A TOOLKIT FOR CONSTRUCTION OF AUTHORIZATION SERVICE INFRASTRUCTURE FOR THE INTERNET OF THINGS

AUTHORS: DAVID BROMAN, EUNSUK KANG, HOKEUN KIM, EDWARD LEE

PRESENTED BY: RYAN FISK

INTRODUCTION AND BACKGROUND

- Current network security measures fail to address many IoT challenges
- IoT security needs to take usage and purpose of the device into account
  - Balance security vs performance vs resources

PROBLEMS WITH MODERN SOLUTIONS

Heterogeneity

- Different requirements and resources for devices

Open Environments

- Attackers may have physical or wireless access to devices

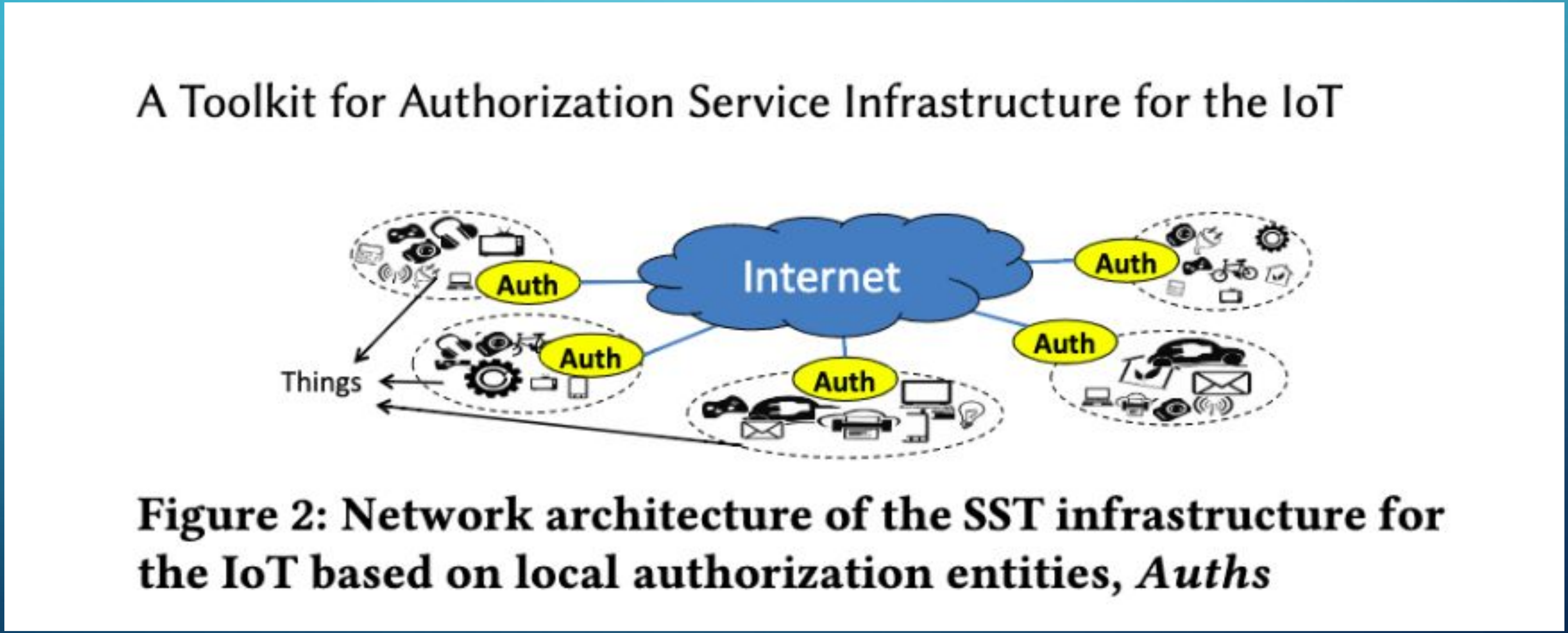
Scalability

- Number of devices and volume of traffic

SOLUTION: SECURE SWARM TOOLKIT

- Uses local authorization entities (Auths)
  - Written in Java (memory safe)
  - Supports connectionless protocols
  - Full database with encrypted credentials
  - https://github.com/iotauth/iotauth

NETWORK ARCHITECTURE USING AUTH



A Toolkit for Authorization Service Infrastructure for the IoT

Figure 2: Network architecture of the SST infrastructure for the IoT based on local authorization entities, *Auths*

SOFTWARE COMPONENTS

- Secure Communication Accessor
  - Internally manages keys for secure communication
  - Developers don’t need to manage keys or operations

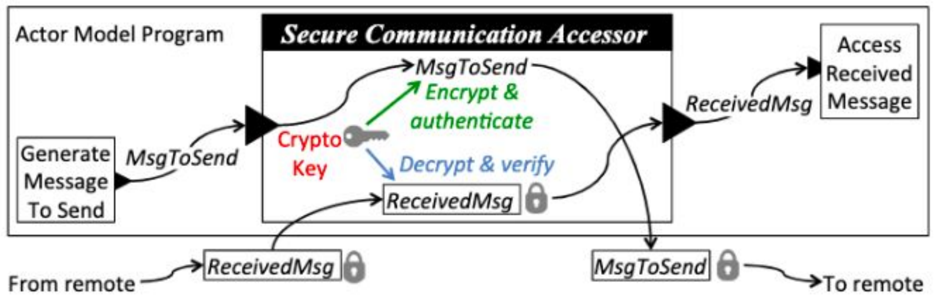


Figure 3: Software component for accessing authorization service, *secure communication accessor*

AUTH KEYS

- Auth shares multiple symmetric keys with entities
- Distribution Key: shared between Auth and an entity (or multiple entities)
  - Used to securely transmit Session Keys
- Session Key: given to two entities authorized to communicate with each other
  - Messages between entities are encrypted with this key

CLIENT – SERVER COMMUNICATION WITH AUTH

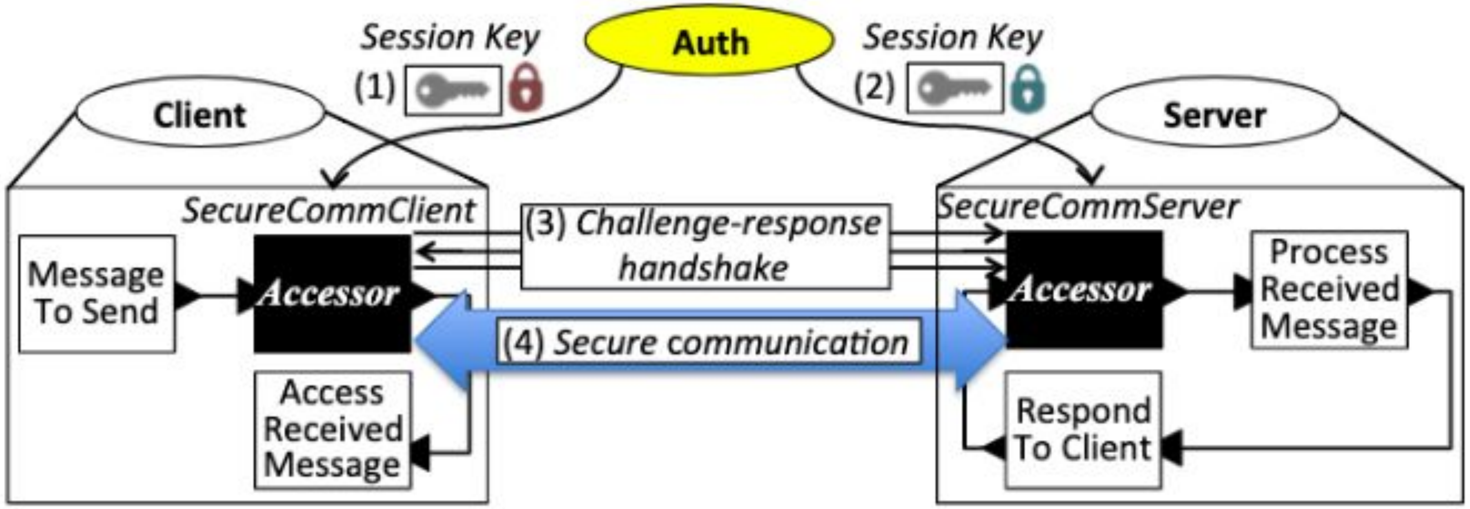
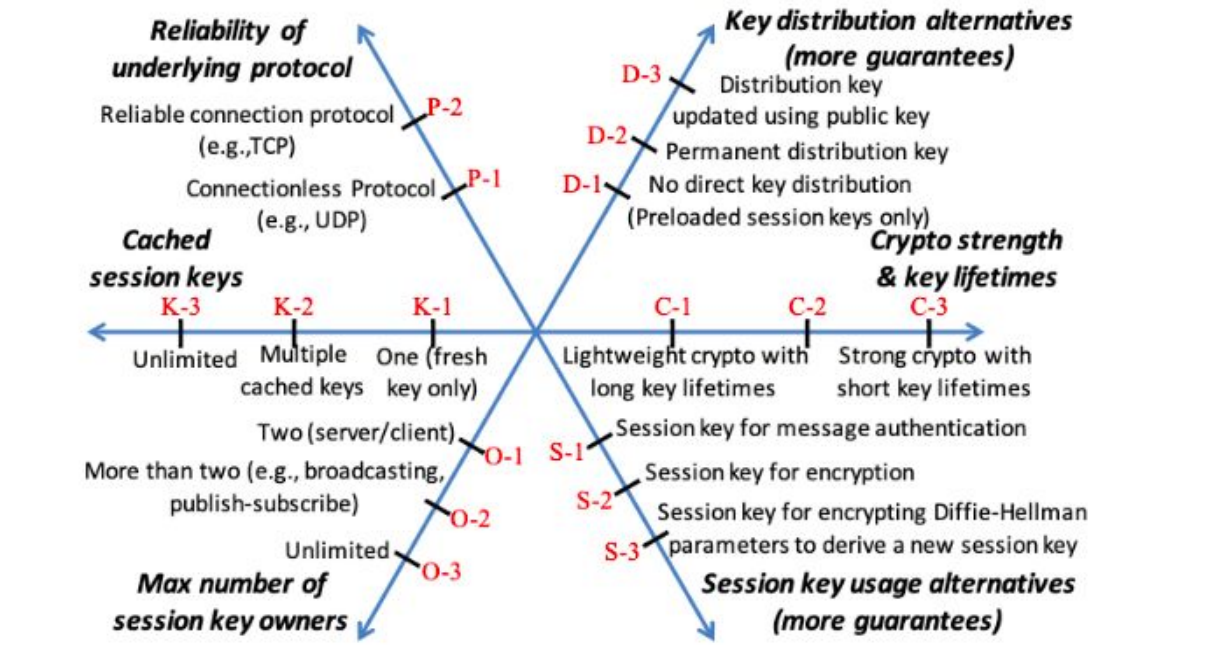


Figure 4: Process of building a secure connection between Client and Server

HETEROGENEITY

- SST supports multiple configurations for different device needs
  - Resource constrained devices trade off computationally expensive features



Reliability of underlying protocol
Reliable connection protocol (e.g.,TCP)
P-2
Connectionless Protocol (e.g., UDP)
P-1
Key distribution alternatives (more guarantees)
D-3
Distribution key updated using public key
D-2
Permanent distribution key
D-1
No direct key distribution (Preloaded session keys only)
Cached session keys
K-3
K-2
K-1
Unlimited
Multiple cached keys
One (fresh key only)
Crypto strength & key lifetimes
C-1
C-2
C-3
Lightweight crypto with long key lifetimes
Strong crypto with short key lifetimes
Two (server/client)
O-1
More than two (e.g., broadcasting, publish-subscribe)
O-2
Unlimited
O-3
Max number of session key owners
S-1
Session key for message authentication
S-2
Session key for encryption
S-3
Session key for encrypting Diffie-Hellman parameters to derive a new session key
Session key usage alternatives (more guarantees)

Figure 5: Security configuration space provided by Auth

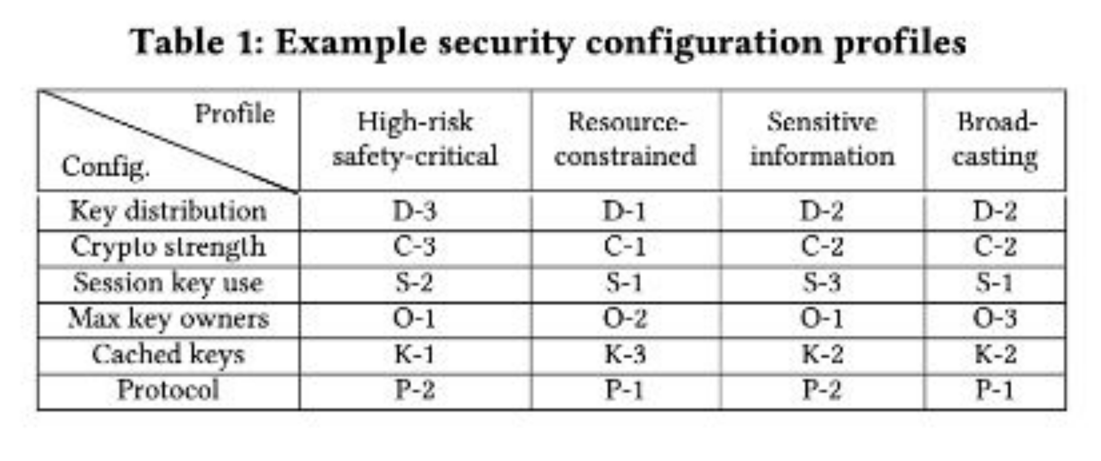


Table 1: Example security configuration profiles

Config. \ Profile	High-risk safety-critical	Resource-constrained	Sensitive information	Broad-casting
Key distribution	D-3	D-1	D-2	D-2
Crypto strength	C-3	C-1	C-2	C-2
Session key use	S-2	S-1	S-3	S-1
Max key owners	O-1	O-2	O-1	O-3
Cached keys	K-1	K-3	K-2	K-2
Protocol	P-2	P-1	P-2	P-1

HETEROGENEITY

- Strong security for critical devices
  - Ex: power grid, banking system, etc.

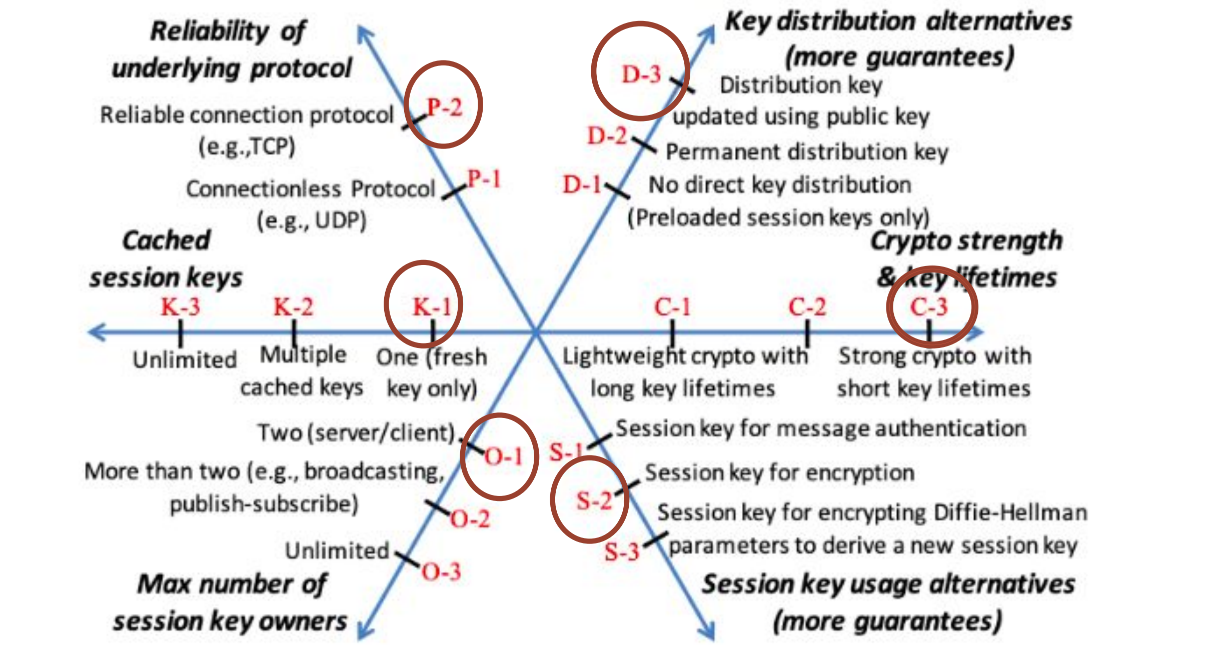

Reliability of underlying protocol
Reliable connection protocol (e.g.,TCP)
P-2
Connectionless Protocol (e.g., UDP)
P-1
Key distribution alternatives (more guarantees)
D-3
Distribution key updated using public key
D-2
Permanent distribution key
D-1
No direct key distribution (Preloaded session keys only)
Cached session keys
K-3
Unlimited
K-2
Multiple cached keys
K-1
One (fresh key only)
Crypto strength & key lifetimes
C-1
Lightweight crypto with long key lifetimes
C-2
C-3
Strong crypto with short key lifetimes
Two (server/client)
O-1
More than two (e.g., broadcasting, publish-subscribe)
O-2
Unlimited
O-3
Max number of session key owners
S-1
Session key for message authentication
S-2
Session key for encryption
S-3
Session key for encrypting Diffie-Hellman parameters to derive a new session key
Session key usage alternatives (more guarantees)

Figure 5: Security configuration space provided by Auth

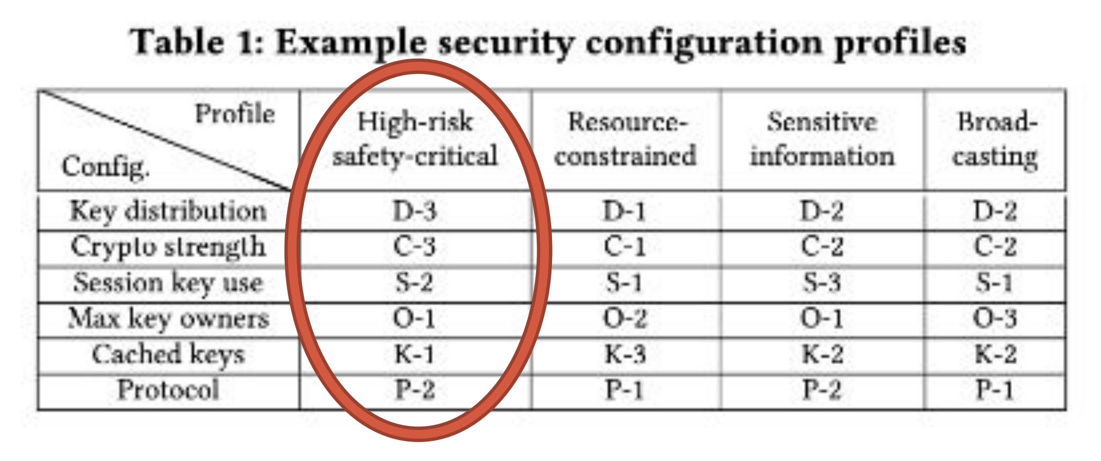
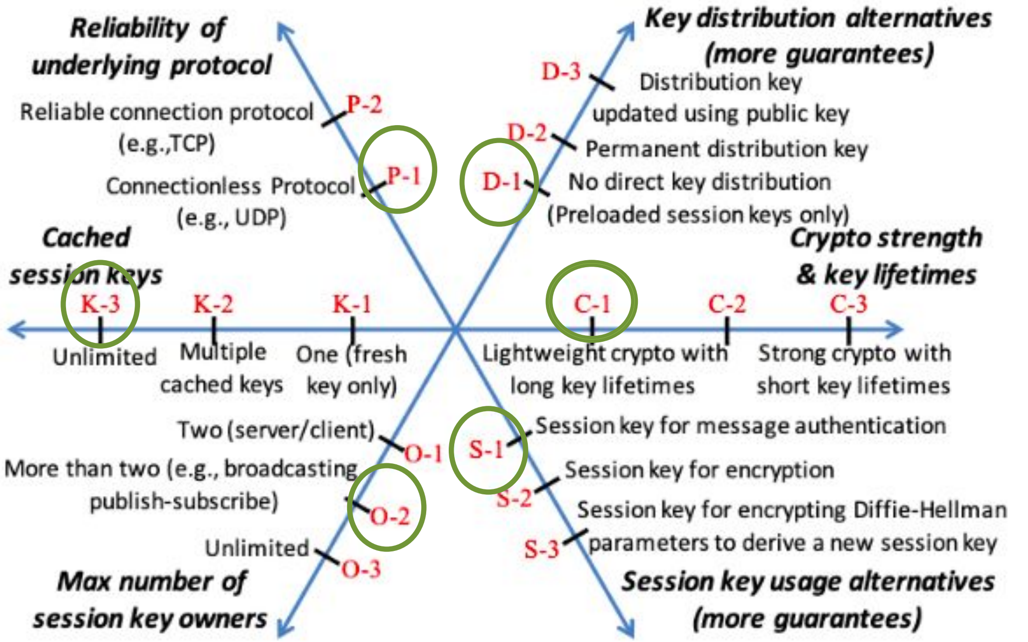


Table 1: Example security configuration profiles

Config. \ Profile	High-risk safety-critical	Resource-constrained	Sensitive information	Broad-casting
Key distribution	D-3	D-1	D-2	D-2
Crypto strength	C-3	C-1	C-2	C-2
Session key use	S-2	S-1	S-3	S-1
Max key owners	O-1	O-2	O-1	O-3
Cached keys	K-1	K-3	K-2	K-2
Protocol	P-2	P-1	P-2	P-1

HETEROGENEITY

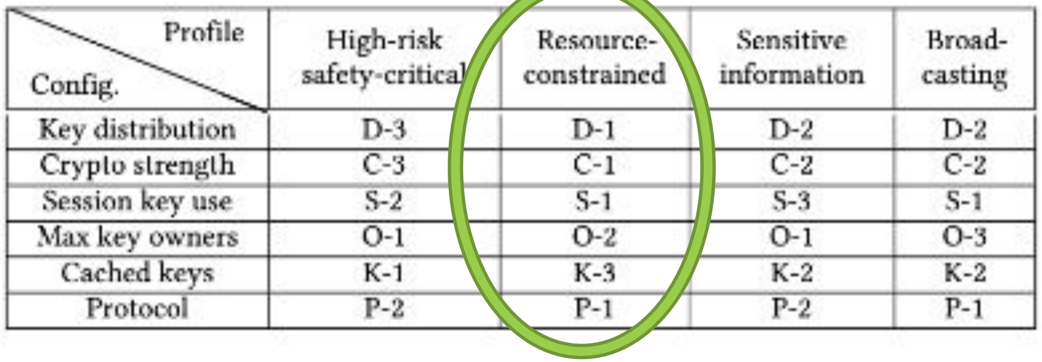
- Lower overhead for resource constrained Devices
  - Ex: battery powered, intermittent connectivity



Reliability of underlying protocol
Reliable connection protocol (e.g.,TCP)
P-2
Connectionless Protocol (e.g., UDP)
P-1
Key distribution alternatives (more guarantees)
D-3
Distribution key updated using public key
D-2
Permanent distribution key
D-1
No direct key distribution (Preloaded session keys only)
Cached session keys
K-3
Unlimited
K-2
Multiple cached keys
K-1
One (fresh key only)
Crypto strength & key lifetimes
C-1
Lightweight crypto with long key lifetimes
C-2
C-3
Strong crypto with short key lifetimes
Two (server/client)
O-1
More than two (e.g., broadcasting publish-subscribe)
O-2
Unlimited
O-3
Max number of session key owners
S-1
Session key for message authentication
S-2
Session key for encryption
S-3
Session key for encrypting Diffie-Hellman parameters to derive a new session key
Session key usage alternatives (more guarantees)

Figure 5: Security configuration space provided by Auth

Table 1: Example security configuration profiles



Profile / Config.	High-risk safety-critical	Resource-constrained	Sensitive information	Broad-casting
Key distribution	D-3	D-1	D-2	D-2
Crypto strength	C-3	C-1	C-2	C-2
Session key use	S-2	S-1	S-3	S-1
Max key owners	O-1	O-2	O-1	O-3
Cached keys	K-1	K-3	K-2	K-2
Protocol	P-2	P-1	P-2	P-1

OPEN ENVIRONMENT

- Intrusion Detection Systems (IDS) can be deployed with Auth
  - All traffic flows through Auth
  - More precise and fewer devices to monitor

SCALABILITY

- Large number of entities
  - Multiple auths can be deployed on a network
  - Only additional overhead is communication between auths

- High volumes of traffic
  - Auth supports one to many communication using shared keys

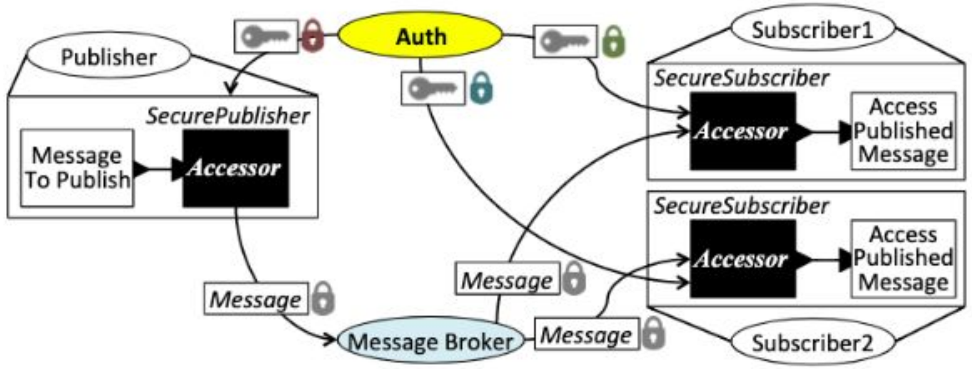


Figure 7: Process of scalable key sharing for publish-subscribe communication

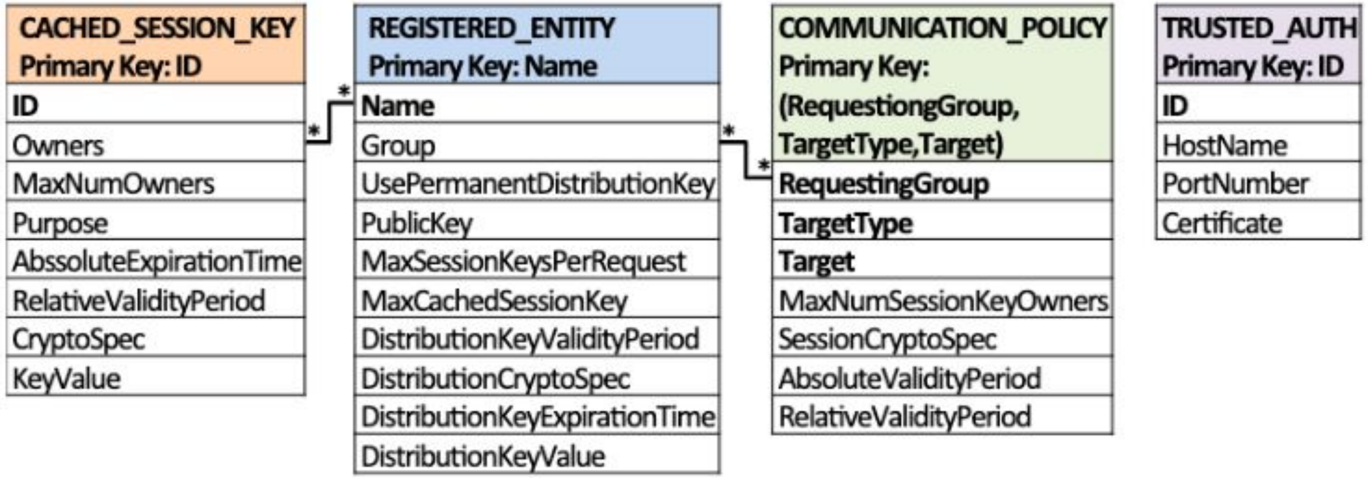
IMPLEMENTATION

(Auth)enticate and authorize locally registered devices

Interact with other Auths for communication with other networks

AUTH DATABASE

- Cached Session Key
- Registered Entity
- Communication Policy
- Trust Auth



CACHED_SESSION_KEY Primary Key: ID
ID
Owners
MaxNumOwners
Purpose
AbssoluteExpirationTime
RelativeValidityPeriod
CryptoSpec
KeyValue

REGISTERED_ENTITY Primary Key: Name
Name
Group
UsePermanentDistributionKey
PublicKey
MaxSessionKeysPerRequest
MaxCachedSessionKey
DistributionKeyValidityPeriod
DistributionCryptoSpec
DistributionKeyExpirationTime
DistributionKeyValue

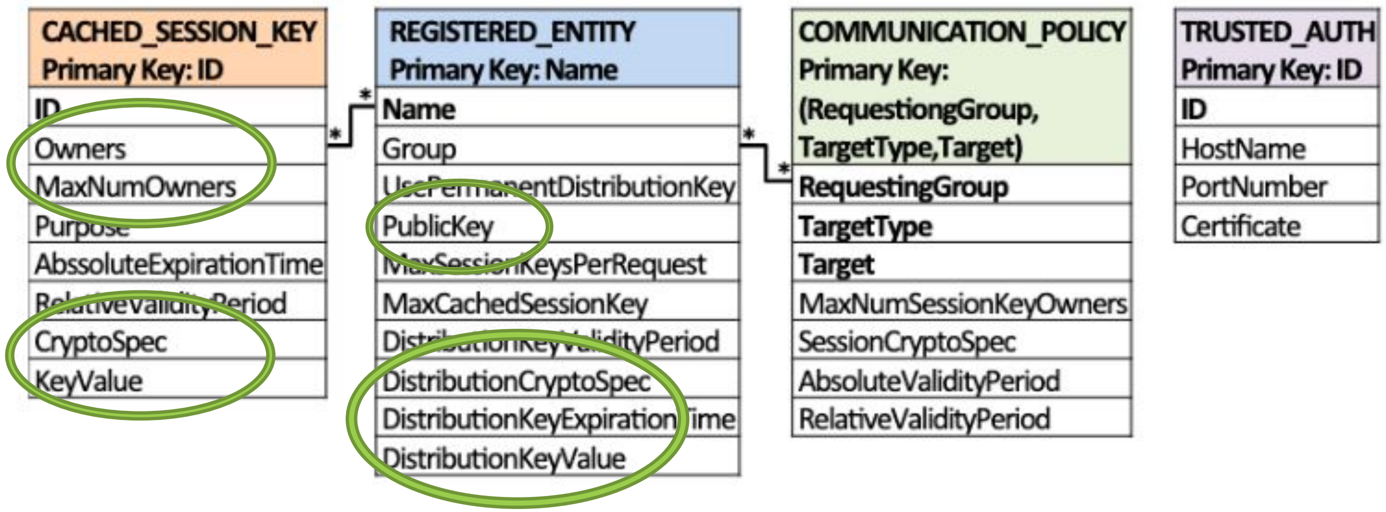
COMMUNICATION_POLICY Primary Key: (RequestiongGroup, TargetType,Target)
RequestingGroup
TargetType
Target
MaxNumSessionKeyOwners
SessionCryptoSpec
AbsoluteValidityPeriod
RelativeValidityPeriod

TRUSTED_AUTH Primary Key: ID
ID
HostName
PortNumber
Certificate

Figure 8: Auth database table schema (* for many-to-many relationship)

AUTH DATABASE

- Cached Session Key
- Registered Entity
- Communication Policy
- Trust Auth



CACHED_SESSION_KEY Primary Key: ID
ID
Owners
MaxNumOwners
Purpose
AbsoluteExpirationTime
RelativeValidityPeriod
CryptoSpec
KeyValue

REGISTERED_ENTITY Primary Key: Name
Name
Group
UsePermanentDistributionKey
PublicKey
MaxSessionKeysPerRequest
MaxCachedSessionKey
DistributionKeyValidityPeriod
DistributionCryptoSpec
DistributionKeyExpirationTime
DistributionKeyValue

COMMUNICATION_POLICY Primary Key: (RequestingGroup, TargetType,Target)
RequestingGroup
TargetType
Target
MaxNumSessionKeyOwners
SessionCryptoSpec
AbsoluteValidityPeriod
RelativeValidityPeriod

TRUSTED_AUTH Primary Key: ID
ID
HostName
PortNumber
Certificate

Figure 8: Auth database table schema (* for many-to-many relationship)

REGISTRATION WITH AUTH

- Entities must be registered to use the Auth infrastructure
- Process depends on device capabilities
  - Can update distribution key using public-key
  - Set up permanent distribution key if public-key is not possible
  - Severely constrained devices can ship with a preloaded session key

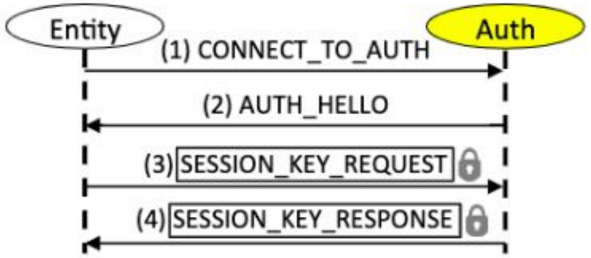


Figure 9: Steps for *Auth* – *Entity* communication for session key distribution; a padlock next to a message indicates that the message is encrypted and/or authenticated

COMMUNICATION

Supports

- Client – Server

- Publisher – Subscriber

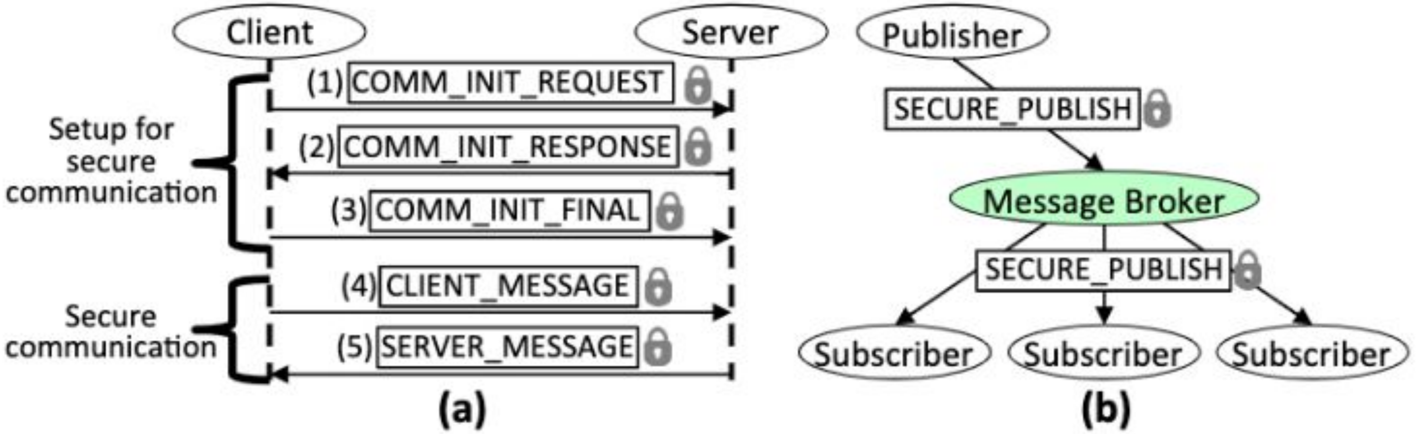


Figure 10: Process of secure communication for (a) Server-client (b) Publish-subscribe

SECURITY ANALYSIS

- CIA principles
- Confidentiality: messages should only be accessible by intended recipients
- Integrity and Authenticity: message content should not change between sender and receiver

ANALYSIS

- Formal analysis tool written in Alloy
- Why Alloy?
  - Allows for bounded testing
    - 5 unique Auth/Entity gourps
    - Up to 10 messages
  - Generates examples where CIA is violated

RESULTS

- With a maximum trace length of 10, analyzer can potentially explore 175^{10} messages
- Analyzer found 17 counterexamples where at least one property was violated.
  - Due to missing assumptions in the model

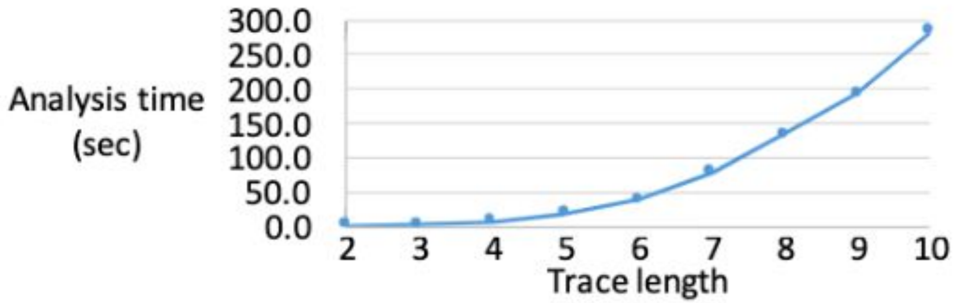


Figure 14: Verification times on the Auth model.

SCALABILITY

- Using Auths on the edge is much more scalable than centralized authentication
- Proof of Scalability
    - R = ratio of entities to auths
    - By adding more auths when new entities connect, R and overhead stay constant

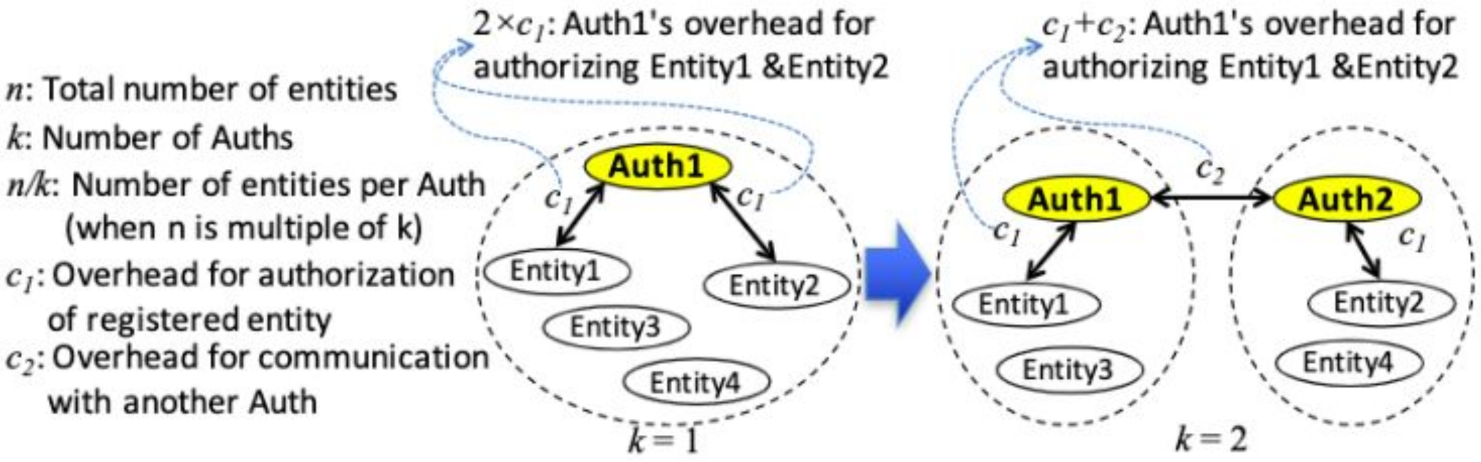

n: Total number of entities
k: Number of Auths
n/k: Number of entities per Auth (when n is multiple of k)
c_1: Overhead for authorization of registered entity
c_2: Overhead for communication with another Auth
$2\times c_1$: Auth1's overhead for authorizing Entity1 &Entity2
c_1+c_2: Auth1's overhead for authorizing Entity1 &Entity2
Auth1
Auth2
c_1
c_2
Entity1
Entity2
Entity3
Entity4
$k=1$
$k=2$

Figure 15: Division of entities into two groups registered with separate Auths

ENERGY CONSUMPTION

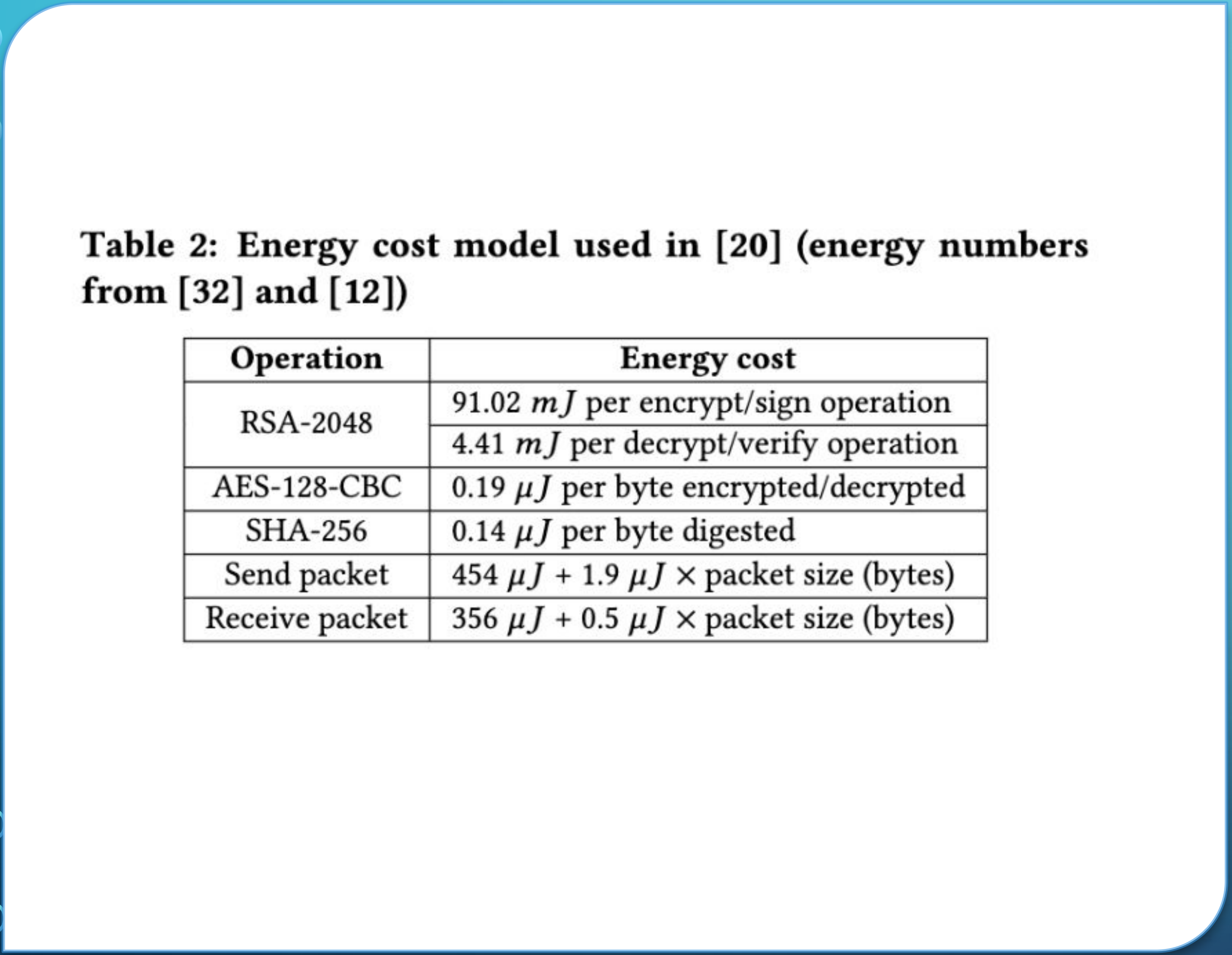


Table 2: Energy cost model used in [20] (energy numbers from [32] and [12])

Operation	Energy cost
RSA-2048	91.02 mJ per encrypt/sign operation
	4.41 mJ per decrypt/verify operation
AES-128-CBC	0.19 μJ per byte encrypted/decrypted
SHA-256	0.14 μJ per byte digested
Send packet	454 μJ + 1.9 μJ × packet size (bytes)
Receive packet	356 μJ + 0.5 μJ × packet size (bytes)

- Cryptographic tradeoffs lead to considerably less power consumption
- Using connectionless protocols (UDP) also decreased power usage

PAPER ASSUMPTIONS

- There are a lot…
  - All Auths are trusted and cannot be controlled by attacker
  - Attacker is not capable of impersonating Auth (no man in the middle attack)
  - Paper does not consider security guarantees against DoS attacks or depletion of resources

CONTRIBUTIONS

- SST has a lot of adaptability and has a lot of potential for many different applications
- Auth can be configured to work on many devices
- Scalability

CRITIQUES

- The paper made some very strong assumptions
  - Particularly that an Auth could not be impersonated
- Auth seems like a prime target for DoS type attacks
  - I think they could have at least done some DoS analysis, especially since they talk about how the system handles large volumes of traffic

QUESTIONS FROM GITHUB

- @nikorev, Critical: Where is the "ideal" in figure 4? This graph seems to through a bunch of ranges across and lining them up over each other in a non-coherent way. Where is the maximum security? Where is the maximum speed? etc.
- @albero94, Critical: It seems that the Auth have knowledge of other Auth in the network. In a very large network, how is this achieved? Do they have direct knowledge, or the paper assumes there is a higher-level component that can provide this information?
- @grahamschock, Critical: The paper describes that current network security solutions are not scalable. Why aren't they scalable? What prevents them from being scalable? Is it because of unrealistic energy consumption?

QUESTIONS FROM GITHUB

- @reesealanj, Critical: The Auth entity described in the paper communicates over UDP. This seems like the exact opposite thing you'd want for a security critical system due to the fact that you can never guarantee information has received?(Maybe this comes from me not understanding networking well enough.)
- @lrhpak, Comprehencion: Why is Auth better than other security systems?