Technology TLS/SSL (and related libs such as OpenSSL)

Encrypted network communication, e.g. over http. Uses certificates to provide initial identification which are issued by a trusted certificate authority. Certificates contain server name, CA and servers' PK. Then generates session keys e.g. using Diffie-Hellman key exchange

Developers are not the enemy

- Usability problems are major cause of IT security incidents
- People strive for usable end user interfaces
- People expect that developers do not make mistakes even if an API is barely usable
- APIs expect developers to have a deeper understanding of crypto than realistic

Principles:

- Best: hide crypto completely behind API, don't force the developer into making crypto architecture decisions (e.g. password hash storage)
- Satisfy security and non-security requirements, otherwise developers will be forced to write their own code instead of using the API
- API is easy to learn, best case self-explanatory. Don't expect the developer to understand the crypto background
- Don't break developer's paradigm, provide the API he will expect (send async enc msg func → API expects public key of recv, not some other weird stuff)
- Easy to use, even without docs, at least not easy to misunderstand without docs
- Create API which is hard to misuse and leads to visible errors when misused and which does not allow further normal execution in these cases
- Don't provide ambiguous or unsafe defaults
- Provide test mode for development, so the developer doesn't has to write custom code to turn off security (will probably forget to remove this code and where it is)
- Easy to read and maintain code which uses the API, don't let the developer choose the amount of iterations for a hash function (will hardcode it and probably not update it), provide a safe setting which is update in newer API versions if necessary
- Assist with end-user interaction, especially in the case of security related errors. Even if the developer understands what went wrong, it is difficult to appropriately raise error messages to the end-user, so he doesn't proceed in a dangerous way (e.g. ssl warnings in browser)

Usability Smells: An Analysis of Developers' Struggle with Crypto Libraries

Tries to identify indicators ("usability smells") for difficult to use APIs similar to code smells, which are an indicator for difficult to maintain or bad code

Identified Usability Issues:

- Missing Documentation
- No example code for common use case
- Clarity of Documentation

Breaking dev's paradigm, providing visible and early errors Documentation is missing or unclear

- How should I use this, can I use this, Dev asks how functionality of API 1 works in API 2

Intuitive principle, easy to learn, dev's paradigm, standard APIs

- What's gone wrong here, unsupported feature, deprecated feature

Hard to misuse, dev's paradigm, safe and sensible defaults, help with end-user interaction to ensure dev's are doing things the standard way, easy to read API code

Implications on SSL/TLS libs

- Provide TLS connection functions which expect a https url and do the rest
- Fail visibly for untrusted certificates and don't allow untrusted connections
- Provide test mode where e.g. self-signed certificates are accepted, which is (automatically) switched of during release compilations
- Don't let the user crucial settings such as the TLS version, automatically select the most secure one and update it in newer versions of the API
- Provide well-crafted warning messages for the user, which can be directly forwarded by the developer and describe the issue, it's consequences and the danger appropriately