We thank the reviewers for their insightful feedback.

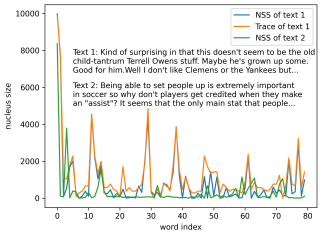
Reviewer 1 **Knowledge of the exact model and weights used by the victim:** Model reuse is indeed commonplace. If the victim is using an off-the-shelf autocompletion app (e.g., as part of a commercial software package), the same app is likely available to many other users, including potential attackers. We will clarify this in Section 4.1. Transferability of fingerprints across similar (but not identical) models is an interesting topic for future research.

Fully generated texts vs. intermittently accepting a completion: For a given text, this does not affect the trace measured by the attacker because in both cases the nucleus is sampled after every word (we will clarify this in the text). Human editing such as deleting or rewriting might "pollute" the attacker's trace with nucleus sizes corresponding to deleted subsequences. The attacker can deal with this by guessing which trace chunks originate from edits, removing them, and trying to match the result. When there are many edits, this has a nontrivial computational cost, which may be offset by using auxiliary information from the side channel to guide the guesses (e.g., timing, nucleus sizes, and control flow of the code that operates the language model). Such methods are hard to evaluate in our lab setting because human editing is indeed hard to simulate. We will expand the discussion of this limitation in Section 5.1.

Conditioning NSS on longer texts: In our experiments, the language model resets on every user post (sometimes referred to as "sentence" in the submission; we will fix the terminology). We will expand on the implications of longer posts in Section 3.2, providing the average post lengths for the 5 subreddits in our experiments (they range from 36 to 61 tokens). Average post length is inversely correlated with the fraction of "variable" sequences (variability ensures that the NSS is a unique fingerprint), but even for the real-world text domains with relatively long posts (e.g., sports), the fraction of variable sequences is significant (>40%). We acknowledge that for NSS generated from much longer individual texts (e.g., entire documents with thousands of tokens), the fraction of variable sequences may be lower.

Additional experiments: We thank the reviewer for their suggestions. (1) We note that our mitigation makes loop iteration counts independent of the nucleus sizes, therefore the attack fails completely. We will explain that it would even fail to distinguish 2 known texts, let alone recognize an open-world fingerprint. (2) We will add another case study, similar to the one in Section 5.3. For example, we already collected traces of 50 users' aggregated posts in the Ubuntu Chat Corpus. The texts are all variable, their NSS are unique (distance $> U(N)$), traces are consistent with fingerprints (noise $< d(N)$), and our attack identifies them.

Reviewer 2. As an additional illustration, we plan to add a plot similar to the one on the right, showing the NSS of two text sequences X and Y and the trace t of Y. This illustrates how the trace "matches" the fingerprint of the correct text and does not match those of other texts.



Reviewer 3. Side channels due to input-dependent branching have been studied in the computer security literature, yet none of the previous work (including none of the work cited by the reviewer) identified *inputs of ML models* as being vulnerable to these attacks. This observation and our analysis are nontrivial, because computation in ML models involves mostly tensor arithmetics whose control flow does *not* depend on the input. Yet we demonstrate that even a few lines of code in a widely used ML building block can unintentionally leak the inputs.

Our main contributions are to show that (1) text generation using language models creates fingerprints for many input texts, and (2) these fingerprints are measurable via side channels. Uniqueness of fingerprints increases with the input text's length *regardless of the specific side channel used*.

Our work neither requires, nor claims any innovation in side-channel techniques or defenses. We view the fact that nucleus size sequences can be fingerprinted via known techniques as a strength of our paper. It means that the vulnerability we identify is dangerous because it can be exploited using well-known, easily available tools and does not require esoteric or new ones.

Reviewer 4. The reviewer correctly notes that loading a local shared object would thwart a Flush+Reload attack. That said, other side channels such as Prime+Probe have been repeatedly shown effective for inferring loop arguments (Section 2.2).

Users can indeed try to avoid using autocompletion features altogether, which is not always the case for cryptographic primitives. Our work helps users make these decisions by identifying nucleus sampling (and ML building blocks in general) as security hotspots, analogous to cryptographic primitives in that they are small functional components that are (1) ubiquitously deployed and integrated into many software apps, and (2) operate on sensitive data. This motivates adoption of secure coding practices for ML code.