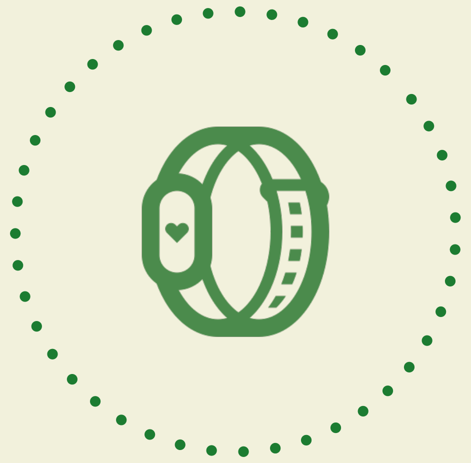
The Internet of Things needs:

Secure Messaging.

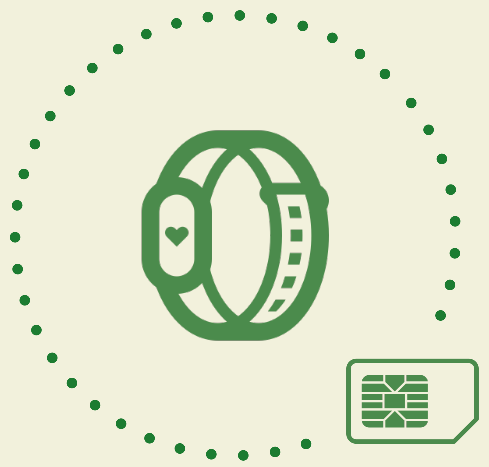
Mrinal Wadhwa
Ockam



Heart Rate Monitor

Heart Rate Application

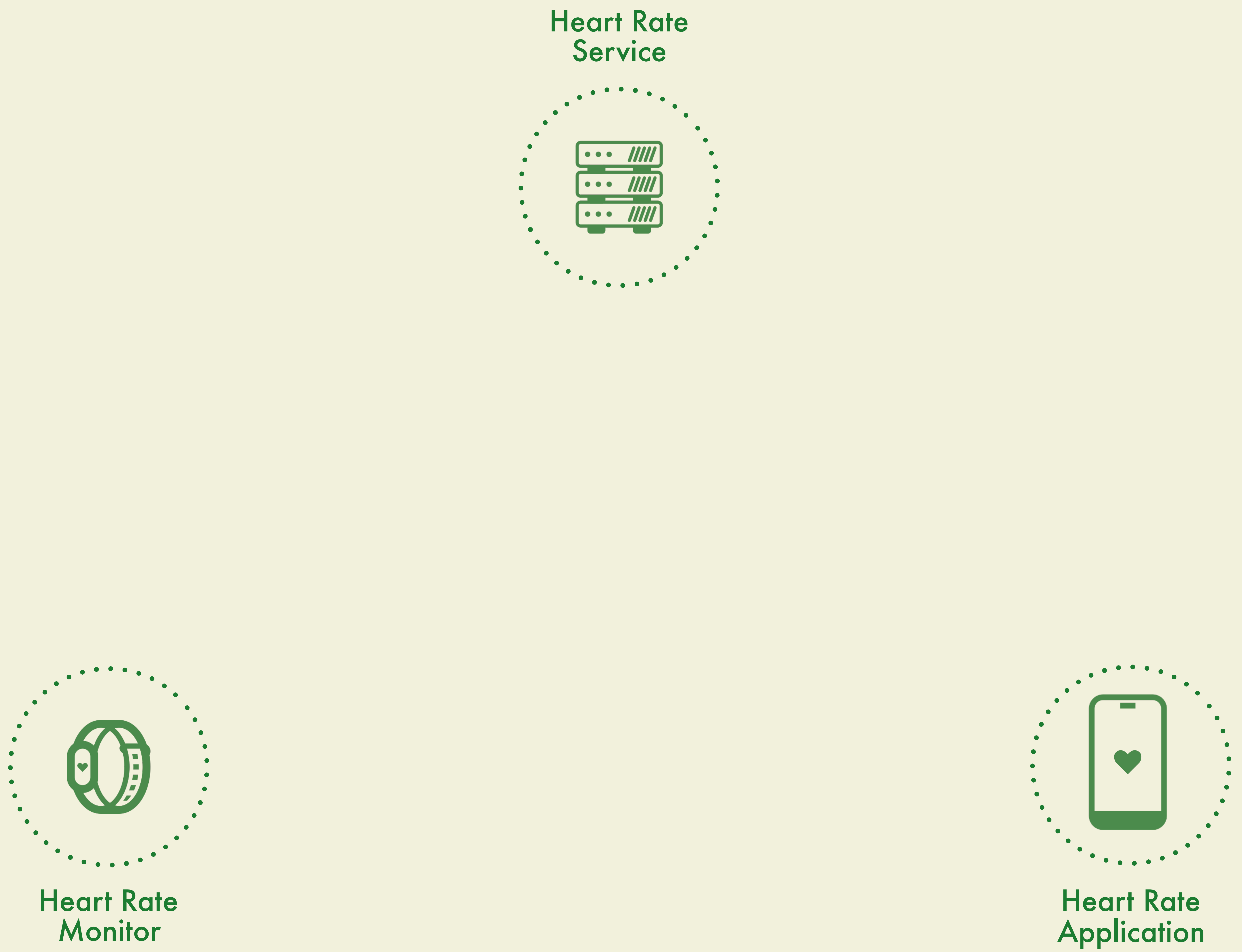
Lets imagine that you’re designing a heart rate monitoring device and an accompanying phone application to track heart rate history …



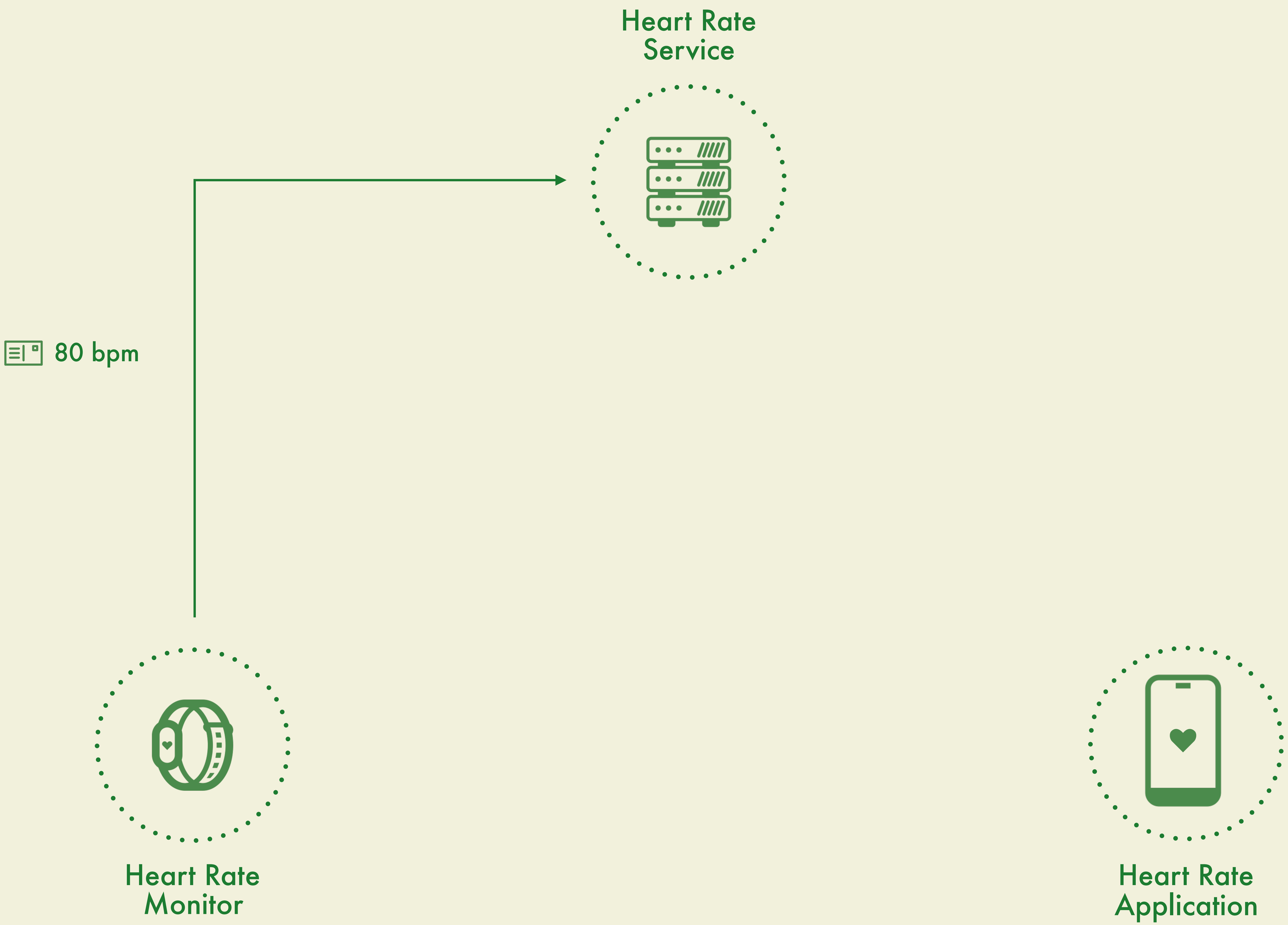
Heart Rate
Monitor

Heart Rate
Application

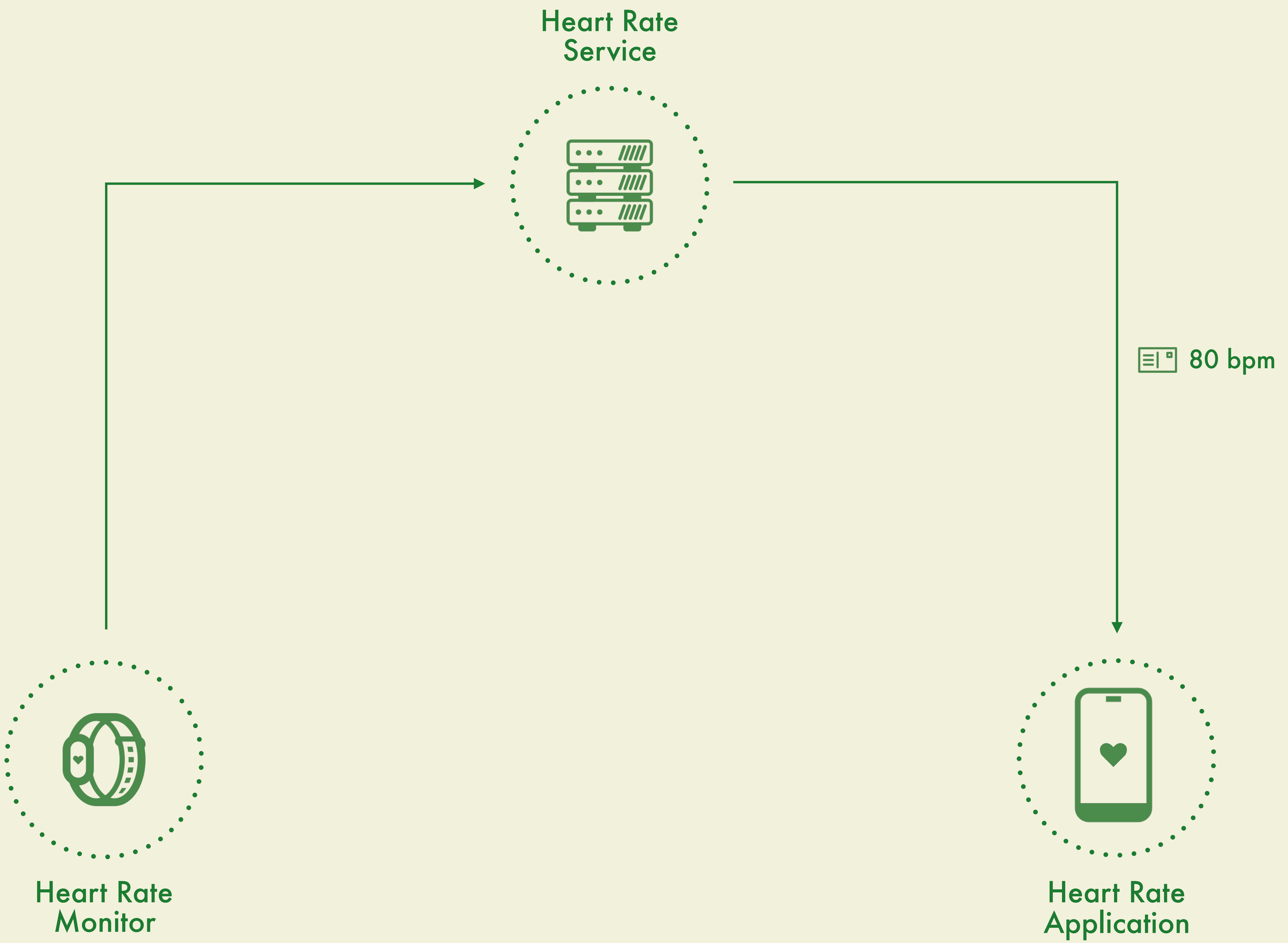
You want the monitor to be usable without having to also carry a phone, so you’ve designed the device to include a cellular modem and it has direct access to the internet ...



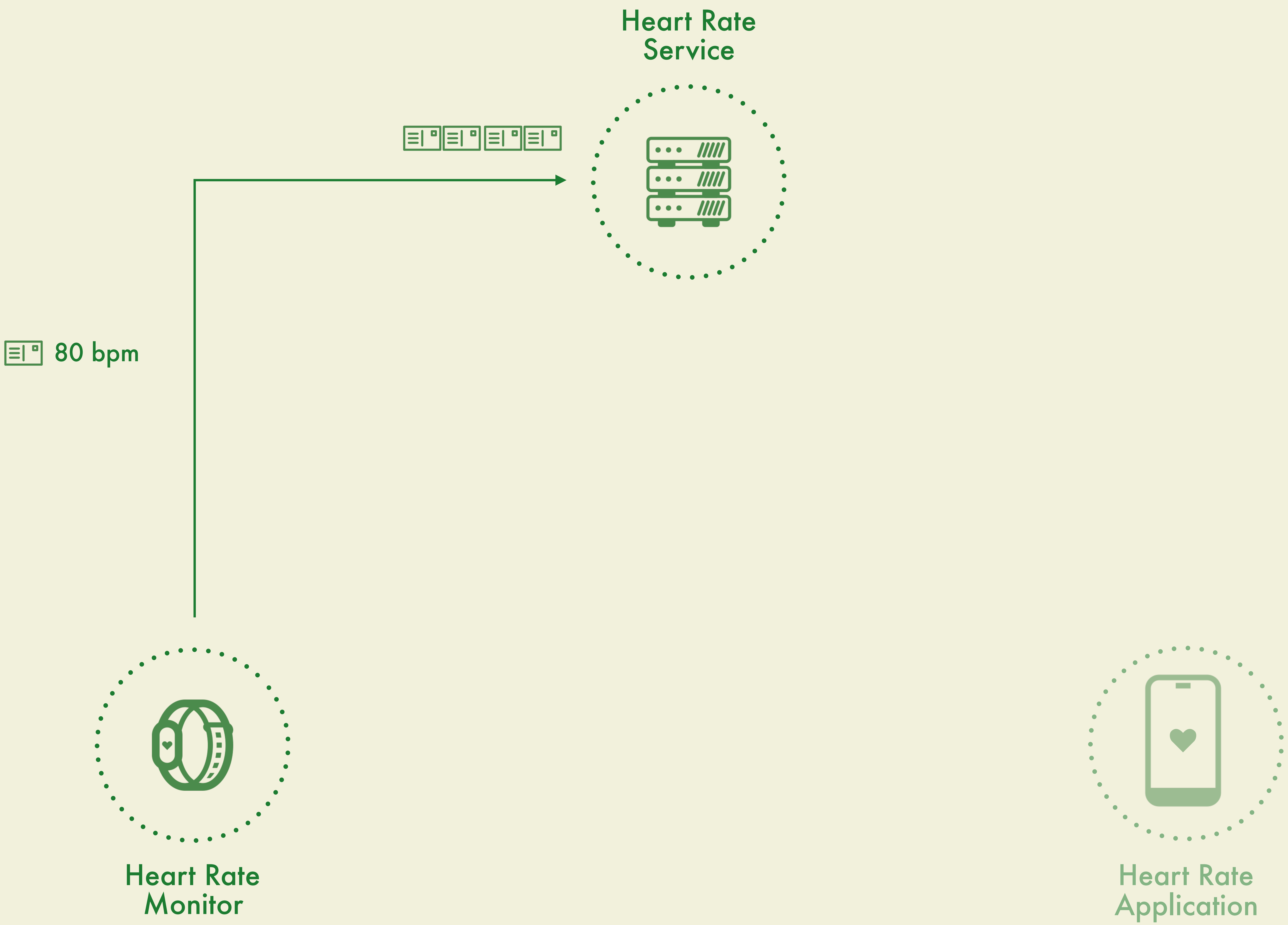
Typically you would setup a web service to deliver heart rate readings to a phone.
Since there is no direct route from the device to the phone.



The monitor would send data to the service …



The service would route the data to the phone ...



The phone may not be online all the time so the service also caches this data to deliver it later ...

Security.

The degree of resistance to encountering an unfortunate event.

Security.

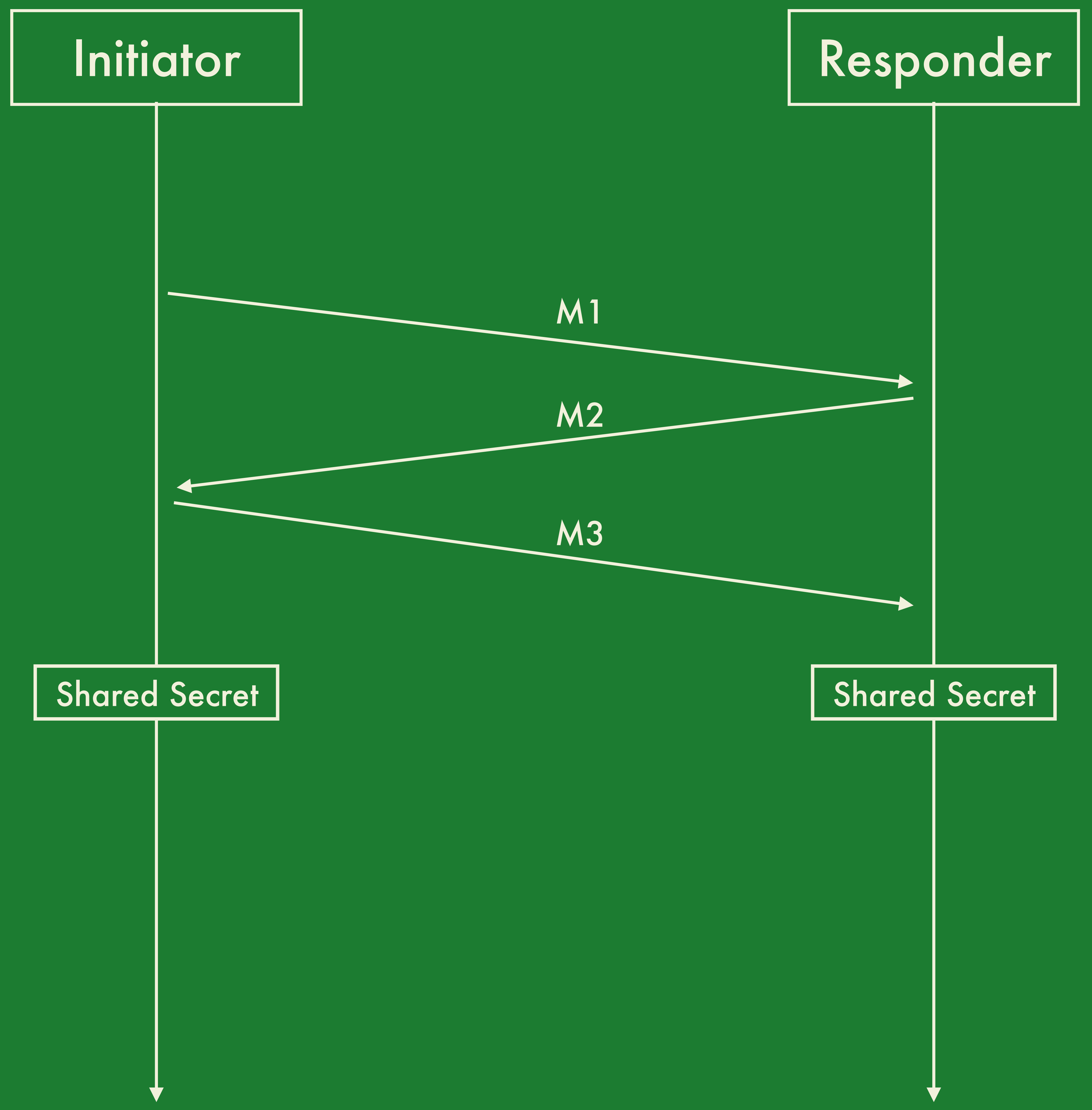
To maximize this degree of resistance, we need to understand the possible set of unfortunate events, <u>the threat model.</u>

The **STRIDE** threat model can help us evaluate every message.

	THREAT	**DESIRED PROPERTY**
S	Spoofing identity	Identification, Authentication
T	Tampering with data	Integrity
R	Repudiation	Non-repudiability (some applications desire the opposite)
I	Information disclosure	Confidentiality
D	Denial of service	Availability
E	Elevation of privilege	Authorization

Note that this model is **very** high level, there is massive amounts of nuance in dealing with each of the rows.

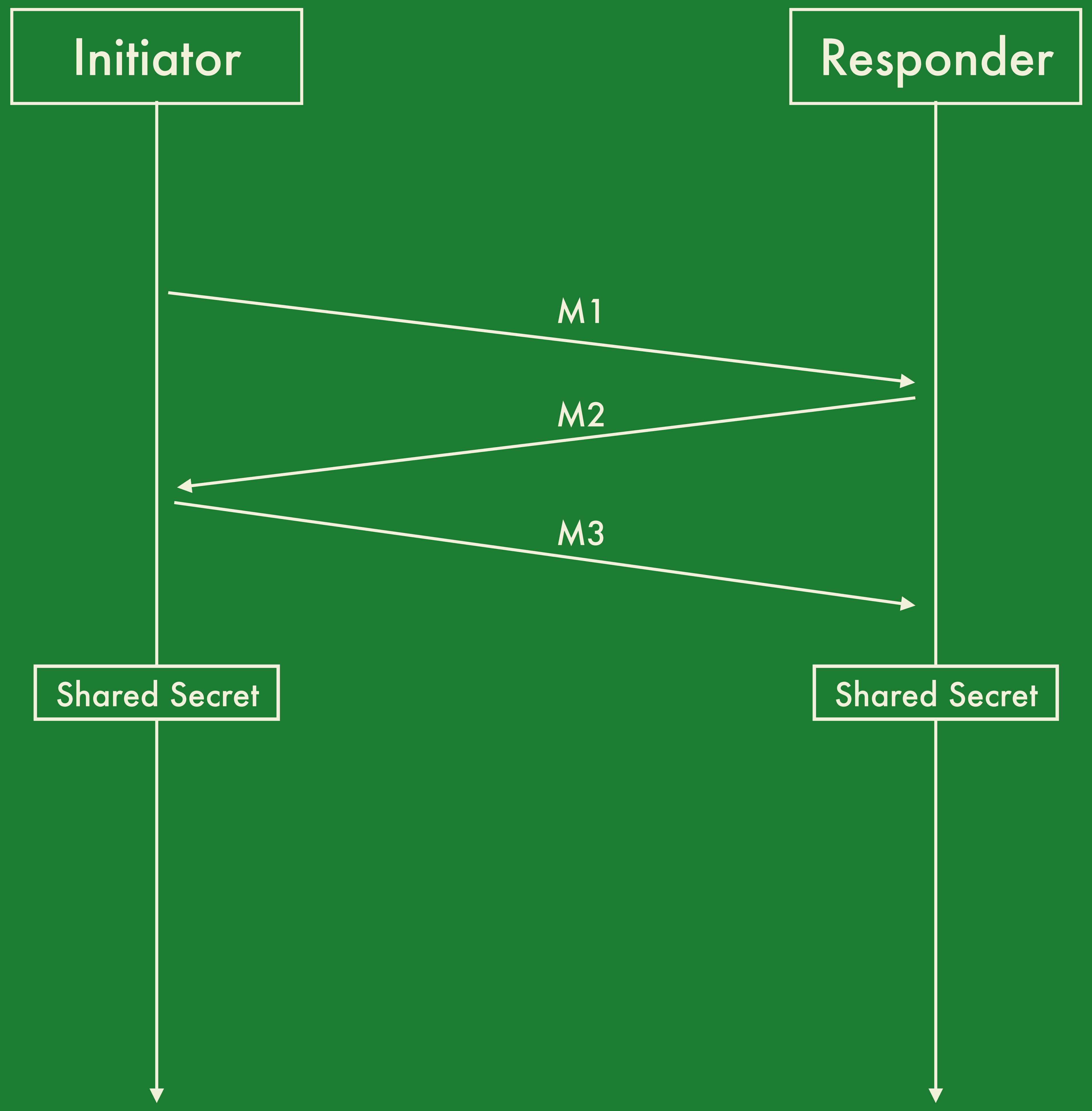
Secure Channels.



The typical approach used to achieve these desired properties.

Authenticated Key Exchange

The entities involved use Public Key Cryptography to authenticate each other and agree on a shared secret.

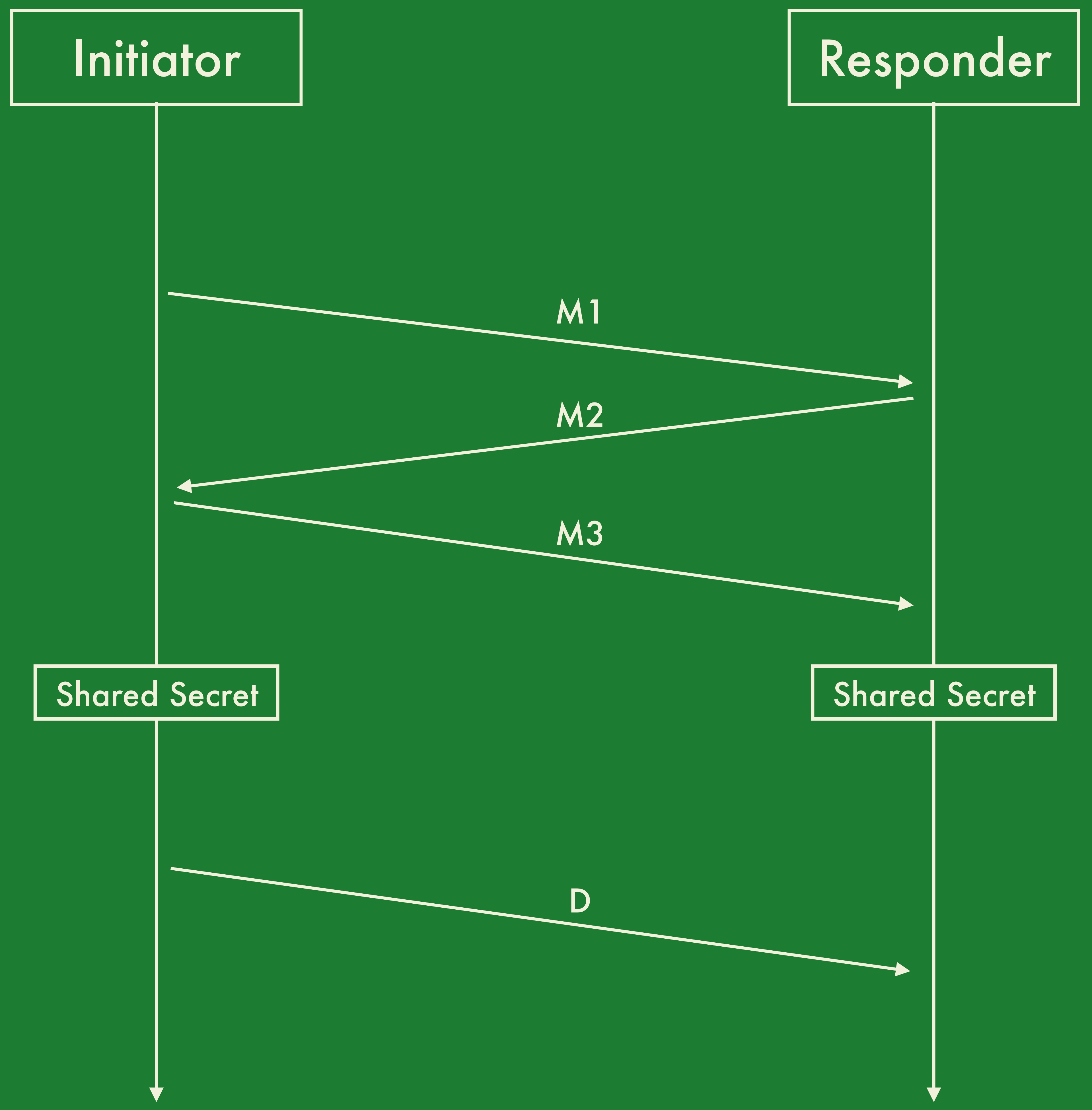


Authenticated Key Exchange

The entities involved use Public Key Cryptography to authenticate each other and agree on a shared secret.

Application Data - Authenticated Encryption

The shared secret is then used as a key in Symmetric Key Cryptography to maintain confidentiality and integrity of application data.



The **STRIDE** threat model.

	THREAT	DESIRED PROPERTY
S	Spoofing identity	Identification, Authentication
T	Tampering with data	Integrity
R	Repudiation	Non-repudiability (some applications desire the opposite)
I	Information disclosure	Confidentiality
D	Denial of service	Availability
E	Elevation of privilege	Authorization

Even when information disclosure is not in a system's threat model, all of the other rows must be, else that system has no security or reliability.
You still need secure channels even when you aren't looking for confidentiality/encryption.

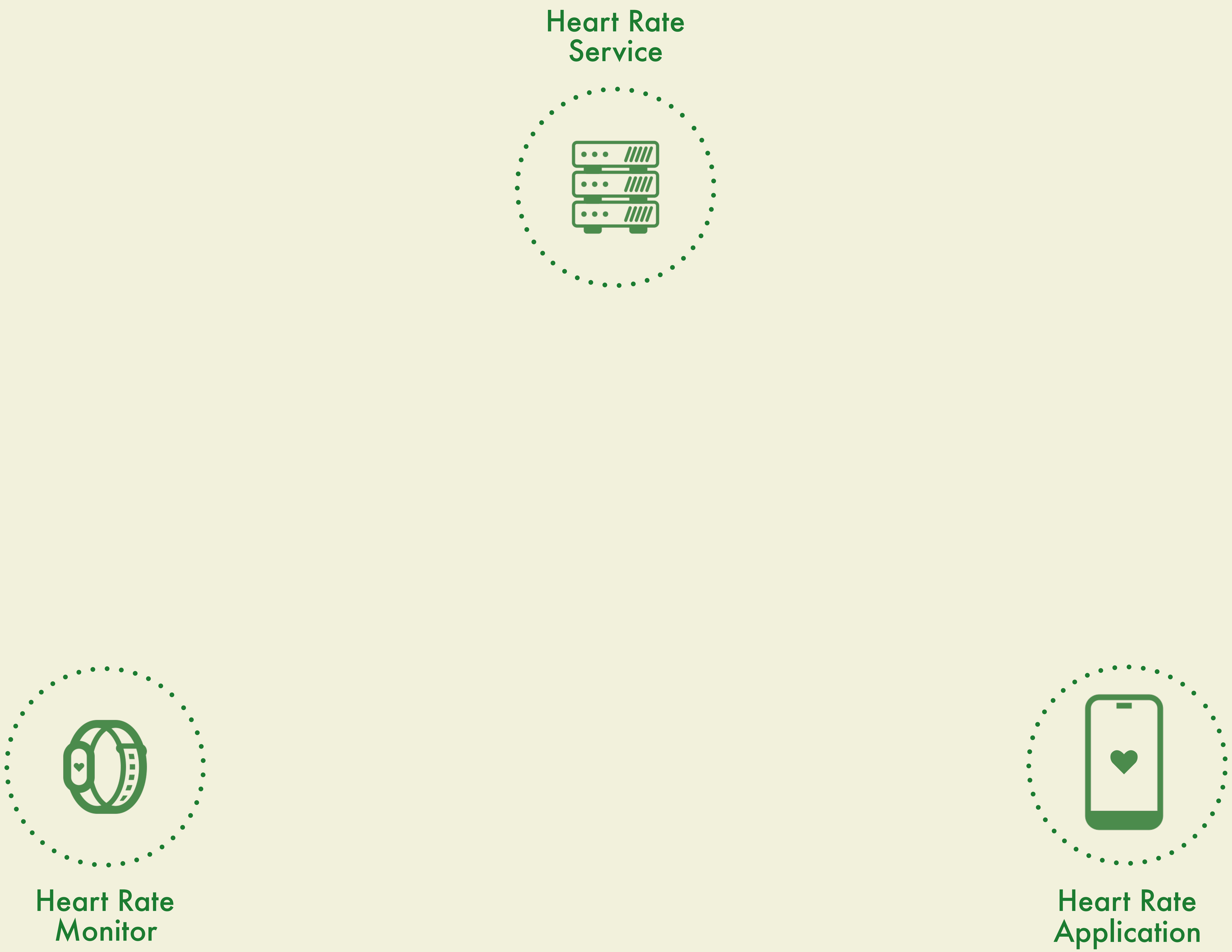
Implementing secure channels correctly is hard:

1. RSA or Elliptic Curves?
2. Which Curve to use? P256, P512, Brainpool, Kolbitz, Curve25519, Curve448 ...
3. Which HASH algorithm to use? SHA1, SHA2, SHA3, Blake2 ...
4. Which MAC algorithm to use? HMAC, GMAC, CMAC, PMAC ...
5. Which AEAD? AES_GCM, ChaChaPoly ...
6. Which Key derivation function?
7. Nonces, uniqueness, nonce length?
8. Which AES mode? AES CTR, CBC, GCM, GCM-SIV, SIV, CCM ...
9. Authenticated Key Exchange? Diffie-Helmann only or Signatures + Diffie-Helmann
10. How to protect against downgrade attacks?
11. How to guarantee Forward Secrecy?
12. How to resist Key Compromise Impersonation attacks?
13. How to protect identities?

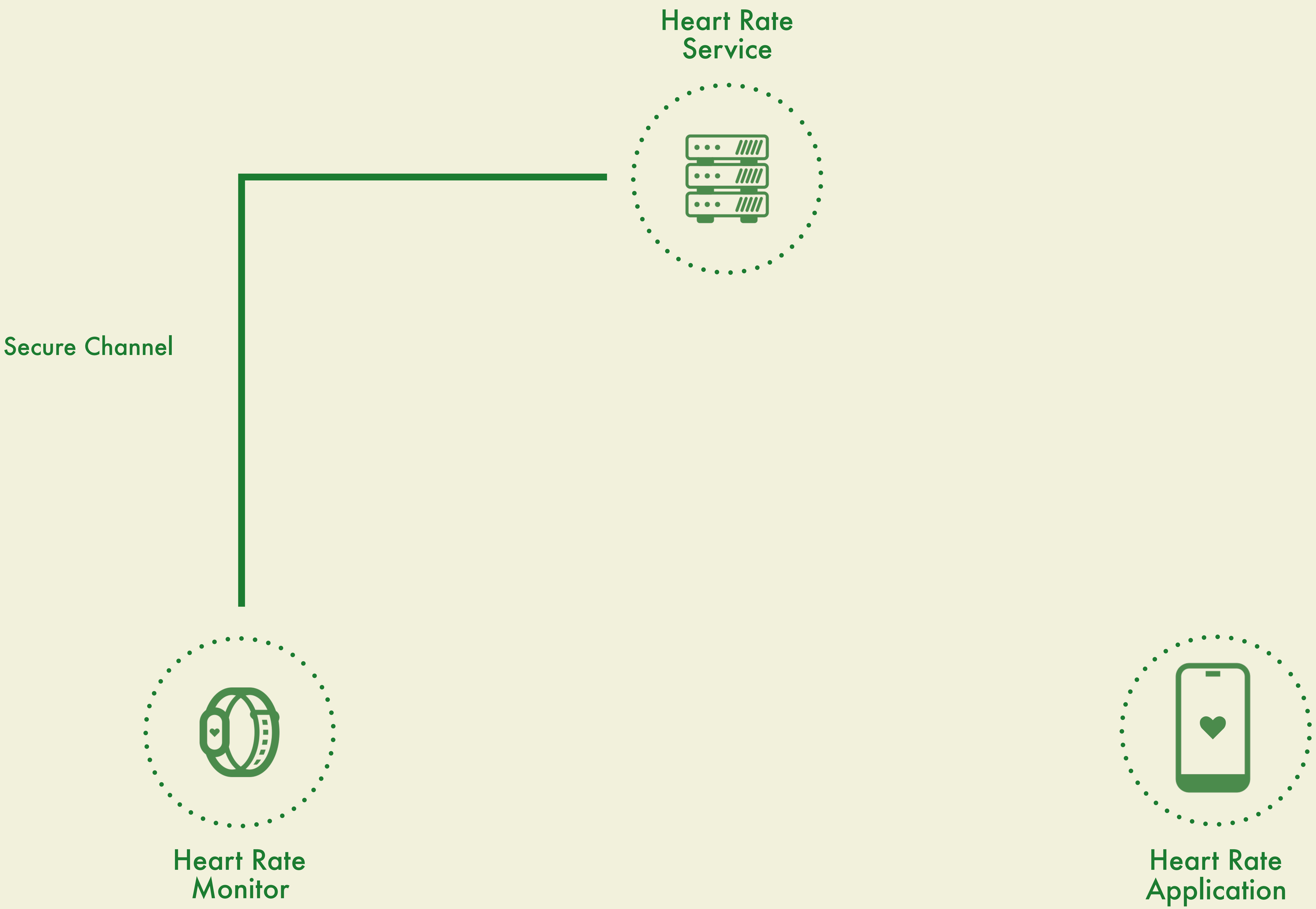
Many transport protocols, that are commonly used within IoT systems, provide some notion of a secure channel.

However, configuring them correctly is hard and their security guarantees can vary in many subtle ways based on how the channel protocol was designed (all the choices from the previous slide)

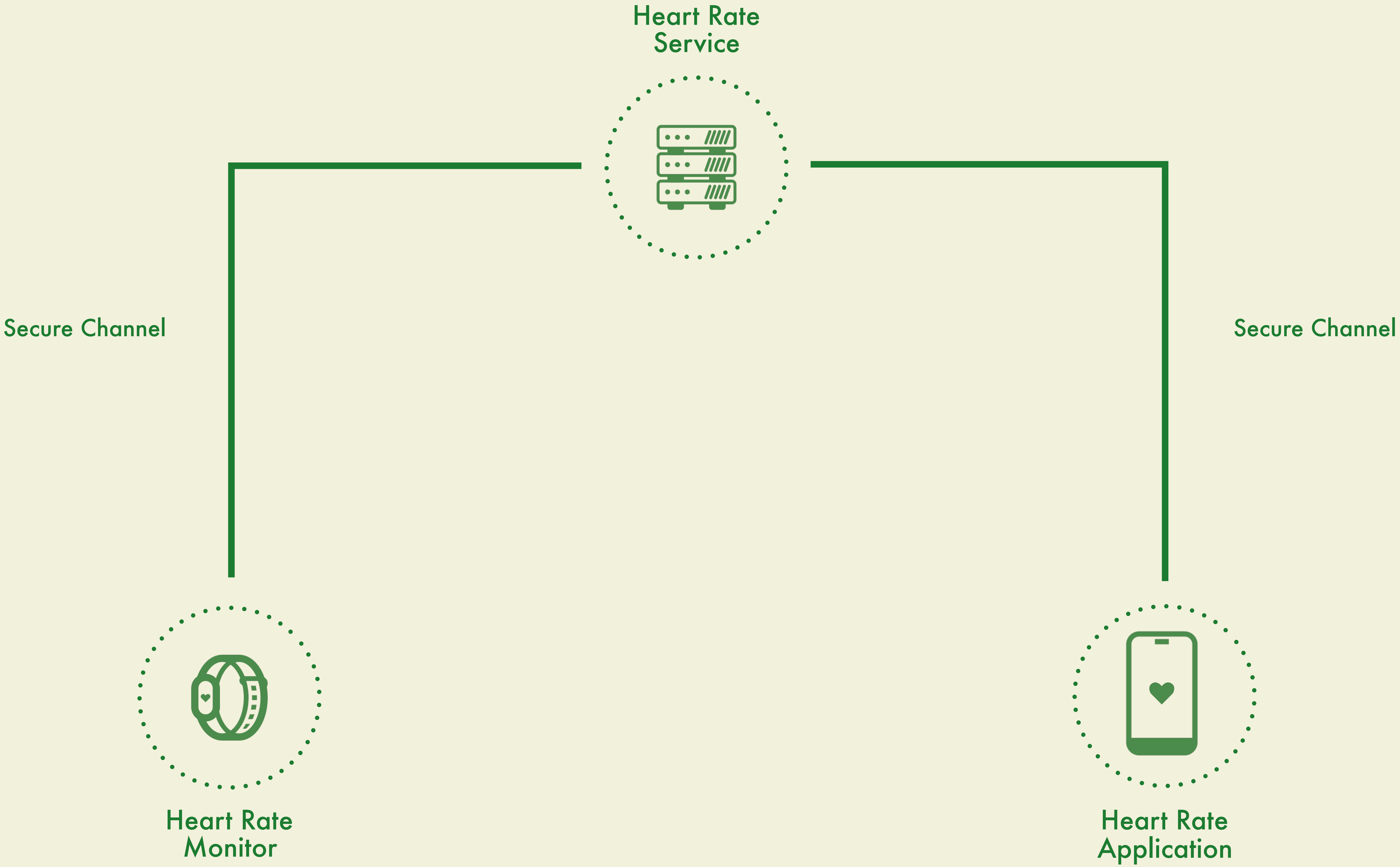
This complexity makes it very challenging for system architects to reason about the overall of security of our systems.



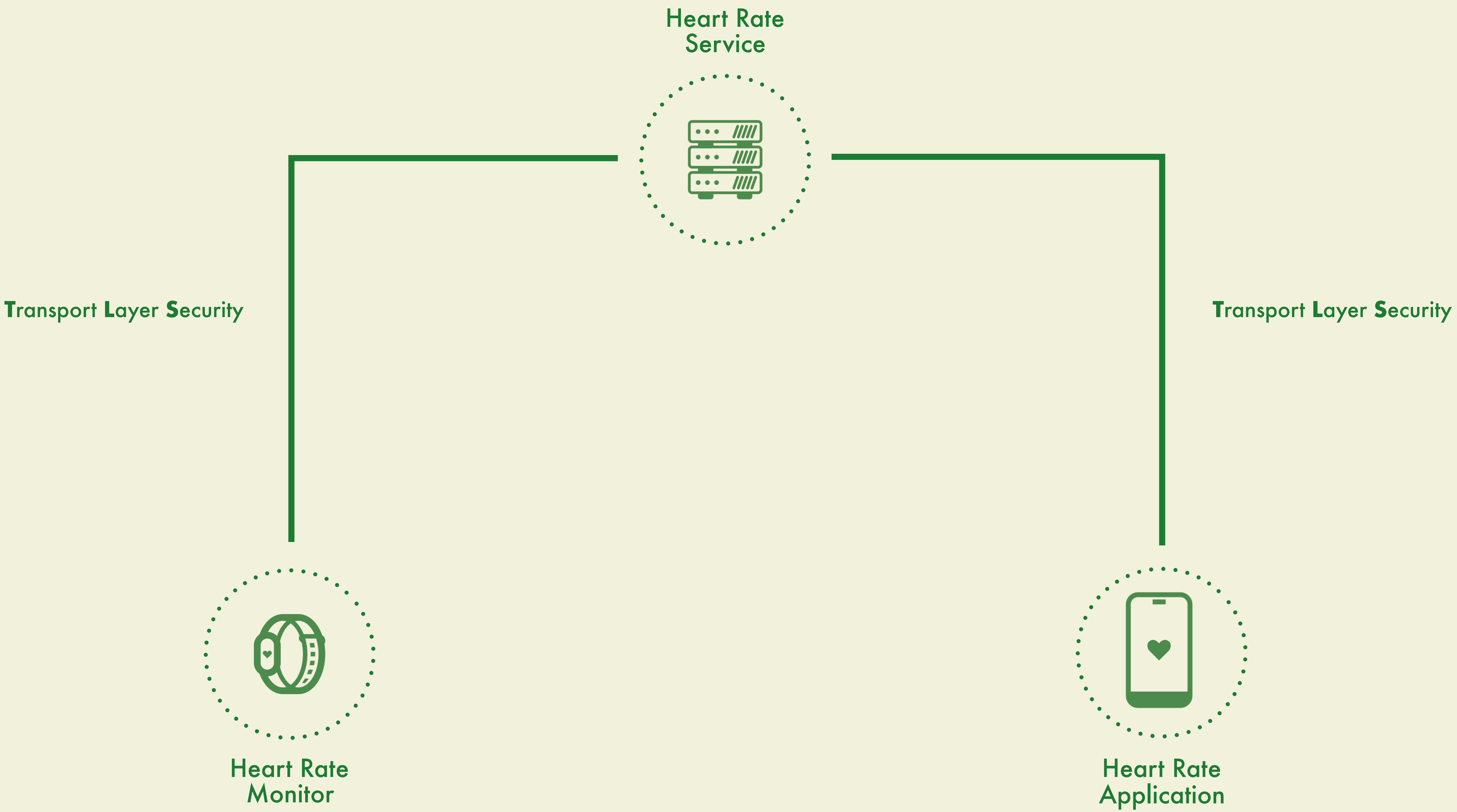
Coming back to our heart rate solution, for secure communication ...



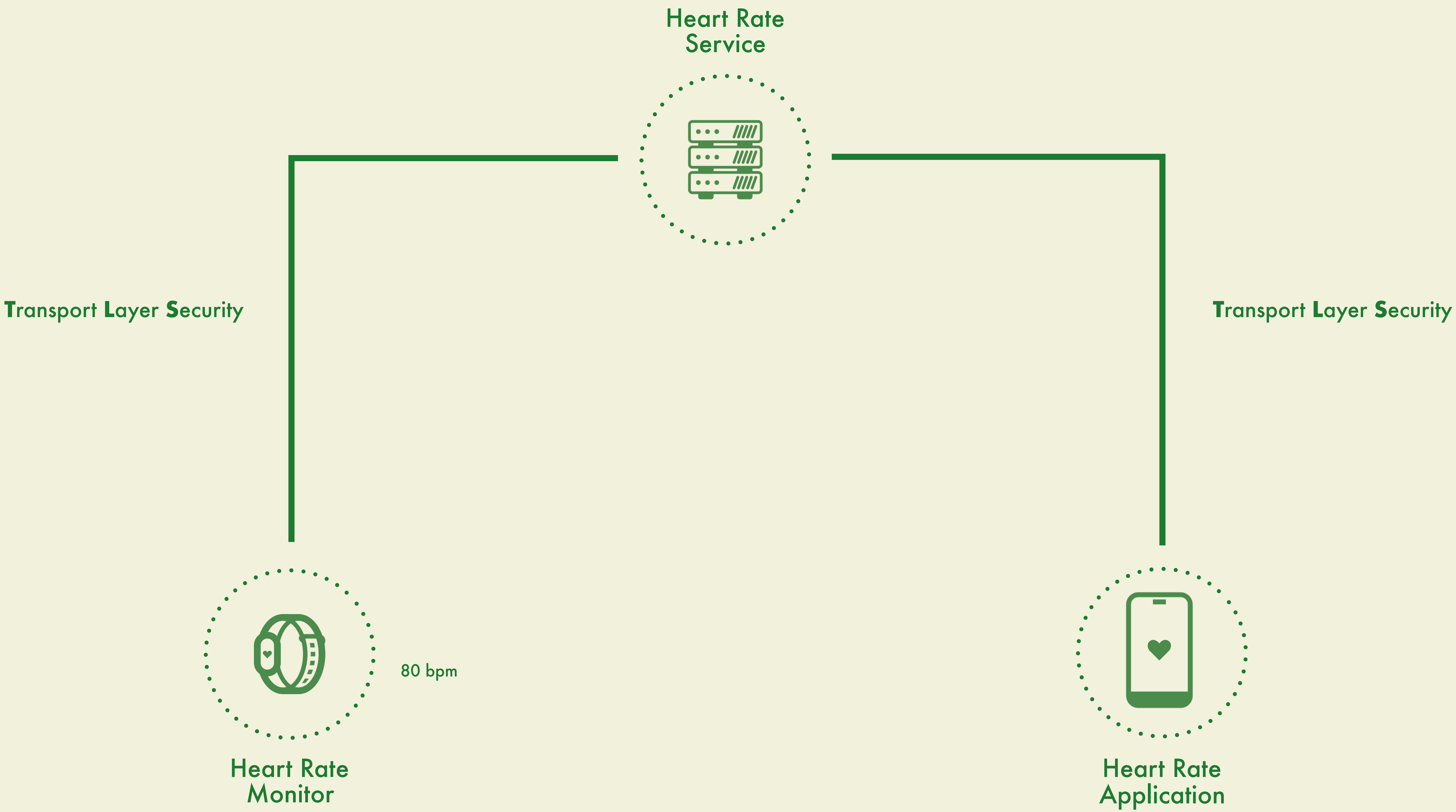
We setup a secure channel between the monitor and the service.

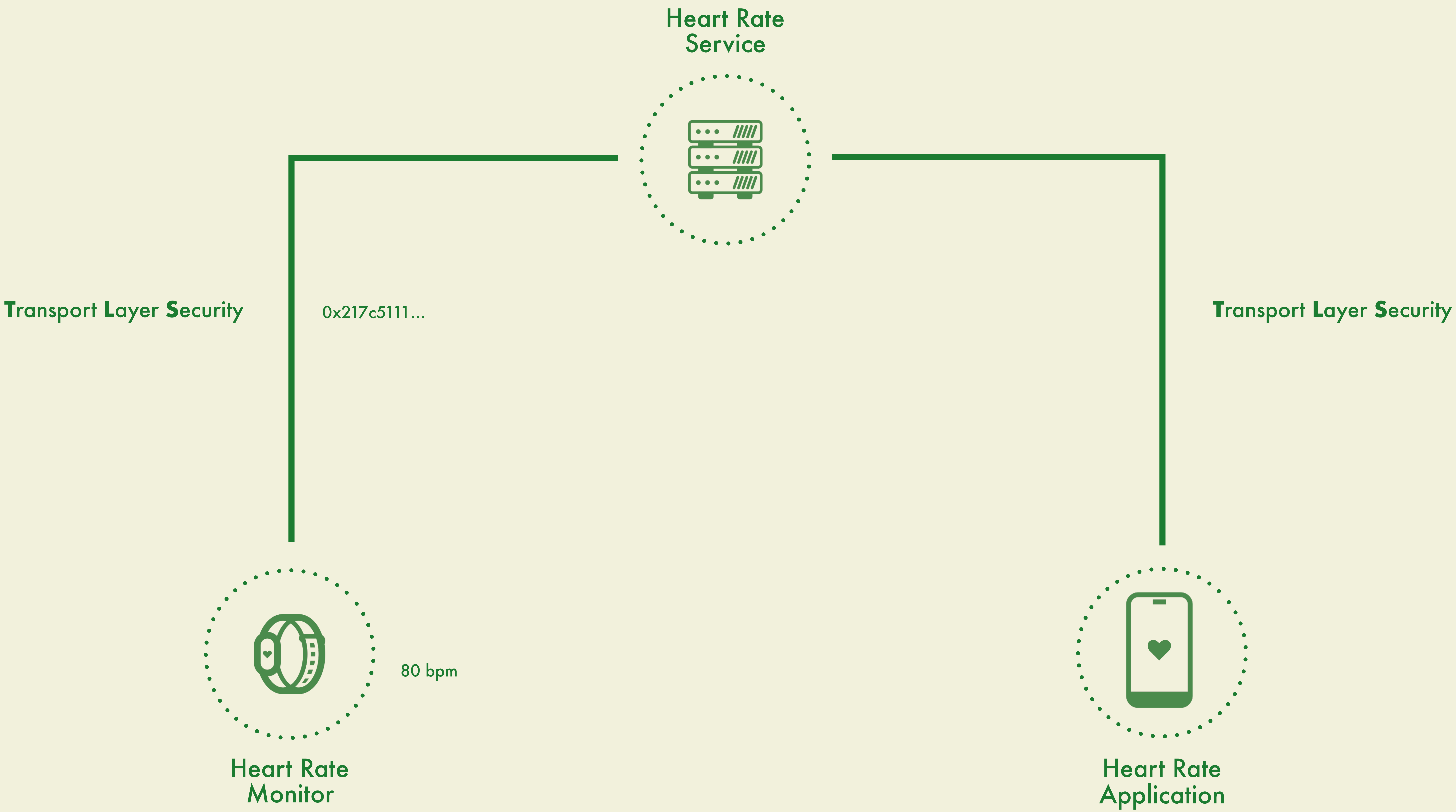


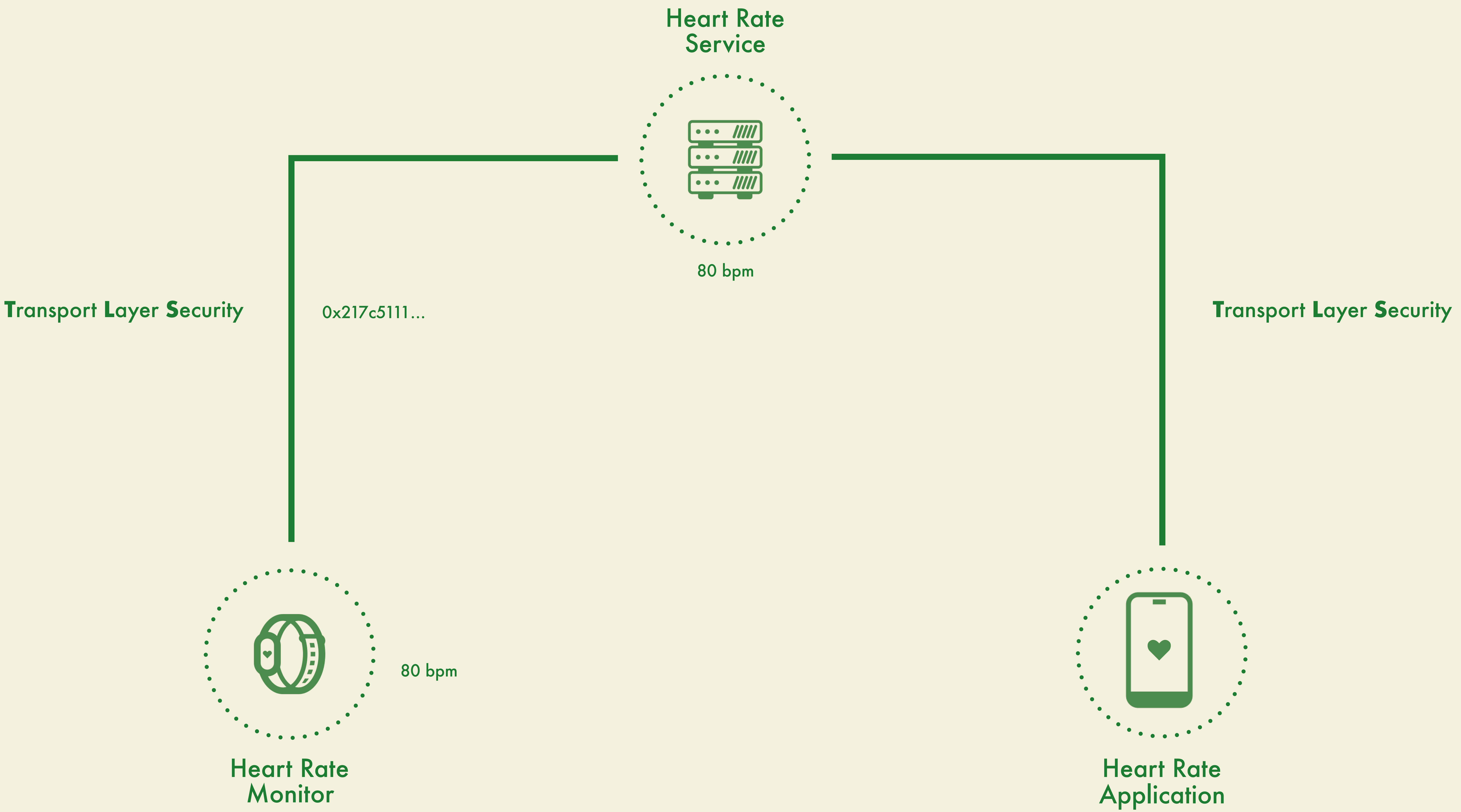
And another secure channel between the phone and the service.

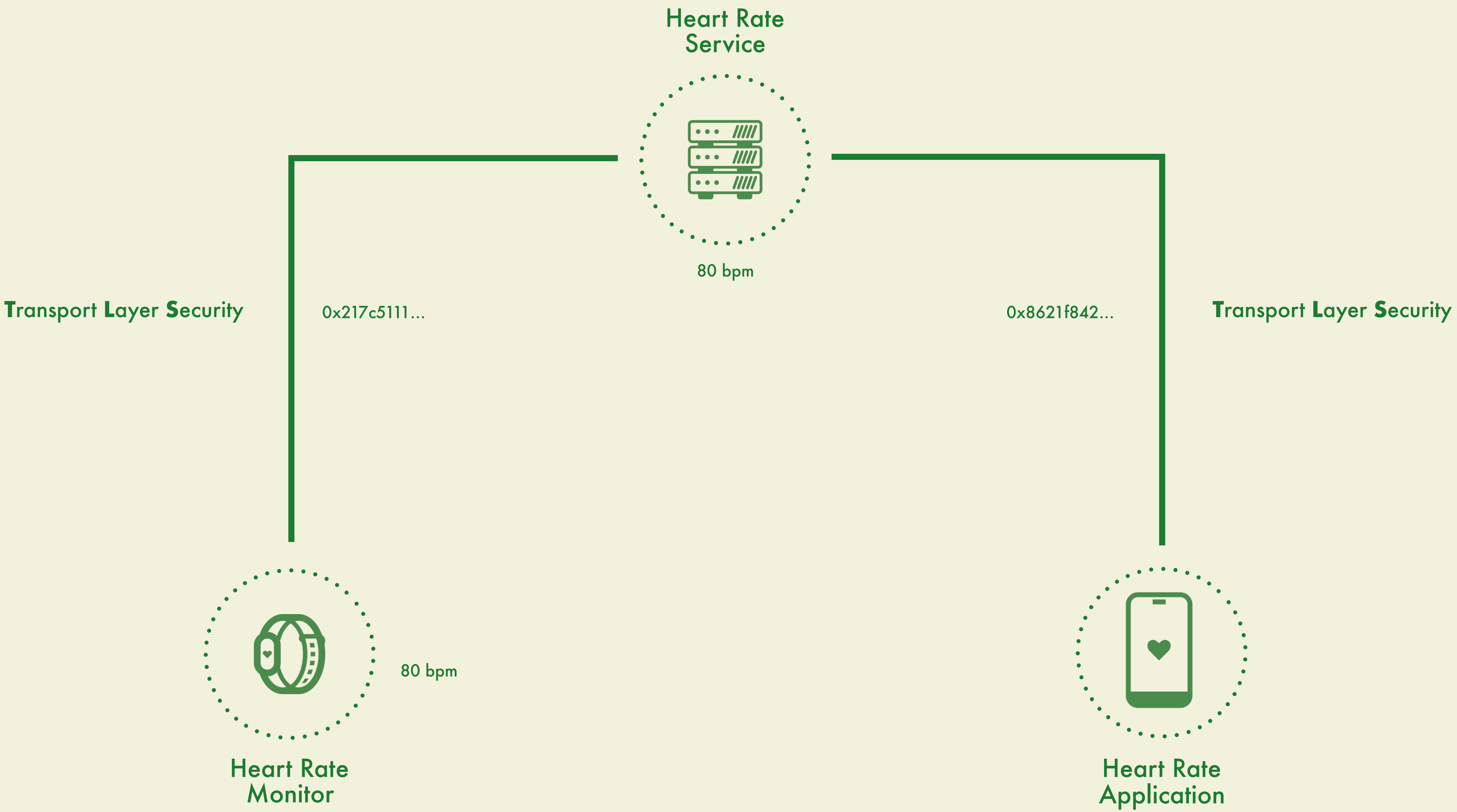


Since these devices have direct access to the internet, with TLS ...

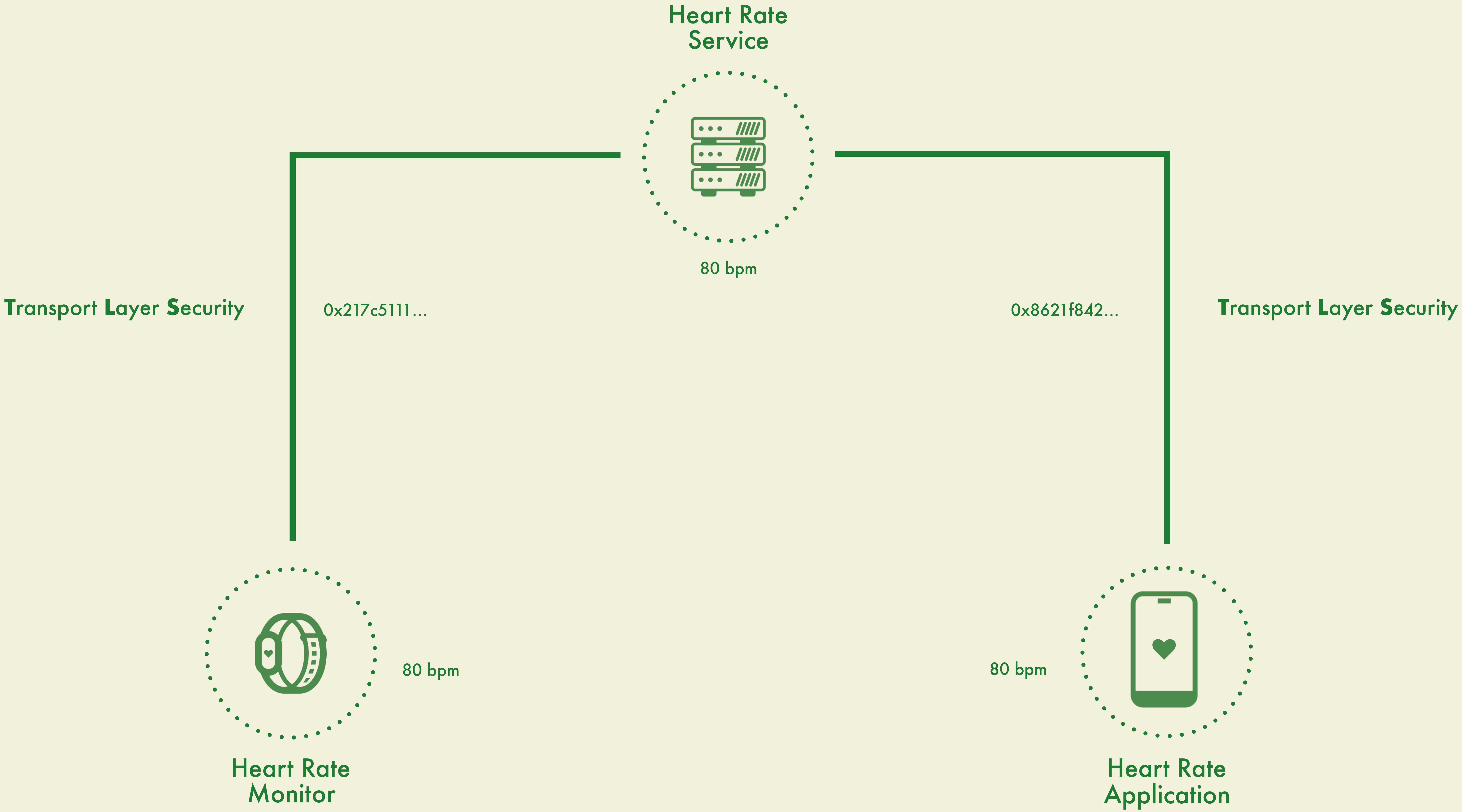

Heart Rate
Service
Transport Layer Security
Transport Layer Security
80 bpm
Heart Rate
Monitor
Heart Rate
Application


Heart Rate
Service
Transport Layer Security
0x217c5111...
Transport Layer Security
80 bpm
Heart Rate
Monitor
Heart Rate
Application

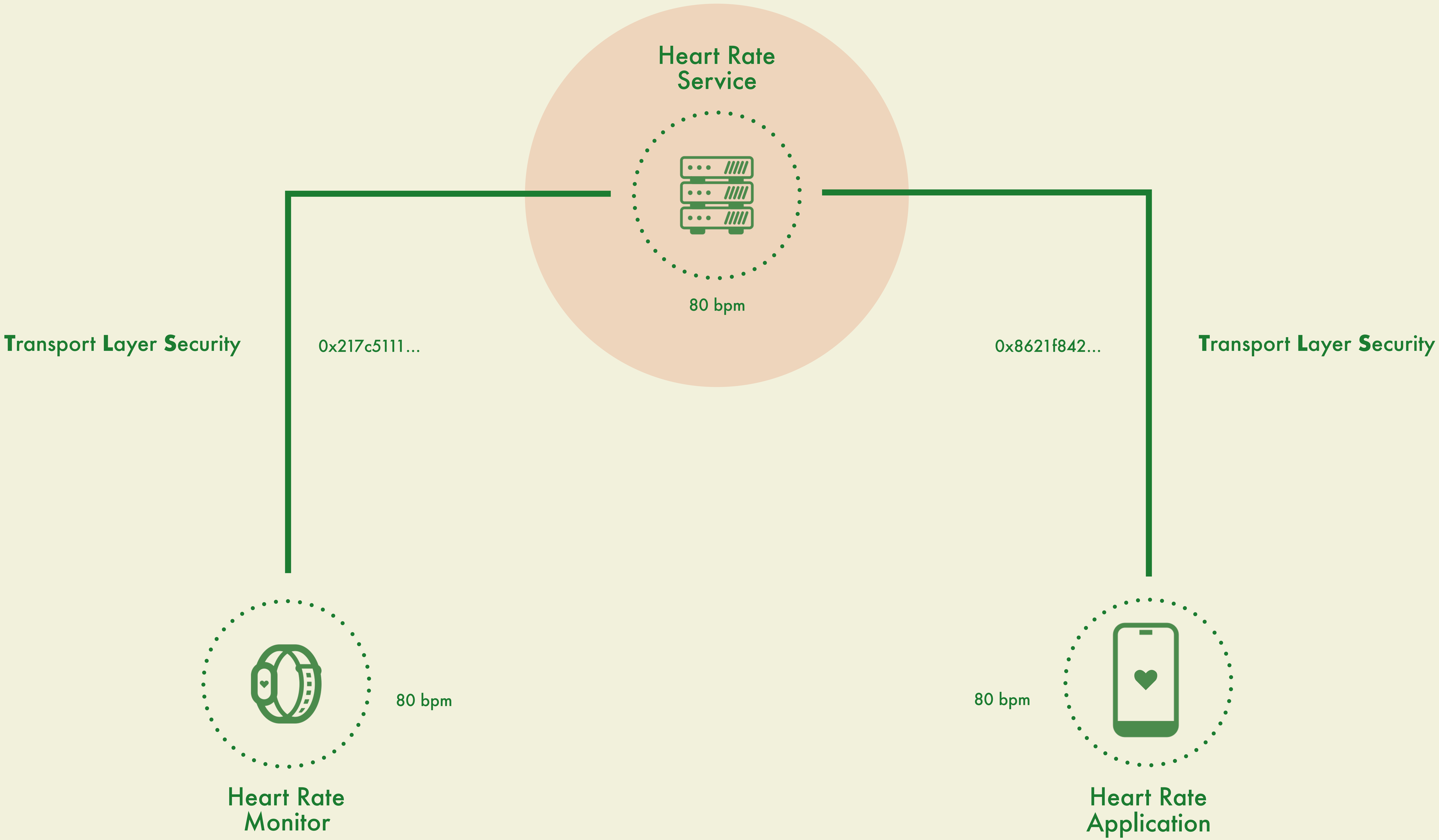

Heart Rate Service
80 bpm
Transport Layer Security
0x217c5111...
Transport Layer Security
80 bpm
Heart Rate Monitor
Heart Rate Application



Heart Rate Service
80 bpm
Transport Layer Security
0x217c5111...
0x8621f842...
Transport Layer Security
80 bpm
Heart Rate Monitor
Heart Rate Application



This type of setup is industry best practice.



But even when we manage to setup the channels correctly the data is still exposed to the service.
The service doesn’t need to know the contents of the message to route and cache messages (its primary job).

Principle of

Least Privilege.

"Every program and every privileged user of the system should operate using the least amount of privilege necessary to complete the job."

– Jerome Saltzer, Communications of the ACM, 1974

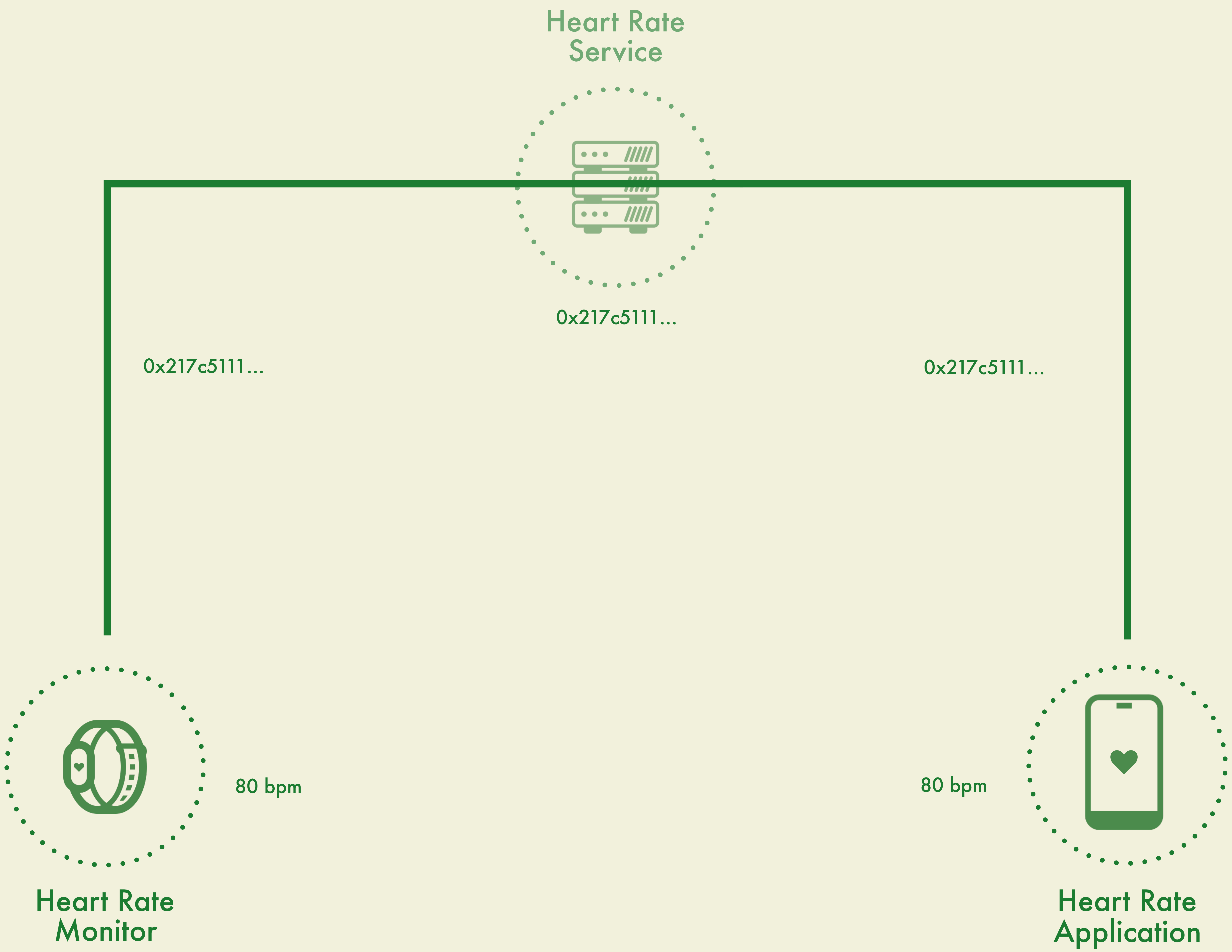
Exposing the data to the service increases the attack surface of the system, creates a honeypot of data and exposes the service operator to liability, risk, and compliance challenges (HIPPA, GDPR, CCPA ...).

Privacy.

The ability of an individual or group to control the flow of information about themselves.

The data being exposed to the service also takes away the end user's control on their data.

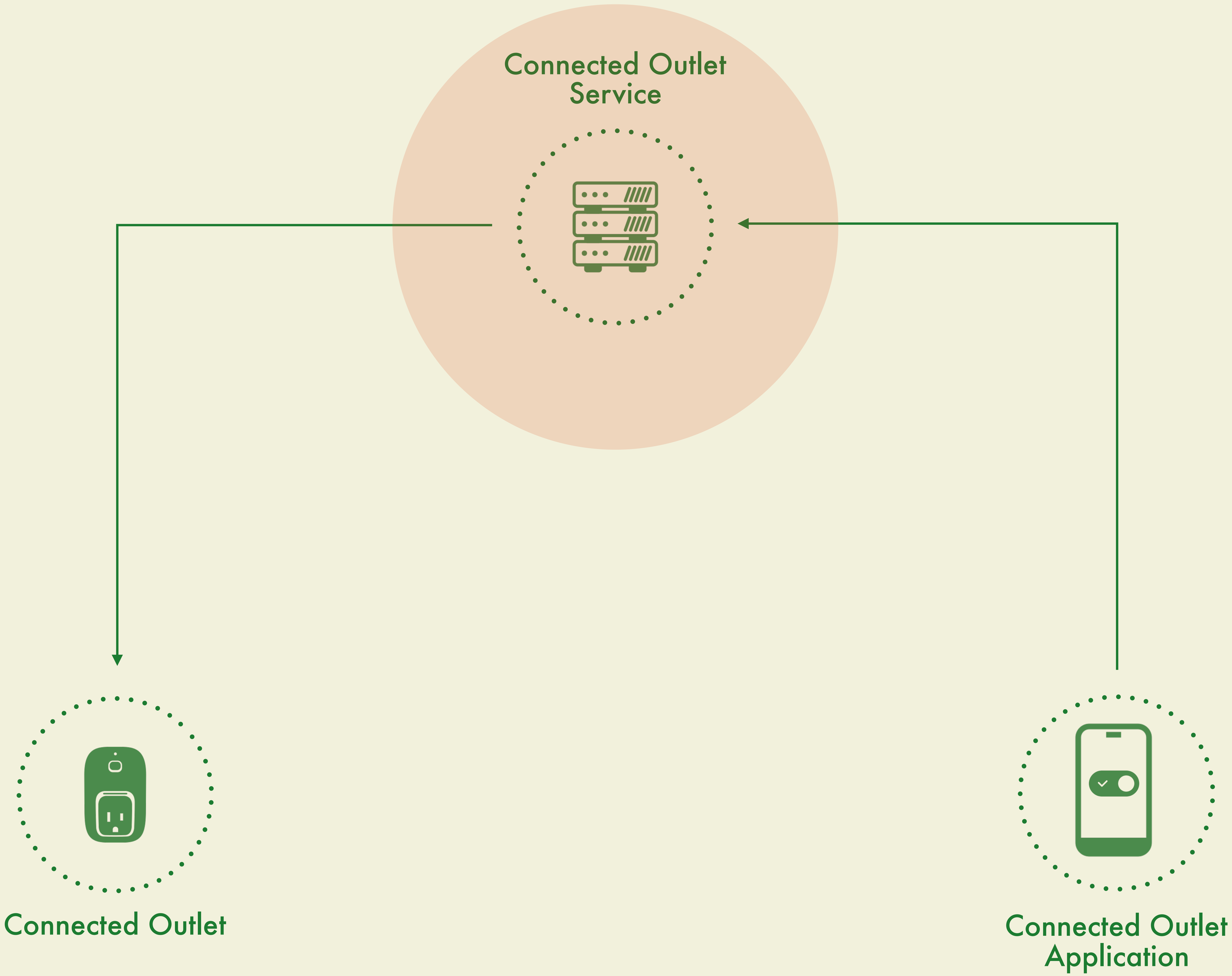
This user could be private data of an individual or proprietary data od a business)



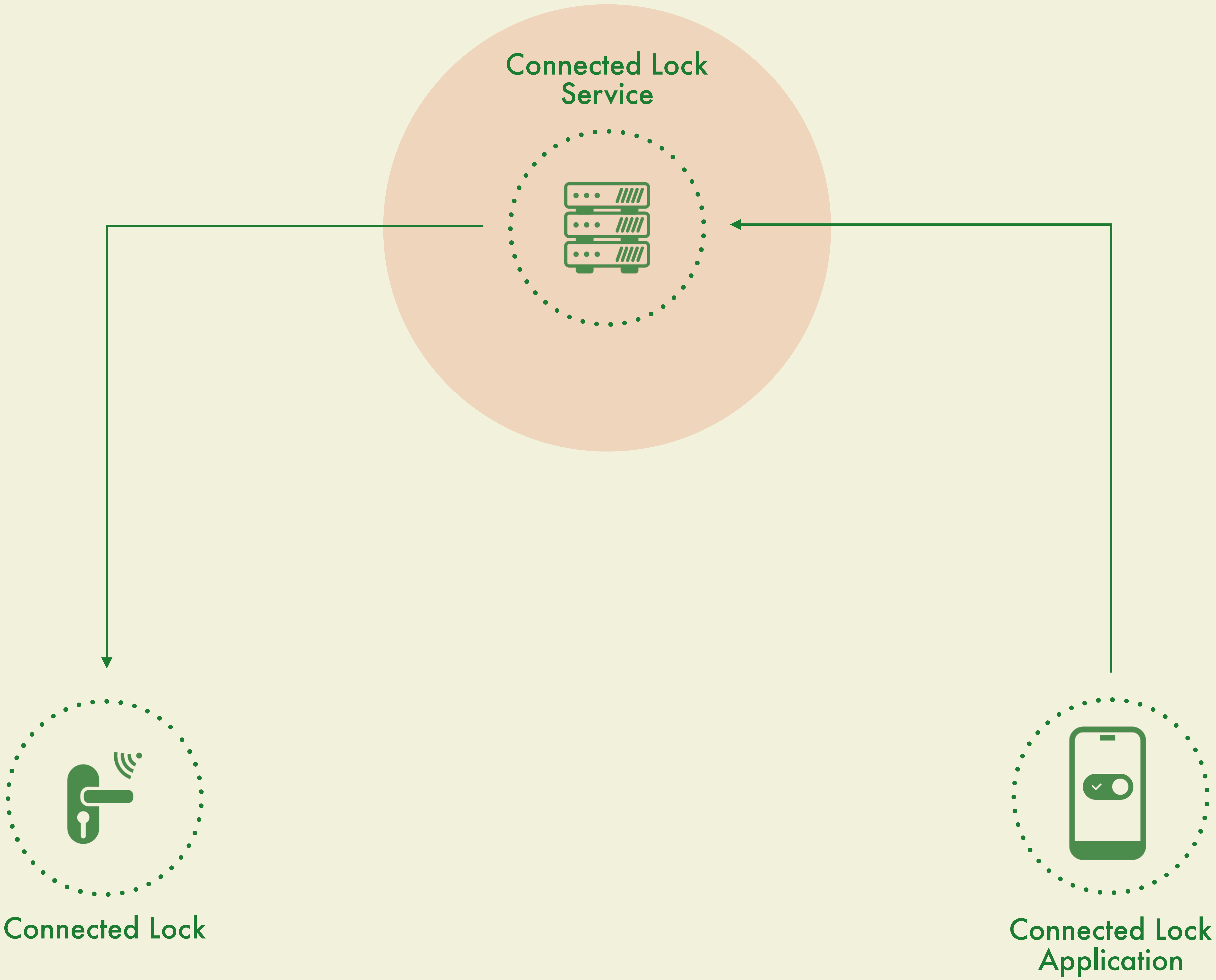
If, instead, we **decouple** the secure channel protocol from the transport protocol, we could have an end-to-end secure and private channel.

Decoupling the secure channel protocol from the transport layer protocols removes complexity, minimizes the attack surface and can enable us to build end-to-end secure and private systems.

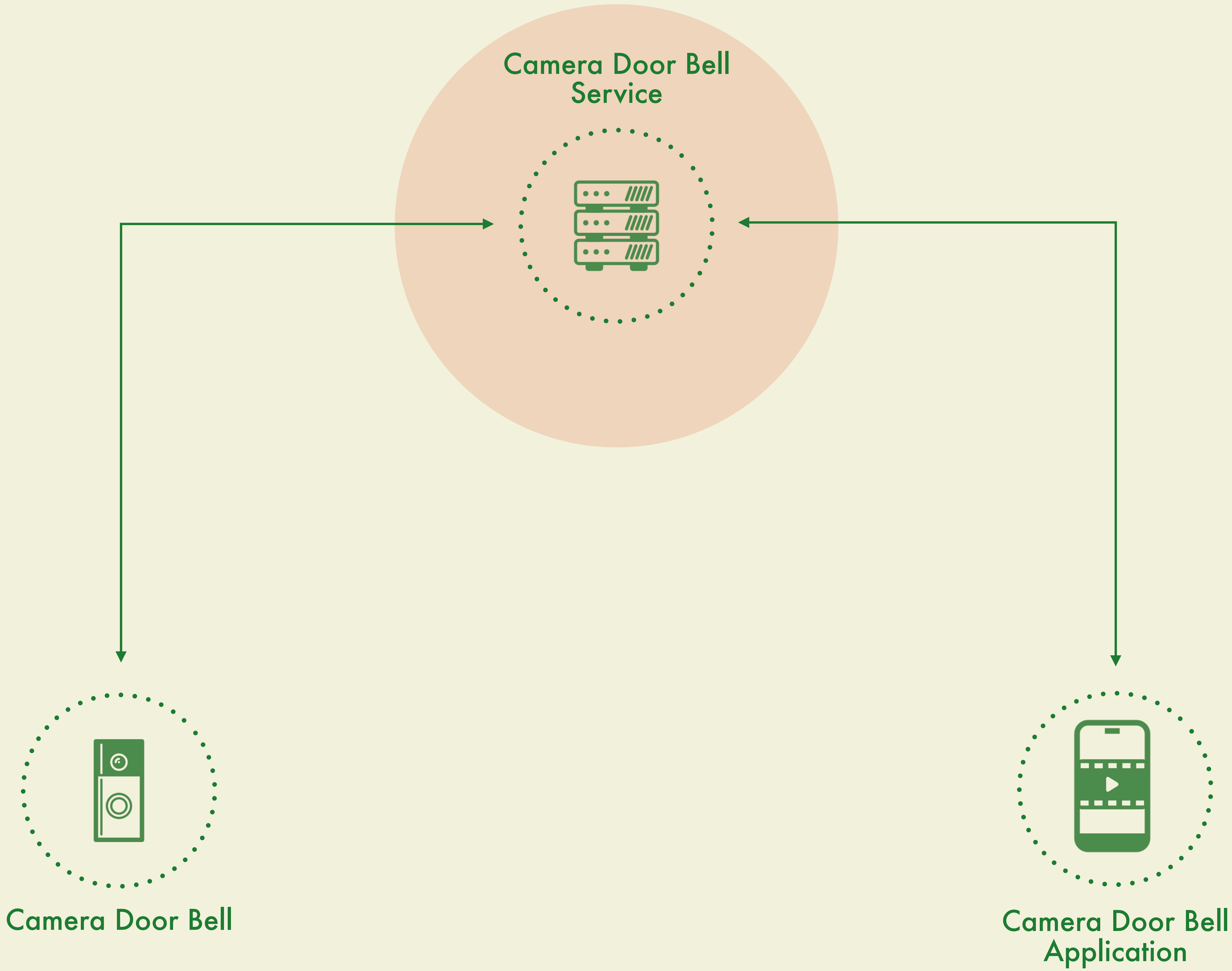
Lots of connected devices only need an internet service for routing and caching ...



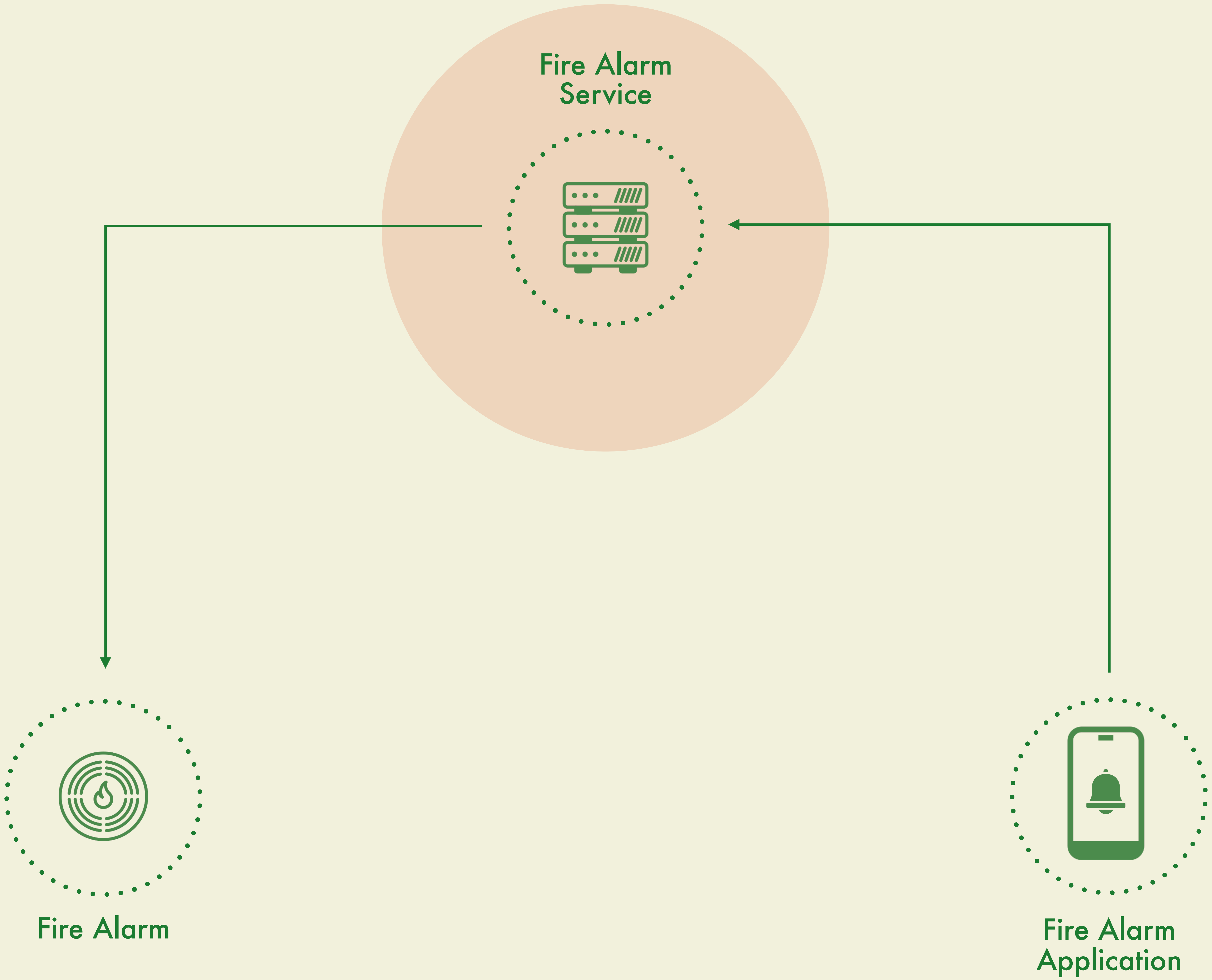
Route on/off instructions.



Route open/close instructions.



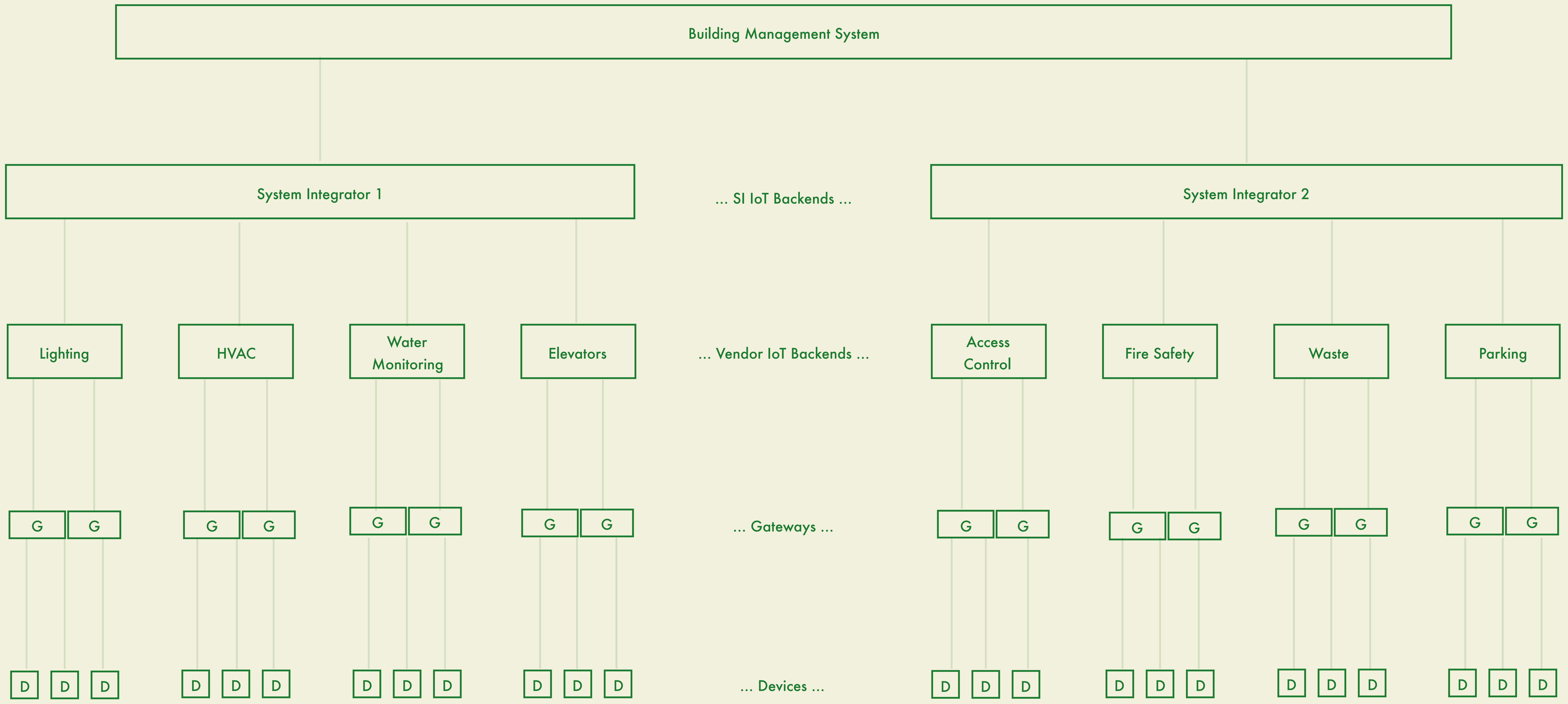
Route/Cache sensor data, alerts and videos.



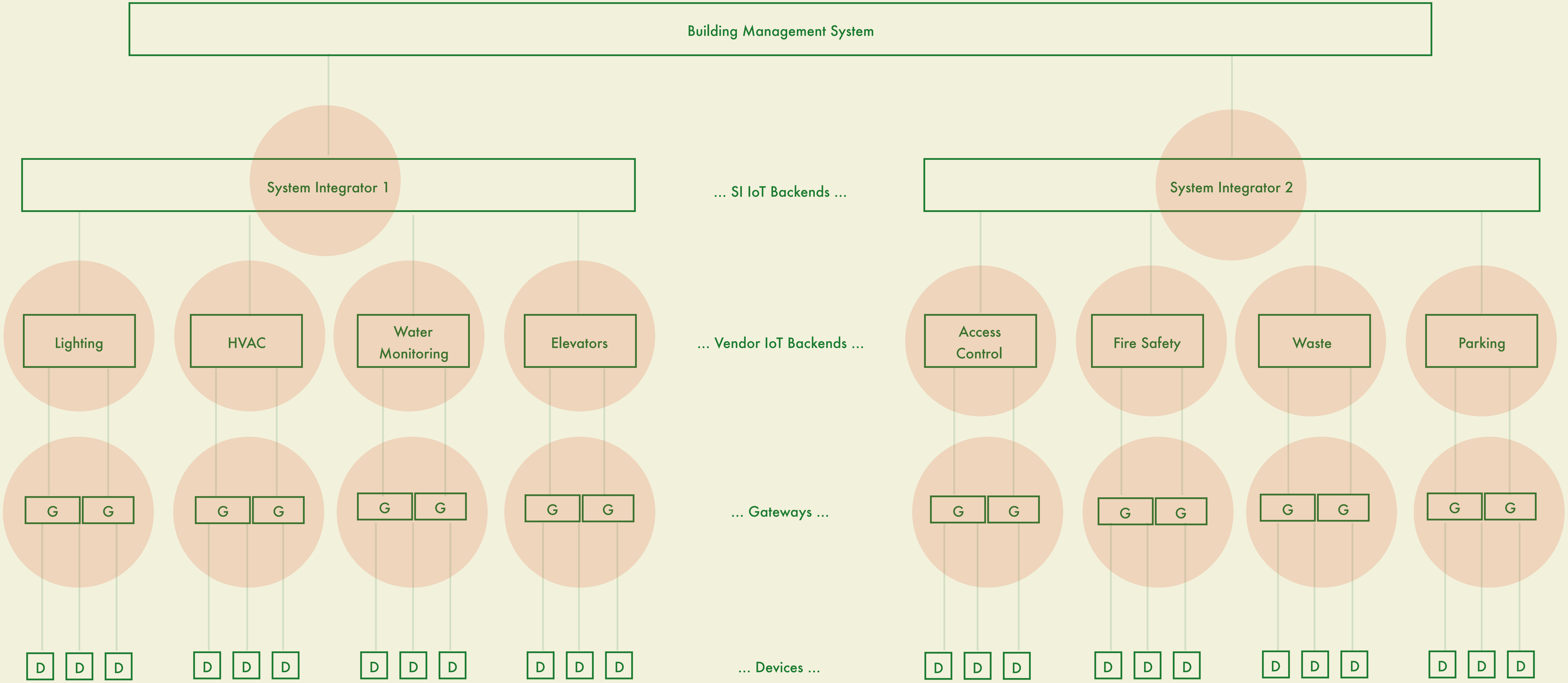
Sensor Alerts

Many, many more.

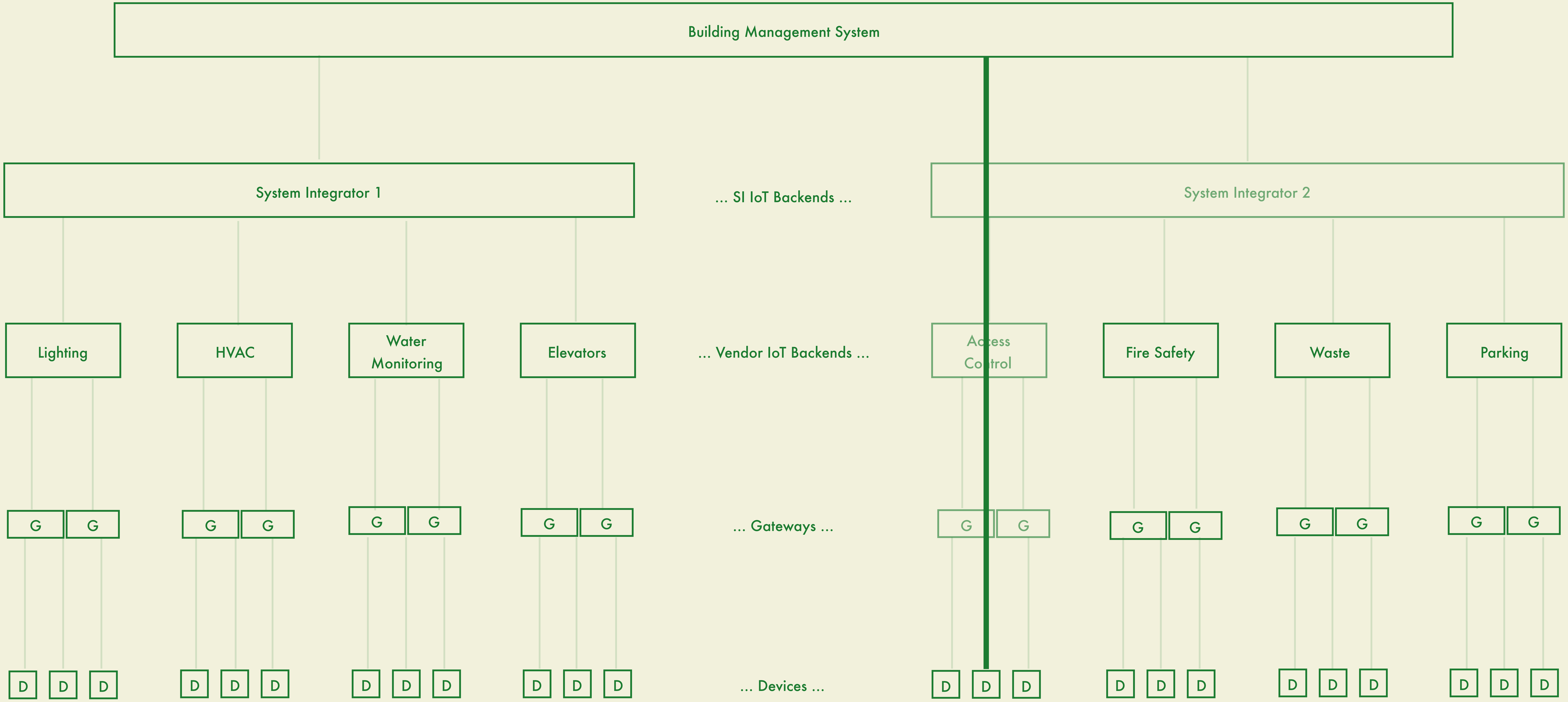
In enterprise settings the problems get worse ...



Several transport protocols, transport layer connections & IoT “platforms” are in the path of one message.



Complexity & attack surfaces grow to be unmanageable. Proprietary data is leaked. Security becomes untenable.



End-to-end secure channels can bring control back in the hands of the end customer.

Control on what data is visible where, in our systems, allows us to be deliberate about who can see our business proprietary data which, in turn, enables new business models.

This is much better than our current game of whack-a-mole, trying to endlessly thwart security bugs, on a wide open surface.

Secure Messaging.

Communication using messages traveling over end-to-end secure and private channels.

Secure Messaging.

Secure Channels

Authenticated Key Exchange, Authenticated Encryption, Proof of possession of secret key,
Session Management, Key Ratcheting, Message Ordering, Delivery Guarantees ...

Trust.

The willingness of one party to rely on the actions of another party.

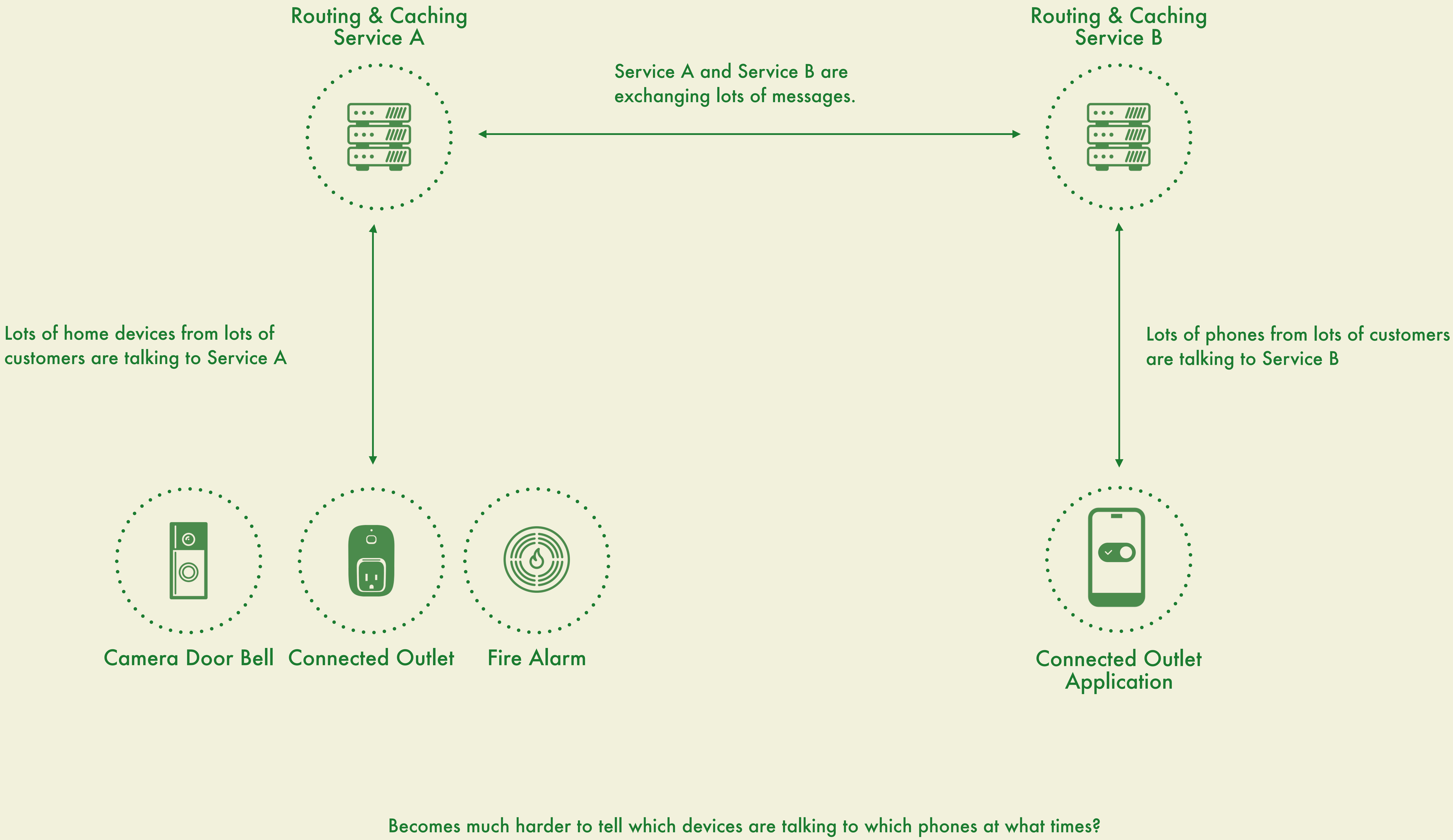
Secure Messaging.

Identity and Trust	Key Rotation, Key Lineage, Key Endorsements, Endorsement Revocation, Credentials, Credential Revocation, Anonymous Credentials, Delegation …
Secure Channels	Authenticated Key Exchange, Authenticated Encryption, Proof of possession of secret key, Session Management, Key Ratcheting, Message Ordering, Delivery Guarantees …

Secure Messaging.

	Routing, Caching, Prioritized Ordering, Encrypted Group Messaging, Publish/Subscribe ...
Identity and Trust	Key Rotation, Key Lineage, Key Endorsements, Endorsement Revocation, Credentials, Credential Revocation, Anonymous Credentials, Delegation ...
Secure Channels	Authenticated Key Exchange, Authenticated Encryption, Proof of possession of secret key, Session Management, Key Ratcheting, Message Ordering, Delivery Guarantees ...

Secure Messaging features like routing, caching, prioritized ordering can be generalized and don’t have to be use case specific, this further improves security and privacy …



Routing & Caching
Service A
Routing & Caching
Service B
Service A and Service B are
exchanging lots of messages.
Lots of home devices from lots of
customers are talking to Service A
Lots of phones from lots of customers
are talking to Service B
Camera Door Bell
Connected Outlet
Fire Alarm
Connected Outlet
Application
Becomes much harder to tell which devices are talking to which phones at what times?

Edge computing, security enhancing and privacy preserving technologies of the near future will need this foundation of secure messaging ...

New Protocols.

Apple’s Find My - when a device goes missing, it begins periodically broadcasting a derived public key, the surrounding devices can then help in the finding of the offline device by encrypting their location to the public key.

A shared foundation of secure messaging would allow us to have similar features in lots of IoT devices

Federated Learning.

Google Keyboard learns out-of-vocabulary words on mobile phones without exposing sensitive text to servers.

Connected sensors could similarly learn to improve accuracy while preserving privacy.

Zero Knowledge Proofs.

Mozilla is using Non-Interactive Zero Knowledge Proofs to collect telemetry from the Firefox browser without collecting any private browser usage.

A large subset of IoT use cases is telemetry collection.

Decoupling the secure channel protocol from the transport layer protocols removes complexity, minimizes the attack surface and can enable us to build better end-to-end secure and private systems.

Ockam

We're building a multi-language open source library that makes it easy to add secure messaging to IoT systems.

github.com/ockam-network/ockam
ockam.io

Mrinal Wadhwa
CTO, Ockam
twitter.com/mrinal