arm

Newcastle University
School of Computing

Security Analysis of IETF ACE-OAuth Protocol

(Luca Arnaboldi & Hannes Tschofenig)

Smart Factory

Smart Home

Smart City

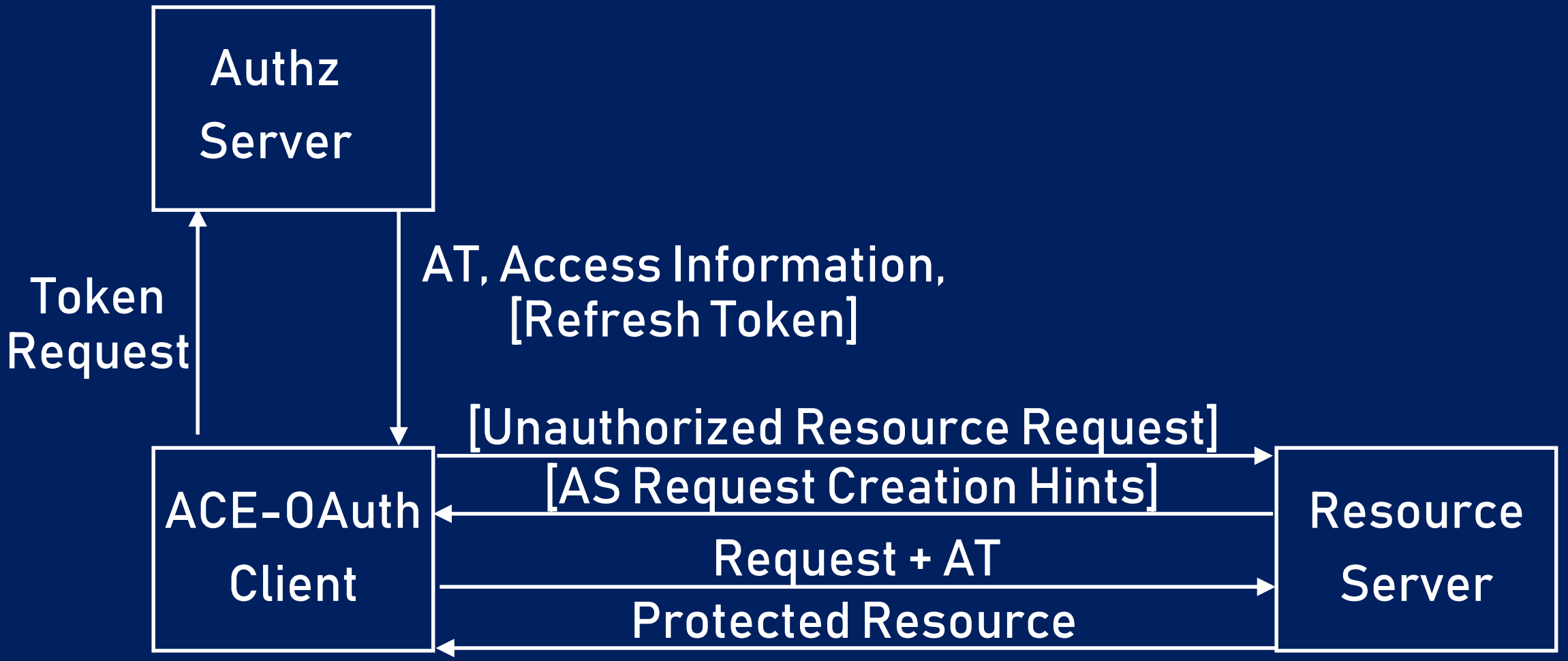
OAuth & IoT

- IoT describes a broad range of deployment scenarios and constraints.
- Authorization problems also surface in IoT environments.
- Why not re-use OAuth?
  - For some deployments OAuth, as is, is perfectly fine.
  - For devices that either lack a browser to perform a user-agent based authorization, or are input-constrained the device flow has been defined.
  - ACE-OAuth has been defined for constrained devices.

ACE-OAuth: What is different?

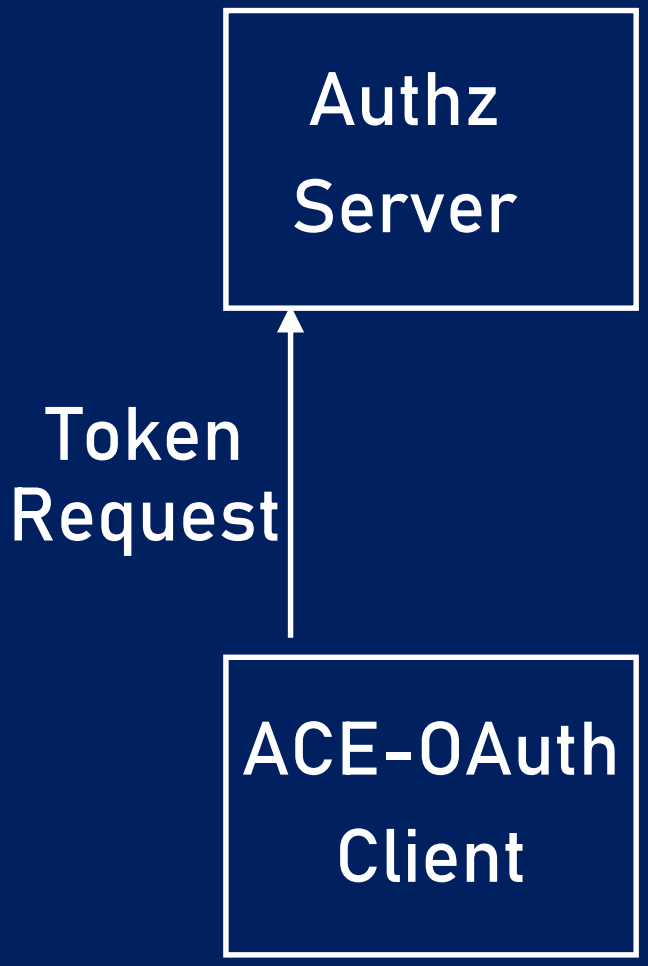
- CoAP instead of HTTP for some communication
- Proof-of-possession tokens only
- CBOR Web Token (CWT) instead of JWT
- Token Introspection over CoAP
- Additional claims (exi) and parameters (rs_cnf, profile)
- AS discovery procedure
- Additional security options:
  - DTLS instead of TLS
  - OSCORE as alternative for (D)TLS
  - Use of PSKs, raw public keys, and certificates

ACE-OAuth: Quick Overview



ACE-OAuth:
Setup Certificate Example

ACE-OAuth: Token Request



Token Request:

- grant_type
- audience
- req_cnf

Resource Server

ACE-OAuth: Token Response

Token Response:
- access_token
- rs_cnf
- cnf
- profile
- refresh_token
- token_type
- scope
- state
- …

ACE-OAuth: RS Request

Authz Server

Details vary depending on the profile being used.

Examples:

- draft-ietf-ace-dtls-authorize-07
- draft-ietf-ace-oscore-profile-07

ACE-OAuth: RS Response

Authz Server

Details vary with application semantics

Objective of Formal Model

Given these incredibly complex systems, Is it possible to generate a model to test possible extensions to this very flexible protocol?

Idea:

We can make use of modular protocol design to plug in different implementations. This would allow us to keep the same core protocol model and only alter it slightly to test various environments!

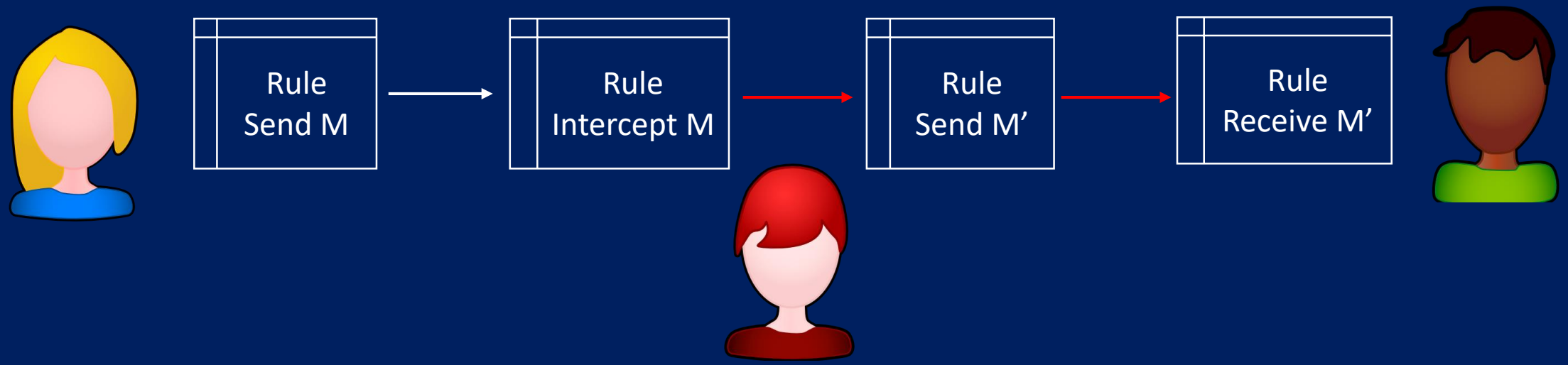
How is this achieved?

Choices for Formalization

- Initial phase : formalising the security properties
  - Outlined the key pieces of information that needed to be secured
- Second Phase: Outlined objectives for each exchange
  - Ensures correctness of flow
- Third Phase: Formalised IETF spec requirements
- Most verification efforts model security of specific implementations.
- *We went the opposite direction:*
  - More flexible
  - Can be easily extended

Quick Tamarin Overview

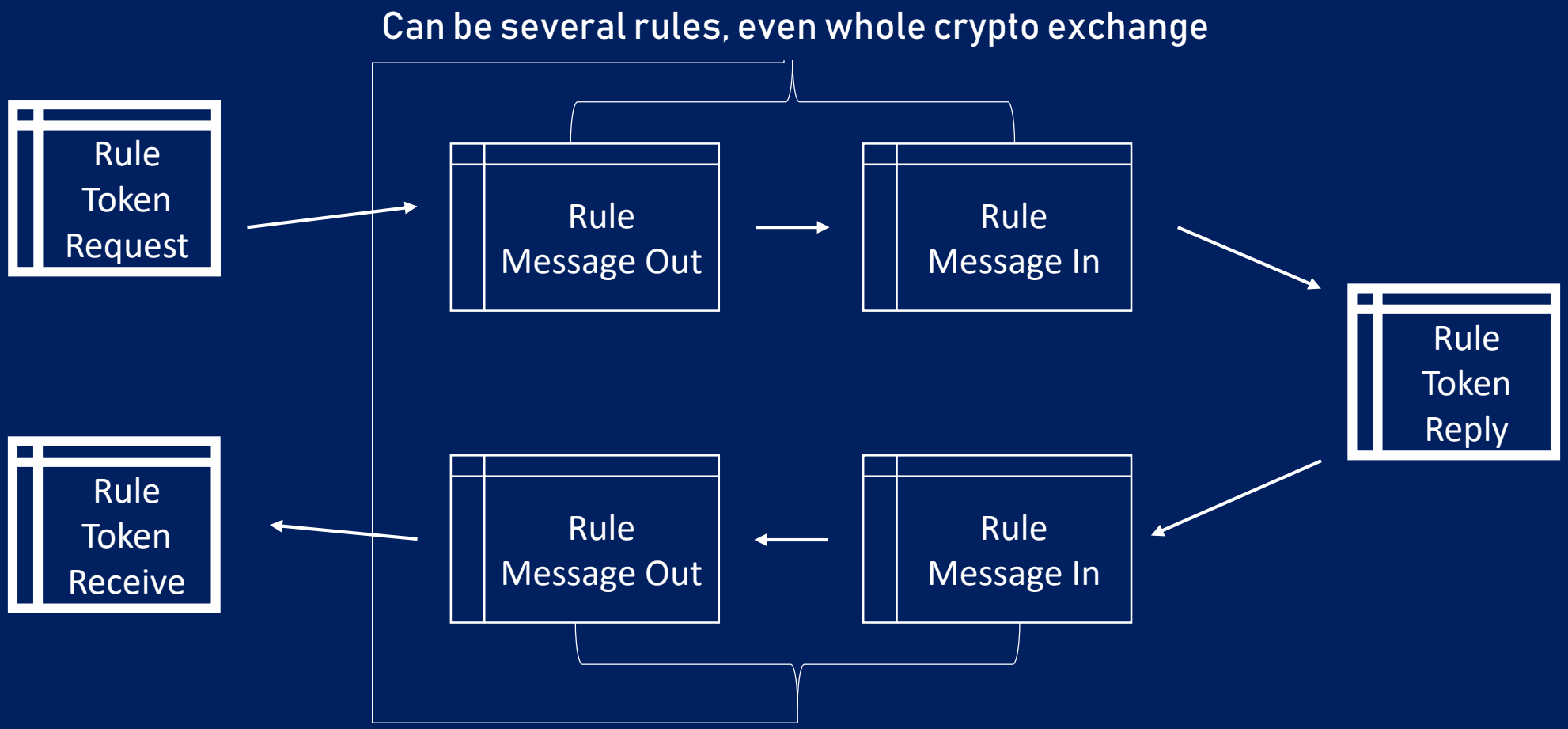
- Writes protocol as a multi set of rules
- Doesn't follow tradition protocol flow of clearly separated parties
- Rules are triggered based on facts in the state
- Attacker can use rules to break the protocol



Protocol Skeleton

- Modelled the framework as FIXED core rules
  - Lemmas (for proofs) are structured around these
  - So these aspects do not need to be touched
- Modelled exchange of messages as customizable rules
  - These second set of rules is up for grabs and can represent various values, cryptographic solutions etc.
- User can only needs to modify part of the protocol (and keep the rest) as is.
- This allows for a user to test setups and press play to get quick results

Extendable rules - Example



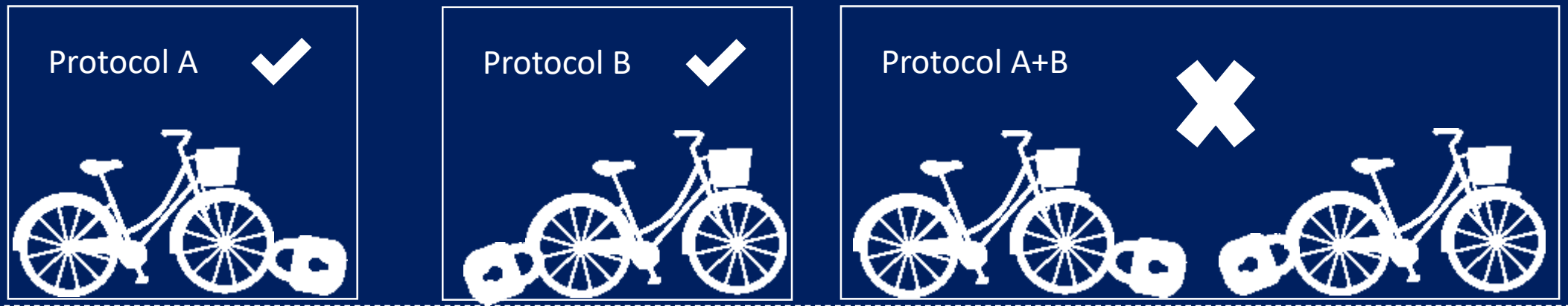
The issue with composability

IETF drafts are by design flexible and allow for a wide range of different implementations (which is great), However:

If security protocol X guarantees properties in scenario A,

DOES NOT MEAN that if used in scenario B same results hold!

Example Bike Shop Security Protocols:



Case Study – Smart Factory

Approved technician

Grey market technician

Emergency services

Factory HVAC and lighting equipment

Case study – Smart Factory:

- Smart factory
  - Asymmetric Cryptography
  - Trust anchor of AS on both Client and Resource Server
  - This token is digitally signed by the AS and includes the proof-of-possession key
  - Proof of Possession between C and RS
  - Communication over BLE and signed
  - No token introspection
  - Need to verify the token is from the right AS

Smart Factory

Results

- We were able to test and verify the security of our setup!
- Found issues we did not expect
  - Initial setup using nonce was implemented wrongly
  - Found timestamps to be more optima
- The implementation really helped design our protocol
- Decisions can easily be backed up with evidence
- Made discussions more concrete and helped visualise the protocol

Conclusion

- The IoT has lots of different scenarios
- Allowing for different extensions is therefore desirable
- Extensions can cause serious security flaws

So we need a way to check this:

- Our approach:
  - Give a generalised model to start with
  - Formalise security goals so that it can be easily verified
  - Add your own extensions
  - Allow the tools to do the hard work!
- Makes everyone's life easier and discussions shorter (hopefully)
- *This is a work in progress , but results are looking promising.*

Model Available At:

https://github.com/Yiergot/ACE-OAuth-FormalModel