Practical Distributed Authorization

Ankur Taly, Google Inc.
ataly@google.com

Two day course at the 2016 International School on Foundations of Security Analysis and Design, Bertinoro, Italy (Aug. 29 - Sep. 2)

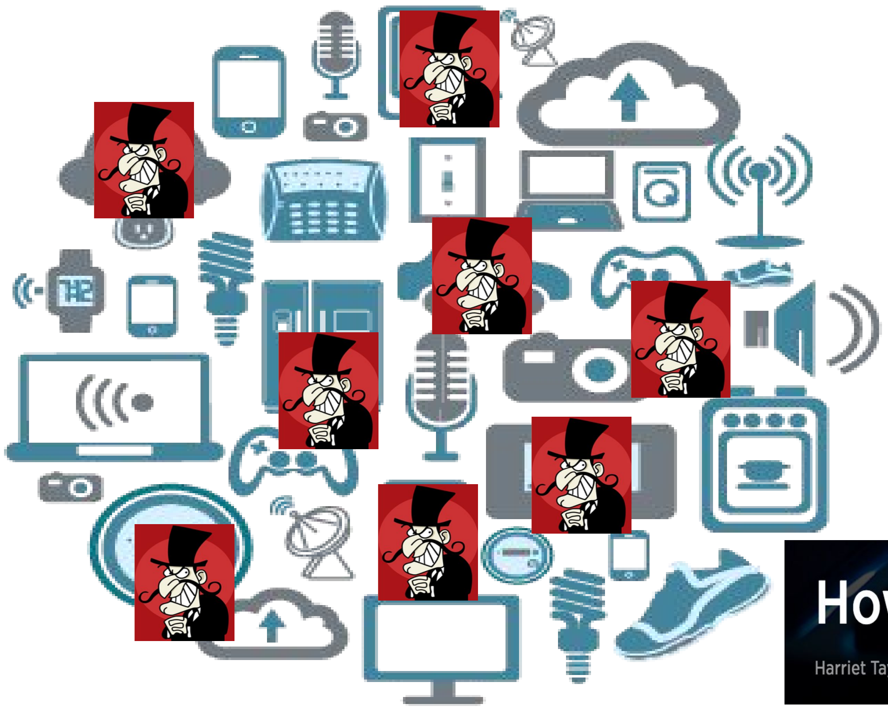
Internet of Things (IoT)

Physical devices made accessible over the network

Exciting new possibilities!

img source: http://www.ti.com/lsds/media/images/wireless_connectivity/50BillionThings.png

Internet of Things Security



Goldmine for the bad guys

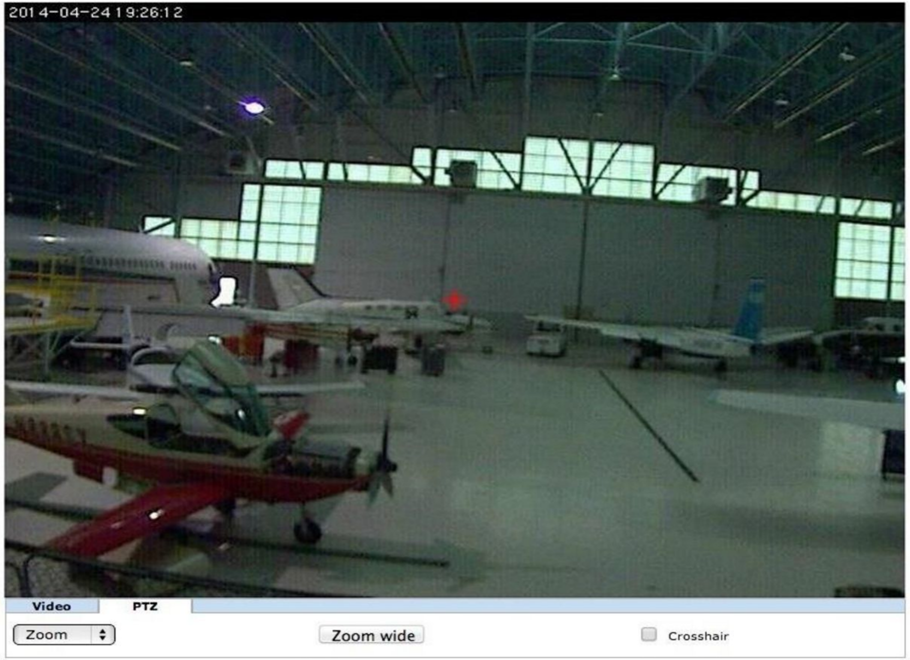
Scary new possibilities!

How the 'Internet of Things' could be fatal

Harriet Taylor | @Harri8t | Friday, 4 Mar 2016 | 1:57 PM ET

This is really scary!

Live feed from an airplane hangar in Norway!!



Found using `shodan.io` --- a search engine for finding devices (IoT), e.g., routers, servers, cameras, SCADA systems, HVAC systems etc.

source: http://img.wonderhowto.com/img/original/32/45/63534020036048/0/635340200360483245.jpg

Securing IoT

Naming and Authentication

How do devices name and identify each other during any interaction?

Delegation

How do users delegate devices to act on their behalf ?

Access control

How are access control policies defined?

Securing IoT

Naming and Authentication

How do devices name and identify each other during any interaction?

Delegation

How do users delegate devices to act on their behalf ?

Access control

How are access control policies defined?

This is in essence the problem of authorization in distributed systems

Authorization in distributed systems

Old problem with decades of amazing research

But, new hype around it (courtesy IoT)

Course overview

- Explore existing ideas and techniques in distributed authorization
- Evaluate their applicability to IoT and large, open distributed systems
- Develop the applications in the context of the “Vanadium” framework developed at Google Inc.

Agenda

Today

- Foundations of distributed authorization
- Authorization requirements for large distributed systems (e.g., IoT)
- Overview of the Vanadium authorization model

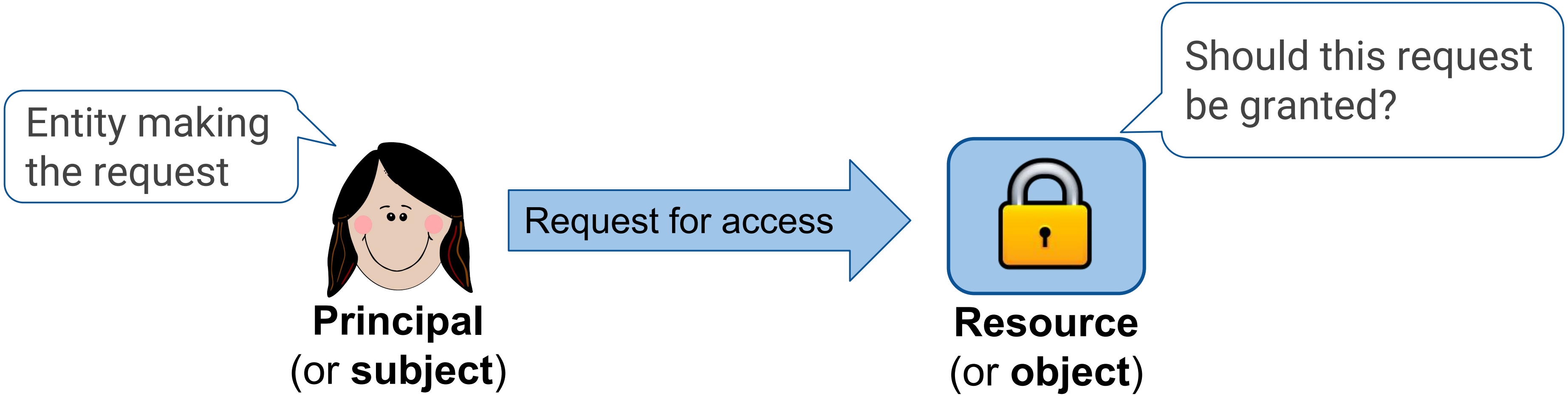
Tomorrow

- Access control policies in Vanadium
- Privacy, discovery and authentication for Vanadium

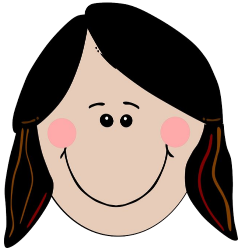
Fundamentals of Distributed Authorization

Authorization

Fundamental problem in computer security that deals with whether a request to access a resource must be granted



Example: Door lock



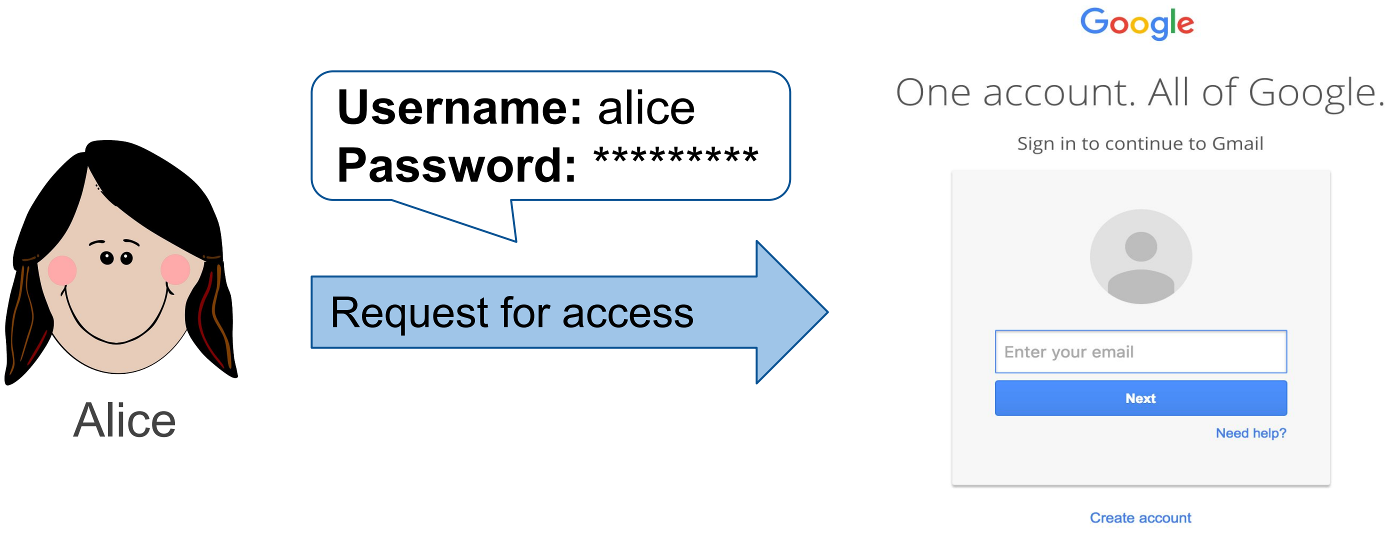
Alice

access code: 4224

Request for access

Request is authorized only if the entered access code is valid

Example: Web login



Login page grants access to **<u>Google properties</u>** (e.g., GMail) only if the entered password is valid

Example where principal is non-user

AccessToken=xxx

Post on user's timeline

Facebook Graph API

Facebook API allows access to user's profile only if provided access token is valid and has the appropriate permissions

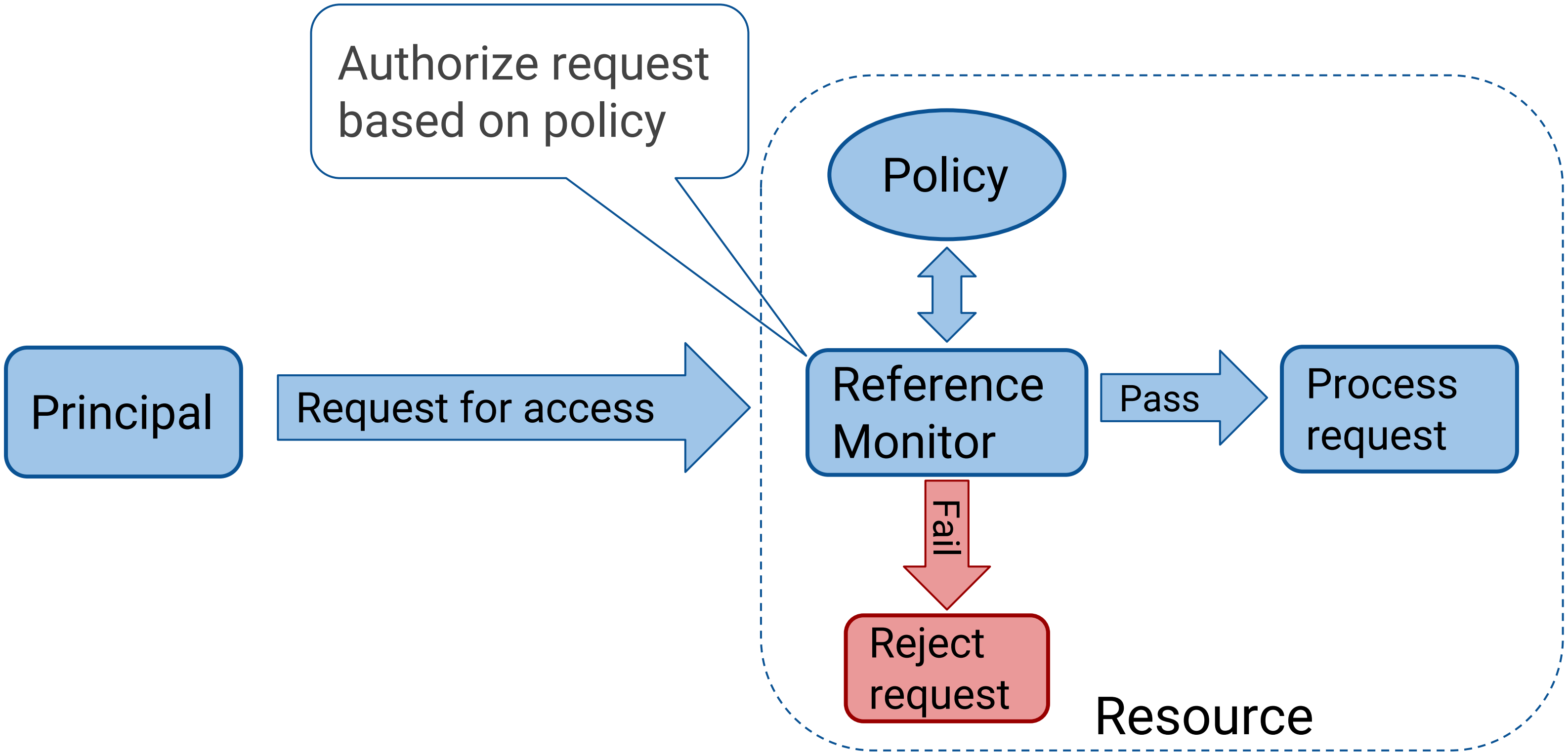
Principals

Entity making the request, can be:

- user
- device
- application
- browser tab
- or some combination of the above

Granularity varies across systems

Authorization model



Reference monitor (in closed systems)

Authentication + Access control

Authentication: Identify the principal making the request

- as a username, email, accountID, etc.

Access control: check if the identified principal is allowed by the policy

	file1	file2	file3
Alice	r	r	rwx
Bob	rw		x
Carol	rw	rw	x

Access control matrix
[Lampson, 1971]

Distributed authorization

Authorization is much more complicated in large, open, distributed systems such as the Web, Internet-of-Things (IoT)

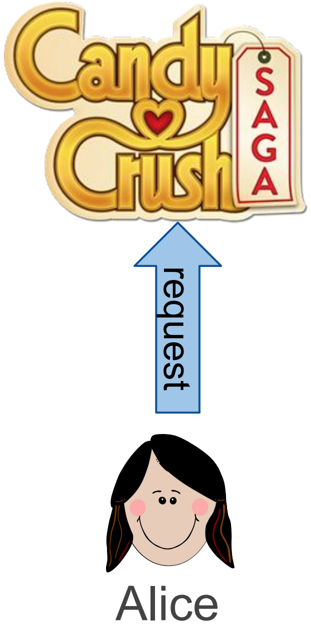
- No relationship between reference monitor and principal prior to request
  - may have to rely on third-parties for issuing and/or validating credentials
- Access control policies may be distributed
- The resource itself may be distributed
- Communication channels cannot be trusted

Delegation of authority and trust is essential

Example: >21 age check

Relying on a government (third-party) issued ID for verifying age > 21

Example: Third-party authentication



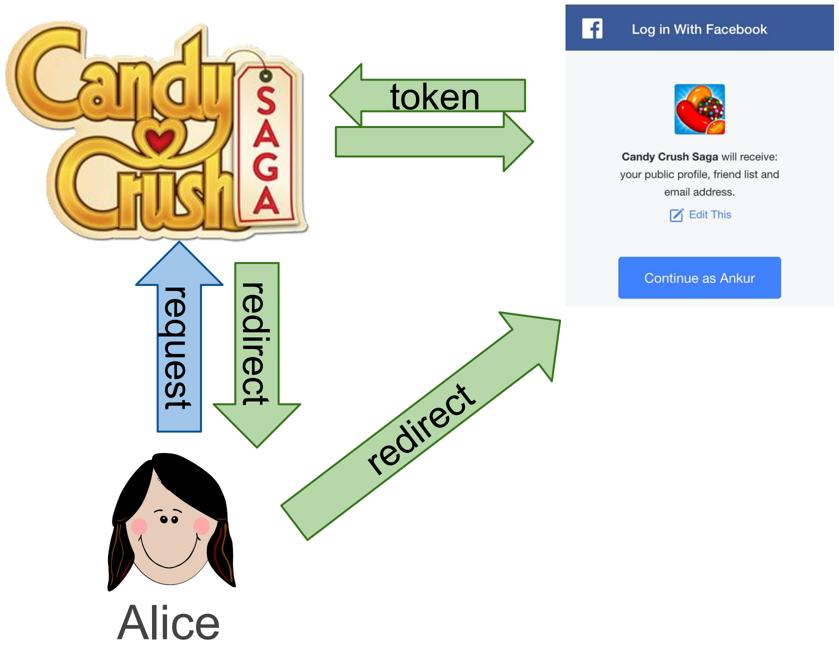
Problem

CandyCrush wants to service a request from Alice

But, it doesn’t know how to authenticate Alice

How does CandyCrush authorize a request from Alice?

Example: Third-party authentication



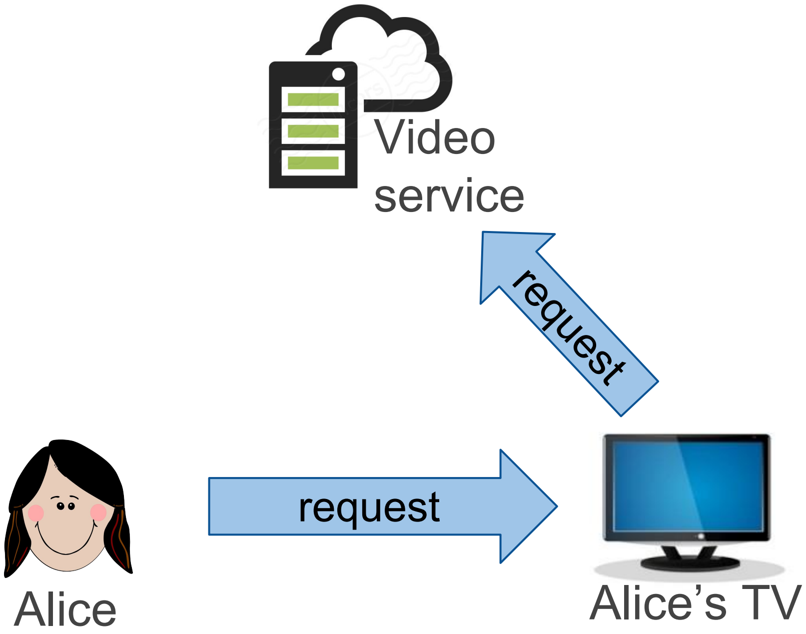
Solution

Both CandyCrush and Alice have a relationship with Facebook

CandyCrush redirects Alice to Facebook and request an OAuth2 access token

It uses the token to obtain Alice's profile information at Facebook

Example: Streaming videos on a TV



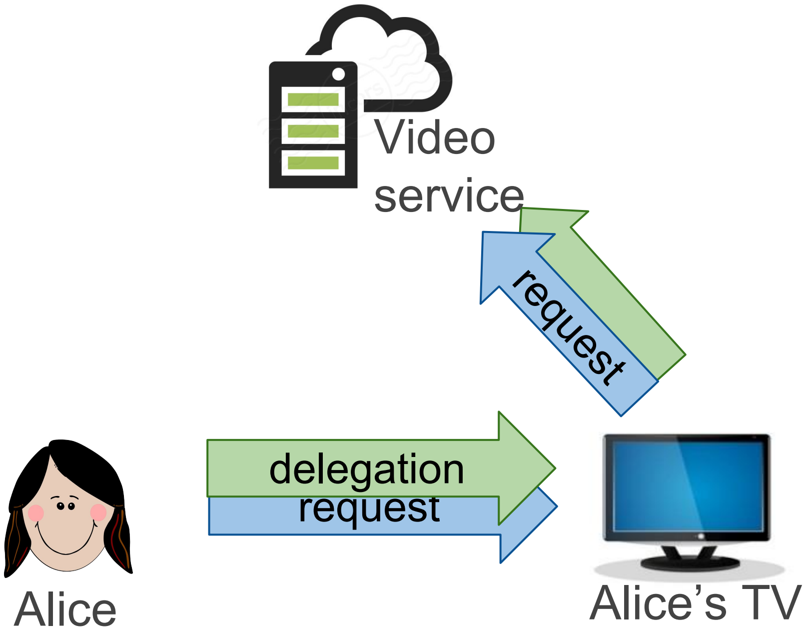
Problem

Alice wants to stream a video from her Video server to her TV

Video service has a relationship with Alice but NOT with Alice's TV

How does the Video service authorize a request from Alice's TV?

Example: Stream a video on a TV



Solution

Alice authenticates the TV and hands it a credential to access the video service

TV presents this credential to the video service proving that it is authorized by Alice

Credentials-based authorization

- Authorization is based on credentials bound to the principal specifying
  - characteristics of the principal, e.g., identity, role, etc.
  - some other aspect of system state, e.g., time, location, etc.

Access control problem: Verify that a set of credentials C satisfy a policy P in the context of a request r

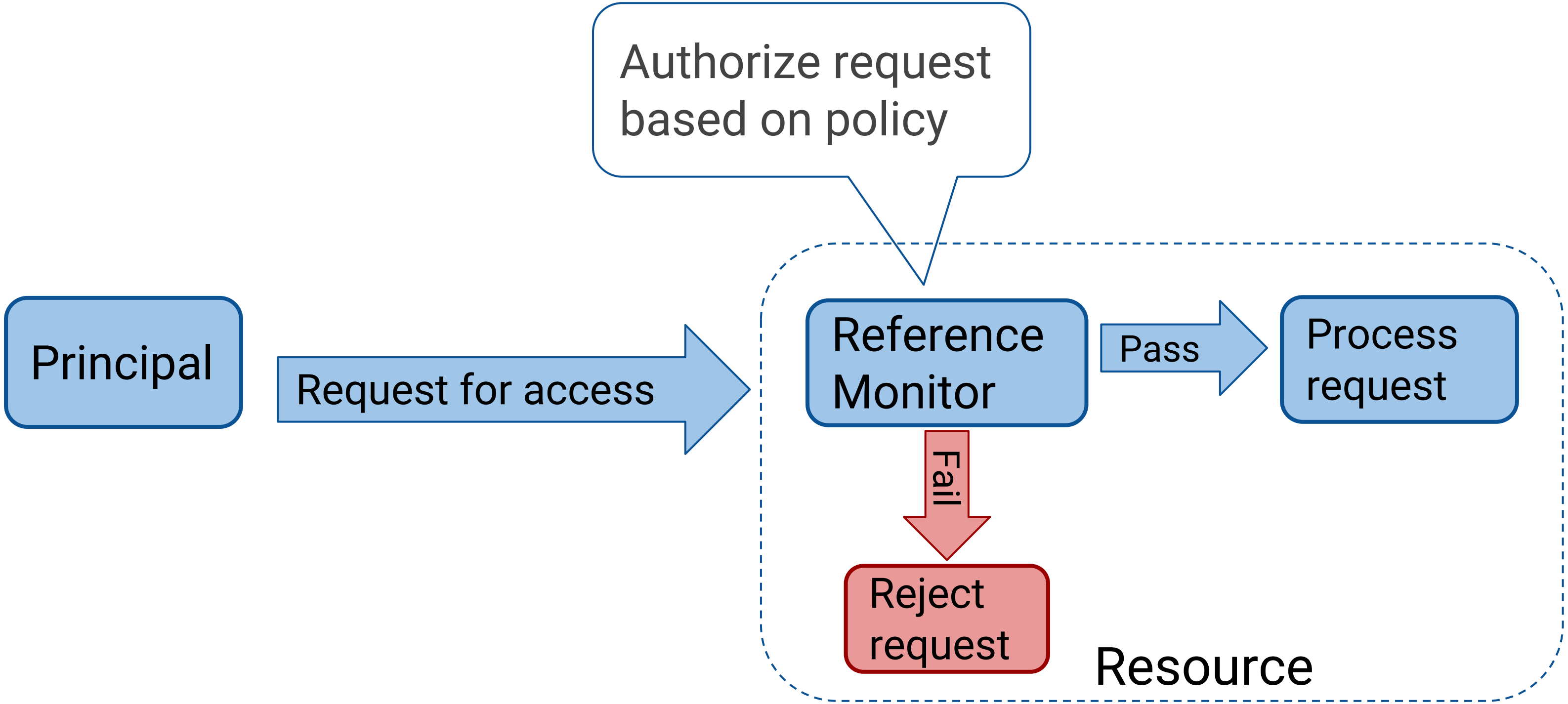
Credentials-based authorization

- Authorization is based on <u>credentials</u> bound to the principal specifying
  - characteristics of the principal, e.g., identity, role, etc.
  - some other aspect of system state, e.g., time, location, etc.

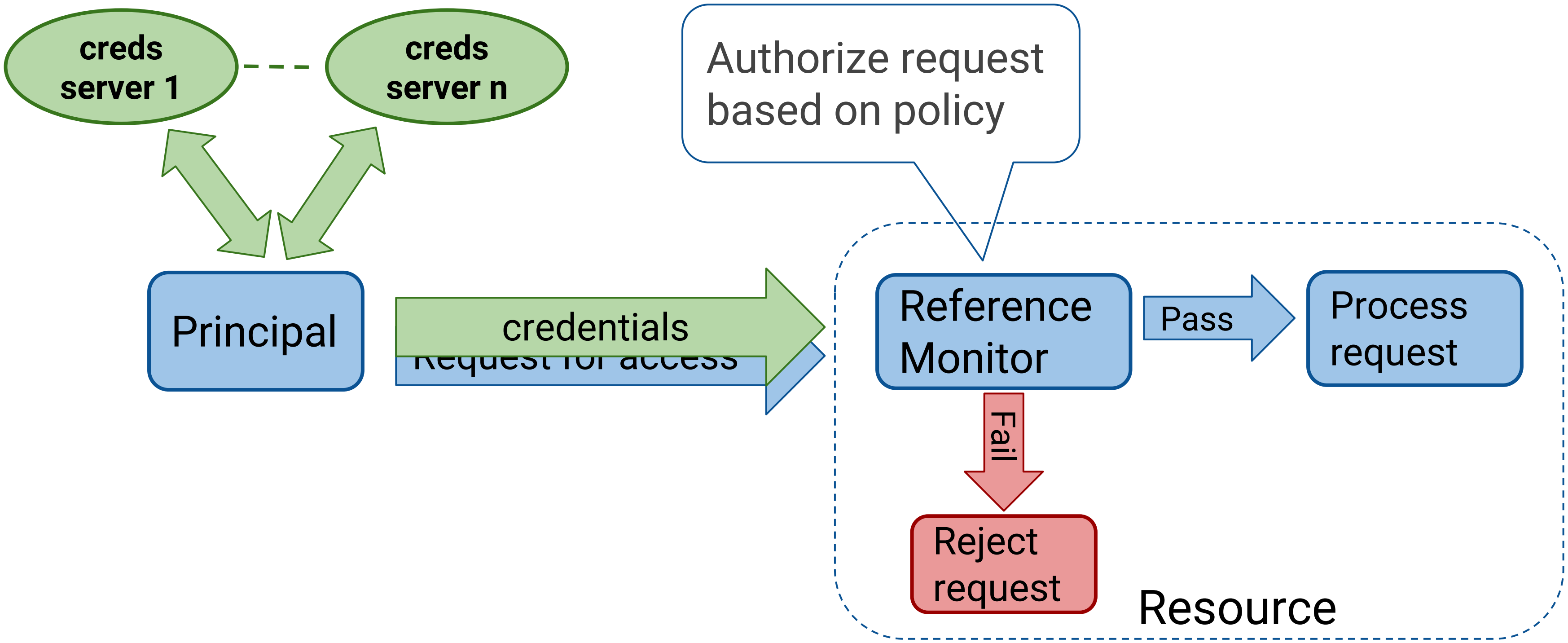
Access control problem: Verify that a set of credentials C satisfy a policy P in the context of a request r

- Different credential issuers are trusted for different purposes
- Credentials are either:
  - presented by the principal OR
  - gathered by the reference monitor on demand

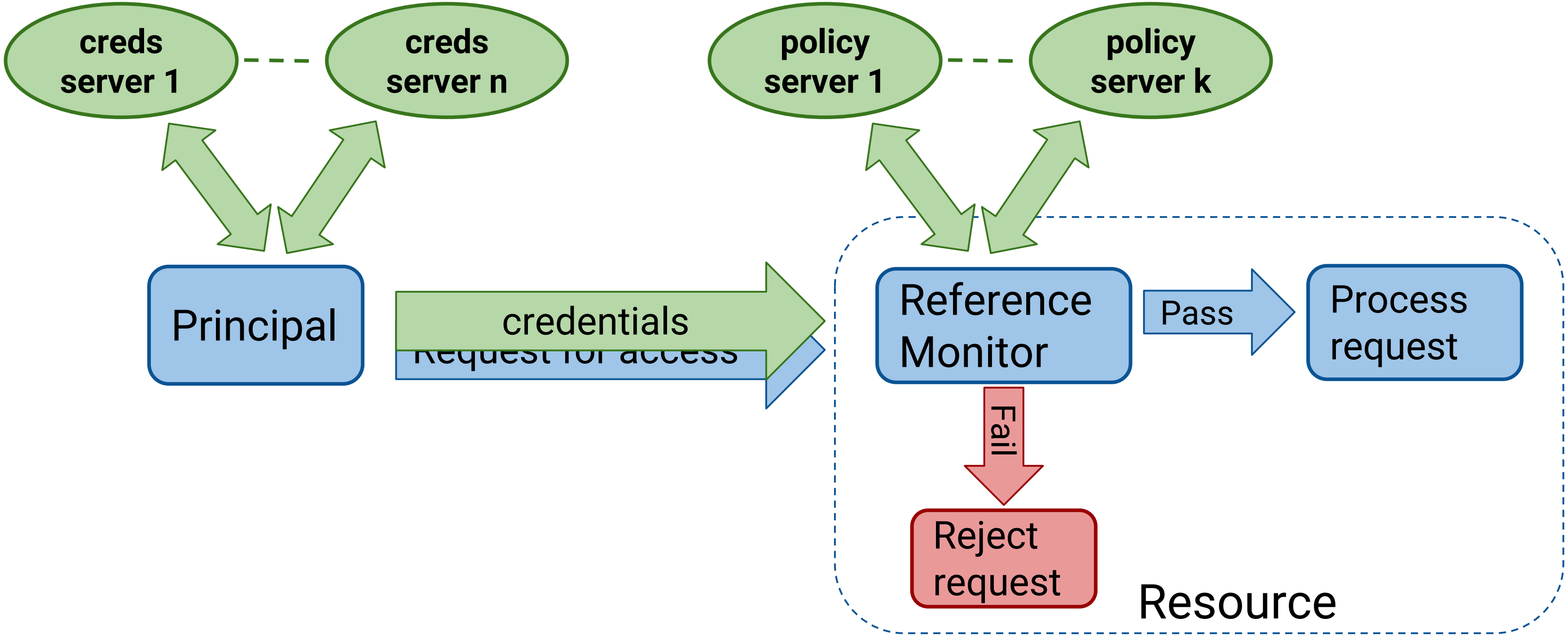
Authorization model



Distributed authorization model



Distributed authorization model



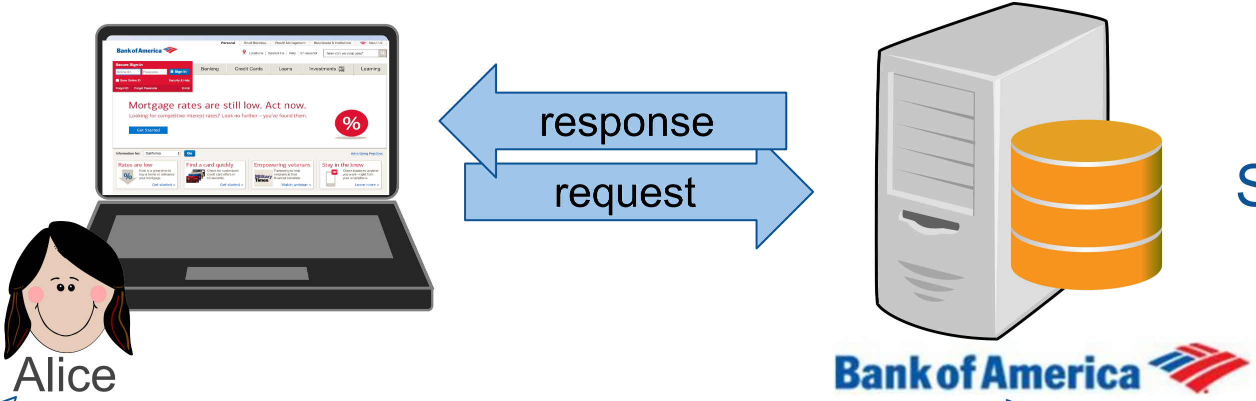
Building blocks

- Mechanisms for generating, distributing, and validating credentials
- Languages for defining access control policies
- Algorithms and logics for checking policies
- Protocols for setting up secure communication channels

Web authorization model

Mutual authorization

Client



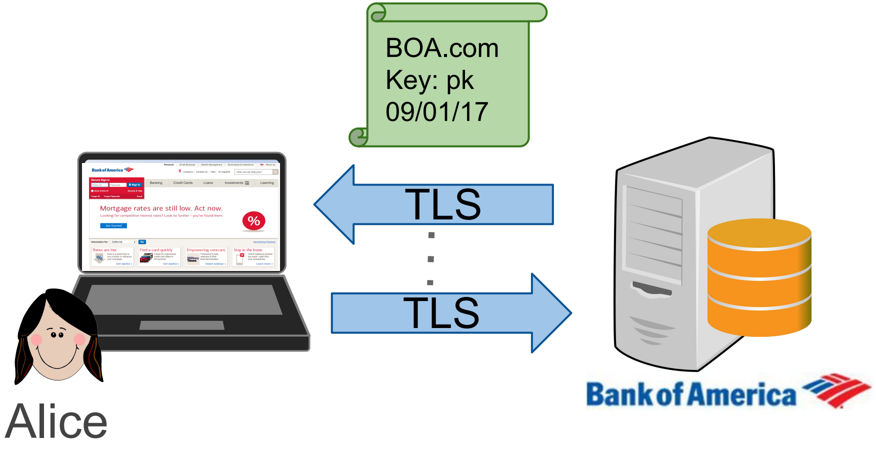
Server

Verify that the server's credentials identify it as `bankofamerica.com`

Verify that the client's credentials identify it as Alice

Server authorization on the Web (under TLS)

TLS protocol allows clients to authorize the server and establishes an encrypted channel between them

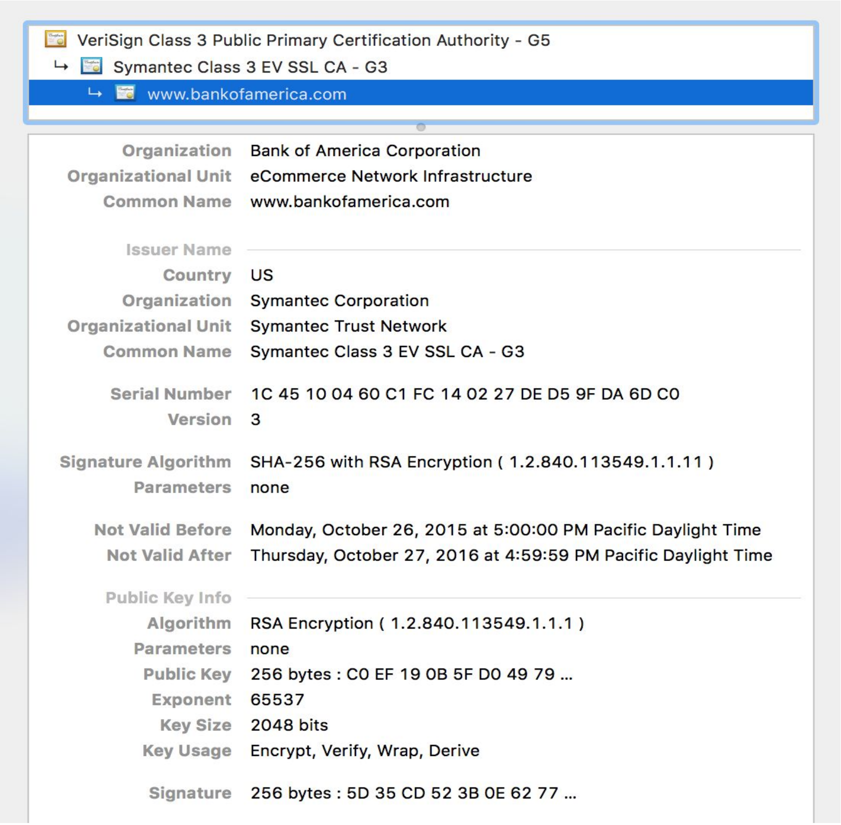


Servers possess

- a digital signature public and secret key pair (pk, sk)
  - **∀ m. Verify(pk, m, Sign(sk, m))**
- a signed x509 certificate binding a domain name (`bankofamerica.com`) to the public key pk

During TLS, the server presents its certificate to a client

Server authorization on the Web (under TLS)

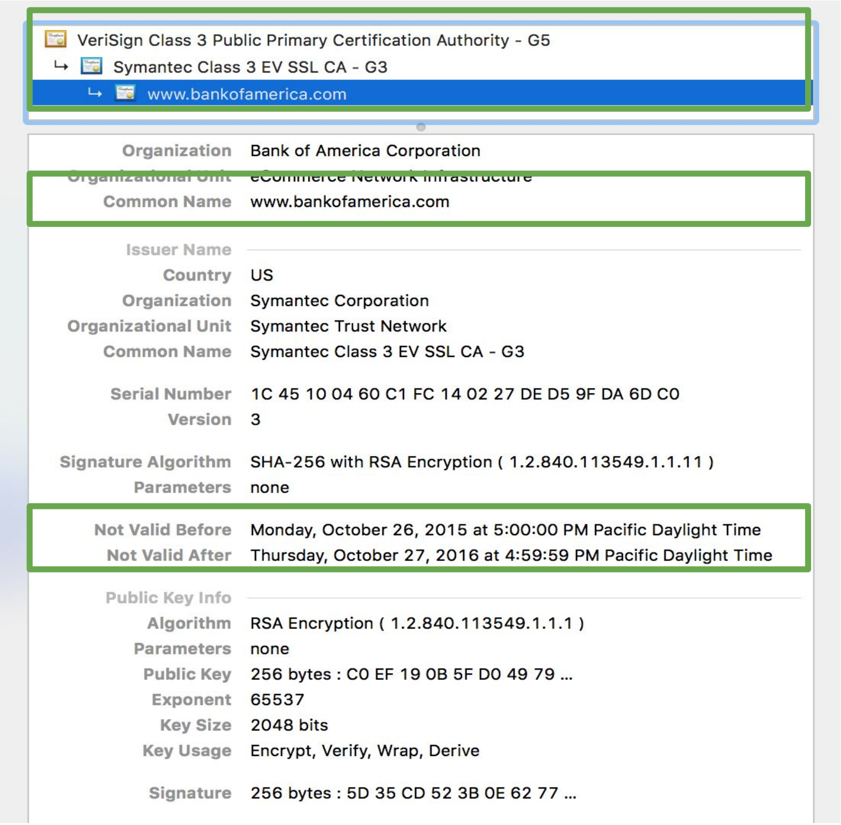


Server certificate is in the X509 format which is very expressive but hard to parse

Client verifies that the certificate

- has not expired
- has the expected domain name
- has a recognized issuer

Server authorization on the Web (under TLS)



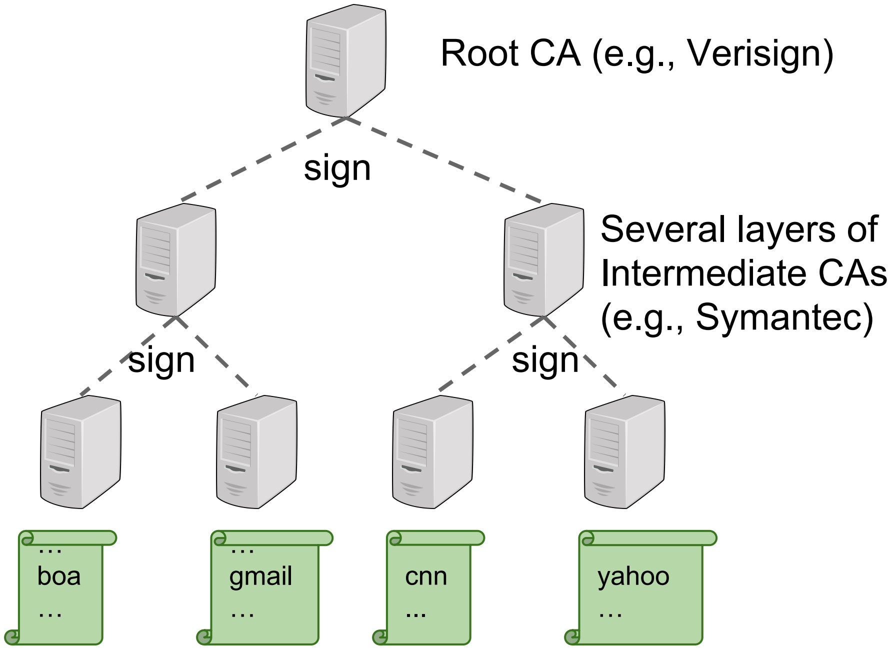
Server certificate is in the X509 format which is very expressive but hard to parse

Client verifies that the certificate

- has not expired
- has the expected domain name
- has a <u>recognized issuer</u>

Web public-key infrastructure (PKI)

Hierarchical network of CAs



- Root CA certifies intermediate CAs which certify Web servers
  - Root CA certificate is **self-signed**
  - About 60 root CAs and 1200 intermediate CAs
- CAs can issue certificates for any domain
- A wrongly issued certificate can be used to impersonate a server

Browsers maintain list of trusted CAs

Name	Kind	Date Modified	Expires	Keychain
TC TrustCenter Universal CA II	certificate	--	Dec 31, 2030, 2:59:59 PM	System Roots
TC TrustCenter Universal CA III	certificate	--	Dec 31, 2029, 3:59:59 PM	System Roots
TeliaSonera Root CA v1	certificate	--	Oct 18, 2032, 5:00:50 AM	System Roots
thawte Primary Root CA	certificate	--	Jul 16, 2036, 4:59:59 PM	System Roots
thawte Primary Root CA - G2	certificate	--	Jan 18, 2038, 3:59:59 PM	System Roots
thawte Primary Root CA - G3	certificate	--	Dec 1, 2037, 3:59:59 PM	System Roots
TRUST2408 OCES Primary CA	certificate	--	Dec 3, 2037, 5:11:34 AM	System Roots
Trusted Certificate Services	certificate	--	Dec 31, 2028, 3:59:59 PM	System Roots
Trustis FPS Root CA	certificate	--	Jan 21, 2024, 3:36:54 AM	System Roots
TÜBİTAK UEK...ğlayıcısı - Sürüm 3	certificate	--	Aug 21, 2017, 4:37:07 AM	System Roots
TÜRKTRUST...a Hizmet Sağlayıcısı	certificate	--	Dec 22, 2017, 10:37:19 AM	System Roots
TWCA Global Root CA	certificate	--	Dec 31, 2030, 7:59:59 AM	System Roots
TWCA Root Certification Authority	certificate	--	Dec 31, 2030, 7:59:59 AM	System Roots
UCA Global Root	certificate	--	Dec 30, 2037, 4:00:00 PM	System Roots
UCA Root	certificate	--	Dec 30, 2029, 4:00:00 PM	System Roots
UTN - DATACorp SGC	certificate	--	Jun 24, 2019, 12:06:30 PM	System Roots
UTN-USERFir...ntication and Email	certificate	--	Jul 9, 2019, 10:36:58 AM	System Roots
UTN-USERFirst-Hardware	certificate	--	Jul 9, 2019, 11:19:22 AM	System Roots
UTN-USERFir...twork Applications	certificate	--	Jul 9, 2019, 11:57:49 AM	System Roots
UTN-USERFirst-Object	certificate	--	Jul 9, 2019, 11:40:36 AM	System Roots
VeriSign Clas...tion Authority - G3	certificate	--	Jul 16, 2036, 4:59:59 PM	System Roots
VeriSign Clas...tion Authority - G3	certificate	--	Jul 16, 2036, 4:59:59 PM	System Roots
VeriSign Clas...tion Authority - G3	certificate	--	Jul 16, 2036, 4:59:59 PM	System Roots
VeriSign Clas...tion Authority - G4	certificate	--	Jan 18, 2038, 3:59:59 PM	System Roots
VeriSign Clas...tion Authority - G5	certificate	--	Jul 16, 2036, 4:59:59 PM	System Roots
VeriSign Clas...tion Authority - G3	certificate	--	Jul 16, 2036, 4:59:59 PM	System Roots
VeriSign Univ...rtification Authority	certificate	--	Dec 1, 2037, 3:59:59 PM	System Roots
Visa eCommerce Root	certificate	--	Jun 23, 2022, 5:16:12 PM	System Roots
Visa Information Delivery Root CA	certificate	--	Jun 29, 2025, 10:42:42 AM	System Roots
VRK Gov. Root CA	certificate	--	Dec 18, 2023, 5:51:08 AM	System Roots
WellsSecure...Certificate Authority	certificate	--	Dec 13, 2022, 4:07:54 PM	System Roots

Browsers maintain list of trusted CAs

Name	Kind	Date Modified	Expires	Keychain
TC TrustCenter Universal CA II	certificate	--	Dec 31, 2030, 2:59:59 PM	System Roots
TC TrustCenter Universal CA III	certificate	--	Dec 31, 2029, 3:59:59 PM	System Roots
TeliaSonera Root CA v1	certificate	--	Oct 18, 2032, 5:00:50 AM	System Roots
thawte Primary Root CA	certificate	--	Jul 16, 2036, 4:59:59 PM	System Roots
thawte Primary Root CA - G2	certificate	--	Jan 18, 2038, 3:59:59 PM	System Roots
thawte Primary Root CA - G3	certificate	--	Dec 1, 2037, 3:59:59 PM	System Roots
TRUST2408 OCES Primary CA	certificate	--	Dec 3, 2037, 5:11:34 AM	System Roots
Trusted Certificate Services	certificate	--	Dec 31, 2028, 3:59:59 PM	System Roots
Trustis FPS Root CA	certificate	--	Jan 21, 2024, 3:36:54 AM	System Roots
TÜBİTAK UEK...ğlayıcısı - Sürüm 3	certificate	--	Aug 21, 2017, 4:37:07 AM	System Roots
TÜRKTRUST...a Hizmet Sağlayıcısı	certificate	--	Dec 22, 2017, 10:37:19 AM	System Roots
TWCA Global Root CA	certificate	--	Dec 31, 2030, 7:59:59 AM	System Roots
TWCA Root Certification Authority	certificate	--	Dec 31, 2030, 7:59:59 AM	System Roots
UCA Global Root	certificate	--	Dec 30, 2037, 4:00:00 PM	System Roots
UCA Root	certificate	--	Dec 30, 2029, 4:00:00 PM	System Roots
UTN - DATACorp SGC	certificate	--	Jun 24, 2019, 12:06:30 PM	System Roots
UTN-USERFir...ntication and Email	certificate	--	Jul 9, 2019, 10:36:58 AM	System Roots
UTN-USERFirst-Hardware	certificate	--	Jul 9, 2019, 11:19:22 AM	System Roots
UTN-USERFir...twork Applications	certificate	--	Jul 9, 2019, 11:57:49 AM	System Roots
UTN-USERFirst-Object	certificate	--	Jul 9, 2019, 11:40:36 AM	System Roots
VeriSign Clas...tion Authority - G3	certificate	--	Jul 16, 2036, 4:59:59 PM	System Roots
VeriSign Clas...tion Authority - G3	certificate	--	Jul 16, 2036, 4:59:59 PM	System Roots
VeriSign Clas...tion Authority - G3	certificate	--	Jul 16, 2036, 4:59:59 PM	System Roots
VeriSign Clas...tion Authority - G4	certificate	--	Jan 18, 2038, 3:59:59 PM	System Roots
VeriSign Clas...tion Authority - G5	certificate	--	Jul 16, 2036, 4:59:59 PM	System Roots
VeriSign Clas...tion Authority - G3	certificate	--	Jul 16, 2036, 4:59:59 PM	System Roots
VeriSign Univ...rtification Authority	certificate	--	Dec 1, 2037, 3:59:59 PM	System Roots
Visa eCommerce Root	certificate	--	Jun 23, 2022, 5:16:12 PM	System Roots
Visa Information Delivery Root CA	certificate	--	Jun 29, 2025, 10:42:42 AM	System Roots
VRK Gov. Root CA	certificate	--	Dec 18, 2023, 5:51:08 AM	System Roots
WellsSecure...Certificate Authority	certificate	--	Dec 13, 2022, 4:07:54 PM	System Roots

Recent CA compromise incidents

2014: Indian NIC (intermediate CA trusted by the Indian CCA root authority) issued unauthorized certificates for several Google domains [link]

Response

- Indian CCA revoked all NIC certificates
- Chrome restricted Indian CCA to 7 domains

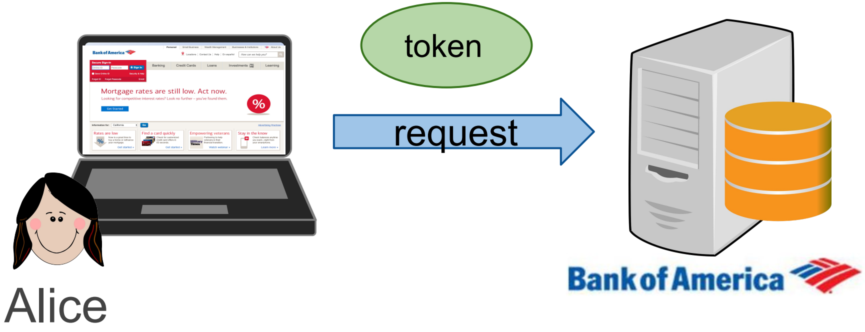
2015: MCS Holdings (intermediate CA trusted by CNNIC root authority) issued unauthorized certificates for several Google domains [link]

Response

- Chrome revoked the malicious certificates and stopped recognizing CNNIC as a root CA

Client authorization on the Web

Clients typically send their credentials after TLS completes and an encrypted channel is established



Credentials are mostly **bearer tokens** but have many flavors

- Username/passwords
- Cookies
- OAuth2 tokens
  - delegated by an identity provider
- Macaroons
  - delegated across multiple third-parties

Designing client credentials has been a far more creative space than server credentials

Token based authorization

Upsides:

- Simple, efficient, easy to deploy
- Tokens can be attenuated and delegated peer-to-peer (e.g., Macaroons)
- Ubiquitous on the Web, standardized with lots of implementations

Downsides:

- Roundtrip to issuer for token creation and verification
- Proliferation of tokens at clients; one per issuer

Alternate public-key infrastructures (PKI)

Decentralized approach to certification

- Pretty good privacy (PGP)
- Simple Distributed Security Infrastructure (SDSI)

Pretty Good Privacy (PGP) [Zimmerman 94]

- Framework for encrypting email
- Principals have encryption public and secret pairs, and certificates binding email address to encryption public keys
- **Web of trust**: *Egalitarian approach => anybody can sign certificates*
    - Alice may sign a certificate for her friend Bob's public key
    - Carol will recognize this certificate as long as she recognizes Alice
    - Trust grows organically rather than through a hierarchy of CAs
- Related startup: https://keybase.io/

Simple Distributed Security Infrastructure (SDSI)

- Also an egalitarian approach
- Principals issue certificates binding ***local names*** to other principals
  - e.g., Alice issues a certificate binding “friend” to Bob’s public key
- **Linked local namespaces**
  - Certificate can be linked to form chains of names
  - Alice’s TV (another principal) who refers to Alice as “Alice” may refer to Bob as “Alice’s friend”
- **Name based access control policies**
  - Alice’s TV may authorize anyone with a name matching “Alice’ s friend”

Simple Distributed Security Infrastructure (SDSI)

History of SDSI

- Originally developed by Rivest and Lampson in 1996
- Later merged with Elisson’s related Simple Public Key Infrastructure (SPKI), and is now jointly referred as **SPKI/SDSI**
- Followed by RFCs for standardization [2692, 2693], several academic papers providing algorithms, semantics, logics, etc.

Distributed authorization history

80s and 90s: Lots of interesting distributed authorization research

Frameworks: KeyNote, PGP, SPKI/SDSI, X509, Active certificates, Macaroons, ...

Policy languages and logic: ABLP, RT, SecPal, Binder, ...

Late 90s: Web authorization model took off

- Centralized x509 PKI for server authorization
- Various token flavors for client authorization

Last few years: Distributed authorization research is back in demand, thanks to **Internet-of-Things (IoT)!**

Internet of Things Security!

WND EXCLUSIVE

'HACKERS REMOTELY KILL A JEEP ON THE HIGHWAY'

Several baby monitors vulnerable to hacking, cybersecurity firm warns

The Associated Press Posted: Sep 02, 2015 1:53 PM ET | Last Updated: Sep 02, 2015 2:13 PM ET

Refrigerator Busted Sending Spam Emails In Massive Cyberattack

The Huffington Post | By Ryan Grenoble

Hackers Can Remotely Hack Self-Aiming Rifles to Change Its Target

Thursday, July 30, 2015 Mohit Kumar

Top IoT Vulnerabilities

HP Study Reveals 70 Percent of Internet of Things Devices Vulnerable to Attack

danielmiessler 07-29-2014 05:09 AM - edited 07-07-2015 12:33 PM

- Insufficient authentication and authorization (80%)
- Lack of transport encryption (70%)
- Insecure web interfaces (60%)
- Insecure software updates (60%)
- Insecure defaults (70%)

Top IoT Vulnerabilities

HP Study Reveals 70 Percent of Internet of Things Devices Vulnerable to Attack

danielmiessler 07-29-2014 05:09 AM - edited 07-07-2015 12:33 PM

- Insufficient authentication and authorization (80%)
- Lack of transport encryption (70%)
- Insecure web interfaces (60%)
- Insecure software updates (60%)
- Insecure defaults (70%)

Security First: Bake in security mechanisms from the ground up

Authorization challenges for IoT

- Devices can behave both as clients and servers
- Far too many IoT devices than Web domains
  - Gartner: There will be **20 billion IoT devices by 2020** [link]
  - Centralized certificate mechanisms may not scale
- Fragmented ecosystem, trust relationships are more nuanced
- Limited network connectivity and bandwidth
- Very little human administration

Authorization Requirements

Summary

- Decentralized deployment
- Mutual authorization
- Fine-grained delegation
- Auditable access
- Revocation
- Ease of use

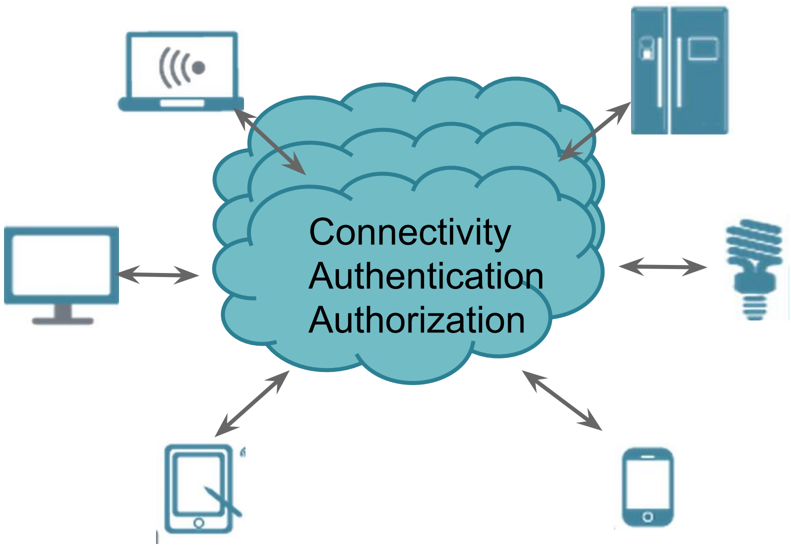
Decentralization

Decentralized deployment and peer-to-peer (p2p) communication are the main guiding principles for this work

Why?

- User privacy
- Service provider liability
- Support offline mode as a first-class citizen

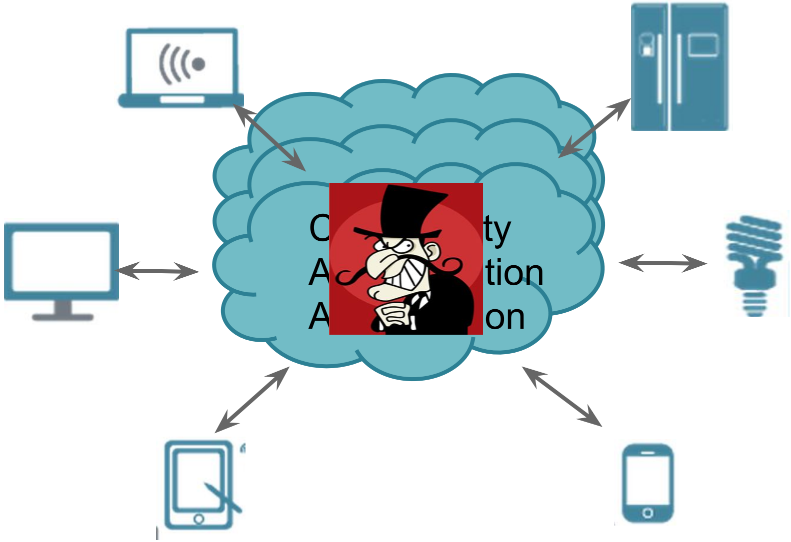
Centralized models



Upsides

- Centralized access management
- Seamlessly jump across networks
- Automatic software updates
- Data storage and backups
- Account recovery

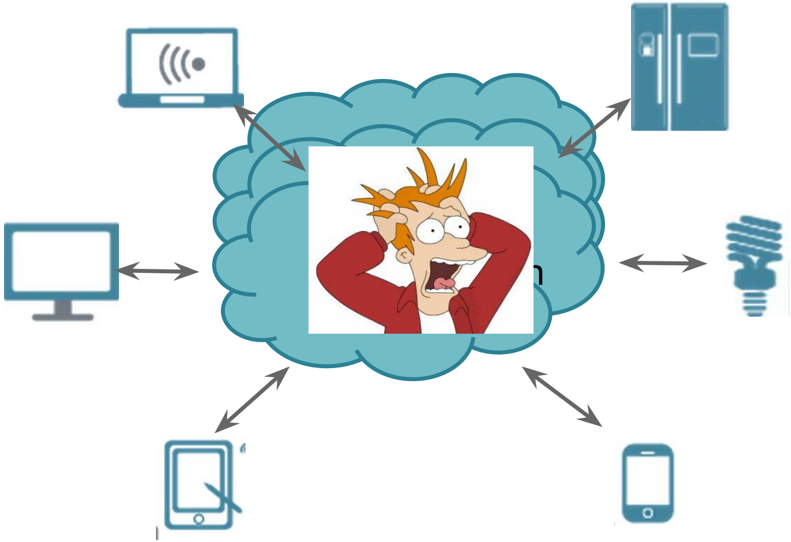
Centralized models



Downside 1: **User privacy**

- Private data leakage
- Tracking (in both digital and physical worlds)
- Growing concern all over the world
- Being taken seriously now
  - e.g., end-to-end encryption in WhatsApp and iMessage

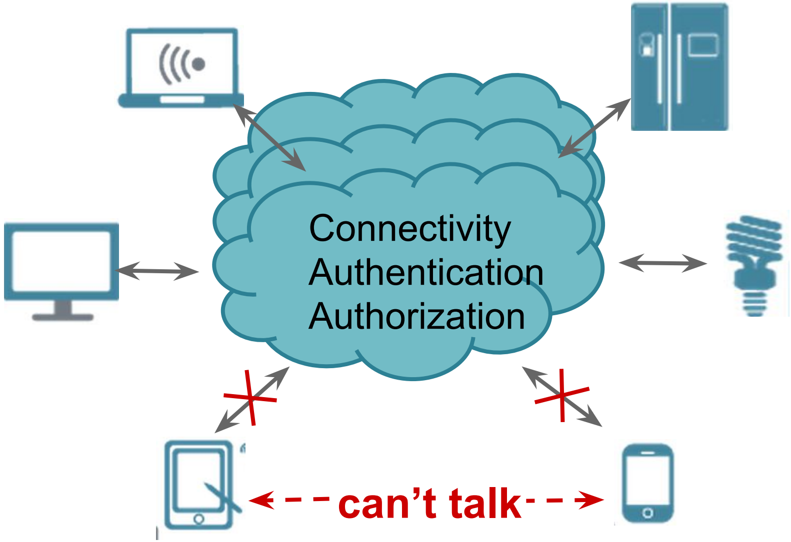
Centralized models



Downside 2: **Service provider liability**

- Subpoenas, break-ins, insiders threats
- Secure storage of personally identifiable information (PII) is a huge pain!

Centralized models



Downside 3: **Reliance on internet connectivity**

- Loss of functionality when internet access is not available
  - e.g., devices on an airplane
  - Internet is still a luxury for a significant chunk of the world
- Fundamentally inefficient

Our objective: Decentralization

- Define an ***egalitarian*** system where any principal can become an authority for some set of other principals
  - e.g., Alice may become an authority for all her home devices
- Minimize dependence on global services, e.g., CAs, proxies, etc.
- Maximize what can be achieved via peer-to-peer interactions

Use the cloud where it offers value!

- Account Recovery

  Delegate credentials to the cloud with usage restrictions

- Transparent Proxy

  Run a transparent cloud service that allows jumping across networks

- Data backup

  Backup a readonly / encrypted copy of the data in the cloud

- Distributing credentials

  Setup a cloud mailbox to distribute credentials, but NOT use them

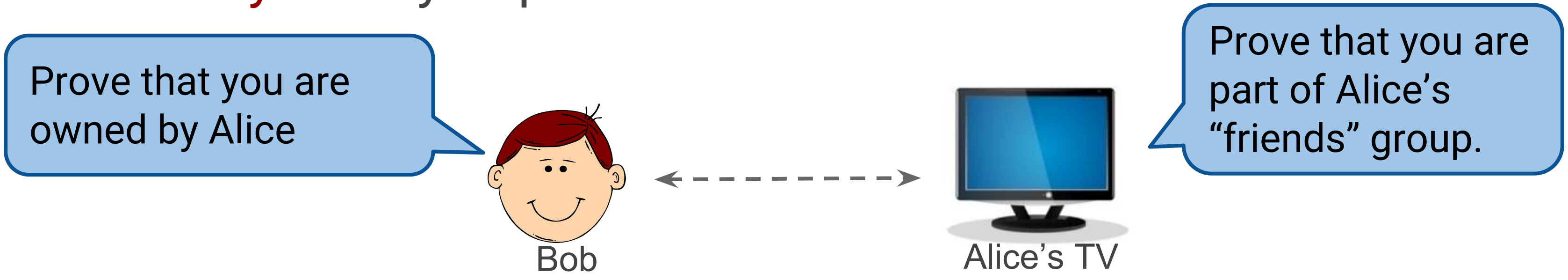
Summary

- Decentralized deployment
- Mutual authorization
- Fine-grained delegation
- Auditable access
- Revocation
- Ease of use

Mutual authorization

During any interaction, each end must verify that the other end is authorized in the context of the interaction

Mutuality is very important

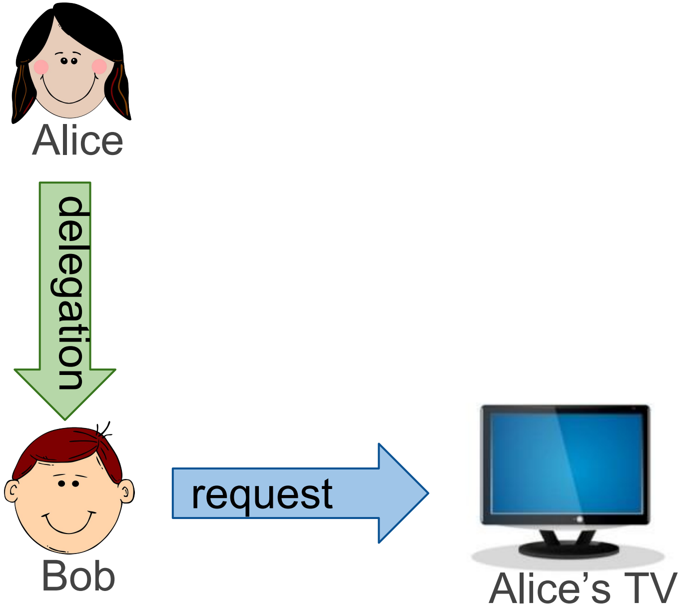


Mutual authentication may be important as well depending on the audit requirements

Delegation of authority

Model must support delegation of authority between principals

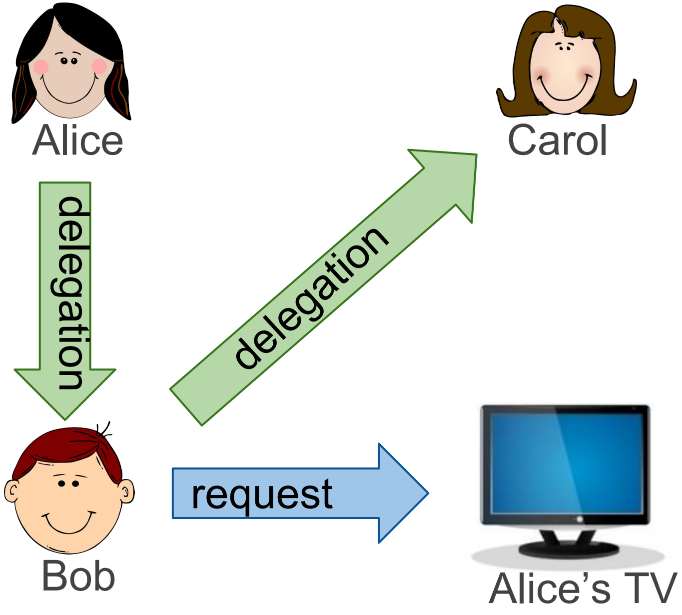
- under fine-grained constraints
  - *only until 6PM*
  - *only for this displaying photos*
  - *only when Alice is in nearby*
- across multiple hops
- in a convenient manner



Delegation of authority

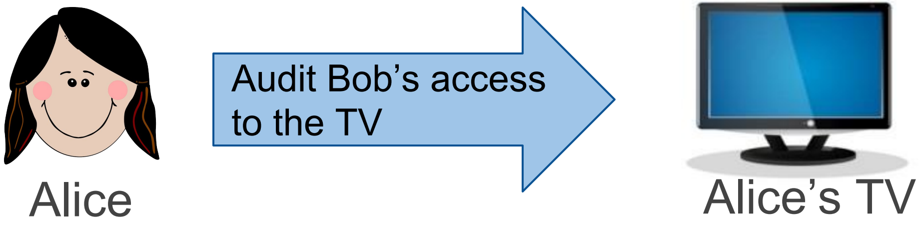
Model must support delegation of authority between principals

- under fine-grained constraints
  - *only until 6PM*
  - *only for this displaying photos*
  - *only when Alice is in nearby*
- across multiple hops
- in a convenient manner



Auditable access

Principals must be able to audit the use of the delegations granted by them



Auditing is the fallback when delegation restrictions cannot be properly codified

e.g., only watch PG-13 movies on the TV

Revocation

Principals must be able to revoke previously granted delegations

- Revoke Bob’s access to all of Alice’s devices
- Revoke all access held by a tablet, when it gets lost or stolen

This is a hard problem, lots of trade-offs

- Instantaneous vs. eventual revocation
- Communication, computation and storage overhead
- Supporting the P2P (Offline) scenario

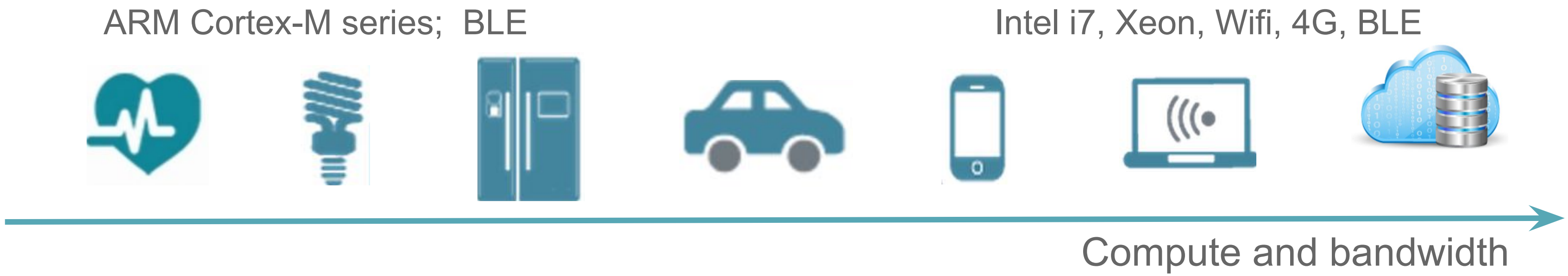
Ease of use

Systems with complex interfaces and mechanisms often have degraded security as users look for insecure workarounds

Therefore, authorization mechanisms must be easy to understand and use, both for end-users and system developers

Read: *Why Johnny can't encrypt?* *(J. D. Tygar and A. Whitten)*

Hardware constraints



IoT devices span a very wide hardware spectrum

For now, we do NOT restrict ourselves with hardware constraints

- Instead, focus on designing a general authorization architecture
- Hardware optimizations will hopefully follow

  (Read: CESEL: Securing a Mote for 20 Years)

Summary

- Decentralized deployment
- Mutual authorization
- Fine-grained delegation
- Auditable access
- Revocation
- Ease of use

Vanadium Authorization Model

Joint work with Asim Shankar, Gautham Thambidorai, and Dave Presotto

What is Vanadium?

Open source, cross-platform application framework for building secure, multi-device experiences

Components

- Identity and authorization model
- RPC framework
- Naming and discovery framework
- Peer-to-peer storage

What is Vanadium?

Open source, cross-platform application framework for building secure, multi-device experiences

Components

- Identity and authorization model
- RPC framework
- Naming and discovery framework
- Peer-to-peer storage

Rest of the lecture

- Vanadium authorization model primitives
  - Identity model
  - Delegation and revocation
  - Authentication protocols
  - Access control policies
- Application: Physical lock
- Practicalities and discussion

Principal

Represented by a unique digital signature public and private key pair (P, S)

- Private key is **never** shared over the network
- Ideally held in a TPM on the device

Fine-grained: Each app, process, service is a different principal

- Distinguish between Alice's son Bob's tablet's Farmville app & Alice's daughter Carol's phone's Amazon app

Blessings

Each principal has a set of hierarchical human-readable strings bound to it, called ***blessings***
e.g., Alice's television (P_{TV}, S_{TV}) may have blessings:

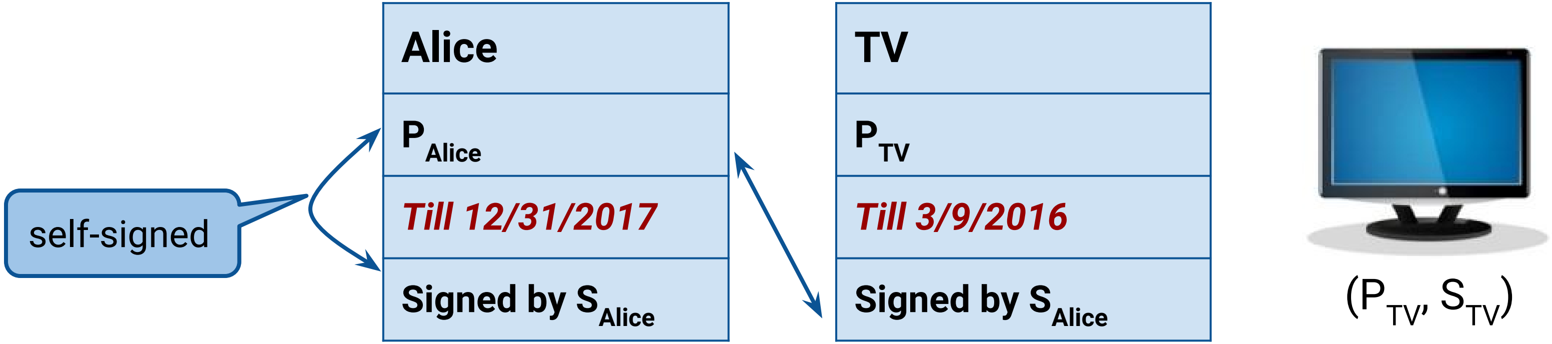
- Alice/TV
- Samsung/Products/TV/123

Principals are authenticated and authorized based on their blessings
e.g., Authorize all principals with blessings prefixed with Alice

Blessings

Blessings are certificate chains bound to the principal's public key

Each certificate has a Name, PublicKey, Caveats and Signature



Very simple certificate format!

The "Bless" operation

Extend one of your Blessings and bind it to another principal

(P_{Alice}, S_{Alice})

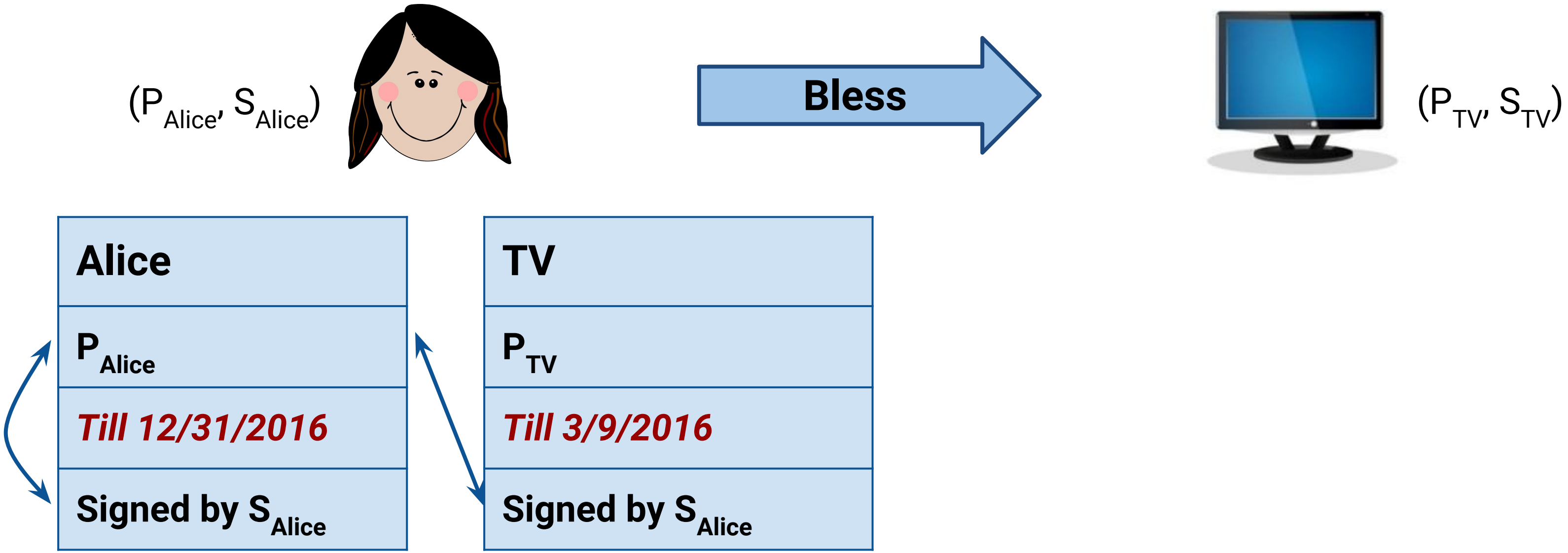
Bless

(P_{TV}, S_{TV})

Alice
P_{Alice}
Till 12/31/2016
Signed by S_{Alice}

The “Bless” operation

Extend one of your Blessings and bind it to another principal



The “Bless” operation

Extend one of your Blessings and bind it to another principal

(P_{Alice}, S_{Alice})

Bless →

(P_{TV}, S_{TV})

Alice
P_{Alice}
Till 12/31/2016
Signed by S_{Alice}

TV
P_{TV}
Till 3/9/2016
Signed by S_{Alice}

Dynamic identity creation OR Bound capability grant!

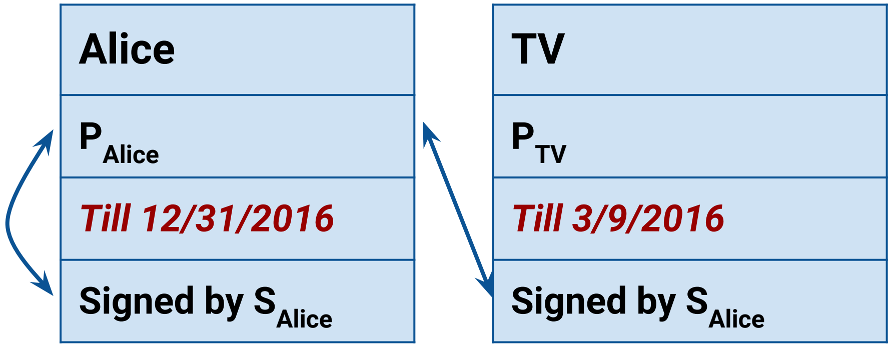
Blessings: Auditability and Binding

Blessings:

- Are bound to a private key that never leaves the device
- Can only be delegated by extending to other private keys
- Encapsulate an auditable delegation trail

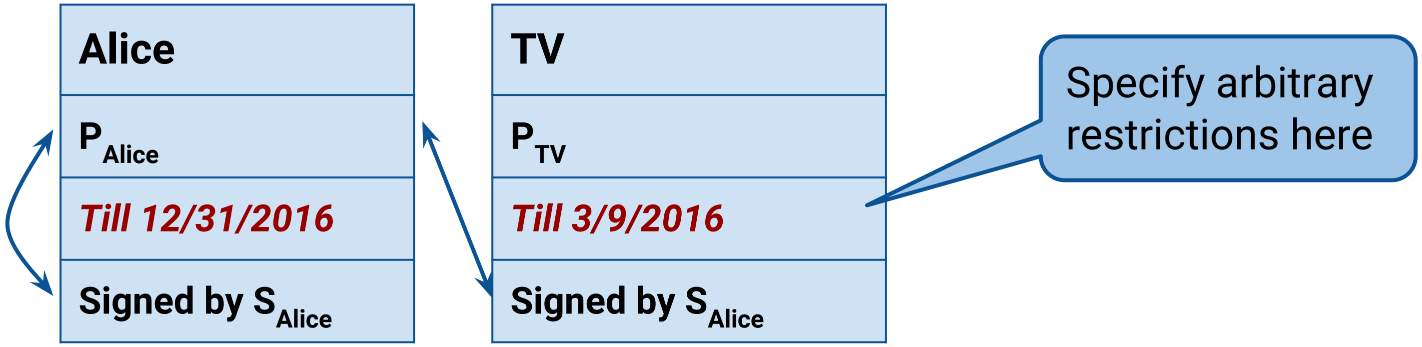
Caveats

But Alice wants her TV to *only* access Youtube, NOT her Bank!



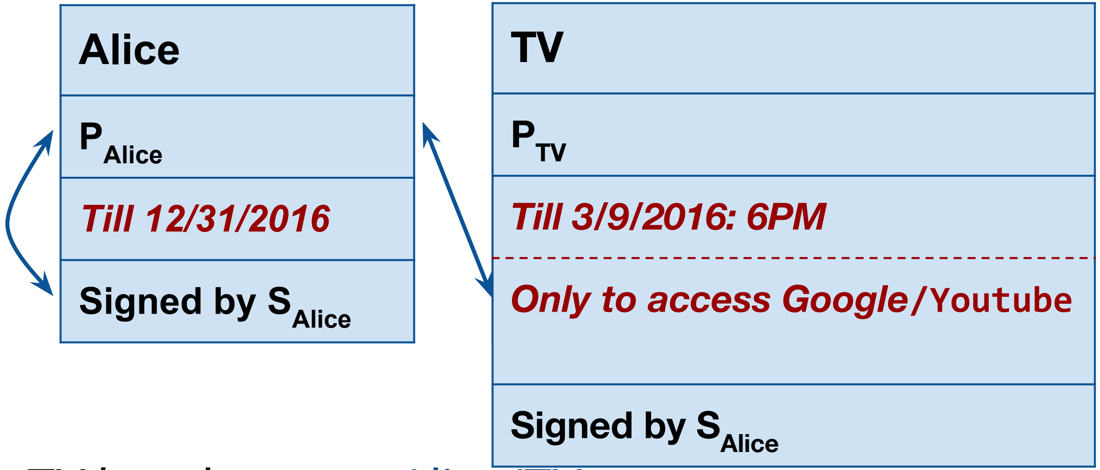
Caveats

But Alice wants her TV to *only* access Youtube, NOT her Bank!



Caveats

But Alice wants her TV to *only* access Youtube, NOT her Bank!



TV has the name Alice/TV

- *as long as the time is before 3/9/2016: 6PM*
- *as long as the service being accessed is* Google/Youtube

Caveats are powerful

Macaroons: Cookies with Caveats for Decentralized Authorization, Politz et al., NDSS 14

Services can define their own caveats, e.g., bless the valet so that:

- valet is only authorized to drive for < 5 miles
- only for the next 3 hours
- cannot access trunk or infotainment system
- but can access GPS

Validated by the target service (first-party) when the blessing is used to make a request (first-party caveats)

Third-party Caveats

- Caveats that must be validated by a specific third-party
- Target service (first-party) only expects a “discharge” (proof) that the caveat has been validated by the specific third-party

Third-party Caveats

- Caveats that must be validated by a specific third-party
- Target service (first-party) only expects a "discharge" (proof) that the caveat has been validated by the specific third-party

Third-party Caveat

ID: <content hash>

Restriction: within 10m proximity

Loc: Alice/Proximity

Verification Key: $P_{Proximity}$

Third-party Caveats

- Caveats that must be validated by a specific third-party
- Target service (first-party) only expects a “discharge” (proof) that the caveat has been validated by the specific third-party

Third-party Caveat

ID: <content hash>

Restriction: within 10m proximity

Loc: Alice/Proximity

Verification Key: $P_{Proximity}$

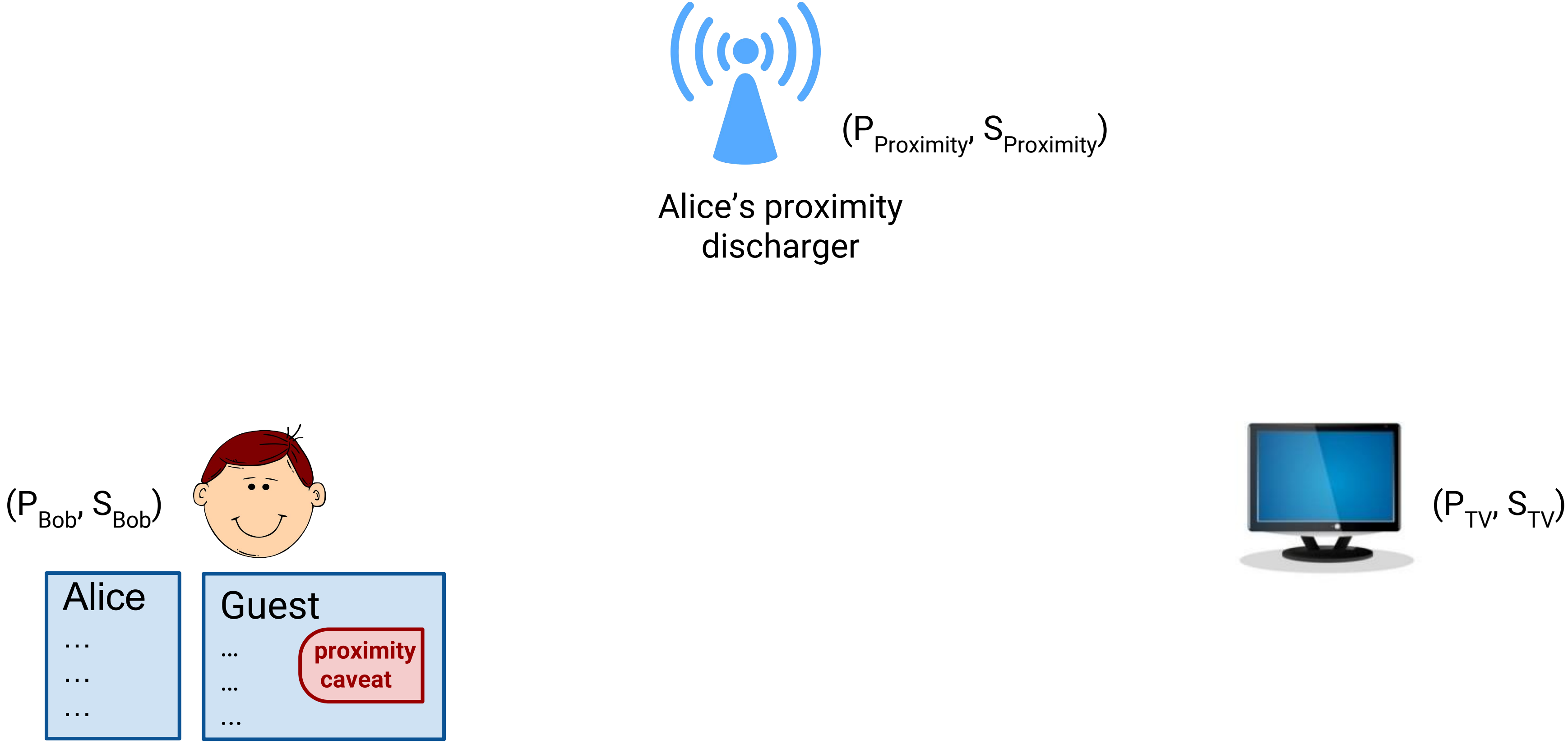
Third-party Discharge

ID: <same as caveat.ID>

Caveat: for next 1 minute

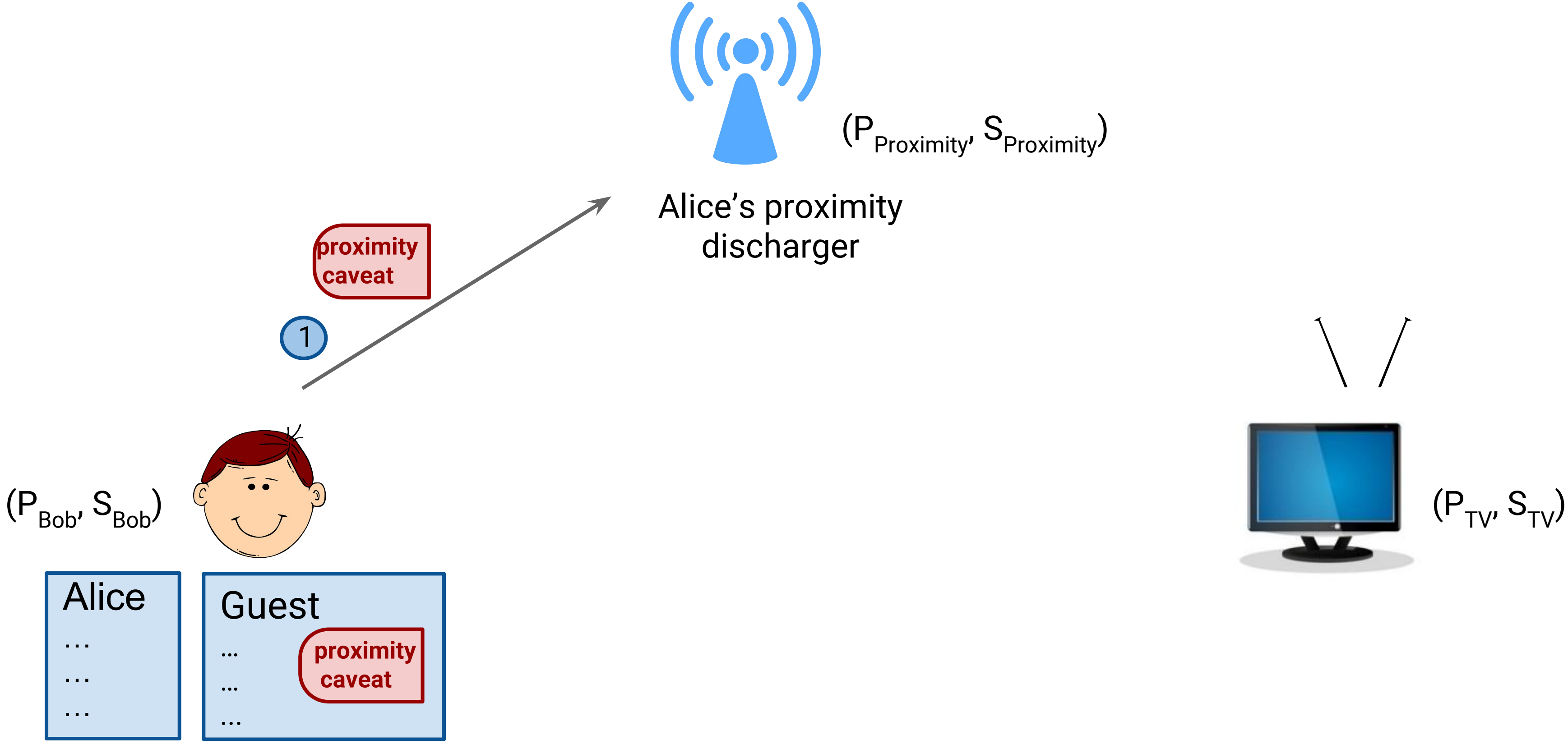
Signed by $S_{Proximity}$

Mechanics



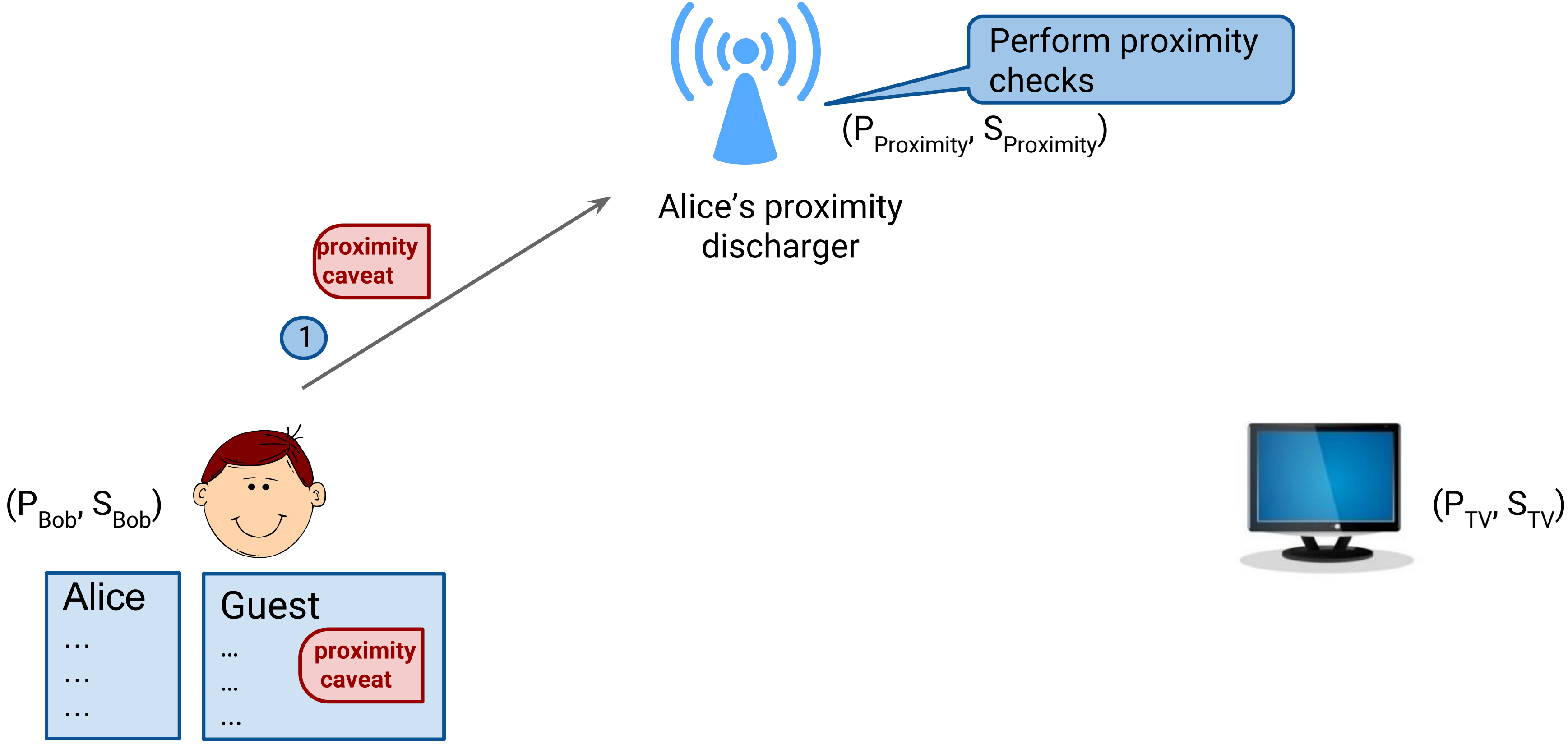
($P_{Proximity}$, $S_{Proximity}$)
Alice's proximity discharger
(P_{Bob}, S_{Bob})
Alice
…
…
…
Guest
…
…
…
proximity caveat
(P_{TV}, S_{TV})

Mechanics



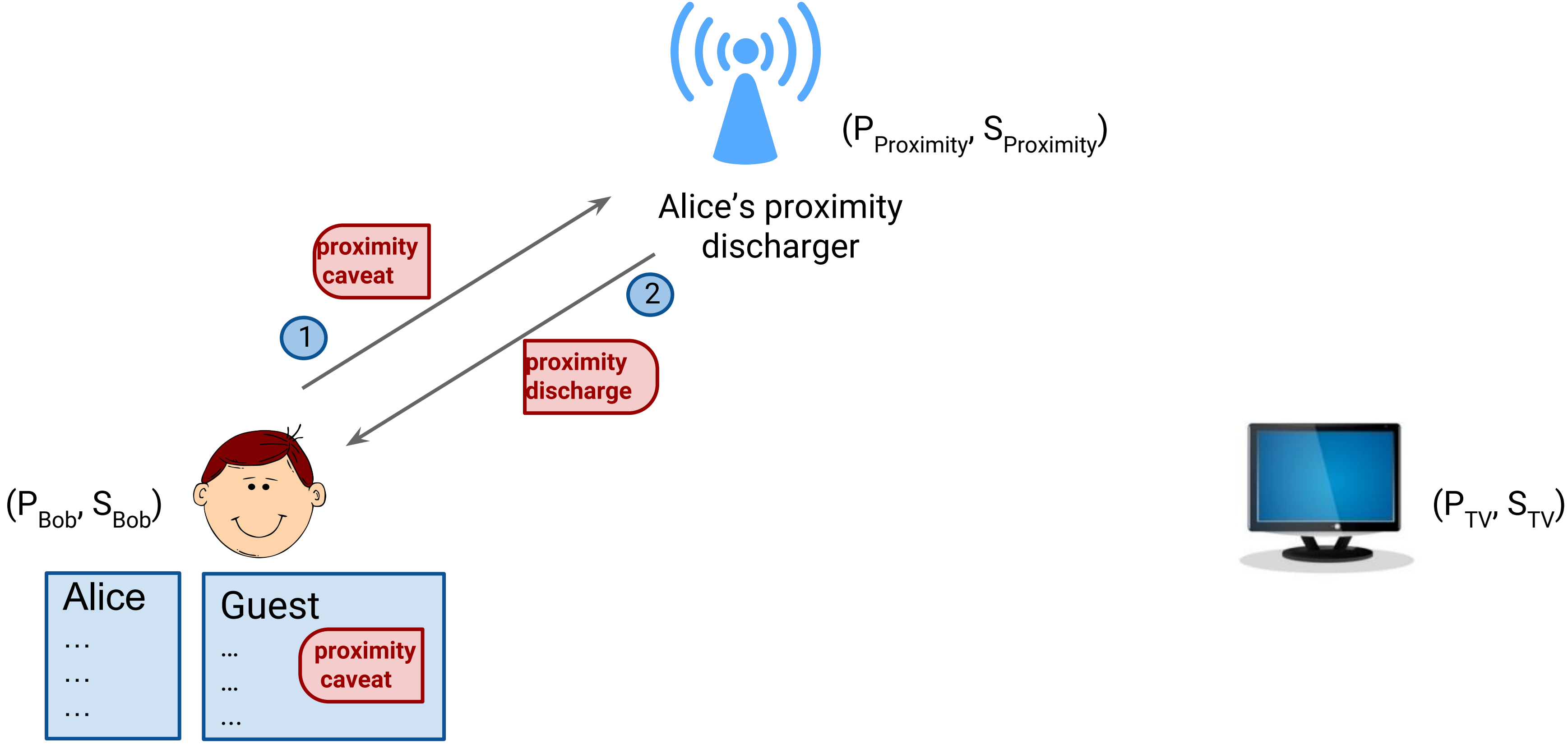
(P_Proximity, S_Proximity)
Alice's proximity discharger
proximity caveat
1
(P_Bob, S_Bob)
Alice
…
…
…
Guest
…
…
…
proximity caveat
(P_TV, S_TV)

Mechanics



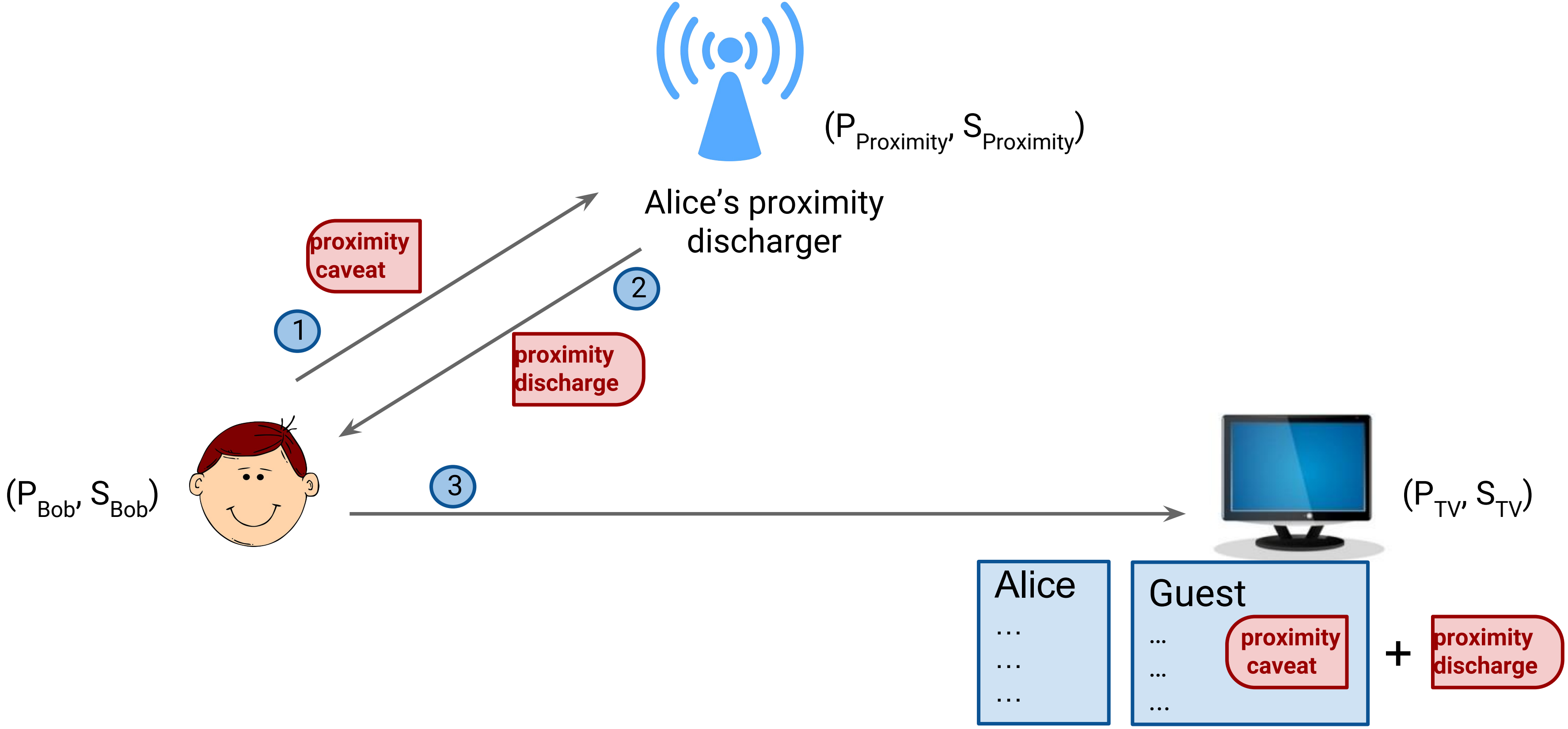
Perform proximity checks
($P_{Proximity}$, $S_{Proximity}$)
Alice's proximity discharger
proximity caveat
1
(P_{Bob}, S_{Bob})
Alice
…
…
…
Guest
…
…
…
proximity caveat
(P_{TV}, S_{TV})

Mechanics



$(P_{Proximity}, S_{Proximity})$
Alice's proximity discharger
proximity caveat
1
2
proximity discharge
(P_{Bob}, S_{Bob})
(P_{TV}, S_{TV})
Alice
…
…
…
Guest
…
…
…
proximity caveat

Mechanics



(P_Proximity, S_Proximity)
Alice's proximity discharger
proximity caveat
1
2
proximity discharge
(P_Bob, S_Bob)
3
(P_TV, S_TV)
Alice
…
…
…
Guest
…
…
…
proximity caveat
+
proximity discharge

Third-party Caveat Examples

- Social networking restrictions
  - GooglePlus must assert membership in “work” circle
  - Or, must be my friend on Facebook
- Parental controls
  - Kids can watch TV only if Mom approves
  - Mom may discharge with a third-party caveat to dad!
- **Revocation**

Revocation

Revocation: Existing approaches

- Certificate revocation lists (CRLs)
  - Validating principals must periodically update CRL
  - Revocation is not instantaneous
  - CRLs tend to get large (delta-CRLs offer a reasonable fix)
- Online certificate status protocol (OCSP)
  - Onus of making OCSP queries is on the validating principal
  - Affects latency per request
  - Another vector for DOS attacks

Revocation: Existing approaches

- Recency evidence
  - “Can we eliminate certificate revocation lists?” ---- Rivest 98
  - Certificate is valid only when accompanied with “recency evidence” supplied by the requestor
  - Recency evidence may be re-validated certificate or a freshly issued certificate

Revocation: Third-party caveat approach

In essence, Rivest's recency proofs idea

- Blessings carry third-party caveats specifying revocation requirements
- Caveat is discharged by a revocation service trusted by the issuer
- Requester must obtain the discharge and supply it along with the blessing

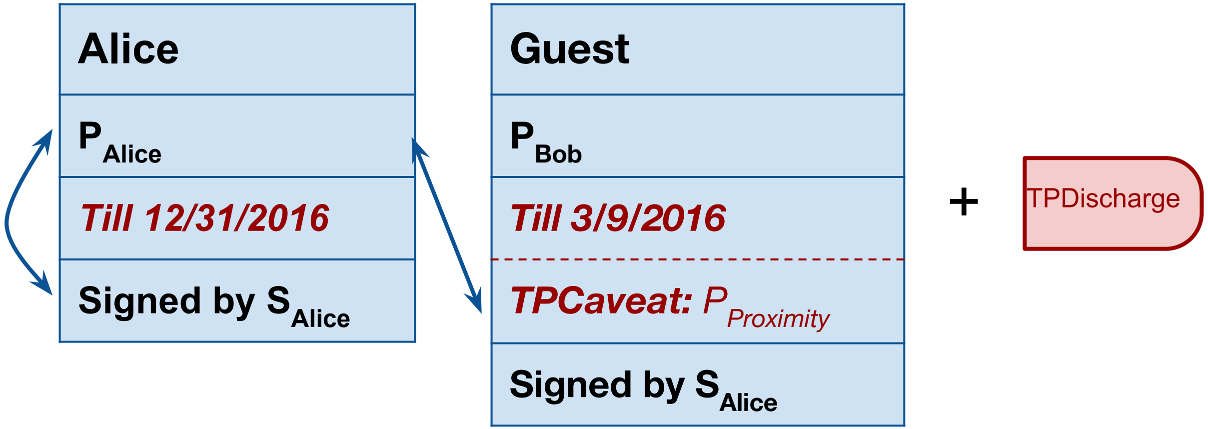
Supports instantaneous revocation

But, places a connectivity requirement on the requester

Validating Blessings

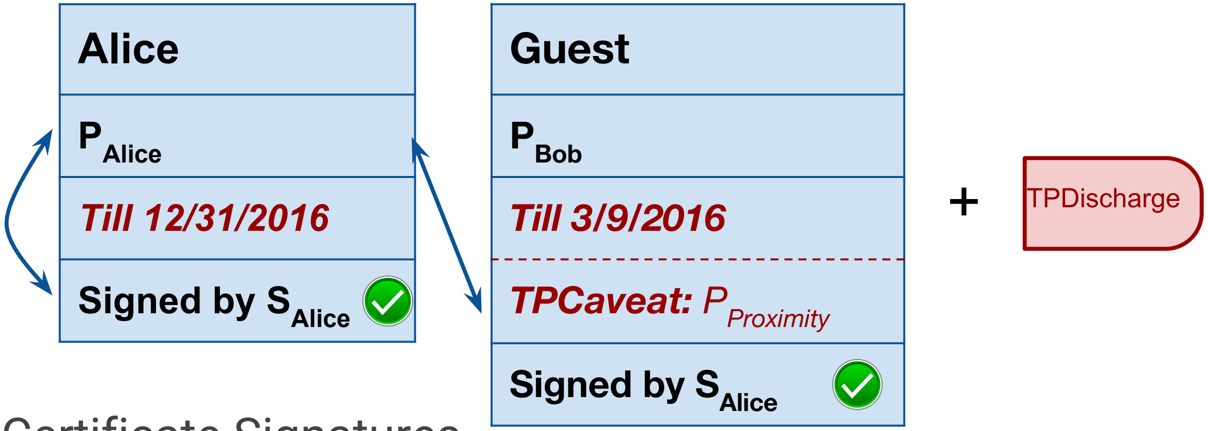
Validating blessings

How does the TV validate Bob's blessings?



Validating blessings

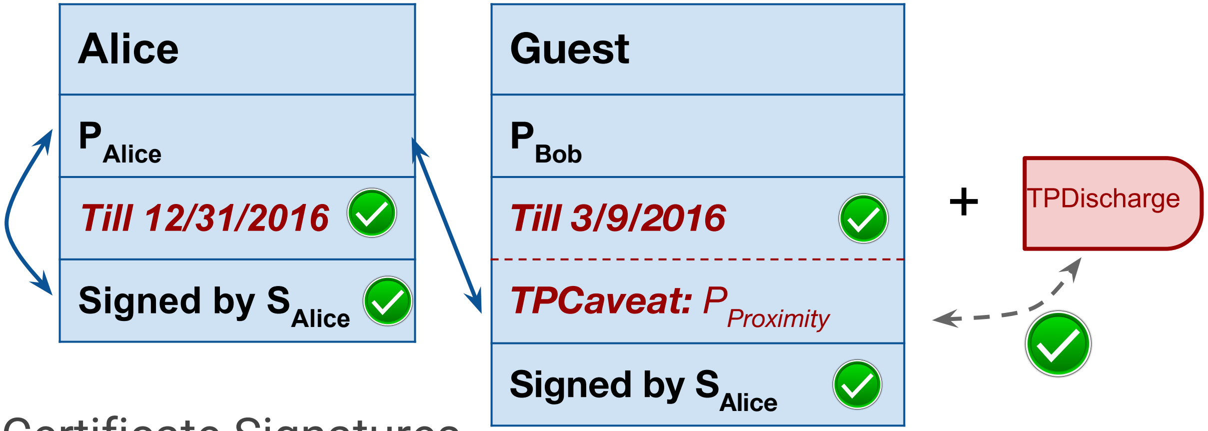
How does the TV validate Bob's blessings?



1. Verify Certificate Signatures

Validating blessings

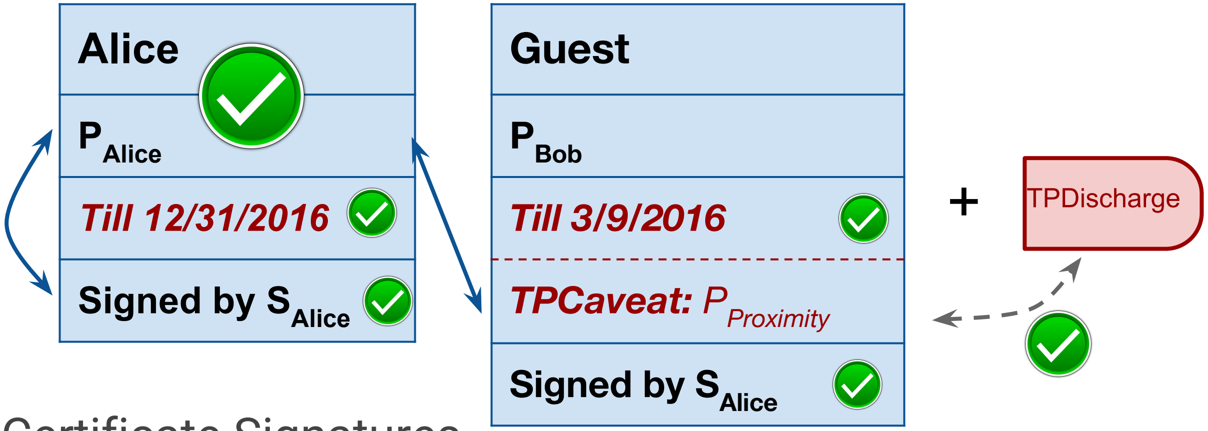
How does the TV validate Bob's blessings?



1. Verify Certificate Signatures
2. Validate all first-party and third-party caveats

Validating blessings

How does the TV validate Bob's blessings?



1. Verify Certificate Signatures
2. Validate all first-party and third-party caveats
3. Verify that the blessing root is recognized

Blessing root

The first certificate of a blessing is self-signed

Anyone can forge a blessing by creating a self-signed certificate

Alice

$P_{Attacker}$

Till 12/31/2016

Signed by $S_{Attacker}$

How do we prevent this forgery?

Blessing root

Blessing root is the name and public of the first certificate of the blessing

Principals maintain a list of recognized blessing roots

Only blessings with recognized roots are considered valid

e.g., blessing with root ($P_{attacker}$, Alice) is rejected by Alice's TV

Public key	Name
P_{Alice}	Alice
$P_{Samsung}$	Samsung

Roots recognized by Alice's TV

Blessing root

List of recognized roots is similar to the list of trusted CAs in Web browsers

But there are some key differences

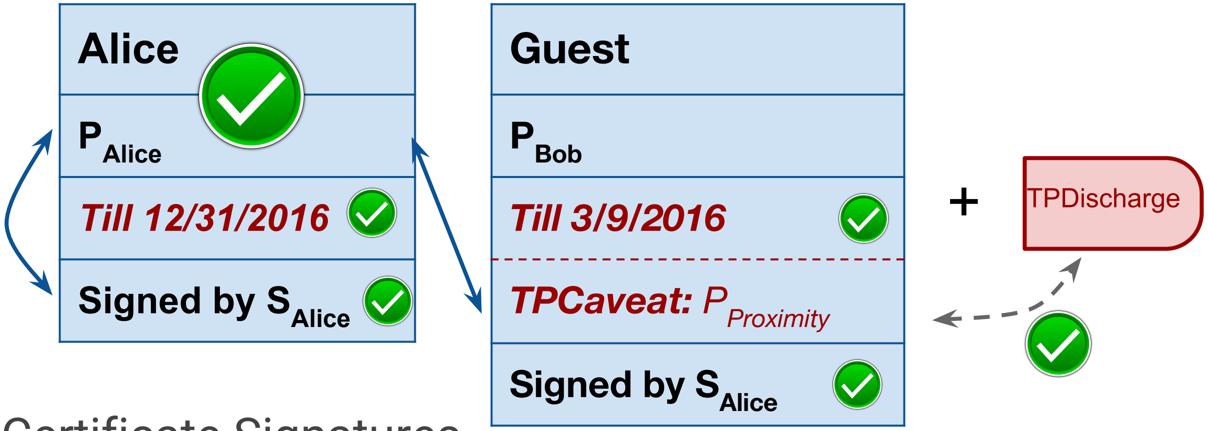
- Any principal can become a blessing root
- Different principals may recognize different roots

  e.g., Alice' TV may recognize (P_{Alice}, Alice) but Bob's TV may not
- A principal is recognized for a specific name

  e.g., Alice's TV recognizes P_{Alice} for Alice and $P_{Samsung}$ for Samsung

Validating blessings

How does the TV validate Bob’s blessings?



1. Verify Certificate Signatures
2. Validate all first-party and third-party caveats
3. Verify that the blessing root is recognized

Bob can be recognized as Alice/Guest

Authentication and Authorization

All communication must be encrypted,
mutually authenticated and authorized

Authentication and Authorization

Client: Initiator of request

Server: Responder of request

Mutual Authentication

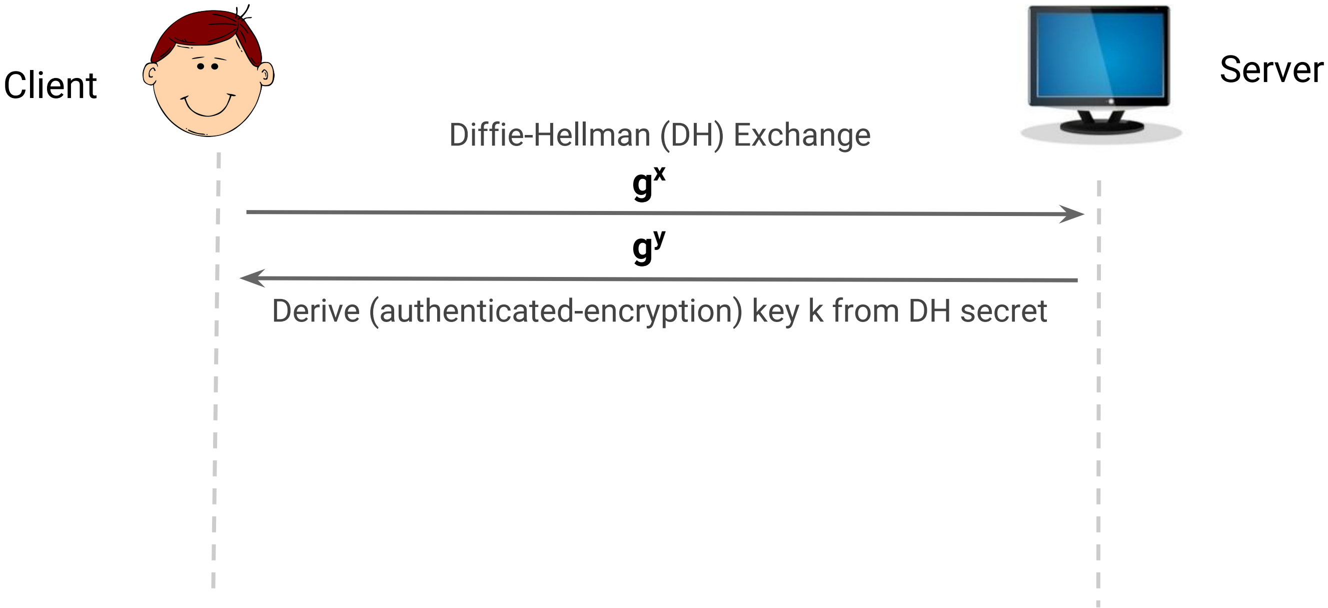
Each end learns the other end's blessings and is convinced that the other end possesses the corresponding private key

Mutual Authorization

Each end validates the other end's blessings and evaluates the blessing names against an access control policy

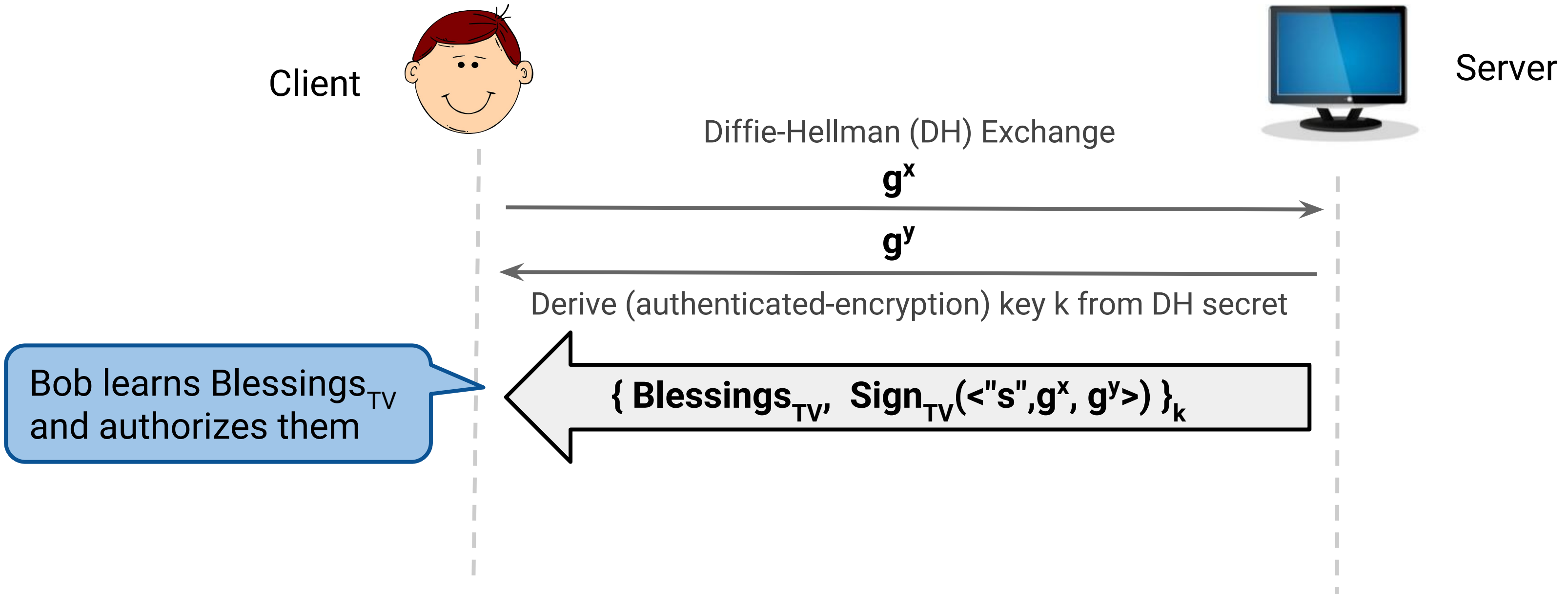
Mutual authentication protocol

SIGMA: The 'SIGn-and-MAc' Approach to Authenticated Diffie-Hellman, Krawczyk et al., CRYPTO 03



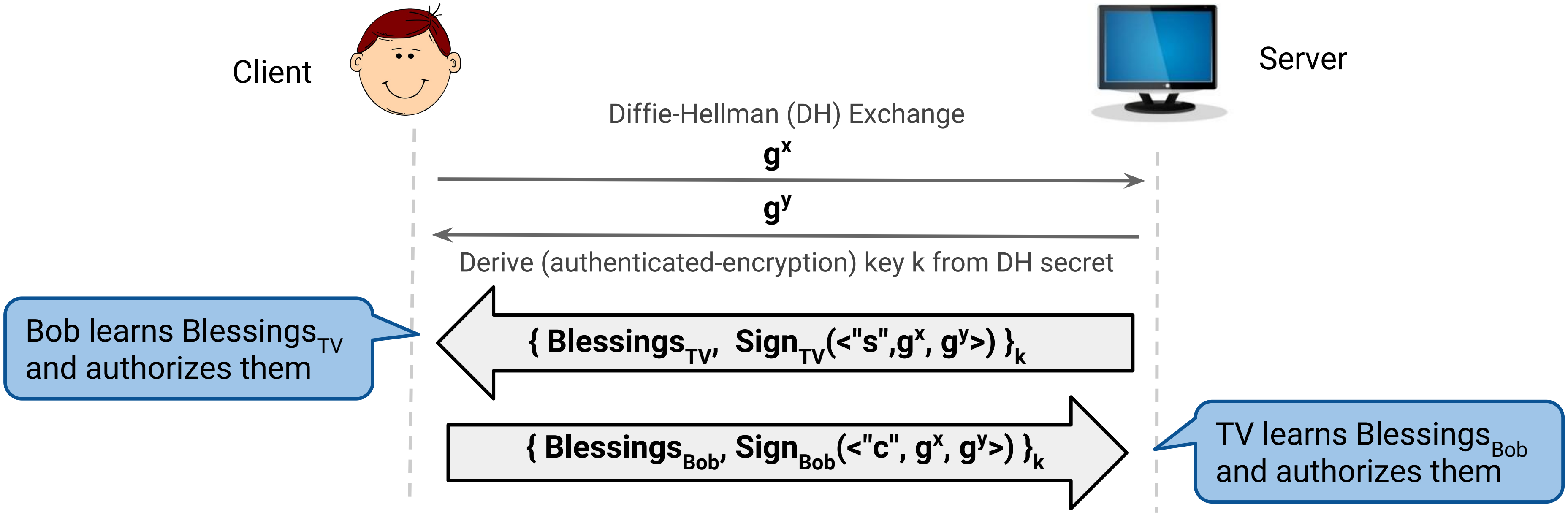
Mutual authentication protocol

SIGMA: The 'SIGn-and-MAc' Approach to Authenticated Diffie-Hellman --- Krawczyk et al., CRYPTO'03



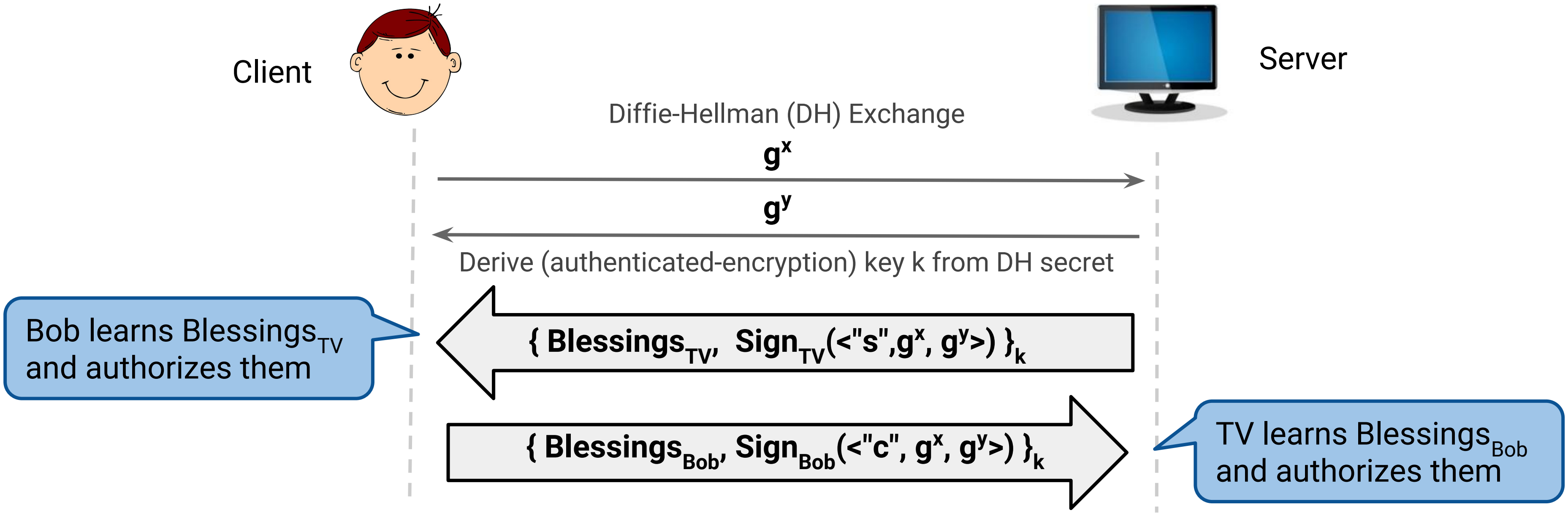
Mutual authentication protocol

SIGMA: The 'SIGn-and-MAc' Approach to Authenticated Diffie-Hellman --- Krawczyk et al., CRYPTO'03



Mutual authentication protocol

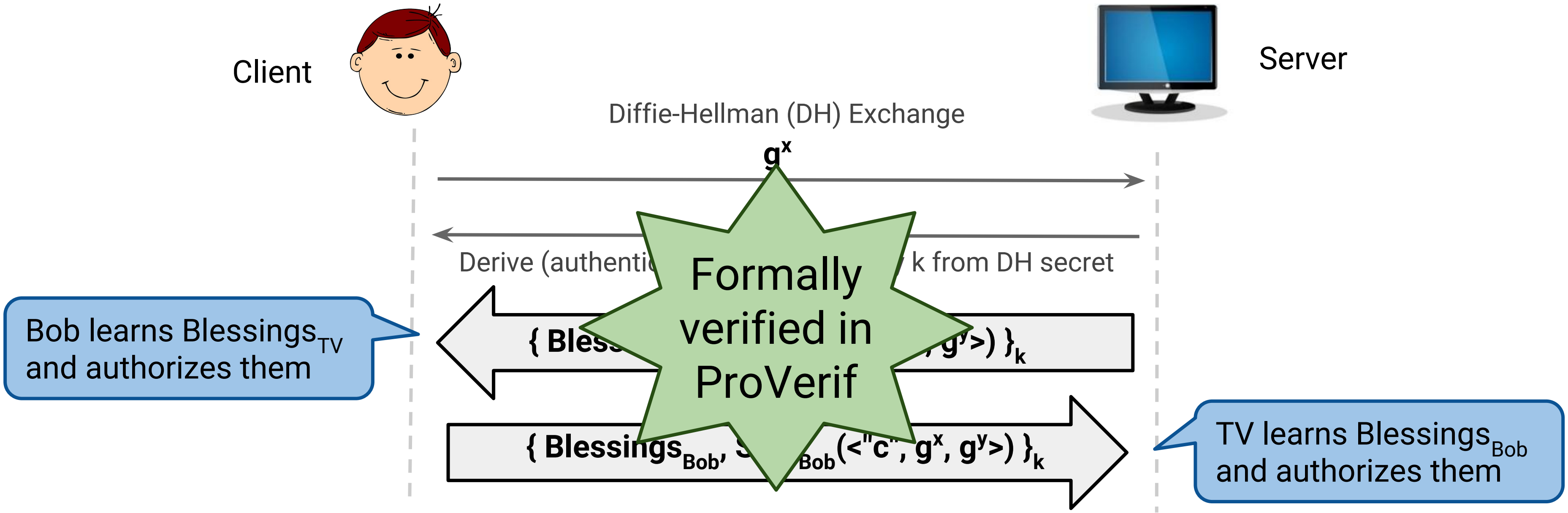
SIGMA: The 'SIGn-and-MAc' Approach to Authenticated Diffie-Hellman --- Krawczyk et al., CRYPTO’03



Server presents its blessings *before* the client

Mutual authentication protocol

SIGMA: The 'SIGn-and-MAc' Approach to Authenticated Diffie-Hellman --- Krawczyk et al., CRYPTO’03

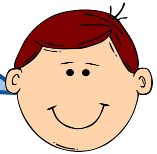


Server presents its blessings *before* the client

Private mutual authentication

Neither the server nor the client wants to present its blessings first

I will only reveal my name to Alice/TV



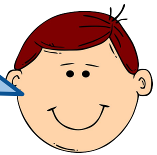
I only reveal my name to delegates of Alice

Can we resolve this deadlock?

Private Mutual Authentication

Neither the server nor the client wants to present its blessings first

I will only reveal my name to Alice/TV



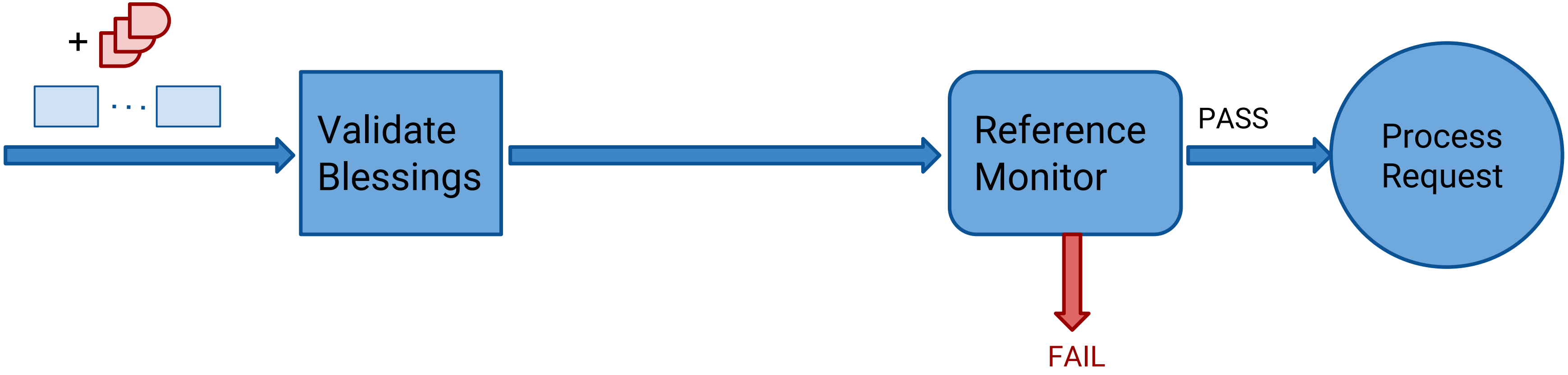
I only reveal my name to delegates of Alice

Can we resolve this deadlock?

Yes, using *identity-based encryption* (tomorrow's lecture)

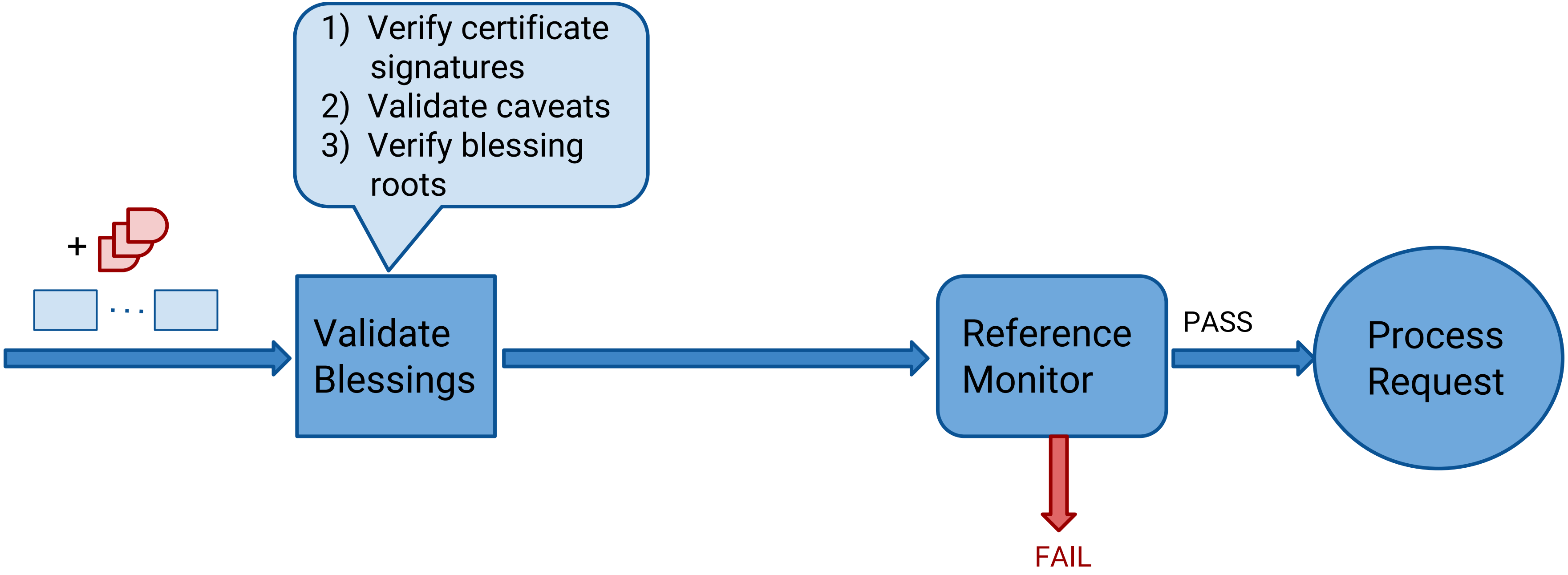
Authorization

Authorization policies are based on blessing names



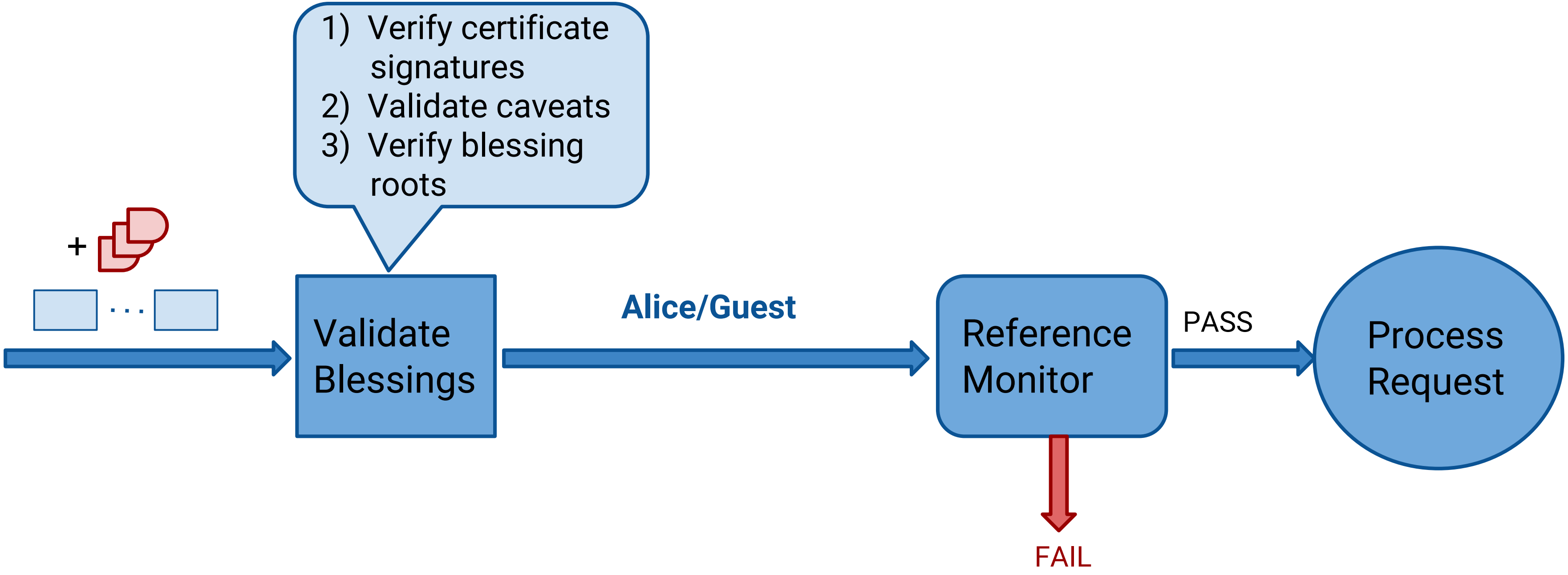
Authorization

Authorization policies are based on blessing names



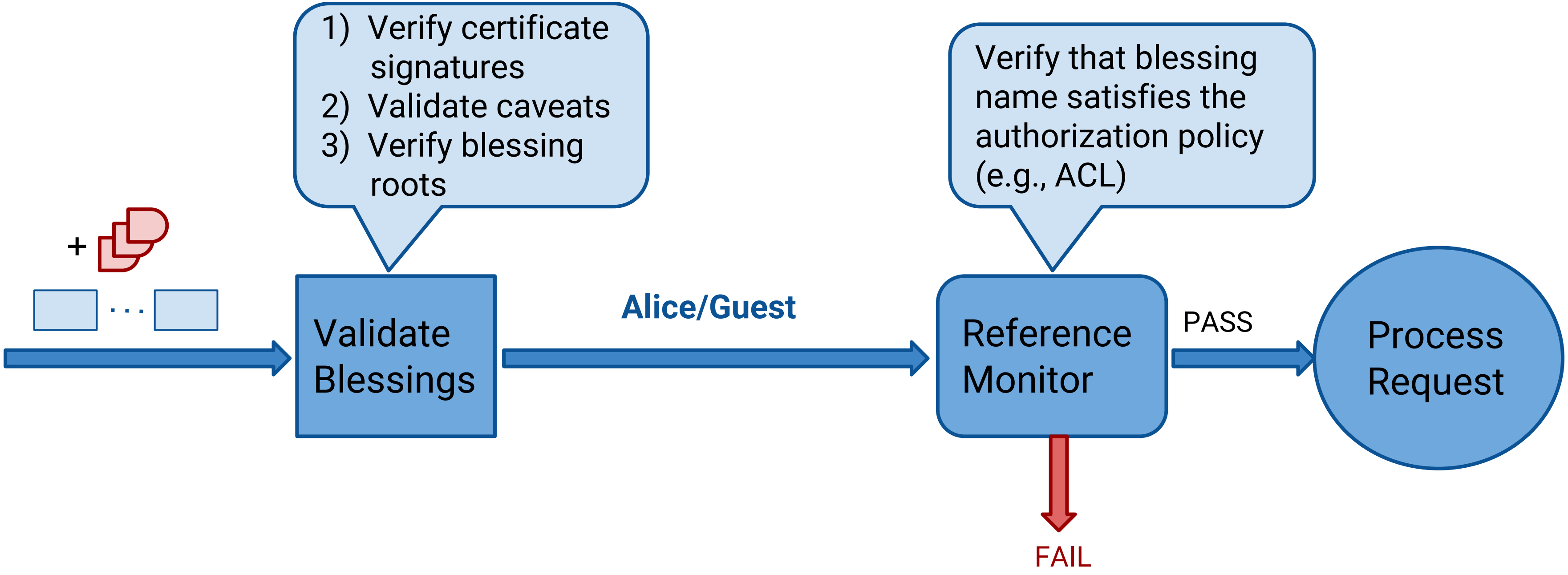
Authorization

Authorization policies are based on blessing names



Authorization

Authorization policies are based on blessing names



Access control policies

Explicitly specify set of authorized blessing names in an ACL

Policy for Alice’s TV

Label	**Policy**
Photos	**Allow**: Alice
Movies	**Allow**: Alice/Friends

Access control policies

Explicitly specify set of authorized blessings

Policy for Alice's TV

Label	Policy
Photos	**Allow**: Alice
Movies	**Allow**: Alice/Friends

This is actually a blessing prefix and is matched by **extensions,** e.g., Alice/Guest, Alice/TV/app

(We will go over the rationale for this tomorrow)

Case Study: Physical Lock

Lockitron

GOji

UNI KEY

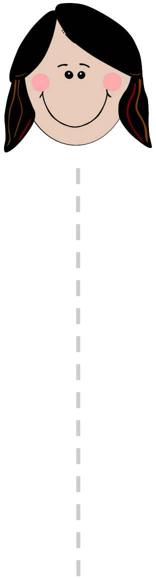
august

Why Smart Locks?

- Remote locking/unlocking
- Keyless proximity-based access
- Maintain an audit log of who got in
- Mint new (virtual) keys and share with others
- Some also have a camera that will take the visitors picture

Lock setup

Blessings
Alice



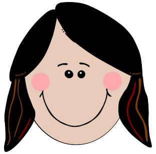
Blessings
LockCorp/1234

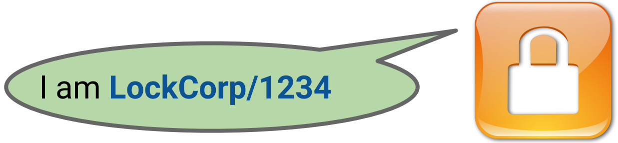
Lock setup


Blessings
Alice
Out of Band
<token>, <LockWiFi>
Blessings
LockCorp/1234

Lock setup

Blessings
Alice

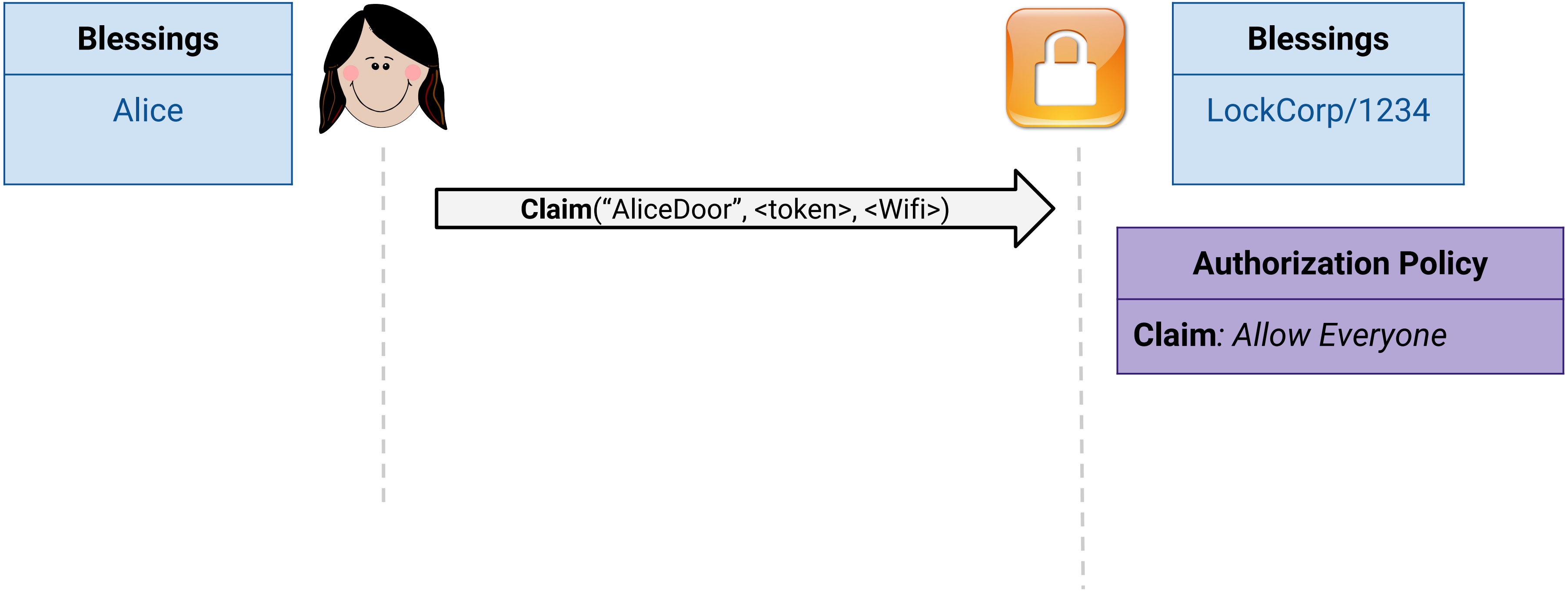




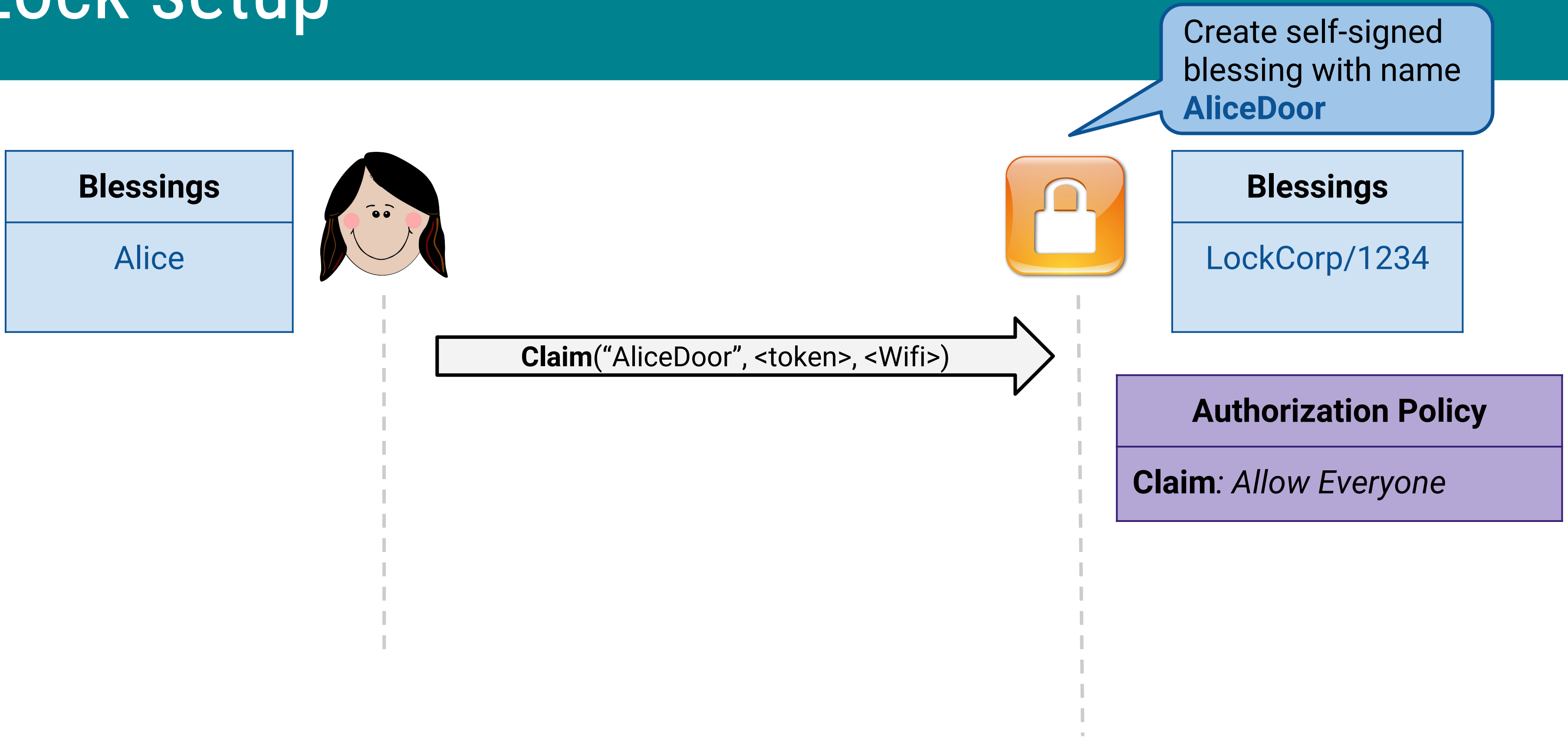
Blessings
LockCorp/1234

Authorization Policy
Claim: *Allow Everyone*

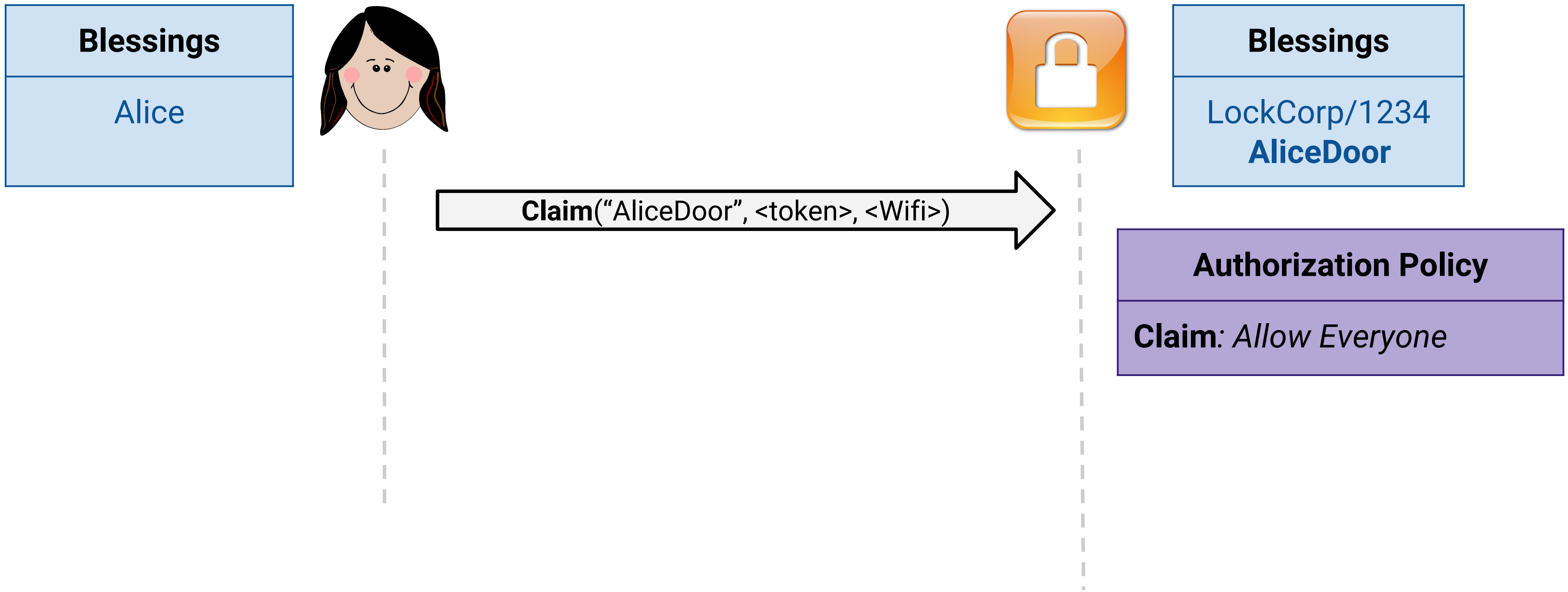
Lock setup


Blessings
Alice
Blessings
LockCorp/1234
Claim("AliceDoor", <token>, <Wifi>)
Authorization Policy
Claim: Allow Everyone

Lock setup

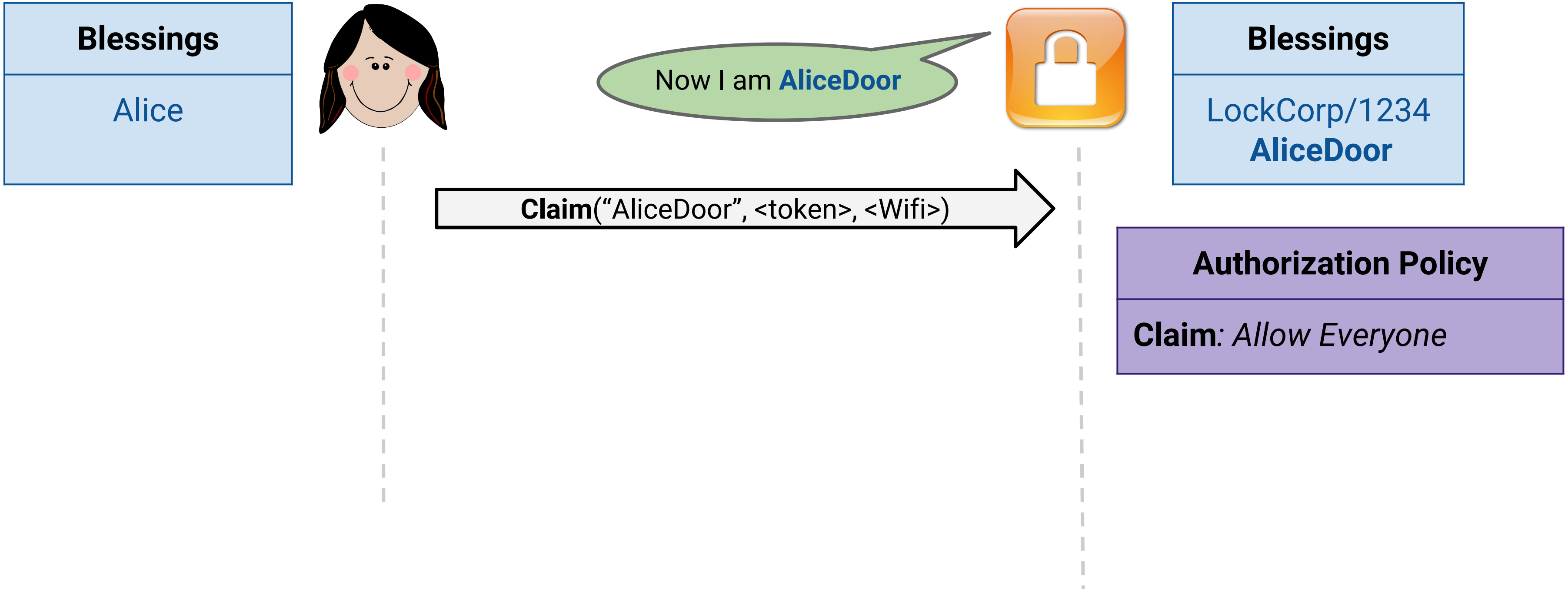

Create self-signed blessing with name **AliceDoor**
Blessings
Alice
Blessings
LockCorp/1234
Claim(“AliceDoor”, <token>, <Wifi>)
Authorization Policy
Claim: *Allow Everyone*

Lock setup



Blessings
Alice
Blessings
LockCorp/1234
AliceDoor
Claim("AliceDoor", <token>, <Wifi>)
Authorization Policy
Claim: Allow Everyone

Lock setup

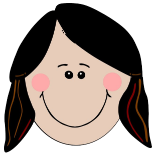


Blessings
Alice
Now I am AliceDoor
Claim("AliceDoor", <token>, <Wifi>)
Blessings
LockCorp/1234
AliceDoor
Authorization Policy
Claim: Allow Everyone

Lock setup

Blessings

Alice
AliceDoor/Key



Claim(“AliceDoor”, <token>, <Wifi>)

Blessings

LockCorp/1234
AliceDoor

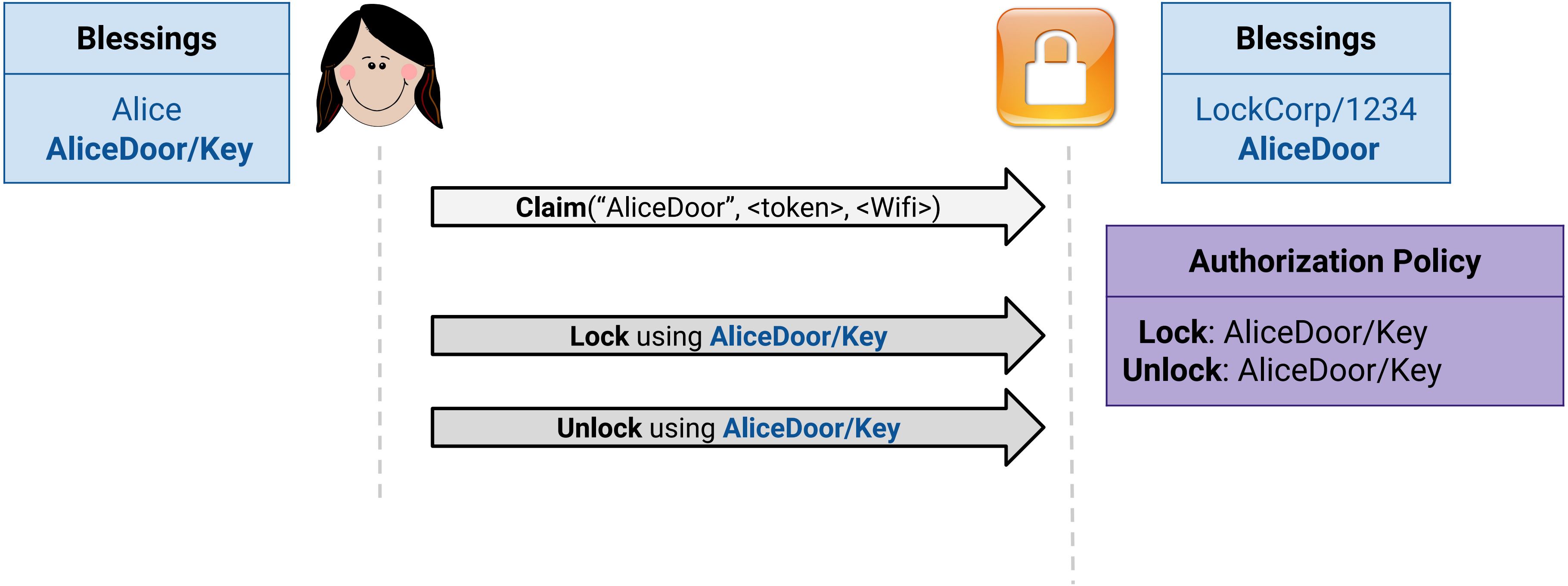
Authorization Policy

Claim: *Allow Everyone*

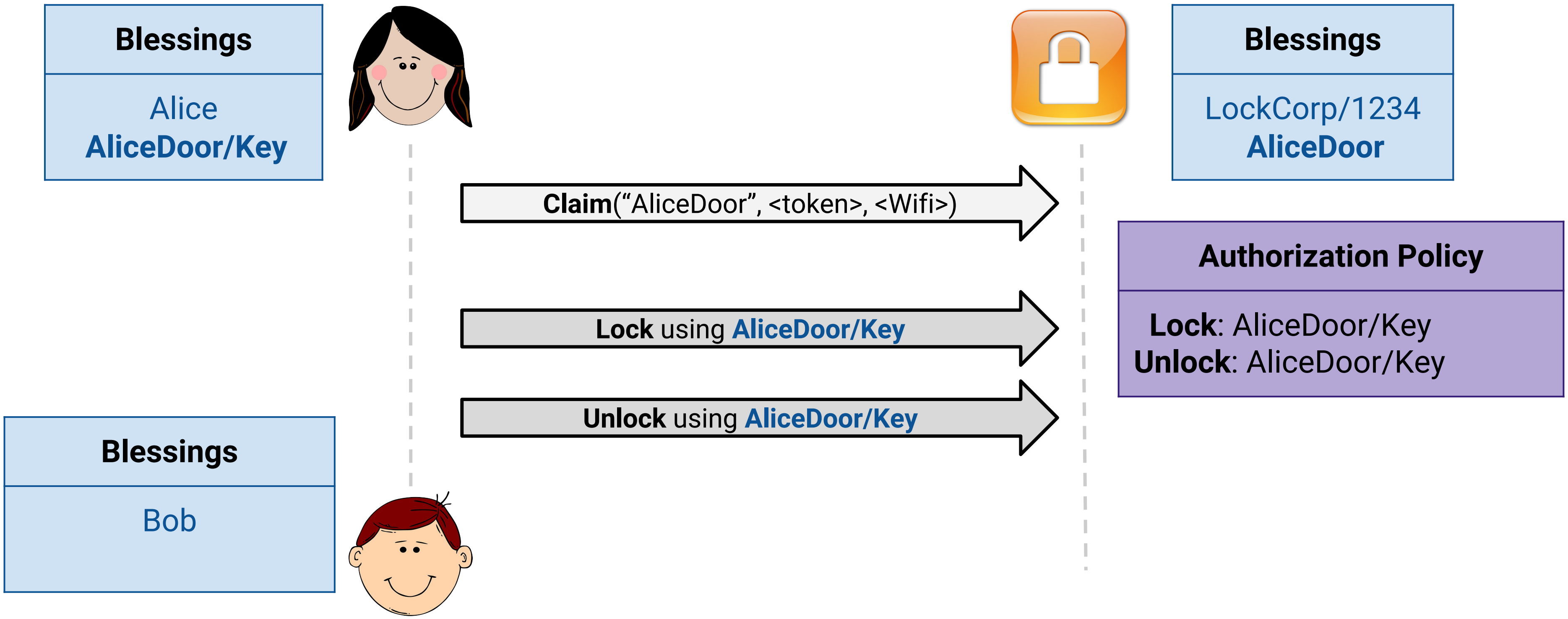
Lock setup


Blessings
Alice
AliceDoor/Key
Claim("AliceDoor", <token>, <Wifi>)
Blessings
LockCorp/1234
AliceDoor
Authorization Policy
Lock: AliceDoor/Key
Unlock: AliceDoor/Key

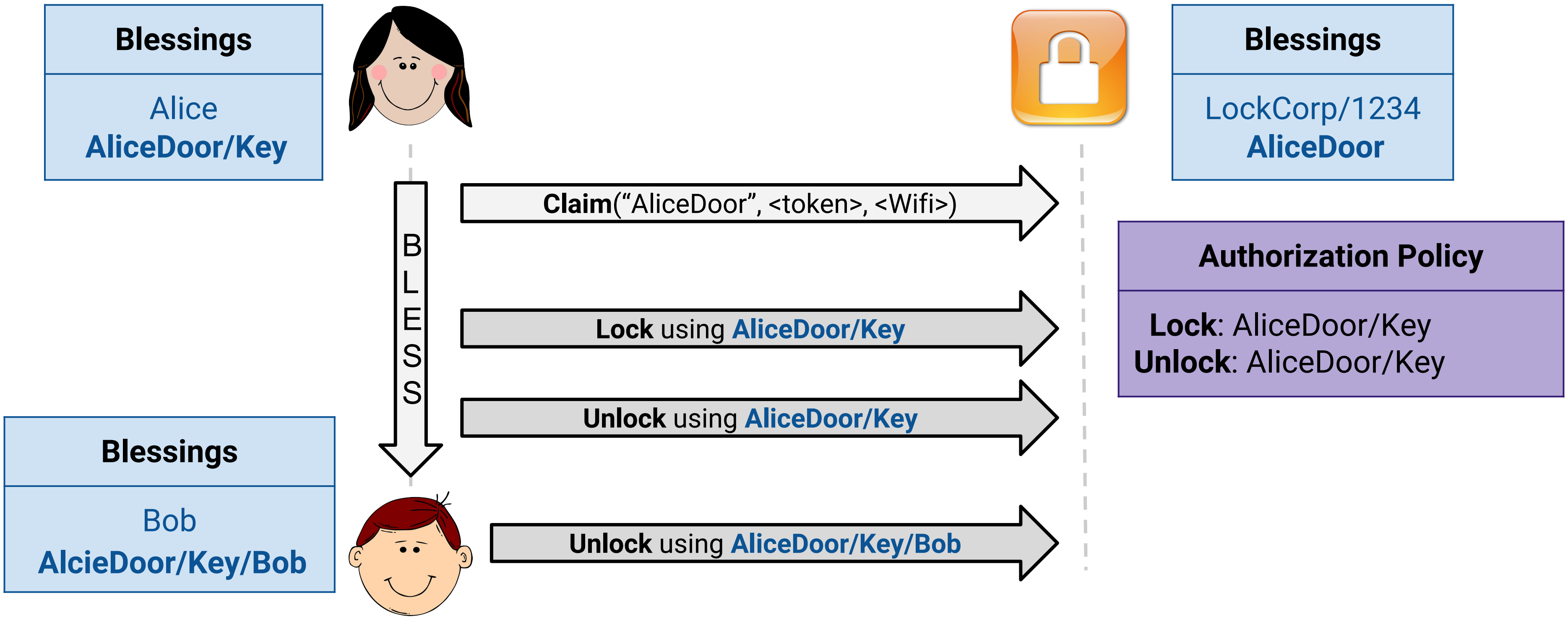
Lock setup


Blessings
Alice
AliceDoor/Key
Claim(“AliceDoor”, <token>, <Wifi>)
Lock using AliceDoor/Key
Unlock using AliceDoor/Key
Blessings
LockCorp/1234
AliceDoor
Authorization Policy
Lock: AliceDoor/Key
Unlock: AliceDoor/Key

Lock setup


Blessings
Alice
AliceDoor/Key
Claim("AliceDoor", <token>, <Wifi>)
Lock using AliceDoor/Key
Unlock using AliceDoor/Key
Blessings
LockCorp/1234
AliceDoor
Authorization Policy
Lock: AliceDoor/Key
Unlock: AliceDoor/Key
Blessings
Bob

Lock setup


Blessings
Alice
AliceDoor/Key
Blessings
LockCorp/1234
AliceDoor
Claim(“AliceDoor”, <token>, <Wifi>)
BLESS
Authorization Policy
Lock: AliceDoor/Key
Unlock: AliceDoor/Key
Lock using AliceDoor/Key
Unlock using AliceDoor/Key
Blessings
Bob
AlcieDoor/Key/Bob
Unlock using AliceDoor/Key/Bob

Properties

- **Works Offline**
  No internet access required to interact with the lock
- **Fully Decentralized**
  No cloud server controls access to all locks
- **Fine-grained Auditing**
  Each lock device can keep track of who accessed it (plus delegation trail)
- **No bearer tokens involved**

Practicalities and discussion

Blessings Management

- Devices and apps would accumulate multiple blessings over time
- How should users visualize and grant blessings?

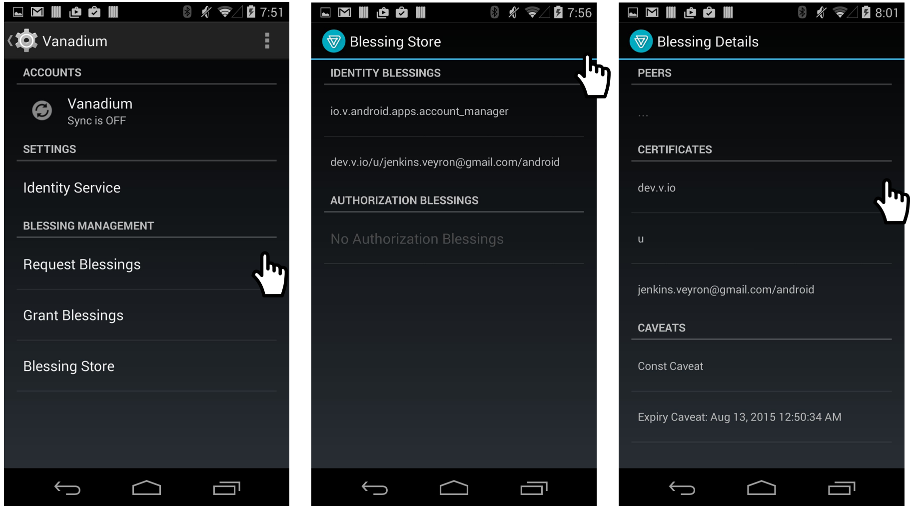
Blessings Management

- Devices and apps would accumulate multiple blessings over time
- How should users visualize and grant blessings?

Vanadium Blessings Manager App

- UI for visualizing blessings
- Grant blessings over NFC, Bluetooth

<ins>Future work</ins>: Blessing mailbox in the cloud



Private Key Management

- Securely storing private keys on device
- Many different hardware architectures and operating systems
- Multiple private keys per device (one for each app)

Private Key Management

- Securely storing private keys on device
- Many different hardware architectures and operating systems
- Multiple private keys per device (one for each app)

An Approach: Use a security agent (e.g., Plan9's factotum)

- Special process that holds private keys and performs crypto
- May store private keys in a TPM, if available
- May adjust itself based on the hardware

Vanadium authorization model: Summary

Principal and Blessings

Principal is a unique public/private key pair with human-readable names bound to it

All communication is encrypted & mutually authenticated

Forward-secrecy safe protocol, client and service identity privacy

Authorization is based on blessing names

Principals authenticated and authorized based on their blessing names

Fine-grained delegation and audit

Principals can bind an extension of their blessings to another principal under caveats

Vanadium authorization model: Summary

Principal and Blessings

Principal is a unique public/private key pair with human-readable names bound to it

All communication is encrypted & mutually authenticated

Forward-secrecy safe protocol, client and service identity privacy

Authorization is based on blessing names

Principals authenticated and authorized based on their blessing names

Fine-grained delegation and audit

Principals can bind an extension of their blessings to another principal under caveats

Distributed Authorization in Vanadium --- Taly and Shankar, FOSAD 16

Tomorrow

- Access control policies in Vanadium
- Privacy and service discovery mechanisms in Vanadium

Vanadium pointers

Homepage: https://vanadium.github.io/core.html

Concepts: https://vanadium.github.io/concepts/security.html

Tutorials: https://vanadium.github.io/tutorials/

Source: https://github.com/vanadium

Further reading

SDSI - A Simple Distributed Security Infrastructure --- Rivest and Lampson, 1996

Authentication in Distributed Systems: Theory and Practice --- Lampson et al., 1992

Delegation Logic: A Logic-based Approach to Distributed Authorization --- Li, 2003

Can we eliminate certificate revocation lists? --- Rivest, 2006

Macaroons: Cookies with Caveats for Decentralized Authorization --- Politz et al., 2014

Questions

email: ataly@google.com

Thank You!

Authorization requirements

Identity and Authorization
Decentralized deployment
Mutual authorization
Fine-grained delegation
Auditing and revocation
Ease of use

Other IOT security requirements

Identity and Authorization
Decentralized deployment
Mutual authorization
Fine-grained delegation
Auditing and revocation
Ease of use

Privacy
Private discovery
Anonymous communication
Transparency

Device Protection
No remote code execution
Automatic and secure updates
Verified boot