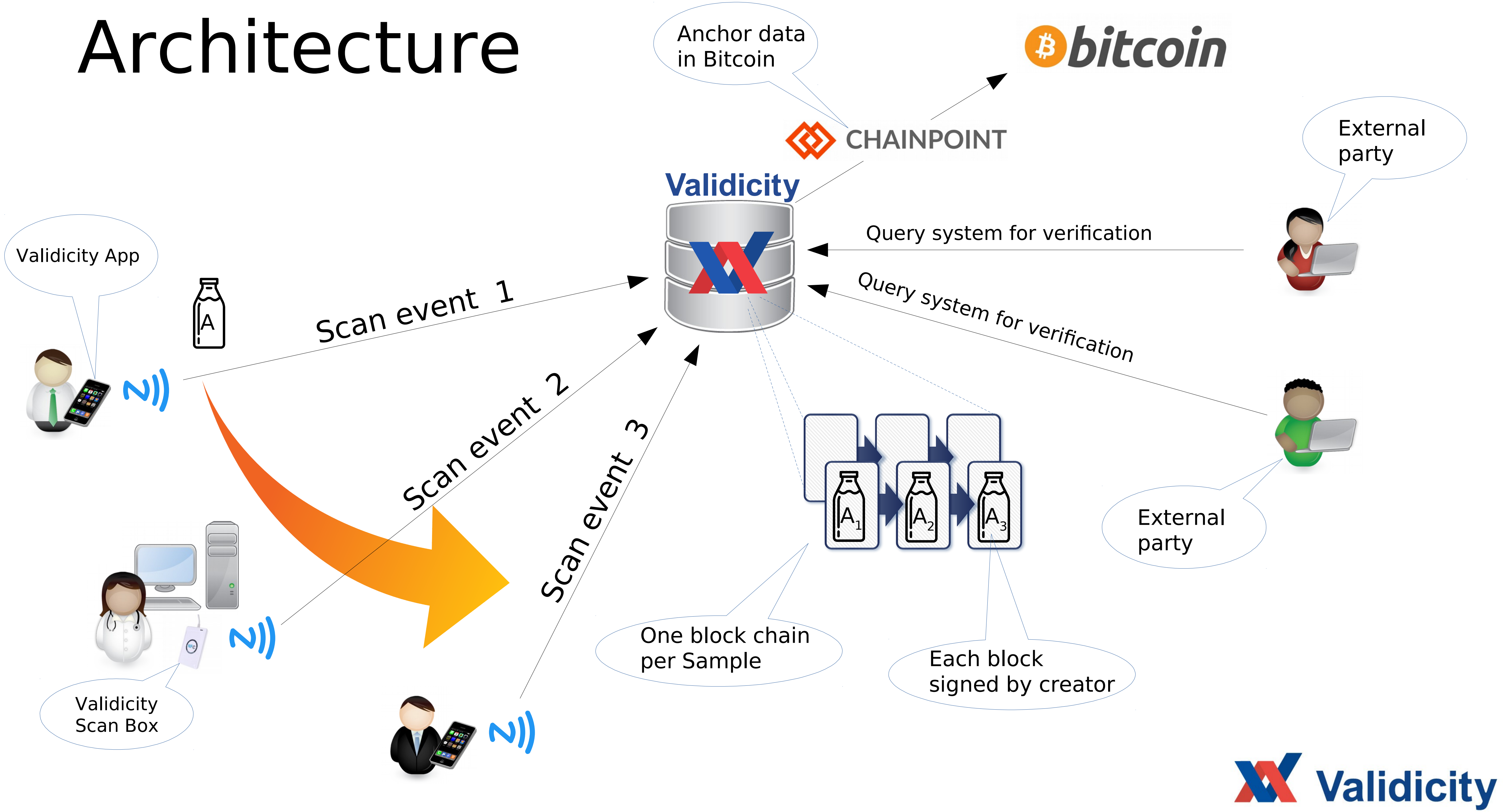
Validicity System

Secure chain of custody using block chain technology

Properties

- Sample handling using **NFC** scanning of RFID tag
- Fully immutable Sample data trail in block chains
- Independently **provable data correctness** with timestamps
- Easy to deploy, manage and integrate with other systems
- Easy for external parties to query and verify data trails

Architecture



Validicity System Components

- Validicity Server ✓
- Validicity Tool ✓
- Validicity Client ✓
- Validicity Scan Box ✓
- Validicity App ✗
- Validicity Chainlink ✗
- Validicity Query ✗

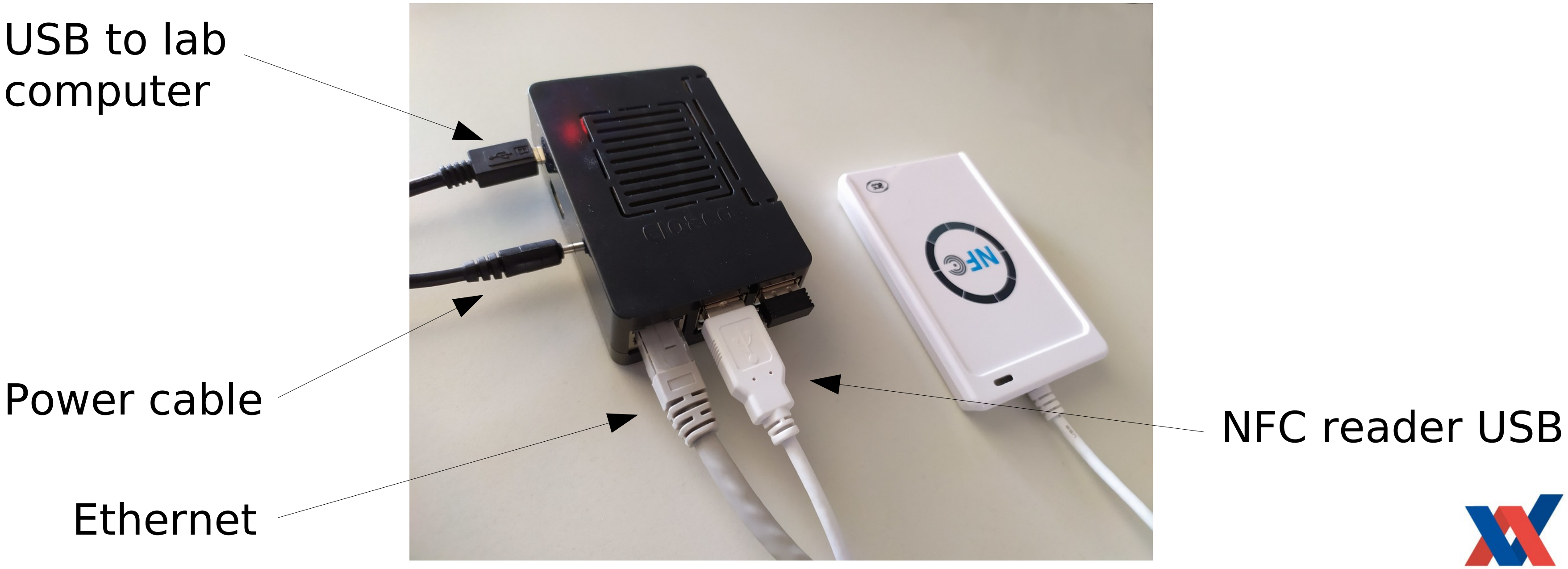
Creating the Sample trail

- Samples are scanned using NFC with the Validicity App or Scan Box
- The scan event creates a new Sample **block** of data
- Each Sample has its own **block chain** representing the custody log
- The Validicity Server stores the block chain for the Sample

Validicity

Validicity Scan Box

- The scan box is an ODROID C2 embedded Linux device with an attached ACR122U NFC reader
- The scan box emulates a keyboard and is attached using a USB cable to a lab computer



USB to lab computer

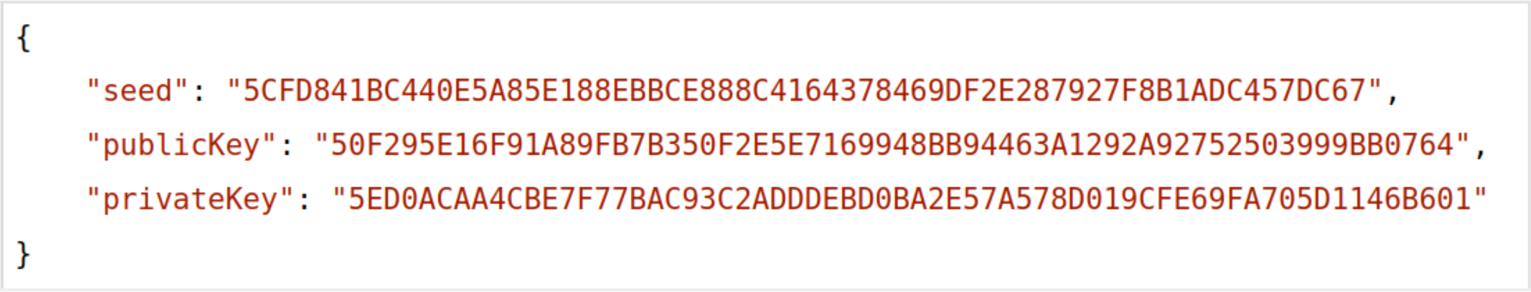
Power cable

Ethernet

NFC reader USB

Validicity Client

- The client software for the Scan Box is written in Dart
- The service scans continuously for RFID tags
- When a new Client is added to a system we need to:
  1. On server: Create an account in with OAuth2 credentials
  2. On box: Enter credentials in the `validicity.yaml` config file
  3. On box: Create keys using the `createkeys` command
  4. On box: Register public key in server using the `register` command



```
{
    "seed": "5CFD841BC440E5A85E188EBBCE888C4164378469DF2E287927F8B1ADC457DC67",
    "publicKey": "50F295E16F91A89FB7B350F2E5E7169948BB94463A1292A92752503999BB0764",
    "privateKey": "5ED0ACAA4CBE7F77BAC93C2ADDDEBD0BA2E57A578D019CFE69FA705D1146B601"
}
```

Scanning a Sample

When the scan box scans an RFID tag it has not scanned in the last 5 seconds it performs the following in a fraction of a second:

1. Queries the Validicity Server API for the last block for this ID
2. Creates a new block including hash of the last block
3. Computes a hash of all pertinent fields in the block
4. Creates a signature for this hash using the private key
5. Submits this new block to the Validicity Server API
6. If all goes well, “types” the ID on the lab computer

Validicity

Anchoring the Sample trail

- The blocks of data constitutes an immutable chain

  … but could still be artificially constructed “in retrospect”!
- This is why the data is also, at regular intervals, “anchored” to a **publically secured** block chain – the Bitcoin network
- Anchoring is performed using Chainpoint, an open standard
- An anchor creates a “timestamp proof” of data

External Party Verification

- An external party can verify correctness using the Validicity API
- This is done using an Open Source query tool that can easily be installed and used by any party given access to the API
- The fact that the source code of the query tool is Open Source guarantees that it does indeed work as claimed

Technology

- Server runs on Linux easily deployable in any server or cloud
- System is written in Dart and uses industry standard PostgreSQL
- The API is a standard REST/JSON API with OAuth2 authentication
- Scan Box runs embedded Linux with software in Dart

Block chain technology

NANO

- All cryptographical code and algorithms in Validicity is the same as used by the Nano crypto currency
- Nano has to date 50 million blocks in its public ledger and zero incidents while being in operation in 5 years
- Nano is the fastest crypto currency to date
- Using the same algorithms and code is a deliberate choice to ensure safety and reliability

Links

- https://nano.org
- https://chainpoint.org
- https://www.postgresql.org
- https://dart.dev