NAMED DATA NETWORKING

ACM ICN Paris 2021

Versec/DCT:

Creating and Using Trust Schemas

Kathleen Nichols, Pollere

Tutorial: Power of Trust Schemas for Easy and Secure Deployment of NDN Applications

Outline

- overview of use of versec language and trust schema enabled application transports to yield trust zones (tools, libraries, examples at https://github.com/pollere/DCT, location for Data-Centric Toolkit)
- developing an example trust schema (“office”)
- schemaCompile to check consistency, correctness of trust schema
- identity bundles: how to create and use, configuration needs
- distributors enable trust zones
- how to run example
- future work

What do we gain from this approach?

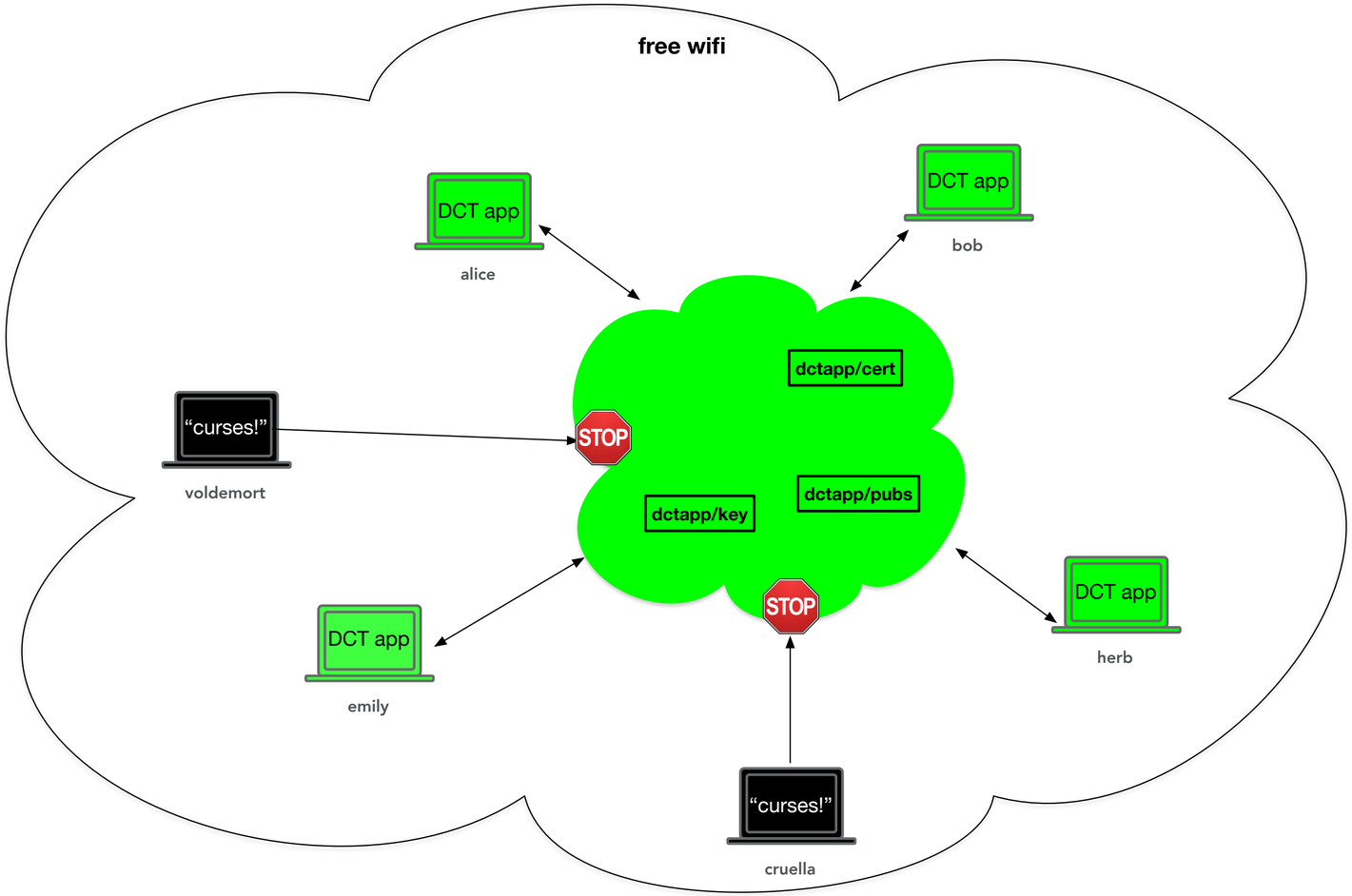
trust schema-based transport enables trust zones

security rules and certificate specifications are turned into a trust schema which is bundled with a trust anchor and a unique signing chain identity which an application's DCT-based transport uses to join a *trust zone* (in green)

1. alice takes her device (configured with an identity bundle) to a wifi cafe

2. the DCT-based app will *automatically* present alice's credentials and learn about all the members of the trust zone

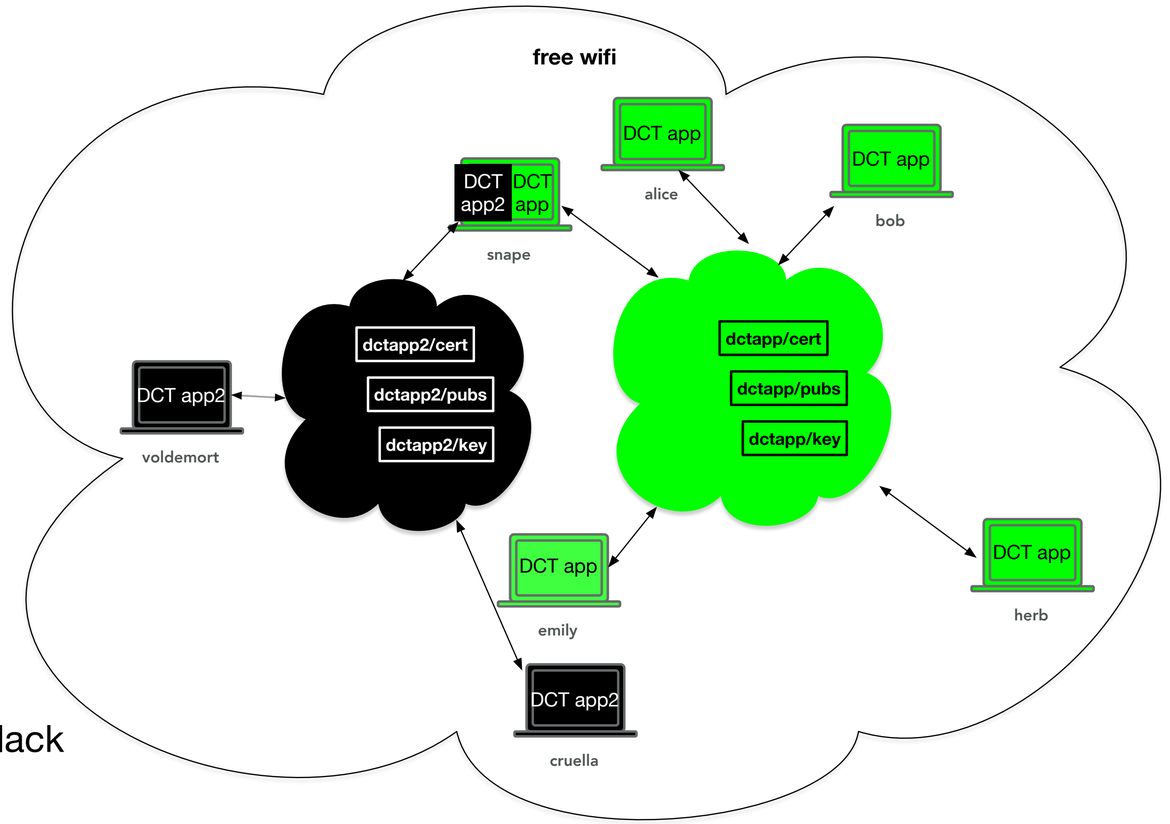
- outsiders can't get their messages past initial validation of packets
- If privacy is enabled, outsiders can't snoop pubs
- public cert chains are not encrypted
- symmetric keys are encrypted using public key of validated trust zone members



particular identity bundles are used to join particular trust zones

An **identity bundle** has three parts:

1. root of trust or **trust anchor** for this zone: a self-signed certificate
2. **trust schema** for this zone: in certificate form, signed by (a chain that ends at) the trust anchor
3. unique **signing identity**: a private key/ private cert pair and its signing chain that:
    a. terminates at the trust anchor
    b. conforms to the trust schema rules



Multiple trust zones can co-exist independently, e.g. green and black

A device can only join both if:

- has an identity bundle for each
- has a different application or a different instance of the same application for each identity bundle

A user with identities in both can share data that has reached the app level so be careful when configuring identities!

Federated sharing architectures between trust zones for future

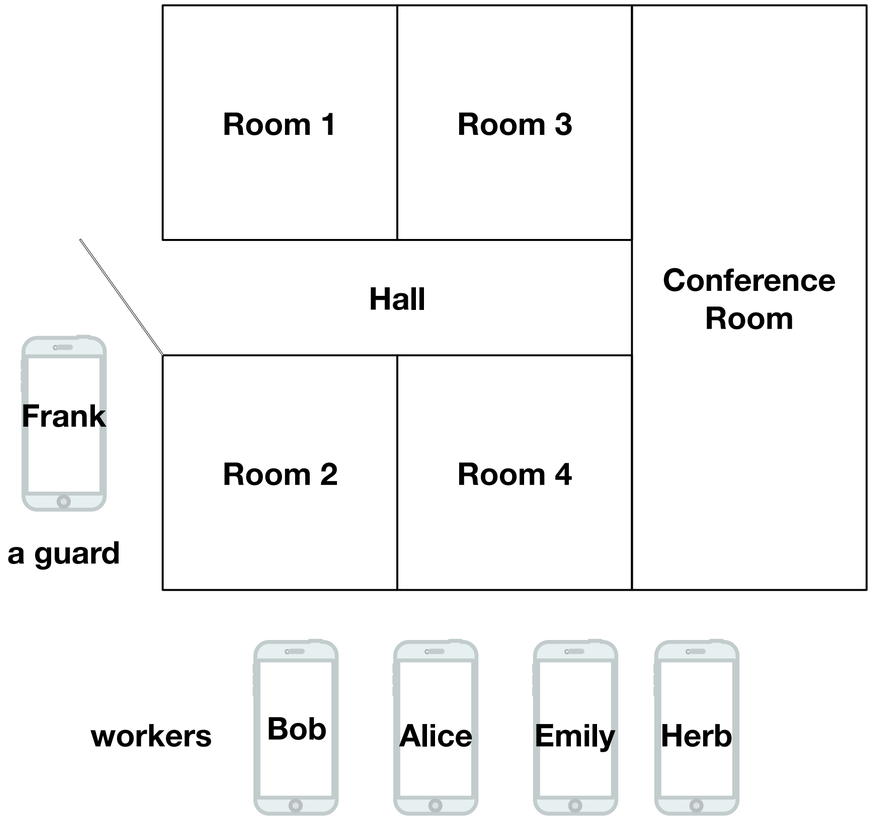
office example

A small office has:

- four people who use phone apps for access: Bob, Alice, Herb, and Emily. Alice is a manager.
- six rooms: four offices, a conference room, and a hall
- a room controller (raspPi-like) in each room that controls door lockset, light, temperature, screen and in hall controls light and door lockset

Rules

- an employee controls all the functions in their assigned office
- a manager can also control the conference room and hall
- a guard can control all door locks, lights, and temperature settings as a group (and not individually)
- a room controller publishes the status of its functions after it executes each command



elements of the rules for the office

Roles:

- employee, manager, guard, (room) controller

Room Controller functions:

- light, screen, temperature, door (lockset)

What needs to be communicated?

- command
  - issuer must be employee/manager/guard
  - commands to each function of: on/off, unlock/lock, out-of-office/heat/cool
- status
  - must be a controller
  - status of functions: on/off, locked/unlocked, out-of-office/heat/cool
- location
  - which room
  - whose phone
  - “all”

rules written in english

room controller can control and publish status of a room
- door: lock/unlock
- light: on/off
- temp: heat, cool, out-of-office settings
- screen: on/off

employee/manager can publish commands to their assigned room controller
- door: lock/unlock
- light: on/off
- temp: heat, cool, out-of-office settings
- screen: on/off

manager can publish commands to conference room and hall controllers
- door: lock/unlock
- light: on/off
- temp: heat, cool, out-of-office settings (except for hall)
- screen: on/off (except for hall)

guard can publish commands to all room controllers (in unison only)
- door: lock/unlock
- light: on/off
- temp: out-of-office setting

office.trust file of rules in versec

```
// trust anchor name for this trust zone
_net:    "office"

// Publication definition.
// Use of mbps shim means publications have mId, sCnt, mts (set by mbps)

#pub: _net/func/topic/loc/args/mId/sCnt/mts

ctrlr: #pub & (
    { func: "light" | "screen", args: "on" | "off" } |
    { func: "door", args: "lock" | "unlock" } |
    { func: "temp", args: "ooo"| "heat" | "cool" })

status: ctrlr & { topic: "status", loc: _roomId } <= cntrlrCert

rmCmd: ctrlr & { topic: "command", loc: _roomId } <= empCert | mgrCert

mgrCmd: ctrlr & { topic: "command", loc: "confRm"|"hall" } <= mgrCert

grdCmd: #pub & { topic: "command", loc: "all" } & ({ func: "light", args: "on"|"off" } |
        { func: "lock", args: "lock"|"unlock" } | { func: "temp", args: "ooo" }) <= grdCert

roleCert: _net/_role/_roleId/_keyinfo
empCert: roleCert & { _role: "employee" } <= roomCert
mgrCert: roleCert & { _role: "manager" } <= roomCert
cntrlrCert: roleCert & { _role: "controller" } <= roomCert
grdCert: roleCert & { _role: "guard" } <= roomCert

roomCert: _net/"room"/_roomId/_keyinfo <= configCert
configCert: _net/"config"/_configId/_keyinfo <= netCert
netCert:    _net/_keyinfo

// Publication prefix and validator type
#pubPrefix:    _net
#pubValidator: "EdDSA"

// Prefix used for syncData (NDN Interest/Data)
#wirePrefix: _ndnprefix/_net & { _ndnprefix: "localnet" }
#wireValidator: "AEAD"

// The final components are KEY,keyID, issuerID and version
_keyinfo: "KEY"/_/"dct"/_
```

rules written in english

room controller can control and publish status of a room
- door: lock/unlock
- light: on/off
- temp: heat, cool, out-of-office settings
- screen: on/off

employee/manager can publish commands to their assigned room controller
- door: lock/unlock
- light: on/off
- temp: heat, cool, out-of-office settings
- screen: on/off

manager can publish commands to conference room and hall controllers
- door: lock/unlock
- light: on/off
- temp: heat, cool, out-of-office settings (except for hall)
- screen: on/off (except for hall)

guard can publish commands to all room controllers (in unison only)
- door: lock/unlock
- light: on/off
- temp: out-of-office setting

office.trust file of rules in versec

```
// trust anchor name for this trust zone
_net:    "office"

// Publication definition.
// Use of mbps shim means publications have mId, sCnt, mts (set by mbps)

#pub: _net/func/topic/loc/args/mId/sCnt/mts

ctrlr: #pub & (
    { func: "light" | "screen", args: "on" | "off" } |
    { func: "door", args: "lock" | "unlock" } |
    { func: "temp", args: "ooo"| "heat" | "cool" })

status: ctrlr & { topic: "status", loc: _roomId } <= cntrlrCert

rmCmd: ctrlr & { topic: "command", loc: _roomId } <= empCert | mgrCert

mgrCmd: ctrlr & { topic: "command", loc: "confRm"|"hall" } <= mgrCert

grdCmd: #pub & { topic: "command", loc: "all" } & ({ func: "light", args: "on"|"off" } |
        { func: "lock", args: "lock"|"unlock" } | { func: "temp", args: "ooo" }) <= grdCert

roleCert: _net/_role/_roleId/_keyinfo
empCert: roleCert & { _role: "employee" } <= roomCert
mgrCert: roleCert & { _role: "manager" } <= roomCert
cntrlrCert: roleCert & { _role: "controller" } <= roomCert
grdCert: roleCert & { _role: "guard" } <= roomCert

roomCert: _net/"room"/_roomId/_keyinfo <= configCert
configCert: _net/"config"/_configId/_keyinfo <= netCert
netCert:    _net/_keyinfo

// Publication prefix and validator type
#pubPrefix:    _net
#pubValidator: "EdDSA"

// Prefix used for syncData (NDN Interest/Data)
#wirePrefix: _ndnprefix/_net & { _ndnprefix: "localnet" }
#wireValidator: "AEAD"

// The final components are KEY,keyID, issuerID and version
_keyinfo: "KEY"/_/"dct"/_
```

rules written in english

room controller can control and publish status of a room
- door: lock/unlock
- light: on/off
- temp: heat, cool, out-of-office settings
- screen: on/off

employee/manager can publish commands to their assigned room controller
- door: lock/unlock
- light: on/off
- temp: heat, cool, out-of-office settings
- screen: on/off

manager can publish commands to conference room and hall controllers
- door: lock/unlock
- light: on/off
- temp: heat, cool, out-of-office settings (except for hall)
- screen: on/off (except for hall)

guard can publish commands to all room controllers (in unison only)
- door: lock/unlock
- light: on/off
- temp: out-of-office setting

office.trust file of rules in versec

```
// trust anchor name for this trust zone
_net:    "office"

// Publication definition.
// Use of mbps shim means publications have mId, sCnt, mts (set by mbps)

#pub: _net/func/topic/loc/args/mId/sCnt/mts

ctrlr: #pub & (
    { func: "light" | "screen", args: "on" | "off" } |
    { func: "door", args: "lock" | "unlock" } |
    { func: "temp", args: "ooo"| "heat" | "cool" })

status: ctrlr & { topic: "status", loc: _roomId } <= cntrlrCert

rmCmd: ctrlr & { topic: "command", loc: _roomId } <= empCert | mgrCert

mgrCmd: ctrlr & { topic: "command", loc: "confRm"|"hall" } <= mgrCert

grdCmd: #pub & { topic: "command", loc: "all" } & ({ func: "light", args: "on"|"off" } |
        { func: "lock", args: "lock"|"unlock" } | { func: "temp", args: "ooo" }) <= grdCert

roleCert: _net/_role/_roleId/_keyinfo
empCert: roleCert & { _role: "employee" } <= roomCert
mgrCert: roleCert & { _role: "manager" } <= roomCert
cntrlrCert: roleCert & { _role: "controller" } <= roomCert
grdCert: roleCert & { _role: "guard" } <= roomCert

roomCert: _net/"room"/_roomId/_keyinfo <= configCert
configCert: _net/"config"/_configId/_keyinfo <= netCert
netCert:    _net/_keyinfo

// Publication prefix and validator type
#pubPrefix:    _net
#pubValidator: "EdDSA"

// Prefix used for syncData (NDN Interest/Data)
#wirePrefix: _ndnprefix/_net & { _ndnprefix: "localnet" }
#wireValidator: "AEAD"

// The final components are KEY,keyID, issuerID and version
_keyinfo: "KEY"/_/"dct"/_
```

rules written in english

room controller can control and publish status of a room
- door: lock/unlock
- light: on/off
- temp: heat, cool, out-of-office settings
- screen: on/off

employee/manager can publish commands to their assigned room controller
- door: lock/unlock
- light: on/off
- temp: heat, cool, out-of-office settings
- screen: on/off

manager can publish commands to conference room and hall controllers
- door: lock/unlock
- light: on/off
- temp: heat, cool, out-of-office settings (except for hall)
- screen: on/off (except for hall)

guard can publish commands to all room controllers (in unison only)
- door: lock/unlock
- light: on/off
- temp: out-of-office setting

office.trust file of rules in versec

```
// trust anchor name for this trust zone
_net:    "office"

// Publication definition.
// Use of mbps shim means publications have mId, sCnt, mts (set by mbps)

#pub: _net/func/topic/loc/args/mId/sCnt/mts

ctrlr: #pub & (
    { func: "light" | "screen", args: "on" | "off" } |
    { func: "door", args: "lock" | "unlock" } |
    { func: "temp", args: "ooo"| "heat" | "cool" })

status: ctrlr & { topic: "status", loc: _roomId } <= cntrlrCert

rmCmd: ctrlr & { topic: "command", loc: _roomId } <= empCert | mgrCert

mgrCmd: ctrlr & { topic: "command", loc: "confRm"|"hall" } <= mgrCert

grdCmd: #pub & { topic: "command", loc: "all" } & ({ func: "light", args: "on"|"off" } |
        { func: "lock", args: "lock"|"unlock" } | { func: "temp", args: "ooo" }) <= grdCert

roleCert: _net/_role/_roleId/_keyinfo
empCert: roleCert & { _role: "employee" } <= roomCert
mgrCert: roleCert & { _role: "manager" } <= roomCert
cntrlrCert: roleCert & { _role: "controller" } <= roomCert
grdCert: roleCert & { _role: "guard" } <= roomCert

roomCert: _net/"room"/_roomId/_keyinfo <= configCert
configCert: _net/"config"/_configId/_keyinfo <= netCert
netCert:    _net/_keyinfo

// Publication prefix and validator type
#pubPrefix:    _net
#pubValidator: "EdDSA"

// Prefix used for syncData (NDN Interest/Data)
#wirePrefix: _ndnprefix/_net & { _ndnprefix: "localnet" }
#wireValidator: "AEAD"

// The final components are KEY,keyID, issuerID and version
_keyinfo: "KEY"/_/"dct"/_
```

rules written in english

room controller can control and publish status of a room
- door: lock/unlock
- light: on/off
- temp: heat, cool, out-of-office settings
- screen: on/off

employee/manager can publish commands to their assigned room controller
- door: lock/unlock
- light: on/off
- temp: heat, cool, out-of-office settings
- screen: on/off

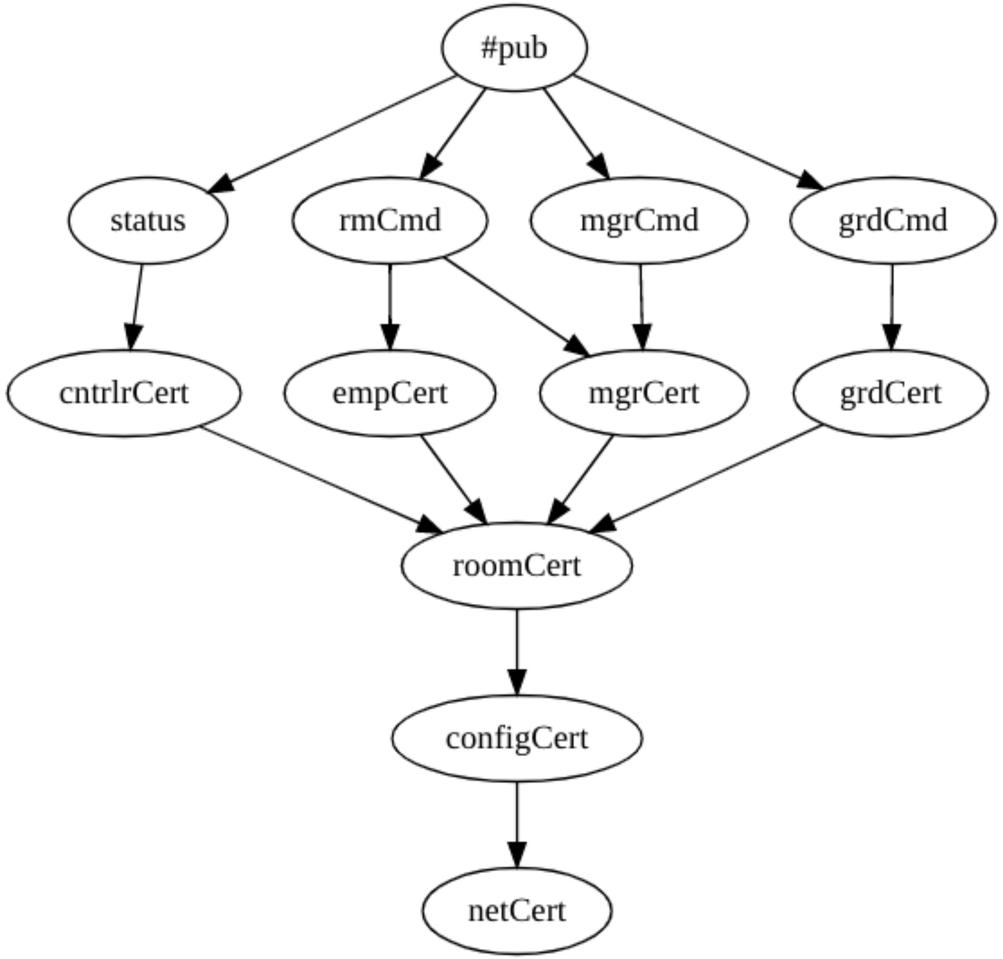
manager can publish commands to conference room and hall controllers
- door: lock/unlock
- light: on/off
- temp: heat, cool, out-of-office settings (except for hall)
- screen: on/off (except for hall)

guard can publish commands to all room controllers (in unison only)
- door: lock/unlock
- light: on/off
- temp: out-of-office setting

office.trust file of rules in versec

```
// trust anchor name for this trust zone
_net:    "office"

// Publication definition.
// Use of mbps shim means publications have mId, sCnt, mts (set by mbps)

#pub: _net/func/topic/loc/args/mId/sCnt/mts

ctrlr: #pub & (
    { func: "light" | "screen", args: "on" | "off" } |
    { func: "door", args: "lock" | "unlock" } |
    { func: "temp", args: "ooo"| "heat" | "cool" })

status: ctrlr & { topic: "status", loc: _roomId } <= cntrlrCert

rmCmd: ctrlr & { topic: "command", loc: _roomId } <= empCert | mgrCert

mgrCmd: ctrlr & { topic: "command", loc: "confRm"|"hall" } <= mgrCert

grdCmd: #pub & { topic: "command", loc: "all" } & ({ func: "light", args: "on"|"off" } |
        { func: "lock", args: "lock"|"unlock" } | { func: "temp", args: "ooo" }) <= grdCert

roleCert: _net/_role/_roleId/_keyinfo
empCert: roleCert & { _role: "employee" } <= roomCert
mgrCert: roleCert & { _role: "manager" } <= roomCert
cntrlrCert: roleCert & { _role: "controller" } <= roomCert
grdCert: roleCert & { _role: "guard" } <= roomCert

roomCert: _net/"room"/_roomId/_keyinfo <= configCert
configCert: _net/"config"/_configId/_keyinfo <= netCert
netCert:    _net/_keyinfo

// Publication prefix and validator type
#pubPrefix:    _net
#pubValidator: "EdDSA"

// Prefix used for syncData (NDN Interest/Data)
#wirePrefix: _ndnprefix/_net & { _ndnprefix: "localnet" }
#wireValidator: "AEAD"

// The final components are KEY,keyID, issuerID and version
_keyinfo: "KEY"/_/"dct"/_
```

pub and cert relationships

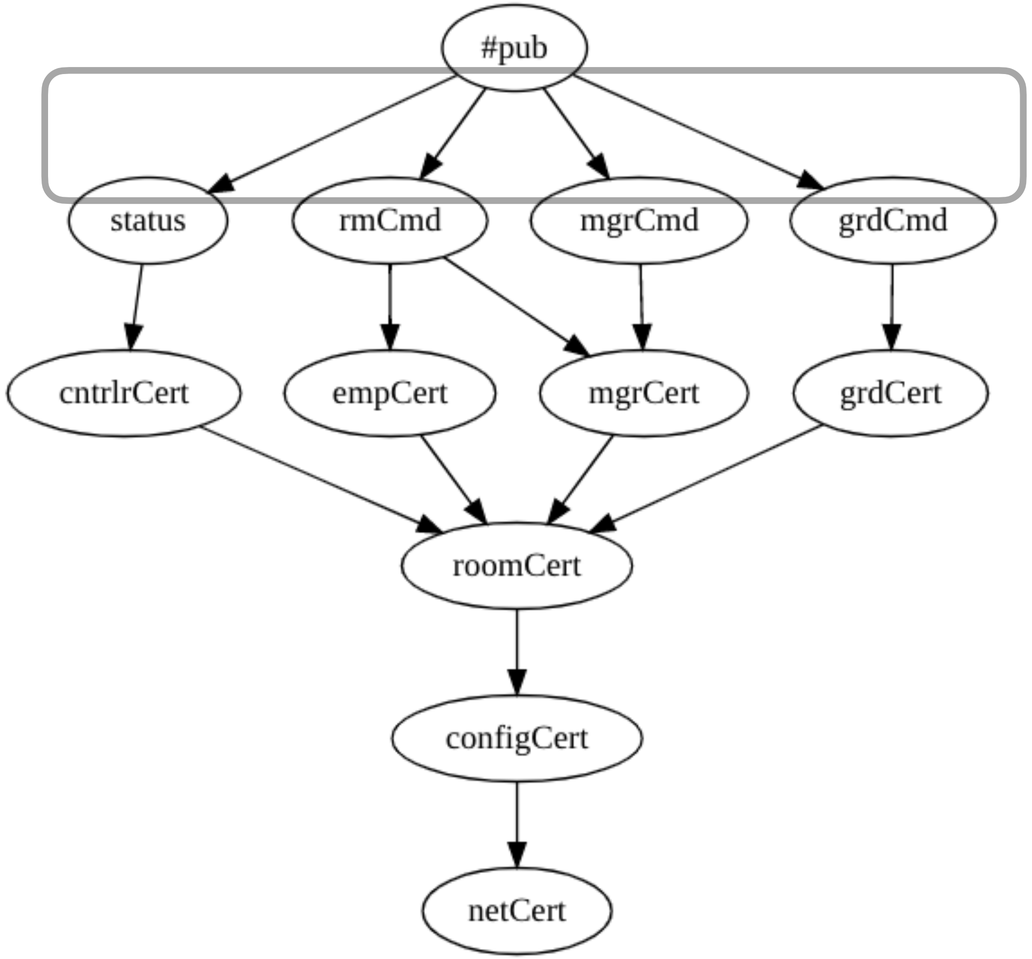


Note: the graphviz commands to construct this picture are the first thing output when schemaCompile is given a '-d' flag. Paste them into a local graphviz tool or a browser-based tool like Sketchviz to get this diagnostic for any schema.

office.trust file of rules in versec

```
// trust anchor name for this trust zone
_net:    "office"

// Publication definition.
// Use of mbps shim means publications have mId, sCnt, mts (set by mbps)

#pub: _net/func/topic/loc/args/mId/sCnt/mts

ctrlr: #pub & (
    { func: "light" | "screen", args: "on" | "off" } |
    { func: "door", args: "lock" | "unlock" } |
    { func: "temp", args: "ooo"| "heat" | "cool" })

status: ctrlr & { topic: "status", loc: _roomId } <= cntrlrCert

rmCmd: ctrlr & { topic: "command", loc: _roomId } <= empCert | mgrCert

mgrCmd: ctrlr & { topic: "command", loc: "confRm"|"hall" } <= mgrCert

grdCmd: #pub & { topic: "command", loc: "all" } & ({ func: "light", args: "on"|"off" } |
        { func: "lock", args: "lock"|"unlock" } | { func: "temp", args: "ooo" }) <= grdCert

roleCert: _net/_role/_roleId/_keyinfo
empCert: roleCert & { _role: "employee" } <= roomCert
mgrCert: roleCert & { _role: "manager" } <= roomCert
cntrlrCert: roleCert & { _role: "controller" } <= roomCert
grdCert: roleCert & { _role: "guard" } <= roomCert

roomCert: _net/"room"/_roomId/_keyinfo <= configCert
configCert: _net/"config"/_configId/_keyinfo <= netCert
netCert:    _net/_keyinfo

// Publication prefix and validator type
#pubPrefix:    _net
#pubValidator: "EdDSA"

// Prefix used for syncData (NDN Interest/Data)
#wirePrefix: _ndnprefix/_net & { _ndnprefix: "localnet" }
#wireValidator: "AEAD"

// The final components are KEY,keyID, issuerID and version
_keyinfo: "KEY"/_/"dct"/_
```

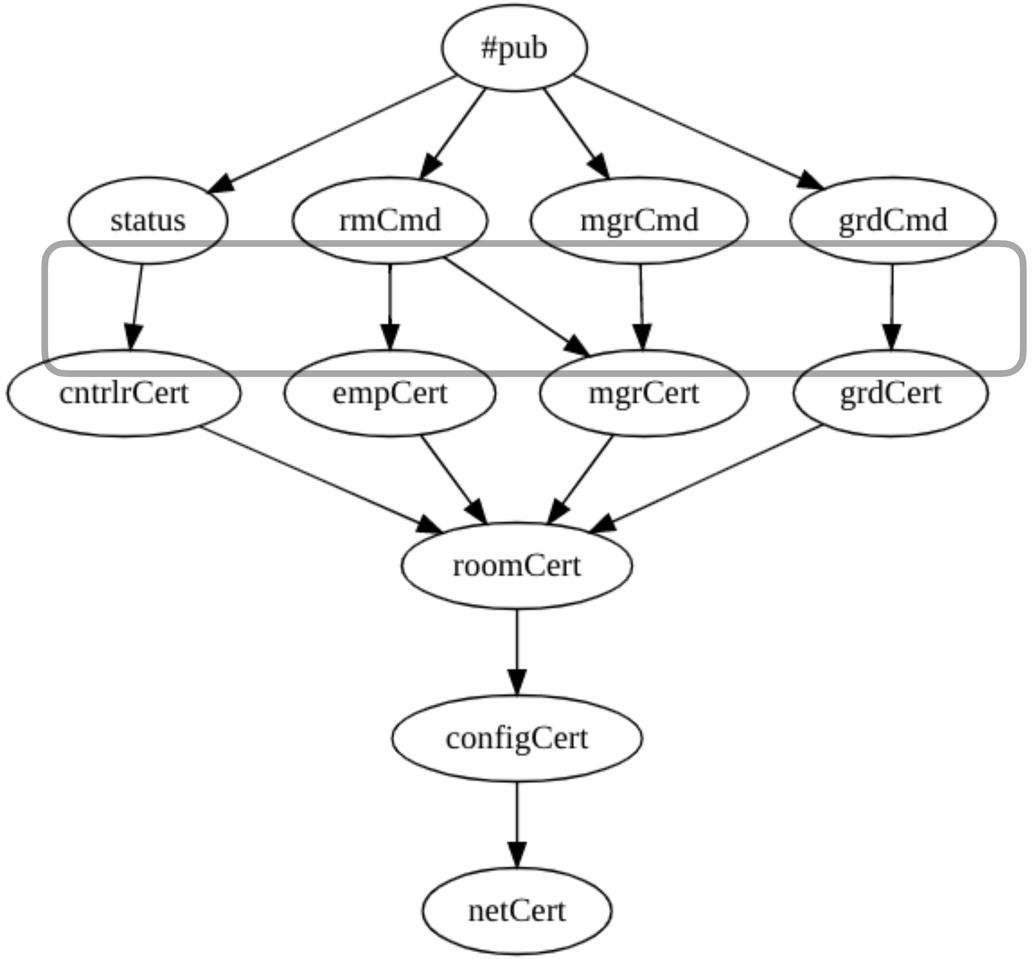
pub inheritance relationships



office.trust file of rules in versec

```
// trust anchor name for this trust zone
_net:    "office"

// Publication definition.
// Use of mbps shim means publications have mId, sCnt, mts (set by mbps)

#pub: _net/func/topic/loc/args/mId/sCnt/mts

ctrlr: #pub & (
    { func: "light" | "screen", args: "on" | "off" } |
    { func: "door", args: "lock" | "unlock" } |
    { func: "temp", args: "ooo"| "heat" | "cool" })

status: ctrlr & { topic: "status", loc: _roomId } <= cntrlrCert

rmCmd: ctrlr & { topic: "command", loc: _roomId } <= empCert | mgrCert

mgrCmd: ctrlr & { topic: "command", loc: "confRm"|"hall" } <= mgrCert

grdCmd: #pub & { topic: "command", loc: "all" } & ({ func: "light", args: "on"|"off" } |
        { func: "lock", args: "lock"|"unlock" } | { func: "temp", args: "ooo" }) <= grdCert

roleCert: _net/_role/_roleId/_keyinfo
empCert: roleCert & { _role: "employee" } <= roomCert
mgrCert: roleCert & { _role: "manager" } <= roomCert
cntrlrCert: roleCert & { _role: "controller" } <= roomCert
grdCert: roleCert & { _role: "guard" } <= roomCert

roomCert: _net/"room"/_roomId/_keyinfo <= configCert
configCert: _net/"config"/_configId/_keyinfo <= netCert
netCert:    _net/_keyinfo

// Publication prefix and validator type
#pubPrefix:    _net
#pubValidator: "EdDSA"

// Prefix used for syncData (NDN Interest/Data)
#wirePrefix: _ndnprefix/_net & { _ndnprefix: "localnet" }
#wireValidator: "AEAD"

// The final components are KEY,keyID, issuerID and version
_keyinfo: "KEY"/_/"dct"/_
```

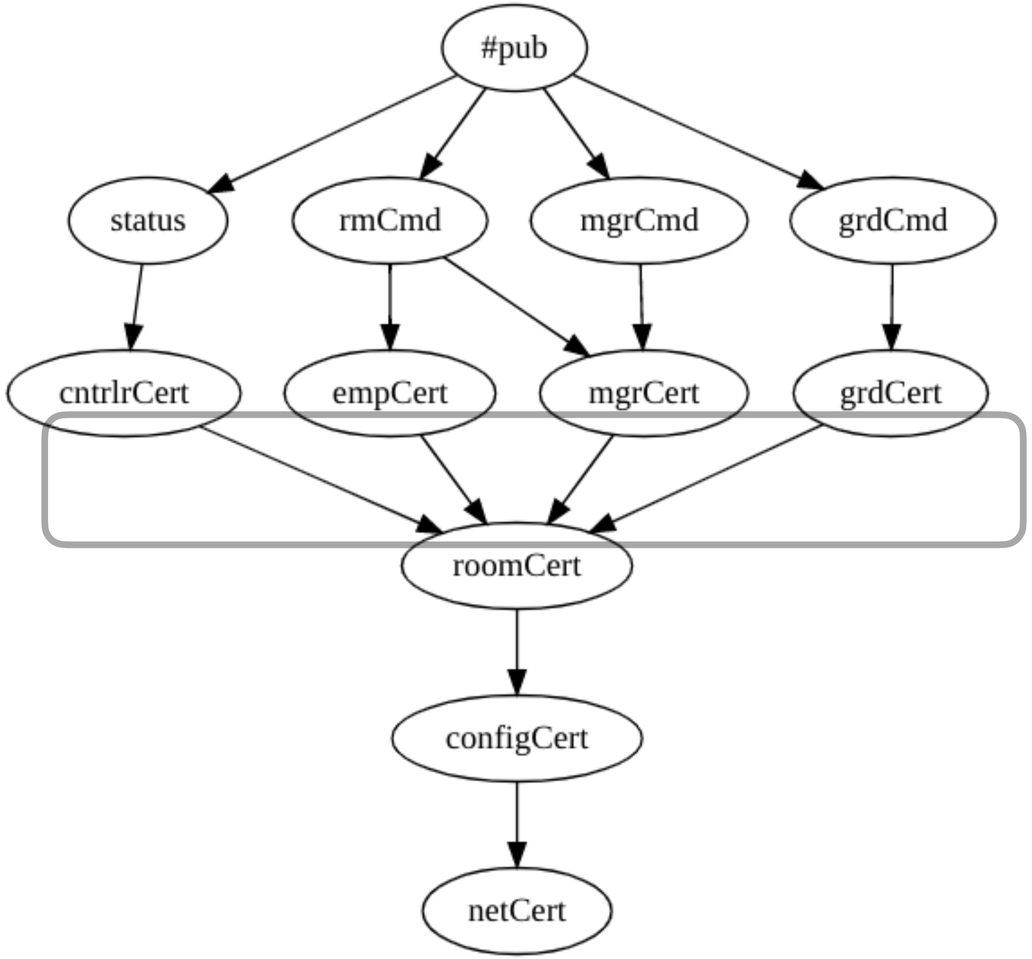
signing relationships



office.trust file of rules in versec

```
// trust anchor name for this trust zone
_net:    "office"

// Publication definition.
// Use of mbps shim means publications have mId, sCnt, mts (set by mbps)

#pub: _net/func/topic/loc/args/mId/sCnt/mts

ctrlr: #pub & (
    { func: "light" | "screen", args: "on" | "off" } |
    { func: "door", args: "lock" | "unlock" } |
    { func: "temp", args: "ooo"| "heat" | "cool" })

status: ctrlr & { topic: "status", loc: _roomId } <= cntrlrCert

rmCmd: ctrlr & { topic: "command", loc: _roomId } <= empCert | mgrCert

mgrCmd: ctrlr & { topic: "command", loc: "confRm"|"hall" } <= mgrCert

grdCmd: #pub & { topic: "command", loc: "all" } & ({ func: "light", args: "on"|"off" } |
        { func: "lock", args: "lock"|"unlock" } | { func: "temp", args: "ooo" }) <= grdCert

roleCert: _net/_role/_roleId/_keyinfo
empCert: roleCert & { _role: "employee" } <= roomCert
mgrCert: roleCert & { _role: "manager" } <= roomCert
cntrlrCert: roleCert & { _role: "controller" } <= roomCert
grdCert: roleCert & { _role: "guard" } <= roomCert

roomCert: _net/"room"/_roomId/_keyinfo <= configCert
configCert: _net/"config"/_configId/_keyinfo <= netCert
netCert:    _net/_keyinfo

// Publication prefix and validator type
#pubPrefix:    _net
#pubValidator: "EdDSA"

// Prefix used for syncData (NDN Interest/Data)
#wirePrefix: _ndnprefix/_net & { _ndnprefix: "localnet" }
#wireValidator: "AEAD"

// The final components are KEY,keyID, issuerID and version
_keyinfo: "KEY"/_/"dct"/_
```

signing relationships



office.trust file of rules in versec

```
// trust anchor name for this trust zone
_net:    "office"

// Publication definition.
// Use of mbps shim means publications have mId, sCnt, mts (set by mbps)

#pub: _net/func/topic/loc/args/mId/sCnt/mts

ctrlr: #pub & (
    { func: "light" | "screen", args: "on" | "off" } |
    { func: "door", args: "lock" | "unlock" } |
    { func: "temp", args: "ooo"| "heat" | "cool" })

status: ctrlr & { topic: "status", loc: _roomId } <= cntrlrCert

rmCmd: ctrlr & { topic: "command", loc: _roomId } <= empCert | mgrCert

mgrCmd: ctrlr & { topic: "command", loc: "confRm"|"hall" } <= mgrCert

grdCmd: #pub & { topic: "command", loc: "all" } & ({ func: "light", args: "on"|"off" } |
        { func: "lock", args: "lock"|"unlock" } | { func: "temp", args: "ooo" }) <= grdCert

roleCert: _net/_role/_roleId/_keyinfo
empCert: roleCert & { _role: "employee" } <= roomCert
mgrCert: roleCert & { _role: "manager" } <= roomCert
cntrlrCert: roleCert & { _role: "controller" } <= roomCert
grdCert: roleCert & { _role: "guard" } <= roomCert

roomCert: _net/"room"/_roomId/_keyinfo <= configCert
configCert: _net/"config"/_configId/_keyinfo <= netCert
netCert:    _net/_keyinfo

// Publication prefix and validator type
#pubPrefix:    _net
#pubValidator: "EdDSA"

// Prefix used for syncData (NDN Interest/Data)
#wirePrefix: _ndnprefix/_net & { _ndnprefix: "localnet" }
#wireValidator: "AEAD"

// The final components are KEY,keyID, issuerID and version
_keyinfo: "KEY"/_/"dct"/_
```

office.trust file of rules in versec

roles translate to cert chains

Exact details of the certificates and how they are combined is critical for well-founded security; this is discussed in the conference paper “Trust Schemas and ICN: Key to Secure Home IoT”

certificates in signing chain need to set roles of guard, employee, manager, room controller and the unique identifiers within those roles which will be conferred at configuration (e.g., alice, room2, 42)

certificates in the role signing chains used to assign rooms by identifier (e.g., room2, hall, all)

configurer certificate in the chain protects usage of trust root

single root of trust must be specified and all certs terminate at its (self-signed) cert

publications use EdDSA signing

Use encryption of “on the wire” (syncData) packets preserves privacy

```
// trust anchor name for this trust zone
_net:    "office"

// Publication definition.
// Use of mbps shim means publications have mId, sCnt, mts (set by mbps)

#pub: _net/func/topic/loc/args/mId/sCnt/mts

ctrlr: #pub & (
    { func: "light" | "screen", args: "on" | "off" } |
    { func: "door", args: "lock" | "unlock" } |
    { func: "temp", args: "ooo"| "heat" | "cool" })

status: ctrlr & { topic: "status", loc: _roomId } <= cntrlrCert

rmCmd: ctrlr & { topic: "command", loc: _roomId } <= empCert | mgrCert

mgrCmd: ctrlr & { topic: "command", loc: "confRm"|"hall" } <= mgrCert

grdCmd: #pub & { topic: "command", loc: "all" } & ({ func: "light", args: "on"|"off" } |
        { func: "lock", args: "lock"|"unlock" } | { func: "temp", args: "ooo" }) <= grdCert

roleCert: _net/_role/_roleId/_keyinfo
empCert: roleCert & { _role: "employee" } <= roomCert
mgrCert: roleCert & { _role: "manager" } <= roomCert
cntrlrCert: roleCert & { _role: "controller" } <= roomCert
grdCert: roleCert & { _role: "guard" } <= roomCert

roomCert: _net/"room"/_roomId/_keyinfo <= configCert
configCert: _net/"config"/_configId/_keyinfo <= netCert
netCert:    _net/_keyinfo

// Publication prefix and validator type
#pubPrefix:    _net
#pubValidator: "EdDSA"

// Prefix used for syncData (NDN Interest/Data)
#wirePrefix: _ndnprefix/_net & { _ndnprefix: "localnet" }
#wireValidator: "AEAD"

// The final components are KEY,keyID, issuerID and version
_keyinfo: "KEY"/_/"dct"/_
```

about schemaCompile

- Compiler uses the rules to ensure constraints are consistent (i.e., no contradictions)
- Compiler makes sure the publications produced by the constraints are “grounded” i.e., every field can be filled in (literal string, component in signing chain, parameter that must be filled in by application)
- Compiler constructs templates for the run-time builder/validator
- The signing rules must form a DAG with a single root of trust

DCT **tools** includes many utilities to look at the elements of the schema.
There is a lot of detail to look at for those who are interested

builder dump – show all pubs some identity can produce

templates for pubs a **confRm** can publish

```
% bld_dump id/confRm.bundle
chain: 0, signing cert: /office/controller/confRm/KEY/?g?&/dct/??e5w?|
parameters: fe { func topic loc args mId sCnt mts }
 args        (4): 20208000000 /office/temp/status/confRm/84/85/86/87
 args        (4):           1 /office/light/status/confRm/off/85/86/87
 args        (4):           1 /office/screen/status/confRm/on/85/86/87
 args        (4):           1 /office/light/status/confRm/on/85/86/87
 args        (4):           1 /office/door/status/confRm/lock/85/86/87
 args        (4):           1 /office/door/status/confRm/unlock/85/86/87
 args        (4):           1 /office/screen/status/confRm/off/85/86/87
```

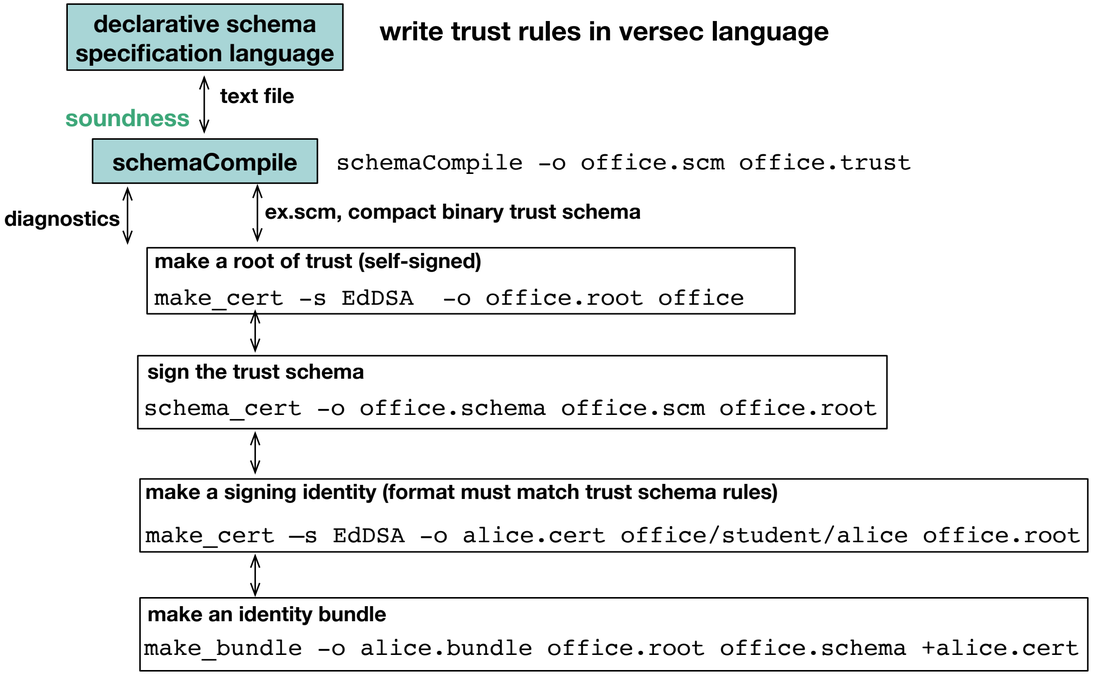
templates for pubs **bob** can publish

```
% bld_dump id/bob.bundle
chain: 1, signing cert: /office/employee/bob/KEY/68?/dct/??e5~M?
parameters: fe { func topic loc args mId sCnt mts }
 args        (4): 20208000000 /office/temp/command/room1/84/85/86/87
 args        (4):           1 /office/light/command/room1/off/85/86/87
 args        (4):           1 /office/screen/command/room1/on/85/86/87
 args        (4):           1 /office/light/command/room1/on/85/86/87
 args        (4):           1 /office/door/command/room1/lock/85/86/87
 args        (4):           1 /office/door/command/room1/unlock/85/86/87
 args        (4):           1 /office/screen/command/room1/off/85/86/87
```

from trust rules to identity bundle

- Use versec declarative language to express trust rules in readable fashion and a compiler to check the rules
- Application's transport uses a verifiable compact version of the trust schema to construct, sign, and validate communications
- A trust schema and its trust root (in cert form) are distributed with a signing identity, in an *identity bundle*, used by the application to join a *trust zone*
- Signing identities consist of a private signing key with its associated public cert and the entire signing chain of public certs as specified by the trust schema, terminating in the trust root. Only the signing key needs to be kept private.
- This is sufficient to join the trust zone. DCT transport handles finding, validating, and importing other certs

From rules to identity bundle:

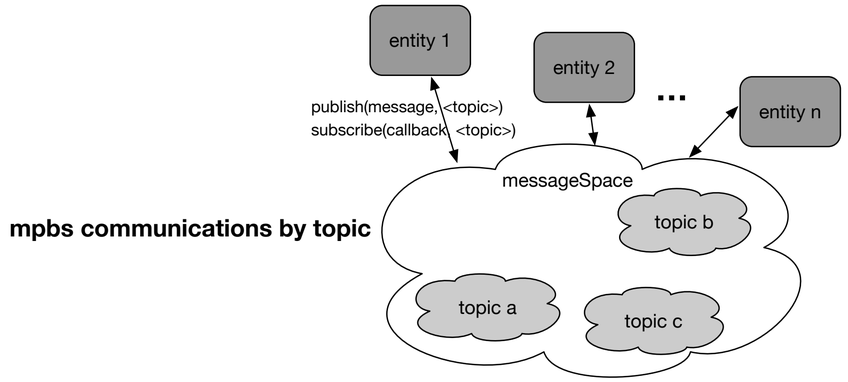


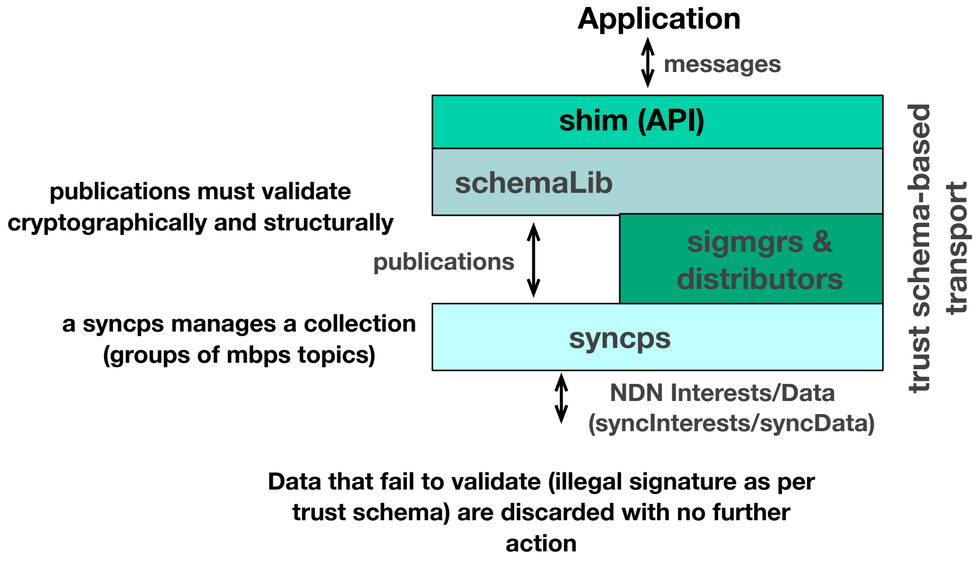
Using the bundle from the command-line:

```
%app alice.bundle
```

digression into what you “need to know” about DCT transport

- DCT’s trust schema enabled transport deals with *publications* (NDN Data), building and validating them according to the trust schema to give structural (fine-grained) security with a *sigmgr* type specified to give cryptographic security for that publication (i.e., signing)
- Sync protocol *syncps* manages *collections* of publications (e.g., pub, cert, key) and passes new, valid pubs to its shim if they match subscribed *topics*
  - shim subscriptions can be made specific by including more levels of the publication Name hierarchy
  - syncps always has an outstanding Interest for its collection which makes communications efficient
- Publications are packaged and unpackaged by syncps into/from NDN Data “wire” packets which have a *signature manager* type specified to give cryptographic security
- using the general-purpose *mbps* (message-based pub/sub) library shim
- applications and mbps publish and subscribe by message *topics* which are contained within the *pub* collection
- focus on the topics this application needs to communicate; mbps takes care of the rest

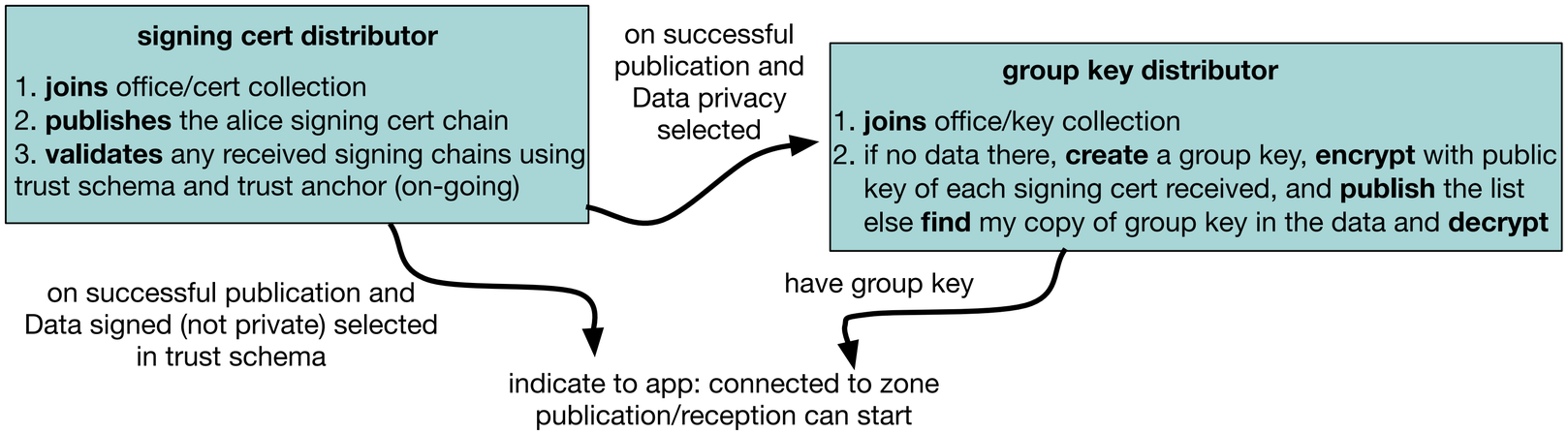




distributors enable trust zone connection and data privacy

- connection and privacy happen *automatically*, based on the signature managers selected for publications and syncData
- distributors manage access to automatically created collections for certs and keys (while mbps manages application pubs through its syncps)
- applications supply a callback so they can be started once this process is complete

1. app creates a DCT transport which parses the trust schema: `client(alice.bundle)`
2. to join trust zone, app calls `client.connect()` which automatically starts:



Note that creating an identity-protected trust zone and data privacy does not require fine-grained trust rules (e.g., see DCT/examples/mbps/mbps0.trust)

notes on running the example

- github.com/pollere/DCT/examples/office will have office.trust, the room and phone apps, mbps.hpp (soon to move to DCT/include/dct) and a handy script to make some identity bundles (mkIDs.sh)
- unfortunately, you currently have to duplicate the NDN library and forwarder versions that pollere uses but working on this
- you can try making changes to office.trust and running schemaCompile without running the code, for example:
  - add a status message from the employees for in-office and out-of-office
  - add a guest role or presenter role that can control the screen in the conference room
- for the adventurous, once the bundles are made, the applications compiled, and an appropriate NFD is running, can run the applications from command line with “%room <room>.bundle” or “%phone <user>.bundle”. It always takes at least two members of a trust zone before anything can happen. After that, new entities can join anytime.

code for the room controller

```
#include "mbps.hpp"

static std::string role{};          // this instance's role
static std::string id{};

static void statusPubr(mbps &cm, const std::string& f, const std::string& a) {
    msgTags tgs{};
    tgs.emplace_back("func", f);
    tgs.emplace_back("args", a);
    tgs.emplace_back("topic", "status");
    cm.publish(flds);
}

msgHndlr cmdRecv(mbps &cm, const mbpsMsg& msg, std::vector<uint8_t>&)
{
   std::string f = msg.tags["func"];
    std::string a = msg.["args"];
    print("{:%M:%S} {} in {} setting {} to {}\n",
            ticks(now.time_since_epoch()), role, id, f, a);
    statusPubr(cm, f, a);
}

int main(int argc, char* argv[])
{
    mbps cm(argv[argc-1]);     //Create the mbps client
    role = cm.attribute("role");
    id = cm.attribute("roleId");
    // Connect and pass in the handler
    cm.connect(    /* main task for this entity */
        [&cm]() {
            std::vector<std::string> acc;
            if(id == "hall") {
                acc = {"light", "door"};
            } else {
                acc = {"light", "door", "screen", "temp"};
            }
            for(auto i=0; i<acc.size(); i++) {
                cm.subscribe(acc[i] + "/command/" + id, cmdRecv);
                cm.subscribe(acc[i] + "/command/all", cmdRecv);
            }
     );
    cm.run();
}
```

commissioning and updating

Using bundle on the command line is great for development, but not deployment

Network deployment brings other issues for good key hygiene that are noted in paper “Trust Schemas and ICN: Key to Secure Home IoT”

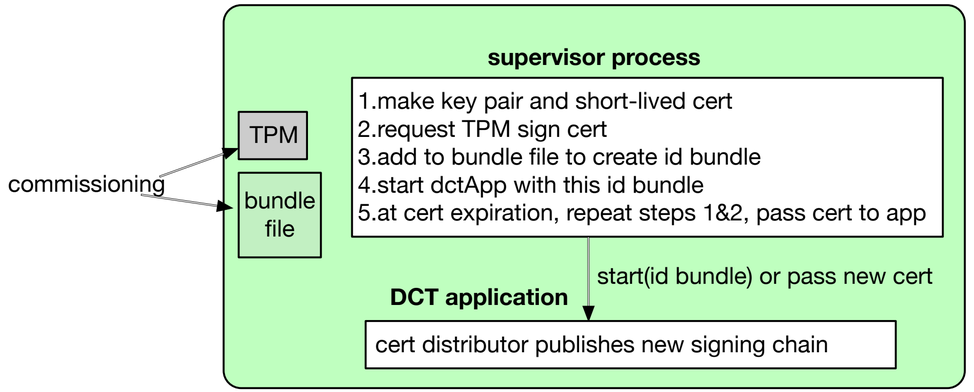
Expect the actual app to be managed by a supervisor or similar process control program which creates short-lived (~hours) signing key pairs to be signed by the installed signing key via TPM

Follow best current practices to commission (on-board) a device by provisioning the bundle file out-of-band (i.e., not over open network)

- remove the secret key from the bundle and secure with TPM
- put the rest of the bundle file in a known location

Other over-the-air approaches possible (Tianyuan will discuss)

Data-centric approaches to updates could be developed



future

On our planned list

- additional features for versec language/compiler and schemaLib to make it easier to express publication variants, to make use of multiple signing certs
- making mbps more general
- a DCT NDN Face for more efficient use of broadcast networks for syncps
- distribute trust zone via relay, etc

Other thoughts

- improved group key distributor
- other types of distributors (on-line updates?)
- updating signing keys and trust schema over network
- shims for other communication models
- other signature managers?