Privacy, Discovery, and Authentication for the Internet of Things

David Wu

Joint work with Ankur Taly, Asim Shankar, and Dan Boneh

The Internet of Things (IoT)

Lots of smart devices, but only useful if users can discover them!

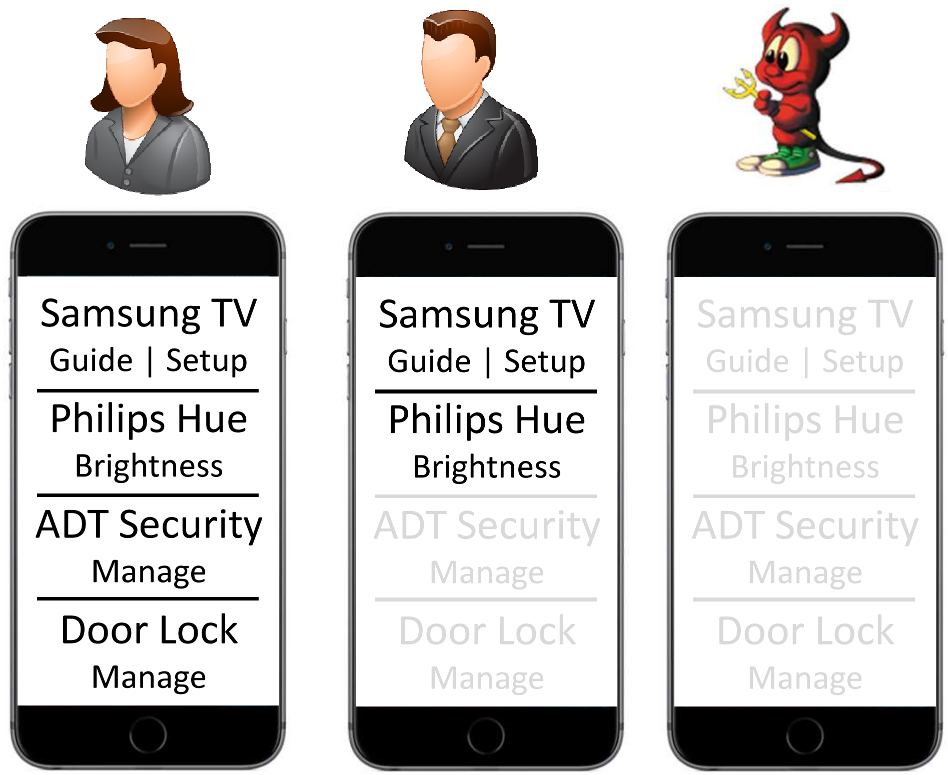
Private Service Discovery

- Many existing service discovery protocols: Multicast DNS (mDNS), Apple Bonjour, Bluetooth Low Energy (BLE)
- But… not much privacy
  - Recent study of mDNS announcements by Könings et al. [KBSW13] show that nearly 60% of devices revealed the device owner's name in the clear (across approximately 3000 devices on a university campus)

- Service advertisements are not authenticated: malicious devices can forge service broadcasts

Private Service Discovery

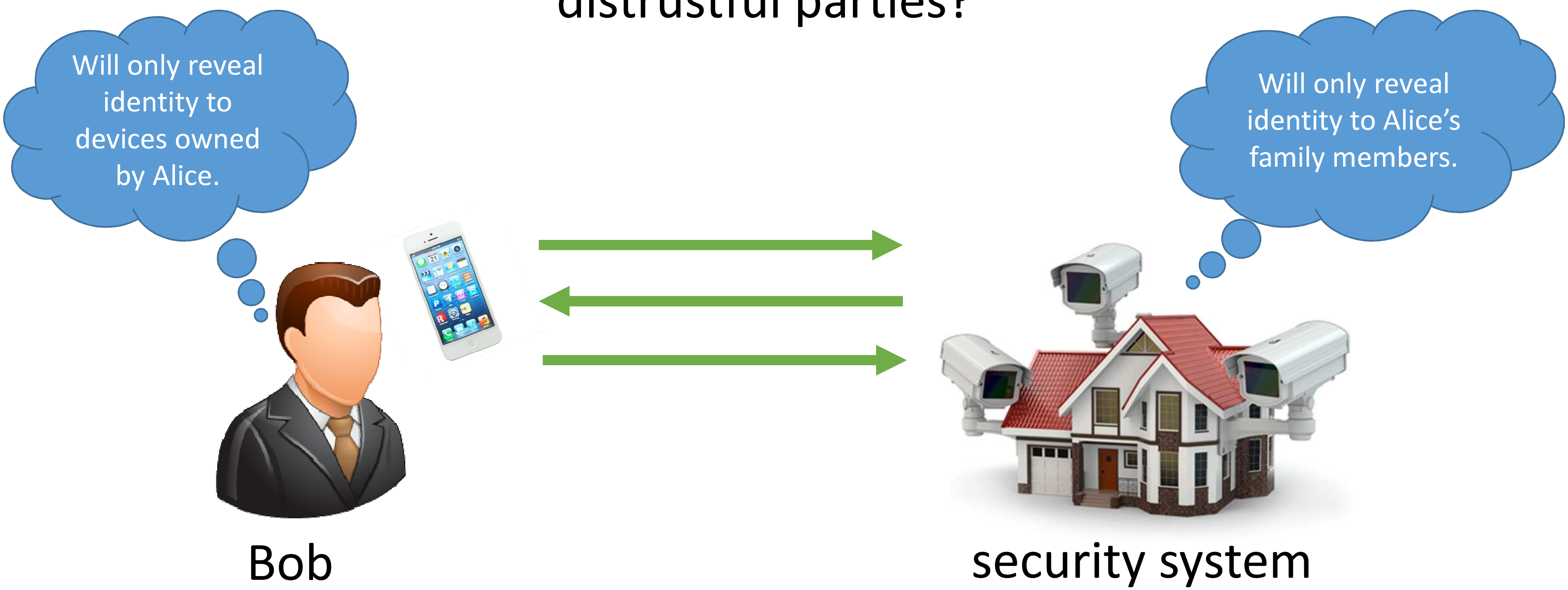
Each service specifies an authorization policy



Alice Guest Stranger

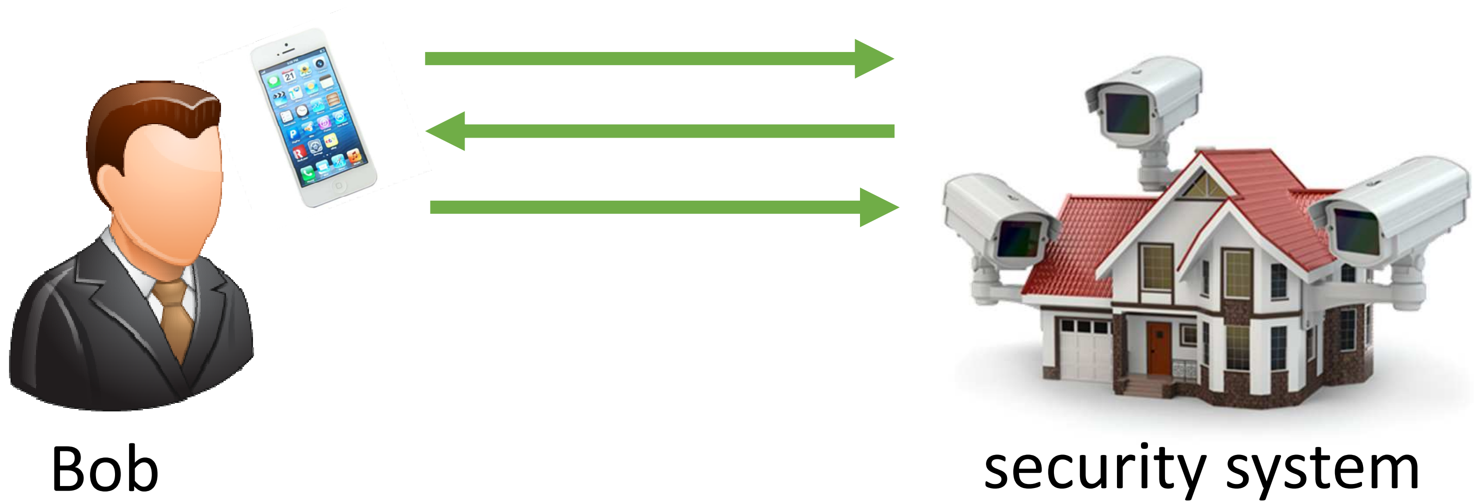
Private Mutual Authentication

How to authenticate between mutually distrustful parties?



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

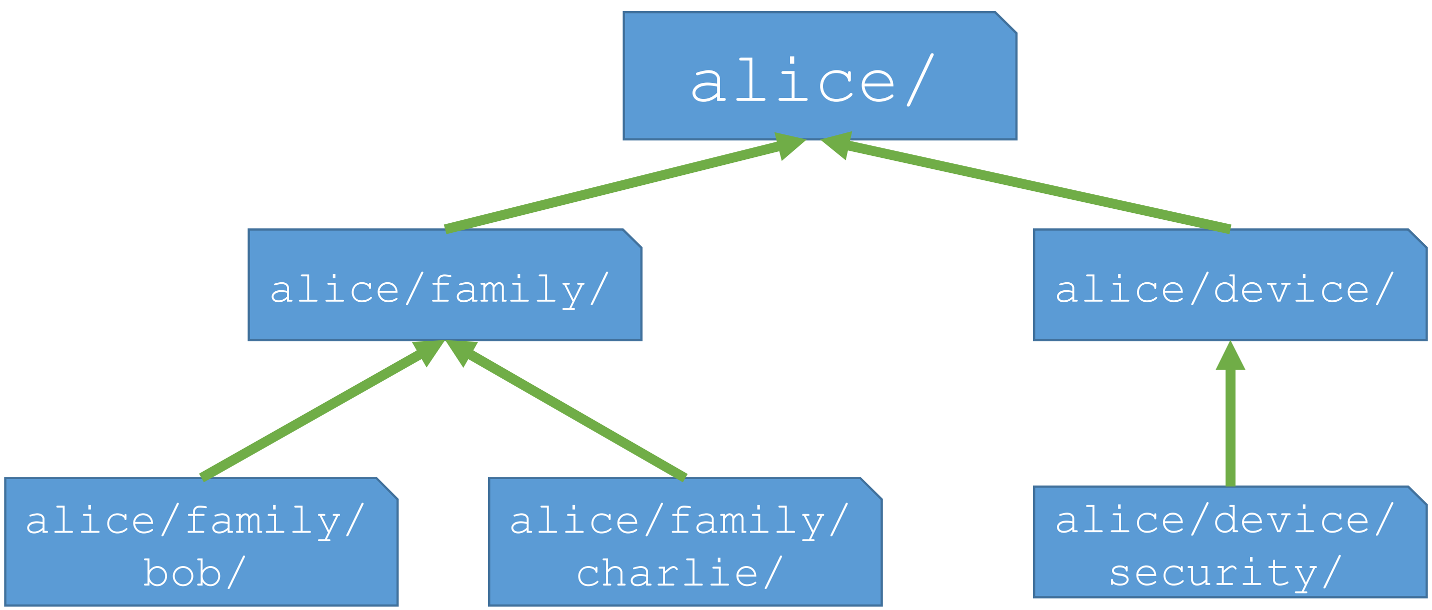
- **Authentic advertisements:** Service advertisements (for discovery) should be unforgeable and authentic

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

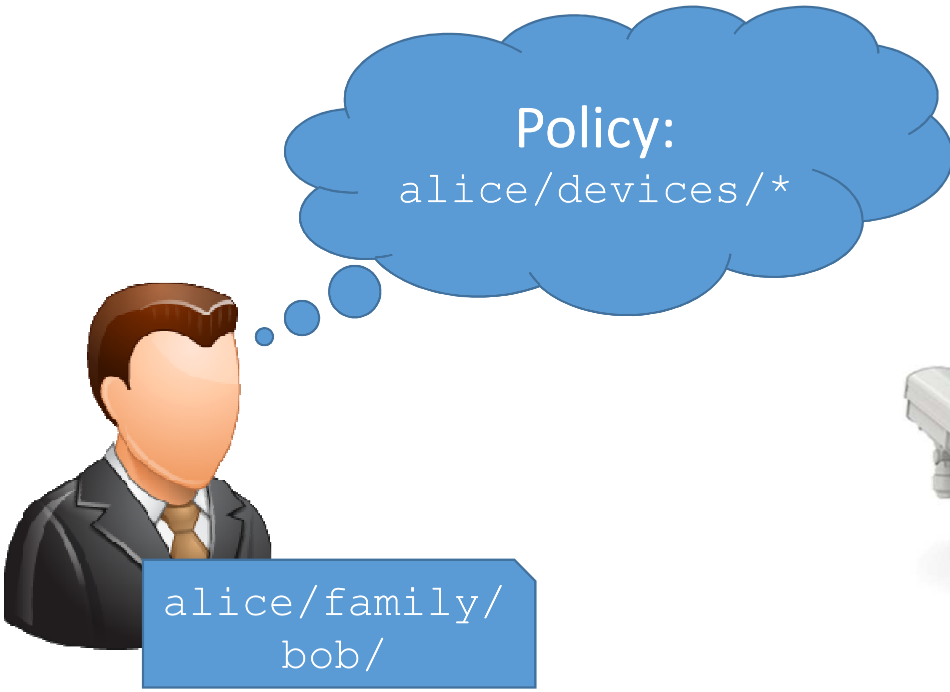
Identity and Authorization Model

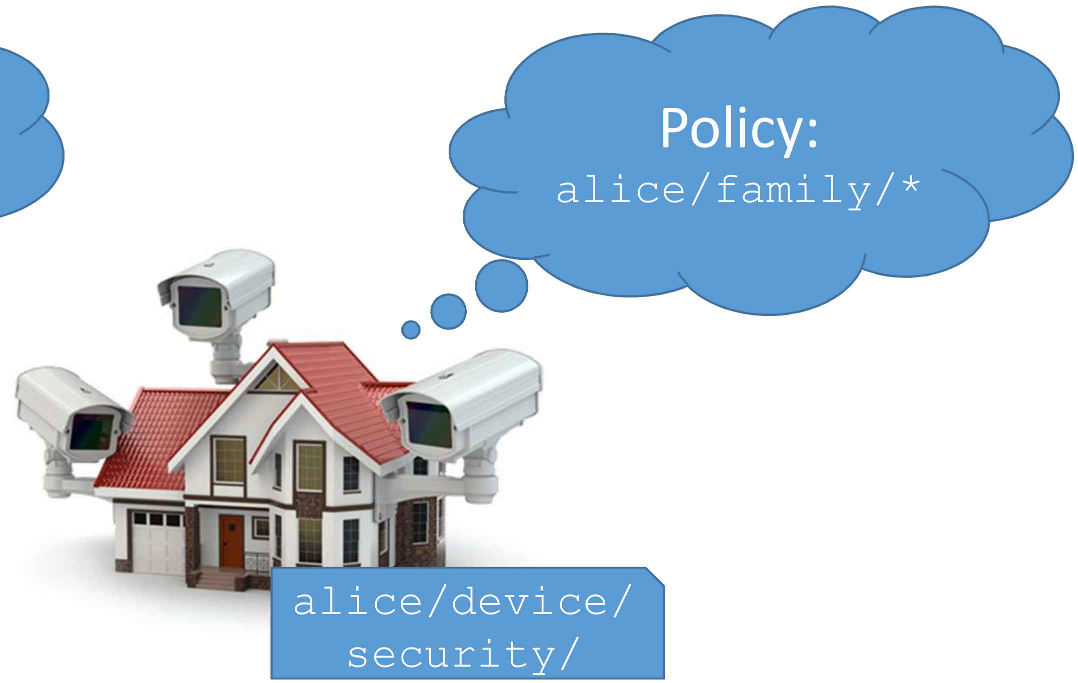
Every party has a signing + verification key, and a collection of human-readable names bound to their public keys via a certificate chain



Identity and Authorization Model

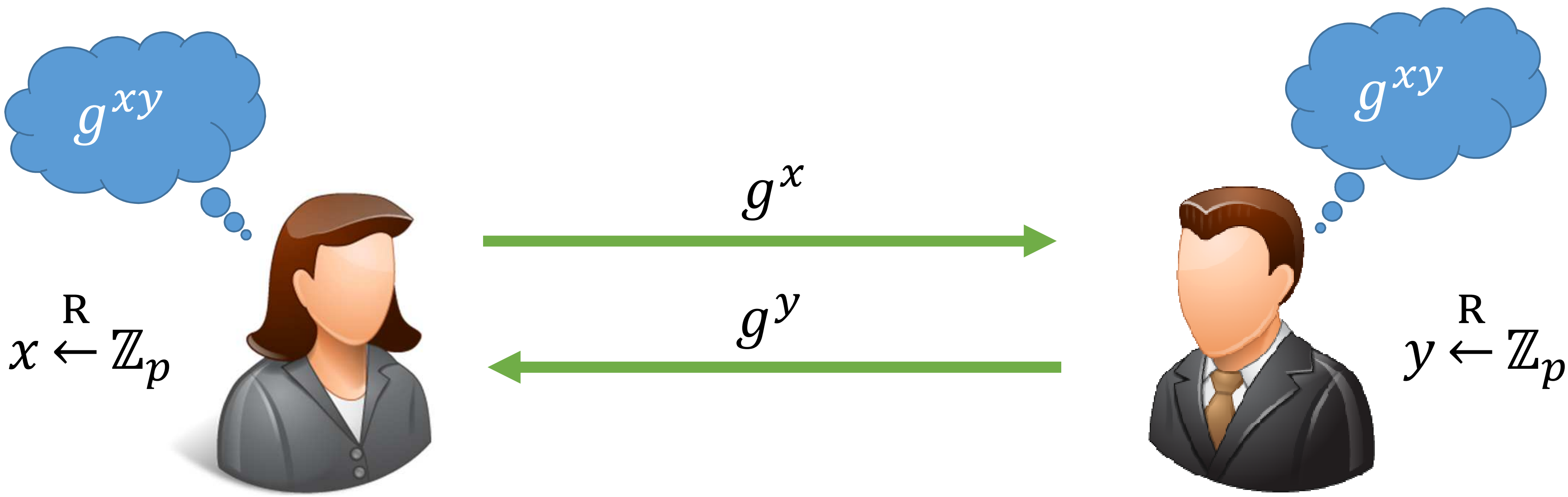
Authorization decisions expressed as prefix patterns





Protocol Construction

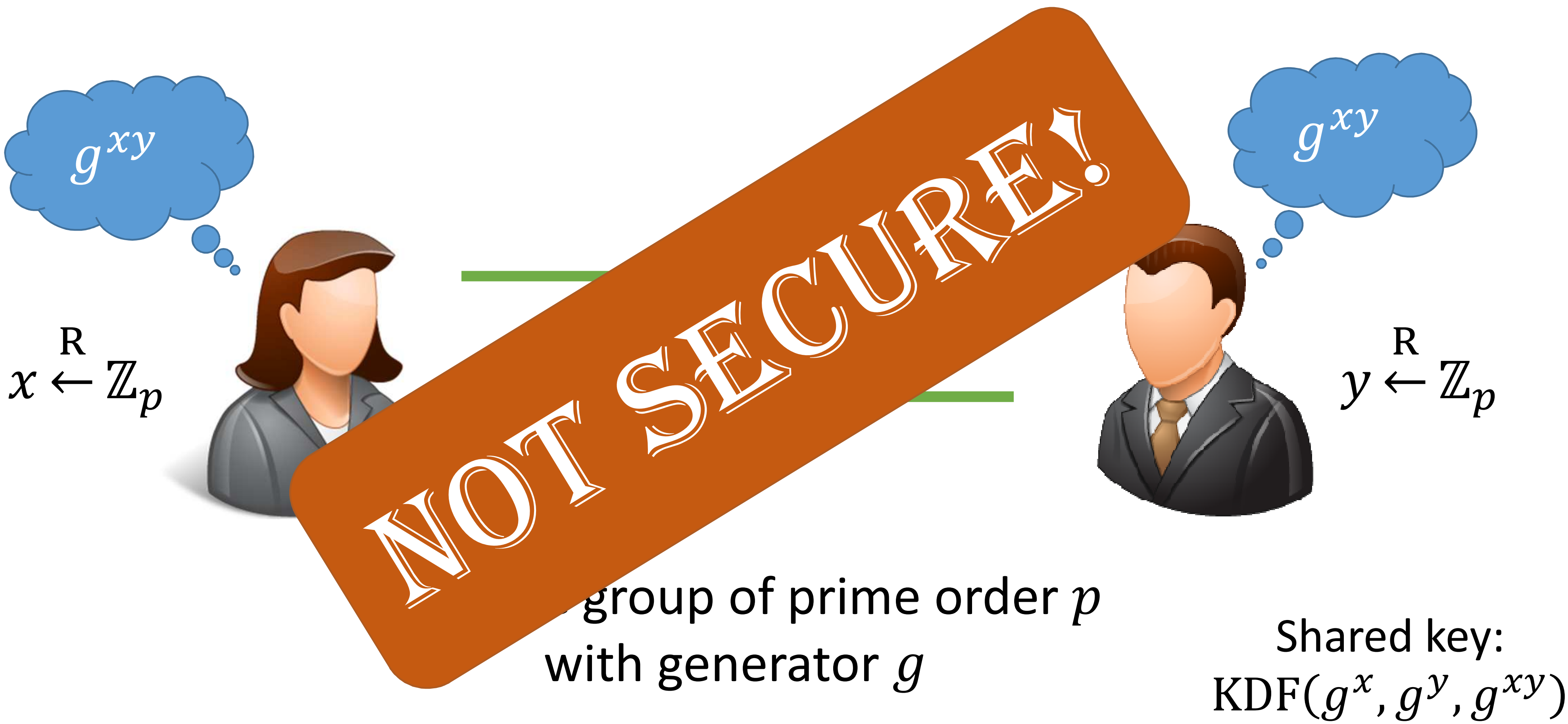
Starting Point: Diffie-Hellman Key Exchange



$\mathbb{G}$: 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



Secure Key Agreement: SIGMA-I Protocol [CK01]

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

g^x

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

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

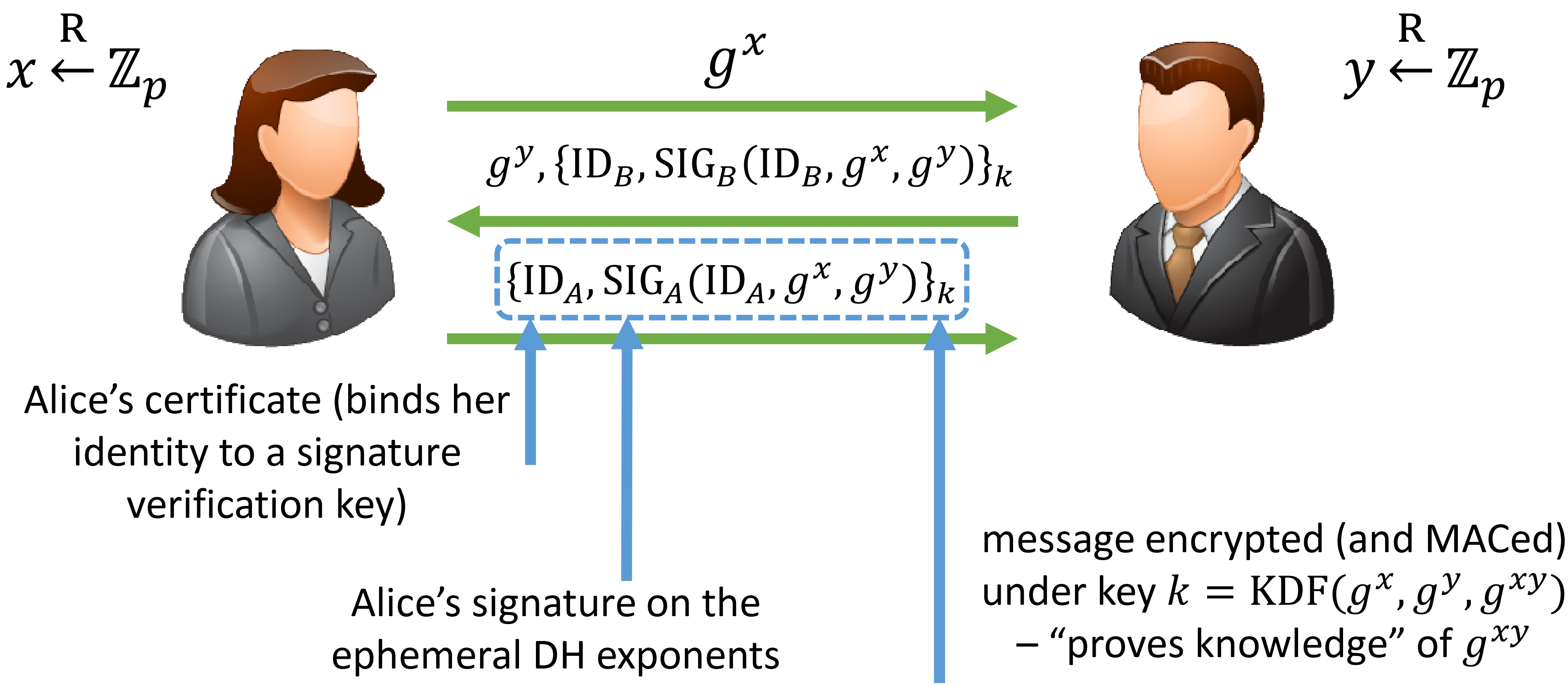
Bob’s certificate (binds his identity to a signature verification key)

Bob’s signature on the ephemeral DH exponents

message encrypted (and MACed) under key $k = \mathrm{KDF}(g^x, g^y, g^{xy})$ – “proves knowledge” of g^{xy}

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

Secure Key Agreement: SIGMA-I Protocol [CK01]



Secure Key Agreement: SIGMA-I Protocol [CK01]

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

g^x

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

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

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

session key derived from

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

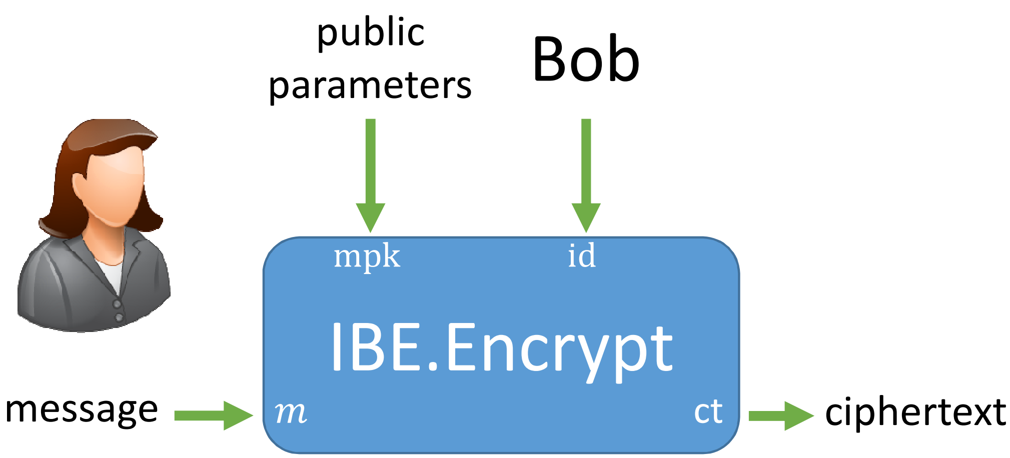
Properties of the SIGMA-I Protocol

- 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!

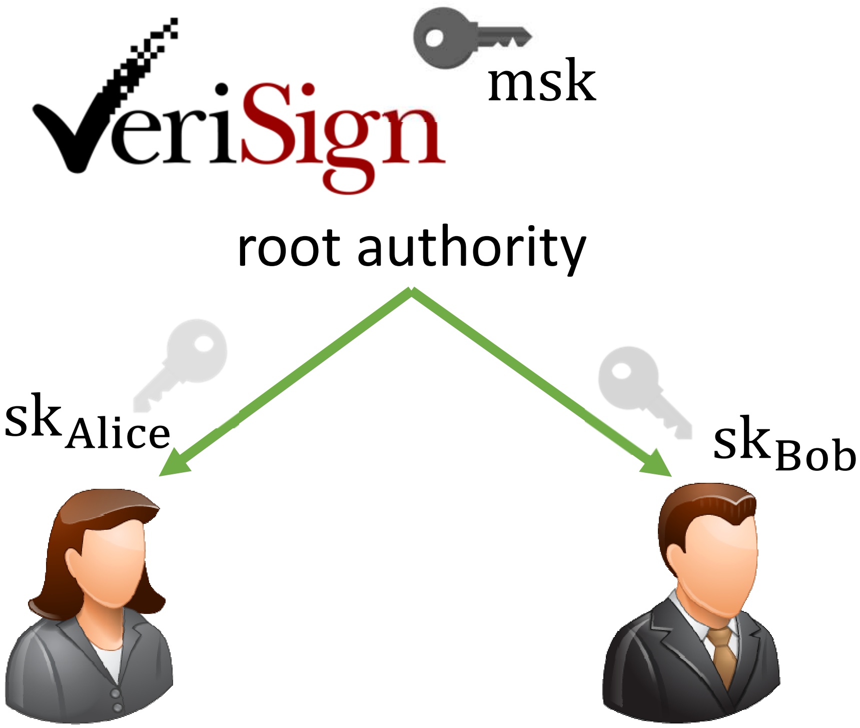
Identity Based Encryption (IBE) [Sha84, BF01, Coc01]

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



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

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

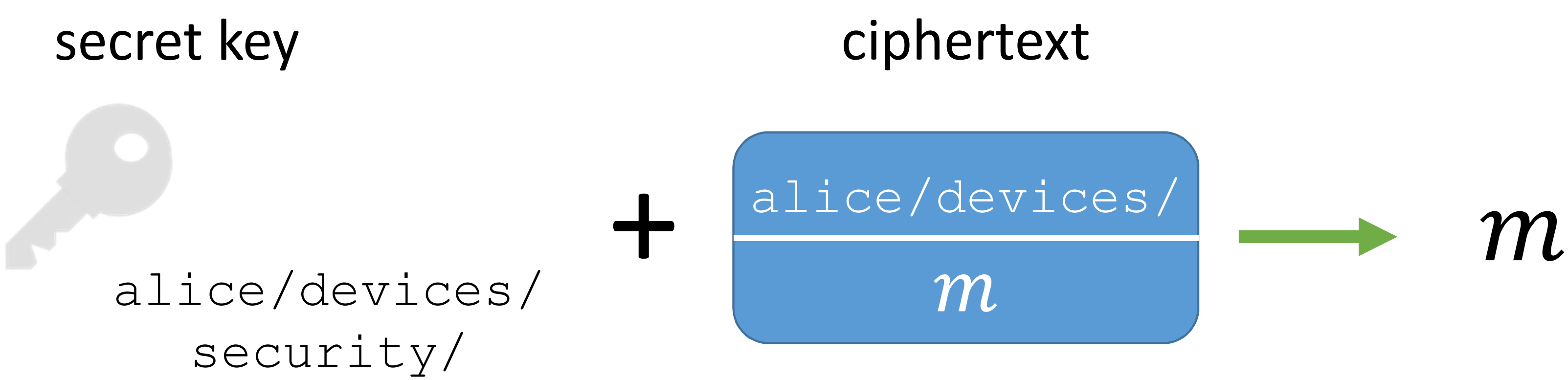


To decrypt messages, users go to a (trusted) identity provider to obtain a decryption key for their identity

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

Prefix-Based Encryption

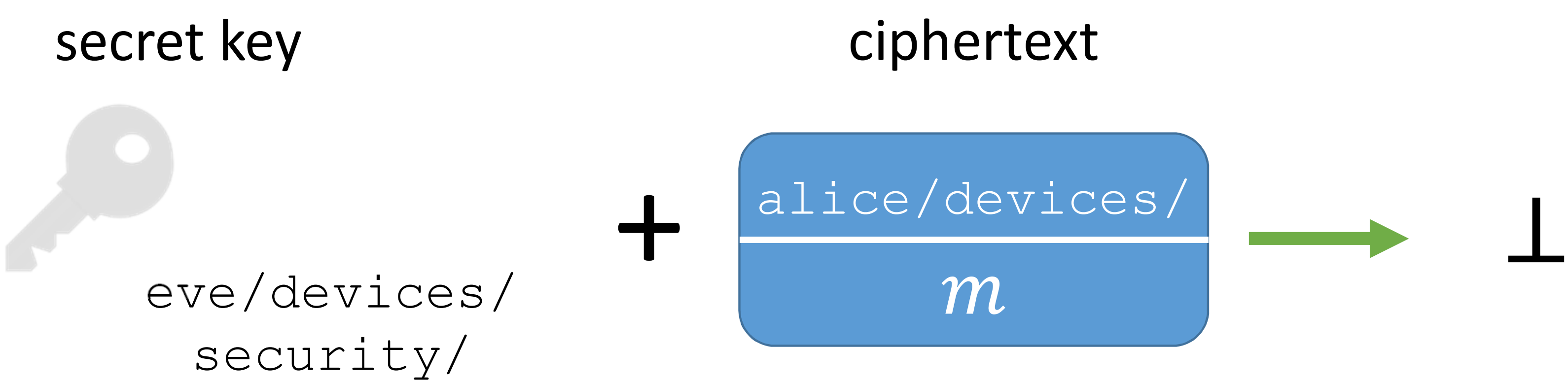
Secret-keys and ciphertexts both associated with names



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



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

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]

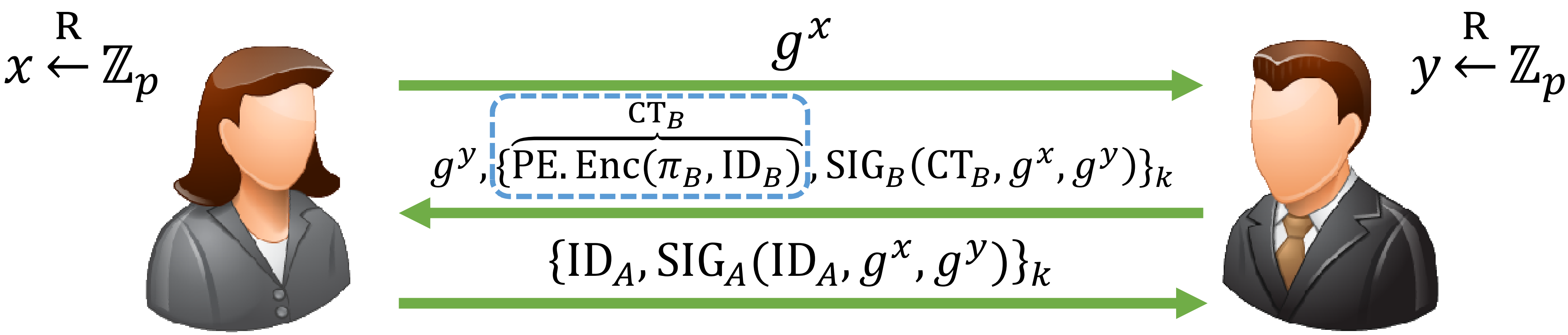
Encryption is just IBE encryption

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

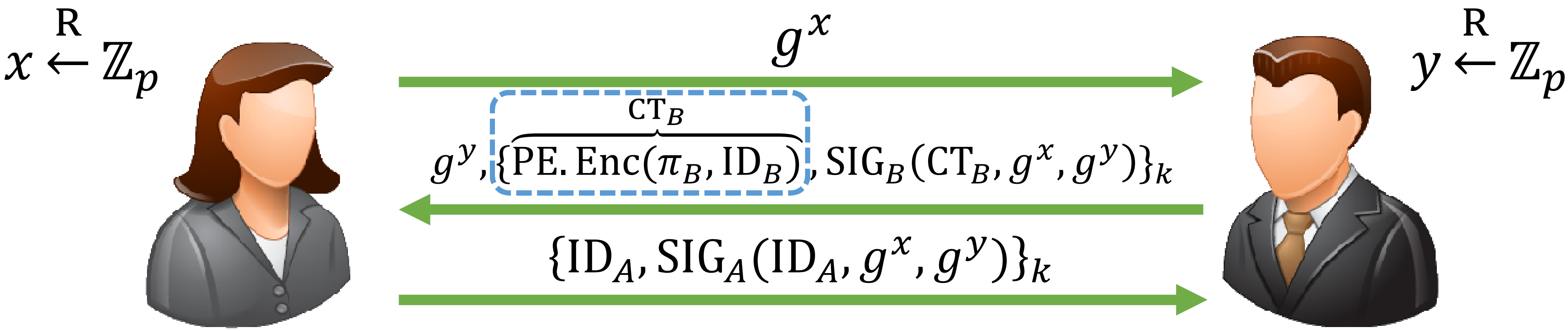
can decrypt encryptions to all prefixes of `alice/devices/security`

Private Mutual Authentication

Key idea: encrypt certificate using prefix-based encryption

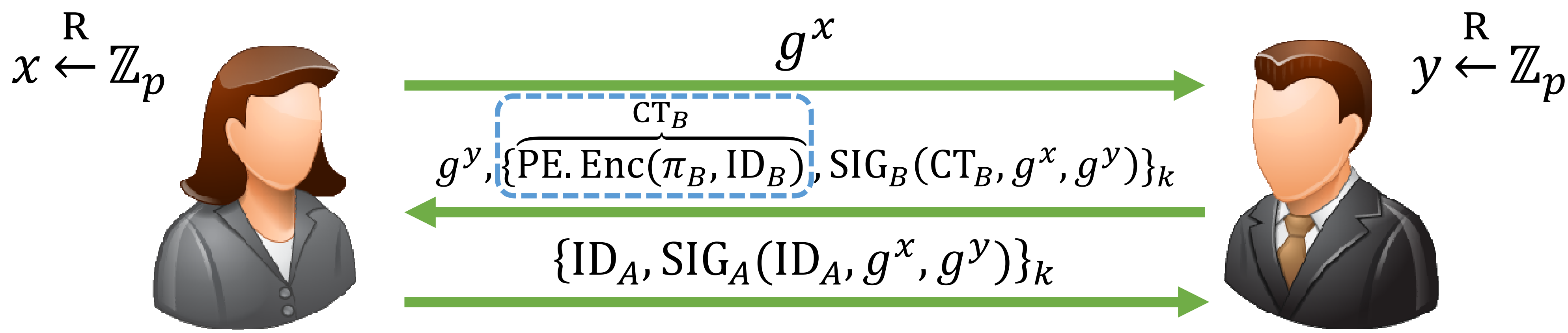


Private Mutual Authentication



- **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

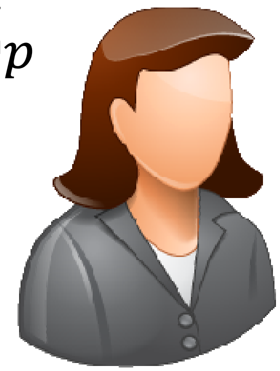
Private Mutual Authentication



- **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

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



g^x

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

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

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

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

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

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

authorization policy

service identity

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

signature for authenticity

Private Service Discovery: Mutual Authentication

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

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

ephemeral DH exponent

sender and recipient's identities

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

Private Service Discovery: Mutual Authentication

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

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

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

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

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

Private Service Discovery: Mutual Authentication

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

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

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

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

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

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

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

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

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

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

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

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
- 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

	Desktop	Nexus 5X	Raspberry Pi 2
SIGMA-I	7 ms	50 ms	87 ms
Private Mutual Auth.	13 ms	291 ms	326 ms
Slowdown	1.9x	5.8x	3.7x

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

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

- 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

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

16 bytes | 208 bytes | 500 bytes | 32 bytes | 64 bytes

Processing advertisement requires 1 IBE decryption and 1 ECDSA verification:

$267\ \mathrm{ms} + 11\ \mathrm{ms} = 278\ \mathrm{ms}$ 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?