

A \LaTeX publishing workflow

Joppe W. Bos, Kevin S. McCurley

Abstract

We describe a new open-source system for copyediting and production of \LaTeX documents for a scholarly journal.

1 Introduction

The \LaTeX system is part of a large software ecosystem that includes packages, document classes, compilers, editors and bibliographic tools. This article describes another addition to the \LaTeX ecosystem, namely a software component to streamline publishing a journal or conference proceedings. The system we describe here is based on a metadata extraction technique that we described in a previous article [4], and it has been used in a workflow for the publication of the diamond open access journal titled *IACR Communications in Cryptology*.¹ The journal was launched in 2024, and has now published six issues with over 200 articles. While this journal has been based on a specific document class, we have also created a separate \LaTeX package called `metacapture` for the metadata extraction that allows the approach to be used with any document class.²

Much has been written about the cost of academic publishing, but most of that cost arises from the need to pay people for performing manual tasks. For this reason, we set a key goal for the system to minimize the amount of human labor required. Luckily, part of the power of \LaTeX is the ability for authors to have control over their final typesetting, while adhering to a style that is implemented in a document class. This can dramatically reduce the amount of human labor required for publishing an academic journal, because it eliminates any need for converting the author-supplied document into a form suitable for typesetting. This is only part of the process of publishing a journal or conference proceedings, and we also needed to optimize the amount of effort required for copyediting and production. More on the economics of our approach can be found in [3].

Most academic publishing is done by commercial publishers or scholarly societies who charge for the service and do not share the software that they use. By contrast, our system has been made open source so that others can potentially benefit from it. There are several other open source systems that can be used for publishing a journal. The most widely used systems are Open Journal Systems (OJS) from

the Public Knowledge Project³ and the Janeway system from Birkbeck University of London.⁴ These are monolithic systems in the sense that they implement an entire traditional workflow for publishing a journal. Unfortunately neither one offered much support for \LaTeX (e.g, see [9]). Moreover, neither one offers any support for alternative peer review workflows that are common in computer science.

2 Components of a publishing workflow

The workflow for publishing a journal or conference proceedings can be broken into three phases shown in Figure 1. The interaction between these three phases is relatively simple, because each phase simply passes data on to the next phase.

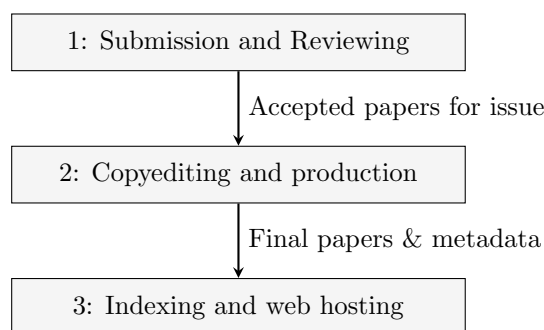


Figure 1: The three phases of a publishing workflow.

One reason why it is convenient to think of a workflow this way is that it allows us to break down the publishing process into separable components, which allows us the flexibility to satisfy different publishing requirements. In our workflow, the three phases are implemented as completely independent web servers, but they can be run on the same machine as they do not require many resources. The majority of our effort has been focused on our \LaTeX package for metadata extraction and our implementation of the server for the copyediting and production phase, but we describe all three components below.

3 Submission and reviewing

Peer review systems can vary quite a bit. The traditional workflow of an academic journal allows authors to submit at any time. This is the submission workflow implemented in OJS and Janeway. Some journals have adopted a submission model that is similar to conferences, where all articles for a given issue of a journal are submitted by the same deadline, and the reviewing takes place on a predictable schedule in the same manner as a conference. The subject of

¹ See cic.iacr.org/.

² See github.com/IACR/latex/tree/main/metacapture.

³ See pkp.sfu.ca/software/ojs/.

⁴ See janeway.systems/.

peer review has recently become a focus of attention in which there is a fair amount of experimentation with things such as open reviewing [1]. We believe that this variability in peer review makes it important to remain flexible when designing software for a publishing workflow.

In the case of our journal, we use a minor modification to the open source version of HotCRP⁵ for the submission and reviewing phase. The submission form accepts a PDF file, along with the title, author names, and author affiliations. The PDF itself is anonymized, and the author and affiliation information is captured separately in order to identify potential conflicts of interest in reviewing.

One problem that we have observed in publishing workflows over the years is that the metadata supplied in the submission phase is often inaccurate or incomplete by the time the article is finally accepted. This is one of the reasons why we developed a system for automatically extracting the metadata from the L^AT_EX sources at compile time. This assures that there is a single source of authority for the metadata during the later production phase. It could also be used in the submission phase, with anonymization of the PDF.

The communication between the first phase and the second phase is implemented simply by providing an authenticated url to the author of an accepted paper for them to upload their L^AT_EX sources to the second phase. It should be easy enough to modify OJS or Janeway in the same way if a different submission and review system is desired. The communication between the second and third phases is similarly accomplished by pushing an authenticated bundle of metadata and final versions of the articles.

4 Copyediting

The purpose of copyediting is to improve the style, formatting, grammar, and readability of published articles. There is no universal agreement about the scope or value of copyediting, and many journals and conference proceedings put almost no effort into it. The style we use might be described as “editorial light” as described in the AMS guide [8], but clearly there is value in having independent human effort put into the process. Unfortunately it can be expensive to pay a professional copyeditor. Some societies like ACM⁶ and AMS⁷ offer access to discounted third-party copyediting services for authors, but authors are responsible for paying for the service. We follow the same strategy to push much of the responsibility

for copyediting back to authors. The result may be that some articles end up being published with grammatical errors or low-quality writing, but in our view this reflects more on the effort put in by the author(s) than the journal. This is not a new problem, and to quote Charles Babbage from 1830 [2]:

“... if members were in the habit of communicating their papers to the Society in a more finished state, it would be attended with several advantages;...”.

When an article is accepted, the author uploads the “candidate” version of their article to the second phase of the workflow. The upload consists a zip file containing a file `main.tex` along with any other T_EX files that are required to compile their paper. When the author uploads their candidate version, the server immediately compiles it using a containerized version of the T_EX Live distribution. The containerization is an important aspect of the design in order to protect against malicious uploads [7]. Not only does the container protect against malicious modification of the server filesystem, but we also protect against infinite loops by placing a time limit on the compilation.

The compilation of the author’s source will often identify problems with the document. Some of these problems can be identified from the log files of the compilation, but the logs are so verbose and hard to understand that authors mostly ignore them. In order to overcome this, we wrote a parser for the output logs from the L^AT_EX and B_IB_TE_X compilations. This extracts only the most serious errors and warnings and presents them to the author in a more readable structured format with pointers to where the problem occurs in the source files and the PDF. A view of the author’s feedback can be seen in Figure 2. The extraction of metadata at compilation time allows us to automatically generate the HTML rendering. Part of the review process involves having the author review this HTML rendering.

We require the author to upload B_IB_TE_X file(s) so that it can be parsed into a more structured format. This also allows us to easily check for missing fields, and it also allows us to produce an HTML version of the references. We have found that one thing authors often overlook is the inclusion of DOIs for journal article references, so we flag those as warnings. We also have a domain-specific user tool for the author to automatically find these DOIs via search, but this could also be built with the Crossref or OpenAlex APIs.

The goal of the system is to have the author proactively fix errors that are found during compilation. We do not provide an online editing system to fix their problems, but they can fix these offline

⁵ See github.com/IACR/hotcrp3.

⁶ See acm.org/publications/pacm/pacm-guidelines.

⁷ See www.ams.org/arc/journals/index.html.

The screenshot shows the IACR Publishing Portal interface. On the left, a green banner says "You may still resubmit your paper to fix warnings". Below it, a green box contains instructions: "Congratulations - your paper was compiled. The next steps are: 1. Check any warnings below. Some of them may be flagged by the copy editor for you to fix. 2. check the HTML, to make sure it looks ok. This is essentially what will appear on the webpage for your paper. 3. check your PDF to make sure it looks ok. 4. Check the metadata in the tab on the right. 5. Once you have checked the PDF, warnings, and metadata, you can submit it for copy editing with the button at the bottom of the metadata tab." Below this is a "Warnings:" section with a list of yellow boxes, each containing a warning message, log line number, PDF page number, and source file name. The right side of the interface has tabs for "HTML", "PDF", "Compilation log", "Inputs", and "Metadata". The "HTML" tab is active, showing a preview of the article's HTML page, including the title "Lowering the Cost of Diamond Open Access Journals", author names (Joppe W. Bos and Kevin S. McCurley), an abstract, and a list of references.

Figure 2: View shown to the author with structured feedback from the \LaTeX and \BibTeX logs. The left column shows a sequence of items that were flagged during the compilation and extracted from the \LaTeX and \BibTeX logs. The right column has tabs to view the HTML, PDF, logs, etc. We rely on authors to fix these problems, but we provide them with a much better way to spot the problems.

using their preferred environment, and keep uploading their source files until they are satisfied with the result. Once this happens, the article is recompiled with line numbers in it and it is picked up by a copyeditor. The copyeditor looks at both the items flagged from the \LaTeX compilation, but also performs a few simple checks to confirm that the paper conforms to the style of the journal.

Some articles are submitted with violations of the default style. Sometimes that is deliberate [5], but we have also observed that sometimes it is accidental because of poor cut-and-paste habits from previous articles. We protect against some of these issues by excluding some packages like `savetrees` from our \TeX Live docker image. In some cases a good \LaTeX document class can automatically detect some violations of the style. For example, the document class

could disable the `\cite` macro inside the `abstract` environment, since bibliographic references in an `abstract` are generally regarded as poor style.

We do *not* assume that the copyeditor is skilled in \LaTeX , and we do *not* expect the copyeditor to edit the author's source files. In practice our copyeditors are volunteer scholars from the field, so they have at least some basic understanding of what causes problems in the article. Unfortunately we have also observed that authors often create extremely complicated \LaTeX constructions, so it is better left to the author to fix their problems anyway. We merely send items back to the author listing the page number, line number, source file, and line in the source file where the problem should be fixed. It is the responsibility of the author to fix things, and we have found them to be mostly cooperative in this process.

The screenshot displays the IACR CIC copyeditor interface. The main window is split into two panes: 'Original' (left) and 'Final' (right). Both panes show a PDF preview and the underlying LaTeX source code. The 'Original' pane shows a table with parameters and matrix dimensions, and a section titled '2 Background'. The 'Final' pane shows the same content but with changes highlighted. On the right side, there is a 'Responses' panel with a status message: 'Status: Pending final review, author email: foo@digicrime.com'. Below this, there are two response entries for 'Page 1, line 1'. The first response is 'Response: Agreed to fix.' and the second is 'Response: Agreed to fix.' There are buttons for 'Approve for publishing' and 'Request more changes'.

Figure 3: View shown to the copyeditor after the author has sent responses to all items flagged for changes. The copyeditor can view both the before and after of the PDFs as well as the differences in the \LaTeX sources from the two versions. In this case the author agreed to both changes that were requested, so the copyeditor can check the result.

The author is required to respond to each item that is sent to them by the copyeditor, either to say they will fix it, they will not fix it (and why), or to request further clarification. This can theoretically generate another round of copyediting where the editor sends it back to the author to fix. The view in Figure 3 shows that the copyeditor is shown after the author has responded to all of the items they were sent during the copyediting phase.

A demonstration video of this copyediting and production system is available at www.youtube.com/watch?v=Ki8Qcai9klg, and a test instance is currently available online at publishtest.iacr.org/. The metacapture package is available at github.com/IACR/latex/tree/main/metacapture.

5 Production of final versions

After the article is accepted by the copyeditor, it moves to the production and indexing phase. The article is recompiled with publisher-supplied metadata elements such as a DOI, Crossmark url, volume number, issue number, dates, etc. Some elements, such as the acceptance date and submission date, are determined in the first phase, and are authenticated by the system.

There are several limitations of the PDF format that we produce as the final version of the article. One limitation is the desire for an accessible format, but that is being addressed by the \LaTeX team in their tagged PDF project. Another limitation of PDF

files is that they are inconvenient when viewed on small screens. For this reason, some journals are now producing HTML output from the author-supplied document. Unfortunately the production of HTML from \LaTeX is far from straightforward [6]. Part of the problem is that authors often tune their documents to the PDF format. Moreover, the converters from \LaTeX to HTML are not reliable enough to use in an automated mode, and most HTML that is produced from \LaTeX requires some manual tuning to produce a good result. This is inconsistent with our goal of reducing the amount of human effort required, so we have decided to limit our system to producing only an HTML page with title, authors, abstract, and bibliographic references. If \LaTeX converters ever become reliable enough to use without human intervention, then that would be a desirable outcome because HTML is much better suited to being read on different screen sizes.

We believe that if \LaTeX is to thrive in the modern age of publishing, then it should be less tied to paper and PDF, and more reflective of document structure. There are no intrinsic problems with \LaTeX that cannot be solved, but it will require authors to think differently about their articles when they produce them, and to have more consistent and robust software to produce HTML. As a start toward this, we have constructed a \LaTeX XML package to produce front matter from the metadata captured by the metacapture package.

Our journal runs in a traditional model where articles are bundled into issues and issues are bundled into volumes. If an author misses the final version deadline for the issue, then the article is kept behind in the system for publication in the next issue. When the editor decides that the issue is complete, they export a zip file to the indexing and hosting phase that contains the final PDF versions of the papers, along with a JSON file with metadata for each article, and some metadata to describe the issue. This exported bundle becomes the input to the final indexing and web hosting phase.

6 Indexing and hosting

When an issue has been exported from the copyediting and production phase, there are a few automated steps that finally publish the issue. One step is to assign a permanent url to the article and register this for the DOI. When DOIs are registered, the system sends XML metadata to Crossref that contains the usual title, abstract, authors, and affiliations, but also funding information and a structured form of bibliographic references that is produced from our parsed version. The fact that our document compilation automatically captures all of the required metadata eliminates any need for human labor in this process, and it becomes simply the click of a button to import the issue and start serving it. It's worth noting that the `aomart` document class implements a similar metadata extraction method.⁸

The system for indexing and hosting is much simpler than the system for copyediting and production. It is essentially just a web server with the ability to import some content and register the DOIs with Crossref. We plan to remove dependencies on our journal from this server and then release it as open source so that it would also be useful for other journals.

7 Future directions

Our current system is highly dependent on the metadata extraction technique described in [4] and now implemented in the `metacapture` package. We plan to refine the `metacapture` package before releasing it on CTAN, and we intend to use this when we move two other IACR journals to the copyediting system. In the long term we hope that the `metacapture` package can also enable other journals to apply their own styling and make use of the copyediting system. We also plan to define a better API for our system to be used as a plugin to other peer-review systems such as OJS and Janeway.

Some tasks that were traditionally done by copyeditors have ended up being automated (e.g., spelling correction). We have not accomplished full automation, but we have at least reduced the amount of human labor required in practice. The last few years has seen enormous advances in use of large language models, and we are hopeful that this can become an important tool for copyediting in the future.

References

- [1] B. Aczel, A.S. Barwich, et al. The present and future of peer review: Ideas, interventions, and evidence. *Proceedings of the National Academy of Sciences*, 122(5):e2401232121, 2025. doi.org/10.1073/pnas.2401232121
- [2] C. Babbage. Reflections on the decline of science in England, and on some of its causes. London, 1830. www.gutenberg.org/files/1216/1216-h/1216-h.htm
- [3] J. Bos, K.S. McCurley. Lowering the cost of diamond open access journals, 2025. arxiv.org/abs/2504.10424
- [4] J.W. Bos, K.S. McCurley. Metadata in journal publishing. *TUGboat* 44(1):71–76, 2023. doi.org/10.47397/tb/44-1/tb136bos-metadata
- [5] W. Duivesteijn, S. Hess, X. Du. How to cheat the page limit. *WIREs Data Mining and Knowledge Discovery*, 10(3):e1361, 2020. doi.org/10.1002/widm.1361
- [6] C. Frankston, J. Godfrey, et al. HTML papers on arXiv – why it is important, and how we made it happen, 2024. doi.org/10.48550/arXiv.2402.08954
- [7] G. Lacombe, K. Masalygina, et al. Can you accept LaTeX files from strangers? Ten years later, 2021. doi.org/10.48550/arXiv.2102.00856
- [8] M. Letourneau, J.W. Sharp. AMS style guide: Journals, 2017. www.ams.org/publications/authors/AMS-StyleGuide-online.pdf
- [9] C. Schöch, P. Trilcke, E. Gius. Editorial. *Journal of Computational Literary Studies*, 1(1), Apr. 2023. doi.org/10.48694/jcls.3627

- ◇ Joppe W. Bos
NXP Semiconductors, Belgium
joppe.bos (at) nxp dot com
- ◇ Kevin S. McCurley
San Jose, California
publishing (at) digicrime dot com

⁸ See ctan.org/pkg/aomart.