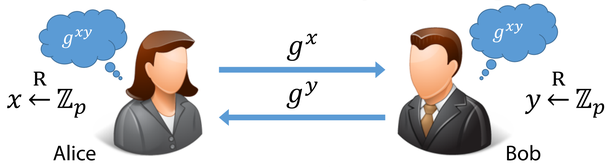
Privacy, Discovery, and Authentication for the Internet of Things

David J. Wu, Ankur Taly, Asim Shankar, and Dan Boneh

Privacy and the Internet of Things

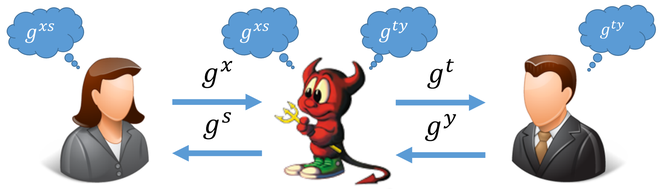
The ability to automatically discover services and seamlessly connect to them is essential to realizing the full potential of the Internet of Things (IoT). While discovery protocols like Multicast DNS, Apple AirDrop, and Bluetooth Low Energy have gained widespread adoption among IoT devices, most of these protocols do not offer any form of privacy control for the service, and often leak sensitive information such as service type, device hostname, device owner's identity, and more in the clear. Similarly, most existing mutual authentication protocols such as TLS 1.2, SIGMA, and JFK do not provide strong privacy guarantees (if any at all).

Diffie-Hellman Key Exchange



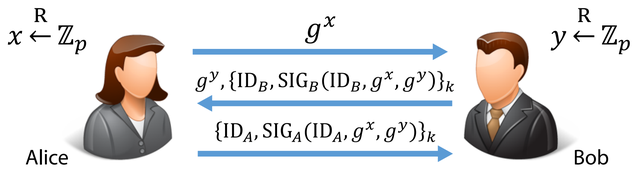
$\mathbb{G}$: cyclic group of prime order p with generator g

Vulnerable to an active (man-in-the-middle) attacker:



Anonymous key-exchange is not secure!

The SIGMA Protocol



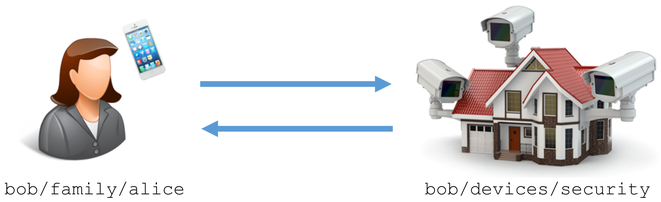
ID_A ID_B	Alice and Bob's certificates (binds identities to signature verification keys)
SIG_A SIG_B	signatures under Alice and Bob's signing keys
k	key used to encrypt + authenticate handshake messages (derived from Diffie-Hellman secret)

Negotiated session key also derived from Diffie-Hellman secret

Key limitation: one of the parties must reveal their identity first *without* knowing the identity of their peer.

The Need for Privacy

Suppose Bob has a network-configurable home-security system.



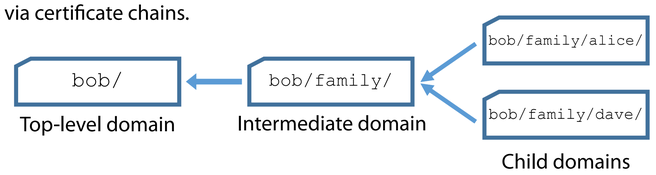
Privacy and functionality requirements:

- Family members (e.g., users with IDs of the form `bob/family/*`) should be able to connect to and configure the security system. However, members outside this group should not be able to do so.
- Security system does not want to reveal its identity (e.g., device type, firmware version, and other identifying information) to non-authorized entities (e.g., a potential burglar).
- Alice does not want to reveal her identity (e.g., her name) unless she is sure she is communicating with her security system and not an impostor device.

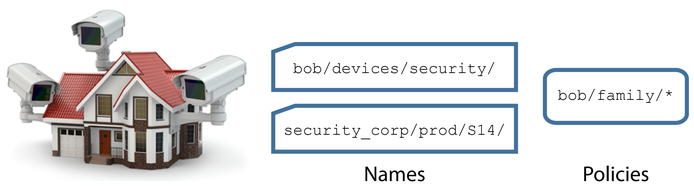
Similar considerations for private service discovery: the security system does not want to broadcast its identity / existence to all nodes on the network, but rather, only wants to reveal its identity to nodes that it trusts.

Framework and Desired Features

Each party is associated with one or more hierarchically-structured names (e.g., `bob/family/alice` or `bob/devices/security`). Names are bound to identities via certificate chains.



Each party has a collection of names and authorization policies:

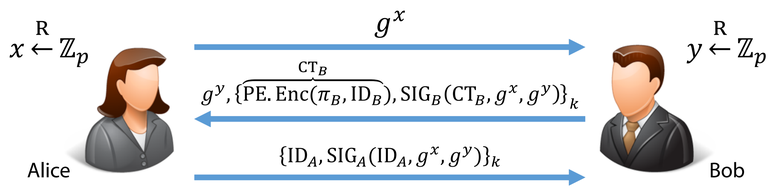


Desired protocol properties:

- **Mutual privacy:** The protocol must ensure that a party's identity is only ever revealed to an authorized recipient. In the service discovery setting, this applies to both the service being advertised and the client trying to discover it.
- **Authentic advertisements:** In the service discovery setting, the service advertisements should be authentic and unforgeable. Otherwise, an adversary can forge an advertisement to determine if a client is interested in it.

Private Mutual Authentication

Key idea: The party that "goes first" encrypts their identity to the prefix corresponding to their authorization policy.



π_B: Bob's authorization policy

- **Privacy for Alice's identity:** Alice sends her identity only after verifying Bob's identity satisfies her policy.
- **Privacy for Bob's identity:** By security of the prefix encryption scheme, only users that satisfy Bob's policy are able to decrypt and learn his identity.
- **Protocol overhead (compared to SIGMA):** Alice needs to perform an extra decryption operation after she receives Bob's message. Bob needs to perform prefix encryption on each handshake. However, note that Bob can cache his encrypted identity; this way, the cost of the prefix encryption is amortized over multiple handshakes.

Private Service Discovery

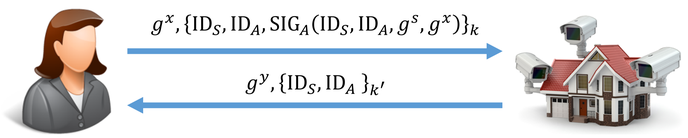
Two main components: service broadcast (announcement) and mutual authentication (for clients to connect to service). Our protocol is inspired by the design of TLS 1.3.

Service broadcast: a prefix encryption of the service details along with a semi-static Diffie-Hellman share (to enable 0-RTT mutual authentication)

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

π_S: service's authorization policy
g^s: semi-static Diffie-Hellman share

0-RTT mutual authentication: application data can be sent on first flow using key derived from the service's semi-static key. Service subsequently replies with an ephemeral key to ensure perfect forward secrecy of subsequent messages.

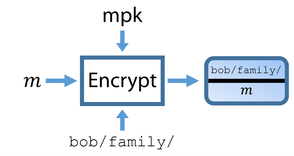


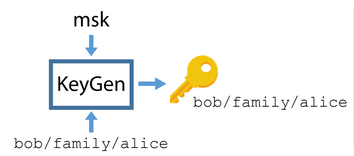
k	keys used to encrypt and authenticated handshake messages (derived from server's semi-static secret and client's ephemeral secret)
k'	

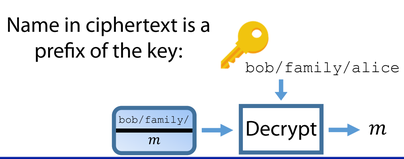
Resulting protocol provides privacy (via prefix encryption) and authenticity (via the service's signature on its broadcast message).

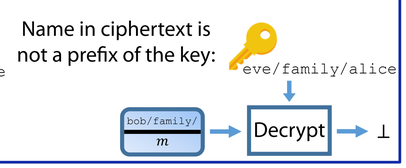
Prefix Encryption (PE)

Secret keys and ciphertexts are associated with names. Decryption succeeds if the name associated with the ciphertext is a *prefix* of the name associated with the secret key. This is easily constructed from any identity-based encryption scheme.









Experiments and Conclusion

We implemented both the private mutual authentication and the private discovery protocol as part of the Vanadium distributed computing framework. We then benchmarked our protocol on several architectures.

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

In our private service discovery protocol, a typical service advertisement is approximately **820 bytes**. Multicast DNS (mDNS) broadcasts can be up to 1300 bytes, so we can use mDNS to broadcast the service announcements.

Automatic service discovery and mutual authentication are integral to the development of the Internet of Things. However, most existing protocols do not provide much in terms of privacy. In this work, we introduce two simple and lightweight protocols for private mutual authentication and service discovery. Our protocols require minimal modification to and only incur a small amount of overhead on top of existing key-exchange protocols, thus making them suitable for a wide range of IoT deployment scenarios.