How we use LaTeX in the CrypTool project

Doris Behrendt, Bernd Busse

Contents

Abstract

The CrypTool project with its long history uses LaTeX in two different scenarios: First, we use KaTeX for maths on our CrypTool online web page. Second, we used LuaLaTeX for building the recently re-edited CrypTool book (in German [3] and English [4]).

In this talk we limit ourselves to the topic of mathematical typesetting for interactive web applications in the browser with KaTeX.

1 The CrypTool project

You can visit the CrypTool project landing page here: `www.cryptool.org/en`. From there, you can start to explore our many interesting subprojects.

As the name already suggests, we offer tools for cryptography. They are free of cost and advertising and we do not collect personal data. We were fully open source once, but not anymore because "unfriendly" actors copied our whole web page, flooded it with ads and we had to deal with the angry messages from our user base all over the world.

Feel free to send us an e-mail and we will give you access to our private GitHub repository.

We use Umami (`umami.is`) for GDPR compliant analytics, and as you can see in fig. 1, we have about 1k visitors per work day and not so many on the weekends.

Our work splits into maintenance of existing software and development of new products — in the last years, mainly online web apps. We have a collection of, as of today, 21 web apps on CrypTool-Online (CTO), a part of which is shown in fig. 2. Two more web apps will be released soon, one quite large that dives deep into the recently NIST-standardised post-quantum key encapsulation mechanism ML-KEM [8] and one for an undergraduate audience that introduces the BB84 quantum key distribution protocol [1] in a playful way.

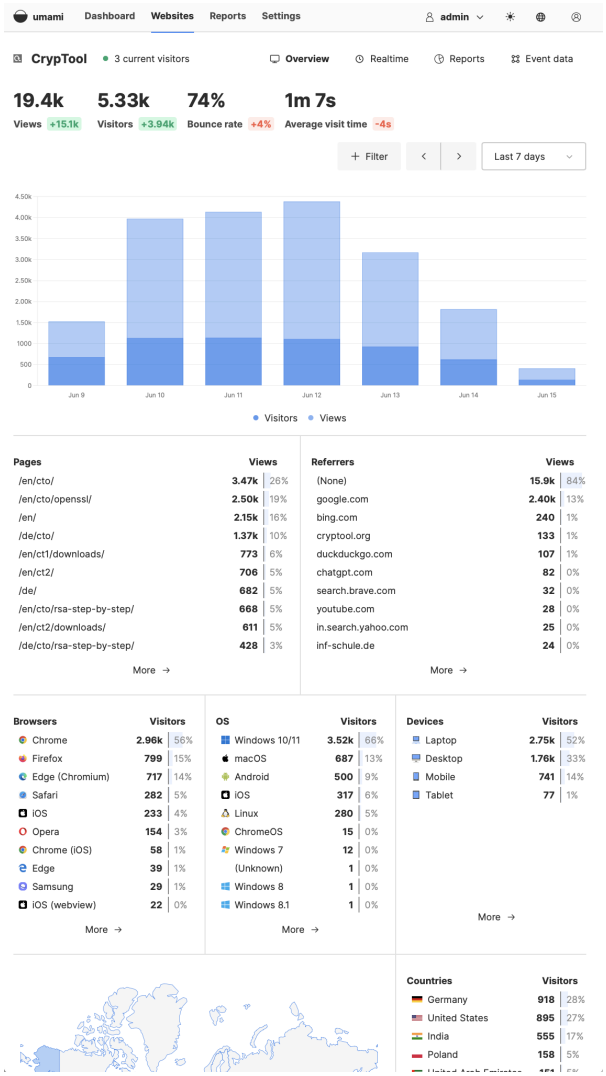

Figure 1: CrypTool usage statistics with Umami

2 Mathematical typesetting in the browser

There are three common ways to generate mathematical typesetting in the browser: MathJax [6], KaTeX [5], and MathML [7].

MathML is an XML dialect for describing mathematical expressions in the typical angle-bracket syntax, but it apparently has not fully caught on. MathML was developed as a subset of HTML5, but it was kept too general and therefore was never implemented consistently. With MathML Core, this is supposed to improve, as it builds upon LaTeX-style styling. A list with details on the supported elements can be found here: `developer.mozilla.org/en-US/docs/Web/MathML/Reference/Element`.

However, the browser must also have a technique to turn this "character salad" into a readable formula, which ultimately brings one back again to HTML

doi.org/10.47397/tb/46-2/tb143behrendt-cryptool

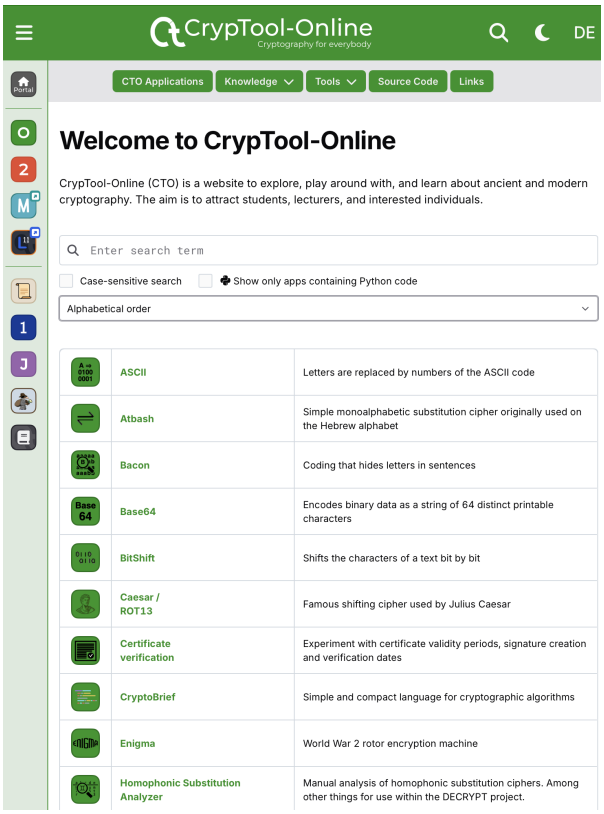

Figure 2: CrypTool-Online

and CSS — essentially similar to what KaTeX and MathJax do.

Both KaTeX and MathJax are based on JavaScript, with their input being essentially TeX math-mode text. Their main difference is performance: KaTeX is newer and faster. It renders formulas directly into HTML and CSS. KaTeX does generate MathML as well, but only as a fallback for screen readers for accessibility (example in the next section). MathJax, however, supports more LaTeX commands.

Responsive behavior in modern web browsers is achieved through the so-called Flexbox layout ("CSS Flexible Box Layout module"), which is a CSS layout model. As we all know, boxes exist in TeX as well. However, for TeX boxes there is no dynamic resizing, which is what enables responsiveness in browsers. KaTeX does not use Flexboxes for rendering, but instead does it "manually" using `line-height`, `font-size`, and font-size-dependent positioning with `margin`, `height`, `top` in CSS. The LaTeX glue is simulated in KaTeX with static spacing.

3 KaTeX CLI

You can use KaTeX on the command line as in fig. 3 by following the instructions of the documentation on their website: `katex.org/docs/cli`.

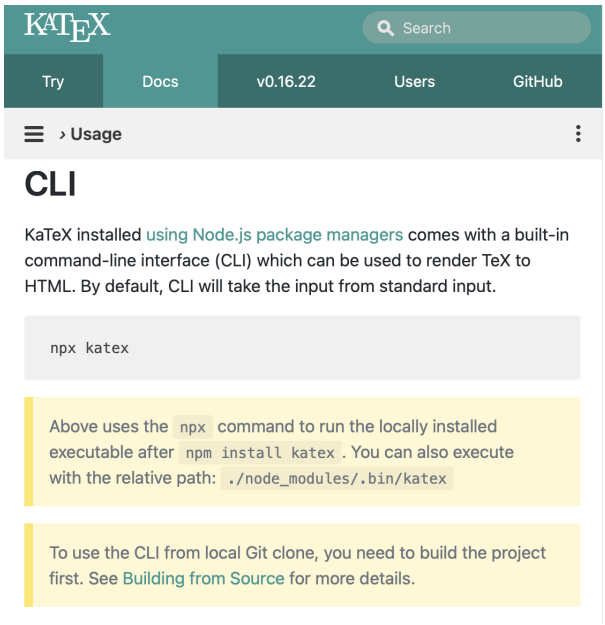

Figure 3: KaTeX CLI documentation

If you already have the `Node.js` package manager `npm` available, you can install the current version of KaTeX as described via `npm install katex` (listing 1), and then execute it in the current working directory, here `tug25`, using the command `npx` to give our example input (`a^2+b^2=c^2`) (listing 2). In this case, the `katex` package is located in the home directory under `node_modules`, not in a subdirectory of the working directory.

The result is shown in listing 3 and fig. 4 (bottom): The output shows the MathML output from line 1 in the listing while KaTeX's HTML output corresponding to the very long line 2 in the listing occurs on the right — without any CSS import, so it's not rendered as we would expect. If the directive

```
<link rel="stylesheet"                                    1
      href="node_modules/katex/dist/katex.css"            2
      type="text/css" />                                  3
```

is added near the top of the HTML file, the rendering looks like that in fig. 4 (top), with the MathML output being suppressed.

4 CrypTool-Online technologies

At CrypTool, we do not follow the approach of KaTeX's so-called in-browser rendering as shown in listing 4, but instead use server-side rendering. In order to explain how we implement this, let's start with a short overview of the web technologies we use. We also describe them in our GitHub repository; see fig. 5.

The first in the list on the screenshot is `Next.js` [9], which is a React framework for building full-stack

```
tug25 % npm install katex

added 7 packages, and audited 30 packages in 601ms

1 package is looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

Listing 1: `npm install katex`

```
tug25 % npx ~/node_modules/.bin/katex \
            -o pythagoras.html
a^2+b^2=c^2
tug25 % vi pyth*html
tug25 % open -a safari pythagoras.html
```

Listing 2: `npx` workflow with our example input

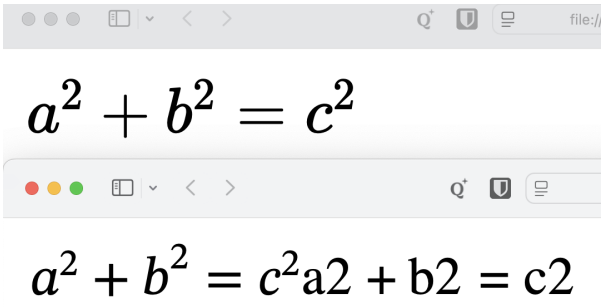

Figure 4: Viewing the output in Safari: with CSS (above), and without (below)

web applications. React lets you build user interfaces out of individual pieces, called components, that then are combined into our apps and pages.

React components receive data and return what should appear on the screen. You can pass them new data in response to an interaction, as when the user types into an input field. React will then update the screen to match the new data [10].

Components are JavaScript functions that combine GUI markup structure with code logic. Their source files are given the suffix `.jsx`. Examples of such components are shown in listings 7 and 8.

Webpack [11] is a module bundler for JavaScript. It parses imports in the code, optimizes away unused code, and divides the remaining code into "bundles" and "chunks" that can run in browsers. It's automatically executed when running `Next.js`.

Chakra-UI [2] is a set of pre-styled components for React. You can choose from among the predefined Chakra-UI's components, which already include

```
<span class="katex"><span class="katex-mathml"><math xmlns="http://www.w3.org/1998/Math/MathML"><semantics><mrow><msup><mi>a</mi><mn>2</mn></msup><mo>+</mo><msup><mi>b</mi><mn>2</mn></msup><mo>=</mo><msup><mi>c</mi><mn>2</mn></msup></mrow><annotation encoding="application/x-tex">a^2+b^2=c^2
</annotation></semantics></math></span><span class="katex-html" aria-hidden="true"><span class="base"><span class="strut" style="height:0.8974em;vertical-align:-0.0833em;"></span><span class="mord"><span class="mord mathnormal">a</span><span class="msupsub"><span class="vlist-t"><span class="vlist-r"><span class="vlist" style="height:0.8141em;"><span style="top:-3.063em;margin-right:0.05em;"><span class="pstrut" style="height:2.7em;"></span><span class="sizing reset-size6 size3 mtight"><span class="mord mtight">2</span></span></span></span></span></span></span></span><span class="mspace" style="margin-right:0.2222em;"></span><span class="mbin">+</span><span class="mspace" style="margin-right:0.2222em;"></span></span><span class="base"><span class="strut" style="height:0.8141em;"></span><span class="mord"><span class="mord mathnormal">b</span><span class="msupsub"><span class="vlist-t"><span class="vlist-r"><span class="vlist" style="height:0.8141em;"><span style="top:-3.063em;margin-right:0.05em;"><span class="pstrut" style="height:2.7em;"></span><span class="sizing reset-size6 size3 mtight"><span class="mord mtight">2</span></span></span></span></span></span></span></span><span class="mspace" style="margin-right:0.2778em;"></span><span class="mrel">=</span><span class="mspace" style="margin-right:0.2778em;"></span></span><span class="base"><span class="strut" style="height:0.8141em;"></span><span class="mord"><span class="mord mathnormal">c</span><span class="msupsub"><span class="vlist-t"><span class="vlist-r"><span class="vlist" style="height:0.8141em;"><span style="top:-3.063em;margin-right:0.05em;"><span class="pstrut" style="height:2.7em;"></span><span class="sizing reset-size6 size3 mtight"><span class="mord mtight">2</span></span></span></span></span></span></span></span></span></span></span>
```

Listing 3: KaTeX's HTML output: explicit positioning.

```
<html>
  <head>
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/katex@0.16.22/dist/katex.min.css" integrity="sha384-5TcZemv2l/9On385z///+d7MSYlvIEw9FuZTIdZ14vJLqWphw7e7ZPuOiCHJcFCP" crossorigin="anonymous">
    <!-- The loading of KaTeX is deferred to speed up page rendering -->
    <script defer src="https://cdn.jsdelivr.net/npm/katex@0.16.22/dist/katex.min.js" integrity="sha384-cMkvdD8LoxVzGF/RPUKAcvmm49FQ0oxwDF3BGKtDXcEc+T1b2N+teh/OJfpU0jr6" crossorigin="anonymous"></script>

    ...
  </head>
  ...
</html>
```

Listing 4: In-browser rendering, as described in `katex.org/docs/browser`

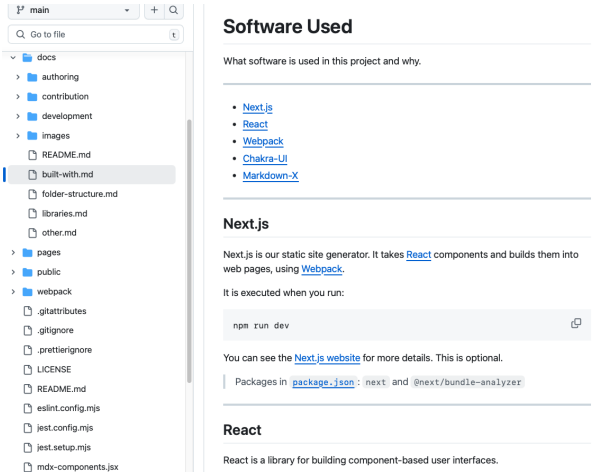

Figure 5: CTO project documentation on GitHub

the necessary logic and have examples in their documentation, as shown in fig. 6.

Behind the scenes, Chakra-UI uses the CSS-in-JS library Emotion (`emotion.sh/docs/introduction`) for creating the necessary CSS on the fly.

It is also possible to configure your own components, which is what we do for some special needs in CTO, e.g. the KaTeX component in listings 7 and 8.

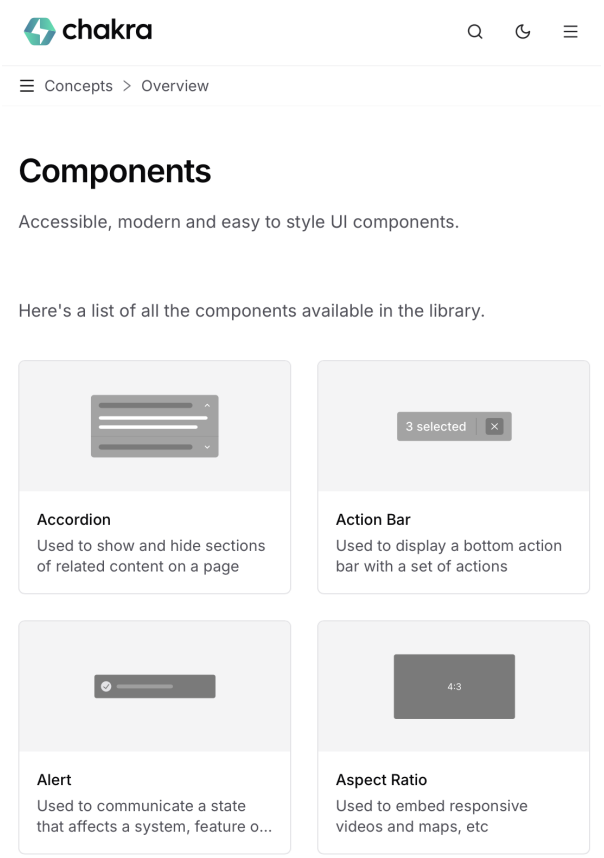

Figure 6: Chakra-UI components from `chakra-ui.com/docs/components/concepts/overview`

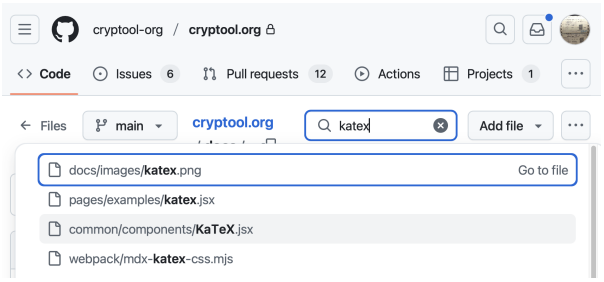

Figure 7: KaTeX files in CTO

5 KaTeX in CTO

When you search our GitHub Repo for filenames containing "katex", you will find four different files as shown in fig. 7.

When building the website locally as a developer, more `katex` files come into play; these are shown in listing 5.

On our repo you find only the code that is needed to build the website and was written by the CrypTool developers. For development you have to install the project's `Node.js` package dependencies using

```
cryptool.org % find . -type f -name "*katex*"
./node_modules/katex/types/katex.d.ts
./node_modules/katex/dist/katex.css
./node_modules/katex/dist/katex.js
./node_modules/katex/dist/katex.min.css
./node_modules/katex/dist/katex.min.js
./node_modules/katex/dist/katex.mjs
./node_modules/katex/katex.js
./node_modules/katex/contrib/copy-tex/katex2tex.js
./node_modules/katex/src/styles/katex.scss
./docs/images/katex.png
./webpack/mdx-katex-css.mjs
./pages/examples/katex.jsx
```

Listing 5: Finding KaTeX files after local build process

the command `npm clean-install`, which will generate the folder `node_modules`, get — amongst many others — the prebuilt `dist/katex` folder from the external KaTeX npm-website `www.npmjs.com/package/katex` and copy them there.

Webpack then extracts the necessary rules from `katex.min.css`, optimizes them, and concatenates the result into a single bundle.

Let's go back to the four files in our repo from fig. 7. The second file in fig. 7 — `katex.jsx` — is a sample component for new CTO contributors that shows them how to use KaTeX in our apps, which you can see in listing 6. The output should look like this png (see also fig. 7):

$$a = b \cdot c = 3 \cdot 2 = 6$$

Two different use cases occur for using KaTeX in our web apps: Inside markdown (`.mdx` files) and in JSX components.

When using markdown, double dollar signs `$$` work analogously to standard code tags. The only difference from what we are used to in LaTeX is that `$$x^2$$` produces inline math and

```
$$
x^2
$$
```

produces display math.

When `Next.js` parses an `.mdx` file with Webpack, they're passed through special plugins that analyze the input for the use of KaTeX formulas after they've been processed to automatically add the required CSS stylesheet.

The second use case of KaTeX is in JSX components. For this you have to import the `TeX` tag which is CTO-specific. It's defined in the component file

```
/**
 * Example for ‘common/components/KaTeX.jsx‘
 with variable values
 */

import { Box, Separator } from "@chakra-ui/react"

import TeX, { latex } from "common/components/KaTeX"

export default function TeXample() {
  const b = 2
  const c = 3
  const a = b * c

  const formula = latex‘a = b \cdot c = ${b} \cdot ${c} = ${a}‘

  return (
    <>
      <TeX math={formula} />
      <Separator />
      <TeX block math={formula} />
      <Separator />
      <TeX block="aligned" math={latex‘a &= b\\c \cdot e&= 3 \cdot 110‘} />
      <Separator />
      <Box width="10ch" outline="1px solid gray">
        <TeX math={formula} whiteSpace="nowrap" />
      </Box>
    </>
  )
}
```

Listing 6: CTO KaTeX example file `katex.jsx`

`KaTeX.jsx` beginning at line 51, which is shown in listings 7 and 8 and used in listing 6 from line 18 on.

If you use variable values in your `TeX` expressions, use the `latex` helper, which is defined in line 14 in listing 7, then imported in line 7, and finally used in line 14 in listing 6.

The interesting part is line 37 in listing 8 where we use the `renderToString` method. We then embed the HTML string that this method returns directly into our component (also listing 8, line 77). This is the earlier mentioned server-side rendering that is described in the API documentation on the KaTeX website.

```
"use client"

/** React component for rendering math using KaTeX */
import KaTeX from "katex"

import { forwardRef, memo, useMemo } from "react"
import { chakra } from "@chakra-ui/react"

import useStableValue from "common/hooks/useStableValue"

import "katex/dist/katex.min.css"

/** Tag for raw TeX/LaTeX strings which support variable substitution */
export const latex = (...s) => String.raw(...s).replace("\\‘", "‘")

const BLOCK_ENVIRONMENTS = Object.freeze([
  "equation",
  "split",
  "align",
  "gather",
  "align*",
  "gather*",
  "aligned",
  "gathered",
])
const useEnvironment = (content, environment) => {
  if (BLOCK_ENVIRONMENTS.includes(environment)) {
    return ‘\\begin{${environment}}${content}\\end{${environment}}‘
  } else return content
}

const useKaTeX = (content, options) => {
  const stableOptions = useStableValue(options)

  return useMemo(() => {
    try {
      const html = KaTeX.renderToString(content, stableOptions)
      return [html, undefined]
    } catch (error) {
      if (!(error instanceof KaTeX.ParseError || error instanceof TypeError))
        throw error

      return [error.message, error]
    }
  }, [content, stableOptions])
}
```

Listing 7: CTO KaTeX component file `KaTeX.jsx`

```
/**
 * Component wrapper around KaTeX.renderToString()
 */
export const TeX = memo(
  forwardRef(function TeX(
    { children, math, block, errorColor, renderError, settings, ...props },
    ref
  ) {
    const displayMode = !!block
    const content = useEnvironment(children ?? math, block)

    const [html, error] = useKaTeX(content, {
      displayMode,
      errorColor,
      throwOnError: !!renderError,
      ...settings,
    })

    const Component = useMemo(
      () => chakra(displayMode ? "div" : "span"),
      [displayMode]
    )

    if (error && typeof renderError === "function") return renderError(error)

    return (
      <Component
        {...props}
        ref={ref}
        dangerouslySetInnerHTML={{ __html: html }}
      />
    )
  })
)
TeX.displayName = "TeX"

export default TeX
```

Listing 8: CTO KaTeX component file `KaTeX.jsx`, continued

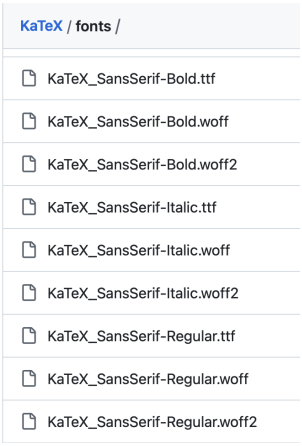

Figure 8: KaTeX sans serif fonts: `github.com/KaTeX/KaTeX/tree/main/src/fonts`

6 Live demo

One of our smaller web applications that demonstrates all use cases of KaTeX is our "RSA step by step", available at `www.cryptool.org/en/cto/rsa-step-by-step`. In the talk, we show how users interact with the application in the browser and present the corresponding source code, highlighting where KaTeX is used within the codebase.

7 Fonts

On the KaTeX website there is also documentation on how to generate new font metrics: `github.com/KaTeX/KaTeX/blob/main/src/metrics/README.md` The metrics of the KaTeX fonts are available as a JSON file: `github.com/KaTeX/KaTeX/blob/main/src/fontMetricsData.js`. Those metrics are used by KaTeX to compute the distances and positions for the CSS rules.

At the moment we use the default math fonts, but we would like to switch to sans serif math. The screenshot in fig. 8 shows what's available at the moment.

Any help on this would be very much appreciated.

References

[1] C.H. Bennett, G. Brassard. Quantum cryptography: Public key distribution and coin tossing. In *Proceedings of the International Conference on Computers, Systems and Signal Processing*, pp. 175–179, Bangalore, India, 1984. `doi.org/10.48550/arXiv.2003.06557`

[2] Chakra-UI. Chakra Systems Inc. `chakra-ui.com`

[3] B. Esslinger. *Das CrypTool-Buch: Kryptografie lernen und anwenden mit CrypTool und SageMath*. Lehmanns Media, Berlin, 2024. `www.cryptool.org/de/ctbook`

[4] B. Esslinger. *Learning and Experiencing Cryptography with CrypTool and SageMath*. Artech House, Norwood, MA, 2024. `us.artechhouse.com/Learning-and-Experiencing-Cryptography-with-CrypTool-and-SageMath-P2378.aspx`

[5] KaTeX. KaTeX contributors (`github.com/KaTeX/KaTeX/graphs/contributors`). `katex.org`

[6] MathJax (`www.mathjax.org/#about`). `www.mathjax.org`

[7] MathML. W3C (`w3c.github.io/mathml-core`). `developer.mozilla.org/en-US/docs/Web/MathML`

[8] National Institute of Standards and Technology (NIST). *Module-Lattice-Based Key-Encapsulation Mechanism Standard*. U.S. Department of Commerce. `nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.203.pdf`

[9] `Next.js`. Vercel Inc. (`vercel.com`). `nextjs.org`

[10] React. Meta Platforms Inc. (`opensource.fb.com/legal/terms`). `react.dev`

[11] webpack. OpenJS Foundation. `webpack.js.org`

◇ Doris Behrendt
Universität der Bundeswehr München
Werner-Heisenberg-Weg 39
Neubiberg, 85579
Germany
`doris.behrendt (at) unibw dot de`
`www.unibw.de/datcom/mitarbeiter/dr-rer-nat-doris-behrendt`

◇ Bernd Busse
Universität der Bundeswehr München
Werner-Heisenberg-Weg 39
Neubiberg, 85579
Germany
`bernd.busse (at) unibw dot de`
`www.unibw.de/datcom/mitarbeiter/bernd-busse-m-sc`