How to write a thesis

January 29, 2021

Abstract

This document is intended to help guide you (a student) through the process of working on your Bachelor's or Master's thesis. It answers frequently asked questions (and questions that *should* be frequently asked). Of course, your thesis advisor will also answer any questions you have (but sometimes it's just nice to have a resource to look into privately).

Contents

1 So. . . you're generally interested in writing a thesis. What do you do?

First of all: great! The first step will be to find a topic for you.

We may have some open topics on our website. Note that this list represents just a *subset* of topics we'd be interested in supervising. More often than not, we come up with new topics on a per-student basis. To help us with this, contact us (e.g., one of the research staff via email) and tell us:

- Do you enjoy theoretical work or do you want to implement something? Or both?
- Do you have any helpful prior knowledge? (e.g., lectures, seminars, theses, anything you may have read in your free time)
- Do you have an idea of what topic/direction you're interested in? (Usually, you won't. Don't worry, we'll have suggestions)

The process of finding a topic varies a lot from student to student. See Section 3 for more information.

2 How do you have to imagine writing a thesis?

This is what's going to happen:

- We, together with you, find a topic to write about. (Section 3.1)
- You'll write a proposal, defining the goals of the thesis. (Section 3.2)
- You'll officially register your thesis. (Section 4)
- You'll write a thesis (Section 5) and do a starter talk a few weeks in (Section 6).
- You'll submit the thesis (Section 7) and do a final presentation on your results (Section 8).

Finding a topic and writing the proposal should take roughly 1 month (as a guideline). After registering the thesis, you'll have 5 months to write it. Work on a Bachelor's thesis is supposed to be part-time (i.e. roughly 20 hours a week), work an a Master's thesis is supposed to be full-time (i.e. roughly 40 hours a week). The final presentation happens shortly (at most four weeks) after submitting your thesis. You'll have a thesis advisor (usually one of the research staff) who will help you through this process with advice and feedback.

3 Finding/defining the topic: The proposal phase

The proposal phase is the time before officially starting with your thesis. The goal of this phase is to find a topic and to define goals for the thesis.

3.1 Finding a topic

Finding a topic can happen very quickly or it may take several suggestions and reading into several topics. The goal is to find something that is interesting for you and useful for us. This process looks something like this:

1. Take a rough look at our suggested topic (either you got this from talking to us or you found it on the website).
2. Talk to the advisor for the topic. Clear up first questions and get your advisor's perspective of what the topic should be.
3. Take a few days looking into it more. If you're mostly convinced you like the idea of this topic, announce that to your advisor. (You can still change your mind later, but ideally you have a rough idea now)
4. After announcing your interest, you and your advisor start meeting regularly (usually weekly). Your goal is to understand the topic better and get an idea of what you want to write into the proposal later. This is your chance to get a better feeling of what the topic will be like. You want to get a good feeling of what you're buying into. Talk to your advisor (let us know of ways we can help you) and come up with a good plan for the proposal together.

3.2 Writing a proposal

After you've decided on a concrete topic and have understood its basics, you can write your *proposal*. We have a template for proposals on our website. The template is written in LaTeX (cf. Section A.1 for technical hints on LaTeX). It already contains all the required sections and general layout. The proposal should be roughly 3-4 pages long. Talk to your advisor and ask for feedback.

Why do you write this? The proposal is formally required. It's also useful. It serves two functions: (1) make sure that your advisor and you have the same understanding of the scope and goals of the thesis, and (2) in the end, we'll judge your thesis on whether or not you've achieved the goals outlined in the proposal.

What language do you write this in? In the language you also want to write the thesis in. If you do not feel comfortable with English, you can write the thesis in German. This does not incur any formal disadvantage for Bachelor students. For Master's theses, note that writing in German means you won't get the "*Englischsprachiger* Masterstudiengang Informatik" tag on your final certificate.

What language do we recommend? It's hard to say generally. We'd argue that writing in English has several advantages:

- Basically all research-level papers are written in English. Newer terminology usually doesn't have a German equivalent, so you'd be forced to write Denglish.
- Writing in German severely limits the set of people who can read your thesis.
- English writing is a very useful skill, especially for computer scientists.

However, we also stress:

- these advantages completely evaporate when all your English content is obfuscated by awkward language.

So what you need to do is take a critical look at your English skills. You should be able to communicate effectively in written English. This means that you should generally be able to avoid obvious grammar mistakes and to avoid jarringly unnatural language (don't translate a German sentence word-by-word; don't use words that don't mean what you think they mean). It does *not* mean that you must know lots of fancy words or use flowery language. Indeed, for technical topics, you should avoid writing too flowerily (cf. Section 5.4.3).

You can use the proposal as a testbed for your English skills. Consider two aspects: (1) how difficult do you find writing in English? Do you feel like it's seriously slowing you down? (2) Is your writing comprehensible? Ask your advisor for honest feedback on your language. Make an informed decision on this basis (you may have to rewrite your proposal in German, so ideally you decide well before finishing the whole thing).

For additional information on use of English, see Section 5.5.3.

4 Registering the thesis

You can register the thesis through PAUL (manual). During this process, you may be asked to specify a secondary examiner. Talk to your thesis advisor for recommendations and ask the prospective secondary examiner via email if they are okay with it.

After registration in PAUL, the University will check that you have all the prerequisites to write the thesis (hint for Bachelor students: make sure that mentoring has been entered into PAUL). When this is done, you'll get a registration form through PAUL to print out.

Have the registration form and two copies of your proposal signed by the first examiner (that's usually Prof. Blömer). Submit the registration form to the Prüfsek (as noted on the form) and keep one of the proposal copies for yourself (the other one is kept by us).

5 Working on the thesis

In this section, we'll list some of the typical situations that may occur when writing the thesis and our corresponding advice. Of course, if you encounter problems, you should also ask your thesis advisor for advice.

5.1 Who is your target audience?

Whenever writing any sort of text, you should definitely ask yourself: who will read this? From experience, readers are:

- Bachelor or Master students also writing their thesis
  - "I need to understand the proof because I'm building an extension for this"
  - "Hey, this thesis also deals with attribute-based encryption. I wonder how they defined security."
- Research staff (PhD students, Professors)
  - "What turned out to be the best coordinate system for elliptic curves again?"
  - "How did this algorithm work again?"
  - "Can I give this thesis to my new student as a starting point?"

Keep this in mind when writing the thesis. In particular:

- Other students should be able to understand your work.
- Ideally, quick lookups are somewhat easy (have clear structure, name theorems/definitions, highlight important insights and changes to original work, . . .).

5.2 Setting up your writing environment

A good setup of your writing environment makes the whole thesis process easier. It pays to invest a little bit of time into it.

5.2.1 Setting up the document

You can use our thesis template as a starting point. See Section A.1 for hints on what LaTeX software we tend to use.

5.2.2 Formatting

This is pretty boring, but doing this correctly from the start will spare you some pain later.

- `$gmsk$` = $gmsk$ is different from `$\mathit{gmsk}$` = gmsk. The former denotes the product $g \cdot m \cdot s \cdot k$, the latter is a variable called "gmsk" (e.g., "group manager secret key"). Use macros for all multi-letter variables: `\newcommand{\gmsk}{\mathit{gmsk}}`. Then you only have to type `\gmsk` and you'll get the right formatting. It also makes your code much more readable.
- There is *some* consensus on how to format identifiers, but only the first two are strict rules.
  - Variables: `\mathit{...}` (e.g., gmsk)

  - Constants: `\mathrm{...}` (e.g., NP)
  - Algorithms: `\mathrm{...}` (e.g., Sign) or `\mathsf{...}` (e.g., Sign)
  - Adversaries, simulators, etc.: usually `\mathcal{...}` (e.g., $\mathcal{A}$)
  - Some people prefer to handle keys and similar artifacts differently from other variables, e.g., using `\mathsf{sk}` (sk) instead of `\mathit{sk}` (sk).
  - Know the difference: L_{gmsk} is a variable that depends on the value of $gmsk$ (one can imagine a sequence of such L_{gmsk}, one for each possible $gmsk$). For example, L_{gmsk} could be the length of a particular $gmsk$.
    In contrast, L_{gmsk} is a variable that does not depend on a particular $gmsk$, it just has gmsk as part of its name. For example, L_{gmsk} may be generated without knowledge of $gmsk$, but occurs in equations like $K = L_{\mathrm{gmsk}} \cdot gmsk + L_{\mathrm{pk}} \cdot pk$ (and $L_{\dots}$ is just generally used as a name for the coefficients in such equations).

  As mentioned before, you should definitely create macros for these.
  You can also use `\mathcal{...}` ($\mathcal{G}$) `\mathfrak{...}` ($\mathfrak{G}$), `\mathbb{...}` ($\mathbb{G}$) for basically anything to avoid name clashes, just have it make sense somehow. For example, you could decide to consistently name protocol roles $\mathfrak{P}, \mathfrak{V}$.

- LaTeX is infamous for ignoring the page margins. It doesn't handle math lists like $\mathsf{Enc}(pk, m_0, m_1, \dots, m_n)$ very well. There are two ways to fix this (sometimes you need a combination).
  - Use `\allowbreak` to tell LaTeX that it's okay to break at a certain position (for example, `$\Enc(\pk,\allowbreak m_0,\allowbreak m_1,\dots,m_n)$`) would fix the issue above. It doesn't *necessarily* insert a line break, you're just giving it the option.
  - Rewrite the sentence so that the formula moves into a position where no line-break is necessary. Sometimes it's enough to add or remove a "it holds that"-like phrase or to split a sentence into two.

- Generally: Other than being proactive with macros, don't waste too much time on formatting. Use downtime to do this, not your most productive time when you're busy producing actual content. You can send your advisor draft versions where the formatting isn't perfect.

5.2.3 Versioning

We strongly suggest you use a versioning system to store the thesis (you can easily create a git repository at git.cs.upb.de). This serves as a backup mechanism and you can feel at ease deleting stuff from the thesis knowing old versions can always be retrieved. See Section A.2 for the git tools we tend to use.

5.2.4 Organizing your thoughts

You should spend a little bit of time thinking about how to organize your work. While working on the thesis, you may have to juggle:

- Feedback from your advisor (usually in annotated pdf form, in emails, and in meetings)
- Agreements with your advisor ("a proof sketch suffices here")
- Questions you want to ask your advisor in the next meeting (+ answers)

- Work in progress proof sketches or code architecture plans
- A plan: what content is still missing?

For solutions, consider

- `todonotes` directly within your thesis document
- directories for files in your git repository
- a dedicated physical or virtual notebook to keep your notes/sketches in
- annotated pdfs (e.g., tick marks next to advisor feedback)
- a todo list

You don't need a *perfect* plan (it's something you can improve over time). Organization is also something you can waste lots of time optimizing. Just . . . think about it a little bit. Don't be the person that somehow misses half of our feedback, is stuck but never has any questions, and notices a week before submission that a chapter is missing ;-).

5.3 How do you interact with your advisor?

Use your advisor's help! We're highlighting this because it's important. Writing a thesis is very clearly a learning process and consulting your advisor is the best way to learn how to do it. **Ask for feedback on what you've written** (see Section 5.3.2). **Ask for help when you're stuck** (see Section 5.3.1).

That's not to say that the advisor-student relationship is completely one-sided. Your advisor has their own goals with the thesis. Usually, we're interested in the topic and want to learn more about it.

It's not your job to satisfy your advisor's curiosity. Generally, you should focus on your thesis. However, on a human level, we will appreciate it if we learn something from the process, too. So if you run across interesting problems during your work, you have an idea how to improve the result, or if you found some new insights or an interesting paper: discuss it with us!

The other thing that makes us happy is seeing you do well and improving because of our feedback. Having a positive impact always feels good.

5.3.1 Common pitfalls

These are the most common mistakes when it comes to handling your advisor:

Being unprepared for meetings. Consider making a list of things you want to discuss. If you want help: can you explain your problem somewhat adequately?

Not allowing your advisor to prepare. We don't magically have all the answers at a moment's notice. If we did, we wouldn't ask you to work on the topic :-). Send us any hard questions *in advance* of the meeting. This gives us time to think about it. You don't have to do this with every single question you have (especially the ones you know will be easy to answer). But if you do, your chances of getting a good answer improve. When in doubt, warn us in advance.

Not providing context. Be aware that your advisor is juggling lots of things besides your thesis. So we may not immediately know which "Theorem 5.2" you're talking about. Generally: Give us context for questions.

    - We're generally happy to give you fast and good answers, but that's not easy if you first send us on a wild goose chase trying to find out what you're talking about. The better you can explain your question, the better the answer will be! The more conveniently your "quick question" is phrased, the more likely you'll actually get a quick answer.
    - If you're stuck dealing with some devil in the detail, start by explaining a rough outline of the proof/construction/code you're talking about (and in what cases the problem arises). You may have been spending a lot of time thinking about this thing, but we usually need at least a quick reminder.
    - In emails, if you really want to make things easy for us: include your current pdf and point specifically to what you're talking about (page number). Hyperlink any external papers you have questions about. Screenshots of the thing that's unclear are also *greatly* appreciated.

Just blindly nodding at explanations. There is absolutely no need to impress your advisor. Ask for clarification, or even just say "I don't understand". It's fine. Give us a chance to explain better.

Forgetting what we've been talking about. Just take some sort of notes. Whiteboard photos also sometimes do the trick. It's also generally good style for meetings to summarize complex discussions briefly (which is what you're doing taking notes). It makes it easy to check that you and your advisor are on the same page.

5.3.2 Getting feedback

Feedback is the most helpful thing we can provide for you. It's how you learn to improve your thesis. However, it's important to interpret feedback correctly.

- We don't expect documents (or parts of documents) that we haven't read before to be perfect. This stuff is difficult and it's *completely unreasonable* to expect everything to be without any flaws on the first try.
    - Even after having been working on this stuff for two years, we still consistently make subtle and not-so-subtle errors in first and sometimes second versions.
    - You shouldn't be afraid that negative feedback negatively affects your grade. For grading, we don't care about the first version, we care about the final version. Just improve, it's fine!
- Review remarks are to be interpreted as "when I read this, I thought: [remark]". In particular:
    - We may be wrong.
    - Even if we're right, we don't usually claim that you absolutely *have to* change something. There may be good reasons to keep it as it is (that we may not be aware of). It's okay to disagree!
        - Discuss disagreements with your advisor.
    - Remarks are not accusations. And they don't pose any threat to grades or our perception of you.

  - We definitely, definitely never mean it as ”why are you so stupid?”. We know that some errors just look shockingly obvious in hindsight. We make those too. You don't have to feel bad about it.

- Standard review policy is to just give all remarks we can think of.
  - This includes things that are obvious and you probably are aware of yourself (e.g., “the order of sections is strange right now” or “a large bit of the proof is missing here”).
  - This includes remarks on sections you already told us aren't done yet or you still need to proofread yourself.
  - That's the way it's done in academia. For everyone. It's safer. It would be stupid to filter out potentially helpful remarks just to avoid offending the writer. Just ignore everything you already knew; it's fine.

So relax and stop apologizing for whatever remarks we give you and whatever obvious errors we find ;-). We don't mean to accuse you. We're just trying to be helpful.

5.4 Questions about “scientific” writing

5.4.1 How do you avoid plagiarism?

If you follow the news, plagiarism often seems like someone copy/pasted something without attribution. In computer science academic reality, it's actually more nuanced than that and doesn't usually have to do much with copy/pasting things.

Merriam-Webster[1] defines plagiarizing as follows: “to steal and pass off (the ideas or words of another) as one's own: use (another's production) without crediting the source”. Note that the quote explicitly mentions *ideas*.

Basically, plagiarism happens if and only if your text makes it look like something was your idea even though it wasn't. This can be avoided easily.

Reproducing or building on someone else's idea. Just explain in the text what parts you contribute and what was someone else's idea. This can be as easy as “In this section, we explain the construction of [1]” (where the reader would assume that the explaination is yours and the idea and construction is [1]'s) or “our proof is based on the sketch in the original paper [1]” (where the reader would assume that [1] provides a proof sketch and your proof is more detailed). Just state clearly who contributed what.

Using someone else's things verbatim. There is no need to paraphrase everything in your thesis. You can, for example, use someone else's definition or theorem verbatim (i.e. without changing anything). Just briefly explain this clearly in text (e.g., “the following definition is taken [verbatim/with minor notation changes] from [1]”).

For other parts, this is more complicated. Let's say some paper states “It seems like building a signature scheme from the RSA assumption is impossible”. Let's say you include this sentence verbatim into your thesis like “It seems like building a signature scheme from the RSA assumption is impossible [1]”. Then a reader would naturally assume that these are your *words* (and that just the *idea* is from [1]). You can use quotes to denote that these are not your words, then you've done your disclaiming duty and readers again wouldn't assume these are your words. In

[1] https://www.merriam-webster.com/dictionary/plagiarizing

most cases, however, the best way to handle this situation is not to use the direct quote at all (and also not to paraphrase it forcibly). Your thesis is not a sequence of things you found in other papers. Instead, it has its own flow of thoughts and its own style of presenting them. So usually, you'd write something that naturally probably wouldn't use the quote. For example "We can prove security of this signature scheme only in the random oracle model. Note that it would be surprising if we could prove it secure under the plain RSA assumption [1]". See Section 5.7 for more hints about doing your own thing without parroting other papers.

Trivial facts. Let's say you're using (mathematical) rings in your thesis. Nobody will ever assume that you've come up with that notion yourself and everyone will be already familiar with them. Hence there is no need to cite for this. The same goes for any common knowledge that may be taught in the very basic lectures at university: EUF-CMA security for signatures; CPA/CCA security for encryption schemes; ElGamal encryption; etc. These things are in your thesis mainly to establish notation and to pin down the specific flavor of definition you're using.

More information If you do want more information, consider the CS department's document on this.

5.4.2 How do you find papers and how do you cite them?

This consists of many small knowledge bits:

- Use BibTex's `\cite{x}` command. For this to work, "x" needs to be an entry in your `references.bib` file (see thesis template for an example).
- Don't write `references.bib` entries yourself. Use dblp to export BibTex entries (We recommend the "condensed" option like this).
- Alternatively, if you're cool with including more than 25 MB of bib files into your project (some editors don't handle that gracefully and compile times increase), you can use cryptobib, which is a giant bib file containing (almost) all cryptography papers.
- For literature research (to find new stuff), use scholar.google.com. It's a good idea to do this within the university network (via eduroam, or VPN if you're not physically there), which grants you free access to most papers.
    - For Springer, you may have have to clear cookies or use anonymous browsing if you've tried to access the paper before connecting to VPN.
- Papers in theoretical computer science often come in multiple versions:
    - Conference version: *usually* the first time something is published is on a conference. Conference versions have a page limit and are often incomplete because of that.
    - ePrint/arXiv versions: authors often upload a version of their paper to ePrint or arXiv. These don't have any page limit, hence authors often publish full versions that contain more details than conference versions. You may also find papers on these sites that have not yet been accepted at a conference. ePrint/arXiv papers are freely accessible to anyone but the version posted there has not necessarily been peer-reviewed.
    - Journal version (usually something published in the "Journal of ..."): sometimes, authors write a journal paper. This is comparable to a conference publication, but less restrictive on page limits and with more exhaustive peer review. The process of

    publishing in a journal is usually much slower than for conference publications. Hence authors often first publish a conference version and then *maybe* an extended journal version afterwards.

  Which version do you cite? If possible, cite the journal version. Otherwise, if possible, cite the conference version. Otherwise, cite the eprint/arXiv versions. If you need to reference something that only exists in a specific version, obviously cite that one.

- Usually, there is no need to specify page numbers for your citations.

5.4.3 What writing style is appropriate?

People usually think that scientific writing uses especially fancy language and complex sentence structure. For theoretical computer science, we aspire to the opposite: ideally, text is *as easily readable as possible*. In particular, this ideal informs the following style guides:

- Use short sentences as much as possible.

  > Signature schemes, consisting of three algorithms $\mathrm{Setup}, \mathrm{Sign}, \mathrm{Verify}$, are very basic building blocks, which, although they rule out any information-theoretical security notions (given that signature schemes cannot be constructed information-theoretically but must always rely on computational assumptions), we will use extensively in this thesis in order to authenticate values our computationally bounded adversary must not be able to change.

  > Signature schemes consist of three algorithms: Setup, Sign, and Verify. Such schemes are very basic building blocks: they prevent adversaries from changing authenticated values. Unfortunately, signature schemes cannot be constructed information-theoretically. Instead, signature schemes rely on computational assumptions. We will use signature schemes extensively in this thesis. Because our constructions will inherit the signature schemes' computational assumptions, they are only computationally secure. For this reason, we do not present information-theoretical security notions.

- There is no need to use fancy words. Just write normal English, you don't need to run every word through Thesaurus.

- Don't use synonyms to make your sentences feel less repetitive. In formal contexts, this is very confusing for the reader.

  > The user sends x to the server. The host replies with y. Then the client transmits z.

  Here, the reader is left to wonder whether or not "user"/"client" and "server"/"host" are different roles and whether there is a difference between "transmitting" and "sending" something. Better:

  > The user sends x to the server. Then the server sends y to the user. Afterwards, the user sends z.

- It's okay to write in the first person. Don't do forced workarounds like

> It can be deduced that $\mathbf{P} \neq \mathbf{NP}$.

Use “we” to make sentences less difficult to read. “We” can either be used for “the author and the reader”, e.g.,

> We can deduce that $\mathbf{P} \neq \mathbf{NP}$.

or for “the author(s)”, e.g.,

> In Section 3, we present our results on **NP**.

instead of

> In Section 3, the results of this thesis regarding **NP** are presented.

Even for single-author papers (such as your thesis), people don't use “I”. This is probably because single-author papers are very rare for conferences and journals that using “I” really stands out within the genre. Hence typical readers, who are used to seeing “we”, may feel like you're overly emphasizing your single authorship. This comes across as arrogant.

- Don't use contractions.

  > Don't use contractions.

  > Do not use contractions.

  We think that's just mostly useless tradition?! People say it looks unprofessional. We haven't found any *good* reasons to avoid contractions other than peer pressure.

- Default to present tense. But use whatever tense you need.

- Use bullet points if they're nicer to read than inline enumerations. They're also a good tool for emphasis (e.g., “these are the three most important questions we are addressing in this thesis: . . . ”).

5.5 How to write text

The challenge in writing text is linearizing and explaining your complex thoughts.

5.5.1 General approach: Planning your text

It's crucial that you plan your text: this is what distinguishes hard-to-understand ramblings from well-structured and helpful texts. Everyone has their own techniques for writing, but here's what works for us:

- Start with a rough story draft of what you want so say. We usually write a list of simplistic statements like “[X exists]. [It's based on Y]. [Y is slow]. [Z is more efficient than Y]. [Want to replace Y with Z]”. You should be able to immediately glean the structure from this and understand the relation between the statements.

- Depending on the situation, expand each simplistic statement to (1) a few sentences or to (2) a paragraph each. Fill in what you really meant with your statements, e.g.,
  - "[X exist]" becomes "X is a library for detecting subspace microfractures."
  - "[Y is slow]" becomes "Y uses axial force flux detection, whose runtime is exponential in the number of hyperwave particles. Because of this, it is unsuited for use on chroniton wormholes, where the number of hyperwave particles is rather high."
- Iterate! Don't be afraid to just throw away parts you've already written and redo them. Don't be afraid to change the story and rewrite parts to fit.
  - If you get feedback that indicates your structure is not strong, don't try to salvage it by just adding a word here or there. Re-think the paragraph(s); maybe restructure.
  - Make sure that you're hyper-clear about what's going on. When in doubt, just write a sentence like "We are now describing x."

5.5.2 Some tried and tested text-structures

How do you come up with a good story/structure? Here are a few thoughts:

Problem-solution-oriented. General problem → naive solution → problem with that solution → idea for solving that problem → discussion of that approach – does it work? What's the main challenge?
A simplified example to describe these problems is often useful.

> **A central problem** in cryptocurrencies is transferring coins from some user A to another user B. At first glance, **the solution seems simple**: We use a digital signature scheme. A and B are identified by their public keys. To transfer coins, A simply signs "I now give n coins to B" and gives this signature to B. B can then use this signature as a certificate that he owns those n coins that previously belonged to A.
>
> **The problem** with this simple solution is double-spending. Even after A has sent his n coins to B, A can sign a second message "I now give n coins to C" and give that to another party C. In this scenario, B and C are each convinced that they have received n coins from A, who only had n coins to spend.
>
> **To fix this**, all transaction signatures are recorded on a public ledger. Signatures on this ledger are well-ordered. If there are two signatures trying to spend the same coins, the second one will be deemed invalid. In our example above, B would record the signature on the public ledger. When C receives his signature, he would consult the public ledger to notice that A does not own the coins anymore and that his transaction would be considered invalid.
>
> This public ledger can be realized through a centralized trusted database server that records all transactions and allows the public to query them. Obviously, such a trusted party is **difficult to realize** in the real world. Note that this trusted party may have a financial incentive to lie about transaction data (e.g., if A controlled the database, he would be able to double-spend).
>
> The **current best practice** is to use a *distributed* public ledger, where security does not rely on trust in a single party but on a large number of participants. **The main challenge** for a distributed ledger is establishing consensus between the participants,

which we discuss in the following.

Construction-oriented. What do we already have/know? → What's the ultimate goal? → what ingredients do we need? → how do you get those ingredients? → how do those ingredients fit together?

5.5.3 Writing English text

Writing English text is not always easy for non-native speakers. We try to list some common pitfalls here.

- Avoid thinking of German sentences and then translating them. This often results in unnatural English sentences. Ideally, your internal monologue (your "thoughts") should already be English while writing.

- If you don't know a word, but know its German equivalent, don't just take the first word suggested by a German-English dictionary. You'll usually get *a bunch* of related words. Use the one that *fits* what you want to express (and be prepared to dismiss all of them!). Use an English-English dictionary to check the exact meaning of a candidate word.
  As a simplified example, you want to say "Das Signaturschema wird als sicher angesehen". If you look up the German word "sicher", the first English word for it is *certain.* This doesn't mean you would ever write

  > The signature scheme is considered certain.

  When looking up "certain", you'll find it means "able to be firmly relied on to happen or be the case", hopefully noticing the problem with your word choice (signature schemes don't "happen").
  This is a pretty obvious example, but if your're insecure with English, errors of this type happen very often.

- Don't trust machine translations (especially not for complex research topics). The same remarks as for dictionaries apply here.

- American vs. British English – it doesn't matter. Pick one and stick to it. (Completely unscientifically, our impression is that the most common language for research papers is some sort of internationalized American English)

- Commas are . . . difficult. There is no single clear-cut ruleset. Usually, it's fewer commas than you'd think as a German.

  - Because technical writing uses a lot of linking phrases to start sentences (e.g., "Because of this, the error probability is at most $1/\ell$." or "However, this equation only holds for $x < 7$."), it may be a good idea to look up what an *introductory word/phrase/clause* is. They require commas where you wouldn't put any in German.

- Capitalization of titles and section names: The title of your thesis should use title case. For example: "Secure Computation for Interesting Things without Prover-Specific Random Oracles". There are a bunch of (differing) guidelines for this. Google "title case" and just stick to one consistent format, for example http://titlecase.com. For chapter/section names, you can, but don't have to, use title case.

5.6 Writing proofs

The first rule of writing proofs is to only write true statements. Actually, this is the first rule for most texts in your thesis. Indeed, writing proofs is not unlike writing any other text. Be aware that the reader of the proof is also human. If the proof is difficult, offer some help to understand it.

- Give a good proof outline that summarizes the main arguments and structure. Establish a good intuition of what's going to happen and why – on a high level – the proven statement should be true.
- It is very important that you use your definitions for the proof. Since you have to use what you have defined to write a comprehensible proof, Two examples: if you use a previously defined algorithm $\mathsf{Alg}(x, y, z)$ with inputs x, y, z it is important that you use it consistently (e.g. order of inputs), and if you do a reduction using a security game you have to stick to the steps (and oracles) of this game.
  - For this you should open your thesis to see your own definitions next to your document that you are editing for the proof.
  - There are also tools that let you snap your definitions and the snap stays on top of all open windows, e.g. http://snappy-app.com (macOS), https://www.snipaste.com (Windows, macOS).
- Throughout the proofs, refer to pre-established intuitions. For example:

  > $\mathcal{A}$ chooses a random index $i \leftarrow \{1, \dots, q\}$ (hoping that $\mathcal{B}$ will output a forgery for the ith message).

Ideally, a proof not only demonstrates that some statement is true but also offers a deeper insight into *why* it is true.

Here is the standard way to write proofs:

- Sketch the proof idea for yourself (e.g., on paper). How do all the pieces fit together?
- Write down the basic formal definitions into your thesis document. Make sure the theorem you're proving is perfectly well-defined. Don't skip this step.
- Write the proof into your thesis document. This should include explanation.
- Iterate! Your first proof can usually be improved (simplified, better structure, better intuition, etc.)

Other hints:

- Roughly honor the "at most one idea per sentence" guideline.
- For every statement that only comes into play later in the proof: you should give readers a hint ("... because [short repetition of the statement] as seen above") when you use earlier statements. (And – if possible – you may even give a hint for early statements what their function *will be*).

5.7 You feel like you're just paraphrasing the original paper

You're doing it wrong then :). Read the original paper first. Understand it. Then put it aside and create a structure: how would *you* explain the paper? Forget how the original paper did it. Maybe you can do better (e.g., because you don't have such strict page restrictions). Maybe you have a different point of view on the topic (e.g., knowing something the original paper's authors didn't).

After you've come up with *your* structure, fill it with content. Don't consult the paper for writing or structure or explanation. Only consult when you actually find you don't yet understand their ideas. Then put it away again.

It's okay to use some parts of the original paper – don't feel like you *have to* forcibly change definitions or algorithm descriptions just so that they're different. If your way of presenting it coincides with the paper's, that's okay. There are plenty of ways to stand out that are more meaningful than changing multiplicate to additive group notation for no reason ;).

5.7.1 Setting your thesis apart from the original paper

If you're looking for more ways to set your thesis apart from the original paper on a content level, consider the following ideas.

Better explanations. The original paper is likely somewhat minimal with regards to explanation. You can explain a lot more and make it easier to understand for your readers.

Formal proofs. Full proofs are often omitted. You can provide them (and make sure that the original paper is correct)

Simplify. Come up with a simplified version of the original paper's contribution. The simplified version should be meaningful but easier to understand than the full version. It may even give insights into the full construction.

Examples and figures. Give examples for some of the more complicated stuff in the original paper. What's a useful application of Theorem 2? How does this complicated algorithm work for simple inputs? Add figures to illustrate how something works or how something is structured.

Related work. Explain and compare related work (especially work that appeared after the original paper has been published)

5.8 It's hard to motivate yourself to work

This is very common. Unfortunately, there is not really a silver bullet for it. From experience, there seem to be two very common productivity-killers:

Giant amorphous tasks. You're standing in front of the huge task of *writing the thesis*. That's an intimidating task. It's also very abstract; it's not like you can sit down and just start writing the thesis from start to finish. As a consequence, it's hard to motivate yourself to work on it – maybe you don't even know how to start.

Make a plan. Try to have a lot of *actionable* items (i.e. things you can definitely just *do* and make measurable progress; like writing down some basic definition). If you have a hard time coming up with a concrete plan, ask your advisor for inspiration. Plan deadlines of such tasks for yourself.

Being stuck. Some tasks, like coming up with a formal proof, cannot really be subdivided into smaller actionable items. You may not quite know how to begin to tackle these. Or you may be trying to come up with ideas, but are stuck somehow.

Discuss with your thesis advisor. Try to articulate your problem in detail: what are you trying? Why doesn't it work? What components would you expect to play what role on an abstract (intuitive) level? Often, just explaining a problem already leads to new solutions. Your advisor probably doesn't know the perfect solution, but may help by asking questions to narrow down the problem, offer a different perspective, or contribute some new ideas. It should at least get you unstuck so that you can continue working on a proper solution.

5.9 You're done with the theorems and proofs, now you just need to write the "filler text"

Your thesis should not be just a sequence of definitions and theorems. Instead, you should guide the reader through what's happening and help him understand it more clearly. For this, you put some text between your sections, definitions, theorems, etc. This is a *huge* chance to improve the quality of your thesis, so don't waste it by writing boilerplate filler text like:

```
\section{X}
In this section, we introduce X.
\begin{definition}[X]
...
```

Instead, use these texts to deepen the reader's understanding of the matter. Here are some suggestions what you could write, but this list is neither exhaustive nor generically applicable in all contexts. It is merely meant for inspiration.

- For sections:
  - what's going to happen? How does it relate to what has happened/will happen in other sections?
  - why is the topic of this section challenging/interesting?
  - what is an example for the thing this section will deal with?
  - if the section only gives an overview – where should the reader look for more detail (literature)?
- For definitions:
  - a simple example of something fulfilling the definition
  - a short proof/argument related to the definition:
    - it implies something useful or interesting
    - it's equivalent to some alternative characterization
  - why is that a useful definition?
  - why do you do this non-obvious thing in step 4? (this and generally any design decisions that may be interesting)
  - how does this definition relate to some other definitions you presented? Does one imply the other? (or is it noteworthy that they don't imply one another? Give a counterexample)

  - did you choose to alter the definition from some original paper? Where and why?
  - if there are alternative definitions, why did you choose this one? Is it easier to understand/equivalent/stronger/weaker/...

- For theorems:
  - why is the statement not obvious? Is there a simple example where the statement is not clear a priori?
  - what is an example where the theorem could be applied? Why is it helpful?
  - what is a special case of the theorem that may be easier to understand?
  - what would happen if one drops some non-obvious requirement from the premise? Would the statement still be true?
  - for implications: is the other direction also true?

- For proofs:
  - what's the rough proof outline? (only as much as can be reasonably explained in a few sentences)
  - what's going to be "the hard part"? Why is the proof not obvious?
  - any special proof techniques you'll apply?

Of course, don't write anything that's too obvious or too inconsequential. Whatever you write should be noteworthy or helpful for understanding. Good candidates for textual discussions are discussions that you had with your thesis advisor and examples/special cases you developed in order to better understand the definition/theorem yourself.

Decide wisely what you present to the reader *before* or *after* formal descriptions. Don't try to present anything other than a *very* rough intuition before presenting something formally (see Section 5.10).

5.10 You get feedback that your writing is fuzzy or hard to understand

The most important point here is: If you get feedback like "I don't understand this word here", it's not always a good idea to just change the word and be done with it. Ask yourself *why* that wasn't comprehensible. Did you not explain that word before? More generally: Rethink your story and make sure the readers are ready for your thought when they get to it.

In the following, we're listing a bunch of typical pitfalls that lead to "this is not comprehensible" feedback.

Don't try to summarize too much information into a single sentence Be concise. Don't feel like your explanation is too short. It needs to be *complete* and *comprehensible*, not *long*. Do not be unnecessarily concise. You do not have a page limit. For example, you may not want to explain exactly what vector spaces are (it's not that important), so you write "Vector spaces are sets of elements with certain operations on them and certain laws that must hold.". This helps exactly nobody, so either require the reader to already know vector spaces or explain them properly. If what you're explaining is related work (and not part of a standard computer science study course), spend some more sentences on them. You can of course generalize and explain, but don't try to summarize details.

Invite the reader into your thoughts: Add extra context Let's say you wrote "We essentially use the school method of multiplication, but after each step, we apply the modulo operation". That may be a great intuition. Everyone knows what the school method of multiplication is. But ... what is a "step" here? What do you apply the modulo operation to? The reader, who doesn't have exactly the same thought model of school method multiplication may simply dismiss this intuition because it's too fuzzy.

Invite the reader into your thoughts. Explain the school method of multiplication and give the steps names. Or just write it down in pseudocode notation and number the steps. It takes a few lines of text/code and you may feel silly explaining such a trivial concept in your thesis, but it's a *great* way to establish your way of thinking and notation. After explaining this, you can write "We essentially use the school method as above, but after executing line 5, we additionally reduce b modulo n". Now everyone knows *exactly* what you're talking about.

You can also go through multiple steps of transforming some trivial version step-by-step into a proper solution, see Section 5.5.2.

Do not try to explain something you didn't present formally yet Do not try to explain a definition *before* the formal definition itself (same for theorems, lemmas, proofs, etc.). It sounds like that would help the reader prepare, but it's a trap – you'll implicitly write down the definition twice: once informally for that explanation (high chance that nobody can understand that) and then again formally (which is the part that no reader will reach after being confused by the informal explanation). Instead, dare to present the reader a formal definition that he just needs to read and accept at first. Then after that definition, you can start explaining why that definition makes sense and why you chose to model certain parts the way you did (note that you can now be very concrete in your explanation and point to certain parts of the definition).

What you *can* do before the formal definition is a rough high-level intuition. For example:

> "in a credential system, there are two roles: issuers and users. Users receive credentials from issuer.“

> "We want to achieve that the user stays anonymous in the presence of an adversarial verifier."

Don't: (because it's incomprehensible)

> In this definition, the adversary $\mathcal{A}$ will be given the public key pk and will be allowed to query an oracle in the first phase, but not in the second phase.
> **Definition** (Credential security):
>
> - $\mathcal{A}^{\mathcal{O}(\cdot)}(pk)$ ...

Also, don't write the whole definition in text:

> For this definition, $\mathcal{A}$ is given the public key pk and outputs y. $\mathcal{A}$ can then query an oracle with input x, as long as $x \neq y$ to receive a credential *cred* on attribute x. Eventually, $\mathcal{A}$ outputs z and wins if z is a valid credential on y.

That's what formal pseudocode-like definitions are for. They're much more easily readable and, often, more precise.

Do not overuse formulas You may be tempted to think “in order to be very precise, I need to rely on formulas”. This is a myth; being precise does not require formulas. As an easy example, writing “let $(a_1, \dots, a_n)$ be a sorted list of integers (ascending)” is exactly as precise as “let $(a_1, \dots, a_n) \in \mathbb{Z}^n, \forall 1 \leq i < n: \ a_i \leq a_{i+1}$”. Obviously, the former version is much easier to parse for humans.

Use formulas! (in moderation) On the internet, you’ll sometimes be advised to avoid formulas when explaining a concept. But even in “informal” explanations, formulas have a useful function. For example:

- Give things names if that makes it easier to follow. For example, don’t write

  > The verifier sends two challenges. The first prover computes an answer for the first challenge and generates an additional challenge. The second prover computes his answer for the verifier’s second challenge and for the first prover’s challenge. The verifier then checks that the first prover’s challenge is well-formed and that the answers of the first and second prover are consistent to their respective challenges.

  Instead, write

  > The protocol is between a verifier V and two provers P_0, P_1. V sends two challenges c_0, c_1. P_1 computes an answer a_0 for c_0 and generates an additional challenge c_2. P_2 computes his answers a_1 for c_1 and a_2 for c_2. V then checks that c_2 is well-formed and that each answer a_i is consistent to its challenge c_i.”

- Use formulas if they make it easier to follow. For example, don’t write

  > the verifier computes a proof, which is a list of group elements. Each group element is the previous group element in the list raised to the power of its own list index

  Instead, write

  > the verifier computes a proof $\pi = (g_1, \dots, g_k) \in \mathbb{G}^k$, where $g_i = g_{i-1}^i$ for $i > 1$”.

5.11 How long should the thesis be? Do you have too few pages?

We don’t care about page numbers. We care about content. Page numbers are a bad approximation of content.

Generally: if your thesis is short but contains everything outlined in the proposal, you’re fine. If your thesis is very long but doesn’t feel like you’re just filling it with bullshit, you’re fine.

According to the examination regulations (“Prüfungsordnung”), Bachelor’s theses usually shouldn’t exceed 60 pages. Master’s theses usually shouldn’t exceed 120 pages. Theses in our group tend to be pretty dense (i.e. lots of content per page). For this reason, it’s normal to have a lot less pages than 60/120.

6 Starter talk

The goal of the starter talk for you is to show that you are able to handle the more complicated details you are confronted with during your work. That is, you should dive into details for one small thing that plays a role in your thesis. Depending on your topic, this can be, for example:

- a basic formal definition
- a basic formal proof
- a software architecture
- example code (/structure)

The goal is just to show that you can handle the basics, there is no need to choose anything exceedingly complicated here. Other remarks:

- The starter talk should consist of (1) very brief introduction of the context of what you're talking about next (why are we talking about this? What role does it play for your thesis?), then (2) go explain the thing you decided to explain.
- The talk should be a few weeks after starting work on the thesis. There is no fixed timeline for this, but the idea is that you are at a point where you understand some basics in detail already.
- Ask your advisor for suggestions for suitable topics.
- Usually, you should be able to do this at a whiteboard. If your topic is such that you would have to write *a lot*, then consider making slides or a handout.
- Prepare this! Don't waste too much time with it, but at least have a good plan what to write and say. Usually students will have a piece of paper with them that has all the formulas and stuff.
- Plan your explanation to take roughly 20 minutes.
- This is a closed-door meeting with only your advisor and first examiner (and maybe one more person or so).

7 Submitting the thesis

Here are the formalities for submitting the thesis.

- The central examination office (Zentrales Prüfungssekretariat) needs:
  - Two printed copies of the thesis (use appropriate book binding, e.g., hardcover).
    - Make sure to sign the "Eidesstattliche Versicherung" in both.
  - A physical medium containing the thesis as a digital file (e.g., pdf on USB memory stick). This is not necessary anymore according to the new "Rahmenprüfungsordnung".

  You can submit these directly at the central examination office, if they're open, or you can hand it to the ServiceCenter in an envelope addressed to central examination office (look up what person to address). If everything else fails, you can instead give this envelope to the Pförtner ("doorman"?).

- Send a digital copy of your thesis to your first and second examiners and to your advisor. This copy *must* be the same as you printed.

Congratulations! That's it. :) Take a deep breath. Only the final presentation is left now.

8 Final presentation

In your final presentation you show that you're able to explain your topic in a talk of 20-25 minutes plus discussion (Bachelor) or 30-40 minutes plus discussion (Master). Because of the time constraints, this is very different from explaining the topic in the thesis: in your thesis, you have almost unlimited time and space to explain every detail of your work. In the final presentation, however, you are very limited in time. Hence the goal here is to give a more high-level explanation of what you've done. With the thesis, you have already shown that you understand all the details. **Now you need to show that you understand (and can explain) the high-level ideas, how they relate, and what everything means in a broader context.**

You would usually talk about things like:

- Why is the topic interesting?
- What are the big ideas?
- How do things fit together?
- What are the *implications* of this theorem? (Instead of its proof).
- *Maybe* a rough proof idea, if the proof was very central to the thesis.
- What is the role of this part of the construction?

For this reason, you need to be selective in what you (can) talk about. There is no generic recipe for doing this. Don't try to explain *everything*; however, you must explain *something* in sufficient detail that the audience can understand (don't keep to the surface!). It's a delicate balance. On the one hand, don't explain to us why encryption is useful to humanity and dates back to Julius Caesar. On the other hand, don't explain to us why equation (3) on page 27 holds.

You should **highlight what your achievements are** and what the **most notable problems** you encountered were. Make sure to convey what you came up with yourself for the thesis and what was already there in other people's papers.

8.1 Structuring the presentation

There is no standard recipe for structuring the final presentation. It depends a lot on the thesis. The following is a rough example structure (maybe a starting point for your own ideas):

- What's the problem this thesis tackles? (brief)
- A good and formally precise explanation of some *basic version* of your topic (this does not need to be the main contribution). This should give the audience some starting point to be able to understand your later explanations better. For example:
  - the standard construction for group signatures/anonymous credentials.
  - the naive way to compute a bilinear pairing.
  - the standard security definition for adaptive/selective IBE.
  - the construction(s) you base your thesis on.

- Building upon the basic version before: explain problems with the basic version. Explain your extension. It's okay to be less formal here, if it becomes too complicated. It's also okay to still leave out things (it does not have to be the exact version that's in your thesis).

- Overview slide at the end: what has happened in the thesis? This includes things you did not get to talk about during the presentation. The slide should be useful for questions and discussion afterwards.

8.2 Some random hints on presentations

- To getting started do a graphical abstract. Take a DIN A5 page and try to convey your topic with important details graphically on one page. Examples for details are security, usage, and specific problems. The goal is that a person that you meet in an elevator can easily understand your topic and problems by just looking at the graphic abstract. The person should then be eager to know when your final presentation will take place.

- Be clear to explain what's happening right now. A good way is to do "now that we've seen what our requirements for X are, let's talk about how to securely construct X" segues. You can show the structure of the talk very briefly (because it's often just a waste of time) in the beginning, but that does not replace good segues during the talk.

- It is *not* the goal to cover everything you did. You may have spent a lot of time to get technical details right, but now is not the time to showcase those. Now is the time to show that you can explain what you've done. If this consists of 80% bringing the audience up to speed on the previous state of research, so that they can understand the problem you tackled, and 20% a rough intuition of your solution, so be it.

- It's okay to simplify! This doesn't mean that you take a formal definition and omit a few variable definitions (don't do that, people will think you just got it wrong). There are generally two ways to simplify:

  - Zoom out. What is the *idea* of the thing you're explaining? Don't mention unimportant details at all, just give a good intuition of what's going on. But be careful that you don't end up saying nothing substantial – choose the right level of abstraction.
  - (This may be harder to pull off!) Keep the full formal definition on the slide, but during the presentation say "the details don't matter here, the general idea is that ..." and lead the audience through the most important ideas in that definition.

  Often, (after appropriate exposition, e.g., introducing naive versions), main ideas can be condensed down to 4-5 very carefully chosen sentences. Say them during the presentation and put a summary of them onto the slide.

- Work with colors. If your scheme is a combination of three building blocks, then give each of them a color, e.g. text color or color in a figure.

- Use graphics and figures as a starting point for your basic version of your problem or scheme.

  - Use the graphics as a tool to convey not only simple things, but also complex structures, e.g. how all building blocks are used in the scheme.
  - Use animation states (also possible in LaTeX Beamer)

  - Add formal details to your figures. This can be done on the last animation state or you can add the formal details step by step.

- Reuse your colors and figures. This helps the audience to quickly recognize and identify what you are talking about.

- Standard questions from the audience are:
  - I didn't understand the thing on slide 17.
  - What is future work? Can [thing] still be improved?
  - Did you look at [another thing]? ("no" is a perfectly reasonable answer)
  - You showed the proof outline of [thing] and said that at the end the view of the adversary is independent of the bit b, but you said that there is still something left to prove. Why?

8.3 Making slides

What software to use. Use LaTeX Beamer if your topic is very technical and is mostly talking about formulas. Otherwise, you can use Microsoft's PowerPoint, Apple's Keynote, or OpenOffice's Impress. It's slightly more effort to include formulas in those, but they have the advantage that you can easily sketch illustrations and use animations to make things clearer. Making slides that are not "just text" comes with much less effort in these tools. We have templates for all of those tools on our website, but you can use your own, too. Honorable mentions: Prezi, Jupyter (with RISE).

What are slides for The slides are there to help the audience follow what you're saying. They are *not* there as a script for you to read from. They are also *not* meant to be able to replace you talking.

Basic rules what to avoid

- Avoid long text on slides. Yes,

  > Let the set **P** denote the set of all languages $L \subseteq \{0,1\}^*$ that can be decided by a deterministic Turing machine in polynomial time. Let **NP** denote the set of all languages that can be decided by a nondeterministic Turing machine in polynomial time.

  is a very good definition, but it's impossible to read on a slide. Instead, just put

  > \- **P**: languages decidable by poly-time **DTM**.
  > \- **NP**: languages decidable by poly-time **NTM**.

  emphasizing what is most important for your presentation and summarizing what you want to say. Then fill in the gaps by talking.

- Avoid slides that are just five or more uniformly looking bullet points, like

- In general, this problem is hard.
- This is mostly because of clock drift.
- There are no standard model solutions.
- Solutions exist in joint clock model.
- Open question: other models (e.g., ROM)

The audience has to be able to quickly find on the slide what you're talking about. And they must be able to easily see the structure without reading too much. For complex overviews, use a figure, showing how all this stuff relates (arrows. . .) and giving it a spatial dimension. If you do have such a list and it's not feasible to make it a figure, at least don't reveal all points from the start.
Also, sometimes . . . just throw some bullet points away and only keep what you want to emphasize:

- No standard model solutions.
- Open question: other models (e.g., ROM)

Then just *say* the rest in your presentation.

- If you talk about something important for longer than a minute, put *something* on the slide for it. The audience will sometimes want to check where you are in your structure, give them something to recognize (and regularly point it out yourself).

Random hints

- Number your slides. That's useful for questions.
- As a rule of thumb, you usually need about two minutes per (content) slide. Plan how many slides (and generally how much content) you can reasonably fit.
- Figures can be *very* helpful. A picture says a thousand words; or – it allows you to say a thousand words about it without losing everyone in the process.

8.4 Rehearse your presentation

There is not much to say here, but it gets its own section because it seems to be heavily underrated. **Rehearse. Out loud.** Yes, it feels weird, but it's *very* helpful.

- Make sure you know what to say. You don't have to/shouldn't learn your text word for word, but you should generally know what you need to say and in what order. Don't be surprised by your own next slide ;-)
- Make sure what you're saying feels like people can understand it. This may be difficult, but try to put yourself in your audience's shoes. Can they easily follow your arguments? Or may they get lost in some explanation you're doing? Have you already told them everything they need to know to understand this next point?
- Check how much time you need.

9 Being graded

Three parts play a role when grading your thesis: (1) the thesis itself, (2) our impression of your work process, and (3) the final presentation.

9.1 A good thesis

The thesis is by far the largest part of your overall grade. In the first step we usually compare your achievements in the thesis to your goals (the proposal for bachelor's and master's theses). Of course, if there are good reasons why some initial goals could not be fulfilled (e.g., we grossly underestimated the scope of some tasks), this is taken into account.

Next, we check formal correctness of definitions, claims, proofs, etc. It is most important to us that your proofs are comprehensible and correct, and that definitions, theorems, and explanation fit together.

The grade depends largely on whether your writing can convince us that you really understand what you are talking about. This is mostly achieved by simply being consistently and formally correct; we will be even more convinced if you are able to give good higher-level explanations of formal facts.

Finally, we assess the structure and coherence of the document and check whether it fulfills the standard expectations of scientific works. For this, we check proper citations, language, readability, references to your own definitions/theorems/sections, etc.

9.2 A good work process

The work process usually does not factor into the grade too much. However, we do read the thesis with our impression from the work process in mind, which usually works in your favor:

- We're somewhat more forgiving with errors or sub-par explanations if you have shown a great understanding of a topic during the work process.
- We take time lost on technical issues into account.
- We appreciate difficult proofs more if we know what its challenging parts were during the work process (these do not necessarily show up in the final proof).

Overall, make sure we know what you're working on and where current problems lie.

9.3 A good final presentation

The final presentation is part of your grade, but usually does not count that much. Do a good talk. We usually grade by "does it look like this person understands what they're talking about?". Note that this does not mean that you should try to explain incredibly difficult stuff. You can demonstrate your understanding *much better* by presenting a good presentation-appropriate abstraction of your content.

A Setting up your writing environment

(List of tools last updated December 2019. If you stumble upon better/newer software, let us know!)

A.1 LaTeX

To write LaTeX code, you need a distribution and, ideally, an IDE. You don't have to follow our recommendations below. Let us know if you found better tools.

LaTeX distribution

- Windows: MiKTeX
- Linux: TeX Live (`sudo apt install texlive-full`)
- macOS: MacTeX

LaTeX IDE We have had good experiences with the following:

- TeXstudio (Windows/MacOS/Linux)
- Overleaf (Online, no setup required)
- Texpad (MacOS/iPadOS, paid)
- Atom/vscode (Windows/MacOS/Linux, "build your own editor", very customizable through plugins, needs more setup than the others)

Useful packages/things you may not know

- `\usepackage{cryptocode}`: good (but not always necessary) for writing crypto-pseudocode, protocols, or reductions. Also contains a bunch of predefined macros useful for cryptography text (like `\pk`, `\negl`, . . .).
- `https://sylvainhalle.github.io/textidote/`: Grammar checker
- You can use `\usepackage{cleveref}` for a `\cref{thm:schwartzzippel}` command that evaluates to "Theorem 3.2". Without this package, you'll have to to do more of this manually and write `Theorem~\ref{thm:schwartzzippel}` for the same effect. For example, that's useful if you tend to change lemmas to theorems or vice versa without wanting to go through all references. Look online for more info.

A.2 git

Using a versioning system like git allows you to back up your thesis work reliably. Not only is your current state backed up, but you can feel at ease deleting text given that you can always retrieve old versions of your document through git.

Since you're working on this project alone, you probably don't expect merge conflicts. Hence the standard command-line git (`apt install git`) is pretty feasible.

If you do want a GUI, consider

- fork (Windows, MacOS)
- SourceTree (Windows, MacOS)
- Sublime Merge (Windows, MacOS, Linux)
- Working Copy (iPadOS, paid)