained with an MLP and sinusoidal embeddings $\phi(\cdot)$ (70) over the features. Following (46), the binned displacement of two $C_\alpha$ is given as,

$$\text{disp}_{ij} = \sum_{k=1}^{N_{\text{bins}}} 1\{\tilde{x}^i - \tilde{x}^j < d_k\}, \tag{S1}$$

where $d_1, \ldots, d_{N_{\text{bins}}}$ are linspace(0, 20) are equally spaced bins between 0 and 20 angstroms. In our experiments we set $N_{\text{bins}} = 22$. The initial node representation is initialized with watermark information:

$$s_i^0 = \text{MLP}(\phi(i)) + \text{Linear}(m), \quad h_i^0 \in \mathbb{R}^{D_h}, \tag{S2}$$

$$z_{ij}^0 = \text{MLP}(\phi(i), \phi(j), \phi(j-i), \phi(\text{disp}_{ij})), \quad z_{ij}^0 \in \mathbb{R}^{D_z}, \tag{S3}$$

where $D_h, D_z$ are node and edge embedding dimensions. If the side chains are considered in the watermarked protein generative models, we may optionally encode side chain angles. We use the input structure to initialize the original frames according to AlphaFold2 (2).

**Node Update.** The process of node update is shown below. Invariant Point Attention (IPA) is from AlphaFold2 (2). No weight sharing is performed across layers. We use the vanilla Transformer from (70). We use Multi-Layer Perceptrons (MLP) with 3 Linear layers, ReLU activation, and LayerNorm (71) after the final layer.

$$s_{\text{ipa}} = \text{LayerNorm}(\text{IPA}(s^\ell, z^\ell, T^\ell) + s^\ell), \quad s_{\text{ipa}} \in \mathbb{R}^{N \times D_h} \tag{S4}$$

$$s_{\text{skip}} = \text{Linear}(s_0), \quad s_{\text{skip}} \in \mathbb{R}^{N \times D_{\text{skip}}} \tag{S5}$$

$$s_{\text{in}} = \text{concat}(s_{\text{ipa}}, h_{\text{skip}}), \quad s_{\text{in}} \in \mathbb{R}^{N \times (D_h + D_{\text{skip}})} \tag{S6}$$

$$s_{\text{trans}} = \text{Transformer}(s_{\text{in}}), \quad s_{\text{trans}} \in \mathbb{R}^{N \times (D_h + D_{\text{skip}})} \tag{S7}$$

$$s_{\text{out}} = \text{Linear}(s_{\text{trans}}) + s_\ell, \quad s_{\text{out}} \in \mathbb{R}^{N \times D_h} \tag{S8}$$

$$s^{\ell+1} = \text{MLP}(s_{\text{out}}), \quad s_{\ell+1} \in \mathbb{R}^{N \times D_h} \tag{S9}$$

**Edge Update.** Each edge is updated with a MLP over the current edge and source and target node embeddings. In the first line, node embeddings are first projected down to half the dimension.

$$s^{\ell+1}_{\text{down}} = \text{Linear}(s^{\ell+1}), \quad s^{\ell+1}_{\text{down}} \in \mathbb{R}^{N \times D_h/2} \tag{S10}$$

$$z'_{ij} = \text{concat}(s^{\ell+1}_{\text{down},i}, s^{\ell+1}_{\text{down},j}, z^\ell_{ij}), \quad z'_{ij} \in \mathbb{R}^{N \times (2D_h + D_z)} \tag{S11}$$

$$z^{\ell+1} = \text{LayerNorm}(\text{MLP}(z')), \quad z^{\ell+1} \in \mathbb{R}^{N \times N \times D_z} \tag{S12}$$

**Backbone Update.** Our frame updates follow the BackboneUpdate algorithm in AlphaFold2 (2). We write the algorithm here with our notation,

$$(b_i, c_i, d_i, x_i^{\text{update}}) = \text{Linear}(h_i^L), \tag{S13}$$

$$(a_i, b_i, c_i, d_i) = (1, b_i, c_i, d_i) \,/ \sqrt{1 + b_i^2 + c_i^2 + d_i^2}, \tag{S14}$$

$$R_i^{\text{update}} = \begin{pmatrix} a_i^2 + b_i^2 - c_i^2 - d_i^2 & 2b_ic_i - 2a_id_i & 2b_id_i + 2a_ic_i \\ 2b_ic_i + 2a_id_i & a_i^2 - b_i^2 + c_i^2 - d_i^2 & 2c_id_i - 2a_ib_i \\ 2b_id_i - 2a_ic_i & 2c_id_i + 2a_ib_i & a_i^2 - b_i^2 - c_i^2 + d_i^2 \end{pmatrix}, \tag{S15}$$

$$T_i^{\text{update}} = (R_i^{\text{update}}, x_i^{\text{update}}), \tag{S16}$$

$$T_i^{\ell+1} = T_i^\ell \cdot T_i^{\text{update}}, \tag{S17}$$

where $b_i, c_i, d_i \in \mathbb{R}, x_i^{\text{update}} \in \mathbb{R}^3$. Equ. S15 constructs a normalized quaternion which is then converted into a valid rotation matrix in Equ. S16. Following previous works (4, 45), we use the planar geometry of the backbone to impute the oxygen atoms.

**Sidechain Torsion Angle Update.** To further encode watermark information, we can update sidechain torsion angles based on watermark code and node embeddings:

$$s_\chi = \mathrm{MLP}(s^L), \tag{S18}$$

$$\chi_{\text{watermark}} = (\chi_{\text{orig}} + \mathrm{Linear}(s_\chi + m)) \mod 2\pi, \tag{S19}$$

where $\chi \in [0, 2\pi)^{4N}$. In FoldMark Encoder/Decoder, the number of network blocks is set to 8, the number of transformer layers within each block is set to 4, and the number of hidden channels used in the IPA calculation is set to 16. The node embedding size $D_h$ and the edge embedding size $D_z$ are set as 128. The number of layers $L$ for Encoder/Decoder is set to 6. We removed skip connections and psi-angle prediction.

**Computation of All Atom Coordinates** Following AlphaFold2 (2), FoldMark Encoder outputs the watermarked full-atom structure based on the updated backbone frames $T_i$ and side-chain torsion angles $\chi$ for each residue $i$. These torsion angles are applied to the corresponding amino acid structures, using idealized bond angles and bond lengths, to compute all atom coordinates.

**Watermark Code and Confidence Prediction** In the FoldMark decoder, we extract both a confidence score and a weighted watermark code for robust watermark prediction. The confidence score denoted as $p_{\text{confidence}} = \mathrm{Linear}(a_i)$, is computed by applying a linear transformation to the final node representations $s_i^L$. Concurrently, a per-token watermark code $m_i = \mathrm{Sigmoid}(\mathrm{Linear}(s_i^L))$ is derived through a separate linear transformation followed by a sigmoid activation, ensuring a bounded output between 0 and 1. The final prediction, $m_{\text{pred}} = \sum_i \mathrm{Softmax}(p_{\text{confidence}})_i \cdot m_i$, combines these by weighting each $m_i$ with the corresponding softmax-normalized confidence scores, effectively producing a weighted average that prioritizes contributions based on their predicted reliability.

$$p_{\text{confidence}} = \mathrm{Linear}(s_i^L) \tag{S20}$$

$$m_i = \mathrm{Sigmoid}(\mathrm{Linear}(s_i^L)) \tag{S21}$$

$$m_{\text{pred}} = \sum_i \mathrm{Softmax}(p_{\text{confidence}})_i \cdot m_i. \tag{S22}$$

**Algorithm 1** Equivariant FoldMark Encoder

**Input**: Original Structures $\{x_{\text{orig}}\}$, Watermark Code $m$, Number of Layers $L$

**Output**: Watermarked Structures $\{x_{\text{watermark}}\}$

Set initial frames: $T^0 = \text{ConstructFrame}\{x_{\text{orig}}\}$ # Algorithm 21 in AlphaFold2

Calculate original side chain angles $\chi_{\text{orig}}$.

Compute initial node embeddings: $h_i^0 = \text{MLP}(\phi(i)) + \text{Linear}(m) + \text{Emb}(\chi_{\text{orig}})$ # Optionally encode side chain angles

Compute binned displacements: $\text{disp}_{ij} = \sum_{k=1}^{N_{\text{bins}}} 1\{x^i - x^j < d_k\}$ for $C_\alpha$ coordinates

Compute initial edge embeddings: $z_{ij}^0 = \text{MLP}(\phi(i), \phi(j), \phi(j-i), \phi(\text{disp}_{ij}))$ for each edge $ij$

**for** $\ell = 1$ to $L$ **do**

  $s_{\text{ipa}} = \text{LayerNorm}(\text{IPA}(s^{\ell-1}, z^{\ell-1}, T^{\ell-1})) + s^{\ell-1}$

  $s_{\text{skip}} = \text{Linear}(s^0)$

  $s_{\text{in}} = \text{concat}(s_{\text{ipa}}, s_{\text{skip}})$

  $s_{\text{trans}} = \text{Transformer}(s_{\text{in}})$

  $s_{\text{out}} = \text{Linear}(s_{\text{trans}}) + s^{\ell-1}$

  $s^\ell = \text{MLP}(s_{\text{out}})$

  $s_{\text{down}}^\ell = \text{Linear}(s^\ell)$

  $z'_{ij} = \text{concat}(s_{\text{down},i}^\ell, s_{\text{down},j}^\ell, z_{ij}^{\ell-1})$ for each edge $ij$

  $z^\ell = \text{LayerNorm}(\text{MLP}(z'_{ij}))$

  For each node $i$: $T_i^{\text{update}} = (R_i^{\text{update}}, x_i^{\text{update}}) = \text{BackboneUpdate}(s_i, T_i)$ # Algorithm 23 in AlphaFold2

  $T_i^\ell = T_i^{\ell-1} \cdot T_i^{\text{update}}$ for each node $i$

**end for**

**if** Watermark_Side_Chain **then**

  $s_\chi = \text{MLP}(s_i^L)$ #Compute chi angle features

  $\chi_{\text{watermark}} = (\chi_{\text{orig}} + \text{Linear}(s_\chi + m)) \mod 2\pi$ #Adjust with watermark

**end if**

Set $\{x_{\text{watermark}}\} = \text{SetAllAtomCoordinates}(T^L, \chi_{\text{watermark}})$ # Algorithm 24 in AlphaFold2

**return** $\{x_{\text{watermark}}\}$

**Algorithm 2** Equivariant FoldMark Decoder

**Input**: Watermarked Structures $\{x_{\text{watermark}}\}$, Number of Layers $L$

**Output**: Predicted Watermark $\{m_{\text{pred}}\}$, Watermark Confidence $\{p_{\text{confidence}}\}$

Set initial frames: $T^0 = \text{ConstructFrame}\{x_{\text{orig}}\}$ # Algorithm 21 in AlphaFold2

Calculate original side chain angles $\chi_{\text{watermark}}$

Compute initial node embeddings: $h_i^0 = \text{MLP}(\phi(i)) + \text{Emb}(\chi_{\text{watermark}})$ # Optionally encode side chain angles

Compute binned displacements: $\text{disp}_{ij} = \sum_{k=1}^{N_{\text{bins}}} 1\{x^i - x^j < d_k\}$ for $C_\alpha$ coordinates

Compute initial edge embeddings: $z_{ij}^0 = \text{MLP}(\phi(i), \phi(j), \phi(j-i), \phi(\text{disp}_{ij}))$ for each edge $ij$

**for** $\ell = 1$ to $L$ **do**

  $s_{\text{ipa}} = \text{LayerNorm}(\text{IPA}(s^{\ell-1}, z^{\ell-1}, T^{\ell-1})) + s^{\ell-1}$

  $s_{\text{skip}} = \text{Linear}(s^0)$

  $s_{\text{in}} = \text{concat}(s_{\text{ipa}}, s_{\text{skip}})$

  $s_{\text{trans}} = \text{Transformer}(s_{\text{in}})$

  $s_{\text{out}} = \text{Linear}(s_{\text{trans}}) + s^{\ell-1}$

  $s^\ell = \text{MLP}(s_{\text{out}})$

  $s_{\text{down}}^\ell = \text{Linear}(s^\ell)$

  $z'_{ij} = \text{concat}(s_{\text{down},i}^\ell, s_{\text{down},j}^\ell, z_{ij}^{\ell-1})$ for each edge $ij$

  $z^\ell = \text{LayerNorm}(\text{MLP}(z'_{ij}))$

**end for**

$p_{\text{confidence}} = \text{Linear}(s_i^L)$

$m_i = \text{Sigmoid}(\text{Linear}(s_i^L))$

$m_{\text{pred}} = \sum_i \text{Softmax}(p_{\text{confidence}})_i \cdot m_i$

**return** $\{m_{\text{pred}}\}, \{p_{\text{confidence}}\}$

**Non-Equivariant Architecture** Algorithm 3 and 4 summarize the Non-Equivariant Encoder/Decoder Architecture of FoldMark, which largely follows AlphaFold3 (3). The following paragraphs will detail the feature initialization, the feature processing, watermark structure update, and watermark code/confidence prediction.

**Feature Initialization.** For the Non-Equivariant Encoder, the input features follows AlphaFold3 (3) and can be categorized into six types: Token Features (such as position indexes, chain identifiers, and masks), ESM Features (residue-level embeddings derived from ESM-3B), Reference Features (derived from the reference conformer of residues, nucleotides, or ligands), MSA Features (including genetic search-based features like MSA and deletion matrices), Template Features (obtained from template searches for atom positions), and Bond Features (providing bond information, including polymer-ligand and intra/inter-ligand bonds). The reference features use tools like RDKit and fallback to CCD ideal coordinates if necessary. The single and pair features are initialized as:

$$\{s_i\} = \text{InputFeatureEmbedder}(\{f\}) \tag{S23}$$

$$\{s_i\} += \text{ESMFeature}(\{f\}) \#optional \tag{S24}$$

$$s_{\text{init}} = \text{LinearNoBias}(s_i) + \text{LinearNoBias}(m) \tag{S25}$$

$$z_{\text{init}} = \text{LinearNoBias}(s_i) + \text{LinearNoBias}(s_j) \tag{S26}$$

$$z_{\text{init}} = \text{RelativePositionEncoding}(\{f\}) \tag{S27}$$

$$z_{\text{init}} = \text{LinearNoBias}(f_{\text{token_bonds}}) \tag{S28}$$

The ESM feature is optional in FoldMark Encoder. For the Decoder, the template, ESM, and MSA input features are removed.

**Feature Processing.** The Encoder mainly contains two parts: the feature processing and watermark structure update. The watermark information is simply encoded with linear layers and added into both modules. In the feature processing part, the single and pair representations are initialized, and encoded information from TemplateEmbedder and MSA Modules. We use MSAs from both protein and RNA sequences. The resulting pair representation is then fed into the main PairformerStack, which also takes the single representation as input. Multiple recycling steps are

applied to the aforementioned processing:

$$\{s_i\} = \text{InputFeatureEmbedder}(\{f\}) \tag{S29}$$

$$\{s_i\} += \text{ESMFeature}(\{f\}) \#optional \tag{S30}$$

$$s_{\text{init}} = \text{LinearNoBias}(s_i) + \text{LinearNoBias}(m) \tag{S31}$$

$$z_{\text{init}} = \text{LinearNoBias}(s_i) + \text{LinearNoBias}(s_j) \tag{S32}$$

$$z_{\text{init}} = \text{RelativePositionEncoding}(\{f\}) \tag{S33}$$

$$z_{\text{init}} = \text{LinearNoBias}(f_{\text{token_bonds}}). \tag{S34}$$

**Watermark Structure Update.** In the watermark structure update, a non-equivariant point-cloud structure prediction module over all atoms from AlphaFold3 (3) is used instead of IPA in the Equivariant Encoder/Decoder. The structure is first centered and then input to the structure update module to predict the update. The structure module follows a two-level architecture, first processing atoms, then tokens, and finally atoms again. The transformer uses only a single linear layer to embed all atom positions, and a single linear layer to project the watermark updates at the end, without involving any geometric biases (such as locality or SE(3) invariance), therefore non-equivariant. The structure update process is shown below:

$$x_{\text{orig}} \leftarrow \text{Centering}(x_{\text{orig}}) \tag{S35}$$

$$\{a_i\}, \{q_i\}, \{c_i\}, \{p_{\text{skip}}\} = \text{AtomAttentionEncoder}(\{f\}, \{x_{\text{orig}}\}, \{s_i\}, \{z_{ij}\}) \tag{S36}$$

$$a_i = \text{LinearNoBias}(\text{LayerNorm}(s_i)) + \text{LinearNoBias}(m) \tag{S37}$$

$$a_i = \text{TokenTransformer}(\{a_i\}, \{s_i\}, \{z_{ij}\}) \tag{S38}$$

$$a_i = \text{LayerNorm}(a_i) \tag{S39}$$

$$r_{\text{update}} = \text{AtomAttentionDecoder}(\{a_i\}, \{a_i\}, \{c_i\}, \{p_{\text{skip}}\}) \tag{S40}$$

$$x_{\text{watermark}} = x_{\text{orig}} + r_{\text{update}}. \tag{S41}$$

For the decoder, the predicted structure update $r_{\text{update}}$ is not required.

**Watermark Code and Confidence Prediction.** As for the Decoder, the AtomAttentionDecoder is removed and the per-token watermark code and watermark confidence are first predicted based on the token representations. Then FoldMark Decoder takes a weighted average to obtain the final

watermark code of the whole structure:

$$p_{\text{confidence}} = \text{Linear}(a_i) \tag{S42}$$

$$m_i = \text{Sigmoid}(\text{Linear}(a_i)) \tag{S43}$$

$$m_{\text{pred}} = \sum_i \text{Softmax}(p_{\text{confidence}})_i \cdot m_i. \tag{S44}$$

**Algorithm 3** Non-Equivariant FoldMark Encoder

**Input**: Input Feature $\{f\}$, Original Structure $\{x_{\text{orig}}\}$, Watermark Code $m$, Recycle Number $N_{\text{cycle}}$

**Output**: Watermarked Structure $\{x_{\text{watermark}}\}$

$\{s_i\}$ = InputFeatureEmbedder($\{f\}$)

$\{s_i\}$+ = ESMFeature($\{f\}$) # optional

$s_{\text{init}}$ = LinearNoBias($s_i$) + LinearNoBias($m$)

$z_{\text{init}}$ = LinearNoBias($s_i$) + LinearNoBias($s_j$)

$z_{\text{init}}$ = RelativePositionEncoding($\{f\}$)

$z_{\text{init}}$ = LinearNoBias($f_{\text{token_bonds}}$)

$\{z_{ij}\}, \{s_i\} = 0, 0$

**for** $c \in [1, \ldots, N_{\text{cycle}}]$ **do**

  $z_{ij} = z_{\text{init}}$ + LinearNoBias(LayerNorm($z_{ij}$))

  $z_{ij}$+ = TemplateEmbedder($\{f\}, \{z_{ij}\}$) # Algorithm 16 in AlphaFold3, optional

  $z_{ij}$+ = MsaModule($f_{\text{msa}}, \{z_{ij}\}$) # Algorithm 8 in AlphaFold3, optional

  $s_i = s_{\text{init}}$ + LinearNoBias(LayerNorm($s_i$))

  $\{s_i\}, \{z_{ij}\}$ = PairformerStack($\{s_i\}, \{z_{ij}\}$) # Algorithm 17 in AlphaFold3

**end for**

$x_{\text{orig}} \leftarrow$ Centering($x_{\text{orig}}$)

$\{a_i\}, \{q_i\}, \{c_i\}, \{p_{\text{skip}}\}$ = AtomAttentionEncoder($\{f\}, \{x_{\text{orig}}\}, \{s_i\}, \{z_{ij}\}$) # Algorithm 5 in AlphaFold3

$a_i$ = LinearNoBias(LayerNorm($s_i$)) + LinearNoBias($m$)

$a_i$ = TokenTransformer($\{a_i\}, \{s_i\}, \{z_{ij}\}$) # Algorithm 23 in AlphaFold3

$a_i$ = LayerNorm($a_i$)

$r_{\text{update}}$ = AtomAttentionDecoder($\{a_i\}, \{a_i\}, \{c_i\}, \{p_{\text{skip}}\}$) # Algorithm 6 in AlphaFold3

$x_{\text{watermark}} = x_{\text{orig}} + r_{\text{update}}$

**return** $\{x_{\text{watermark}}\}$

**Algorithm 4** Non-Equivariant FoldMark Decoder

**Input**: Input Feature $\{f\}$, Watermarked Structure $\{x_{\text{watermark}}\}$, Recycle Number $N_{\text{cycle}}$

**Output**: Predicted Watermark $\{m_{\text{pred}}\}$, Watermark Confidence $\{p_{\text{confidence}}\}$

$\{s_i\} = \text{InputFeatureEmbedder}(\{f\})$

$s_{\text{init}} = \text{LinearNoBias}(s_i)$

$z_{\text{init}} = \text{LinearNoBias}(s_i) + \text{LinearNoBias}(s_j)$

$z_{\text{init}} = \text{RelativePositionEncoding}(\{f\})$

$z_{\text{init}} = \text{LinearNoBias}(f_{\text{token_bonds}})$

$\{z_{ij}\}, \{s_i\} = 0, 0$

**for** $c \in [1, \ldots, N_{\text{cycle}}]$ **do**

  $z_{ij} = z_{\text{init}} + \text{LinearNoBias}(\text{LayerNorm}(z_{ij}))$

  $s_i = s_{\text{init}} + \text{LinearNoBias}(\text{LayerNorm}(s_i))$

  $\{s_i\}, \{z_{ij}\} = \text{PairformerStack}(\{s_i\}, \{z_{ij}\})$ # Algorithm 17 in AlphaFold3

**end for**

$x_{\text{watermark}} \leftarrow \text{Centering}(x_{\text{watermark}})$

$\{a_i\}, \{q_i\}, \{c_i\}, \{p_{\text{skip}}\} = \text{AtomAttentionEncoder}(\{f\}, \{x_{\text{watermark}}\}, \{s_i\}, \{z_{ij}\})$ # Algorithm 5 in AlphaFold3

$a_i = \text{LinearNoBias}(\text{LayerNorm}(s_i))$

$a_i = \text{TokenTransformer}(\{a_i\}, \{s_i\}, \{z_{ij}\})$ # Algorithm 23 in AlphaFold3

$a_i = \text{LayerNorm}(a_i)$

$p_{\text{confidence}} = \text{Linear}(a_i)$

$m_i = \text{Sigmoid}(\text{Linear}(a_i))$

$m_{\text{pred}} = \sum_i \text{Softmax}(p_{\text{confidence}})_i \cdot m_i$

**return** $\{m_{\text{pred}}\}, \{p_{\text{confidence}}\}$

## Introduction and Implementation of Protein Generative Models

**AlphaFold3** (3) (https://github.com/google-deepmind/alphafold3) is the latest non-equivariant diffusion model for biomolecular structures developed by DeepMind, which is capable of predicting the joint structure of complexes including proteins, nucleic acids, small molecules, ions, and modified residues. For MSA, AlphaFold3 uses UniRef90 (v2022_05) (43), UniProt (v2021_04) (72), Reduced BFD (73), MGnify (v2022_05) (74) for protein chains and Rfam (v14.9) (75), RNACentral (v21.0) (76), and Nucleotide collection (2023-02-23) (77) for RNA chains. Template structures are retrieved from the corresponding PDB mmCIF file of the searched UniRef90 (43) MSA for protein chains. For inference, the number of diffusion steps is 200 and the number of predictions per seed is 5.

**Protenix** (48) (https://github.com/bytedance/Protenix) is an open-sourced version of AlphaFold3, providing model weights, inference code, and trainable code for research use. Generally, Protenix achieves comparable performance with AlphaFold3 in predicting structures across different molecular types. For MSA, Protenix searches MSAs using MMSEQS2 (42) and ColabFold (60) MSA pipeline, and use MSAs from the Uniref100 (43) database for pairing (species are identified with taxonomy IDs). Protenix does not use MSA for nucleic chains. Protenix also does not use templates. For inference, the number of diffusion steps is 200 and the number of predictions per seed is 5.

**Boltz-1** (49) (https://github.com/jwohlwend/boltz) is an open-source AlphaFold3-like model incorporating innovations in model architecture, speed optimization, and data processing, achieving AlphaFold3-level accuracy in predicting the 3D structures of biomolecular complexes. For example, to overcome the potential structure misalignment issue during model inference, Boltz-1 adds a rigid alignment with Kabsch algorithm (78) after every step during the inference procedure before $\hat{x}_t^{\text{noisy}}$ and $\hat{x}_t^{\text{denoised}}$ are interpolated. Under the assumption of a Dirac distribution, the interpolated structure is more similar to the denoised sample than the noisy structure. MSAs are constructed for the full PDB data using the colabfold search tool (60). Unlike AlphaFold3, Boltz-1 does not include input templates, due to their limited impact on the performance of large models. The training and inference code, model weights, datasets, and benchmarks of Boltz-1 are released under the MIT open license, aiming at democratizing biomolecular interaction modeling.

**Chai-1** (50) (https://github.com/chaidiscovery/chai-lab) is a AlphaFold3-like model for molecular structure prediction developed by Chai Discovery. The Chai-1 neural architecture largely follows AlphaFold3. One key difference of Chai-1 is the incorporation of protein language model embeddings (ESM-3B) (39) as optional input. Per-residue embeddings from ESM are added into single representations of protein chains. Residues that do not belong to a protein chain (e.g., DNA, RNA, or ligands) are assigned a mask token. During inference, Chai-1 can be run with any combination of MSAs, templates, and language model embeddings. MSAs are generated and deposited in OpenProteinSet (79) for databases UniRef90 (43), UniProt (72), MGnify (74), and UniClust30+BFD (73). Templates are generated using PDB70 (44).

**ESMFold** (39) (https://github.com/facebookresearch/esm) is based on ESM protein language models to generate accurate structure predictions end to end directly from the sequence of a protein. This difference significantly reduces inference time, making ESMFold approximately ten times faster than AlphaFold series. The structure folding head begins with 48 folding blocks. Each folding block alternates between updating a sequence representation and a pairwise representation. The output of these blocks is passed to an equivariant transformer structure module, and 3 steps of recycling are performed before outputting a final atomic-level structure and predicted confidence.

**RFDiffusion** (4) (https://github.com/RosettaCommons/RFdiffusion) is one of the state-of-the-art method for *de novo* protein backbone generation. It combines the RoseTTAFold structure prediction network with the diffusion probabilistic models (DDPMs) framework. When generating ligand-binding protein binders, we use a heuristic attractive-repulsive potential to encourage the formation of pockets with shape complementarity to a target molecule following the suggestions of RFDiffusion (4) and RFDiffusionAA (40).

**RFDiffusionAA** (40) (https://github.com/baker-laboratory/rf_diffusion_all_atom) is the latest version of RFDiffusion which combines a residue-based representation of amino acids and atomic representations of all other groups to model protein-small molecules/metals/nucleic acids/covalent modification complexes. Starting from random distributions of amino acid residues surrounding target small molecules, RFDiffusionAA can directly generate the small molecule binding protein backbone.

**FrameDiff** (46) (https://github.com/jasonkyuyim/se3_diffusion) is a diffusion model designed for generating protein backbones, leveraging SE(3) symmetry to ensure the generated structures respect the geometric properties of proteins. It operates on frames, which are rigid bodies in 3D space, and employs an SE(3)-invariant diffusion process to create new protein structures, marking a significant advancement in de novo protein design. The model's architecture includes an SE(3) equivariant score function over multiple frames, developed through a theoretical framework for diffusion on SE(3). This allows FrameDiff to generate designable protein monomers up to 500 amino acids in length without relying on a pre-trained protein structure prediction network.

**FrameFlow** (45) (https://github.com/microsoft/protein-frame-flow) is a method for fast protein backbone generation, utilizing SE(3) flow matching, which adapts the state-of-the-art diffusion model FrameDiff to a flow-matching generative modeling paradigm. This adaptation is significant as it reduces the number of sampling timesteps required. The technical architecture of FrameFlow includes an SE(3)-equivarient neural network, specifically the FramePred architecture with IPA updates. It introduces modifications such as weighting the rotation loss by 0.5, clipping the scaling factor at 1/(1-mint,0.9), and using an exponential scheduler $\kappa(t) = e^{(-ct)}$ for SO(3) inference, with c=10 found effective.

**FoldFlow2** (47) (https://github.com/DreamFold/FoldFlow) is a state-of-the-art protein structure generative model that integrates both protein sequence and structure information, building on its predecessor, FoldFlow. It employs a multi-modal approach, combining sequence encoding from the ESM-2 protein language model, trained on millions of sequences, with structure encoding from FoldFlow's structure module. This dual approach addresses the interdependence of sequence and structure, a shortcoming in previous models that tackled them separately. The architecture features a trunk network with multiple attention-based layers for fusing sequence and structure data, ensuring effective integration. The structure decoder mirrors the encoder, providing geometrically informed predictions. FoldFlow2 is trained on a dataset eight times larger than its predecessor FoldFlow. This large-scale training enhances its capability to generate designable, diverse, and novel proteins.

## Supplementary Text

### More Comparison with Other Watermarking Methods

Traditional watermarking techniques developed for Large Language Models (LLMs) (27, 80–82) and diffusion models (33–35, 83) are not directly transferable to protein structure data due to the distinct and complex characteristics of protein structures. Protein structures exhibit flexibility, sensitivity, and geometric intricacy, requiring specialized methods for embedding and retrieving watermarks without compromising data integrity or model performance.

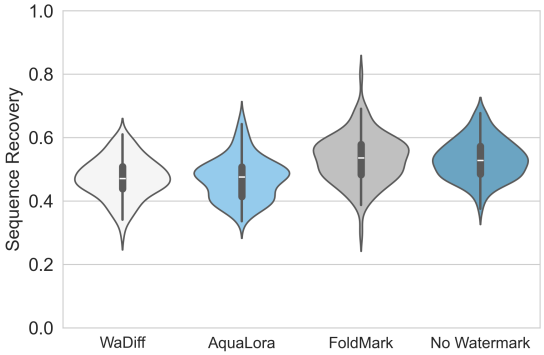
Previous methods, such as WaDiff (34) and AquaLoRA (35), embed watermarks into the U-Net backbone of Stable Diffusion models for image generation. While effective in the image domain, these approaches face significant challenges in protein generative models. The use of convolutional neural networks (CNNs) as encoder-decoder components in WaDiff and AquaLoRA limits their performance in the protein domain, as CNNs are not inherently designed to handle the spatial and rotational properties of protein structures. FoldMark overcomes these limitations by leveraging state-of-the-art SE(3)-equivariant or non-equivariant graph transformers for both the Watermark Encoder and Decoder, ensuring geometric consistency and superior performance. Additionally, FoldMark introduces customized data augmentation strategies, such as structure cropping, rotation, and noising, to enhance the robustness of training. These strategies are tailored to protein structures and are not directly transferable from image-based methods. To address the challenges of protein flexibility and sensitivity, FoldMark adaptively embed watermark information across all residues guided by evolutionary signals (e.g., genetic search, protein language model embeddings, and structure templates). Therefore, FoldMarks learsn to minimize noise in conserved regions while exploiting flexible regions for enhanced watermark capacity.

FoldMark employs a gating vector derived from the watermark code for watermark-conditioned Low-Rank Adaptation (LoRA), ensuring independence from rank selection. This design avoids the parameter overhead associated with AquaLoRA's reliance on diagonal matrix modifications, which require large ranks (e.g., 320) to embed watermark information effectively.

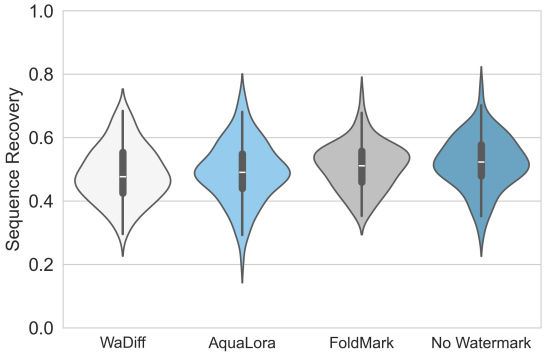
Finally, FoldMark incorporates an additional, time-conditioned fine-tuning step for the watermark decoder, allowing it to adapt to the intricate protein structures. The message retrieval loss assigns dynamic weights to different time steps (e.g., larger weights as $t \rightarrow 0$). This approach

balances the preservation of generation quality with explicit improvements in watermark retrieval success rates.

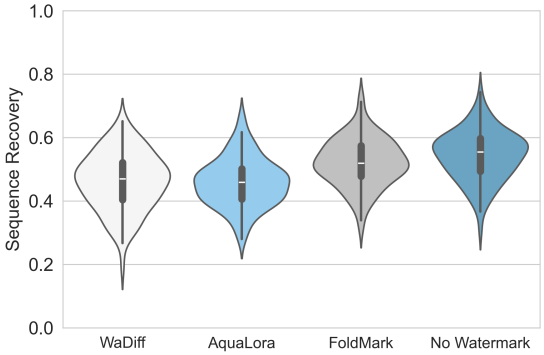
In summary, FoldMark addresses the unique challenges of protein generative models by combining advanced architectural design, robust watermark strategies, and innovative fine-tuning approaches, achieving superior performance in protecting protein generative models compared to existing methods.



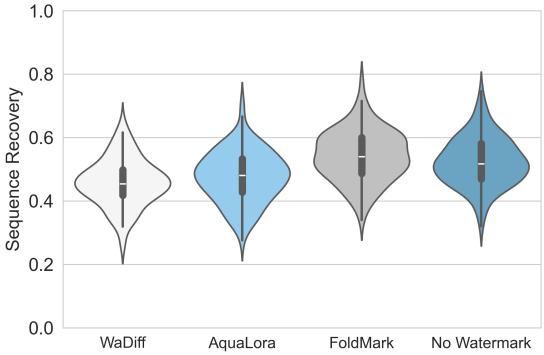

(a) AlphaFold3



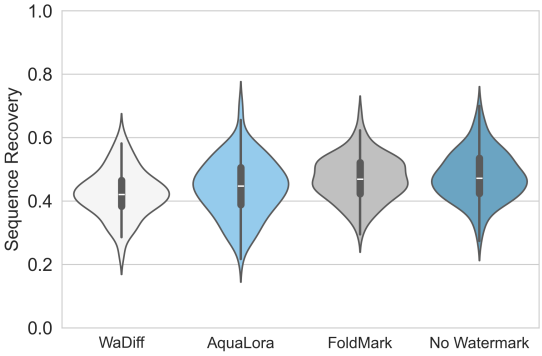

(b) Boltz-1




(c) Chai-1



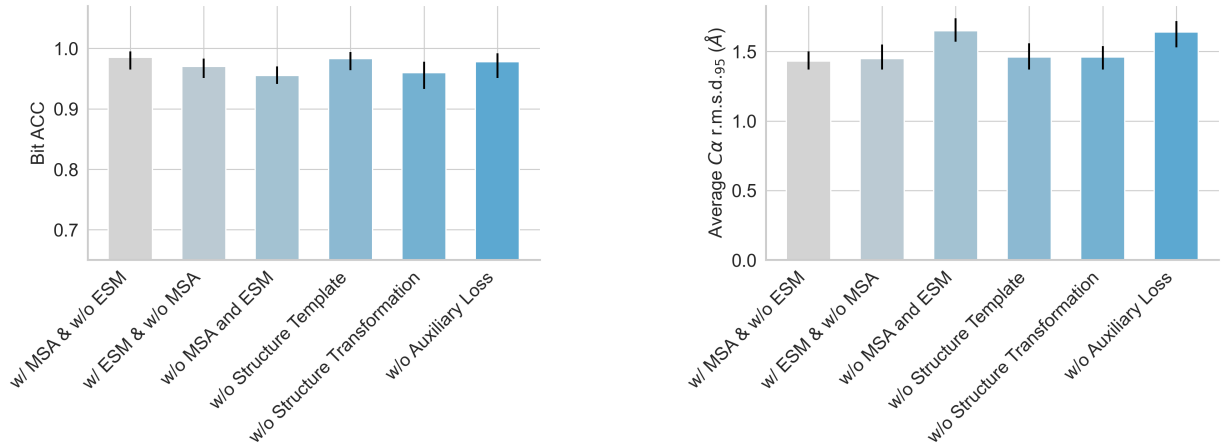

(d) Protenix




(e) ESMFold

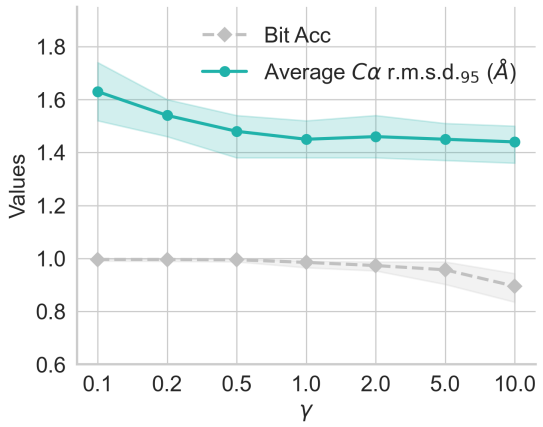
**Figure S1**: Sequence Recovery Performance of Predicted Structures Generated by WaDiff, AquaLora, FoldMark, and Baseline Vanilla Models. Experiments were conducted on filtered single-chain protein datasets with AlphaFold3, Boltz-1, Chai-1, Protenix, and ESMFold models. Protein sequences were inferred using ProteinMPNN from the predicted structures, and sequence recovery rates were calculated as the percentage of correctly predicted amino acids relative to the native sequences, highlighting the models' ability to recapitulate biologically relevant protein sequences.

**Figure S2 (continued).** In the violin plots, the embedded box plot includes a thick vertical bar indicates the interquartile range (IQR, 25th to 75th percentiles) and a horizontal line marking the median. Thin whiskers extend to the minimum and maximum values within 1.5 times the IQR.

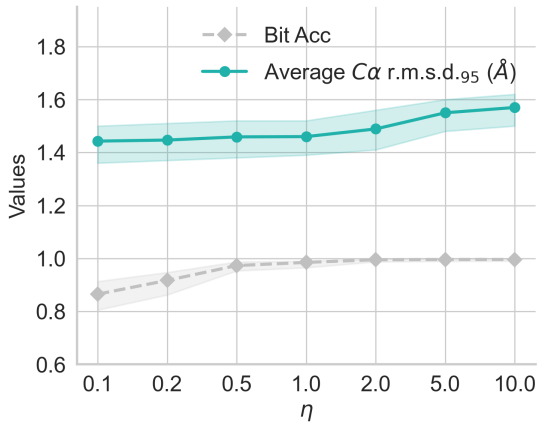



(a) (b)

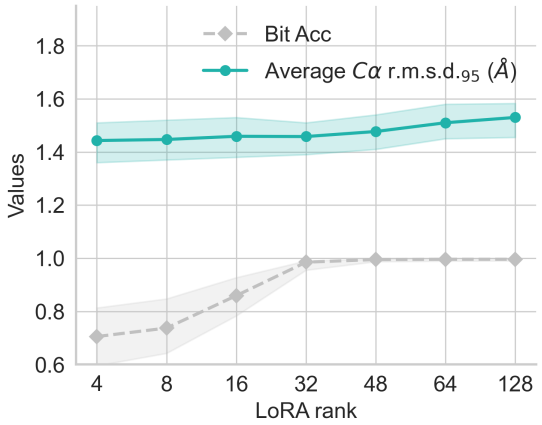
**Figure S2**: Ablation studies of FoldMark with AlphaFold3. The MSA, ESM, and Template modules in FoldMark Encoder, the structure transformations in FoldMark Encoder/Decoder pertaining, and the auxiliary loss in FoldMark Finetuning are removed respectively. Note that the FoldMark Encoder can utilize either the MSA module, the ESM module, or operate without any MSA/ESM module. The shadows indicate the 95% confidence intervals obtained by 10,000 bootstrap resampling.




(a) $\gamma$ in $\mathcal{L}_{Pretrain}$ (Equation. 4)




(b) $\eta$ in $\mathcal{L}_{Finetune}$ (Equation. 5)




(c) Watermark conditioned LoRA rank (Equation. 6)

**Figure S3**: Hyperparameter Analysis of FoldMark. The shadows indicate the 95% confidence intervals obtained by 10,000 bootstrap resampling.

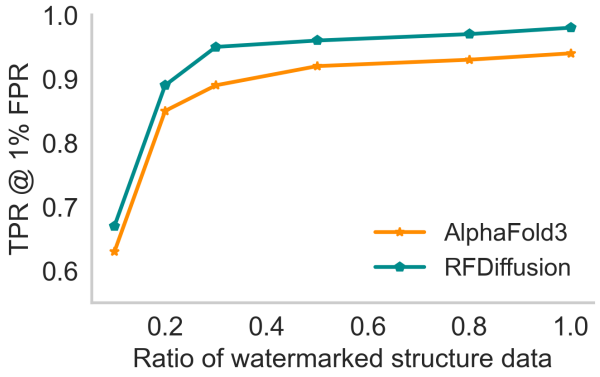
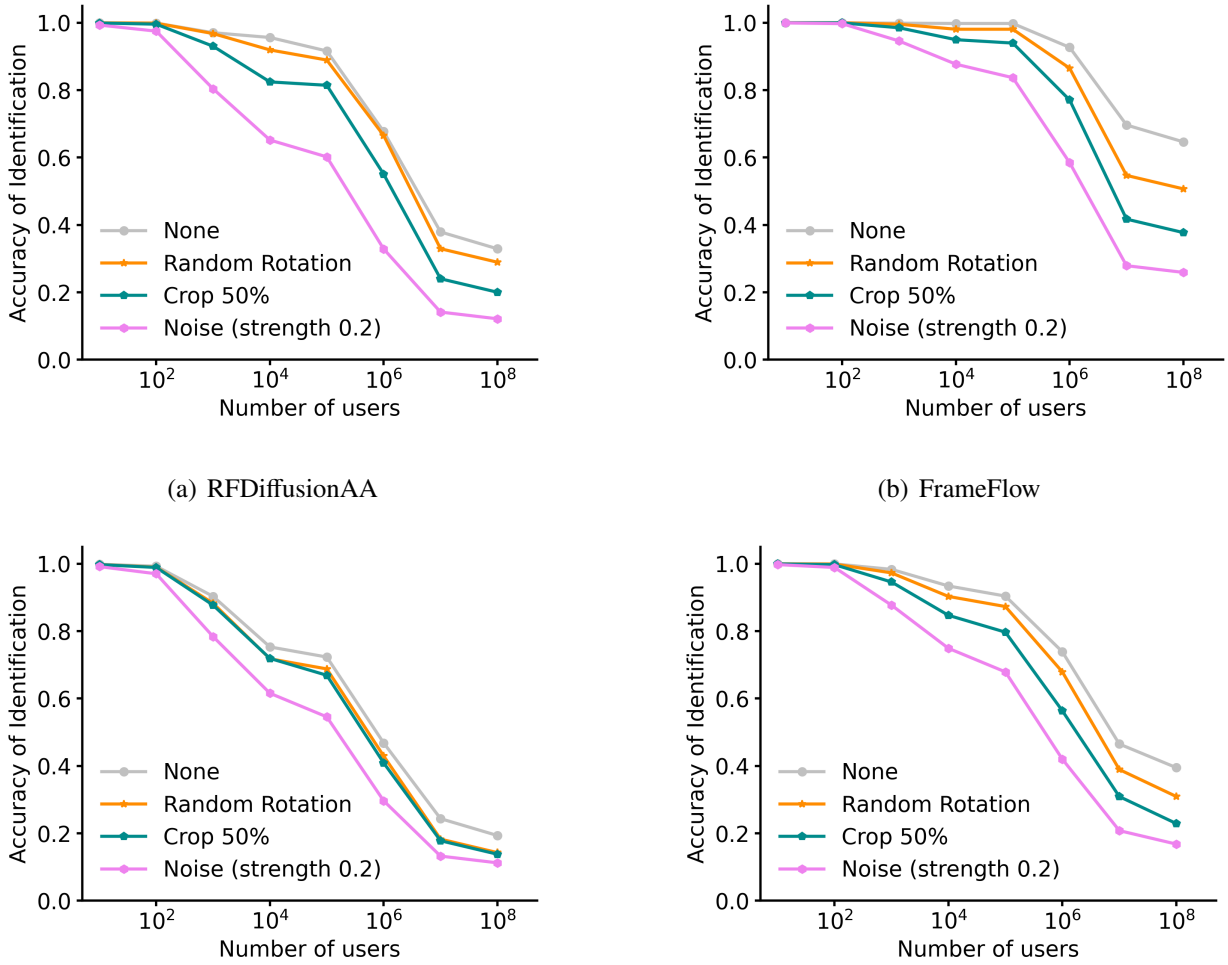



**Figure S4**: Influence of the ratio of watermarked structures on detection true positive rates at 1% false positive rates.




(a) RFDiffusionAA

(b) FrameFlow

(c) FrameDiff

(d) FoldFlow

**Figure S5**: Performance of FoldMark for user tracing under different post-processing with different numbers of users (FPR=$10^{-3}$). (a) RFDiffusionAA; (b) FrameFlow; (c) FrameDiff; (d) FoldFlow are compared here. Protein post-processing methods include randomly rotating the watermarked strictures (Random Rotation), cropping the structure to keep 50% of the whole sequence (Crop 50%), and adding Gaussian noise with a strength of 0.2 to the coordinates (Noise, strength 0.2).

## More Results of FoldMark for Different Models

In Table. S1 and S2, we show more detailed results of applying FoldMark to different protein structure prediction models and *de novo* protein structure generative models.

**Table S1**: Watermarking different protein structure prediction models by FoldMark with different watermark bits. The average watermark bit accuracy (BitAcc) and the C$\alpha$ r.m.s.d. at 95% coverage (RMSD) are reported.

| **Watermark Method** | | **4bit** | | **8bit** | | **16bit** | | **32bit** | |
|---|---|---|---|---|---|---|---|---|---|
| | | BitAcc ↑ | RMSD ↓ | BitAcc ↑ | RMSD ↓ | BitAcc ↑ | RMSD ↓ | BitAcc ↑ | RMSD ↓ |
| Protenix | No watermark | - | 1.464 | - | 1.464 | - | 1.464 | - | 1.464 |
| | WaDiff | 66.3% | 1.542 | 64.0% | 1.583 | 60.4% | 1.606 | 52.1% | 1.723 |
| | AquaLoRA | 64.0% | 1.526 | 62.9% | 1.530 | 61.2% | 1.648 | 50.4% | 1.691 |
| | FoldMark(Ours) | **99.5%** | **1.465** | **98.5%** | **1.468** | **97.3%** | **1.470** | **95.4%** | **1.568** |
| ESMFold | No watermark | - | 1.807 | - | 1.807 | - | 1.807 | - | 1.807 |
| | WaDiff | 78.5% | 1.920 | 74.1% | 1.943 | 71.6% | 2.069 | 54.0% | 2.200 |
| | AquaLoRA | 79.4% | 1.895 | 76.8% | 1.957 | 75.3% | 2.055 | 59.4% | 2.120 |
| | FoldMark(Ours) | **99.6%** | **1.815** | **97.2%** | **1.819** | **96.2%** | **1.821** | **90.1%** | **1.845** |

**Table S2**: Watermarking *de novo* protein structure generative models by FoldMark with different watermark bits. The average watermark bit accuracy (BitAcc) and the self-consistency TM scores (scTM) are reported.

| **Watermark Method** | | **4bit** | | **8bit** | | **16bit** | | **32bit** | |
|---|---|---|---|---|---|---|---|---|---|
| | | BitAcc ↑ | scTM ↑ | BitAcc ↑ | scTM ↑ | BitAcc ↑ | scTM ↑ | BitAcc ↑ | scTM ↑ |
| FrameFlow | No watermark | - | 0.921 | - | 0.921 | - | 0.921 | - | 0.921 |
| | WaDiff | 77.1% | 0.875 | 76.4% | 0.836 | 70.5% | 0.826 | 54.6% | 0.811 |
| | AquaLoRA | 79.6% | 0.860 | 75.4% | 0.829 | 74.5% | 0.810 | 55.1% | 0.805 |
| | FoldMark(Ours) | **99.6%** | **0.920** | **99.5%** | **0.919** | **96.7%** | **0.916** | **95.4%** | **0.904** |
| FrameDiff | No watermark | - | 0.890 | - | 0.890 | - | 0.890 | - | 0.890 |
| | WaDiff | 76.8% | 0.863 | 73.3% | 0.848 | 62.2% | 0.810 | 50.4% | 0.795 |
| | AquaLoRA | 64.3% | 0.852 | 59.1% | 0.839 | 56.2% | 0.794 | 51.6% | 0.778 |
| | FoldMark(Ours) | **98.7%** | **0.890** | **98.3%** | **0.875** | **88.4%** | **0.873** | **82.0%** | **0.850** |
| FoldFlow | No watermark | - | 0.945 | - | 0.945 | - | 0.945 | - | 0.945 |
| | WaDiff | 88.2% | 0.869 | 85.0% | 0.828 | 80.1% | 0.777 | 64.3% | 0.773 |
| | AquaLoRA | 73.5% | 0.854 | 72.6% | 0.830 | 71.7% | 0.786 | 62.8% | 0.778 |
| | FoldMark(Ours) | **99.9%** | **0.937** | **99.7%** | **0.936** | **98.9%** | **0.931** | **94.5%** | **0.924** |

**Table S3**: Hyperparameters for protein structure prediction models.

| Hyperparameter | Value | Description |
|---|---|---|
| | | AlphaFold3 |
| N_cycle | 10 | Number of recycles in implementing Pairformer. |
| n_steps | 200 | The number of diffusion steps for inference. |
| c_s | 384 | The size of single representation. |
| c_z | 128 | The size of pair representation. |
| | | Protenix |
| N_cycle | 10 | Number of recycles in implementing Pairformer. |
| n_steps | 200 | The number of diffusion steps for inference. |
| c_s | 384 | The size of single representation. |
| c_z | 128 | The size of pair representation. |
| | | Boltz-1 |
| N_cycle | 3 | Number of recycles in implementing Pairformer. |
| n_steps | 200 | The number of diffusion steps for inference. |
| c_s | 384 | The size of single representation. |
| c_z | 128 | The size of pair representation. |
| | | Chai-1 |
| N_cycle | 4 | Number of recycles in implementing Pairformer. |
| n_steps | 200 | The number of diffusion steps for inference. |
| c_s | 384 | The size of single representation. |
| c_z | 128 | The size of pair representation. |
| | | ESMFold |
| embed_size | 2560 | Dim size of embedding. |
| N_blocks_fold | 48 | Number of folding trunks. |
| N_blocks_structure | 8 | Number of equivariant transformer structure modules. |

**Table S4**: Hyperparameters for unconditional protein generative models.

| Hyperparameter | Value | Description |
|---|---|---|
| | | RFDiffusion |
| potential_scale | 1.0 | Scale of auxiliary substrate contact potential. |
| crd_scale | 0.25 | Scale of coordinates during the diffusion process. |
| self-conditioning | True | Iterative refinement via self-conditioning. |
| n_steps | 50 | Number of diffusion steps. |
| | | RFDiffusionAA |
| crd_scale | 0.0667 | Scale of coordinates during the diffusion process. |
| n_steps | 100 | Number of diffusion steps. |
| | | FrameFlow |
| node_embed_size | 256 | Dim size of the node embeddings. |
| edge_embed_size | 128 | Dim size of the edge embeddings. |
| n_steps | 100 | Number of sampling steps. |
| | | FrameDiff |
| node_embed_size | 256 | Dim size of the node embeddings. |
| edge_embed_size | 128 | Dim size of the edge embeddings. |
| n_steps | 500 | Number of diffusion steps. |
| | | FoldFlow2 |
| node_embed_size | 256 | Dim size of the node embeddings. |
| edge_embed_size | 128 | Dim size of the edge embeddings. |
| n_steps | 50 | Number of diffusion steps. |