OAuth Security Workshop 2017

NTT Group

Asserting Access Tokens from the Transport Layer

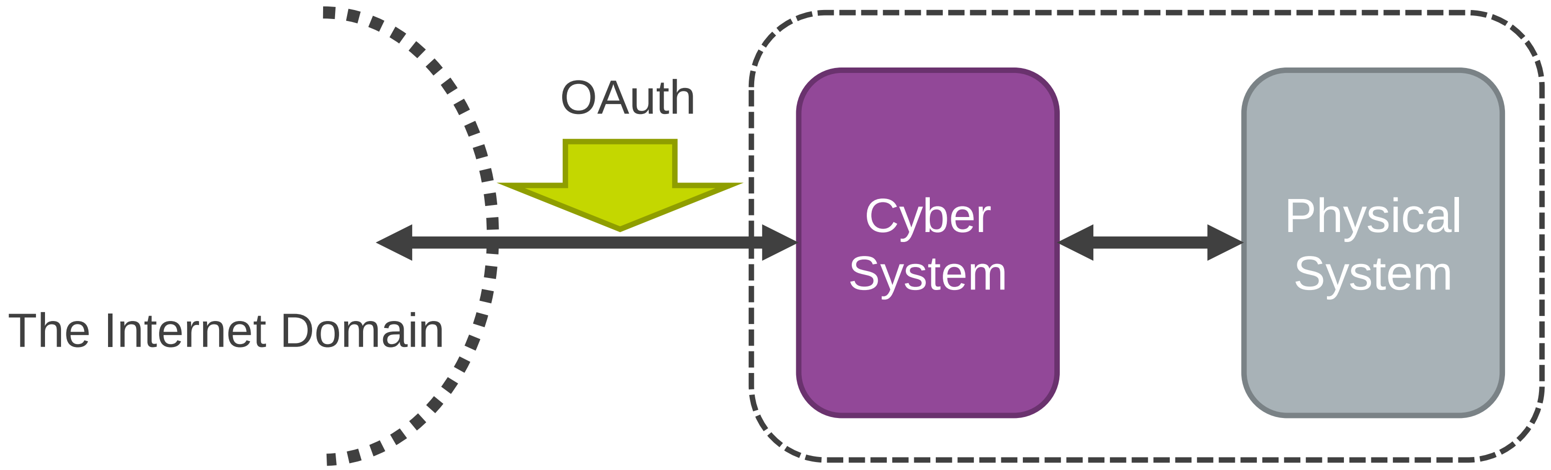
Go Yamamoto, Richard Boyer, Kenji Takahashi (NTT),
Nat Sakimura (NRI)

OVERVIEW

- Owners of critical Cyber-Physical systems may require stronger security model for authorization mechanisms than the current OAuth implementations offer.
- Reliability and resilience will be required when applications assumedly have vulnerability or when operators mistake.
- We propose discussion on delivery methods for access tokens from the transport layer.
- A proof of concept is shown that focuses on simplicity, and compatibility with existing OAuth infrastructure.

OAUTH FOR INDUSTRY

- The owner of a Physical System would like to add a set of REST APIs so that one can monitor/control the Physical System from the Internet domain.
- The owner would like to authorize the access using the OAuth framework.

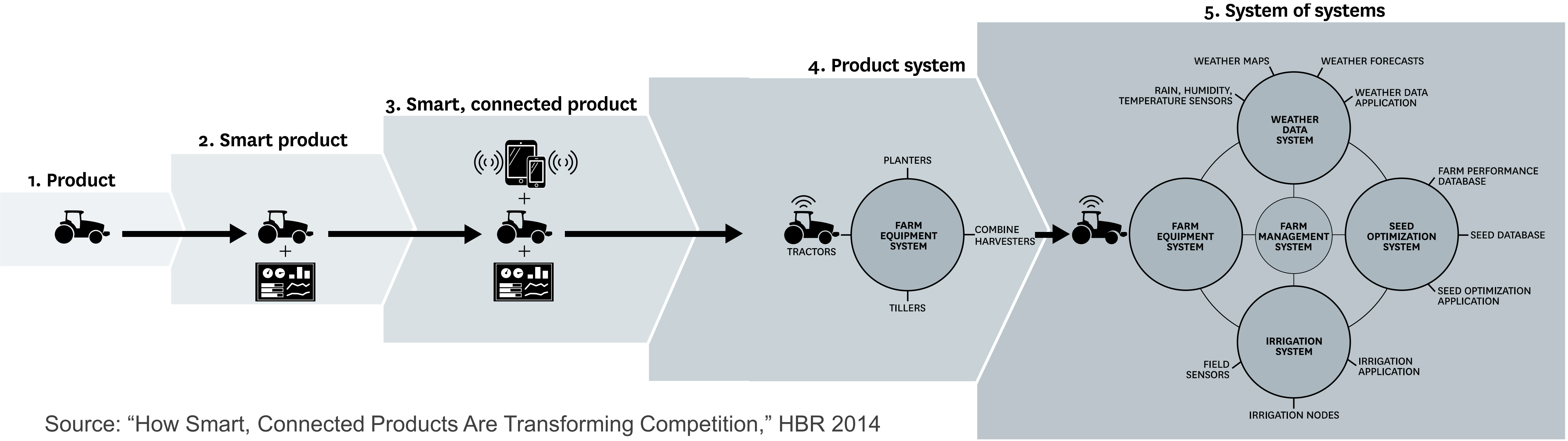


OAUTH FOR INDUSTRY

- Are the bearer tokens on HTTP header acceptable for the owner who has concern on connecting Physical Systems?
- Probably no. Why do we feel so?

THE SYSTEM OF SYSTEMS

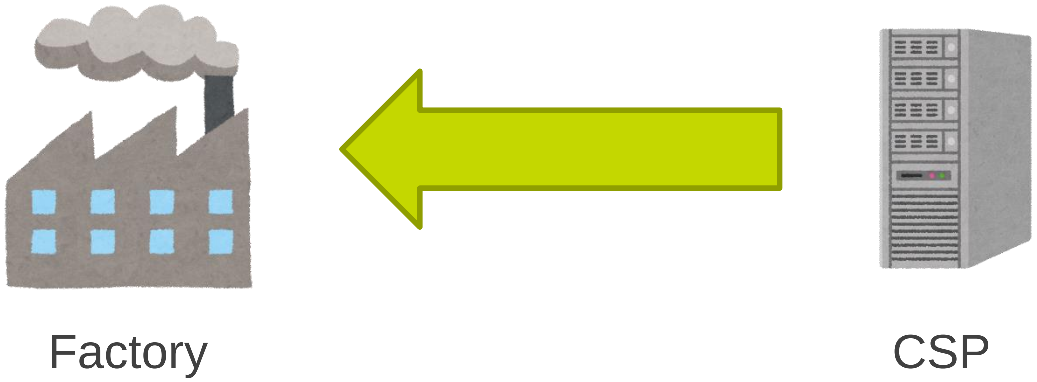
- Systems obtain more advanced competence by connecting together.
- All the surviving systems will be connected at the end.



Source: “How Smart, Connected Products Are Transforming Competition,” HBR 2014

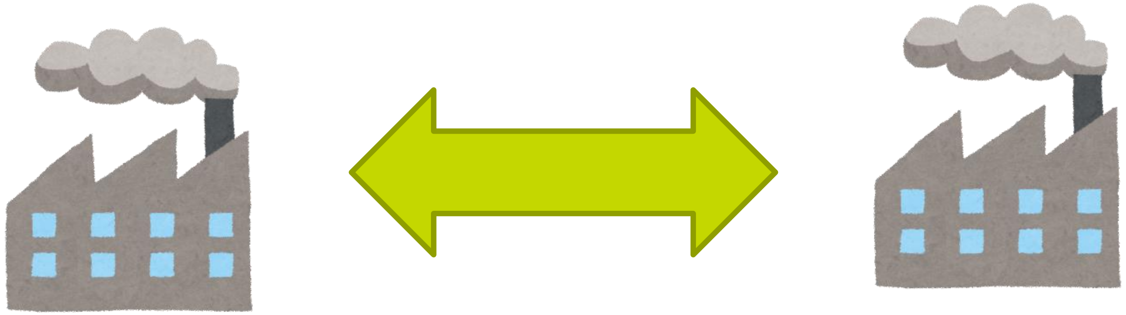
DIFFERENCE IN CONNECTIVITY

- Cloud Service Provider (CSP) provides computing power to Factory.
- Factory connects to CSP for better performance.
- Factory subscribes services from CSP, so the owner of Factory can request CSP to serve under the Factory’s security management.

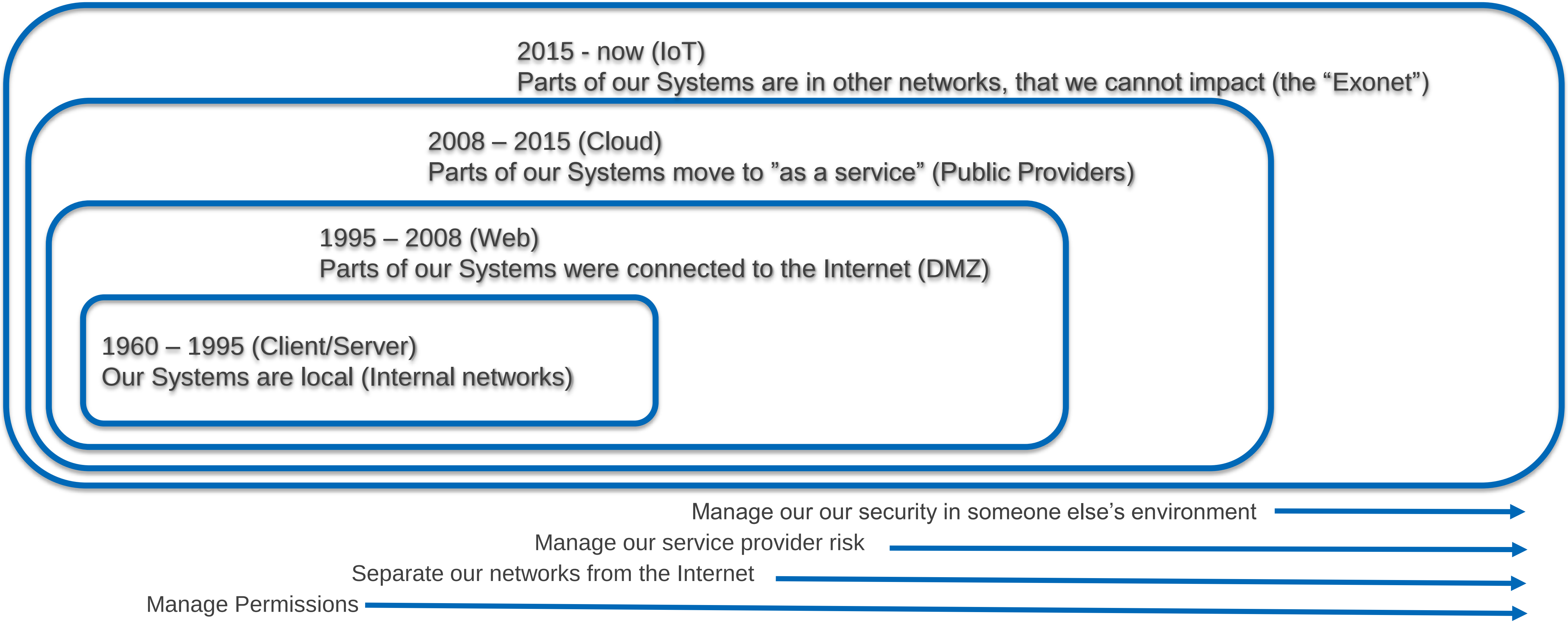


DIFFERENCE IN CONNECTIVITY

- Factory B provides parts for the products from Factory A.
- They connect their systems for each better performance.
- They collaborate, but will not be a part of the consolidated security management.



THE EXONET CHALLENGE FOR CONNECTED SYSTEMS



2015 - now (IoT)
Parts of our Systems are in other networks, that we cannot impact (the “Exonet”)

2008 – 2015 (Cloud)
Parts of our Systems move to ”as a service” (Public Providers)

1995 – 2008 (Web)
Parts of our Systems were connected to the Internet (DMZ)

1960 – 1995 (Client/Server)
Our Systems are local (Internal networks)

Manage our our security in someone else’s environment

Manage our service provider risk

Separate our networks from the Internet

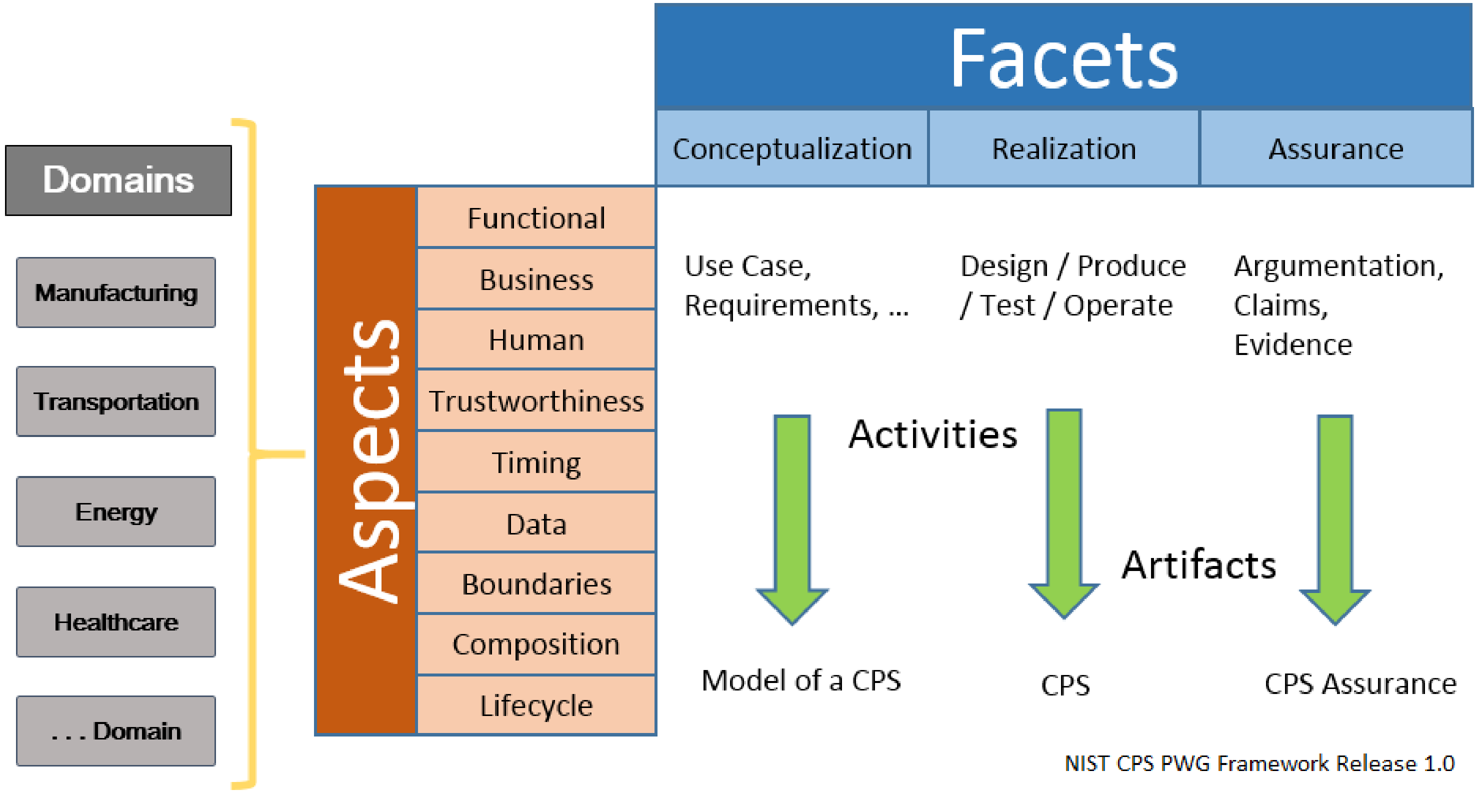
Manage Permissions

CYBER-PHYSICAL SYSTEMS (CPS)

- **Framework for Cyber-Physical Systems**
  - **Published May 2016**
  - **Audience: Designer, Builder, Verifier of CPS**
  - **Goal**
    - Derive a unifying framework that covers the range of unique dimensions of CPS, smart systems that include engineered interacting networks of physical and computational components.
    - Populate a significant portion of the CPS Framework with detail.

KEY ELEMENTS OF THE CPS FRAMEWORK

- Specify the Domain of the target CPS
- For each Aspect in the Domain, formulate Concerns and analyze Facets
  - Conceptualization
  - Realization
  - Assurance



Domains

Manufacturing

Transportation

Energy

Healthcare

. . . Domain

Aspects

Functional

Business

Human

Trustworthiness

Timing

Data

Boundaries

Composition

Lifecycle

Facets

Conceptualization	Realization	Assurance
Use Case, Requirements, …	Design / Produce / Test / Operate	Argumentation, Claims, Evidence
Model of a CPS	CPS	CPS Assurance

Activities

Artifacts

NIST CPS PWG Framework Release 1.0

CONCERNS FOR TRUSTWORTHINESS

A) Concern on reliability requires no unpredictable factors in the system. Active attackers affects unpredictably.

B) Concern on resilience requires minimal availability and recover processes under cyber-attacks.

SECURITY MODEL A

- Concern on reliability requires no unpredictable factors in the system. Active attackers affects unpredictable.
- Honest-but-curious attackers remain honest and impossible to turn active.
- Model A: Assuming attackers eavesdrop all the transcripts in the Internet domain, it requires no credentials are compromised that grant access to Resource Server.

SECURITY MODEL B

- Concern on resilience requires minimal availability and recover processes even under cyber-attacks.
- The impact of security incidents remains bounded and controllable.
- Model B: Assuming the access control mechanism on the application layer does not work at all, it requires the impact from possible unintended use of Resource Servers bounded and recoverable.

ACCESS TOKEN FROM THE TRANSPORT LAYER

```
+--------+                               +---------------+
|        |--(A)- Authorization Request ->|   Resource    |
|        |                               |     Owner     |
|        |<-(B)-- Authorization Grant ---|               |
|        |                               +---------------+
|        |
|        |                               +---------------+
|        |--(C)-- Authorization Grant -->| Authorization |
| Client |                               |     Server    |
|        |<-(D)----- Access Token -------|               |
|        |                               +---------------+
|        |
|        |                               +---------------+
|        |--(E)----- Access Token ------>|    Resource   |
|        |                               |     Server    |
|        |<-(F)--- Protected Resource ---|               |
+--------+                               +---------------+
```

Figure 1: Abstract Protocol Flow

ACCESS TOKEN FROM THE TRANSPORT LAYER

- We only change the delivery method for the token from the transport layer.
  - The AuthZ Server contains a Private CA and issues certificates with the tokens embed.
  - (D) is encapsulated in a client certificate signed by the CA.
  - (E) becomes a TLS connection using the certificate. No explicit transfer for the Token.

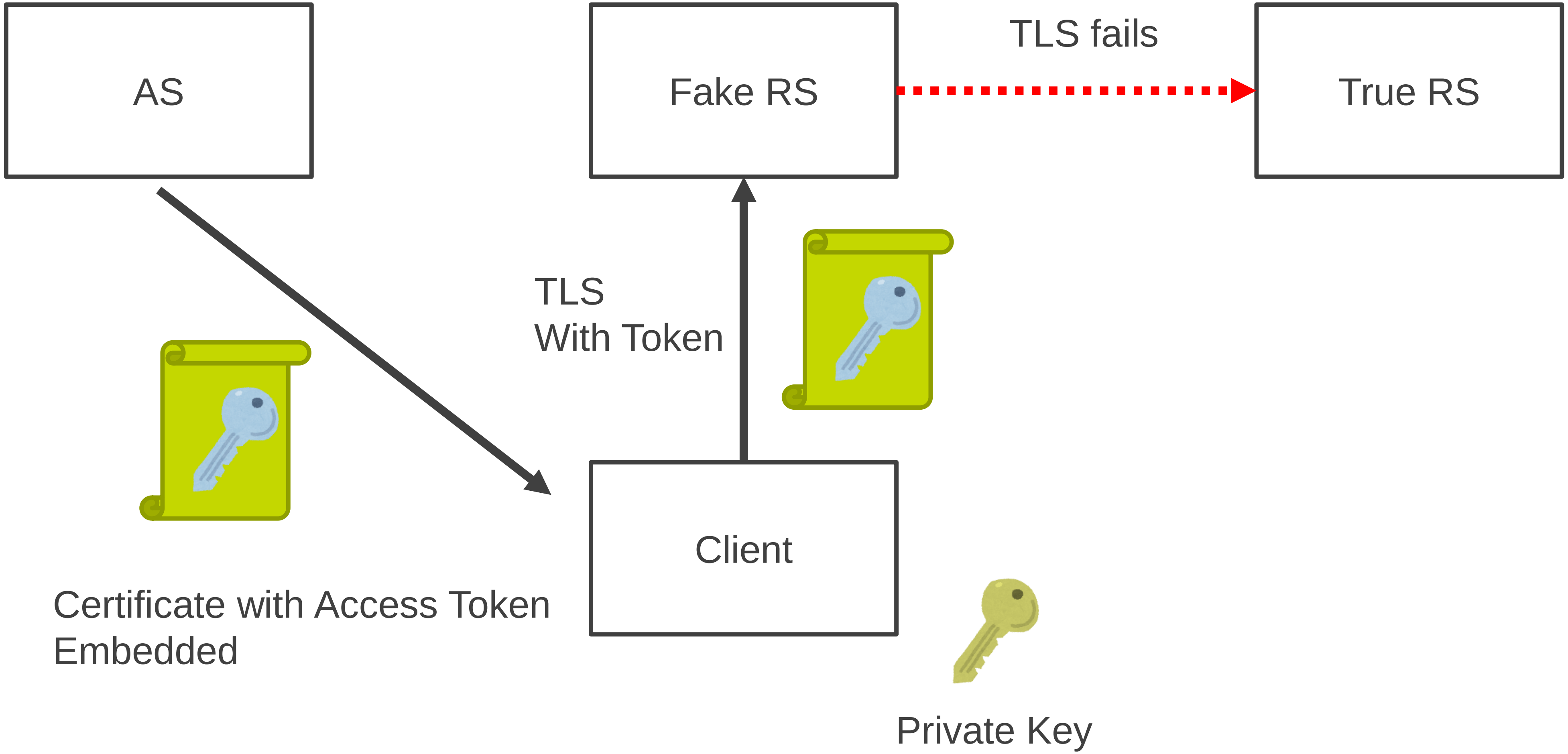
```
+--------+                               +---------------+
|        |--(A)- Authorization Request ->|   Resource    |
|        |                               |     Owner     |
|        |<-(B)-- Authorization Grant ---|               |
|        |                               +---------------+
|        |
|        |                               +---------------+
|        |--(C)-- Authorization Grant -->| Authorization |
| Client |                               |     Server    |
|        |<-(D)----- Access Token -------|   Private CA  |
|        |       X.509 Certificate       +---------------+
|        |
|        |                               +---------------+
|        |--(E)----- Access Token ------>|    Resource   |
|        |   TLS with Client Certificate |     Server    |
|        |<-(F)--- Protected Resource ---|               |
+--------+                               +---------------+
```

Figure 1: Abstract Protocol Flow

FOR EXAMPLE

1. Authorization Server (AS) maintains a private CA service as a part.
2. On issuing an access token, AS embeds JWT to subjectAltName fields of X.509 Certificates and signs the certificate using the CA service. The certificate has short life as well as the corresponding JWT is.
3. Client receives from AS an X.509 Certificate in place of a JWT.
4. Client accesses to Resource Server (RS) using the X.509 Certificate as a Client Certificate. RS requires a valid client certificate to accept the access.
5. RS reproduces the JWT from the X.509 Certificate presented by the Client.

HOW IT PREVENTS PHISHING



AS

Fake RS

TLS fails

True RS

TLS
With Token

Client

Certificate with Access Token Embedded

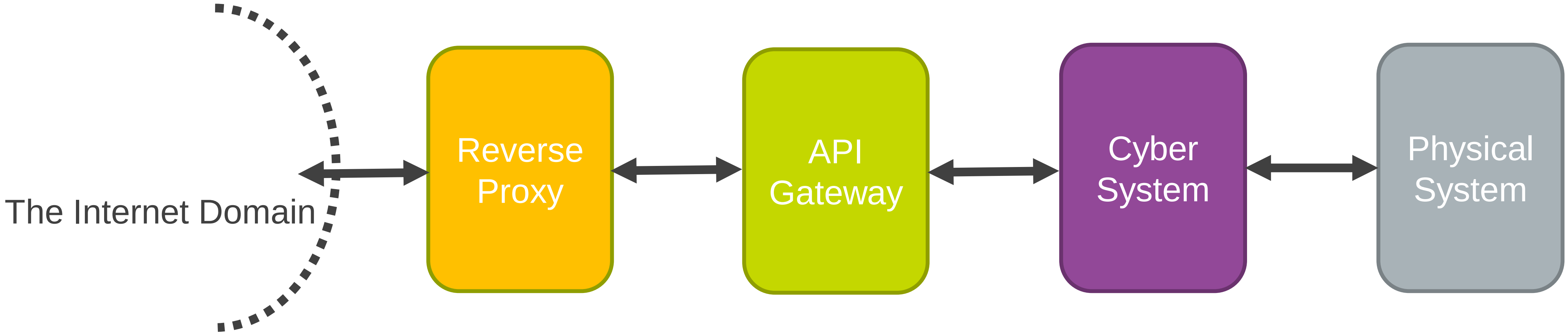
Private Key

THE PROBLEMS SOLVED

- Satisfies MODEL A because the private key for the X.509 certificate is not accessible from the application layer since the key is generated under the transport layer and stored there.
  - It might be a good idea to add a boolean flag that enforces access token delivered from the transport layer.
- Satisfies MODEL B because it is only the entities with X.509 certificates signed by AS that can connect to RS.
  - Track the log from AS that records identities and attributes for which AS signed the certificates, and execute the recovery process for each entities tracked.

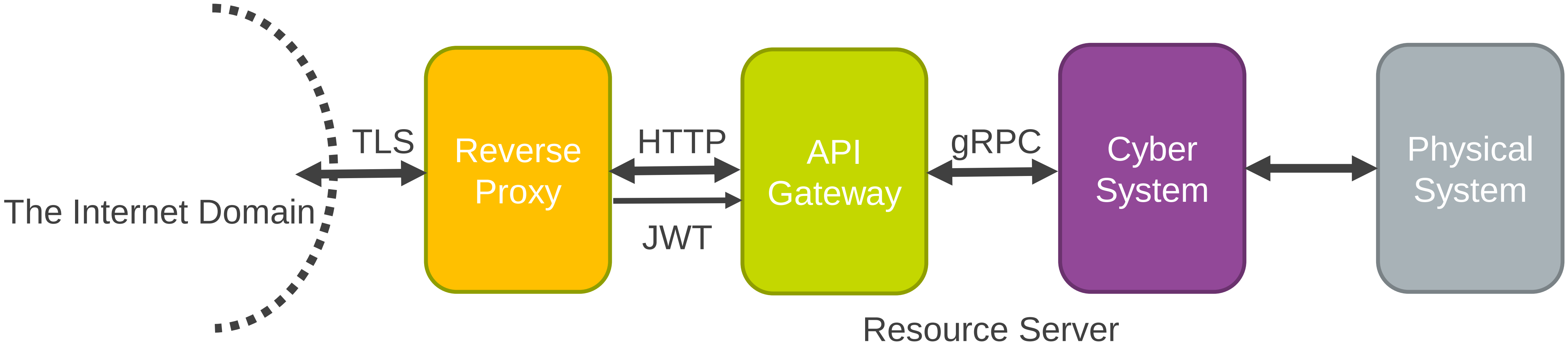
IMPLEMENTATION MODEL

- Consider a typical microservice architecture.



IMPLEMENTATION MODEL

- Reverse proxy verifies the TLS certificate and recovers JWT for API Gateway or Resource Server on the back.



PROOF OF CONCEPT

- Embed JWT to X.509 Certificates
  - Use SubjectAltName to store JWT tokens

```
[req]
req_extensions = v3_req
distinguished_name = req_distinguished_name
[req_distinguished_name]
[ v3_req ]
basicConstraints = CA:FALSE
keyUsage = nonRepudiation, digitalSignature, keyEncipherment
subjectAltName = @alt_names
[alt_names]
otherName.1 = msUPN;UTF8:${BEARER_TOKEN}
```

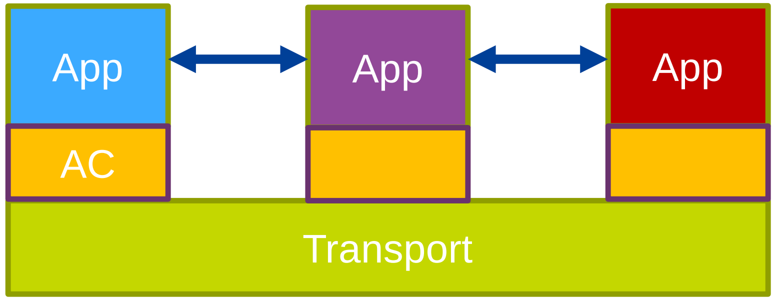
PROOF OF CONCEPT

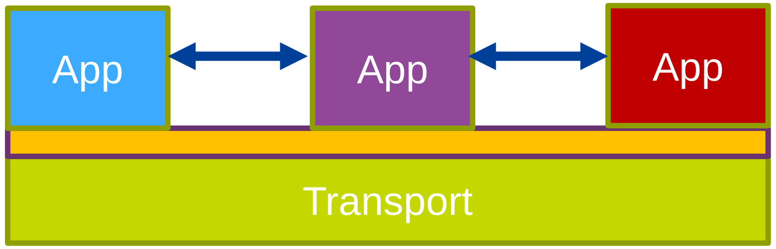
- Recover JWT from X.509 Certificates
  - Set Headers at Reverse Proxy using Apache httpd.

```
RequestHeader set Authorization "Bearer %{SSL_CLIENT_SAN_OTHER_msUPN_0}s"
```

THE TRUSTED TRANSPORT LAYER FOR CPS

- Reduce the complexity of Assurance in Trustworthiness
  - Applications are always updated asynchronously
  - Applications consist of variety of microservices with different technologies and design
- The complexity is critical for the connected systems.





AC: access control layer

DISCUSSION

- What fields are the best for embedding JWT for access tokens? We are aware of RFC 5755, which extends fields for Attribute Certificate Profile for Authorization. Are they better places?

DISCUSSION

- Refresh token. We think refresh tokens can be embedded into X.509 certificate in the same way.
- We may try another idea to use the X.509 certificates for refresh.
  - Verifying a valid client certificate with an expired access token, AS re-issues the access token embedded in a new X.509 certificate.
  - To enforce preventing refresh, add “allow-refresh” boolean flag to the certificate.