Privacy, Discovery, and Authentication for the Internet of Things

David J. Wu
Stanford University

Ankur Taly
Google

Asim Shankar
Google

Dan Boneh
Stanford University

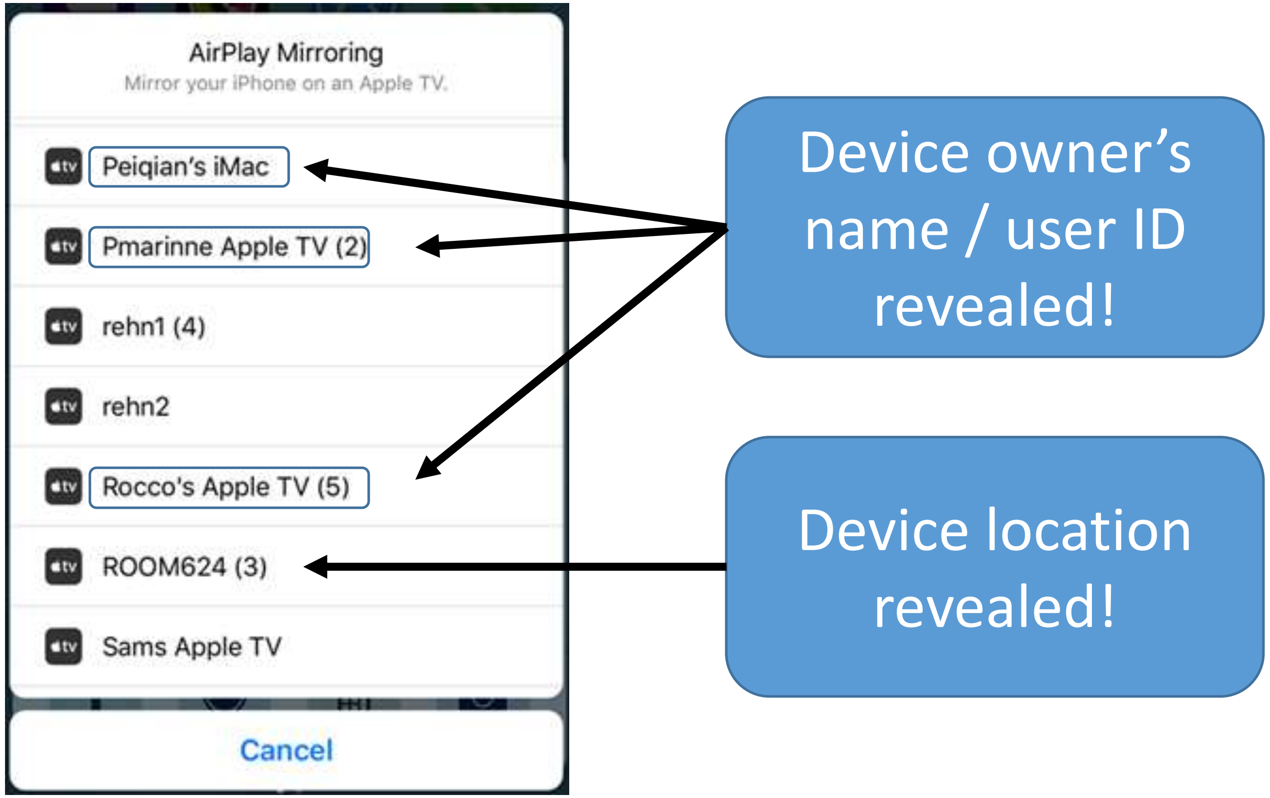
The Internet of Things (IoT)

Lots of smart devices, but only useful if users can <u>discover</u> them!

Private Service Discovery

Many existing service discovery protocols: Multicast DNS (mDNS), Apple Bonjour, Bluetooth Low Energy (BLE)

A typical discovery protocol



AirPlay Mirroring
Mirror your iPhone on an Apple TV.

- Peiqian's iMac
- Pmarinne Apple TV (2)
- rehn1 (4)
- rehn2
- Rocco's Apple TV (5)
- ROOM624 (3)
- Sams Apple TV

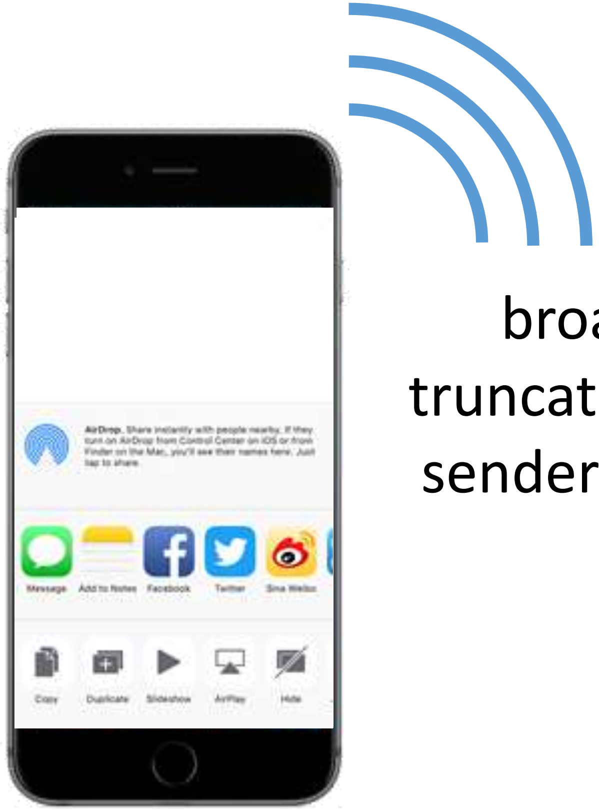
Device owner's name / user ID revealed!

Device location revealed!

Screenshot taken on a public Wireless network

Private Service Discovery

Privacy problems exist in many protocols


AirDrop. Share instantly with people nearby. If they turn on AirDrop from Control Center on iOS or from Finder on the Mac, you'll see their names here. Just tap to share.

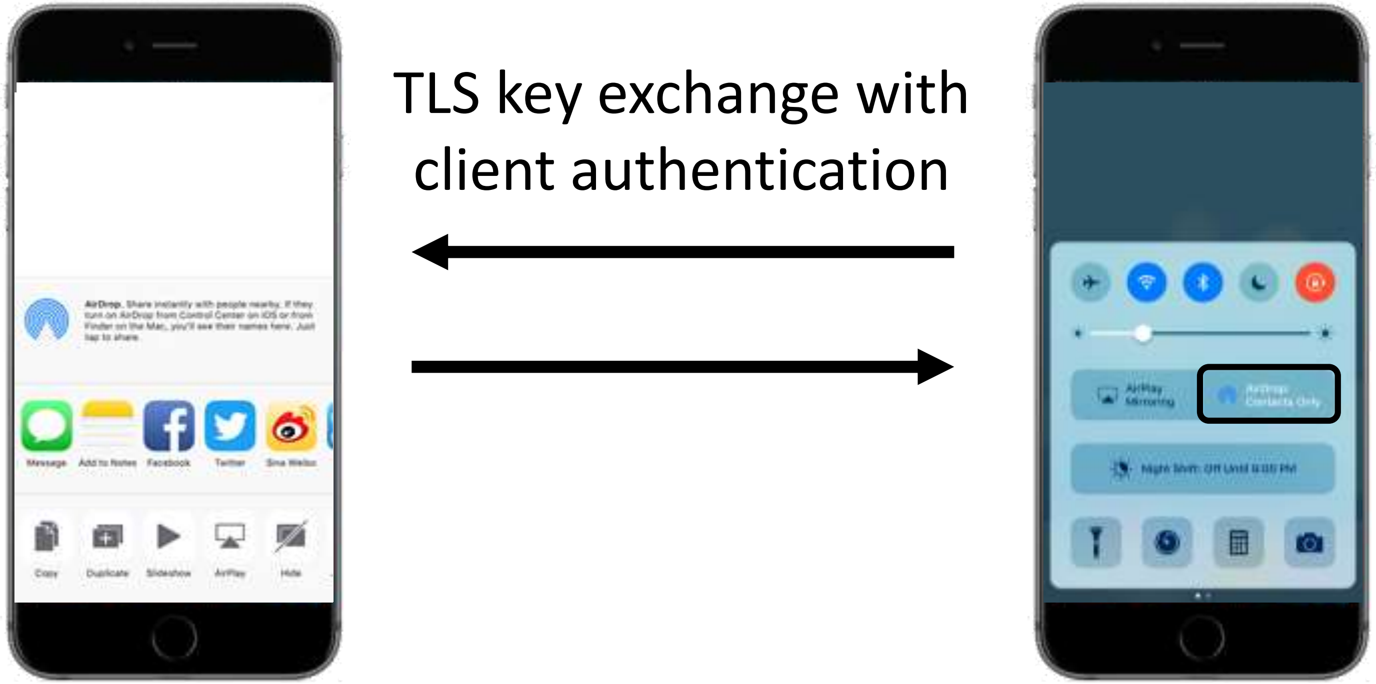
broadcast:
truncated hash of
sender’s identity

contacts-only mode:
device should only
be discoverable by
users in their
contacts list

AirDrop protocol for peer-to-peer file sharing

Private Service Discovery

Privacy problems exist in many protocols



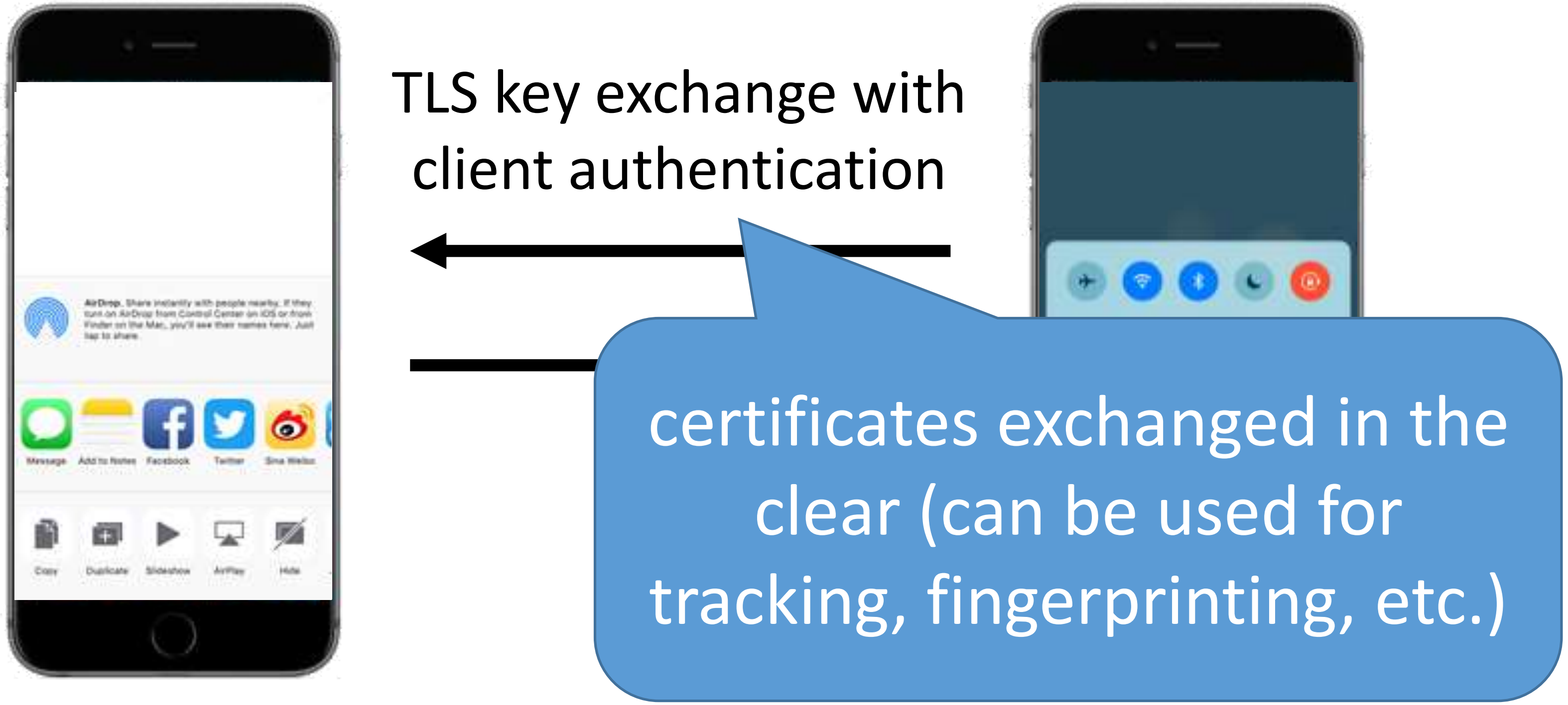
TLS key exchange with client authentication

if broadcast containing ID of user in contact list, then start local service and advertise over mDNS

AirDrop protocol for peer-to-peer file sharing

Private Service Discovery

Privacy problems exist in many protocols



AirDrop. Share instantly with people nearby. If they turn on AirDrop from Control Center on iOS or from Finder on the Mac, you'll see their names here. Just tap to share.

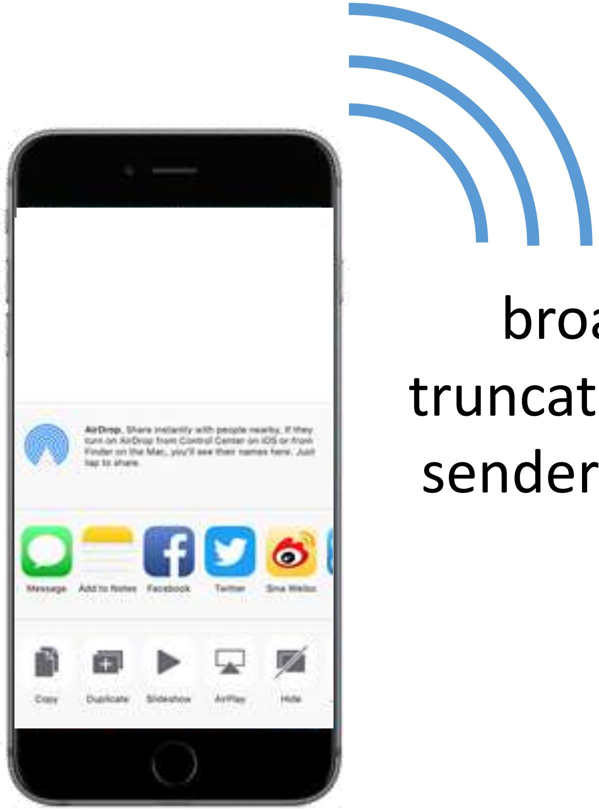
TLS key exchange with client authentication

certificates exchanged in the clear (can be used for tracking, fingerprinting, etc.)

AirDrop protocol for peer-to-peer file sharing

Private Service Discovery

Privacy problems exist in many protocols



AirDrop. Share instantly with people nearby. If they turn on AirDrop from Control Center on iOS or from Finder on the Mac, you'll see their names here. Just tap to share.

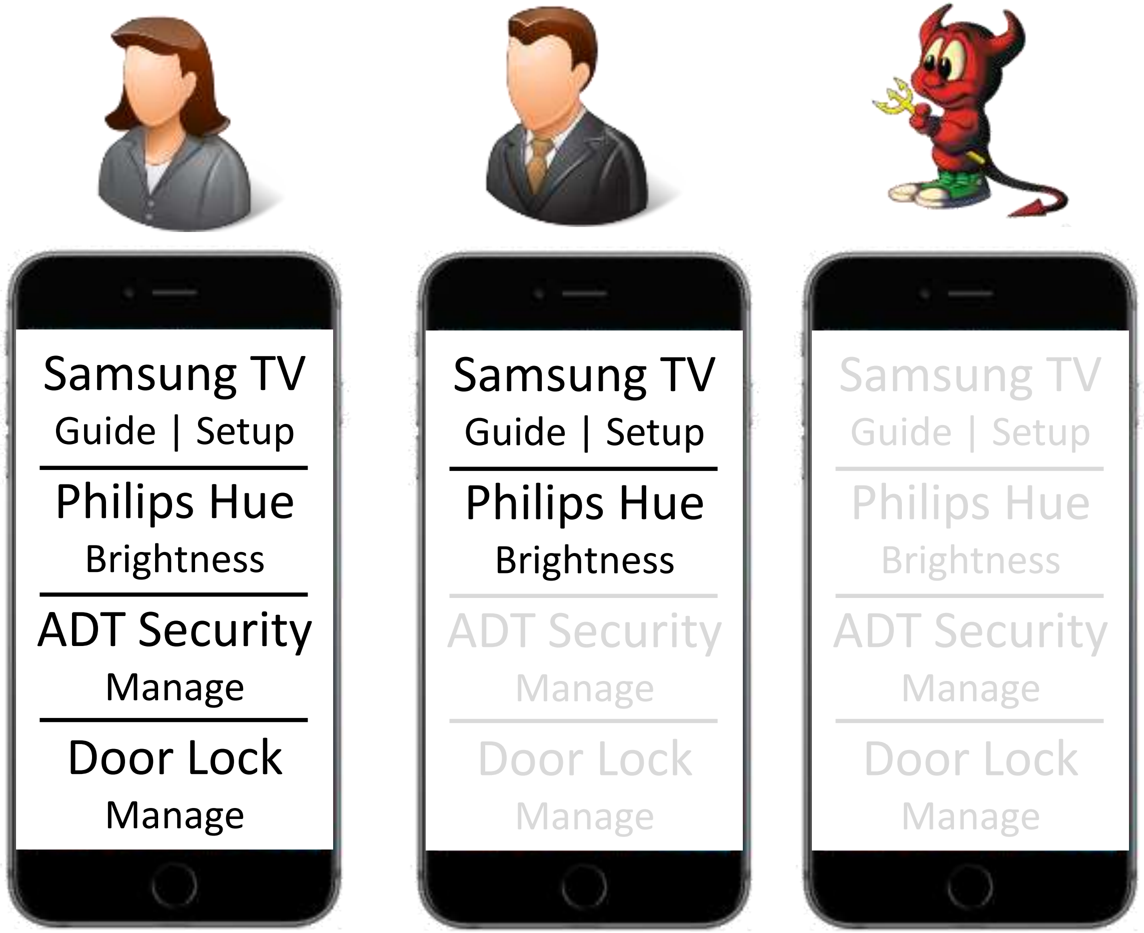
broadcast:
truncated hash of
sender's identity

no authenticity for broadcast – can be replayed to see if particular user in target's contact list

AirDrop protocol for peer-to-peer file sharing

Private Service Discovery

Each service specifies an authorization policy



Samsung TV
Guide | Setup
Philips Hue
Brightness
ADT Security
Manage
Door Lock
Manage

Alice

Samsung TV
Guide | Setup
Philips Hue
Brightness
ADT Security
Manage
Door Lock
Manage

Guest

Samsung TV
Guide | Setup
Philips Hue
Brightness
ADT Security
Manage
Door Lock
Manage

Stranger

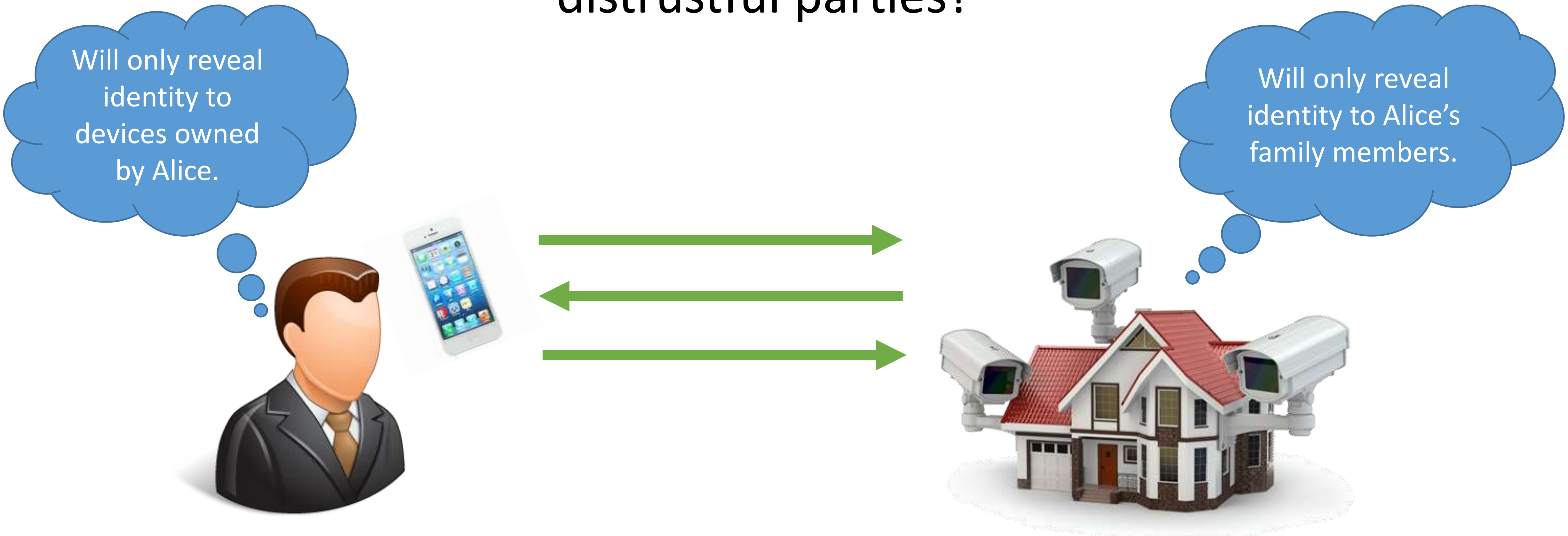
Private Service Discovery

Each service specifies an authorization policy

Mutual privacy: privacy should also hold for devices trying to discover services!

Private Mutual Authentication

How to authenticate between mutually distrustful parties?



Will only reveal identity to devices owned by Alice.

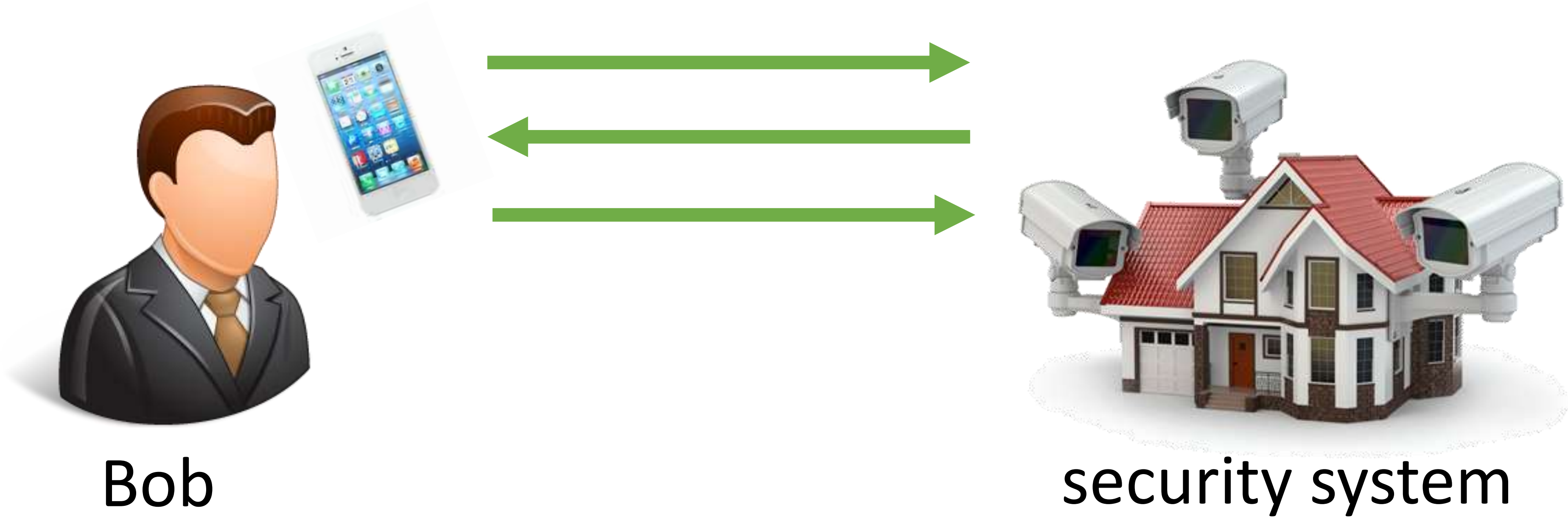
Will only reveal identity to Alice’s family members.

Bob

security system

Private Mutual Authentication

In most existing mutual authentication protocols (e.g., TLS, IKE, SIGMA), one party must reveal its identity first



Primary Protocol Requirements

- **Mutual privacy:** Identity of protocol participants are only revealed to <u>authorized</u> recipients
- **Authentic advertisements:** Service advertisements (for discovery) should be unforgeable and authentic
- **Lightweight:** privacy should be as simple as setting a flag in key-exchange (as opposed to a separate protocol – e.g., using secret handshakes [BDSSSW03])

Identity and Authorization Model

Every party has a signing + verification key, and a collection of human-readable names bound to their public keys via a certificate chain

verification key

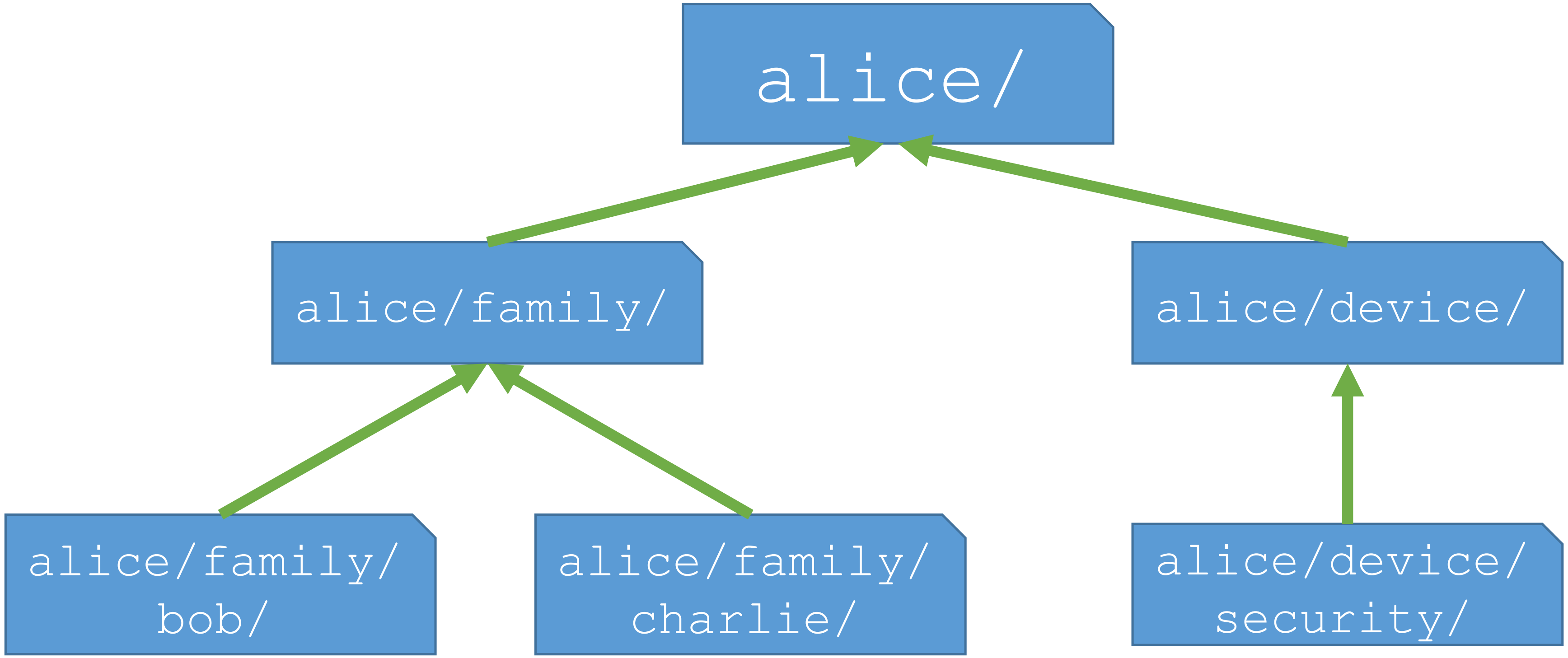
alice/family/
bob/

alice/device/
security/

popular_corp/
prod/S1234

Identity and Authorization Model

Every party has a signing + verification key, and a collection of human-readable names bound to their public keys via a certificate chain



`alice/`

`alice/family/`

`alice/device/`

`alice/family/bob/`

`alice/family/charlie/`

`alice/device/security/`

Identity and Authorization Model

Authorization decisions expressed as prefix patterns

Policy:

`alice/devices/*`

`alice/family/bob/`

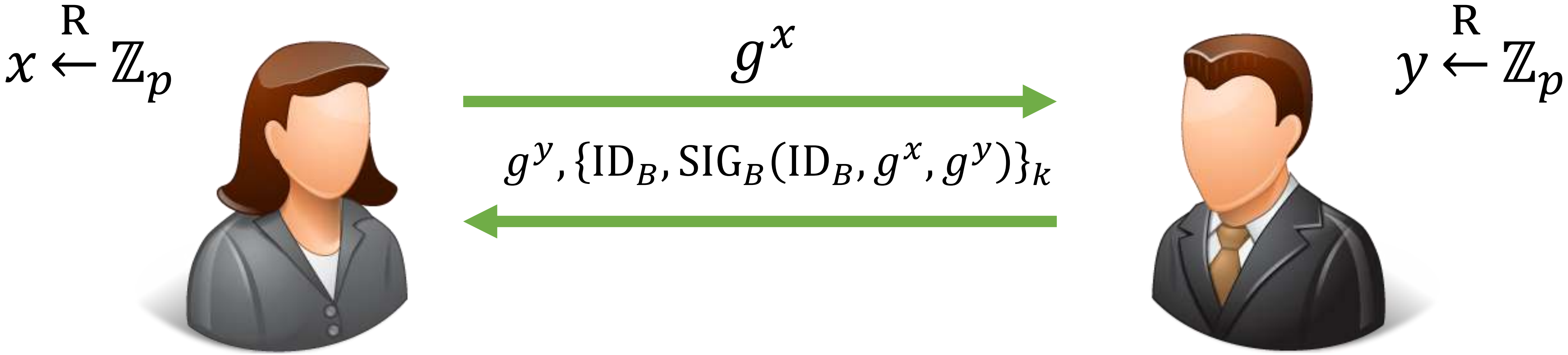
Policy:

`alice/family/*`

`alice/device/security/`

Protocol Construction

Secure Key Agreement: SIGMA-I Protocol [CK01]



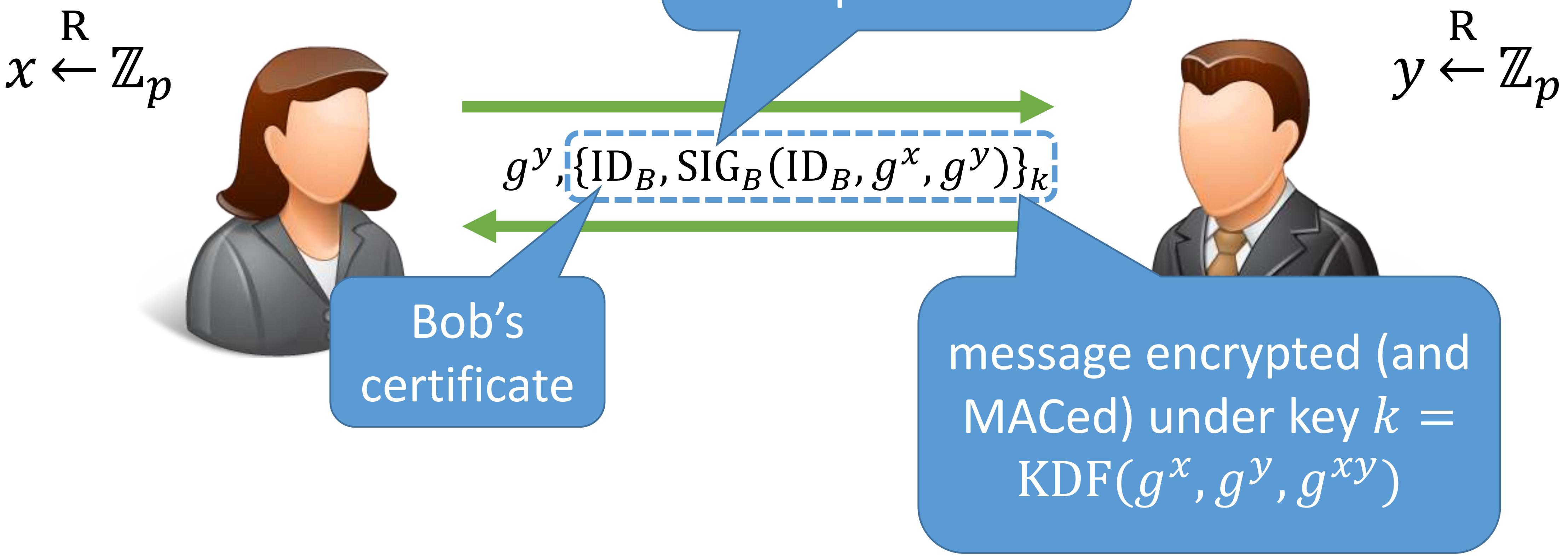
$x \xleftarrow{\mathrm{R}} \mathbb{Z}_p$

g^x

$g^y, \{\mathrm{ID}_B, \mathrm{SIG}_B(\mathrm{ID}_B, g^x, g^y)\}_k$

$y \xleftarrow{\mathrm{R}} \mathbb{Z}_p$

Secure Key Agreemen ocol [CK01]



$x \overset{\mathrm{R}}{\leftarrow} \mathbb{Z}_p$

$y \overset{\mathrm{R}}{\leftarrow} \mathbb{Z}_p$

$g^y, \{\mathrm{ID}_B, \mathrm{SIG}_B(\mathrm{ID}_B, g^x, g^y)\}_k$

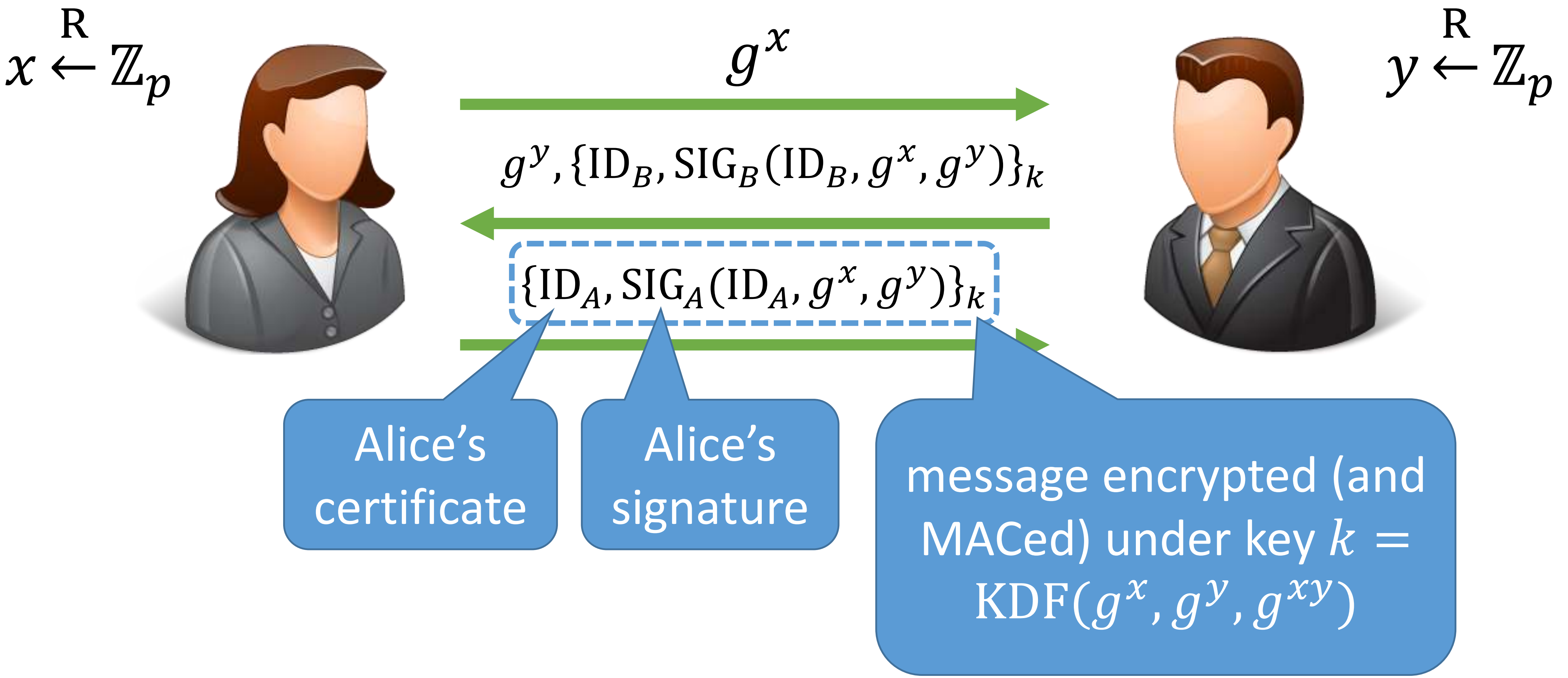
Bob's signature of the ephemeral DH exponents

Bob's certificate

message encrypted (and MACed) under key $k = \mathrm{KDF}(g^x, g^y, g^{xy})$

Note: in the actual protocol, session ids are also included for replay prevention.

Secure Key Agreement: SIGMA-I Protocol [CK01]



$x \xleftarrow{\text{R}} \mathbb{Z}_p$

$y \xleftarrow{\text{R}} \mathbb{Z}_p$

g^x

$g^y, \{\mathrm{ID}_B, \mathrm{SIG}_B(\mathrm{ID}_B, g^x, g^y)\}_k$

$\{\mathrm{ID}_A, \mathrm{SIG}_A(\mathrm{ID}_A, g^x, g^y)\}_k$

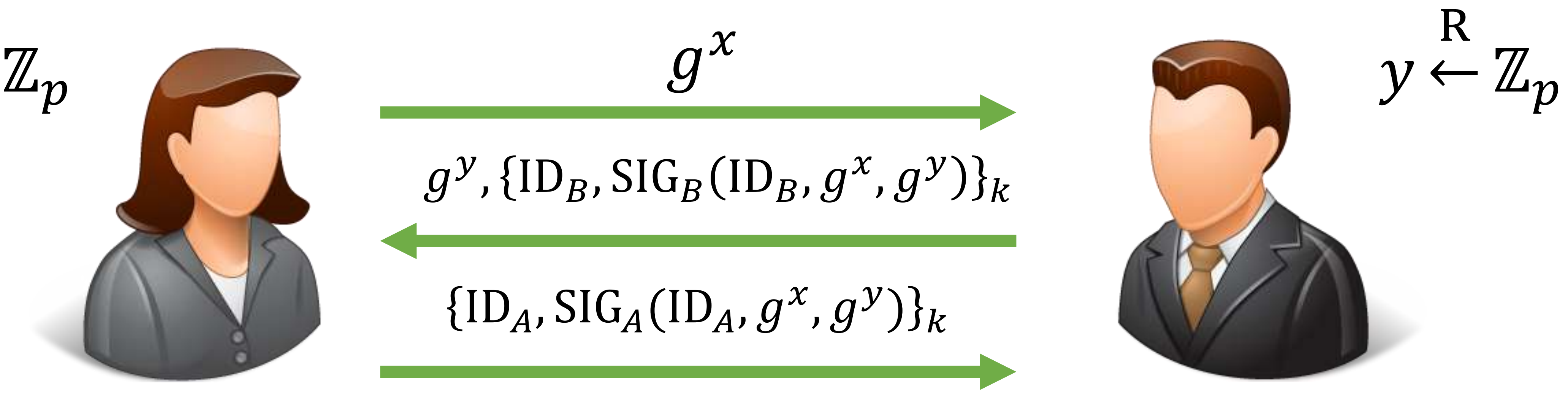
Alice’s certificate

Alice’s signature

message encrypted (and MACed) under key $k = \mathrm{KDF}(g^x, g^y, g^{xy})$

Note: in the actual protocol, session ids are also included for replay prevention.

Secure Key Agreement: SIGMA-I Protocol [CK01]



$x \overset{\mathrm{R}}{\leftarrow} \mathbb{Z}_p$

$y \overset{\mathrm{R}}{\leftarrow} \mathbb{Z}_p$

g^x

$g^y, \{\mathrm{ID}_B, \mathrm{SIG}_B(\mathrm{ID}_B, g^x, g^y)\}_k$

$\{\mathrm{ID}_A, \mathrm{SIG}_A(\mathrm{ID}_A, g^x, g^y)\}_k$

session key derived from

$$(g^x, g^y, g^{xy})$$

Note: in the actual protocol, session ids are also included for replay prevention.

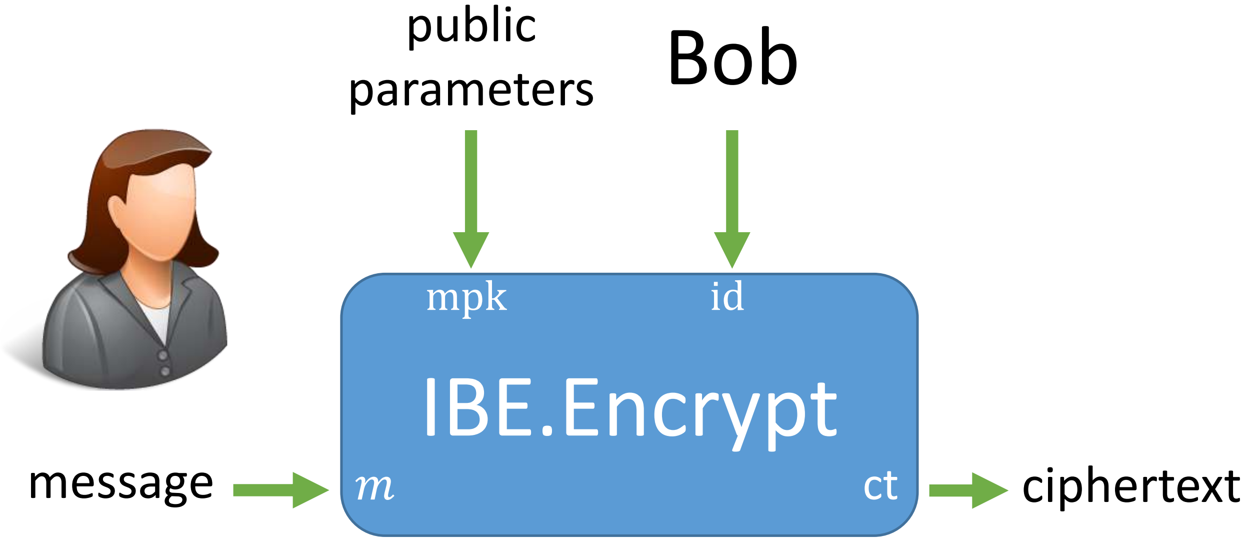
Properties of the SIGMA-I Protocol

- Mutual authentication against active network adversaries
- Hides server’s (Bob’s) identity from a passive attacker
- Hides client’s (Alice’s) identity from an active attacker

- Bob’s identity is revealed to an active attacker!

Identity Based Encryption (IBE) [Sha84, BF01, Coc01]

Public-key encryption scheme where public-keys can be arbitrary strings (identities)



public parameters

Bob

mpk

id

IBE.Encrypt

message

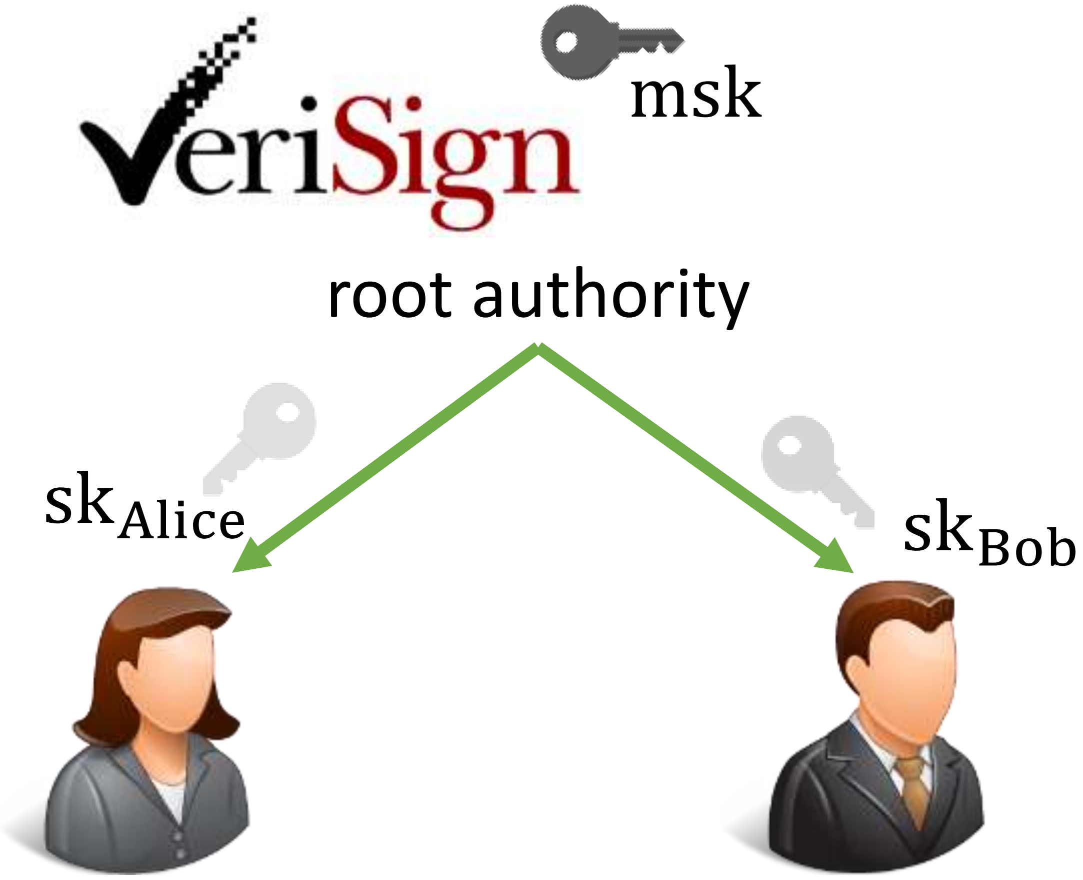
m

ct

ciphertext

Alice can encrypt a message to Bob without needing to have exchanged keys with Bob

Identity Based Encryption (IBE) [Sha84, BF01, Coc01]

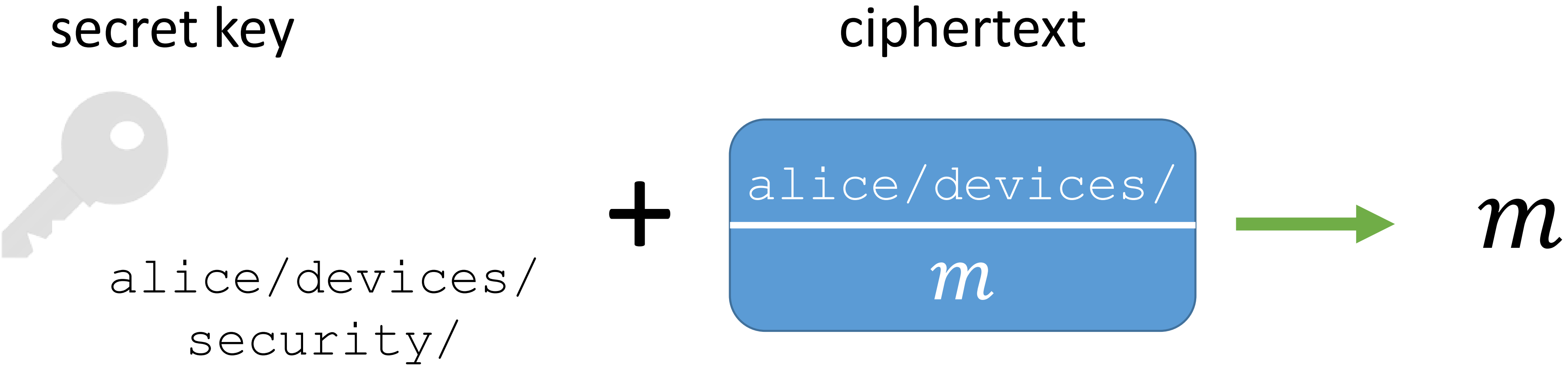


To decrypt messages, users go to a (trusted) identity provider to obtain a decryption key for their identity

Bob can decrypt all messages encrypted to his identity using $\mathrm{sk}_{\mathrm{Bob}}$

Prefix-Based Encryption

Secret-keys and ciphertexts both associated with names



secret key

`alice/devices/`
`security/`

+

ciphertext

`alice/devices/`

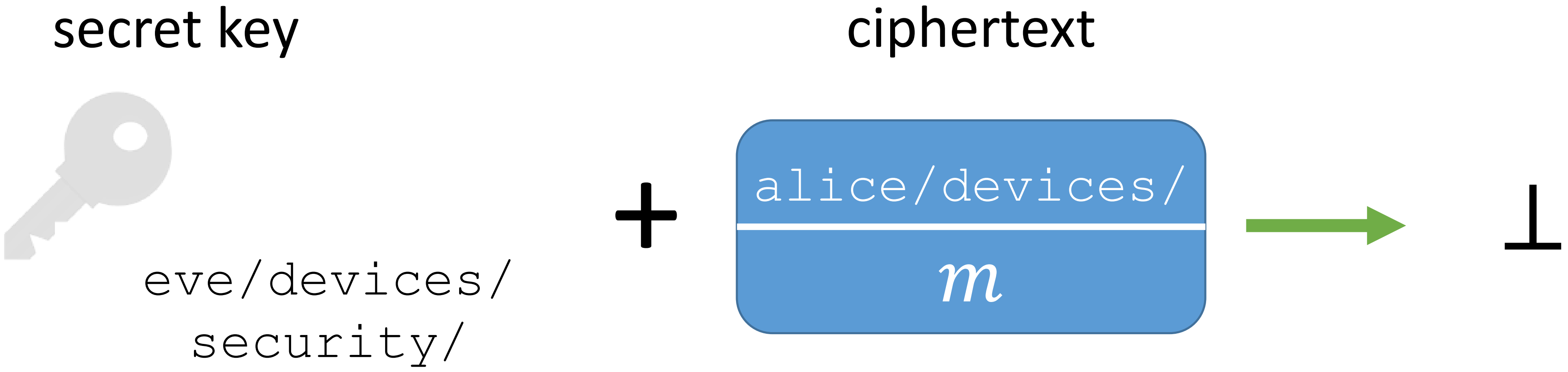
m

$\longrightarrow m$

Decryption succeeds if name in ciphertext is a prefix of the name in the secret key

Prefix-Based Encryption

Secret-keys and ciphertexts both associated with names



secret key

`eve/devices/security/`

+

ciphertext

`alice/devices/`

m

$\longrightarrow \perp$

Decryption fails if name in ciphertext is <u>not</u> a prefix of the name in the secret key

Prefix-Based Encryption

Can be leveraged for prefix-based policies

Policy:

`alice/devices/*`

Bob encrypts his message to the identity `alice/devices/`. Any user with a key that begins with `alice/devices/` can decrypt.

Prefix-Based Encryption from IBE [LW14]

Encryption is just IBE encryption

Secret key for a name is a collection of IBE secret keys, one for each prefix:

`alice/devices/security/`

`alice/`

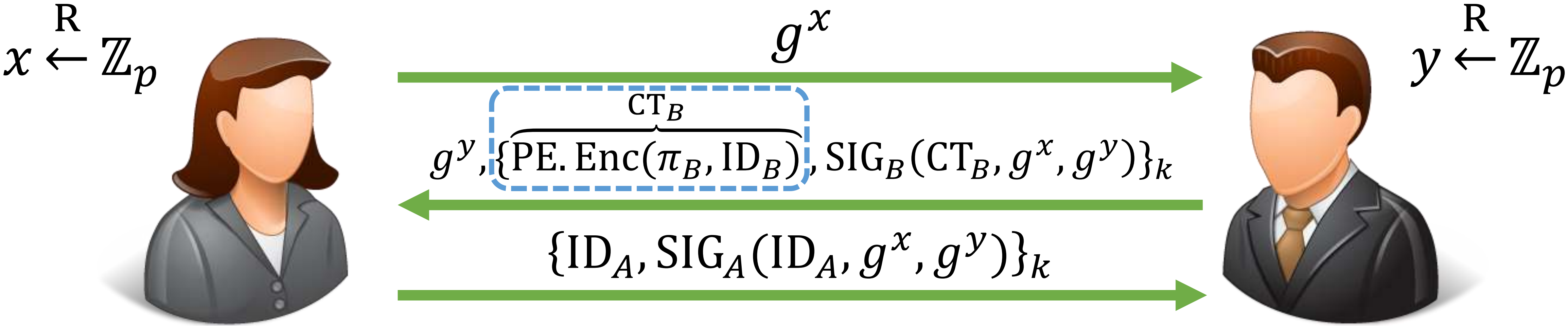
`alice/devices/`

`alice/devices/security/`

can decrypt encryptions to all prefixes of `alice/devices/security`

Private Mutual Authentication

Key idea: encrypt certificate using prefix-based encryption



$x \xleftarrow{R} \mathbb{Z}_p$

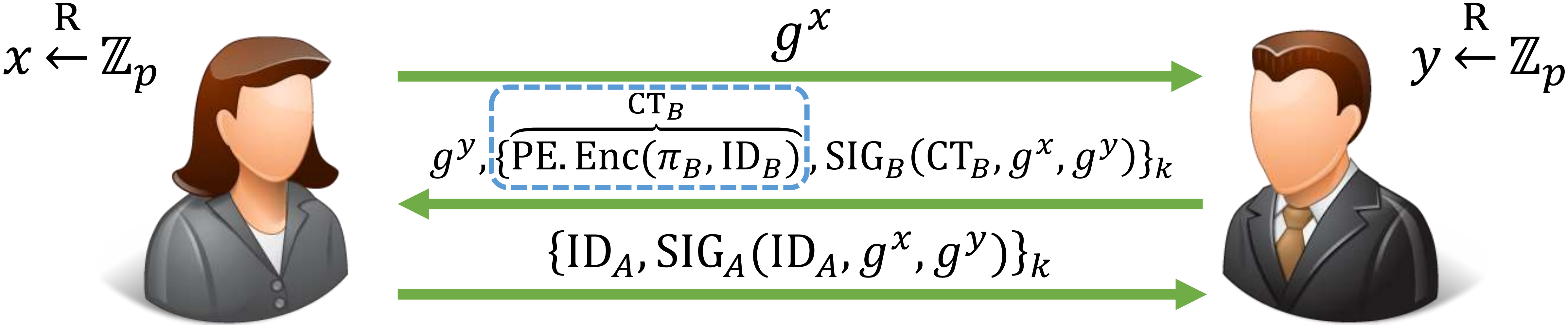
$y \xleftarrow{R} \mathbb{Z}_p$

g^x

$g^y, \boxed{\{\overbrace{\text{PE.Enc}(\pi_B, \text{ID}_B)}^{\text{CT}_B}}, \text{SIG}_B(\text{CT}_B, g^x, g^y)\}_k$

$\{\text{ID}_A, \text{SIG}_A(\text{ID}_A, g^x, g^y)\}_k$

Private Mutual Authentication



$x \xleftarrow{\text{R}} \mathbb{Z}_p$

$y \xleftarrow{\text{R}} \mathbb{Z}_p$

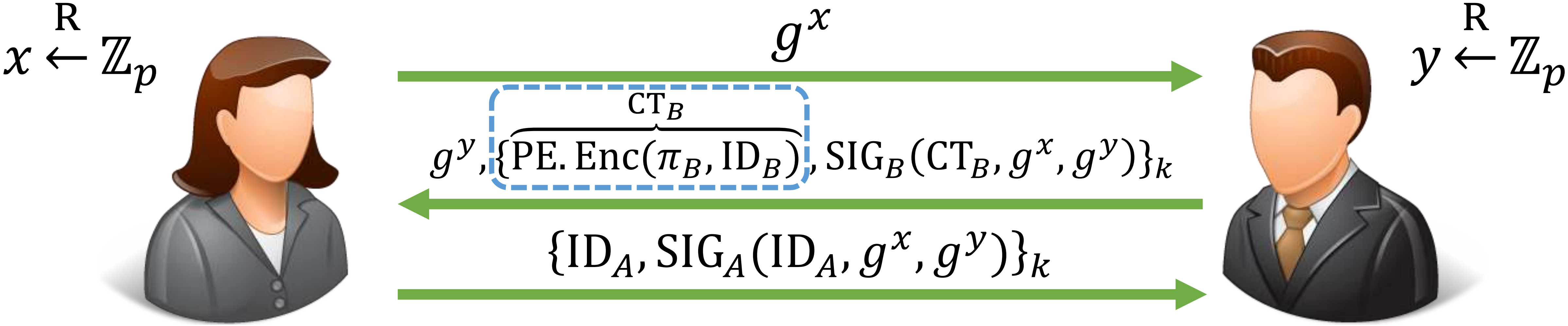
g^x

$g^y, \boxed{\{\overbrace{\text{PE.Enc}(\pi_B, \text{ID}_B)}^{\text{CT}_B}}, \text{SIG}_B(\text{CT}_B, g^x, g^y)\}_k$

$\{\text{ID}_A, \text{SIG}_A(\text{ID}_A, g^x, g^y)\}_k$

- **Privacy for Alice's identity:** Alice sends her identity only after verifying Bob's identity
- **Privacy for Bob's identity:** Only users with a key that satisfies Bob's policy can decrypt his identity

Private Mutual Authentication



$x \overset{R}{\leftarrow} \mathbb{Z}_p$

g^x

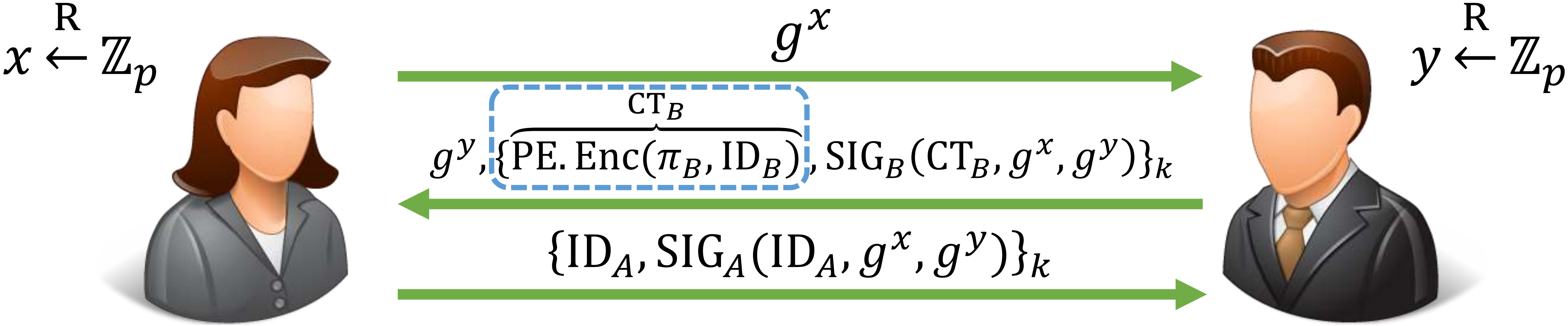
$g^y, \boxed{\{\overbrace{\mathrm{PE.Enc}(\pi_B, \mathrm{ID}_B)}^{\mathrm{CT}_B}}, \mathrm{SIG}_B(\mathrm{CT}_B, g^x, g^y)\}_k$

$\{\mathrm{ID}_A, \mathrm{SIG}_A(\mathrm{ID}_A, g^x, g^y)\}_k$

$y \overset{R}{\leftarrow} \mathbb{Z}_p$

- **Client overhead:** Alice must perform prefix-based decryption on each flow
- **Server overhead:** Bob must perform prefix-based encryption on each handshake, but this encrypted identity can be cached and reused

Private Mutual Authentication



$x \xleftarrow{\mathrm{R}} \mathbb{Z}_p$

$y \xleftarrow{\mathrm{R}} \mathbb{Z}_p$

g^x

$g^y, \{\boxed{\overbrace{\mathrm{PE.Enc}(\pi_B, \mathrm{ID}_B)}^{\mathrm{CT}_B}}, \mathrm{SIG}_B(\mathrm{CT}_B, g^x, g^y)\}_k$

$\{\mathrm{ID}_A, \mathrm{SIG}_A(\mathrm{ID}_A, g^x, g^y)\}_k$

Provably secure in the Canetti-Krawczyk model of key-exchange assuming Hash-DH and security of underlying cryptographic primitives

Private Service Discovery

Two pieces: service announcements and private mutual authentication

Principal design goals:

- **Private discovery:** Only authorized clients can learn service details
- **Authentic service announcements:** Announcements are authenticated and unforgeable
- **0-RTT private mutual authentication:** Clients can subsequently connect to service and include application data on initial flow

Private Service Discovery: Broadcast

Key idea: encrypt service broadcast using prefix encryption

$s \overset{\text{R}}{\leftarrow} \mathbb{Z}_p$

$$\text{PE.Enc}\left(\pi_S, \left(\text{ID}_S, g^s, \text{SIG}_S(\text{ID}_S, g^s)\right)\right)$$

Private Service Discovery: Broadcast

Key idea: encrypt service broadca efix

$s \xleftarrow{\text{R}} \mathbb{Z}_p$

service identity

signature for authenticity

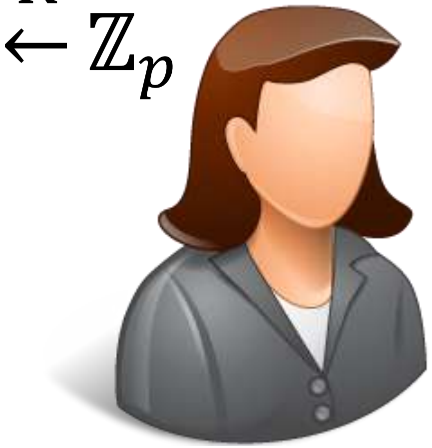
$$\mathrm{PE.Enc}\left(\pi_S, \left(\mathrm{ID}_S, g^s, \mathrm{SIG}_S\left(\mathrm{ID}_S, g^s\right)\right)\right)$$

authorization policy

semi-static DH share (for 0-RTT authentication)

Private Service Discovery: Mutual Authentication

$x \xleftarrow{R} \mathbb{Z}_p$

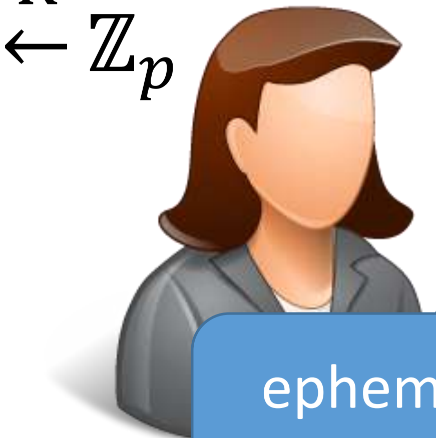


$g^x, \{\mathrm{ID}_S, \mathrm{ID}_A, \mathrm{SIG}_A(\mathrm{ID}_S, \mathrm{ID}_A, g^s, g^x)\}_k$

Private Service tual Authentication

sender and receiver identities

$x \overset{\text{R}}{\leftarrow} \mathbb{Z}_p$



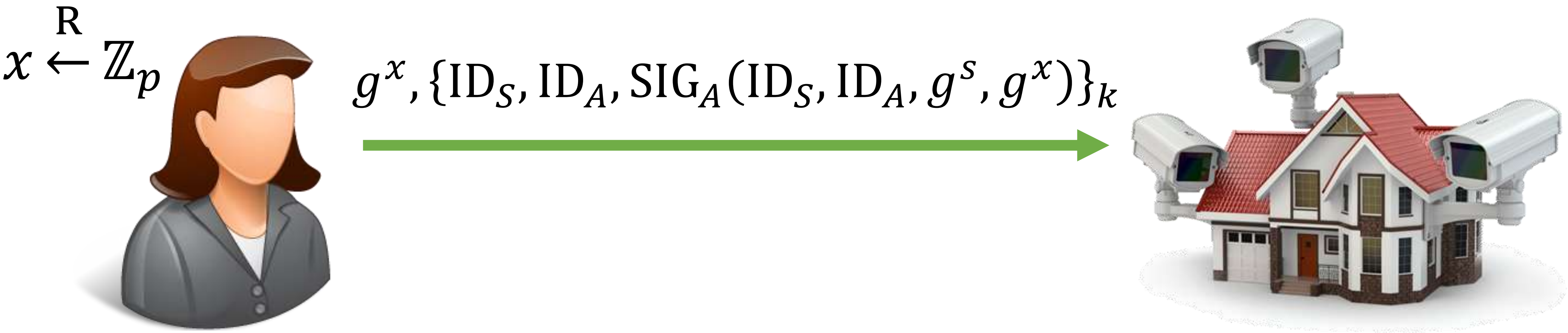
$g^x, \{\text{ID}_S, \text{ID}_A, \text{SIG}_A(\text{ID}_S, \text{ID}_A, g^s, g^x)\}_k$

ephemeral DH exponent

message encrypted (and MACed) under handshake key

$k = \text{KDF}(g^s, g^x, g^{sx}, \text{C} \rightarrow \text{S})$

Private Service Discovery: Mutual Authentication



$x \xleftarrow{\mathrm{R}} \mathbb{Z}_p$

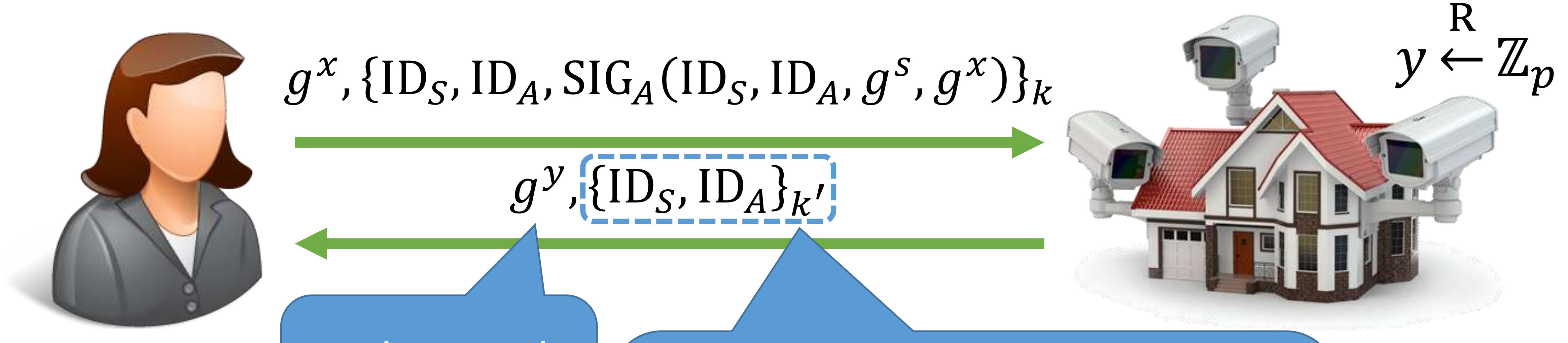
$g^x, \{\mathrm{ID}_S, \mathrm{ID}_A, \mathrm{SIG}_A(\mathrm{ID}_S, \mathrm{ID}_A, g^s, g^x)\}_k$

application data can also be sent in the first message flow under another key derived from g^s, g^x, and g^{sx}:

$$k_{\mathrm{app}} = \mathrm{KDF}(g^s, g^x, g^{sx}, \mathrm{app})$$

No forward secrecy for early application data sent during lifetime of broadcast.

Private Service Discovery: Mutual Authentication



$y \xleftarrow{\mathrm{R}} \mathbb{Z}_p$

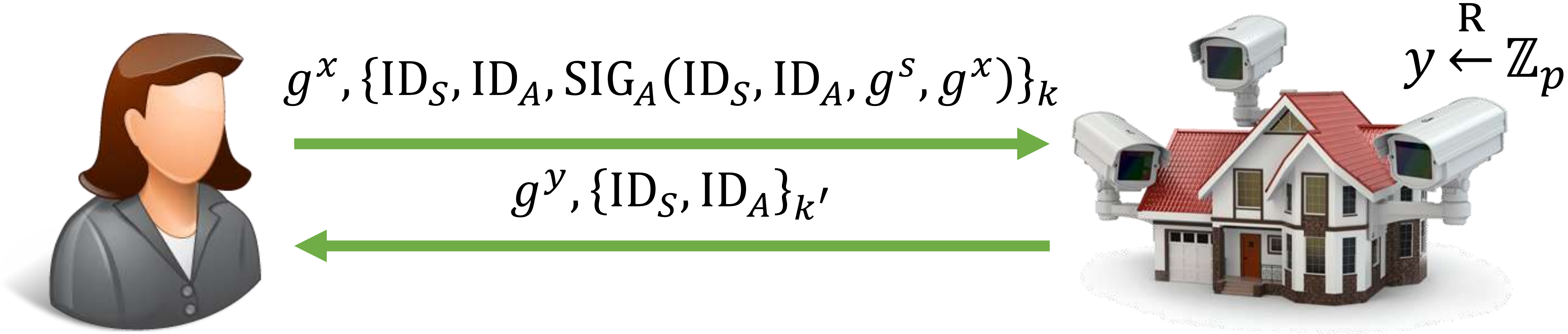
$g^x, \{\mathrm{ID}_S, \mathrm{ID}_A, \mathrm{SIG}_A(\mathrm{ID}_S, \mathrm{ID}_A, g^s, g^x)\}_k$

$g^y, \{\mathrm{ID}_S, \mathrm{ID}_A\}_{k'}$

ephemeral DH exponent

message encrypted (and MACed) under handshake key $k' = \mathrm{KDF}(g^s, g^x, g^{sx}, \mathrm{S} \rightarrow \mathrm{C})$

Private Service Discovery: Mutual Authentication



$y \overset{R}{\leftarrow} \mathbb{Z}_p$

$g^x, \{\mathrm{ID}_S, \mathrm{ID}_A, \mathrm{SIG}_A(\mathrm{ID}_S, \mathrm{ID}_A, g^s, g^x)\}_k$

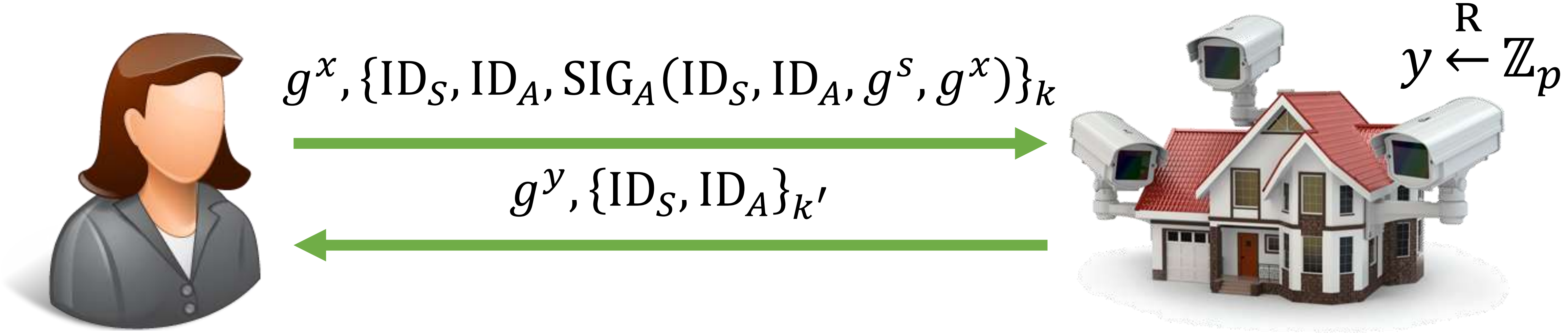
$g^y, \{\mathrm{ID}_S, \mathrm{ID}_A\}_{k'}$

final session key derived from both semi-static and ephemeral shares:

$$\mathrm{KDF}(g^s, g^x, g^y, g^{sx}, g^{xy})$$

Recovers forward secrecy for session messages.

Private Service Discovery: Mutual Authentication



$y \overset{R}{\leftarrow} \mathbb{Z}_p$

$g^x, \{\mathrm{ID}_S, \mathrm{ID}_A, \mathrm{SIG}_A(\mathrm{ID}_S, \mathrm{ID}_A, g^s, g^x)\}_k$

$g^y, \{\mathrm{ID}_S, \mathrm{ID}_A\}_{k'}$

Provably secure in an (extended) Canetti-Krawczyk model of key-exchange assuming Hash-DH and Strong-DH in the random oracle model and security of underlying cryptographic primitives

Implementation and Benchmarks

- Instantiated IBE scheme with Boneh-Boyen (BB_2) IBE scheme (`DCLXVI` library)
- Integrated private mutual authentication and private service discovery protocols into the Vanadium open-source framework for building distributed applications

`https://github.com/vanadium/`

Implementation and Benchmarks

	Intel Edison	Raspberry Pi	Nexus 5X	Desktop
SIGMA-I	252.1 ms	88.0 ms	91.6 ms	5.3 ms
Private Mutual Auth.	1694.3 ms	326.1 ms	360.4 ms	9.5 ms
Slowdown	6.7x	3.7x	3.9x	1.8x

Comparison of private mutual authentication protocol with non-private SIGMA-I protocol

Note: x86 assembly optimizations for pairing curve operations available only on desktop

Implementation and Benchmarks

$$\underbrace{\mathrm{id}}_{\text{16 bytes}},\ \underbrace{\mathrm{PE.Enc}}_{\text{208 bytes}}\left(\pi_S, \left(\underbrace{\mathrm{ID}_S}_{\text{500 bytes}}, \underbrace{g^s}_{\text{32 bytes}}, \underbrace{\mathrm{SIG}_S}_{\text{64 bytes}}(\mathrm{ID}_S, g^s)\right)\right)$$

- For private service discovery protocol, a typical service advertisement is $\approx$ 820 bytes (for single policy pattern)
- Can broadcast using mDNS (supports packets of size up to 1300 bytes)

Implementation and Benchmarks

$$\underbrace{\mathrm{id}}_{\text{16 bytes}}, \underbrace{\mathrm{PE.Enc}}_{\text{208 bytes}}\left(\pi_S, \left(\underbrace{\mathrm{ID}_S}_{\text{500 bytes}}, \underbrace{g^s}_{\text{32 bytes}}, \underbrace{\mathrm{SIG}_S(\mathrm{ID}_S, g^s)}_{\text{64 bytes}}\right)\right)$$

Processing advertisement requires 1 IBE decryption and 1 ECDSA verification:

$267 \text{ ms} + 11 \text{ ms} = 278 \text{ ms}$ on Nexus 5x

Conclusions

- Existing key-exchange and service discovery protocols do not provide privacy controls

- Prefix-based encryption can be combined very naturally with existing key-exchange protocols to provide privacy + authenticity
- Overhead of resulting protocol small enough that protocols can run on many existing devices

Questions?