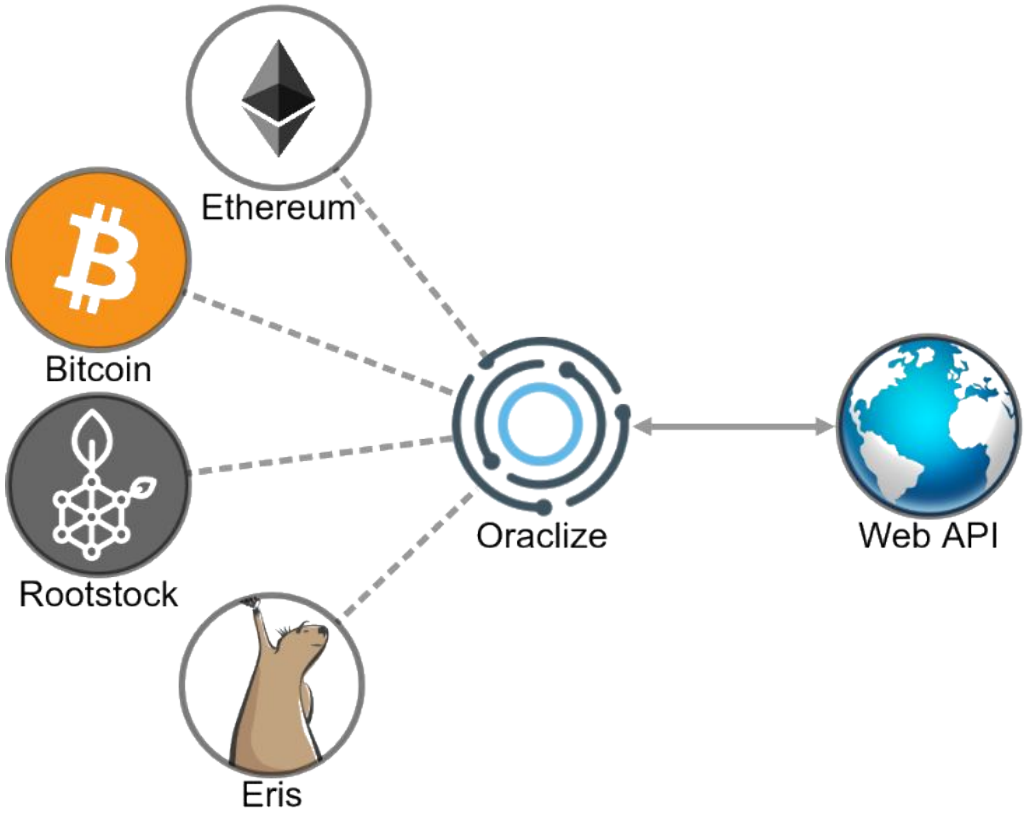
# oraclize

The “state of the art” oracle service.

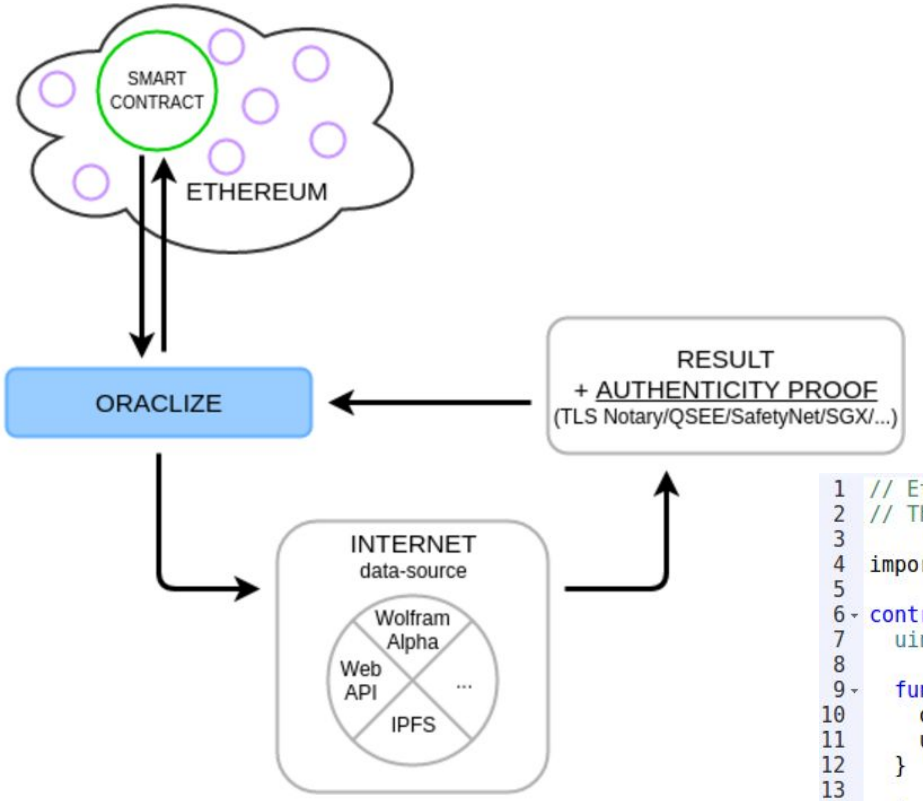
Designing the safest data-transport-layer you can possibly get.

Thomas Bertani


Ethereum
Bitcoin
Rootstock
Eris
Oraclize
Web API

# The way **NOT** to do it

- asking datasources to run blockchain adapters
- asking datasources to signal blockchain support via dedicated schemas
- suggesting dapp developers to run their own oracle
- trusting the provider of the oracle service that he is doing a good job
- asking the oracle to provide quality (??) data




- 180k+ txs sent to the public Ethereum network since Sept 2015 (1.5% of all txs ever)
- most widely used oracle service
- on-demand data = blockchain is not being spammed!

```solidity
// Ethereum + Solidity
// This code sample & more @ dev.oraclize.it

import "github.com/oraclize/ethereum-api/oraclizeAPI.sol";

contract PriceFeed is usingOraclize {
  uint public ETHUSD;

  function PriceFeed(){
    oraclize_setProof(proofType_TLSNotary | proofStorage_IPFS);
    update(0); // first check at contract creation
  }

  function __callback(bytes32 myid, string result, bytes proof) {
    if (msg.sender != oraclize_cbAddress()) throw;
    ETHUSD = parseInt(result, 2); // save it as $ cents
    // do something with ETHUSD
    //update(60); //recursive update disabled
  }

  function update(uint delay){
    oraclize_query(delay, "URL",
      "json(https://min-api.cryptocompare.com/data/price?fsym=ETH&tsyms=USD).USD");
  }
}
```

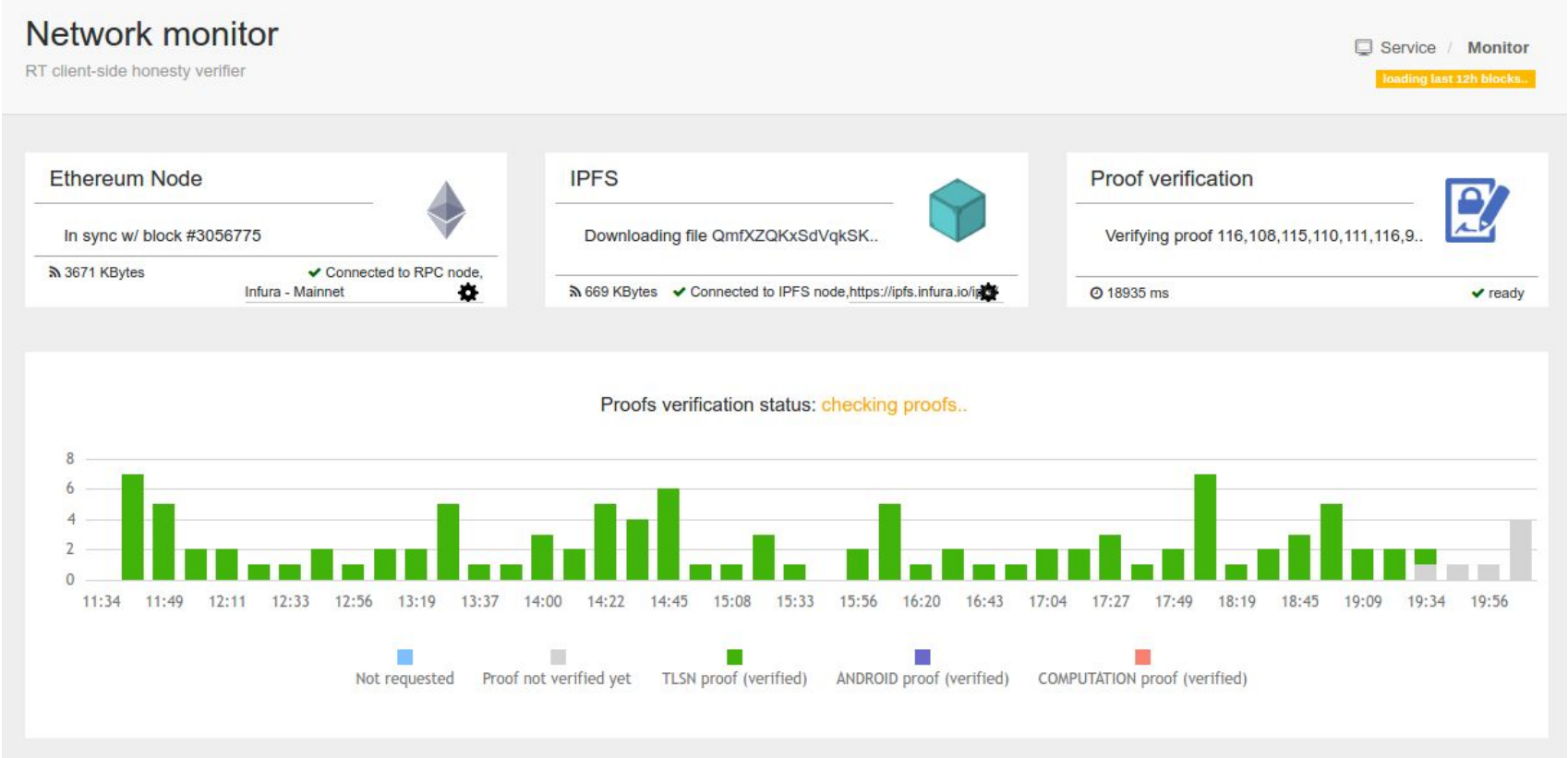
## Transaction Information

| | |
|---|---|
| TxHash: | 0xaabe1db1367df253bdeb193bba0b8af83de6e17831989ebaee4c9d43e3f7479e |
| Block Height: | 2195250 (4839 block confirmations) |
| TimeStamp : | 19 hrs 12 mins ago (Sep-04-2016 04:18:03 AM +UTC) |
| From: | 0x26588a9301b0428d95e6fc3a5024fce8bec12d51 (Oraclize) |
| To: | Contract 0x4b92a948ced9d457b4655abf62ed930a090f8566<br>TRANSFER 1 wei to → 0x7af8513d30bd6ad670cee..<br>TRANSFER 0.001073577549271636 Ether to → 0xa1b5f95be71ffa2f86adefc... |
| Value: | 0 Ether ($0.00) |
| Gas: | 150000 |
| Gas Price: | 0.00000002 Ether |
| Gas Used By Transaction: | 92655 |
| Actual Tx Cost/Fee: | 0.0018531 Ether ($0.02) |
| Cumulative Gas Used: | 665593 |
| Nonce: | 45266 |
| Input Data: | Function: __callback(bytes32 myid, string result, bytes proof)<br><br>MethodID: 0x38bbfa50<br>[0]:e411148f36be501404f4fdfcaae5c6a7eac18c6f127074ffaa1e582a50d1295d<br>[1]:0000000000000000000000000000000000000000000000000000000000000060<br>[2]:00000000000000000000000000000000000000000000000000000000000000a0<br>[3]:0000000000000000000000000000000000000000000000000000000000000004<br>[4]:3332373400000000000000000000000000000000000000000000000000000000 |

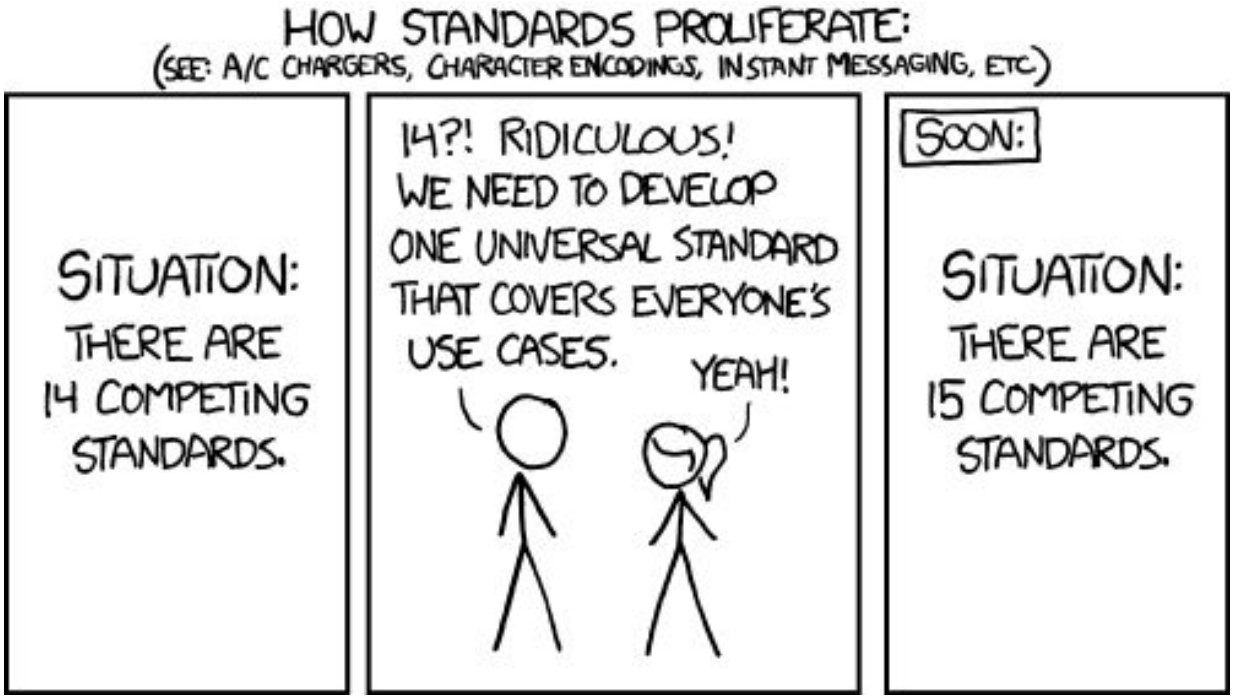
Authenticity proofs: (mostly) Trusted Computing Techniques to prove.. the **authenticity** of data!

In the last 2 years.. we have designed ad-hoc security techniques based on:
Qualcomm TEE, Samsung Knox, Google SafetyNet, AWS sandbox, Intel SGX (?), ..

<u>In general, nothing can be said around the quality of data.</u>




https://github.com/oraclize/proof-verification-tool




Please please please, let's stop proposing new standards to solve blockchain-specific problems only - most of them have a broader scope.

The datasources **can** provide verifiable, native, simple proofs to guarantee the authenticity of data, and eventually will.

https://tools.ietf.org/html/draft-cavage-http-signatures-06

Versions: 00 01 02 03 04 05 06

Network Working Group — M. Cavage
Internet-Draft — Oracle
Intended status: Standards Track — M. Sporny
Expires: July 14, 2017 — Digital Bazaar
January 10, 2017

**Signing HTTP Messages**
**draft-cavage-http-signatures-06**

Such standard proposal is:

- being discussed since 2013
- elegant, simple and secure solution for native authenticity verification
- supported by different datasources already
- “blockchain” ofc is not being mentioned not even once

Thanks to the way Oraclize is designed, setting the proofType to *proofType_native* is enough

```
12  function MyConstructor() {
13      oraclize_setProof(proofType_native);
14      oraclize_query("URL", "https://api.whatever.com/..");
15  }
16
17  function __callback(bytes32 myid, string result, bytes proof) {
18      if (msg.sender != oraclize_cbAddress()) throw;
19      oraclize_verifyNativeProof(proof);
20      //..
21  }
```

What did we just do here?

- native authenticity of data guarantee (the strongest you can possibly get)
- still, no blockchain adapter was needed on the datasource side
- Oraclize works with *any* datasource, providing the strongest proof available (as for user request)

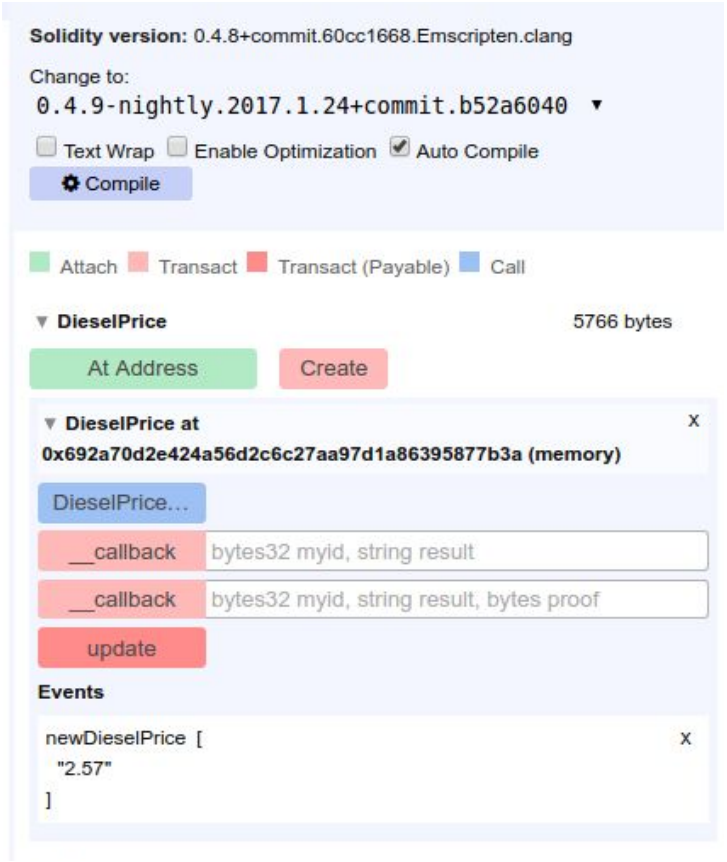
# In a nutshell..

| |
|---|
| fetching authentic data from **any** web API (temperature in London, flights info, random number, ..) |
| triggering events via web API calls (send email, open a lock, ..) |
| scheduling transactions for a future time (call that smart contract method in 1 hour) |
| delegating some code to an offchain auditable context (execute complex ops and give me back the result) |
| getting access to tons of datasources (IPFS/SWARM, Amazon Mechanical Turk, ..) |

No need to install anything (unless you want to),
works on the Ethereum mainnet, public testnet (Ropsten)..

.. and on any private instance (including testprc/truffle/ethercamp).

And even on **browser-solidity**!



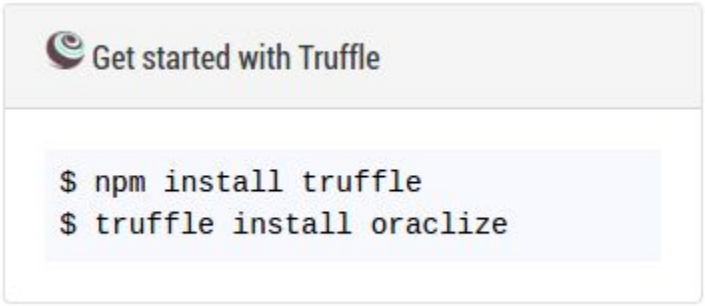

https://goo.gl/Etv2MT

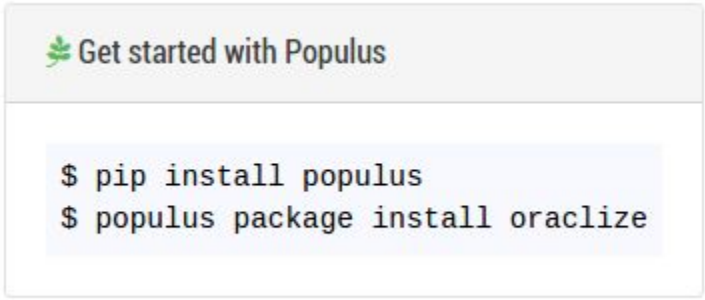
## The Ethereum Package Management

EIP190 package registry for Ethereum smart contract packages

https://www.ethpm.com

Oraclize is already available there as a package!



Questions?

Join the **#ethereum-api**
channel on Gitter

Or come visit us at our HQ in London!

Code examples:
http://github.com/oraclize/ethereum-examples