JACKLINE

A SECURE FUNCTIONAL INSTANT MESSAGING APPLICATION

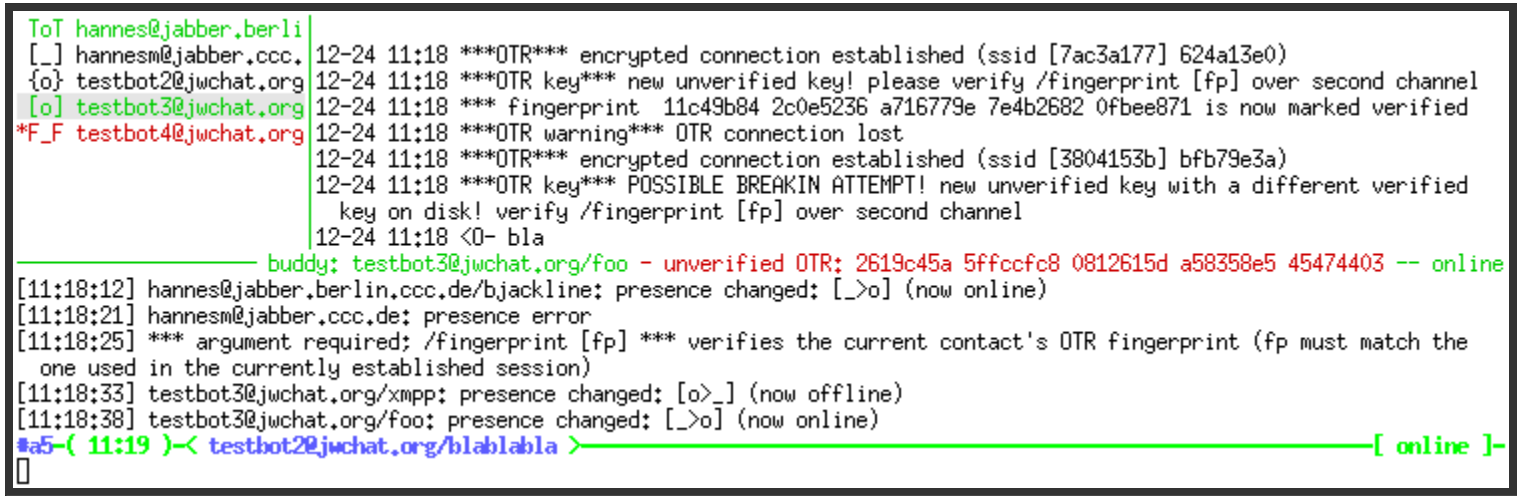
Hannes Mehnert
@h4nnes

University of Cambridge, Computer Labs

bobkonf, Berlin, 19th February 2016

ABOUT MYSELF

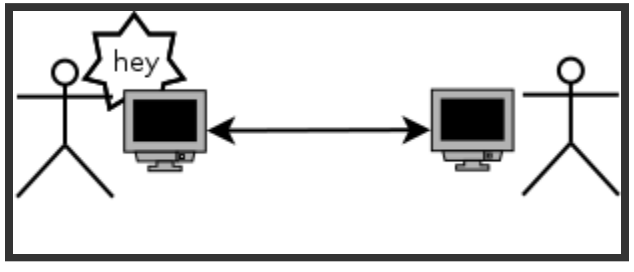
- Full-stack engineer
- Also appreciate good coffee and cycling :)


ToT hannes@jabber.berli
[_] hannesm@jabber.ccc.
{o} testbot2@jwchat.org
[o] testbot3@jwchat.org
*F_F testbot4@jwchat.org
12-24 11:18 ***OTR*** encrypted connection established (ssid [7ac3a177] 624a13e0)
12-24 11:18 ***OTR key*** new unverified key! please verify /fingerprint [fp] over second channel
12-24 11:18 *** fingerprint 11c49b84 2c0e5236 a716779e 7e4b2682 0fbee871 is now marked verified
12-24 11:18 ***OTR warning*** OTR connection lost
12-24 11:18 ***OTR*** encrypted connection established (ssid [3804153b] bfb79e3a)
12-24 11:18 ***OTR key*** POSSIBLE BREAKIN ATTEMPT! new unverified key with a different verified key on disk! verify /fingerprint [fp] over second channel
12-24 11:18 <O- bla
buddy: testbot3@jwchat.org/foo - unverified OTR: 2619c45a 5ffccfc8 0812615d a58358e5 45474403 -- online
[11:18:12] hannes@jabber.berlin.ccc.de/bjackline: presence changed: [_>o] (now online)
[11:18:21] hannesm@jabber.ccc.de: presence error
[11:18:25] *** argument required: /fingerprint [fp] *** verifies the current contact's OTR fingerprint (fp must match the one used in the currently established session)
[11:18:33] testbot3@jwchat.org/xmpp: presence changed: [o>_] (now offline)
[11:18:38] testbot3@jwchat.org/foo: presence changed: [_>o] (now online)
#a5-(11:19)-< testbot2@jwchat.org/blablabla >-[online]-

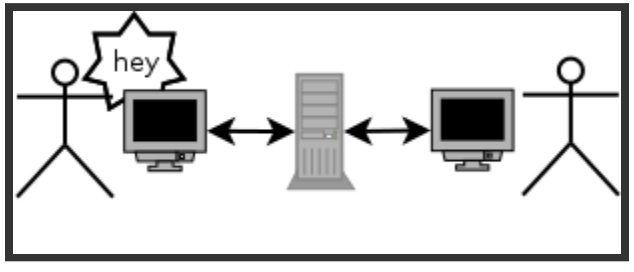
MOTIVATION

- I use instant messaging daily
- Love functional programming
- Use the terminal quite a lot
- Like to build things from the grounds up
- Eat my own dogfood

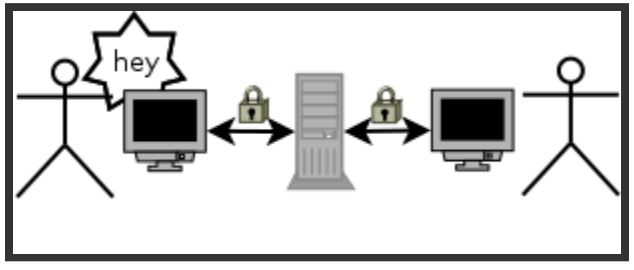
INSTANT MESSAGING



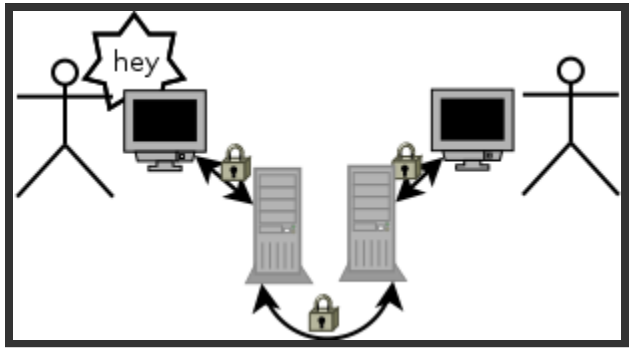
IM WITH SERVER



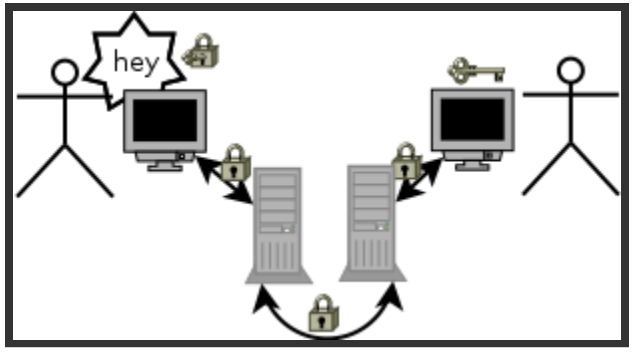
ENCRYPTED TRANSPORT LAYER



FEDERATED (XMPP)



END-TO-END ENCRYPTED



XMPP CLIENT

OTHER AVAILABLE CLIENTS

- Various XMPP clients are around
- Even some using the terminal
- Mostly written in C, suffering from security issues
- I want a tiny human-readable code base

JACKLINE

- Written in OCaml
- Unicode libraries (uutf) already available for OCaml
- Also libraries for XMPP, XML, TCP/IP
- We developed OCaml-TLS
- Terminal library (notty)
- "Only" missing: end-to-end encryption (OTR) and a UI

FUNCTIONAL PROGRAMMING IN OCAML

- Memory safety
- Type safety
- Explicit flows of data
- Containment of side effects
  - Input/Output
  - Mutable state
- Explicit error handling
- Not so much objects and exceptions

MODULE SYTEM

- A module is independent of other modules
- Takes modules as parameters
- Use its signature, not implementation

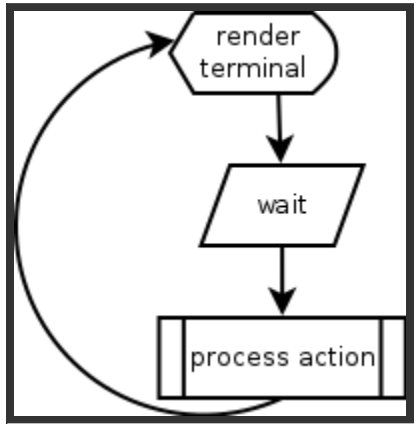
EXAMPLE: STORAGE

- `init : () -> storage`
- `load : storage -> key -> data`
- `store : storage -> key -> data -> unit`
- Can be satisfied using alist, hashtable, map, file system, ...

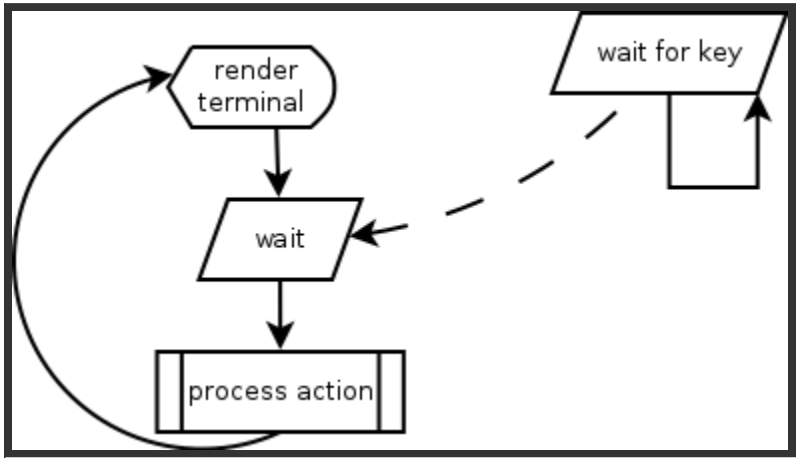
DESIGN ISSUE

Two inputs - terminal and network - both use some shared state

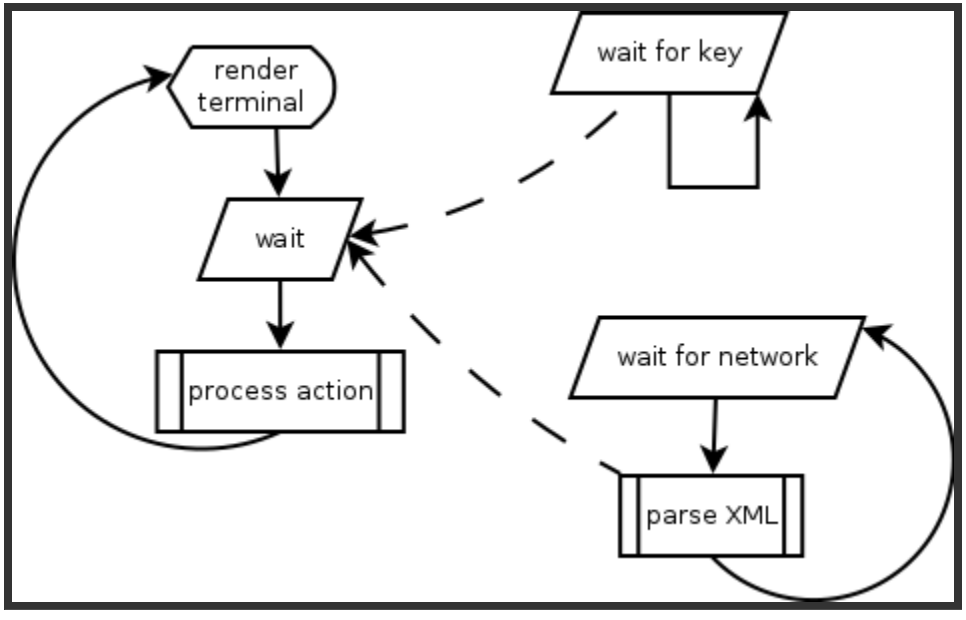
MAIN TASK



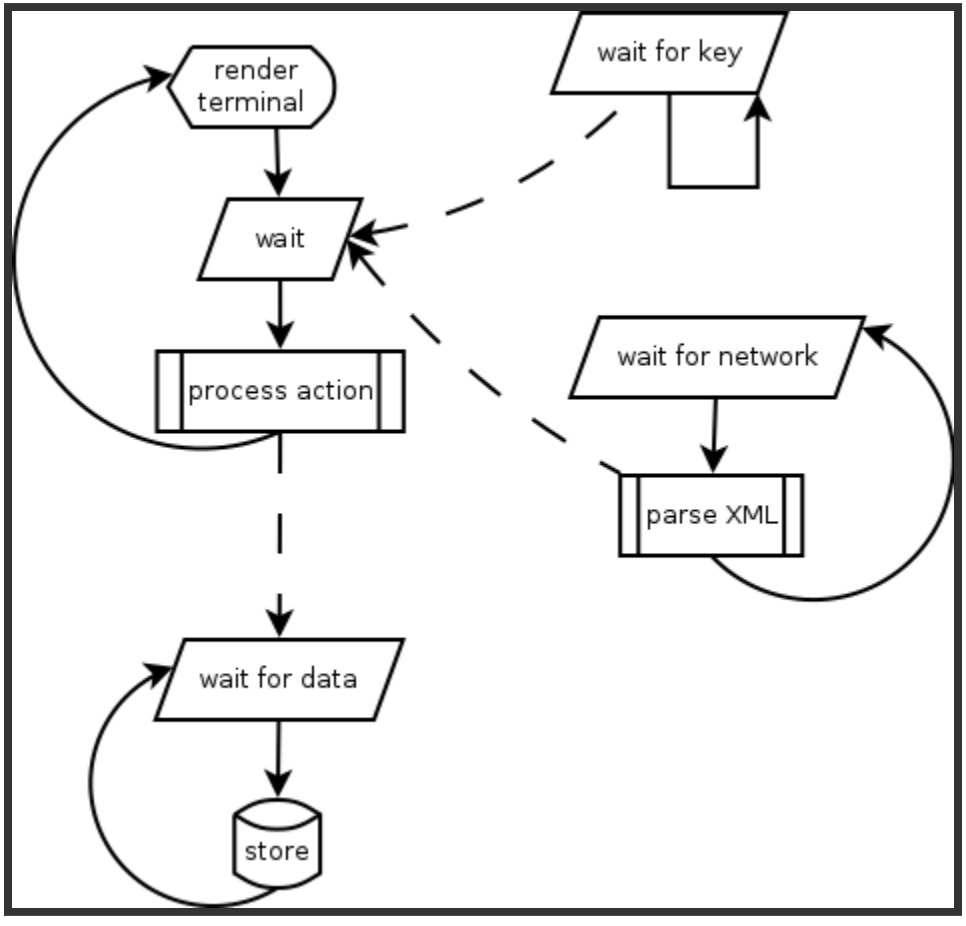
USER INPUT



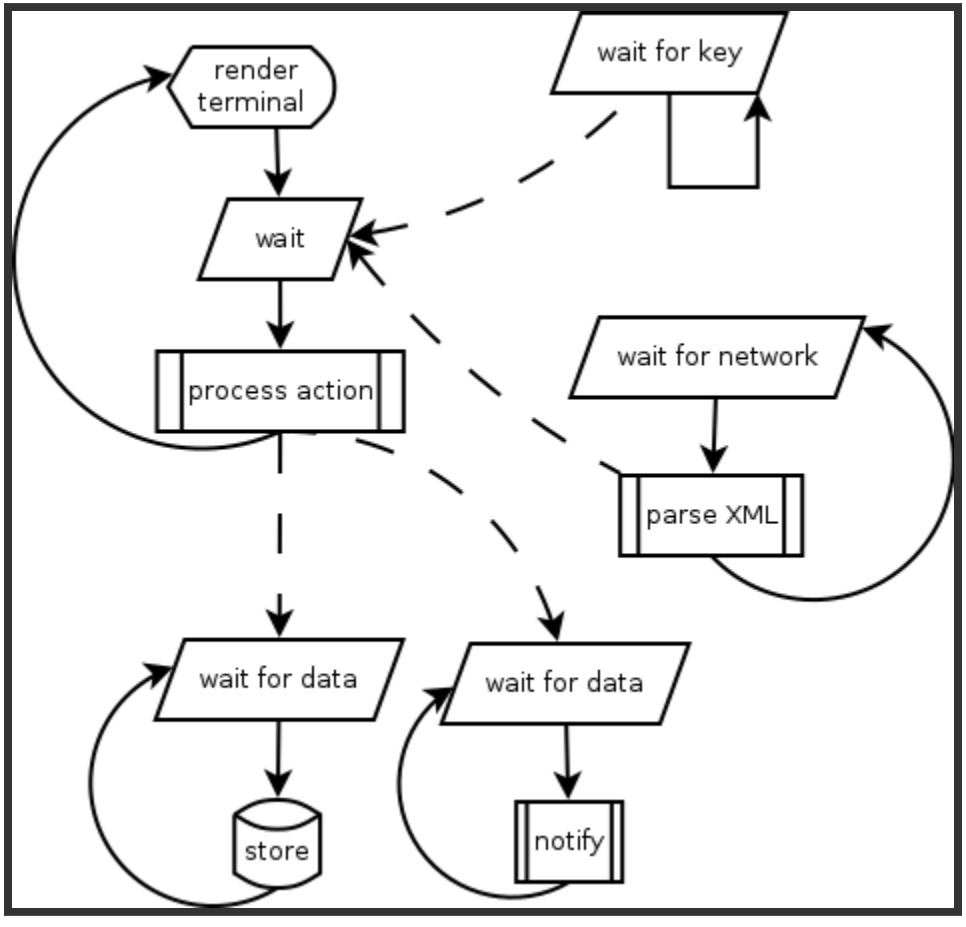
NETWORK INPUT



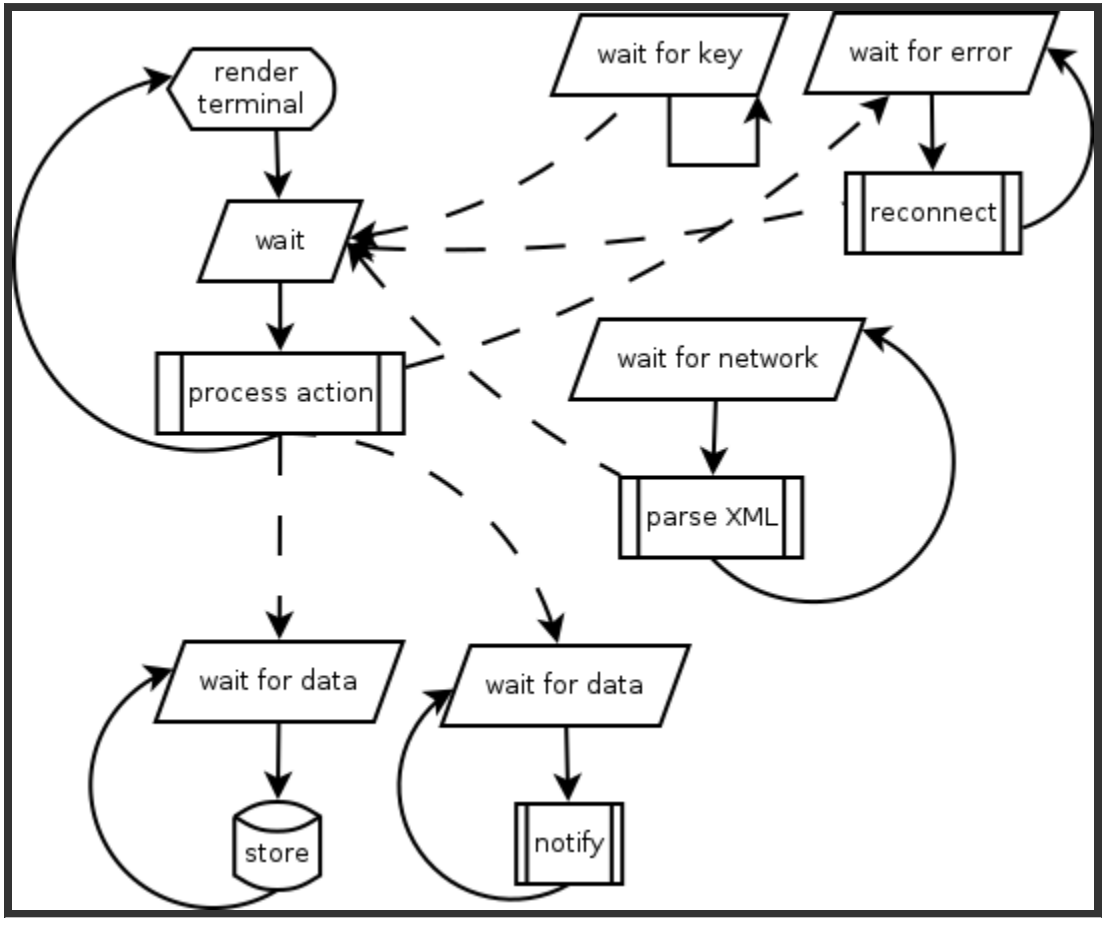
DISK OUTPUT

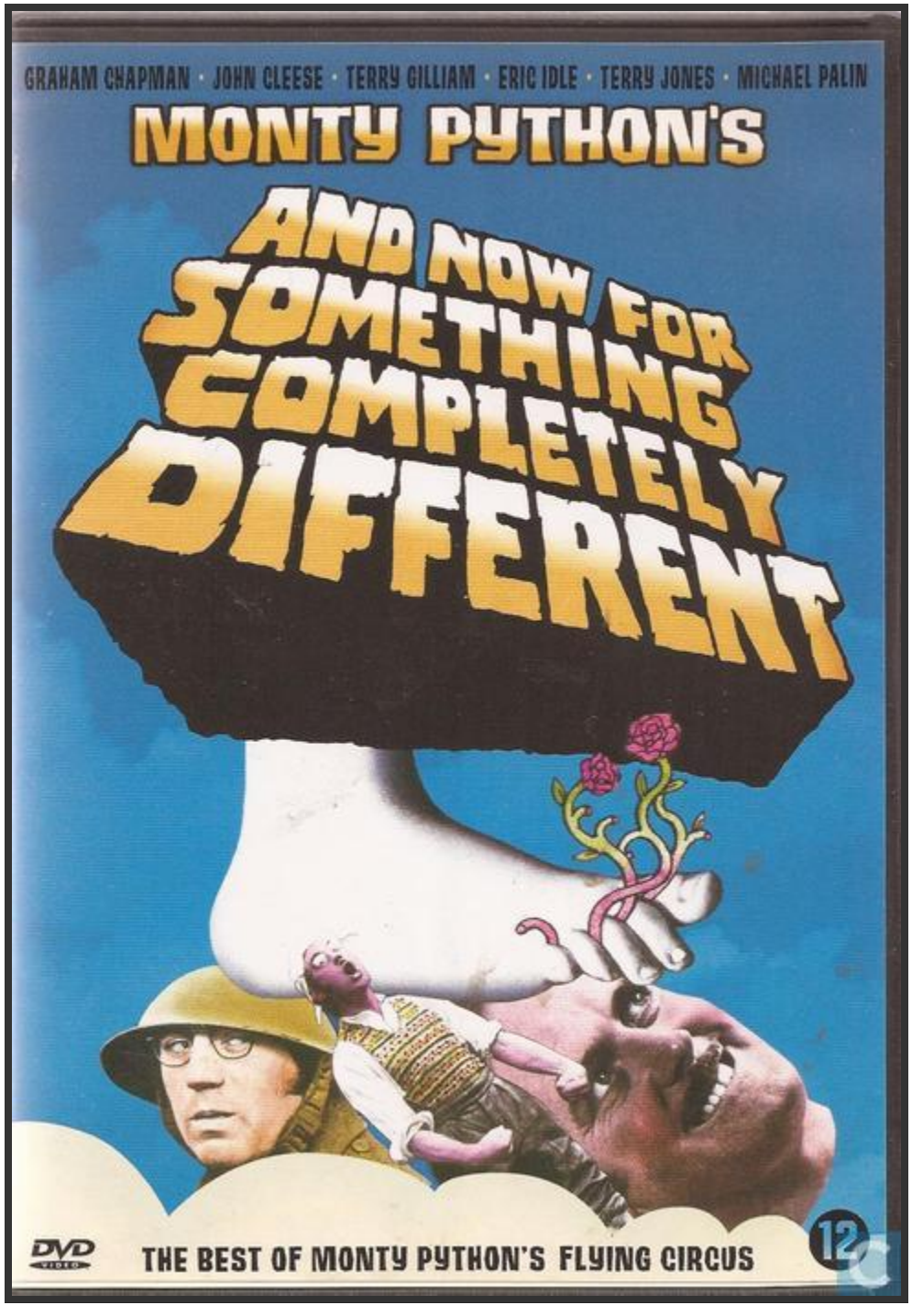

render terminal
wait for key
wait
process action
wait for network
parse XML
wait for data
store

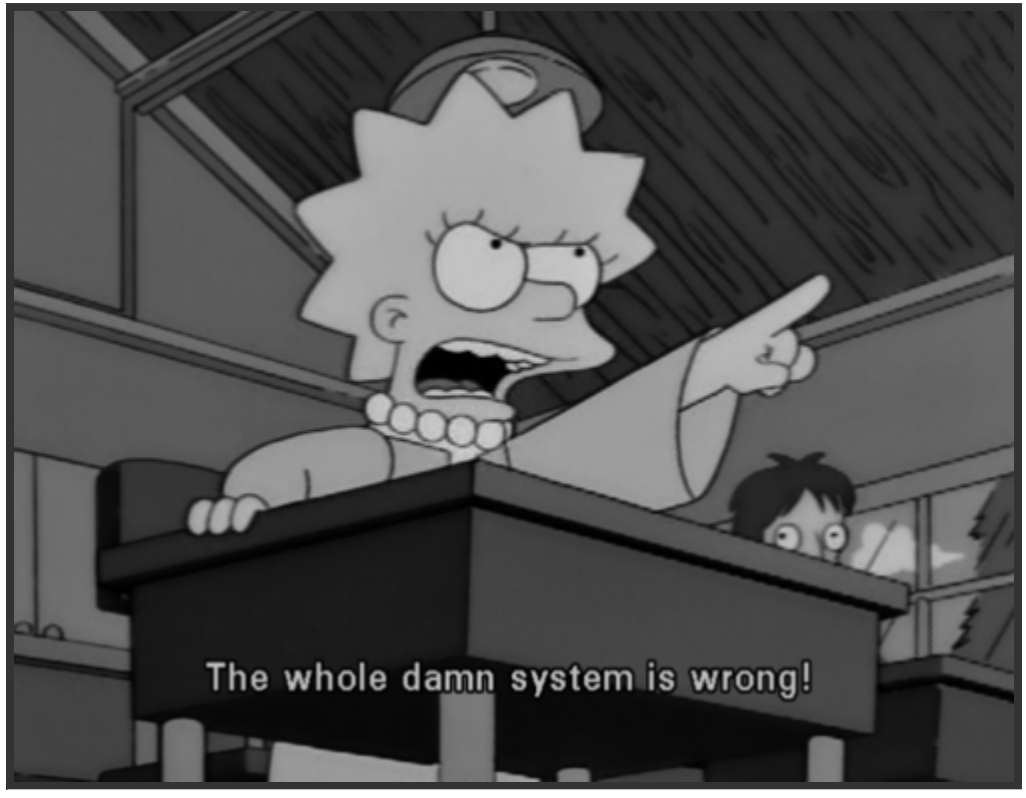
NOTIFICATIONS



NETWORK FAILURES




GRAHAM CHAPMAN · JOHN CLEESE · TERRY GILLIAM · ERIC IDLE · TERRY JONES · MICHAEL PALIN
MONTY PYTHON'S
AND NOW FOR SOMETHING COMPLETELY DIFFERENT
DVD VIDEO
THE BEST OF MONTY PYTHON'S FLYING CIRCUS
12

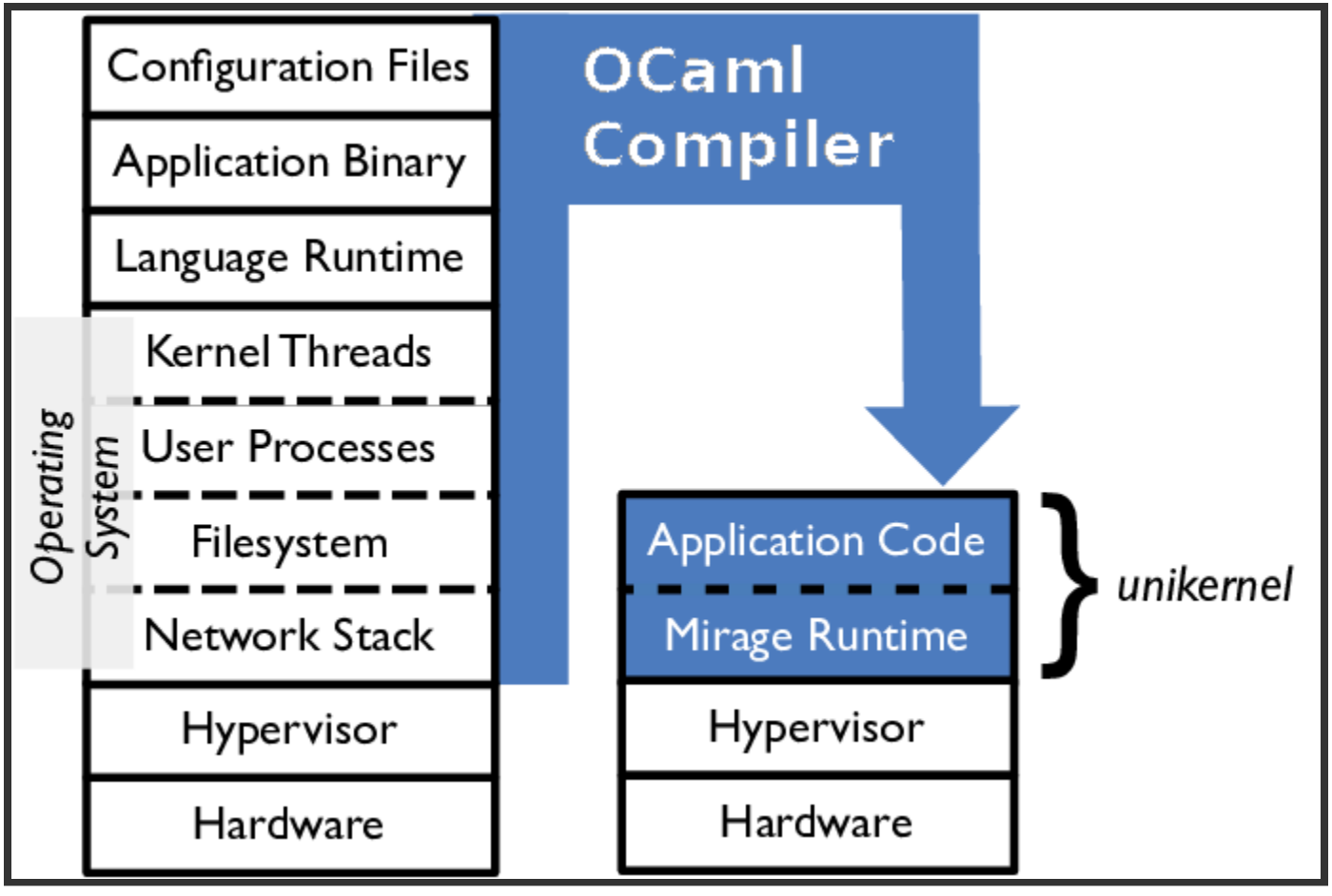

The whole damn system is wrong!


Operating
System
New Fresh Scent

HYPERVISOR

- Isolation and scheduling of virtual machines
- Abstraction from hardware


"Be realistic,"
said the unicorn


Configuration Files
Application Binary
Language Runtime
Kernel Threads
User Processes
Filesystem
Network Stack
Operating System
Hypervisor
Hardware
OCaml Compiler
Application Code
Mirage Runtime
unikernel
Hypervisor
Hardware

MIRAGEOS

- Single purpose operating system
- From the grouns up in OCaml
- No libc
- Developed since 2009 at University of Cambridge

TRANSPORT LAYER SECURITY

- Most widely used security protocol (HTTPS)
- Optional mutual authentication (usual server authentication)
- X.509 encoded certificates (as ASN.1 structures)
- Various implementations, OpenSSL most popular (~20 years)


HEAVY IMPACT
VULNERABILITIES
ON TOUR 2014-2015
HEARTBLEED
SHELLSHOCK
SANDWORM
GHOST
VENOM


POODLE
© Ken Westin 2015
@kwestin

TLS CORE

OCaml helps to enforce state-machine invariants.

```
let handle_handshake ssn hs buf =
  match parse_handshake buf with
  | Error -> fail (`Fatal `ReaderError)
  | Ok handshake ->
     match ssn, handshake with
     | AwaitClientHello, ClientHello ch ->
       answer_client_hello hs ch buf
     | AwaitClientFinished (session, log), Finished fin ->
       answer_client_finished hs session fin buf log
     (* ... *)
     | _ -> fail (`Fatal `UnexpectedHandshake)
```

AUTHENTICATION

- Using certificates, consisting of name, public key, validity, ...
- A chain of certificates is transferred
- Trust anchors distributed with client software

ABSTRACT SYNTAX NOTATION

- Grammar to describe data (key, value)
- Choice, sequence, set; implicit, explicit, optional
- Different encodings (packed, basic, normalised)
- Used in X.509 certificates

ASN.1 (ENCODING OF CERTIFICATES)

```
    TBSCertificate ::= SEQUENCE {
         version           [0] Version,
         serialNumber          CertificateSerialNumber,
         signature             AlgorithmIdentifier,
         issuer                Name,
         validity              Validity,
         subject               Name,
         subjectPKInfo         SubjectPublicKeyInfo,
         issuerUniqueID    [1] IMPLICIT UniqueId OPTIONAL,
         subjectUniqueID   [2] IMPLICIT UniqueId OPTIONAL,
         extensions        [3] Extensions OPTIONAL
}
```

ASN.1 IN OCAML

```
let tbsCertificate   = sequence (
   (opt "version"         (e 0 version))
 @ (req "serialNumber"         certificate_sn)
 @ (req "signature"            Algorithm.identifier)
 @ (req "issuer"               Name.name)
 @ (req "validity"             validity)
 @ (req "subject"              Name.name)
 @ (req "subjectPKInfo"        PK.pk_info_der)
 @ (opt "issuerUID"       (i 1 uniqueId))
 @ (opt "subjectUID"      (i 2 uniqueId))
-@ (opt "extensions"      (e 3 Extension.extensions_der))
)
```

X.509

```
let is_server_cert_valid host time cert =
  match
    validate_time time cert,
    maybe_validate_hostname cert host,
    version_matches_extensions cert,
    validate_server_extensions cert
  with
  | (true, true, true, true) -> success
  | (false, _, _, _) -> fail `CertificateExpired
  | (_, false, _, _) -> fail `InvalidServerName
  | (_, _, false, _) -> fail `InvalidVersion
  | (_, _, _, false) -> fail `InvalidServerExtensions
```

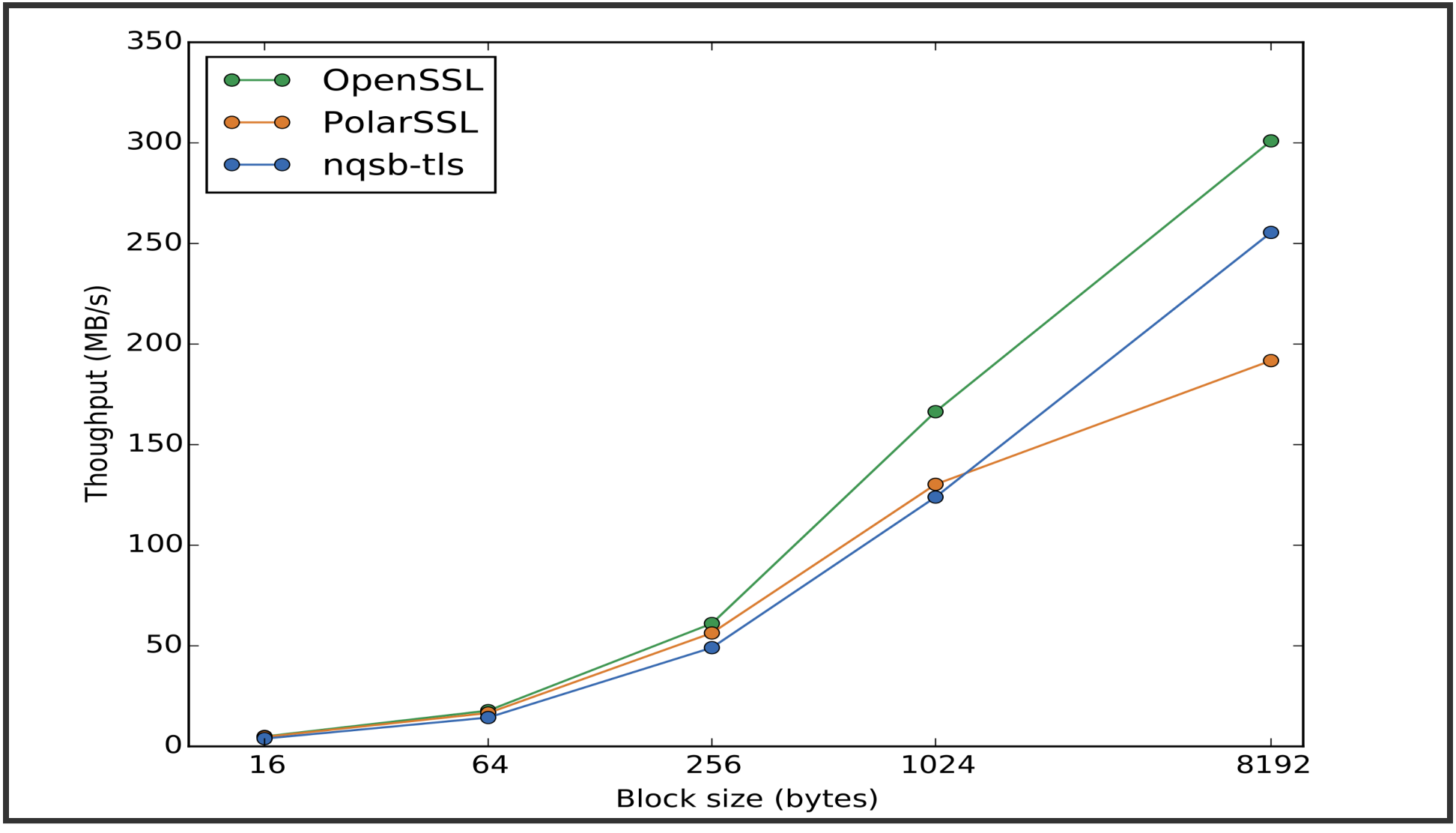
CRYPTOGRAPHY

- Cipher and hash cores in simple C code
- Cipher modes (CTR, CBC, GCM, CCM) in OCaml
- Public-key cryptography in OCaml using GMP
- Entropy / RNG

HANDSHAKE PERFORMANCE

	OCaml-TLS	OpenSSL	PolarSSL
RSA	698 hs/s	723 hs/s	672 hs/s
DHE-RSA	601 hs/s	515 hs/s	367 hs/s

THROUGHPUT


OpenSSL
PolarSSL
nqsb-tls
Thoughput (MB/s)
350
300
250
200
150
100
50
0
16
64
256
1024
8192
Block size (bytes)

TRUSTED COMPUTING BASE

A flaw in any part jeopardizes the security of the entire system!

Subsystem	Linux/OpenSSL	MirageOS
Kernel	1600	48
Runtime	689	25
Crypto	230	23
TLS	41	6
Total	**2560**	**102**

(numbers in **kloc**)

CONCLUSION

- Jackline, standalone functional instant messaging
- Small TCB, reasonable performance
- Program code is communication between human beings
- BSD licensed
- Avoids common flaws (memory safety, type safety)
- Next step telnet server
- Jackline as a unikernel

https://nqsb.io