Examination Cheat Risk Reduction through FIPEs

Josef Spillner[a]

Zurich University of Applied Sciences, Winterthur, Switzerland

Keywords: e-Assessment, Grading, Automation, Variability.

Abstract: Fully Individualised Programmable Exams (FIPEs) are physically or digitally written examinations in which the task descriptions are different for all students according to programmatically controlled variability. FIPEs reduce the risk of collaborative cheating while adhering to institutional equality policies by controlling how many differences are introduced. The individualisations are based on permutations, deviations, randomisations and sampling. They allow for meeting legal constraints and yet gaining the desired ability to further automate solutions checking. This paper contributes a software solution to address the need from exam specification to distribution. It introduces a categorisation for FIPEs, presents a working implementation to generate and disseminate exam documents and delivers an experience report in two curriculums, computer science and engineering sciences.

1 INTRODUCTION

Examinations are one of the traditional and versatile instruments to assess and grade the performance of university students with regards to the learning goals and learning outcomes (Piontek, 2008). Written examinations in particular can be among the most efficient and fair instruments if they adhere to certain forms and are conducted with technology support to address various problems in physical and online settings (Maroco et al., 2019). Cheating is among those traditional yet undesired problems with any assessment, defined by the unauthorised acquisition of help and leading some educators to rethink the conceptual forms of assessment altogether (Lancaster and Clarke, 2017). Simultaneously written end-of-term assessments are favourable in this aspect, but also often mandated and a necessity from an effort perspective. There are multiple kinds of cheating during concurrently written examination sessions – collaborative communication with other participating students during the process of writing, and using external sources of knowledge, including just-in-time contract cheating, which are then reflected in the answers instead of using the own knowledge and capabilities. Such cheating should be at least detected, and should ideally be discouraged and prevented from happening in the first place (van Ommering et al., 2018). Online examinations and e-

assessments, in particular concurrently digitally written ones, make it harder to both prevent and detect cheating due to the inability to holistically supervise dozens or hundreds of participating students and their environments. Such settings lead to more complex threat models (Küppers et al., 2020) that call for improved technology support. Fully individualised programmable exams (FIPEs) are suggested as a means to mitigate the likelihood of success of the first (collaborative) kind of cheating. With appropriate forms (such as avoiding binary 'is this correct' questions quickly answered by a web search) and conditions (such as open book rules), the second (external) kind can also be addressed. FIPEs ensure that all students get exam documents that are different to a controllable and permissable extent, skewing the cost to benefit ratio for collaborative cheating and quickly conveying the infeasibility to exchange information about solutions to anybody who tries. To make the adoption of FIPEs feasible and safe, a novel computer-supported tooling approach – a compiler for exam documents – is proposed.

In this paper, a categorisation and realisation of FIPEs is being introduced along with practical advice for educators. Specifically, the paper contributes a compiler-style software system design and implementation to aid in the automation of FIPE creation, dissemination to students and results scoring. Furthermore, the adoption of FIPEs in the examination of two modules is presented in the form of an experi-

---

[a] https://orcid.org/0000-0002-5312-5996

ence report, covering a programming examination in engineering sciences and a distributed systems examination in computer science. The paper is structured to first report about related work before presenting essential requirements. It then proceeds by introducing the categorisation system, presenting a working implementation, reporting on the experience and arriving at the conclusions.

2 RELATED WORK

Digitalisation is a major trend throughout all aspects of education. Computer-supported examinations are one of the most challenging aspects due to the inability to accept faults and ambiguities. The topic branches out into online examinations, remote proctoring and other forms of digitalised examination processes including task generation and correction (Andersen et al., 2020). A broad review of online examinations (Butler-Henderson and Crawford, 2020) reports that online examinations are preferred by both students and staff and that cheating is a problem as it is made easier by online examination although mitigations exist. The review points out that many mitigations are not effective, and that cheating as a social problem is best tackled by reducing propensity. In other words, altering the cost/benefit ratio by using multiple banks of questions, questions/answers randomisation and similar techniques is promising. However, the review does not point out any technological support to develop such formats with low effort from the educator side.

Randomised questions and topic assignments have been investigated in the examination preparation phase and are understood to be advantageous for the students who come across the same questions in the examination (Denny et al., 2017). For several decades, it is known that there are psychological factors involved that require a careful determination of the minimum threshold for correct answers to pass such randomised tasks (Hubálovská and Satánek, 1971). Intelligent random selection algorithms based on repositories of questions and answers are already discussed in the literature (Alghamdi et al., 2020). However, the use of artificial intelligence is risky from a legal perspective, thus educators might prefer a more controlled approach based on modest stochastics. Furthermore, the scope of the proposed algorithms is limited to repetitive questions/answers tasks. Cheating detection and prevention is understood to be among the best practices for conducting online examinations and e-assessments in general (Gruenigen et al., 2018). The detection of cheating is made easier

in the digital space through solution data analytics including text mining (Cavalcanti et al., 2012). Yet the technological infrastructure to reduce or even prevent cheating is found to be lacking in this survey, a gap that the work on FIPEs intends to narrow down.

A learning at scale study has investigated the degree of randomisation necessary to deter cheating on asynchronous exams (Chen et al., 2018), a setting that requires similar measures as online exams which, even when being conducted synchronously, may lead to the use of uncontrolled communication channels among students. The findings state that while cheater advantages remain even when using random orders and permutations, they drop considerably when using selection of tasks out of a pool.

On the practical side, there is a distinct lack of tooling to help educators with advanced cheat reduction and identification in online settings. Learning Management Systems (LMS) and Virtual Learning Environments (VLE) like Moodle or OLAT typically allow for shuffling questions and implementing question banks. They also support mixing answers within questions, and a certain extent of random questions in their quizzes and tests, including calculated questions with formula-based answers. Still, LMS generally do not support a controlled variability of questions and tasks, in particular for visuals beyond text, and tools like MoodleRanQ have been proposed to address that concern (Pérez-Solà et al., 2017). An orthogonal concern is that LMS/VLE enforce online education while many educators prefer to maintain on-site teaching and examination except for crisis situations. Other researchers have also pointed out the absence of digital forensics architectures to assist with incident investigation related to online examination fraud (Kigwana and Venter, 2018).

3 REQUIREMENTS

In order to achieve a helpful computer-supported process for educators and individualised yet balanced examinations for all students, seven requirements need to be fulfilled.

1. The individualisation only applies to the assignment of exam documents to students through random distribution. The documents themselves must not refer to any particular student, and must not be created with knowledge on which student will eventually receive it. This requirement ensures fairness, avoids discrimination, and reduces psychological stress among students.

2. The amount of different tasks and therefore the level of protection against cheating is limited

when following a pure permutation or shuffling approach. Rather, the combination of permutations with further value deviations, content randomisations and sampling (pooling) needs to be considered.

3. Differences need to balanced and controlled. While syntactic differences are introduced, the semantic equivalence of all exams in terms of difficulty, fairness and alignment with the educational content needs to be assured. This demands a controlled approach in which minor changes beyond pure permutations are bounded by deviation corridors and summarised for verification.

4. Permutations, deviations and randomisations need to be covered by the institutional legal frameworks to exclude the possibility of legal actions by students. This exclusion can only realistically be achieved if the requirements stated above are fulfilled. Institutional requirements such as 80% identical tasks (apart from strict permutations and numeric variables) need to be considered globally.

5. The manual crafting of all permutations, deviations, randomisations and samples is not feasible. Rather, programmatic generation of tasks through appropriate programming interfaces is essential to keep the effort low for educators and hence increase the chance of adoption.

6. Likewise, educators will have a much increased effort with corrections when the correct solutions necessarily differ as well. Again, a programmatic generation of reference solutions, individualised per student and consulted during the corrections, helps reducing the effort. Furthermore, while cheating probability should be reduced, it remains above zero and a solution should aid in uncovering it at the latest before grading.

7. On the technical side, a compiler-style FIPE framework should support constraint adherence and various forms of output, including PDFs to allow for using the framework to produce printed documents for on-site examinations.

4 CATEGORISATION SYSTEM FOR FULLY INDIVIDUALISED EXAMS

A categorisation system is introduced to link the basic exam modifications (permutation, deviation, randomisation and sampling) to common types of tasks and questions, following the achievement test categories from best practices (Piontek, 2008) (binary-

choice/true-false, multiple-choice, and more complex tasks) along with the exam layout as a whole.

Bounding these basic modifications are the degrees of equality according to pedagogic considerations and institutional policies. Fig. 1 positions them as trade-off between several competing fairness terms. The more liberty educators have in trading cheat risk reduction for appeal risk reduction, the more applicable far-ranging and cheat-reducing exam modifications can become, but the less balanced the cognitive demand on students may end up.
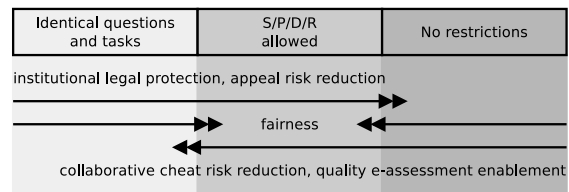

Figure 1: Three degrees of equality for exam questions and tasks from a legal perspective.

4.1 Global Considerations

Making the order of tasks unpredictable may cause slight confusion with students in case they are used to a certain order, but also brings advantages in terms of reduced cheating opportunities. A pure permutation can be coupled with deviations or shuffling in the form of random inclusion of similar (alternative) tasks. A declarative specification (shown in pseudocode) does not imply any order and thus leaves the ordering, and potentially the choice among alternatives, to the individualised creation of the exam documents.

```
Task1 AND Task2 AND (Task3a OR Task3B)
```

To avoid the consultation of online sources, educators must be careful to not ask any question that can quickly be looked up online. For instance, answers about the validity of a particular programming statement can easily be given with the appropriate statement execution tools, whereas filling gaps in a certain statement is harder to do because, in most cases, the required artificial intelligence can not yet be freely obtained through online tools. However, blindly randomising gaps means that students who collaborate could fill each other's gaps, a problem likely sharing aspects with differential privacy concerns (Holohan et al., 2017).

These modifications are considered state of the art in existing LMS/VLE and only included for completeness. A novel FIPE tool should support them, but also go beyond into the individual tasks and questions. Owing to recursion, the modifications also apply to tasks consisting of multiple sub tasks, such as

an exam with five tasks and the first task encompassing ten questions.

4.2 Binary and Multiple Choice

When exams contain binary choice and multiple choice questions, these are often grouped in blocks. A block with ten ordered binary choice (true/false) questions can lead to 1024 unique combinations through re-ordering alone. Further variety can be introduced by negating answers, which is straightforward to automate for any logic or arithmetic task by using appropriate markup as shown below. The appropriate value deviations can automated based on the data types of the marked results that default to true answers.

```
x = 17 + 4. x equals *21*.
True AND True equals *True*.
```

The result may look like the following:

```
Student 1:
1. x = 17 + 4. x equals 20. → False
2. True AND True equals True. → True
Student 2:
1. True AND True equals False. → False
2. x = 17 + 4. x equals 21. → True
```

Extending this concept to multiple choices per question is trivial and requires the specification of all choices. The same applies for variations of the task, as exemplified by the following question about correctness of a URL from the domain of network protocols, where each first element of a conjunction defaults to the correct answer:

```
(GET OR POST OR DELETE) /students("" OR
 "?action=delete"). Retrieves a list of
 students.
```

4.3 Complex Tasks

Beyond simple statements as questions and choices as answers, permutations also apply to more complex examination tasks such as text-based work and those based on advanced data structures including trees and graphs. In such tasks, the permutations can take various forms.

1. Declarative and imperative text structures. Associated tasks are filling random gaps and calculating results based on random facts contained in the text.

2. Graphs. Associated tasks are built around graph processing algorithms such as graph rewriting or identification of problems in the corresponding application domain, for instance in graphs representing software architectures that may contain scalability bottlenecks.

The following example uses markers to control the location as well as the permissible values of the quantitative permutation for numeric arguments. It also uses qualitative randomisation of three text parts, for which the educator needs to be carefully to balance the efforts.

```
Write a program that calculates the cumu-
lative weight of passengers and their
luggage and check whether the plane is
allowed to start with such a configu-
ration. The maximum take-off mass of the
plane is *[400,500,600]*kg.
*--
Use a list to represent the weights of
*[3-6]* passengers and crew, and a second
list to represent the associated luggage
weights.
*--
Use a tuple to represent the weights of
*[2-5]* passengers, and a scalar variable
to represent to pilot weight. Assume that
each passenger carries 10kg luggage.
*--
```

Apart from markup that combines ease of editing with still limited variability, arbitrary questions and answers can be constructed as pairs through regular programming means.

5 IMPLEMENTATION

To make the proposed categorisation and syntax approachable to educators, an examination individualisation software acts as compiler-style FIPE interface. A sample implementation is made available as open source software[1].

The implementation (Fig.2) assumes that the exam tasks are specified with the proposed syntax and other programmatic facilities. It is divided into two main parts. The first processes the input and produces both the permutated, deviated and randomised exam documents, to be distributed to the students, and associated reference solutions, to remain with the educator. At this stage, only the number of students are known, while no further information about them is included to avoid any bias or discrimination in the produced documents. The second step handles the distribution of the documents to students through a server-based provisioning to reduce the management effort for educators. In this step, the names of students are mapped to identifiers so that each student, by using

---

[1]FIPE implementation: https://github.com/serviceprototypinglab/fipe

the personalised identifier, is able to access the corresponding FIPE. The mapping is used for the automated provisioning process and beyond that remains available for the educator who, after grading the exams without necessarily knowing the student identity, is able to assign the scores or grades to the correct students.
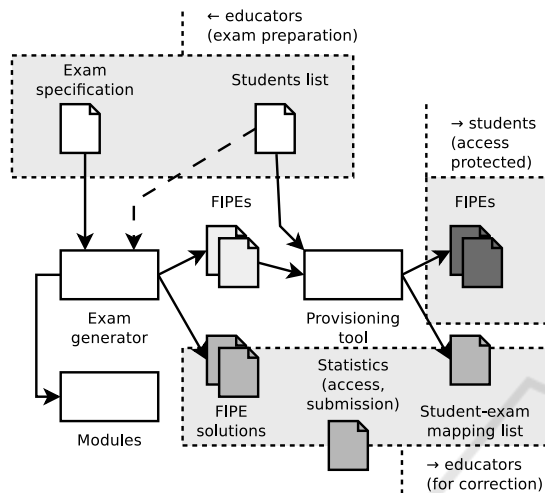

Figure 2: Implementation overview.

The entire process is conveniently wrapped into a single command, `fipe`, that is designed to only fail with actionable advice. Hence, it will guide educators from first use to ready-to-use examination in a streamlined process.

In the next three paragraphs, the possible presentation formats of the exam documents, the exam generation process and the subsequent exam provisioning process are explained in detail.

5.1 Presentation Formats

The concrete format of examination documents depends on the requirements and conventions. To make the system flexible and usable according to dominant e-assessment conventions, it should support at least the following formats:

1. Plain PDF. The PDF can either be printed and scanned, annotated on the screen, or form-filled. This format has the advantage that it will also serve the needs for traditional offline examination, and is thus of high value to educators when the modality of an exam has to be changed on short notice based on for instance the epidemiological situation.

2. Text files. For tasks in which plain text or template documents need to be provided, for instance source code based on templates.

3. HTML. In case the exam should be entirely conducted through web-based systems.

All exam documents are joined into larger files or archives containing all file formats. The output formats and their corresponding inputs are associated in Fig. 3.
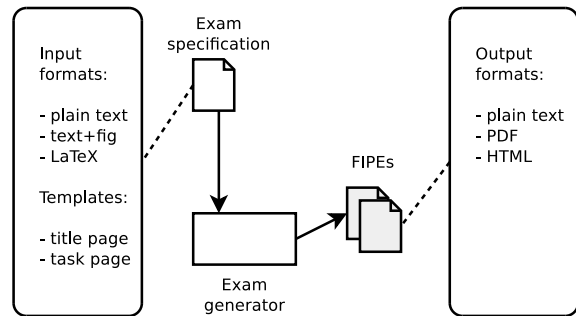

Figure 3: Formats overview.

5.2 Exam Generation

The exam generation happens programmatically. It consists of a number of modules (FIPE mods), one for each type of task, with a function `maketask` to call once per student. As the framework evolves, we expect to make dozens of task types available as inspiration and blueprint for other educators, while new modules can still be added anytime. At the time of publication, fifteen modules are available.

The modules work internally on permutated lists and conditionally included branches and, depending on the task type, are able to process the proposed syntax for controlled variability in task texts. For each task defined in the exam specification, this function returns three representations, the task itself, the solution and the achieved entropy. All tasks along with administrative information, including the solutions and the student identifiers, are stored in a directory that serves as entry point to the subsequent exam provisioning step.

For PDF exam documents, the output is stored as a number of LaTeX files which are compiled into the final document in a termination step after all tasks have been processed.

The entropy returned by each module is used to calculate the degree to which any two exams are identical. For each invocation, the identical points i are determined by the total points p and the entropy e as indicated in Eqn. 1. The entropy is used with a factor of two to account for potential differences in any two exams used for the comparison.

$$i = p - \begin{cases} p & \text{if } 2e > p \\ 2e & \text{otherwise} \end{cases} \tag{1}$$

5.3 Exam Provisioning

Due to the individualisation, each student needs to be informed about the specific modalities for retrieving the exam documents and for submitting the answers and solutions, depending on the exam format (PDF with optional template files or HTML).

The retrieval is the most critical moment in terms of system load, as many students will attempt to access the exam documents within a short period of time. Hence, the provisioning system needs to be sufficiently scalable to either concurrently serve static content (PDF, template files, HTML) or render dynamic content for direct submission (HTML only) in burst situations. The provisioning tool covers the following associated tasks.

1. Preparation of a web server including configuration for load spike at examination start time.

2. Upload of all exam documents with read protection. Each document, along with possible instructions on when and how to submit solutions, is stored in a dedicated folder with individualised and secret name.

3. Notification of students with individualised hyperlinks pointing to these folders along with indication of start time.

4. Removal of read protection at examination start time.

5. Indication of submission link, either on the same server or on a third-party system (Moodle, Teams Tasks and similar alternatives), for answers and solutions in reference to static exam documents.

6. Query of administrative information on behalf of the examiner such as overviews on students having and not having accessed the exam documents. The legal interpretation of any client-side issues of accessing the documents are handled depending on the institutional procedures.

5.4 Exam Corrections

The workflow for corrections requires opening a generated solutions document per student. This overhead is acceptable given the time spent per student is typically in the order of several minutes. Furthermore, per-task corrections, which are considered more fair when spanning multiple classes with different educators, are made possible by generating the solutions in the appropriate format and distributing the solutions documents to the responsible educators per task.

Any cheating requires a comparison of solutions as well as a consultation of auxiliar information such as network access logs in case of suspicions. The FIPE implementation contains a tool that parses web server log files and reveals anomalies. It offers two modes, multiple source hosts per student document and multiple student documents per single source host. The first mode is not suitable for open Internet environments due to students using multiple devices with different Internet connections as well as the involvement of major cloud providers in automated requests from their hosts (e.g. Google SafeBrowsing). The second mode is more suitable but still requires careful interpretation of results, for instance due to host re-allocation for mobile Internet users or shared proxy servers in dormitories.

6 EXPERIENCE REPORTS FROM APPLIED FIPEs

FIPEs have been practically validated on two occasions. Both times, they have shown to result in variable yet balanced exams adhering to institutional policies. A text-only (code and data files) representation has been used in a programming exam with $n = 380$ engineering sciences students (ES exam). A PDF-based representation has been used with $n = 28$ computer science students (CS exam). The difference in scale is exploited to find out how well FIPEs work for smaller and larger groups of students.

6.1 Tasks

The ES exam consisted of six tasks ranging from multiple choice questions (mixed pickles true/false) to complex tasks whose variability was driven by auto-generated data files containing different symbols. The CS exam had greater heterogeneity with nine tasks, including two based on autogenerated graphs representing software dependencies and workflow execution. Templates to set up such questions and tasks are available from the FIPE implementation. Hence, even though the initial effort to set up all FIPE modules to a combination of 70–80 points took several hours per task as the framework evolved, the creation of further exams based on these templates will be more economical by reducing that effort to less than half an hour per task. This effort is justifiable considering the overall time spent on assessment design. Moreover, the attractiveness of the framework will increase as more FIPE mod templates become available over time.

Fig. 4 shows two example graph-based task excerpts from the CS exam that is programmatically generated within the variability and scoring bounds. Each student gets a graph with three or four sequences, each consisting of two to four nodes, and needs to calculate the maximum parallel set of nodes as defined by the product of their weights. Such an individualised task is non-trivial to produce without a FIPE framework.
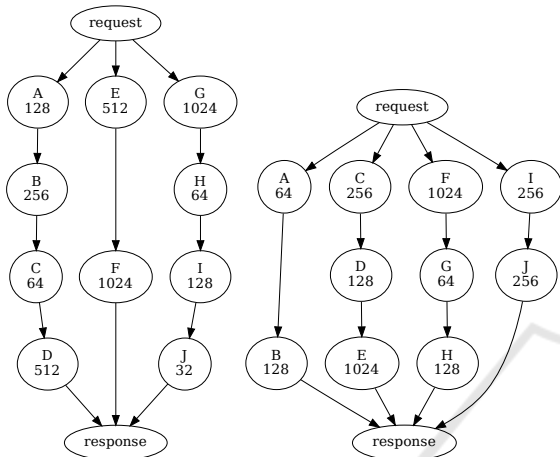

Figure 4: Variability in graph-based task.

6.2 Student Perception

The students were not informed in advance about the use of FIPEs. Rather, they were broadly advised that despite less privacy-invasive examination, allowing them to keep microphone and camera switched off, the educators would have means to detect cheating. Contrary to expectations, the FIPE effects were not brought up by participants, even when a feedback round was organised for the CS exam. This suggests that introducing FIPEs can be done in a non-intrusive way.

6.3 Access Behaviour

Exam documents for both occasions were hosted on the same institutional server. The document access workflow for the ES exam consisted of an HTML entry page, followed by two code file documents along with one data file. Fig. 5 shows the server load from the time of informing students of the (still blocked) link to the actual exam, covering a period of several days and notably spiking around the exam. Fig. 6 shows the exam spike in detail. The FIPE provisioning tool takes this sudden surge behaviour into account when producing the web server configuration. In the ES exam, all documents were delivered successfully without reliance on external services. An

examination of the access link has not revealed any link sharing between students.
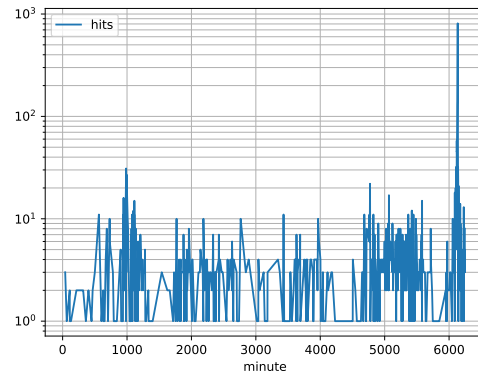

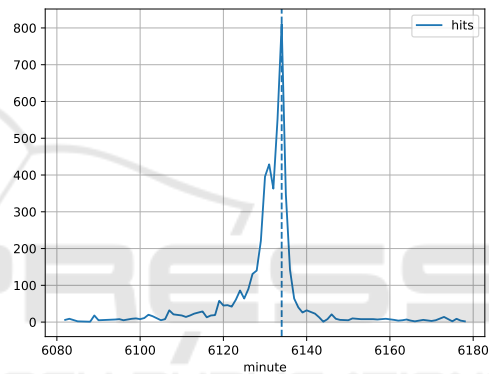Figure 5: Overall timeline from exam information to exam.

Figure 6: Zoom into narrow exam period; the dashed line marks the start of the exam writing.

6.4 Cheat Reduction and Exposure Discussion

In both examinations, suspicions were raised during the manual check of all solutions along with automated log file analysis. In one such case, the evidence led to the exposure of an actual case of unallowed collaborative cheating that took place during the exam writing period. A detailed analysis of the influence of FIPEs on cheat exposure, beyond a-priori reduction of cheat potential, is currently missing but will have to consider the trade-off that by limiting the surface area for plain copied solutions, the corresponding exposure will also become more difficult. The combination with log file analysis increases the chances of exposure, but relies on the curiosity of students to share links instead of undetectable direct sharing of documents.

7 CONCLUSIONS

This paper has introduced a practical path towards fully individualised exams beyond current question bank randomisation. The COVID-19 pandemic has shown that even universities emphasising quality presence teaching are occasionally subject to unanticipated online examinations and should therefore be in a strong position to flexibly choose between different examination modalities without sudden increases in cheating risks. The concept of FIPEs contributes to that flexibility while supporting institutional policies. The compiler-style software implementation further supports the flexibility by being able to produce variable PDF files that can be printed for classroom use or filled on screen in online settings, as well as other formats. It is currently undergoing further discussion and evolution with the aim of lowering the learning curve for educators and assembling further task types as modules. To foster the argumentation and provide a basis for further research, the FIPE implementation is provided as early-stage open source prototype at https://github.com/serviceprototypinglab/fipe.

REFERENCES

Alghamdi, A. A., Alanezi, M. A., and Khan, Z. F. (2020). Design and implementation of a computer aided intelligent examination system. *iJET*, 15(1):30–44.

Andersen, K., Thorsteinsson, S. E., Thorbergsson, H., and Gudmundsson, K. S. (2020). Adapting engineering examinations from paper to online. In *2020 IEEE Global Engineering Education Conference, EDUCON 2020, Porto, Portugal, April 27-30, 2020*, pages 1891–1895. IEEE.

Butler-Henderson, K. and Crawford, J. (2020). A systematic review of online examinations: A pedagogical innovation for scalable authentication and integrity. *Comput. Educ.*, 159:104024.

Cavalcanti, E. R., Pires, C. E., Cavalcanti, E. P., and Pires, V. F. (2012). Detection and evaluation of cheating on college exams using supervised classification. *Informatics in Education*, 11(2):169–190.

Chen, B., West, M., and Zilles, C. B. (2018). How much randomization is needed to deter collaborative cheating on asynchronous exams? In Luckin, R., Klemmer, S., and Koedinger, K. R., editors, *Proceedings of the Fifth Annual ACM Conference on Learning at Scale, London, UK, June 26-28, 2018*, pages 62:1–62:10. ACM.

Denny, P., Tempero, E. D., Garbett, D., and Petersen, A. (2017). Examining a student-generated question activity using random topic assignment. In Davoli, R., Goldweber, M., Rößling, G., and Polycarpou, I., editors, *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Ed-

ucation, ITiCSE 2017, Bologna, Italy, July 3-5, 2017*, pages 146–151. ACM.

Gruenigen, D. V., de Azevedo e Souza, F. B., Pradarelli, B., Magid, A., and Cieliebak, M. (2018). Best practices in e-assessments with a special focus on cheating prevention. In *2018 IEEE Global Engineering Education Conference, EDUCON 2018, Santa Cruz de Tenerife, Tenerife, Islas Canarias, Spain, April 17-20, 2018*, pages 893–899. IEEE.

Holohan, N., Antonatos, S., Braghin, S., and Aonghusa, P. M. (2017). (k, ε)-anonymity: k-anonymity with ε-differential privacy. *CoRR*, abs/1710.01615.

Hubálovská, H. and Satánek, A. (1971). To the question of the random choice of right answers and to the succes at programmed exams. *Kybernetika*, 7(4):328–333.

Kigwana, I. and Venter, H. S. (2018). A digital forensic readiness architecture for online examinations. *South Afr. Comput. J.*, 30(1).

Küppers, B., Eifert, T., Zameitat, R., and Schroeder, U. (2020). EA and BYOD: threat model and comparison to paper-based examinations. In Lane, H. C., Zvacek, S., and Uhomoibhi, J., editors, *Proceedings of the 12th International Conference on Computer Supported Education, CSEDU 2020, Prague, Czech Republic, May 2-4, 2020, Volume 1*, pages 495–502. SCITEPRESS.

Lancaster, T. and Clarke, R. (2017). Rethinking assessment by examination in the age of contract cheating.

Maroco, P., Cambeiro, J., and Amaral, V. (2019). A mobile system to increase efficiency of the lecturers when preventing academic dishonesty during written exams. In *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, volume 1, pages 236–241. IEEE.

Pérez-Solà, C., Herrera-Joancomartí, J., and Rifà-Pous, H. (2017). On improving automated self-assessment with moodle quizzes: Experiences from a cryptography course. In Ras, E. and Guerrero-Roldán, A., editors, *Technology Enhanced Assessment, 20th International Conference, TEA 2017, Barcelona, Spain, October 5-6, 2017, Revised Selected Papers*, volume 829 of *Communications in Computer and Information Science*, pages 176–189. Springer.

Piontek, M. E. (2008). Best practices for designing and grading exams. *Occasional Paper*, 24:1–12.

van Ommering, C. J., de Klerk, S., and Veldkamp, B. P. (2018). Getting to grips with exam fraud: A qualitative study towards developing an evidence based educational data forensics protocol. In Draaijer, S., Brinke, D. J., and Ras, E., editors, *Technology Enhanced Assessment - 21st International Conference, TEA 2018, Amsterdam, The Netherlands, December 10-11, 2018, Revised Selected Papers*, volume 1014 of *Communications in Computer and Information Science*, pages 199–218. Springer.