Criminal Minds: A Plagiarism Study about Getting Away With It (for Science)

Robin Maisch
robin.maisch@kit.edu
Karlsruhe Institute of Technology
Karlsruhe, Germany

Larissa Schmid
lgschmid@kth.se
KTH Royal Institute of Technology
Stockholm, Sweden

Richard Glassey
glassey@kth.se
KTH Royal Institute of Technology
Stockholm, Sweden

Abstract

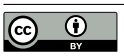
Can we benefit from students cheating? We think so. In the *Criminal Minds* study, first-year students are tasked with plagiarizing code and evading detection, choosing from a variety of strategies. What began as a collaboration between researchers at KIT (Germany) and KTH (Sweden) to address the shortage of open plagiarism datasets also proved to be a highly engaging pedagogical activity, creating a safe space for students to experience and discuss plagiarism from a radically different perspective. We present the study design and the controlled lab setting. We plan to repeat the study in Fall 2026 and use this poster to encourage educators to collaborate with us on future installments of the Criminal Minds study.

ACM Reference Format:
Robin Maisch, Larissa Schmid, and Richard Glassey. 2026. Criminal Minds: A Plagiarism Study about Getting Away With It (for Science). In *Proceedings of the 31st ACM Conference on Innovation and Technology in Computer Science Education V. 2 (ITiCSE 2026), July 10–15, 2026, Madrid, Spain.* ACM, New York, NY, USA, 1 page. https://doi.org/10.1145/3803401.3812035

1 Concept and Research Goals

In introductory programming courses, the academic integrity of coding exercises is constantly threatened by students plagiarizing solutions from peers. Software plagiarism detection research has focused on concrete *obfuscation attacks*, i.e., code modification patterns that aim to reduce structural similarity between original code submissions and derived plagiarized versions. As a result, various countermeasures have been developed to enhance the resilience of plagiarism detection approaches. As LLMs become prevalent auxiliary tools in programming, educators face new challenges in fostering coding skills, which may be impeded by the irresponsible use of these tools. As of now, heuristic features of code may hint at LLM-generated code; however, unambiguous detection remains a challenge. Also, it remains unclear to what extent and for what purpose students rely on LLMs for coding tasks.

With our study setup, we aim to investigate the variety and evolution of code plagiarism strategies used by students. To this end, a live session is organized in which students plagiarize complete code submissions for a given task, aiming to submit an unrecognizable yet working submission within a set time frame. This study setup has already been carried out at KIT and KTH [2], and we plan to continue our efforts and invite others to join.



ITiCSE 2026, Madrid, Spain

ACM ISBN 979-8-4007-2633-0/2026/07
https://doi.org/10.1145/3803401.3812035

2 Setup and Process of the Live Session

We selected a programming task that matched the students' skill level at the time of the live session. Teaching assistants provided their own solutions from their first term, which served as the original submissions to be plagiarized from. We randomly assigned registrants to one of five original submissions and to one of two study groups with different instructions (A: free choice of approaches; B: must include LLM-based tools) across both universities.

At the live session, participants first gave written consent to the use and publication of the resulting artifacts. They then copied their assigned original submission and modified it using the approaches available to their study group. Participants logged each modification as a plagiarism trace using a simple HTML form, generating a uniform CSV change log. They could submit their current program state at any time to receive automatic functional test results, confirming the program still worked as intended. After their final submission, participants completed an online survey to assess their subjective perceptions of their success and the study setup.

3 Evaluation and Reflection

We evaluated participants' modification steps (based on change logs), resulting programs, survey results, and correlated these factors with participants' university, study group, assigned original submission, and performance in the respective university course [2].

Code changes. We mapped each change log item to a two-part phrase (*move method*, *rename class*, etc.) and classified each phrase using the canonical plagiarism taxonomy [3]. We also tracked changes in code size and documentation size relative to the original.

Plagiarism success. We assessed the similarity of each resulting plagiarized submission to its original using JPlag [1] and functional quality using the functional test results.

Accuracy of subjective perception. We compared participants' study responses with properties of their submissions to assess whether they accurately assessed their own performance.

Beyond the dataset, the study proved to be a genuinely engaging pedagogical experience. Participants reported finding the session thought-provoking and noted that approaching plagiarism from the perpetrator's perspective gave them new insights into the topic compared with conventional academic integrity instruction.

References

[1] Lutz Prechelt et al. 2002. Finding plagiarisms among a set of programs with JPlag. *JUCS* 8, 11 (11 2002), 1016. doi:10.3217/jucs-008-11-1016
[2] Robin Maisch et al. 2026. Criminal Minds: How First-Year CS Students Plagiarize Code. In *FSE Companion'26.*
[3] Oscar Karnalim. 2016. Detecting source code plagiarism on introductory programming course assignments using a bytecode approach. In *2016 ICTS.* IEEE, 63–68. doi:10.1109/icts.2016.7910274