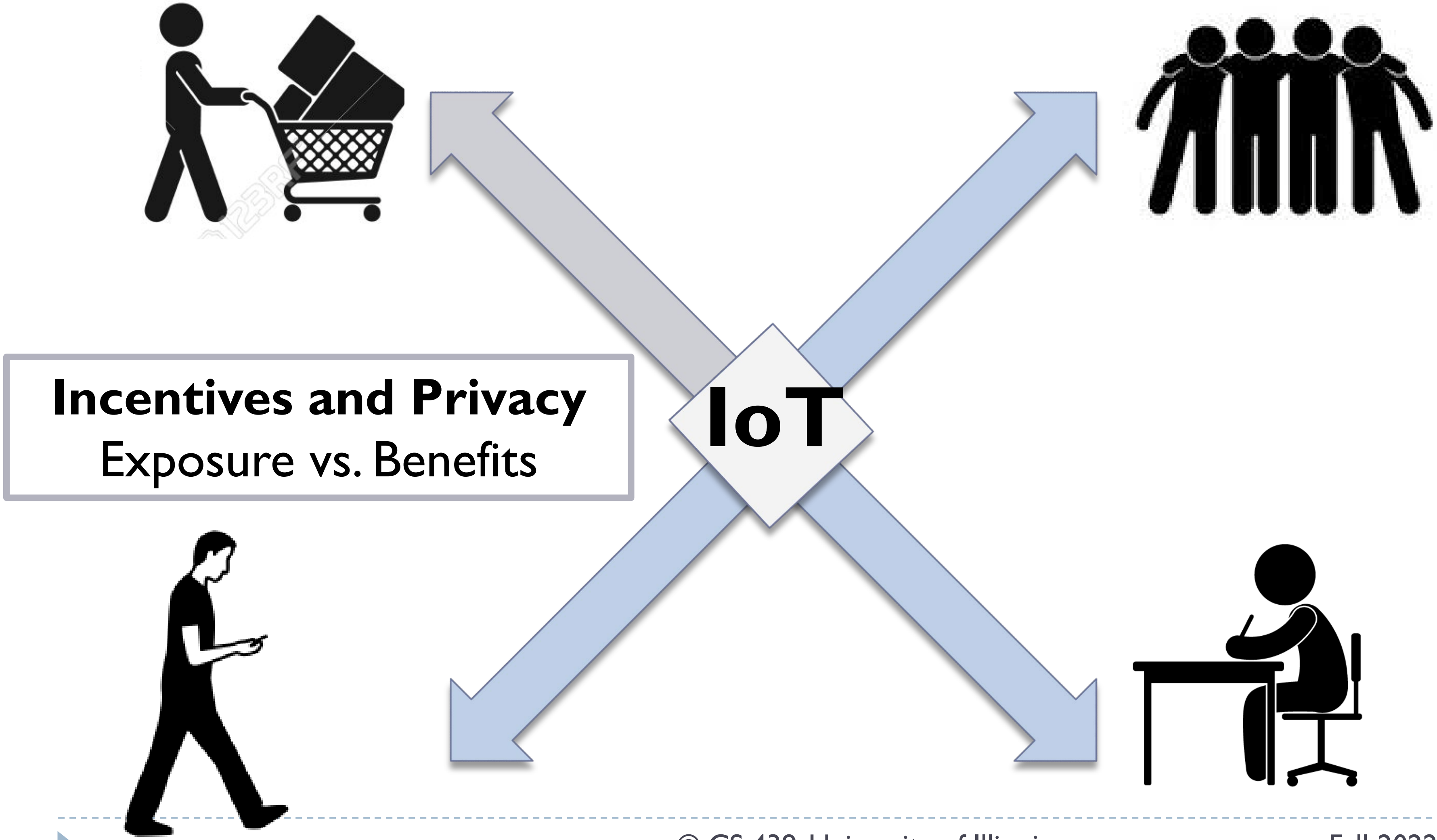
CS 439: Wireless Networking

IoT in Public Spaces

Fall 2022

Bridging the Gap with IoT



IoT

Incentives and Privacy

Exposure vs. Benefits

Designing an IoT Ecosystem

Context Discovery

“What environment am I in?”

The CoffeeShop

Designing an IoT Ecosystem

Context Discovery

User Identity

“What identity should I expose?”

FrenchRoast99

Designing an IoT Ecosystem

Context Discovery

User Identity

User Identity Use and Reuse

FrenchRoast99

Location-based pseudonyms

Espresso42

Designing an IoT Ecosystem

Context Discovery

User Identity

User Identity Use and Reuse

Sharing/Querying

The CoffeeShop

Enable social networking based recommendations

A friend

Designing an IoT Ecosystem

Context Discovery

User Identity

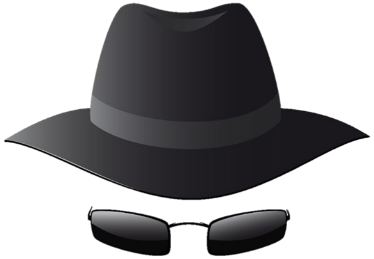
User Identity Use and Reuse

Sharing/Querying

Secure Validation and Privacy

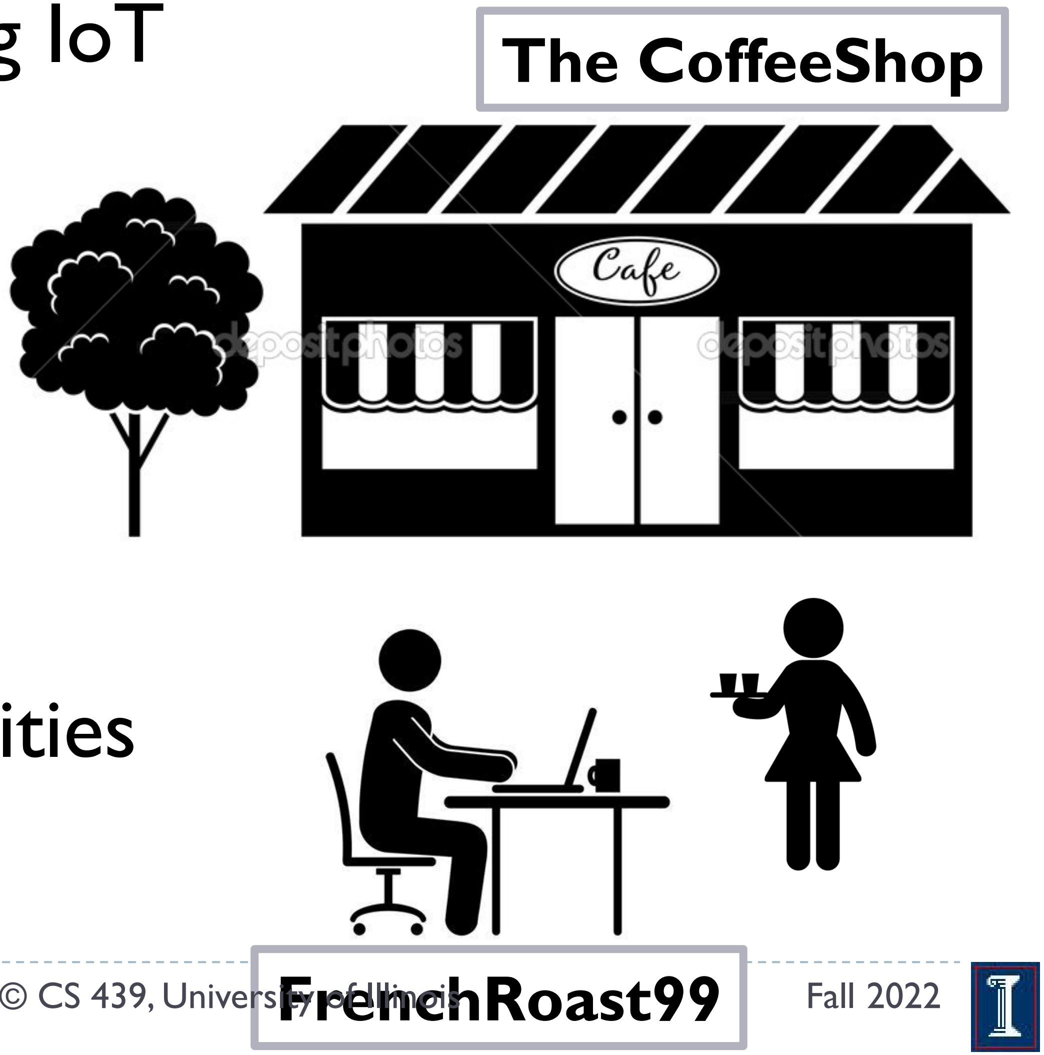
The CoffeeShop

Prevent impersonation and unauthorized access



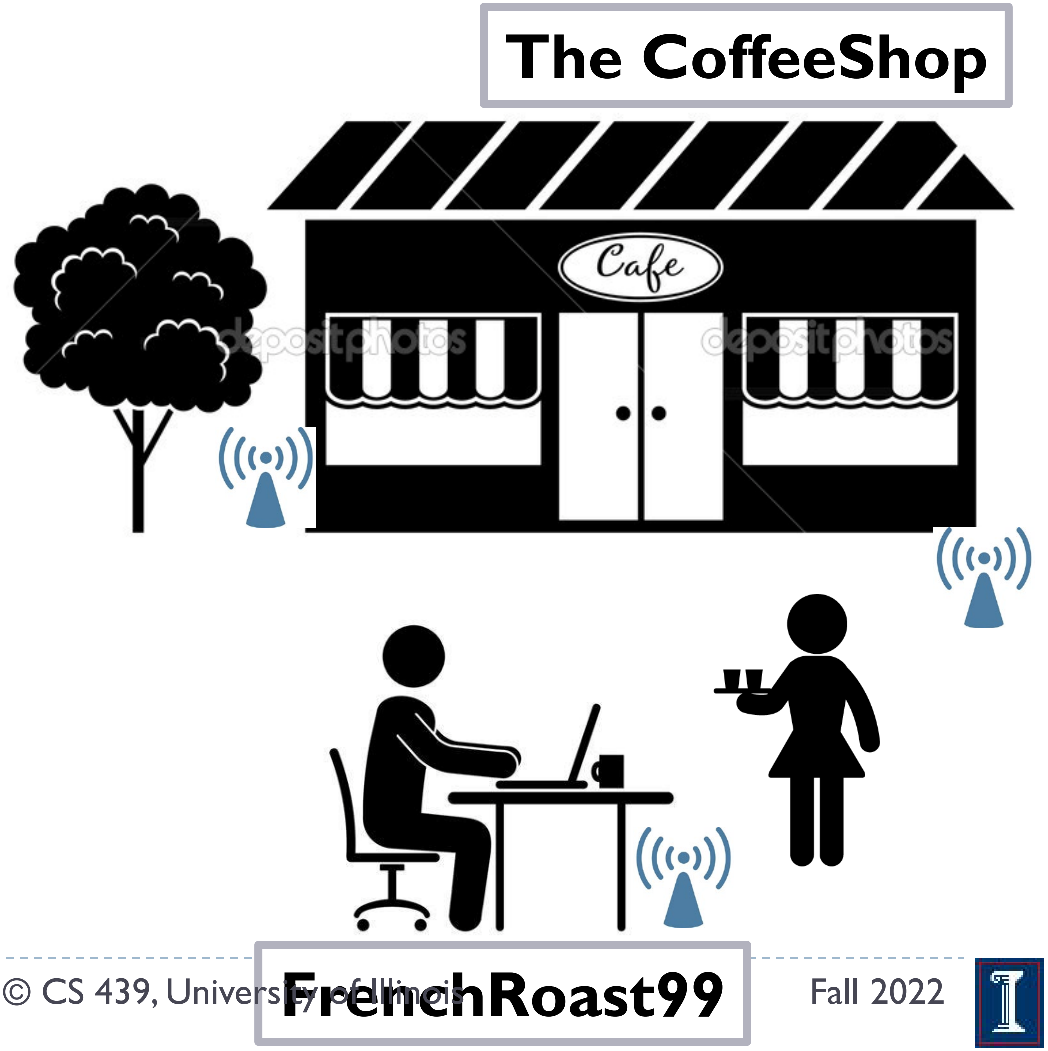
Incognito

- A privacy-preserving IoT ecosystem architecture
  - Users share any desired part of their identity within an environment
- User-managed identities
  - cid: context-based identities



Incognito

- User and Infrastructure Devices
  - Wi-Fi to infrastructure



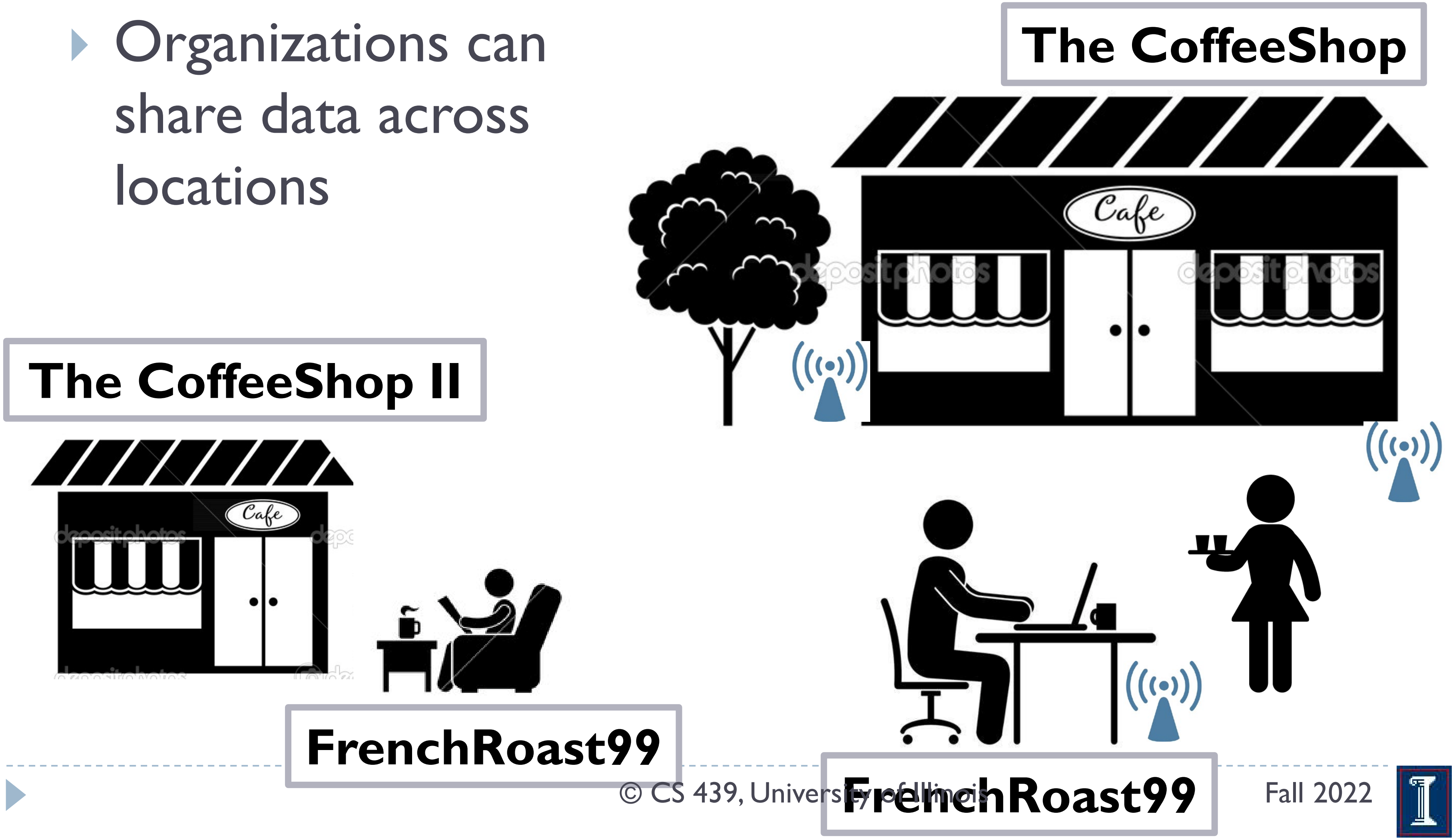
Incognito

- User and Infrastructure Devices
  - Wi-Fi to infrastructure
  - Bluetooth LE for local environment

The CoffeeShop
Cafe
FrenchRoast99

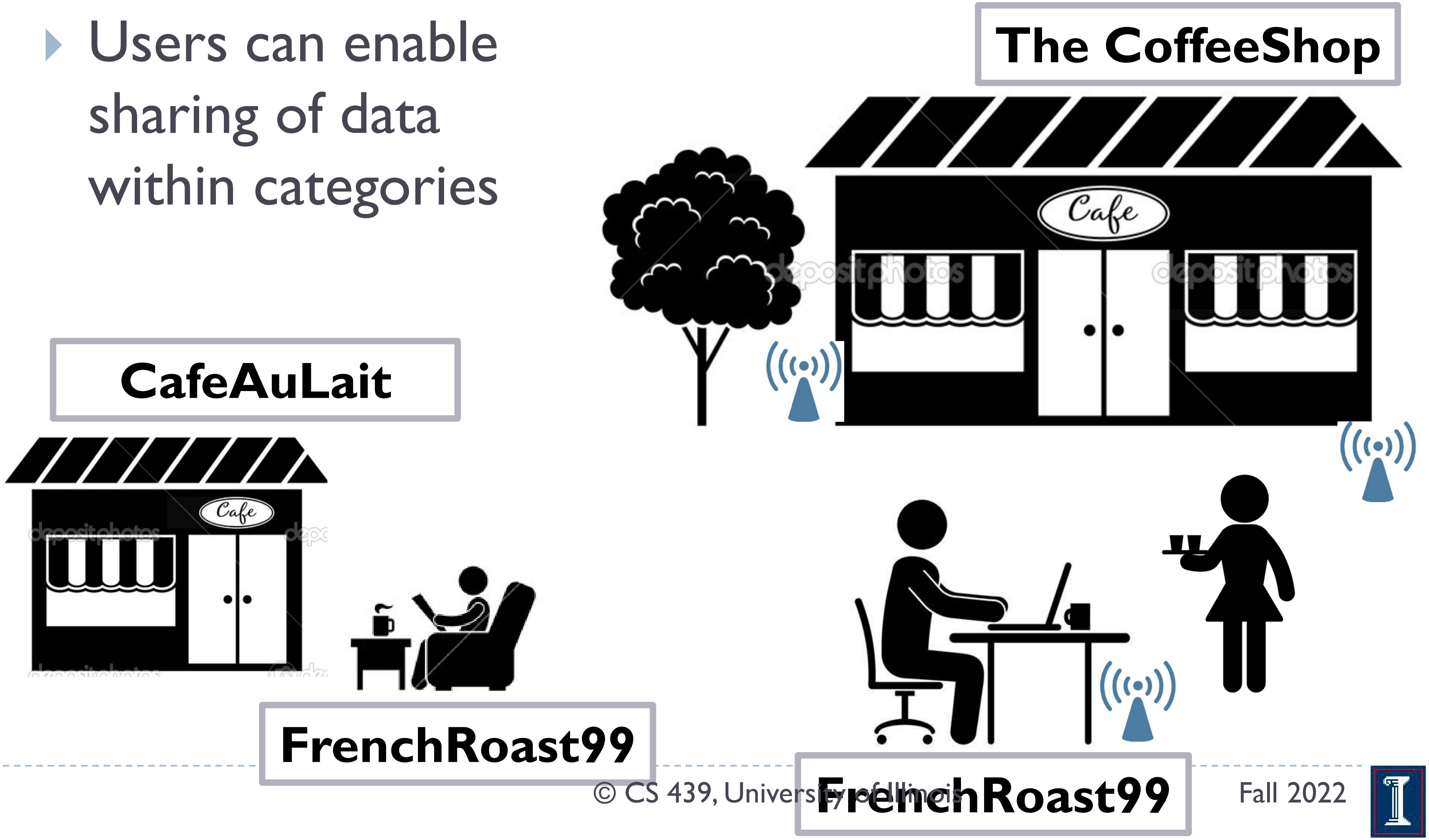
Incognito

- Environments
  - Organizations can share data across locations



Incognito

- Environments
  - Users can enable sharing of data within categories



The CoffeeShop

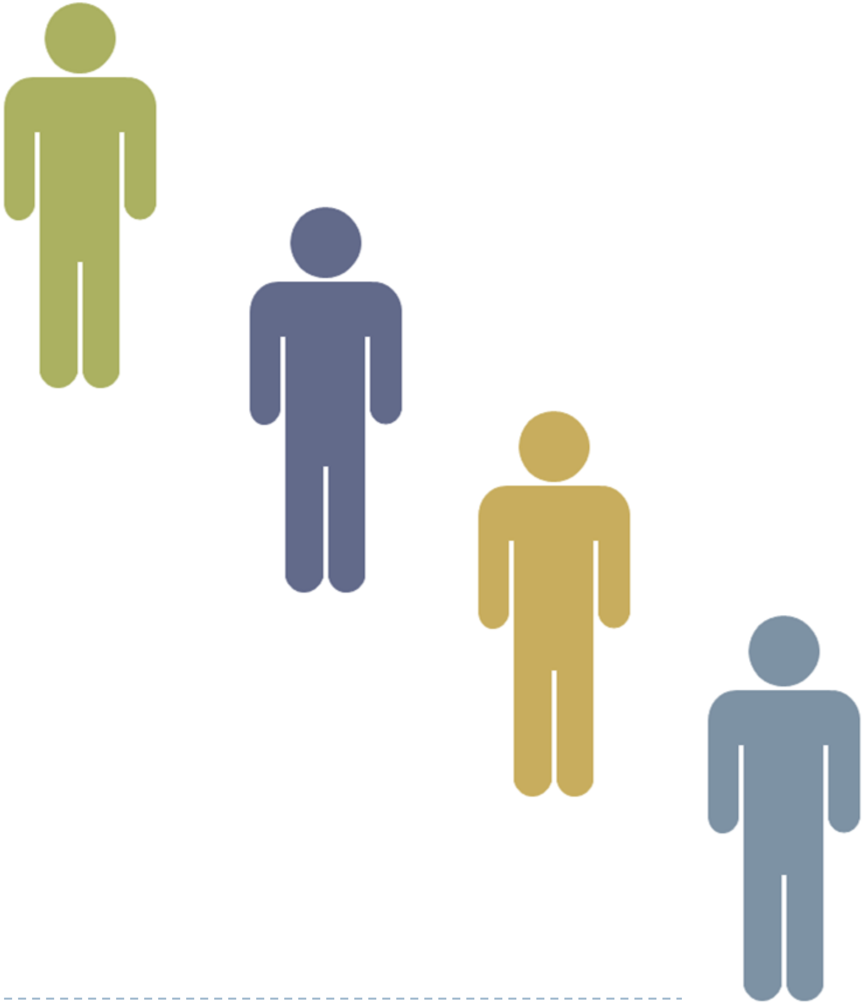
CafeAuLait

FrenchRoast99

FrenchRoast99

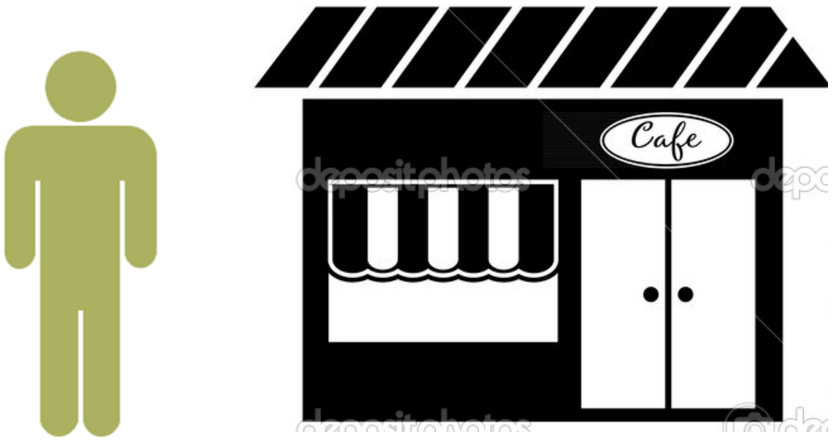
User Identities

- Anonymous
  - Random cid every packet
  - For every message, the user appears as someone new
  - No tracking from environment



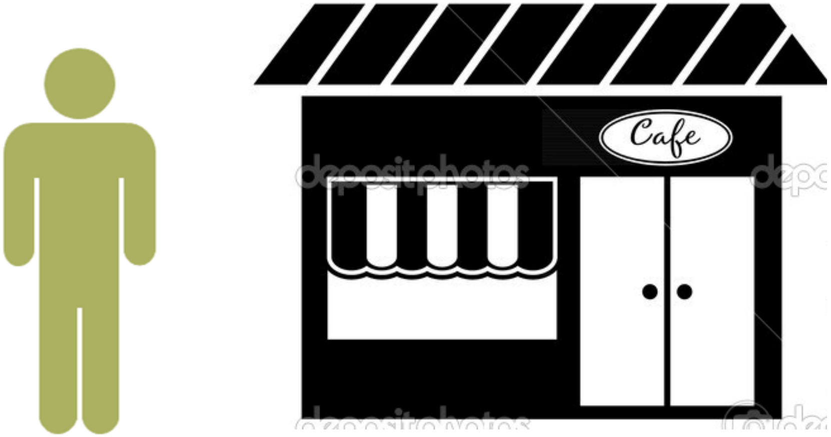
User Identities

- Local-One-Time
  - Random cid per session



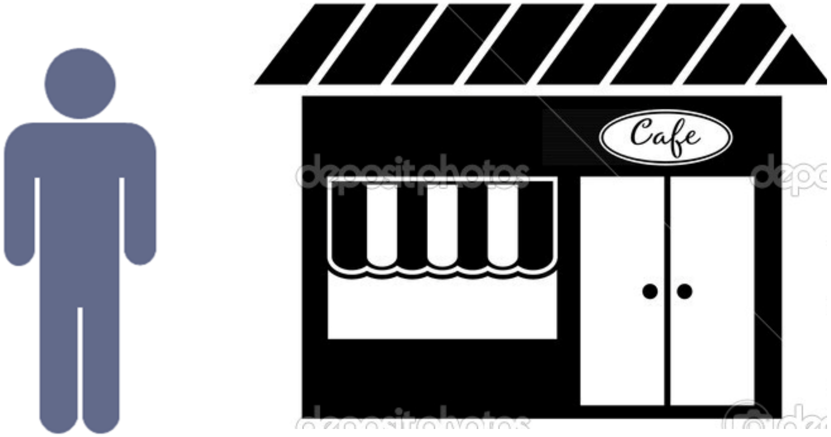
User Identities

- Local-One-Time
  - Random cid per session
  - No connection between multiple sessions from the same user in the same environment



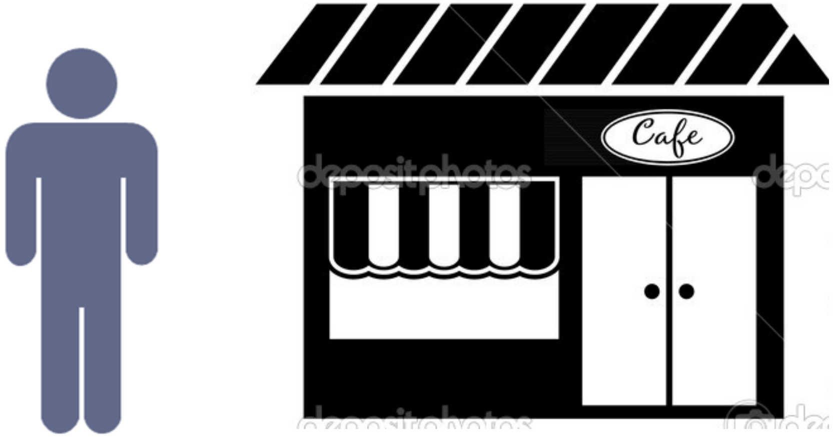
User Identities

- Local
  - Random cid per environment



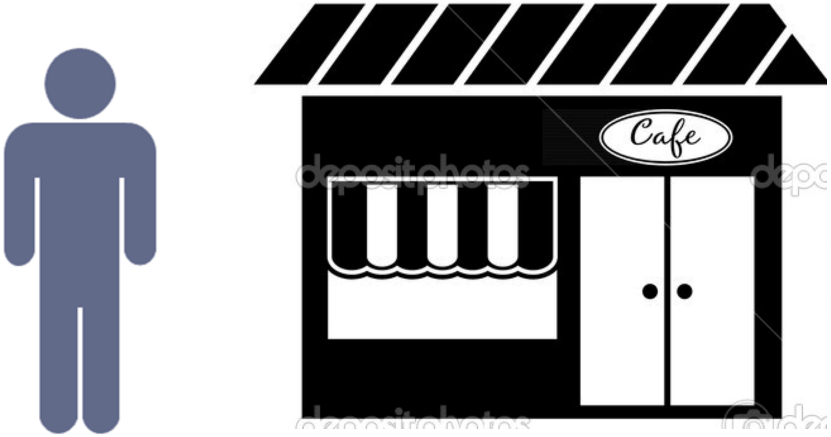
User Identities

- Local
  - Random cid per environment
  - No connection to the same user in a different environment



User Identities

- Cross-Domain
  - Random cid per environment class
  - Track and share user information within an environment class

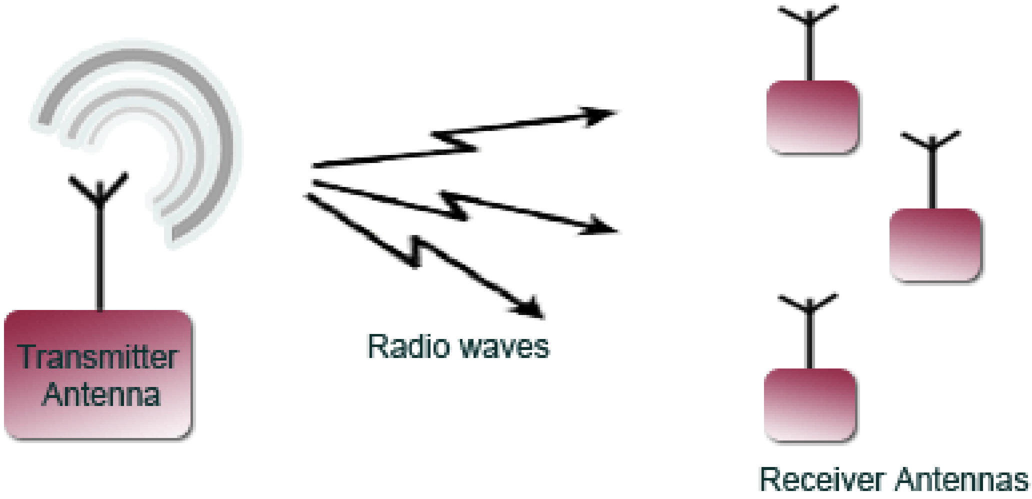


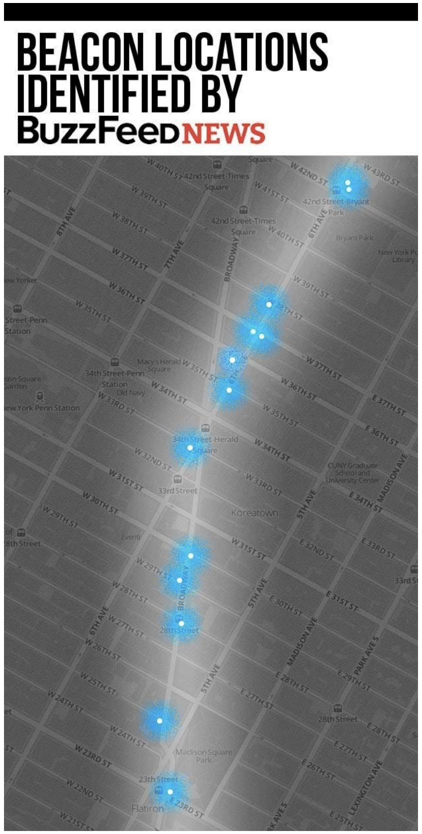
User Identities

- Global
  - Global cid
  - User exposes an identity all of the time

Managing Exposure

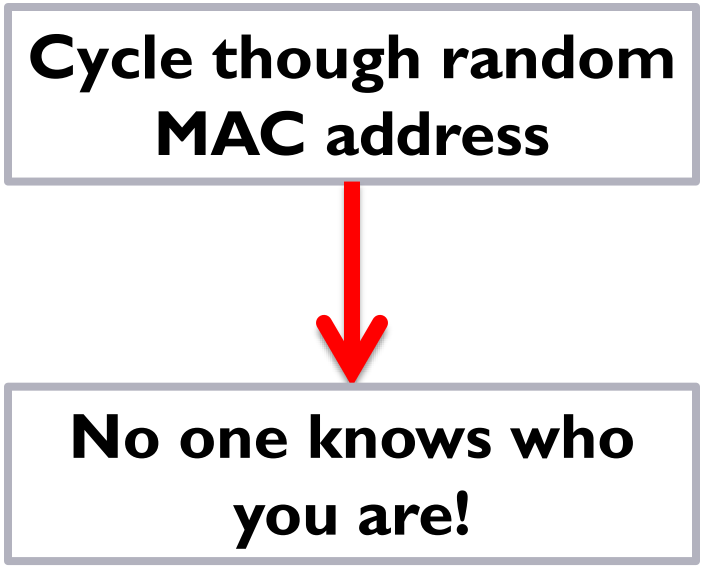
All transmissions contain the identity of the sender (MAC address)





Anyone can listen and track the user

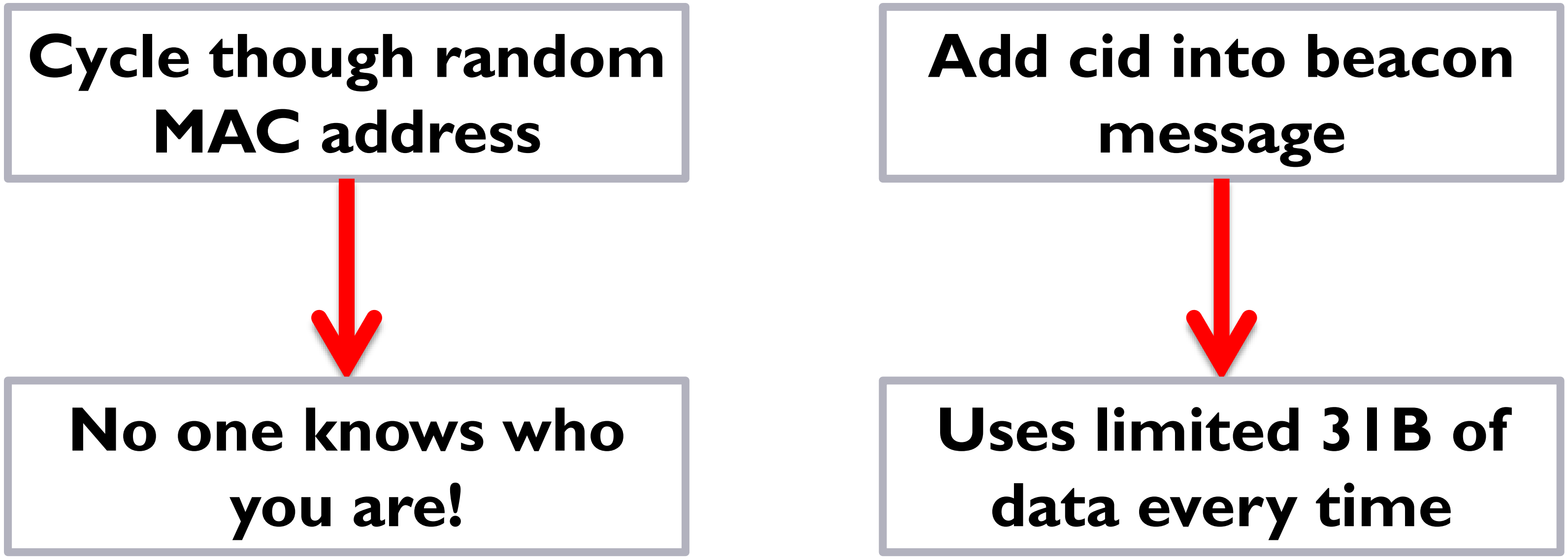
Managing Exposure



Cycle though random MAC address

No one knows who you are!

Managing Exposure



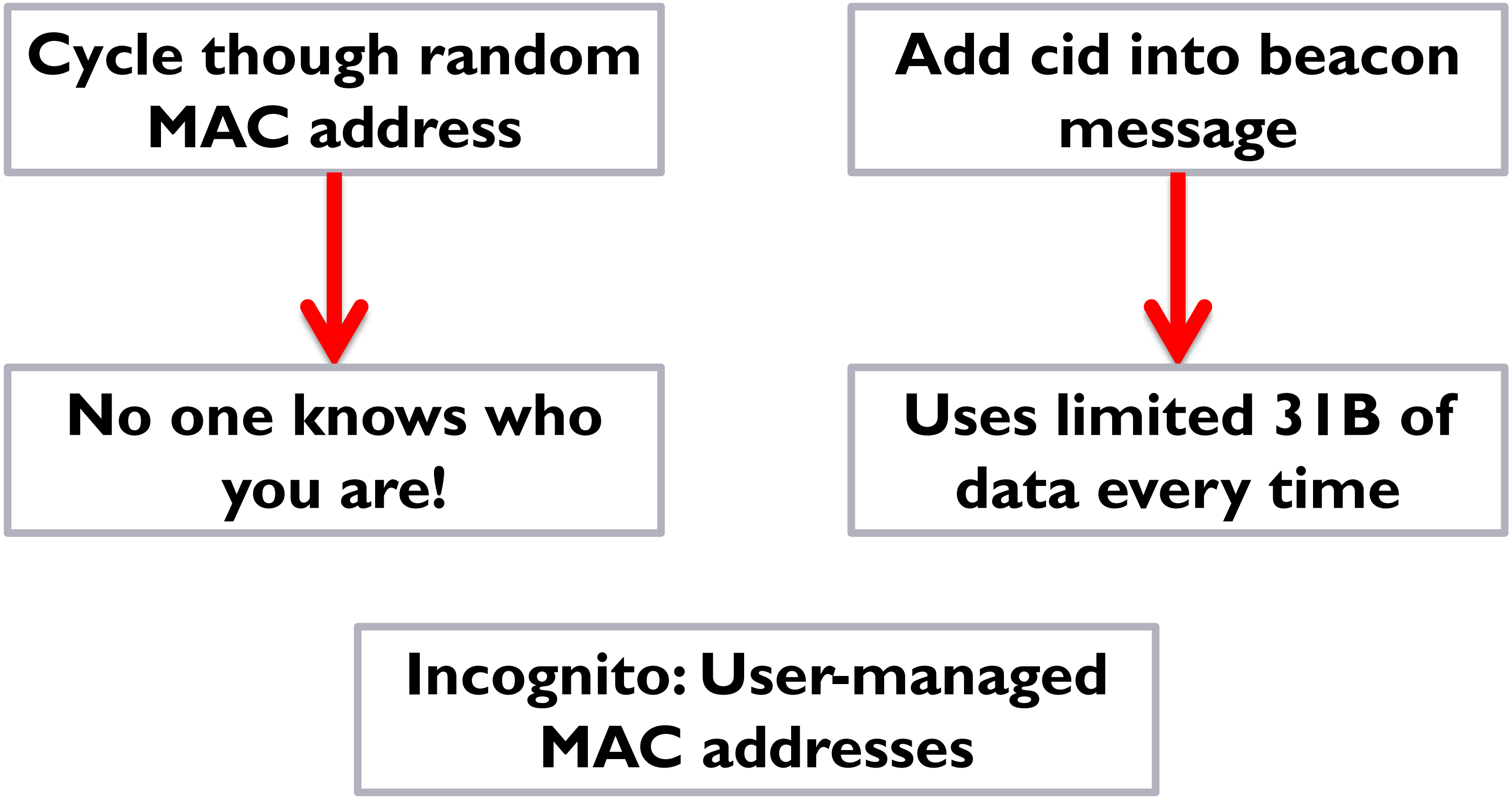
Cycle though random MAC address

No one knows who you are!

Add cid into beacon message

Uses limited 31B of data every time

Managing Exposure



Cycle though random MAC address

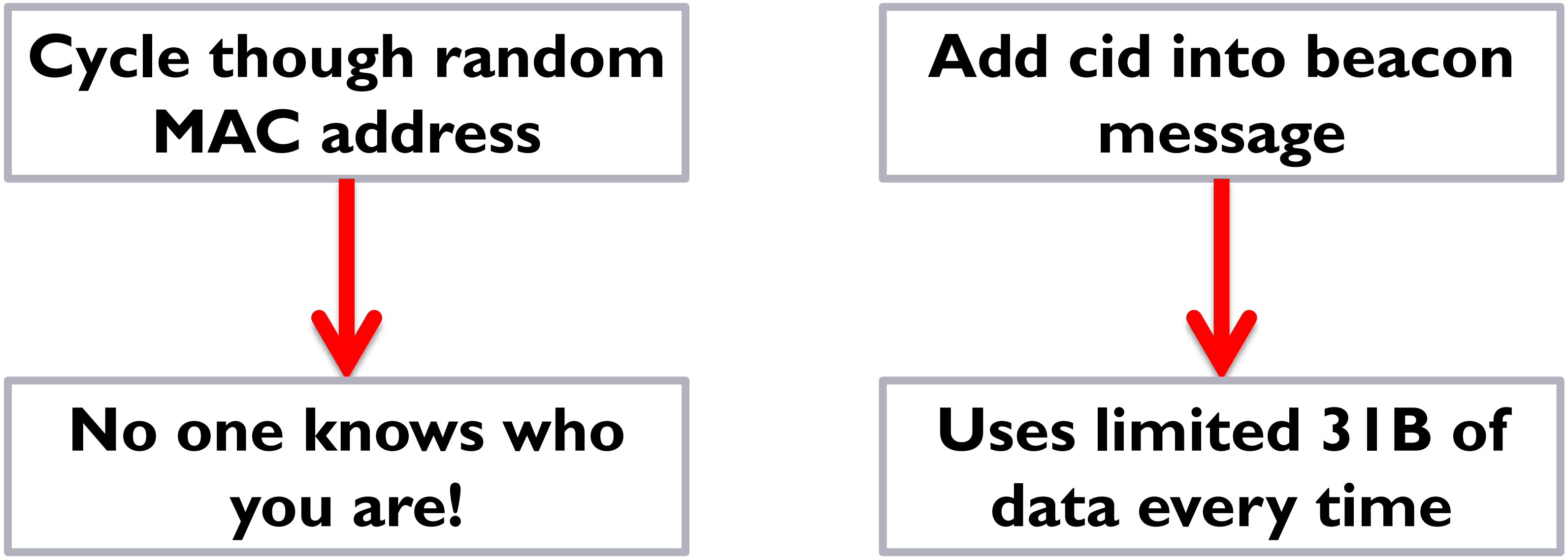
No one knows who you are!

Add cid into beacon message

Uses limited 31B of data every time

Incognito: User-managed MAC addresses

Managing Exposure



Cycle though random MAC address

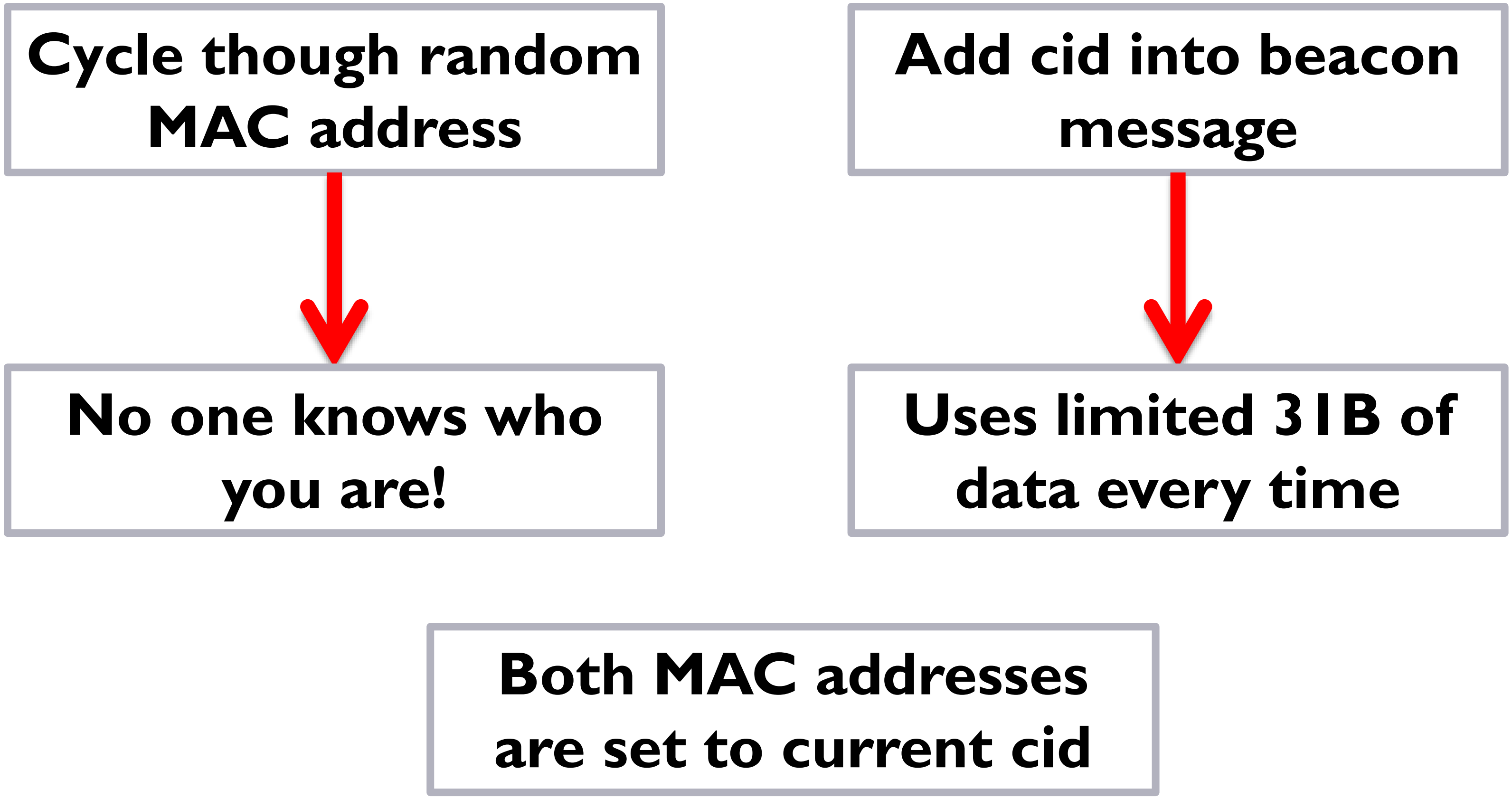
No one knows who you are!

Add cid into beacon message

Uses limited 31B of data every time

Incognito: User-managed MAC addresses BLE and WiFi!

Managing Exposure



Cycle though random MAC address

No one knows who you are!

Add cid into beacon message

Uses limited 31B of data every time

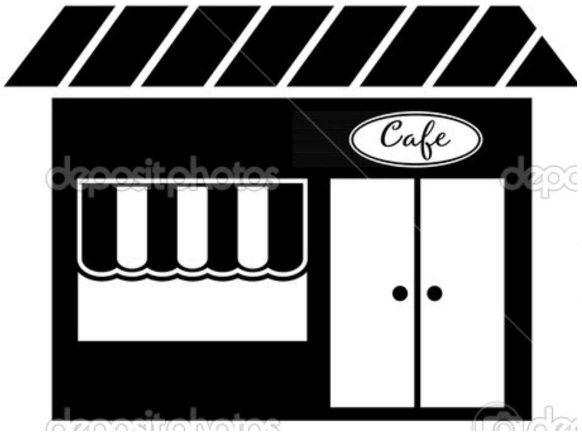
Both MAC addresses are set to current cid

User-to-Environment Sharing

User registers with environment

Wi-Fi

MAC = cid
Data = (PublicKey,
PrivateKeyEncrypti(cid))



Environment specific public-private key pair

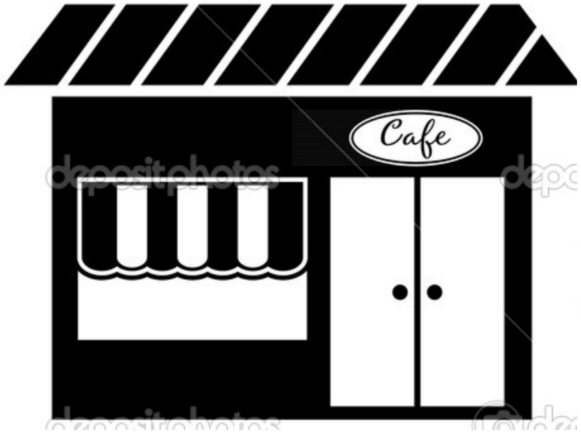
Context ID	Public Key	Encrypted CID
cid	PublicKey	PrivateKey(cid)
cid-a	PublicKey-a	PrivateKey-a(cid-a)

User-to-Environment Sharing

User advertises presence

BLE

MAC = cid
Data = (Hash(TimeStamp),
PrivateKeyEncrypt(TimeStamp))

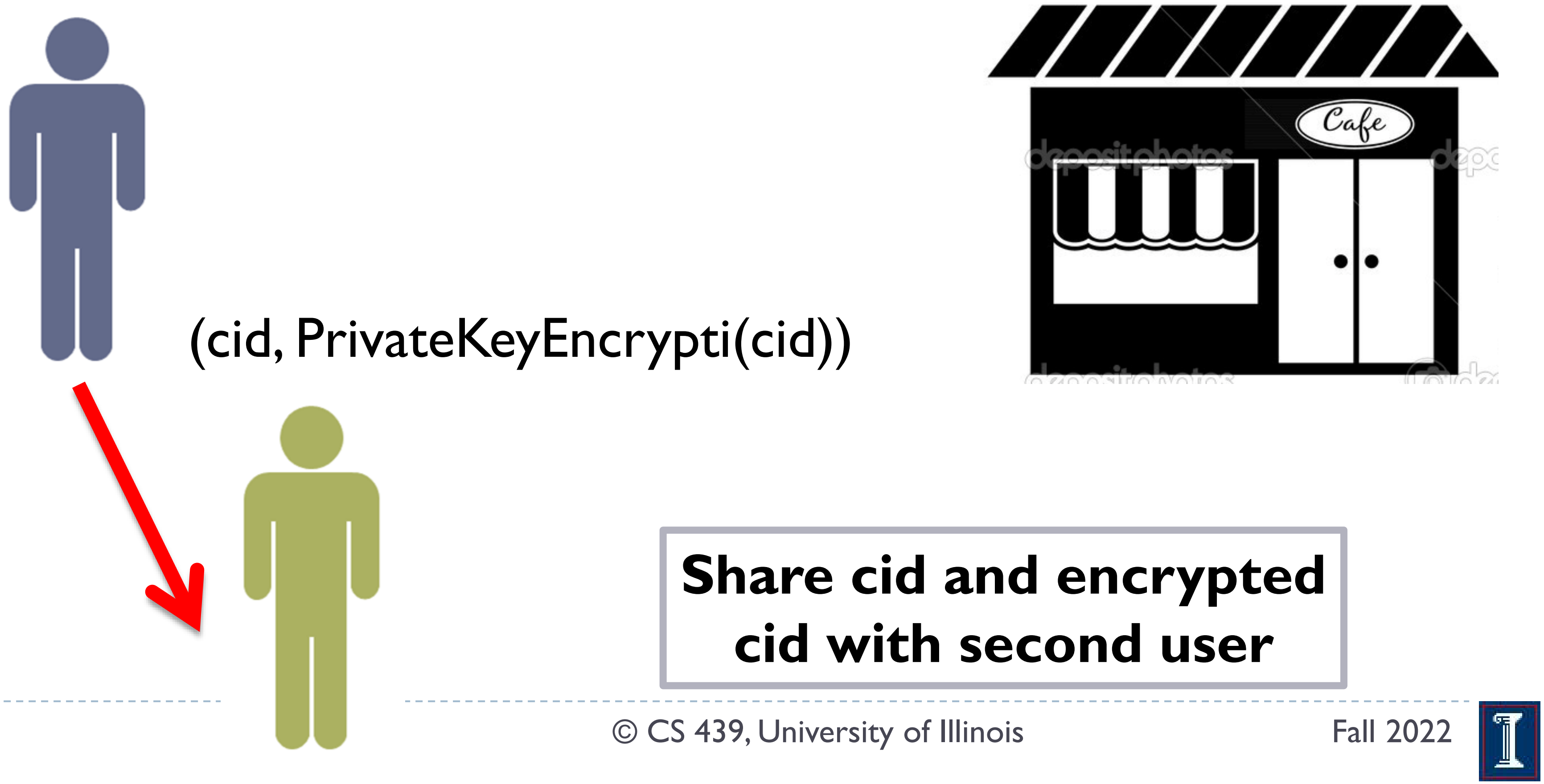


Timestamp added to prevent impersonation

Context ID	Public Key	Encrypted CID
cid	PublicKey	PrivateKey(cid)
cid-a	PublicKey-a	PrivateKey-a(cid-a)

User-to-User Sharing

User enables sharing with friends

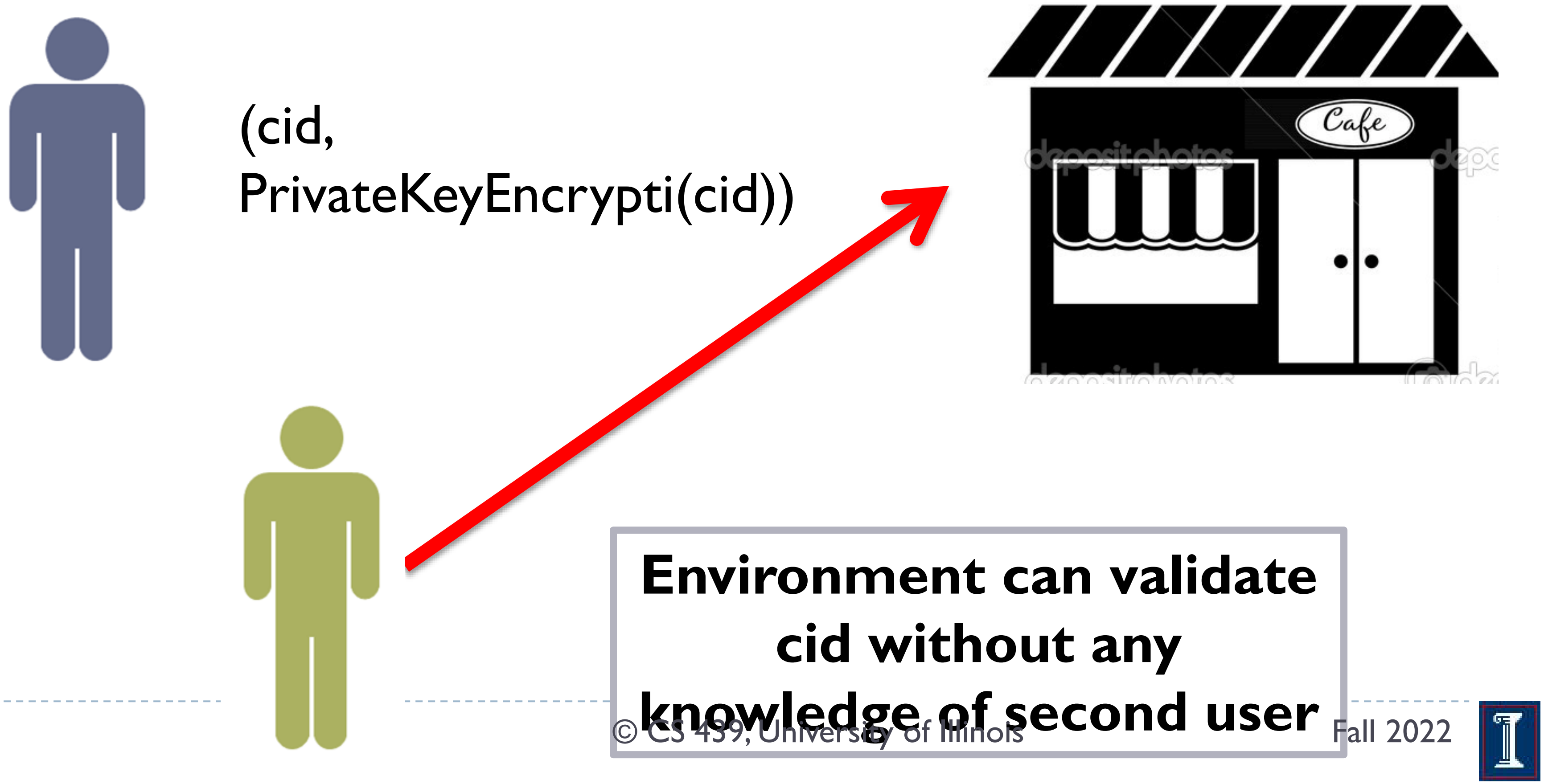


(cid, PrivateKeyEncrypti(cid))

Share cid and encrypted cid with second user

User-to-User Sharing

Friend queries environment data



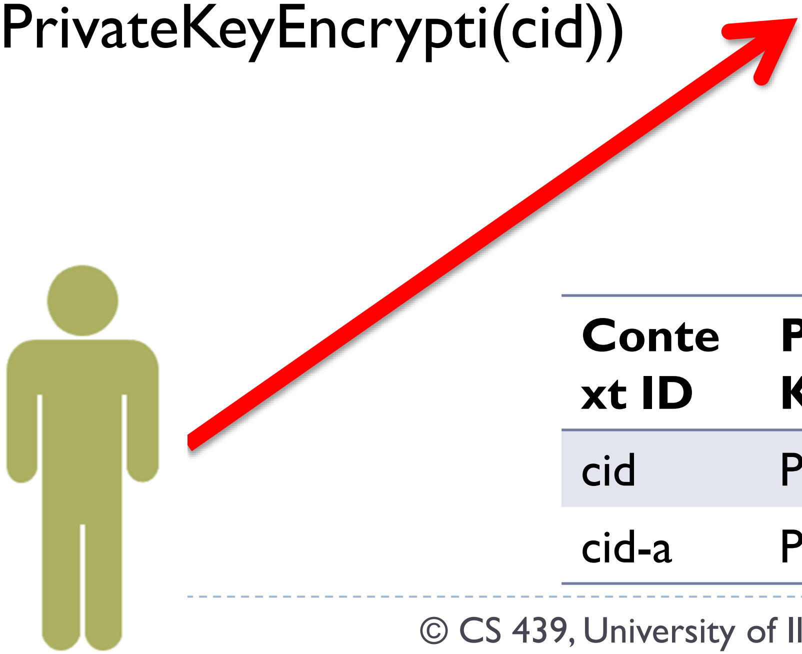
(cid,
PrivateKeyEncrypti(cid))

Environment can validate cid without any knowledge of second user

User-to-User Sharing

Friend queries environment data

(cid, PrivateKeyEncrypti(cid))

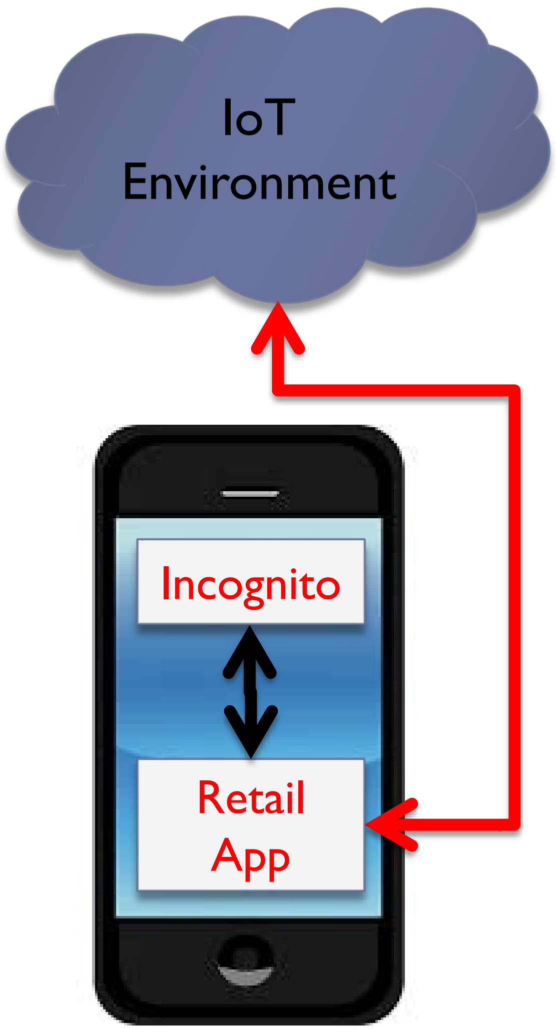


Context ID	Public Key	Encrypted CID
cid	PublicKey	PrivateKey(cid)
cid-a	PublicKey-a	PrivateKey-a(cid-a)

Realizing Incognito

Ensuring privacy from malicious apps

Challenge:
External apps could leak cid

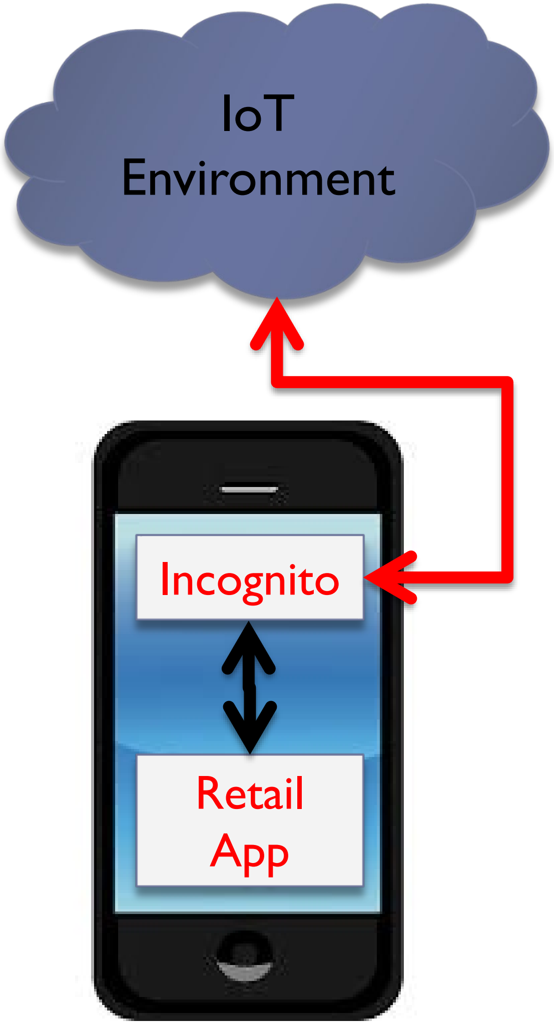


Realizing Incognito

Ensuring privacy from malicious apps

Challenge:
External apps could leak cid

Solution:
Sandbox IoT apps
All communication with IoT infrastructure through incognito



Realizing Incognito

CID Lifetime and Exposure

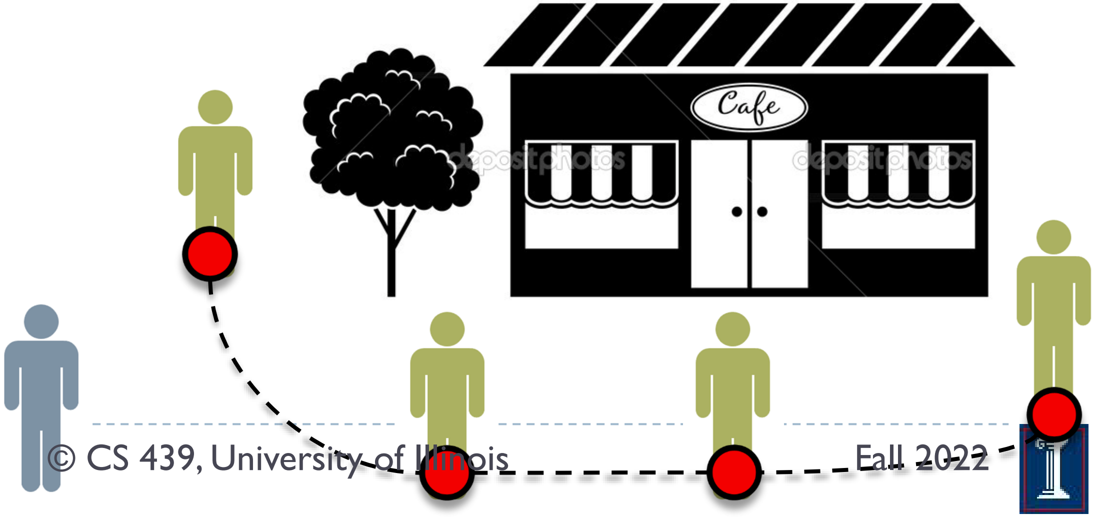
Challenge:
One environment could snoop into the other

Realizing Incognito

CID Lifetime and Exposure

Challenge:
One environment could snoop into the other

Challenge:
A snooper can track the path of a cid



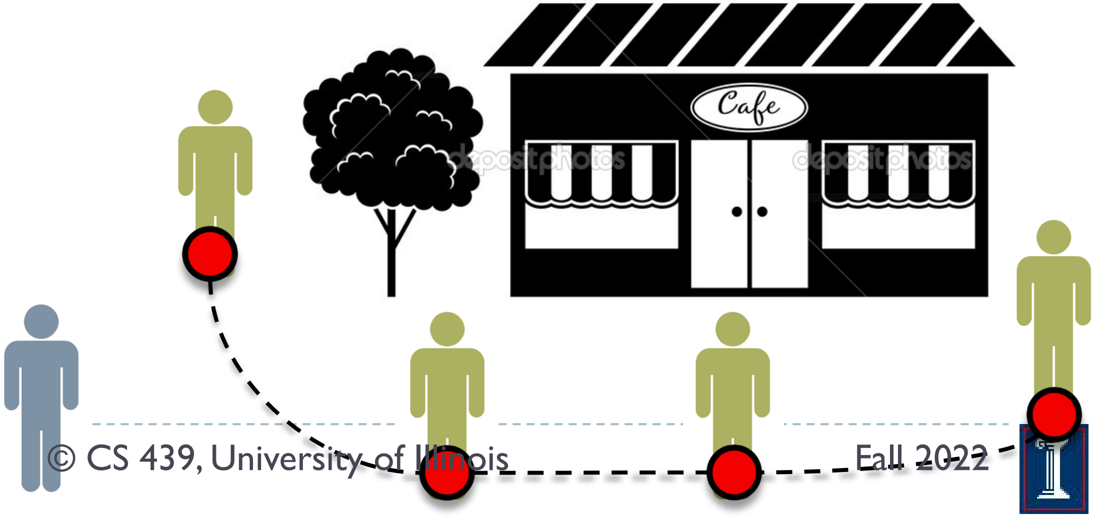
Realizing Incognito

CID Lifetime and Exposure

Challenge:
One environment could snoop into the other

Challenge:
A snooper can track the path of a cid

Can we bring back random MAC addresses and still maintain cid as a location based identifier?



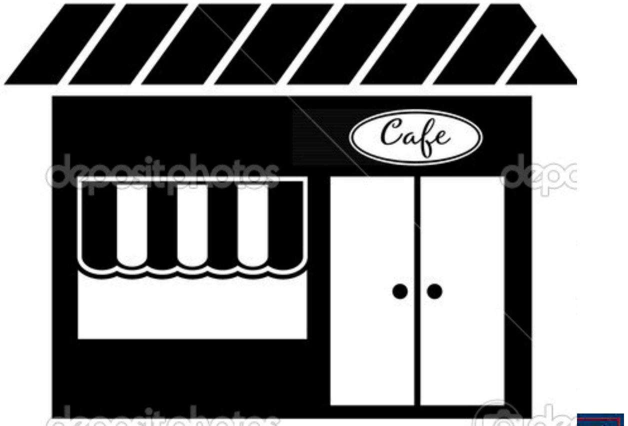
Anonymizing MAC Addresses

Registration

MAC = cid
Data = (PublicKey, PrivateKeyEncrypti(cid))

Advertising

MAC = cid
Data = (Hash(TimeStamp), PrivateKeyEncrypt(TimeStamp))



Anonymizing MAC Addresses

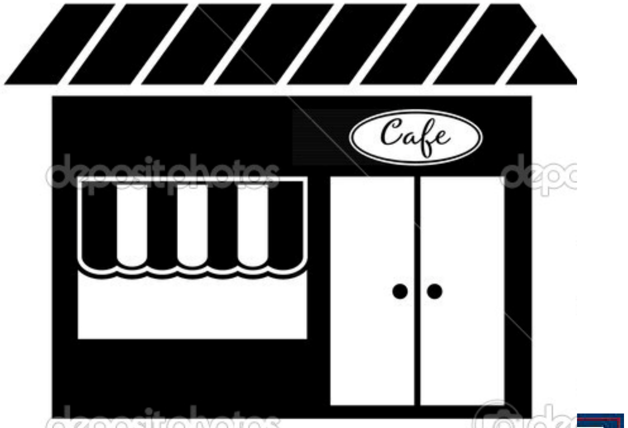
Registration

MAC = cid
Data = (PublicKey,
PrivateKeyEncrypti(cid))

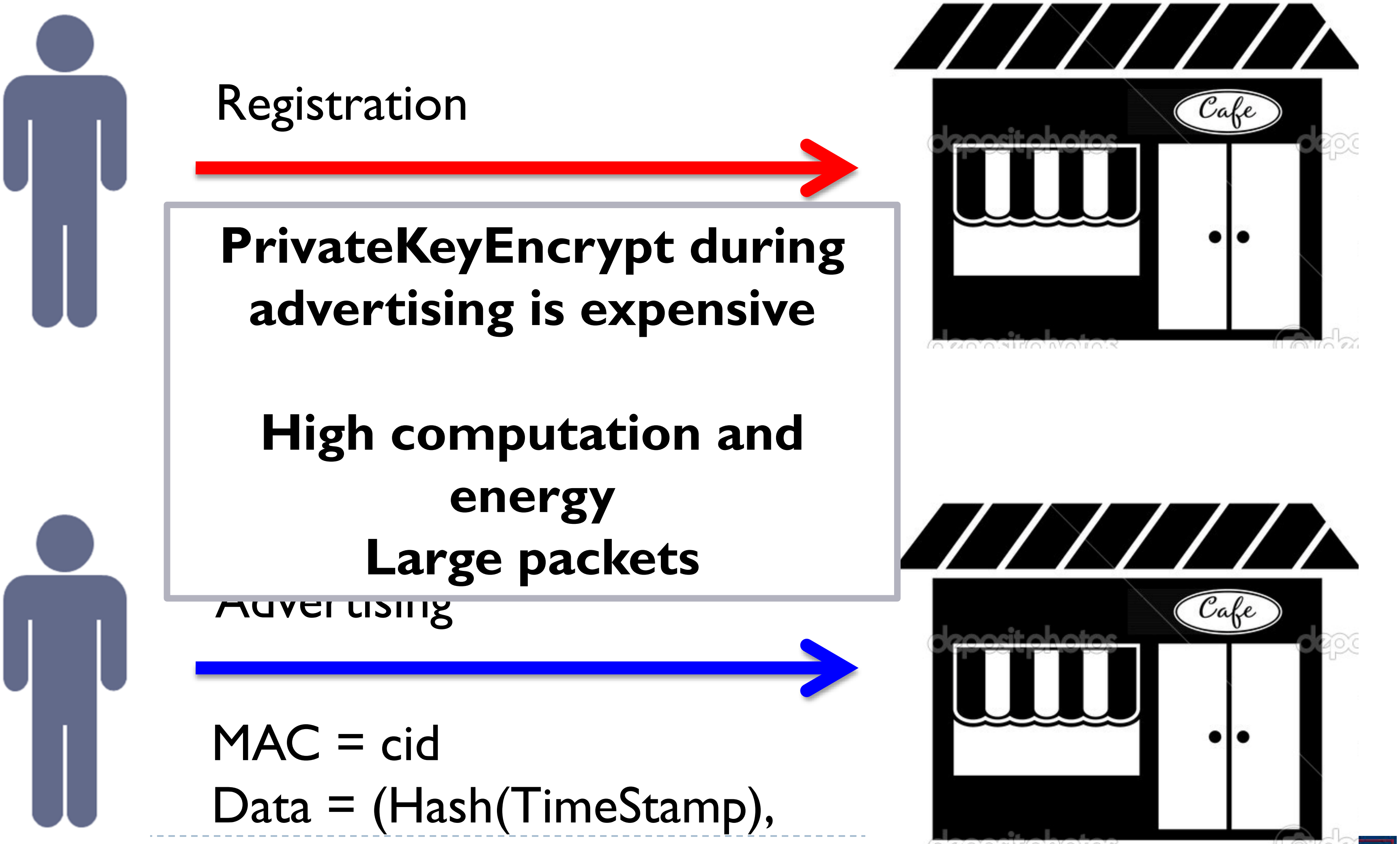
Only prevents impersonation, not tracking

Advertising

MAC = cid
Data = (Hash(TimeStamp),
PrivateKeyEncrypt(TimeStamp))



Anonymizing MAC Addresses



Registration

PrivateKeyEncrypt during advertising is expensive

High computation and energy

Large packets

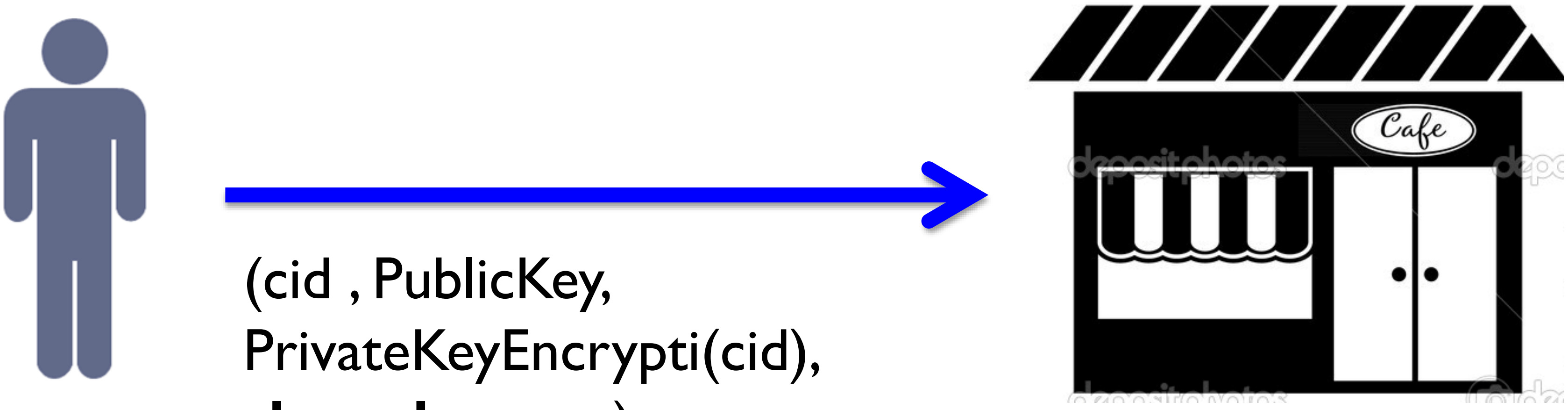
Advertising

MAC = cid

Data = (Hash(TimeStamp), PrivateKeyEncrypt(TimeStamp))

Lamina

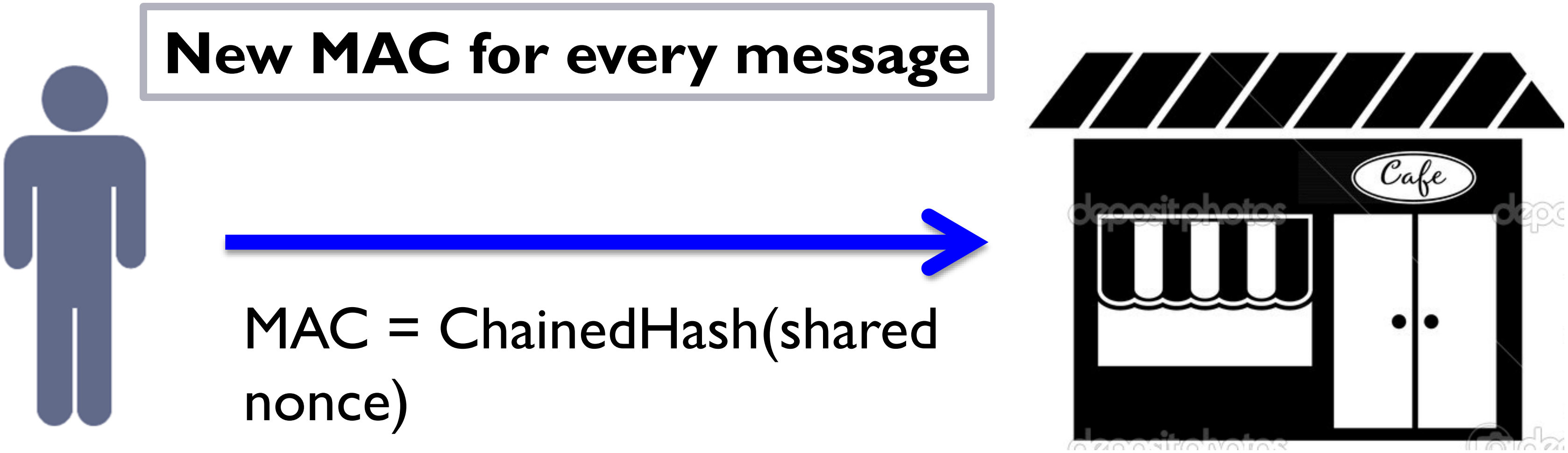
User registers with environment



(cid , PublicKey,
PrivateKeyEncrypti(cid),
shared nonce)

Lamina

User advertises presence

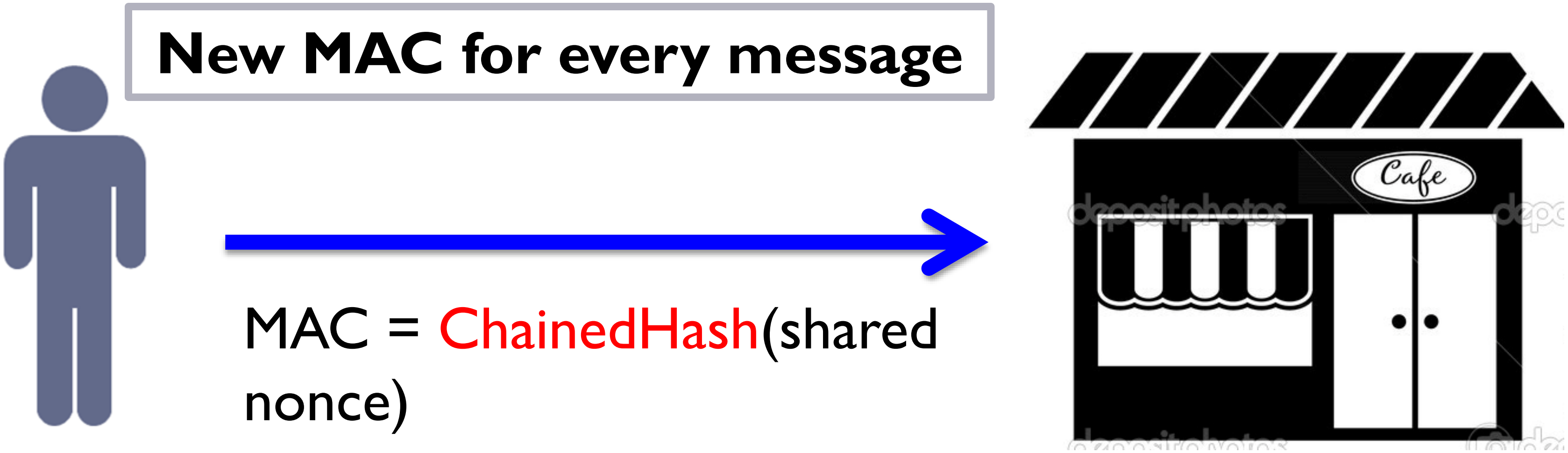


New MAC for every message

MAC = ChainedHash(shared nonce)

Lamina

User advertises presence



New MAC for every message

MAC = ChainedHash(shared nonce)

Problem
Many encryption protocols assume a reliable channel

Lamina

User advertises presence

New MAC for every message

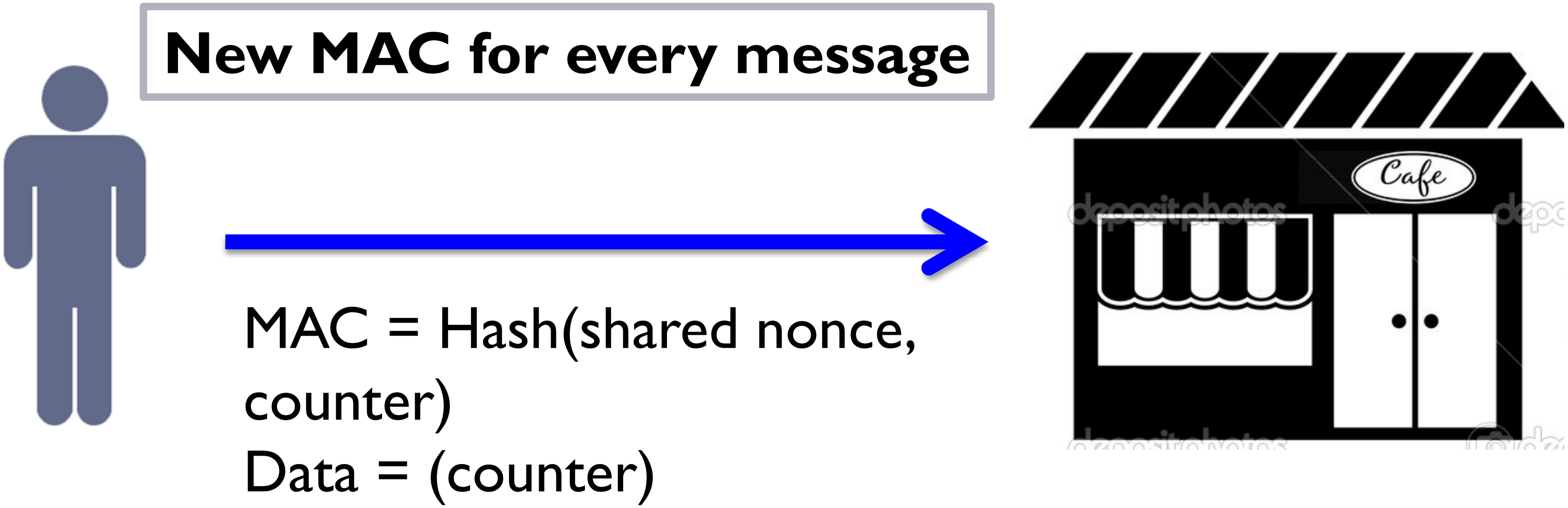
MAC = ChainedHash(shared nonce)

Problem
Many encryption protocols assume a reliable channel

Loss →protocol desync, further decryption of the stream impossible

Lamina

User advertises presence



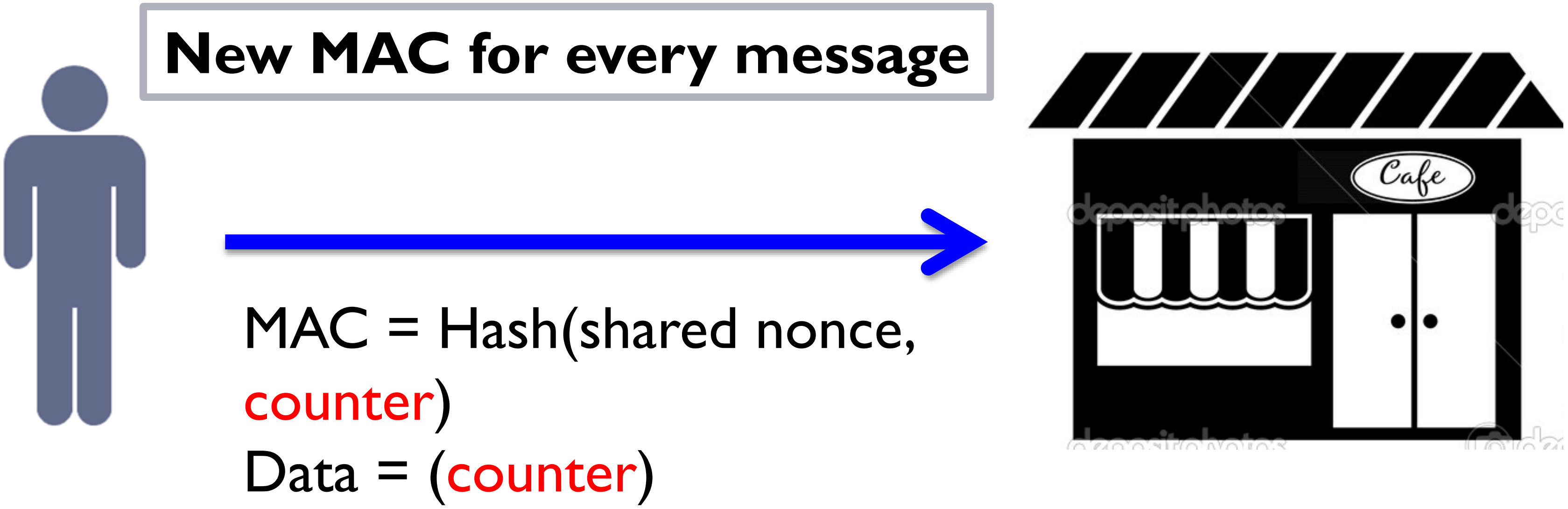
New MAC for every message

MAC = Hash(shared nonce, counter)
Data = (counter)

Use AES Counter mode

Lamina

User advertises presence



New MAC for every message

MAC = Hash(shared nonce, counter)

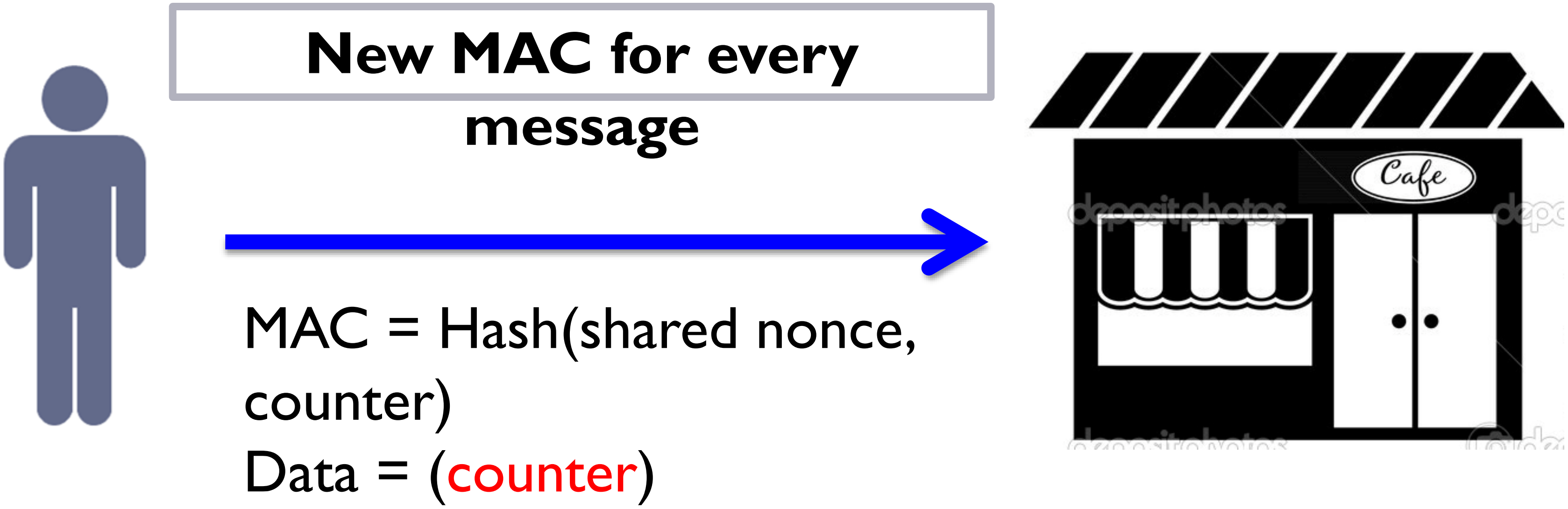
Data = (counter)

Use AES Counter mode

Self-synchronizing cipher maintains encrypted channel with packet loss

Lamina

User advertises presence



New MAC for every message

MAC = Hash(shared nonce, counter)

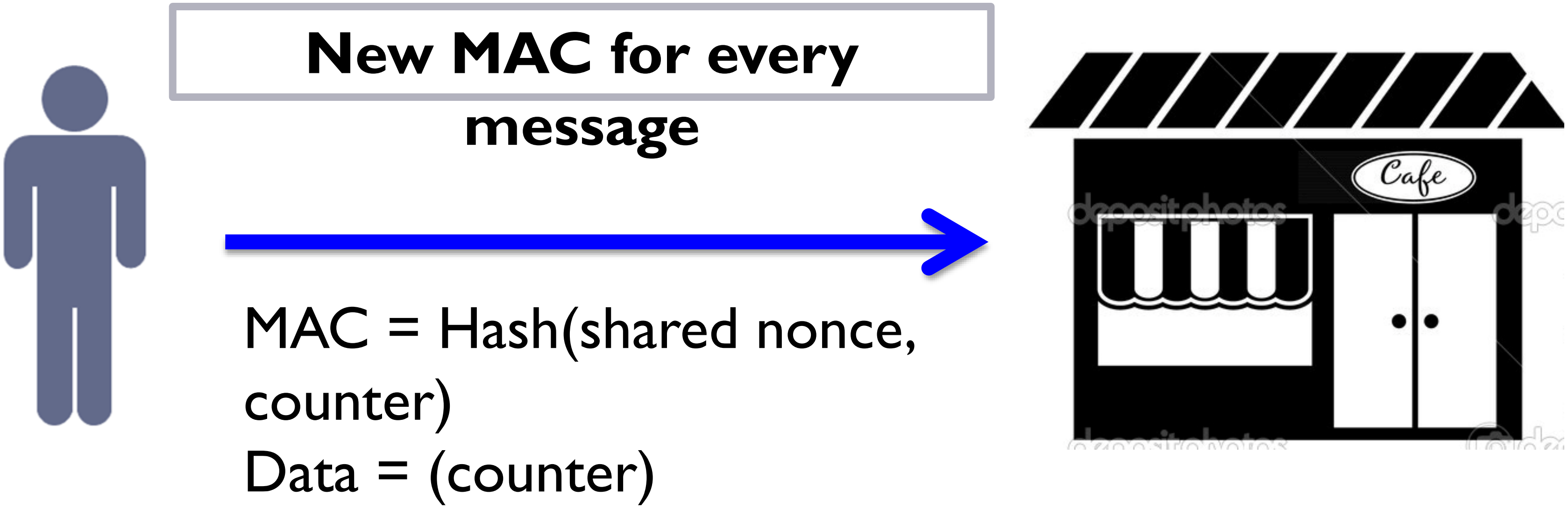
Data = (counter)

Problem

Incremental counters can leak information

Lamina

User advertises presence



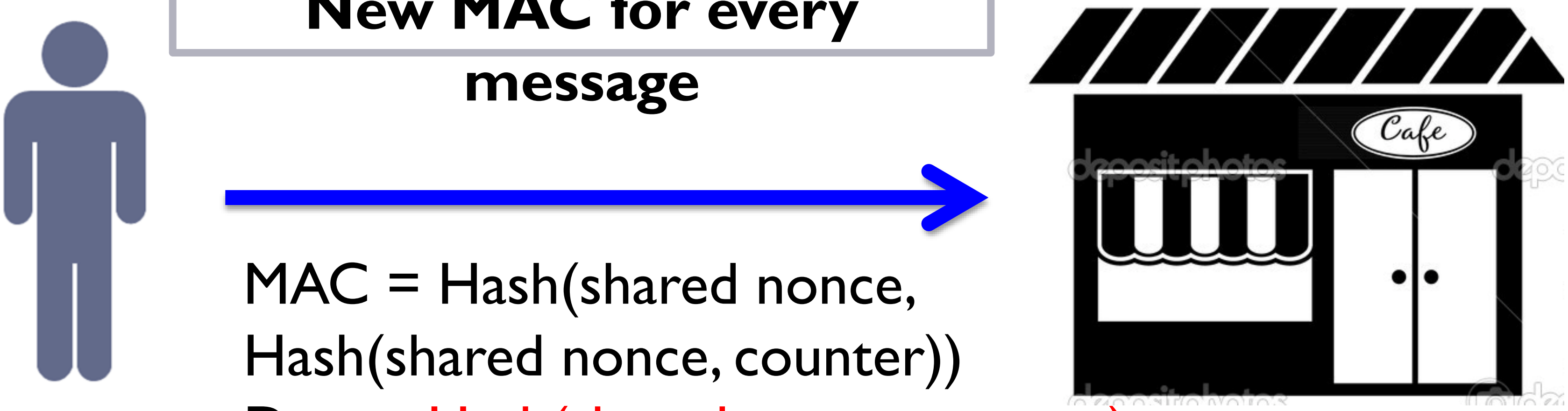
New MAC for every message

MAC = Hash(shared nonce, counter)
Data = (counter)

Counter does not have to be an incremental sequence, it just has to change

Lamina

User advertises presence



New MAC for every message

MAC = Hash(shared nonce, Hash(shared nonce, counter))

Data = Hash(shared nonce, counter)

Counter does not have to be an incremental sequence, it just has to change

Use shared sequence of randomized counters

Lamina

User advertises presence

New MAC for every message

MAC = Hash(shared nonce, Hash(shared nonce, counter))

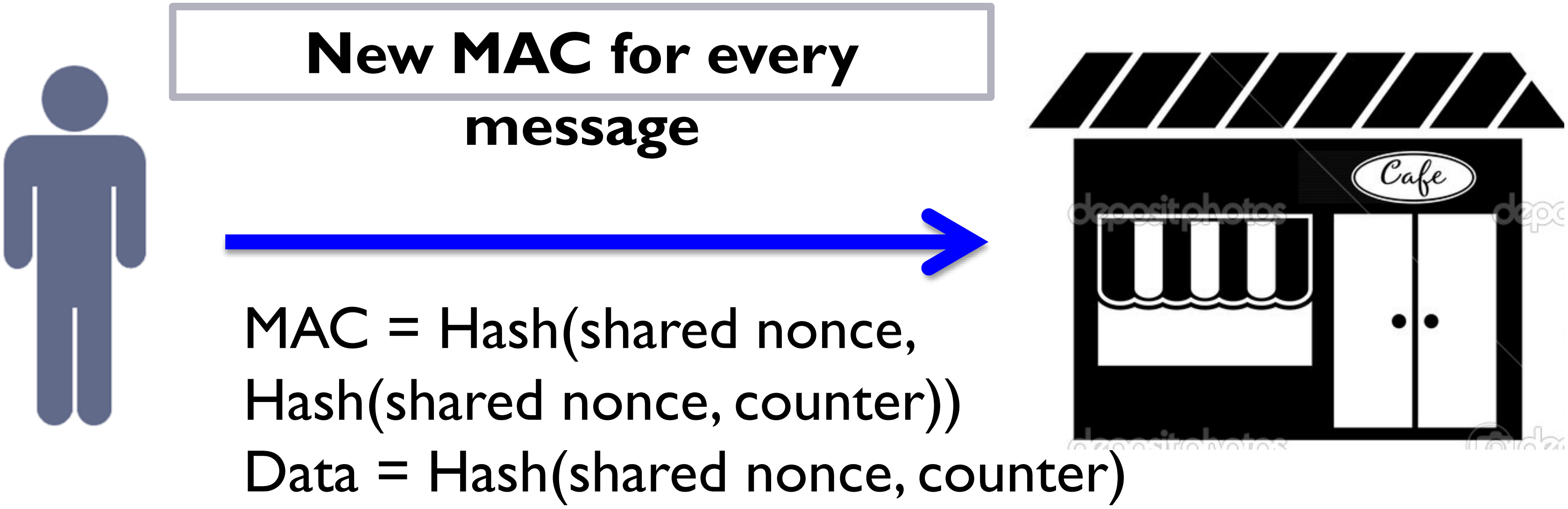
Data = Hash(shared nonce, counter)

Counter does not have to be an incremental sequence, it just has to change

Hashed Counter = Hash(shared nonce, counter)

Lamina

User advertises presence



New MAC for every message

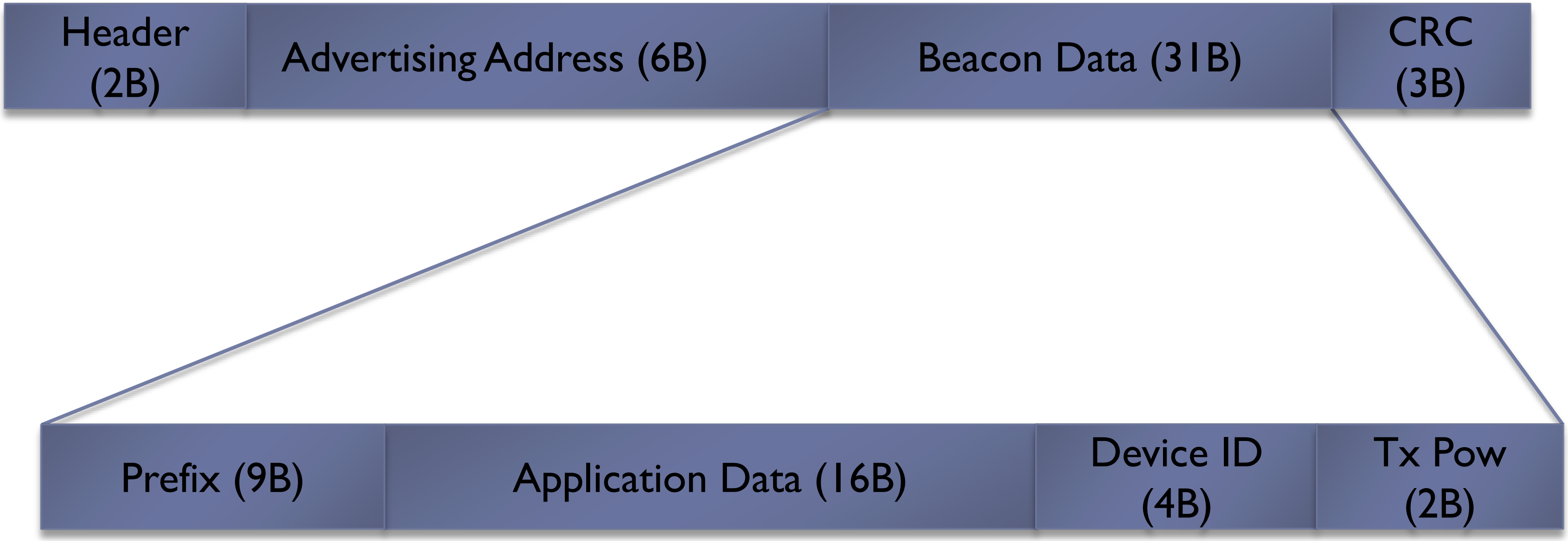
MAC = Hash(shared nonce, Hash(shared nonce, counter))
Data = Hash(shared nonce, counter)

Counter does not have to be an incremental sequence, it just has to change

Counter sequence can be pre-loaded and pre-hashed for low-power and low-computation devices

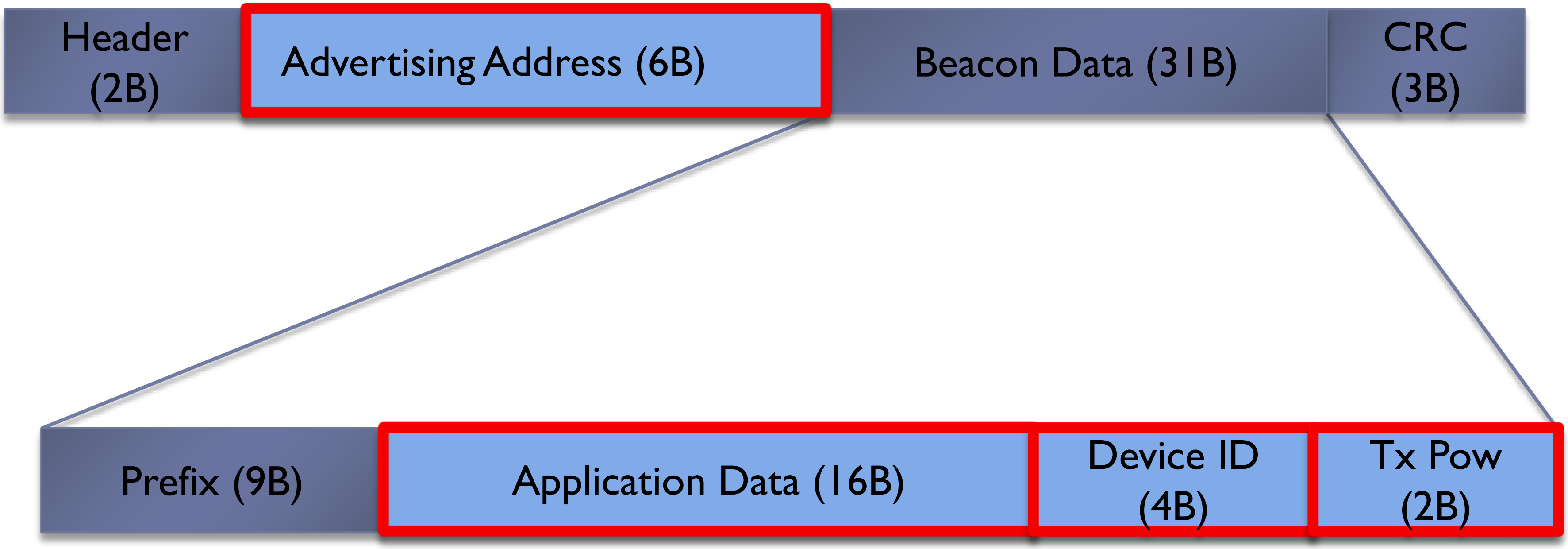
Lamina and BLE

BLE Packet Format



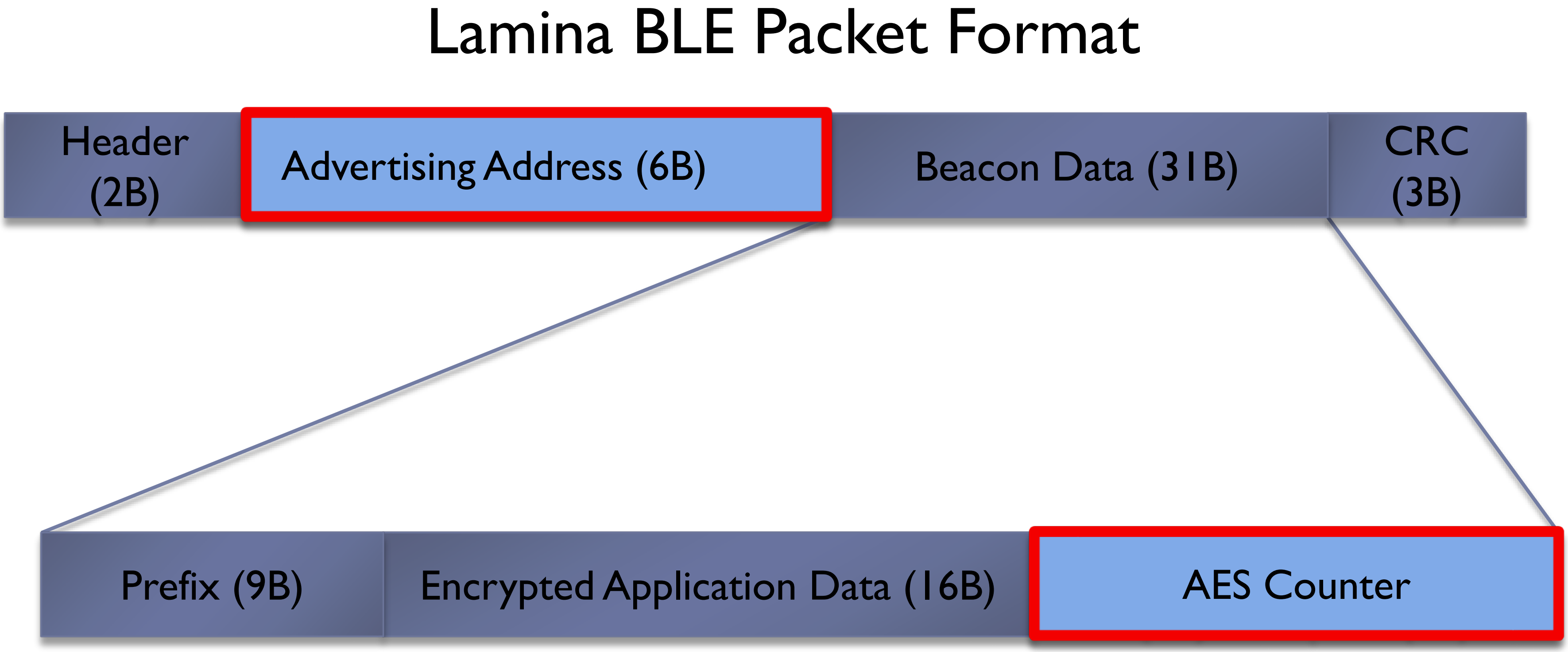
Lamina and BLE

BLE Packet Format



(Potentially privacy-leaking fields)

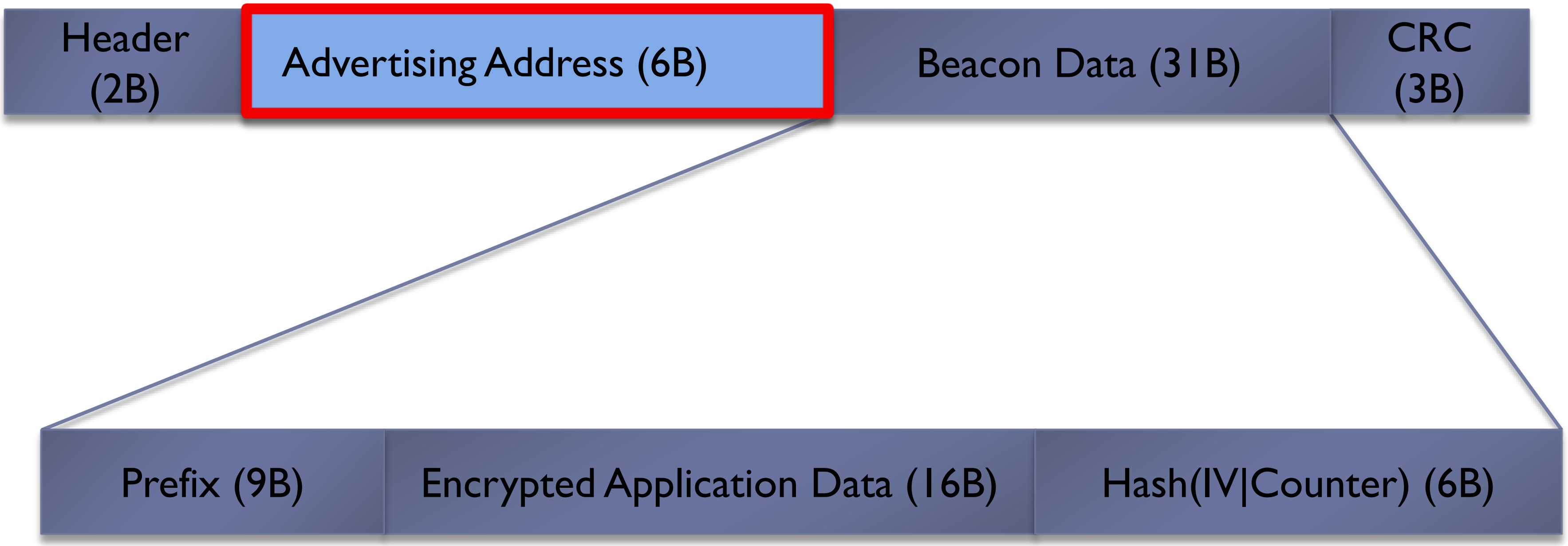
Lamina and BLE



(Potentially privacy-leaking fields)

Lamina and BLE

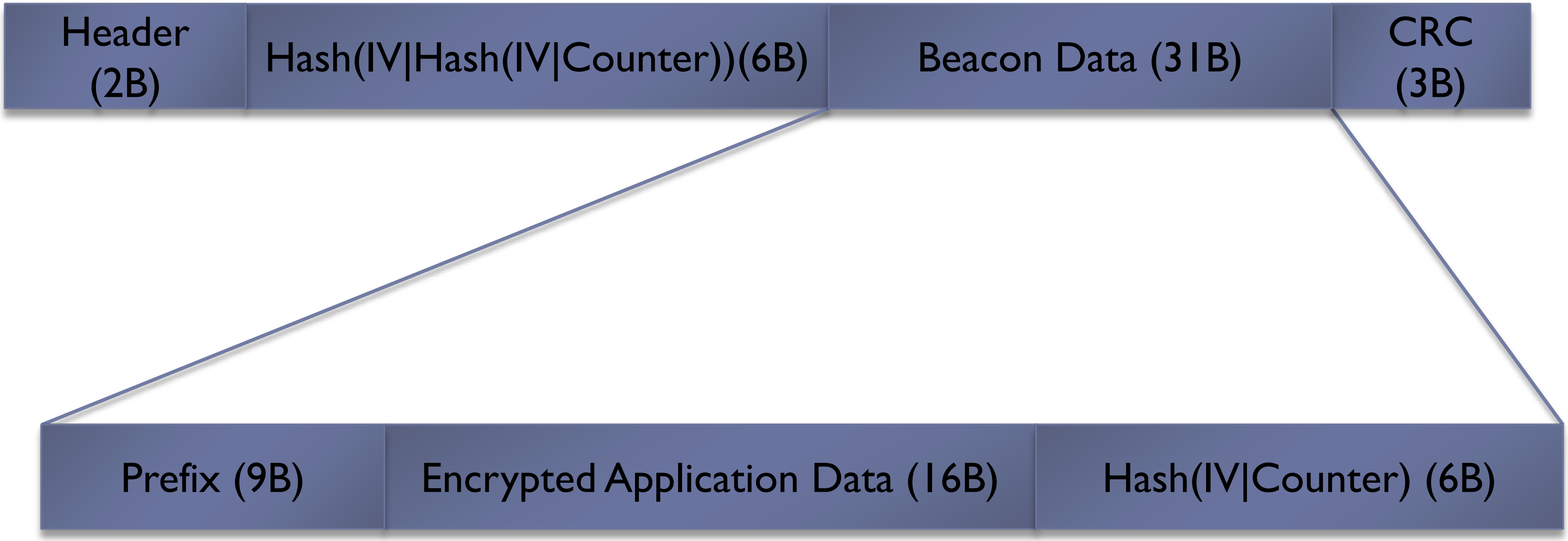
Lamina BLE Packet Format



(Potentially privacy-leaking fields)

Lamina and BLE

Lamina BLE Packet Format



Header (2B)

Hash(IV|Hash(IV|Counter))(6B)

Beacon Data (31B)

CRC (3B)

Prefix (9B)

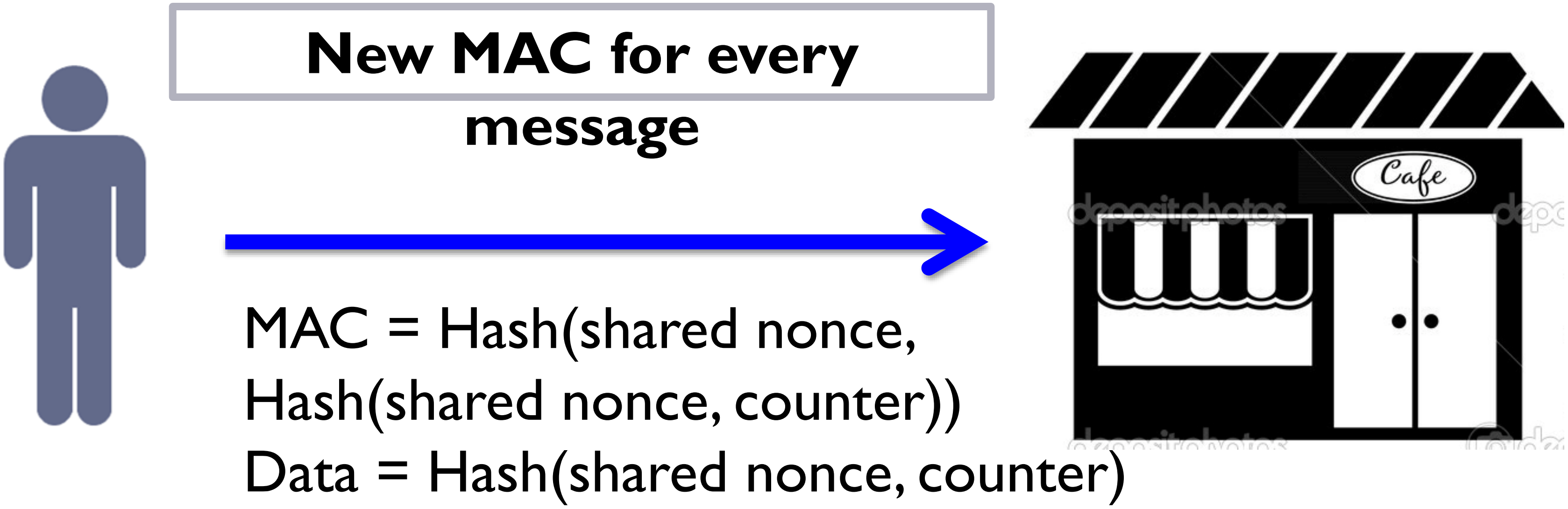
Encrypted Application Data (16B)

Hash(IV|Counter) (6B)

(Potentially privacy-leaking fields)

Lamina

User advertises presence



New MAC for every message

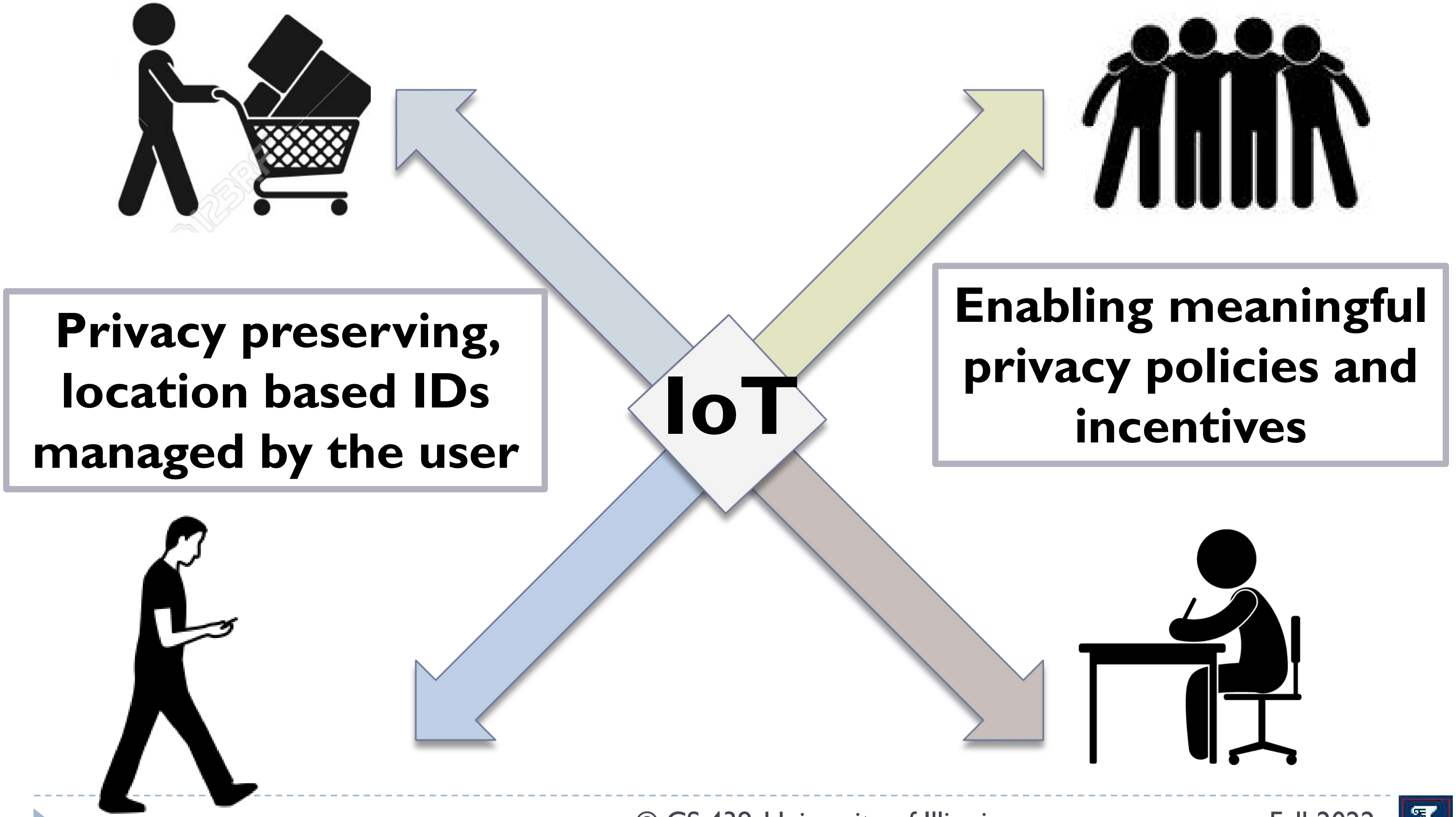
MAC = Hash(shared nonce, Hash(shared nonce, counter))
Data = Hash(shared nonce, counter)

Unique MAC address identifiable by IoT

No public key cryptography

Loss tolerant

Incognito + Lamina



Privacy preserving, location based IDs managed by the user

IoT

Enabling meaningful privacy policies and incentives