Privacy, Discovery, and Authentication for the Internet of Things

David J. Wu
Stanford University

Ankur Taly
Google

Asim Shankar
Google

Dan Boneh
Stanford University

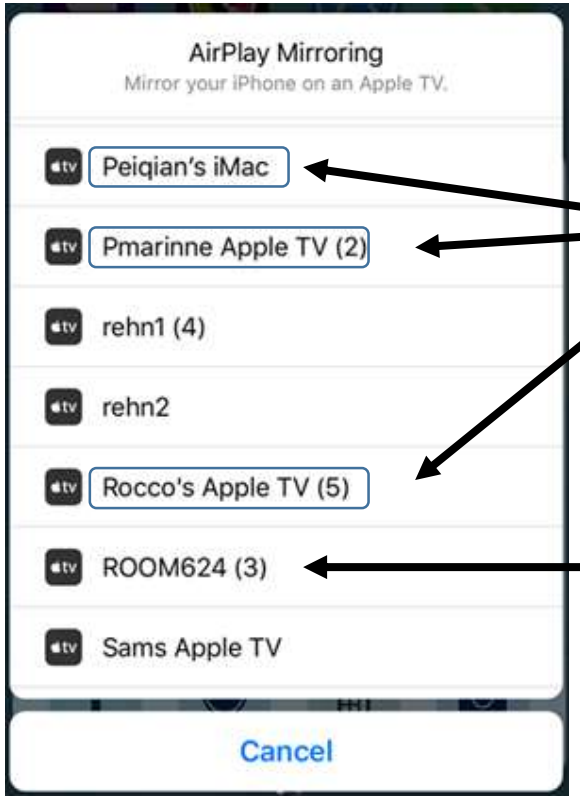
The Internet of Things (IoT)

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



Device owner's name / user ID revealed!

Device location revealed!

Screenshot taken on a public Wireless network

Private Service Discovery

Each service specifies an authorization policy

Alice

Technician

Cleaning Service

Stranger

Private Service Discovery

Each service specifies an authorization policy




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

Alice Technician Cleaning Service Stranger

Private Mutual Authentication

How to authenticate between mutually distrustful parties?

Will only reveal identity to devices owned by Alice.

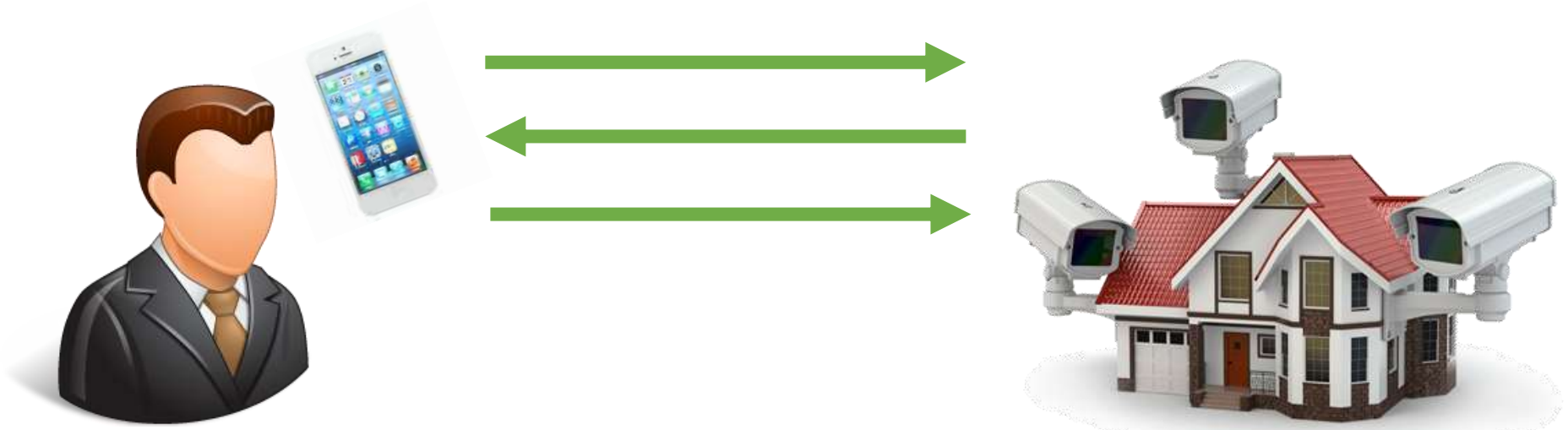
Bob

Will only reveal identity to Alice's family members.

security system

Private Mutual Authentication

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



Bob

security system

Primary Protocol Requirements

- **Mutual privacy:** Identity of protocol participants are only revealed to authorized recipients

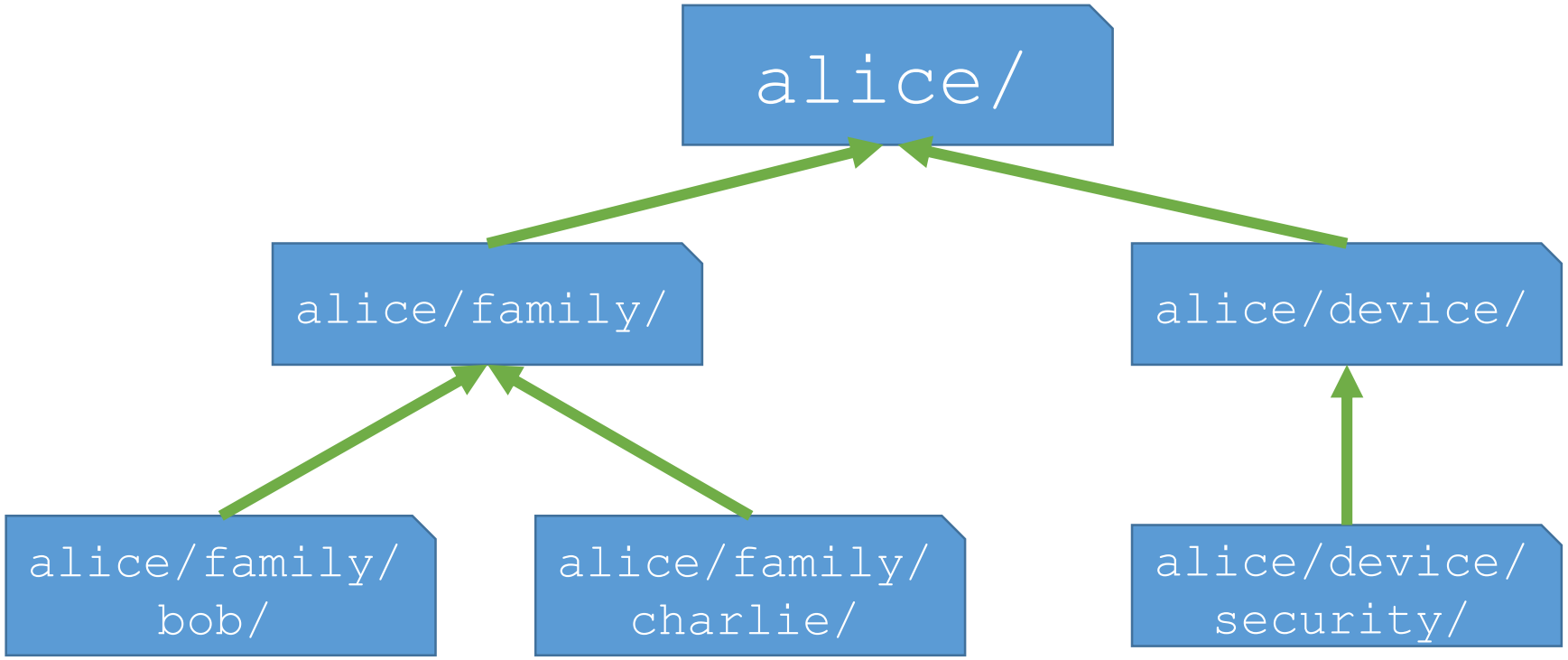
- **Lightweight:** privacy should be as simple as setting a flag in key-exchange (as opposed to a separate protocol – e.g., using secret handshakes [BDSSSW03])

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

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

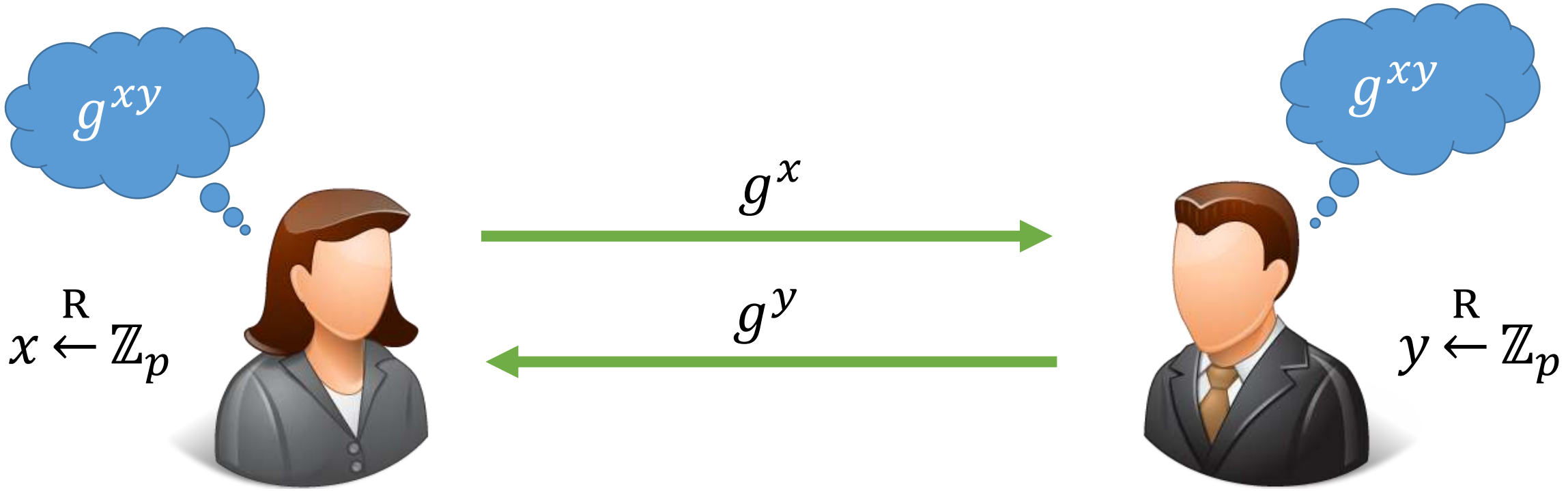


Identity and Authorization Model

Authorization decisions expressed as prefix patterns

Protocol Construction

Starting Point: Diffie-Hellman Key Exchange



$\mathbb{G}$: cyclic group of prime order p
with generator g

Shared key:
$\mathrm{KDF}(g^x, g^y, g^{xy})$

Starting Point: Diffie-Hellman Key Exchange

group of prime order p
with generator g

Shared key:
$\mathrm{KDF}(g^x, g^y, g^{xy})$

Secure Key Agreement: SIGMA-I Protocol [CK01]

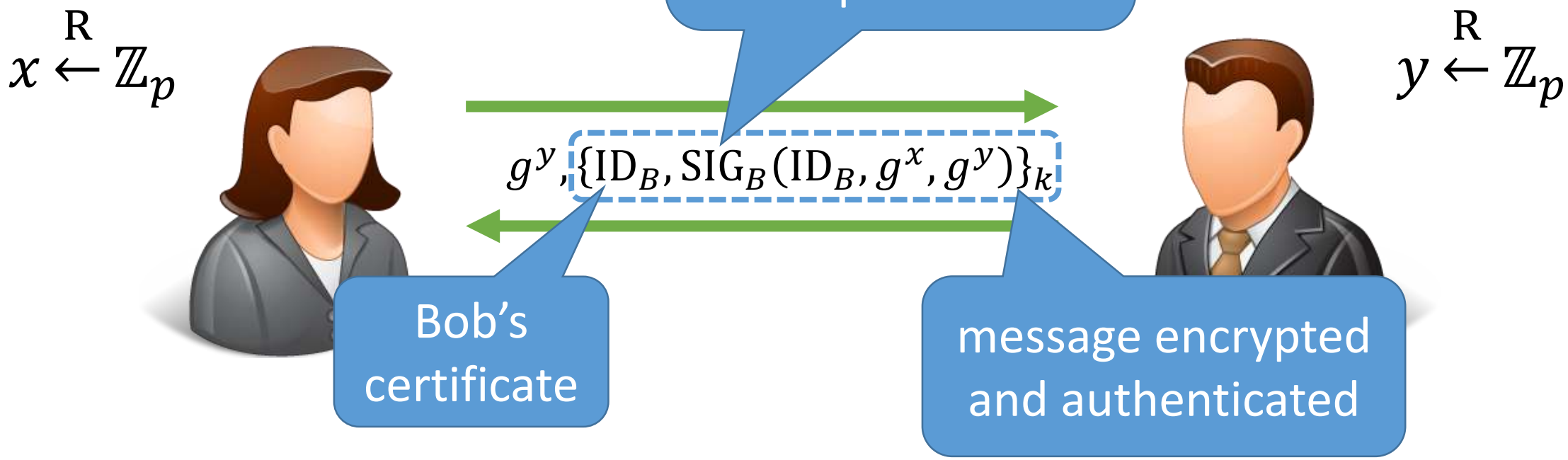
$x \xleftarrow{R} \mathbb{Z}_p$

g^x

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

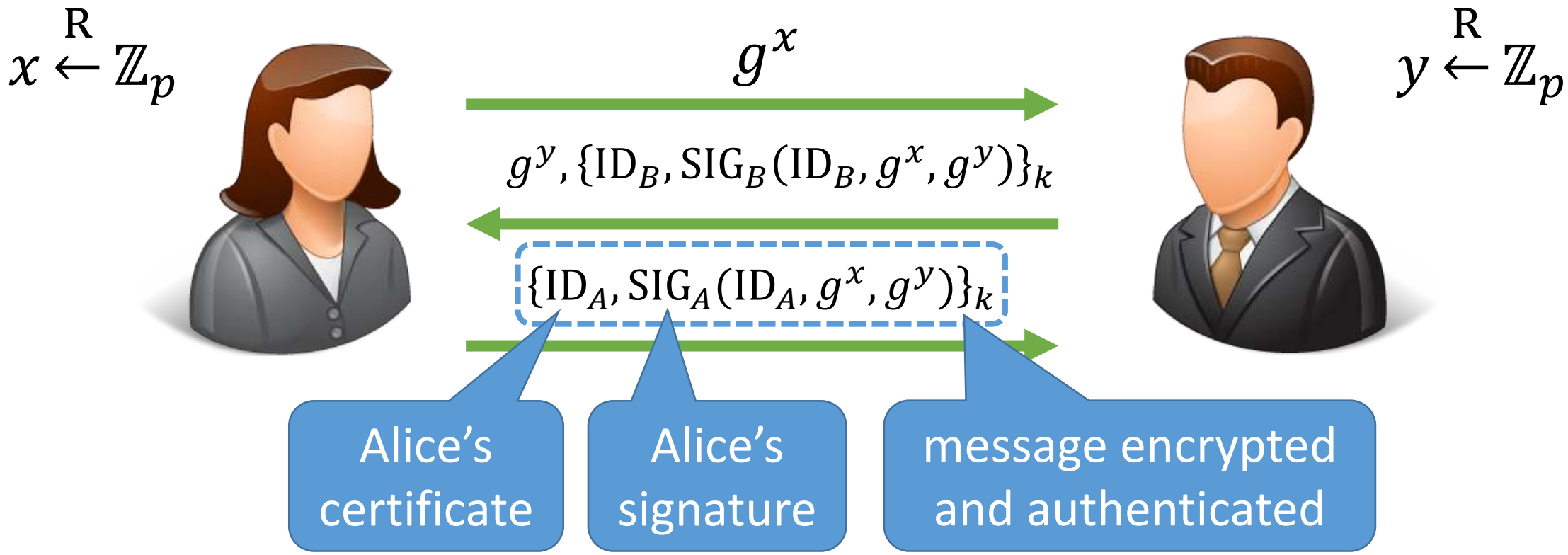
$y \xleftarrow{R} \mathbb{Z}_p$

Secure Key Agreement [CK01]



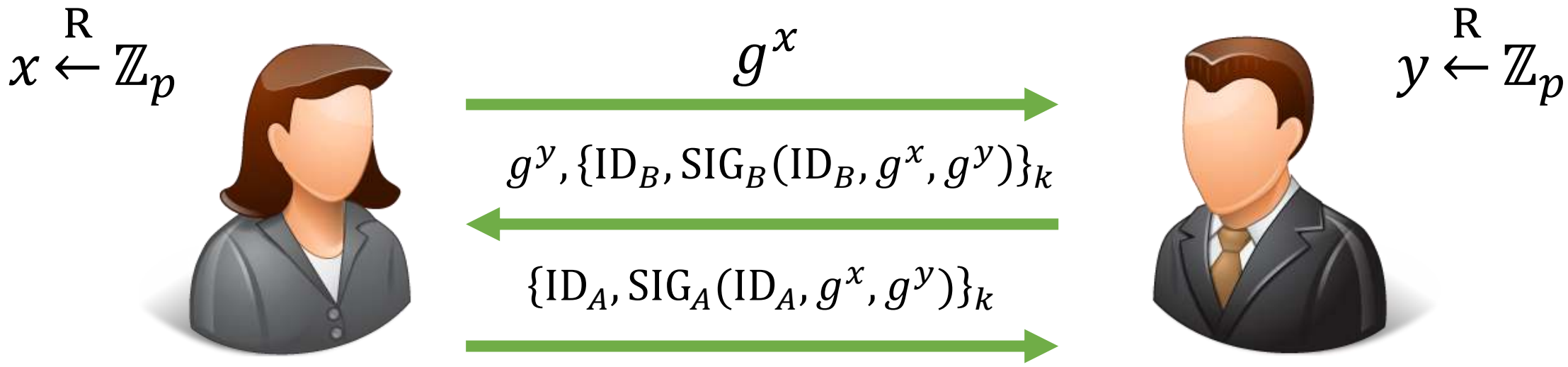
Note: in the actual protocol, session ids are also included for replay prevention.

Secure Key Agreement: SIGMA-I Protocol [CK01]



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

Secure Key Agreement: SIGMA-I Protocol [CK01]



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

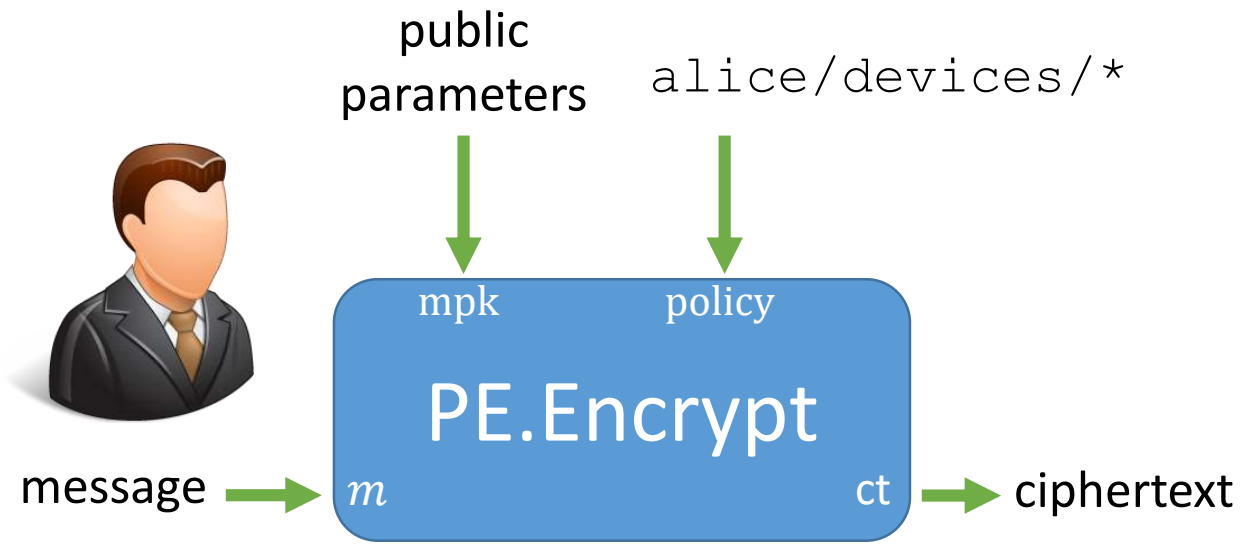
- Mutual authentication against active network adversaries
- Hides server's (Bob's) identity from a passive attacker
- Hides client's (Alice's) identity from an active attacker

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

Chicken-and-egg problem: neither party wants to "go first" in the key exchange.

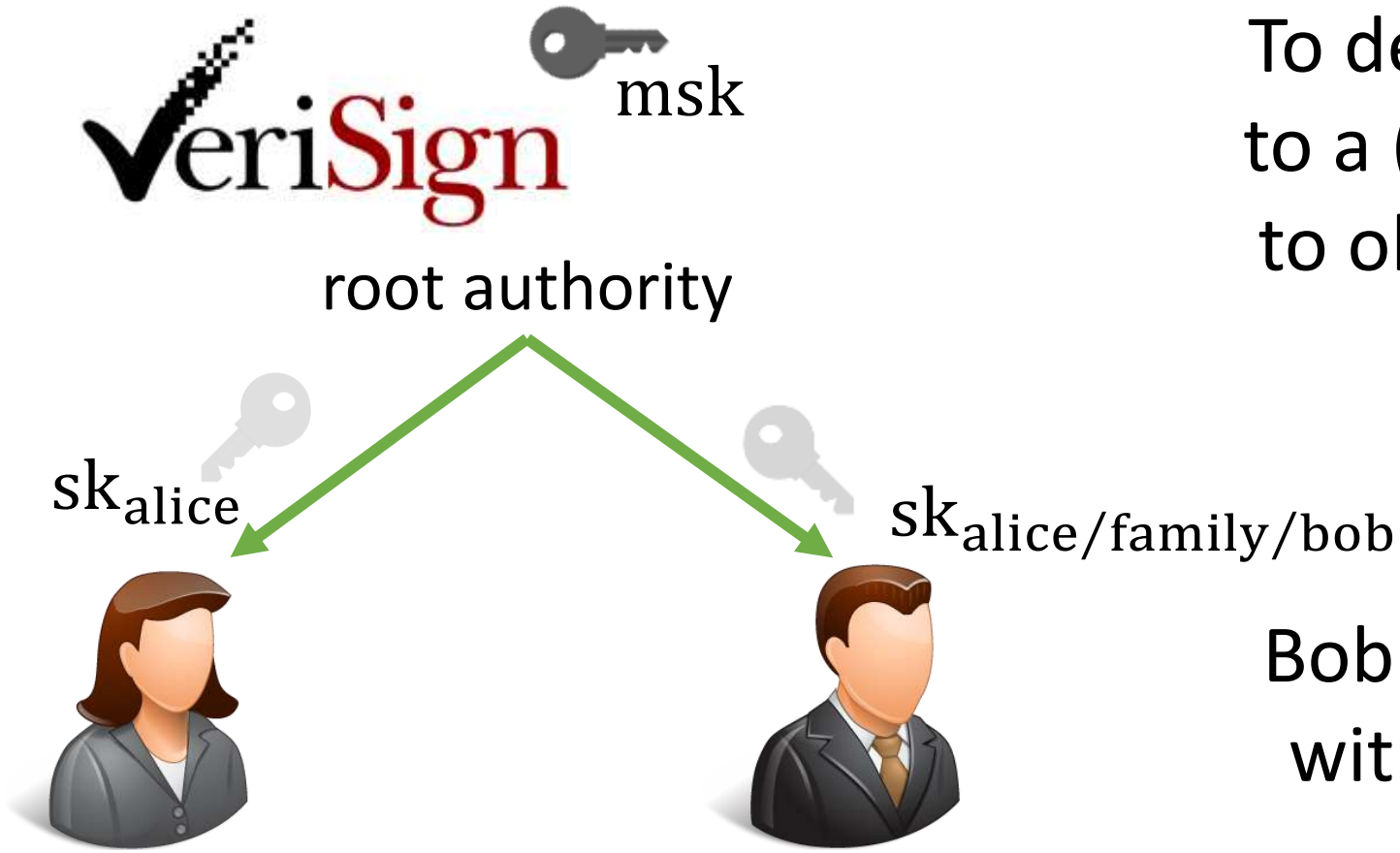
Prefix-Based Encryption

Public-key encryption scheme where ciphertexts are associated with a *policy*



Bob can encrypt a message with respect to a particular policy

Prefix-Based Encryption

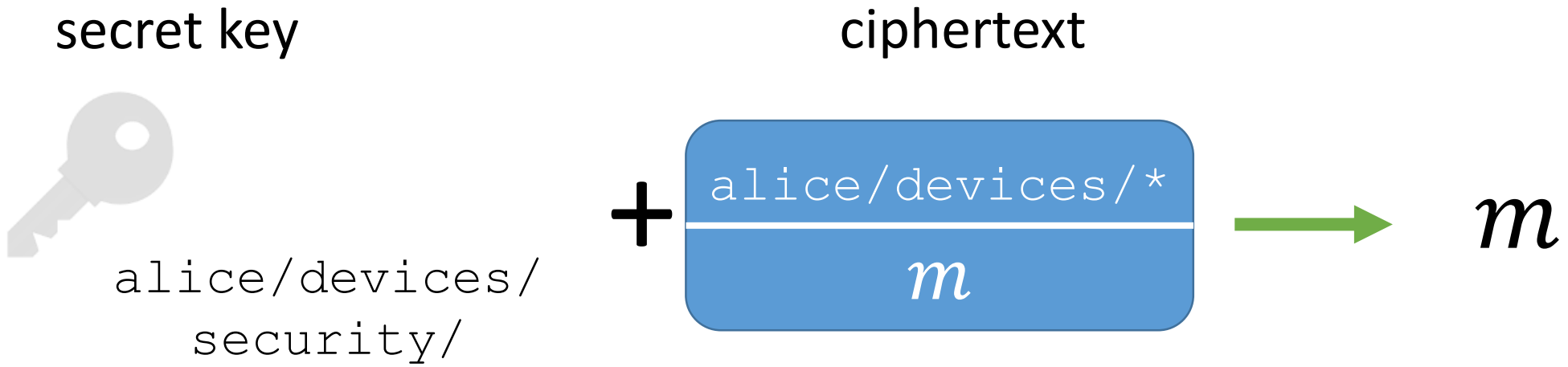


To decrypt messages, users go to a (trusted) identity provider to obtain a decryption key for their identity

Bob can decrypt all messages with policies satisfied by his identity

Prefix-Based Encryption

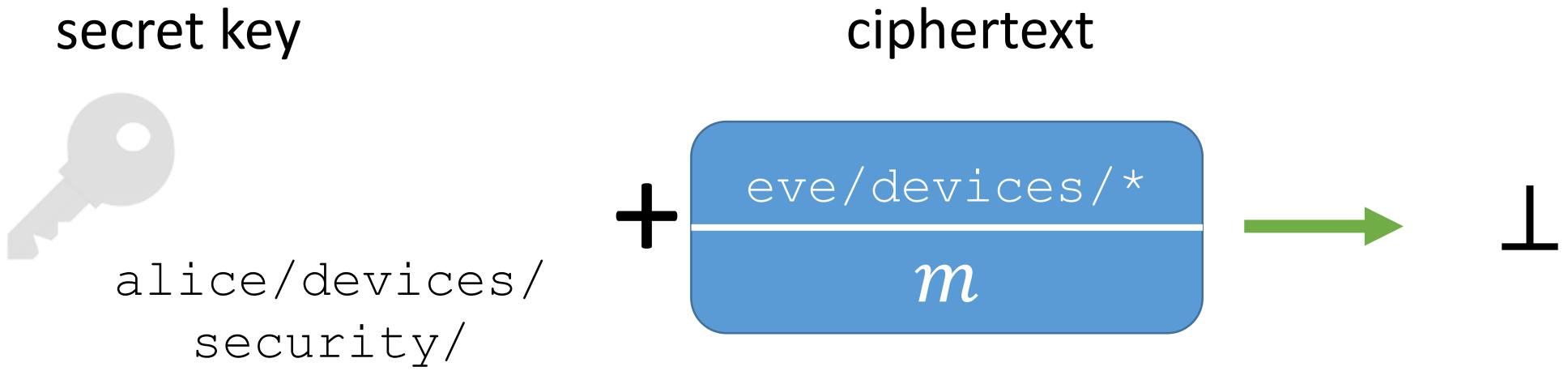
Ciphertexts associated with policies and keys associated with identities



Decryption succeeds if policy is satisfied

Prefix-Based Encryption

Ciphertexts associated with policies and keys associated with identities



Decryption fails if policy not satisfied

Prefix-Based Encryption

Can be leveraged for prefix-based policies

Bob encrypts his message to the policy `alice/devices/*`. Any user with an identity that begins with `alice/devices/` can decrypt.

Prefix-Based Encryption

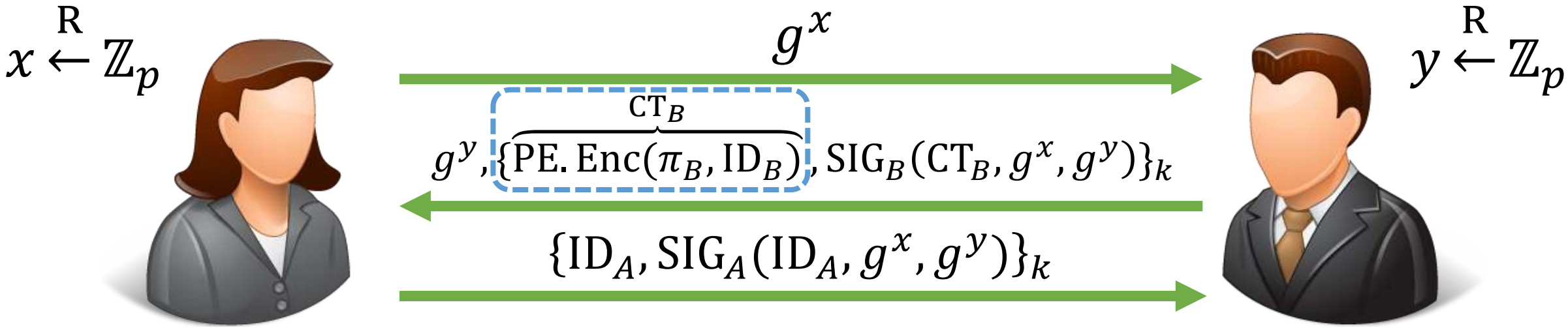
Can be leveraged for prefix-based policies

Bob enc… …e the policy a… …user with a… …with alice/de… …crypt.

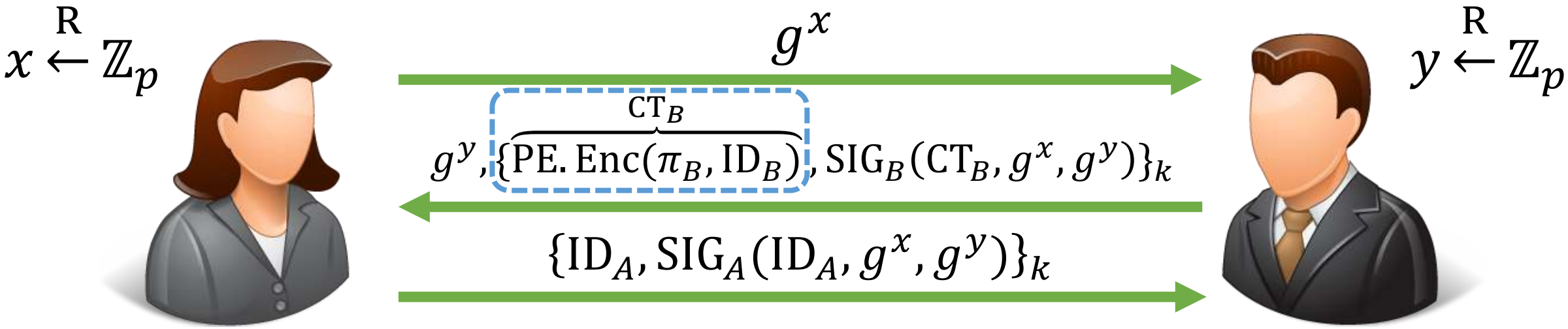
Can be built from identity-based encryption

Private Mutual Authentication

Key idea: encrypt certificate using prefix-based encryption



Private Mutual Authentication



- **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 Service Discovery

Prefix-based encryption can also be leveraged for *private* service discovery

See paper for details:
http://arxiv.org/abs/1604.06959

Implementation and Benchmarks

- 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

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

Thank you!

https://arxiv.org/abs/1604.06959