VerifyThis
The Long-term Challenge

Marieke Huisman, Raúl Monti,
Mattias Ulbrich, Alexander Weigl

November 27, 2020

Tradition of VerifyThis

- **VerifyThis** – *"Verification Competition with a Human Factor"*
- On-site event – *bringing people together, foster discussions*
- Yearly Workshop at ETAPS

VerifyThis – The Long-term Challenge

`https://verifythis.github.io`

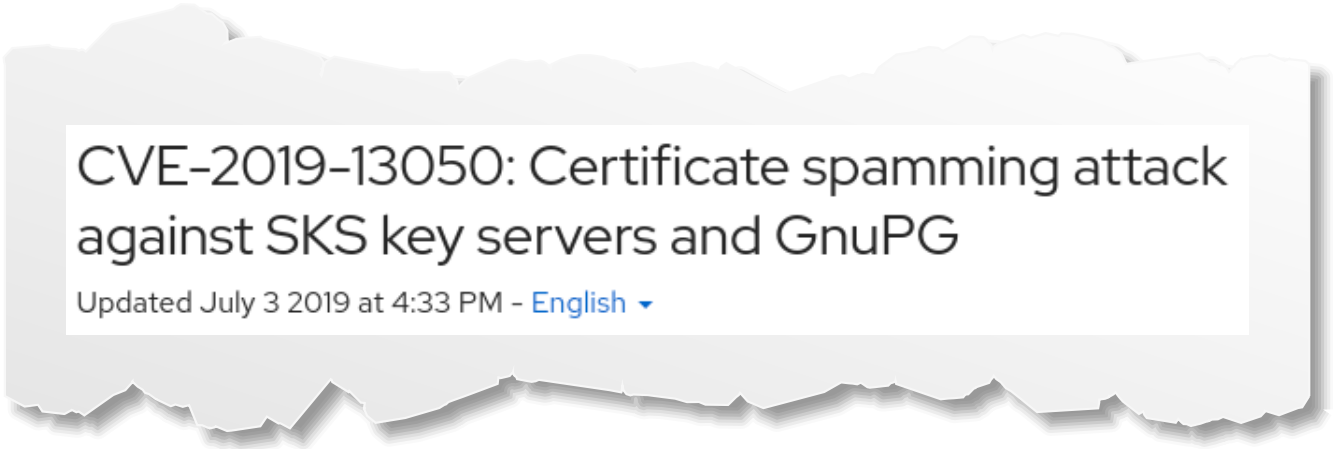
- 6 months time
- Security / Safety real relevant system
- Reference implementation, can be reimplemented
- Requirements given in natural language
- Various degrees of abstraction possible
- Collaboration explicitly promoted

Today

Plan for today:

- Brief introduction to the Long Term Challenge
- Open discussion on program specification (and verification)
  - Guided by few questions
  - Concerning the target of the challenge ...
  - ... but generalising beyond the very concrete points.

Verification Target



- Security issues (anyone can upload keys)
- Denial of service attack ("monster key")

Solution

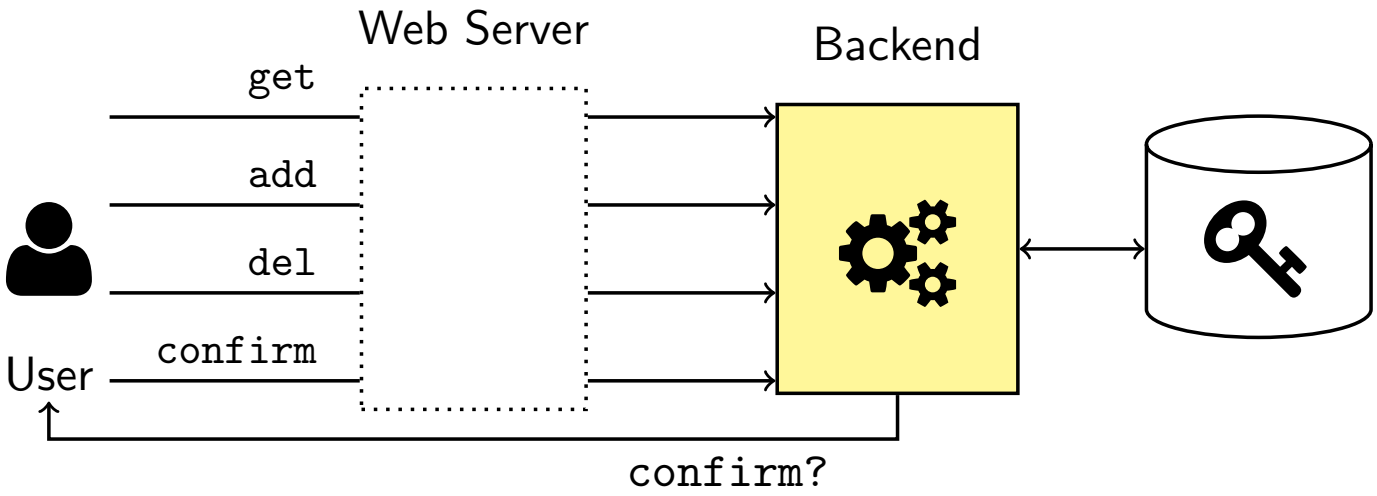
The Verifying Key Server

Even better solution

The Verified *Verifying* Key Server

Verification Target

Verification Target: The *Verifying* Key Server



- Reference Implementation: HAGRID.
- Deployed as default key server keys.openpgp.org
- Prototypical example of a stateful, responsive system

Missions (extract)

1. Safety Verify that the implementation of the key server does not exhibit undesired runtime effects (no runtime exceptions in Java, no undefined behaviour in C, ...)
2. Functionality Specify and verify that if an e-mail address is queried, the respective key is returned if there is one.
3. Pricacy Specify and verify that if an e-mail address has been deleted from the system, no information about the e-mail adress is kept in the server.
4. Thread safety Prove that your implementation is free of data races.
5. Termination Prove that any operation of the server terminates.

...

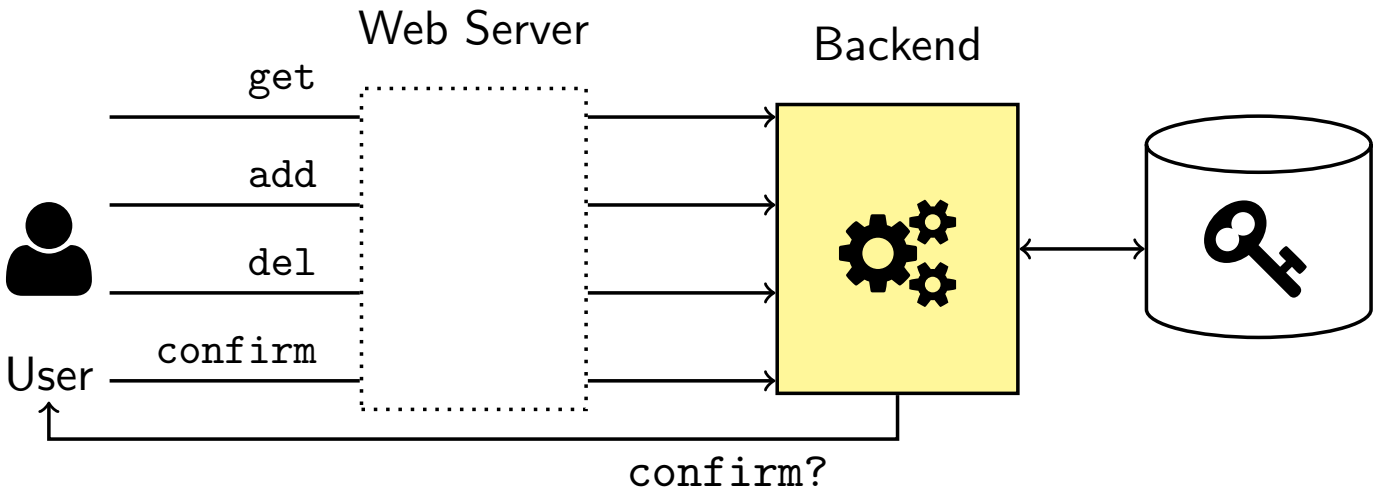
Contributions

- Ernst and Rieger:
  Information Flow Testing of a PGP Keyserver
- Diverio, Loureno and Marché:
  "You-Know-Why": an Early-Stage Prototype of a Key Server Developed using Why3
- de Gouw, Ulbrich and Weigl:
  The KeY Approach on Hagrid
- Dross, Kanig, and Moy:
  A Solution to the Long-Term Challenge in SPARK
- Ernst, Murray and Tiwari:
  Verifying the Security of a PGP Keyserver
- Ulbrich: (not in proceedings)
  Event-B Formalisation of the Key Manager

Proceedings:
`https://publikationen.bibliothek.kit.edu/1000119426`

Verification Target

Verification Target: The *Verifying* Key Server



(Slides with more details on the webpage)

Guiding Questions

Specification Aspects

- What purposes (apart from verification) can a specification have?
- What should a specification express?
  - whitebox props like concurrency (implementation-dependent)
  - blackbox props like security? (implementation-independent)
  - other characterisations?
- What are the reasons that formal specs are little used in practice?

Contracts

- Are constracts the right specification methodology for HAGRID?
- How to specify such services/protocols? Databases?
- Whats the best abstraction level for a contract language?
- One specification language or several langs for spec. aspects/abstr. levels?
- Are there "core clauses"?

Verification Tools

- Which symbolic debugging ideas can be used for formal verification?
- Is guiding the prover in specs via annotations a good idea?

Natural Language Specifications

Requirements for retrieving a key

	get(e : EMAIL) returns k : KEY $\cup \{\perp\}$
Pre	none
Post	If $k \neq \perp$, then the returned key k is associated with the given email address e in the database. $k = \perp$ iff there exists no entry for the given address e.
Effects	No changes on the database or pending (add or delete) confirmations.

Requirements for adding a key

	add(e : EMAIL, k : KEY) returns c : CONF-CODE
Pre	e and k are well-formed entities. e is an e-mail address to which the public key k applies. The tuple (e, k) may or may not already be present in the database or a confirmation for (e, k) may be pending.
Post	The confirmation code c is unique in the system. If (e, k) is present in the database, ... If a request is pending for (e, k), ...
Effects	The database remains unchanged. All pending confirmations are preserved. The only effect of the operation is that a con

And now?

How do we continue from here?