Open Source Development and Sustainability

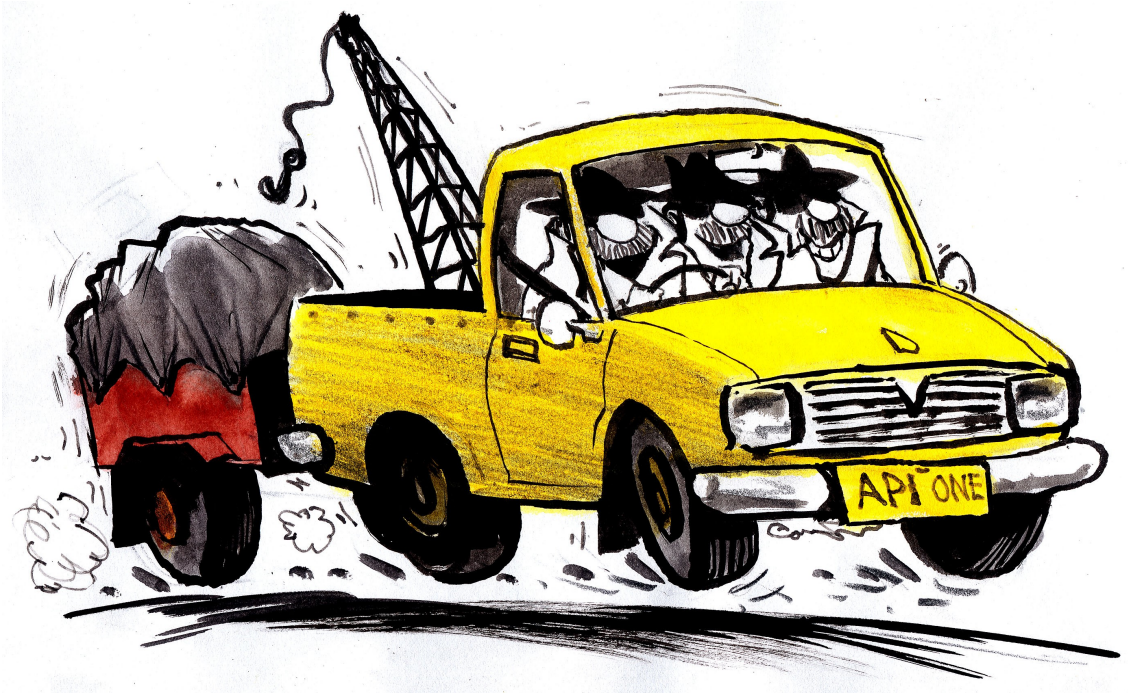
A Look at the Bouncy Castle Project

How It Started

Early Days

- Started with a low level API as one of us was playing around with the J2ME, built a provider on top of it.
- Added functionality for generating X.509 certificates.
- Then, of course, CRLs.
- Over the next couple of years, a few more algorithms (e.g. Elliptic Curve), improvements and additions (PKCS#12 support), and then...

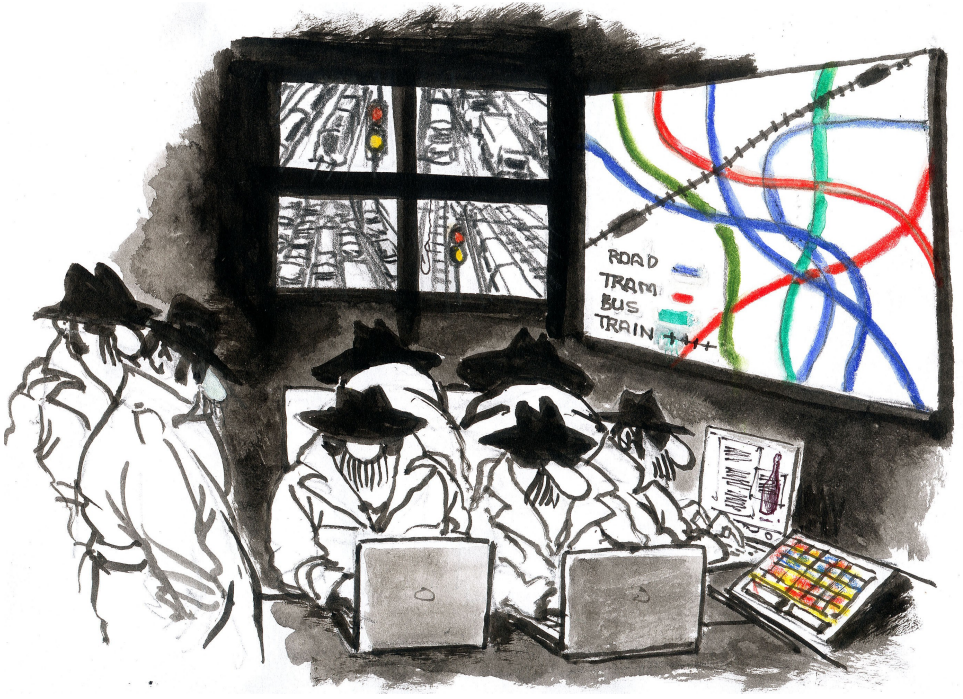
Really Up and Running



More Features!

- Support for Cryptographic Message Syntax
- Support for Time Stamp Protocol
- Support for OpenPGP
- Attribute certificates
- A broader range of standards (paddings, algorithms)
- And more! But...

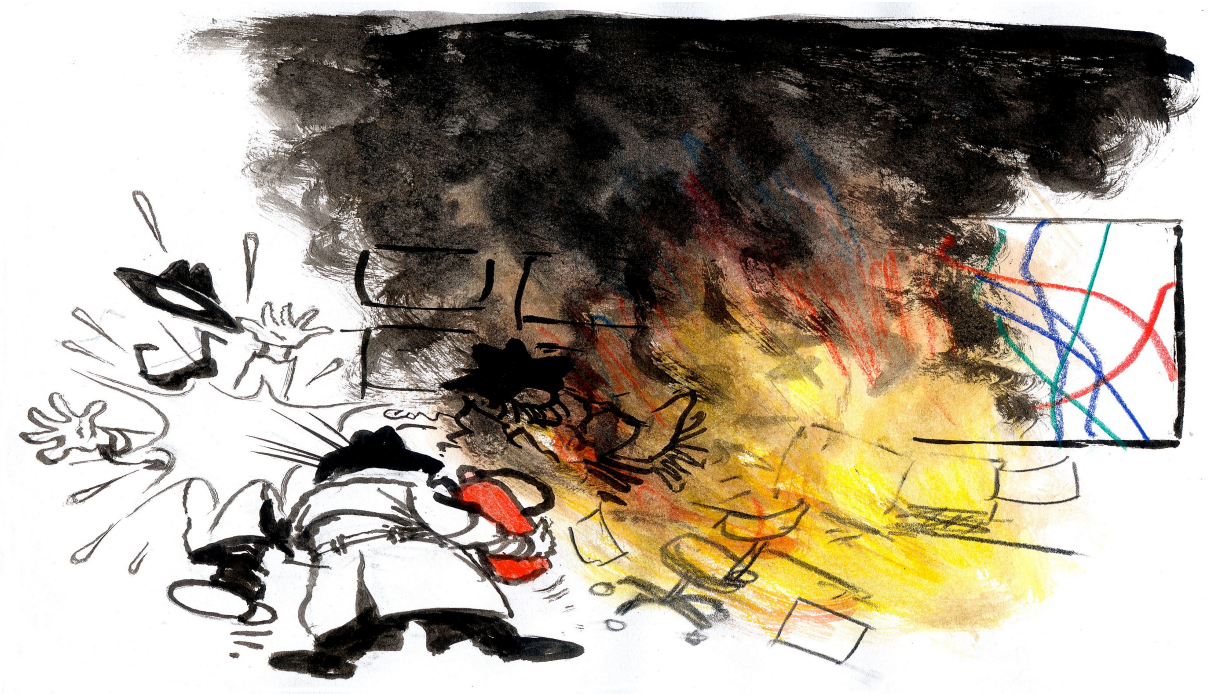
Suddenly, There is Complexity



And Realisation

- It's no longer just something you put on the Internet because you found it useful and thought someone else might.
- People are actually relying on it.
- You even find out your bank is relying on it!
- Reviewing the situation in this light, one rapidly realises that as good as everything is, it's one step from...

Chaos and Disaster



Principal Constraint - Time

- The issue isn't really ever money – it's actually time.
- Money does help free time up but is not a solution in itself.
- Lack of time can result in poor, or incomplete, test coverage, hasty check-ins, incomplete functionality.
- Favourite brother to “lack of time” is interruption. Interruption on second tier work is often and generally lengthy.
- Often open-source work has to be treated as second tier.

Other Constraints

- Equipment – faster computers, quicker turn around, servers for continuous testing.
- Infrastructure – issues trackers, mailing lists, website, managing distributions, download areas, 3rd party deployment (e.g. Maven central).
- These days, the costs for most of these are modest, again the issue is time, time to administer and time to make use of what infrastructure you have.

Other Problems

- Ideally people outside the core team should be able to contribute.
- Suddenly it takes as long, sometimes longer, to review a contribution than it would to do it locally.
- Large contributions become especially problematic, particularly if they involve standards such as ASN.1, as even experienced developers frequently make errors.

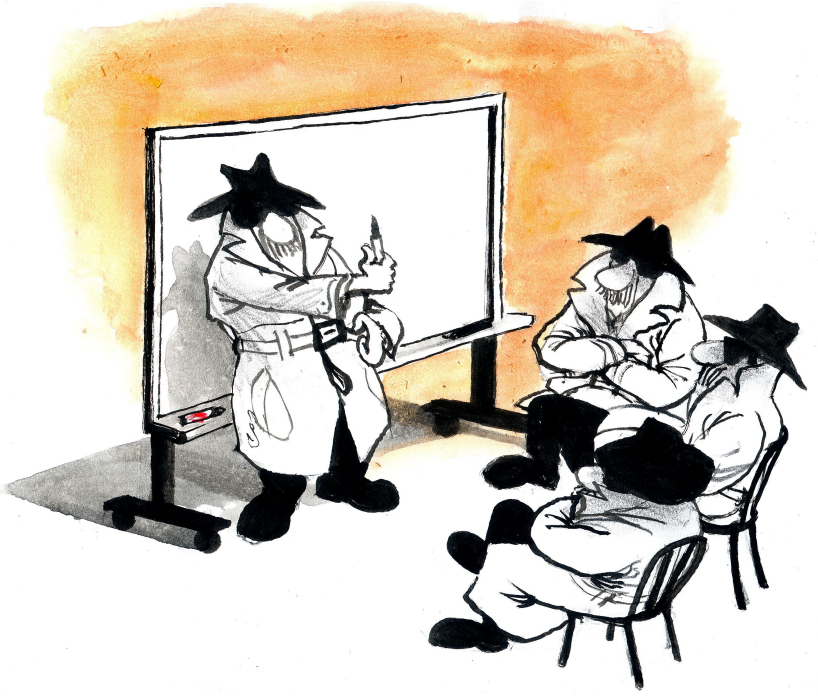
Biggest Danger

- Accrual of technical debt.
- A big issue in security orientated software as in some cases things can go from being great to useless, possibly even dangerous, overnight.
- Can also result in code which is difficult to maintain, again making it difficult to respond to changes.

Then of Course...

- Peoples' lives can change
- External pressures can change
- Children, dogs, cats...
- The bank that's using your software suddenly becomes the one that also holds your mortgage.
- Is this really what I signed up for?

So What to do?



Immediate Thoughts

- Rely on donations?
- Maybe a product company?
- Fund through consultancy work?
- Change license?
- Public/Professional version?
- Run?
- Before doing any of the above, need to consider what you want to preserve as well.

In our Case

- Decided not to run.
- Wanted to preserve open source. Openness the best approach for cryptography software.
- Donations unreliable. Not tax deductible in themselves.
- License fees, community/professional model not really an option. Can't do “partial” cryptography, risk of introducing errors unacceptable.
- Contracting helps a bit, but have to be careful as it rarely means working directly on the APIs. Doesn't buy much time.
- Product built on API approach also problematic, same issue as contracting.

The Solution

- Established a charity with ownership of the code base.
- Established a company for actual commercial work.
- Really had to find a way to make the APIs and the “product” related.
- Only accepted short-term consulting targeted to the APIs.
- Started selling support contracts.

The Product

- Turned out to be support contracts.
- Question then is why would someone buy a support contract?
- Some people will buy one because they want to support the project, or they actually know they need support.
- Most people need something tangible that's different from the public offering.
- In our case, early access to certification work.

Things You Wrestle With

- “Freeloading” - is that what's really happening and what does it mean?
- Do people really understand where the money goes when they buy software?
- Turns out “not paying” and “freeloading” aren't always the same thing.
- That said, there are advantages in having a large user base for a Crypto library if you can keep up with the users.
- These advantages also benefit paying customers.

Other Things That Change

- If something needs to get done, it cannot be treated as second tier work.
- Different risks emerge, a lot of knowledge in the heads of too few people.
- To deal with these it means the project needs to expand, and people need to be paid.
- Not only have to manage the code, but manage the knowledge.

It's not just the code base we need to preserve!

On Reflection

- Many of the issues are really the same you face with any business.
- If you need an income, you have to have something to trade for cash.
- In commerce everything is quite simple, but even simple things can seem quite difficult...
- If you are running, or setting up, an Open Source project you should think about these things early.

Thanks for listening.

Any questions?