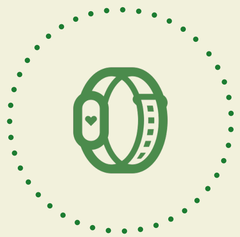
The Internet of Things needs:

Secure Messaging.

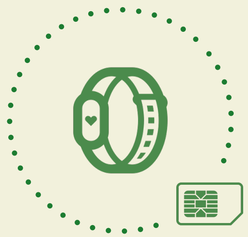
Mrinal Wadhwa
Ockam



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 ...

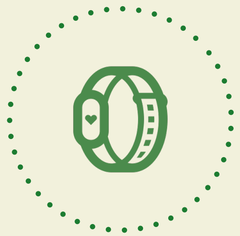


Heart Rate
Monitor

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 ...

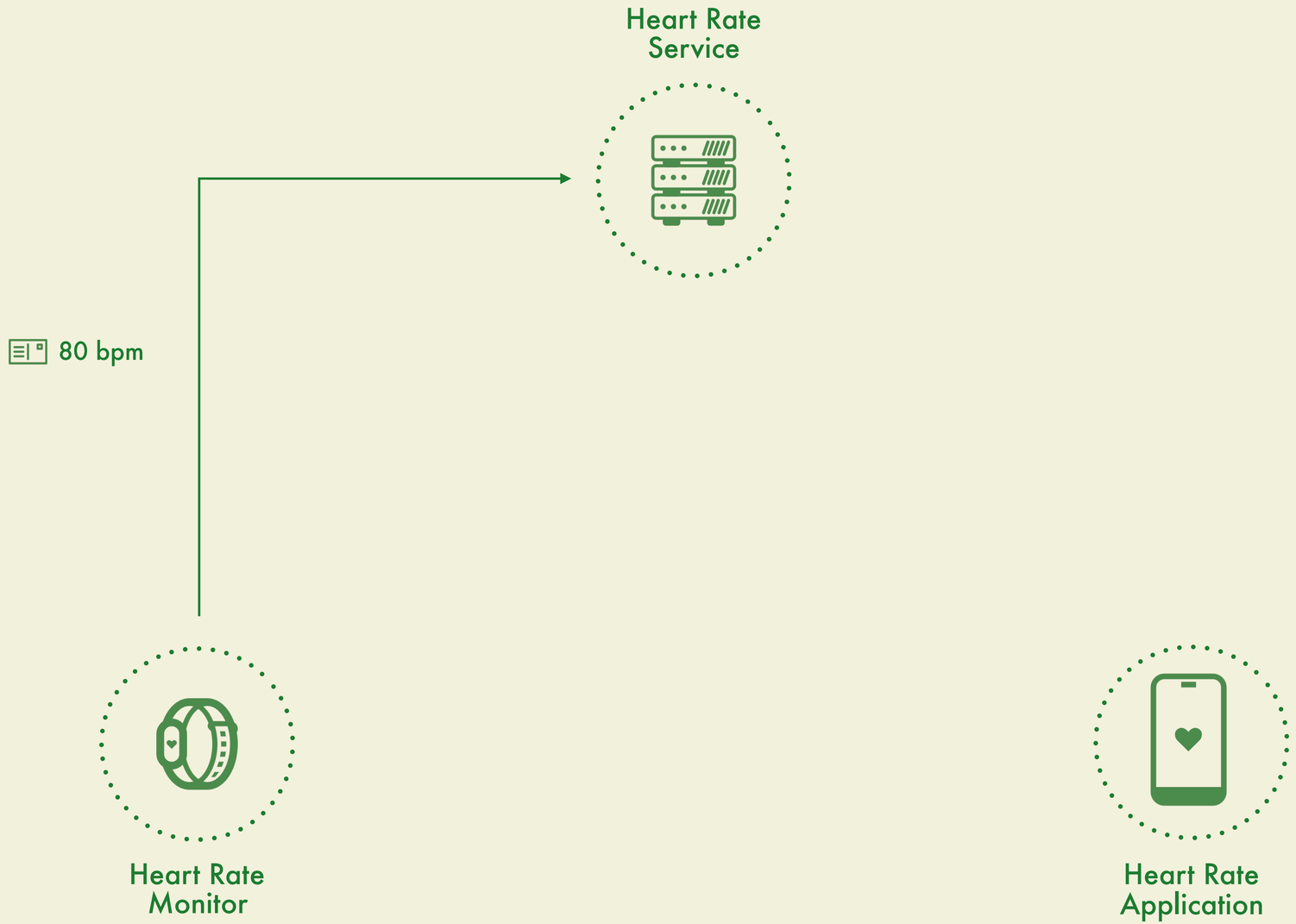
Heart Rate
Service



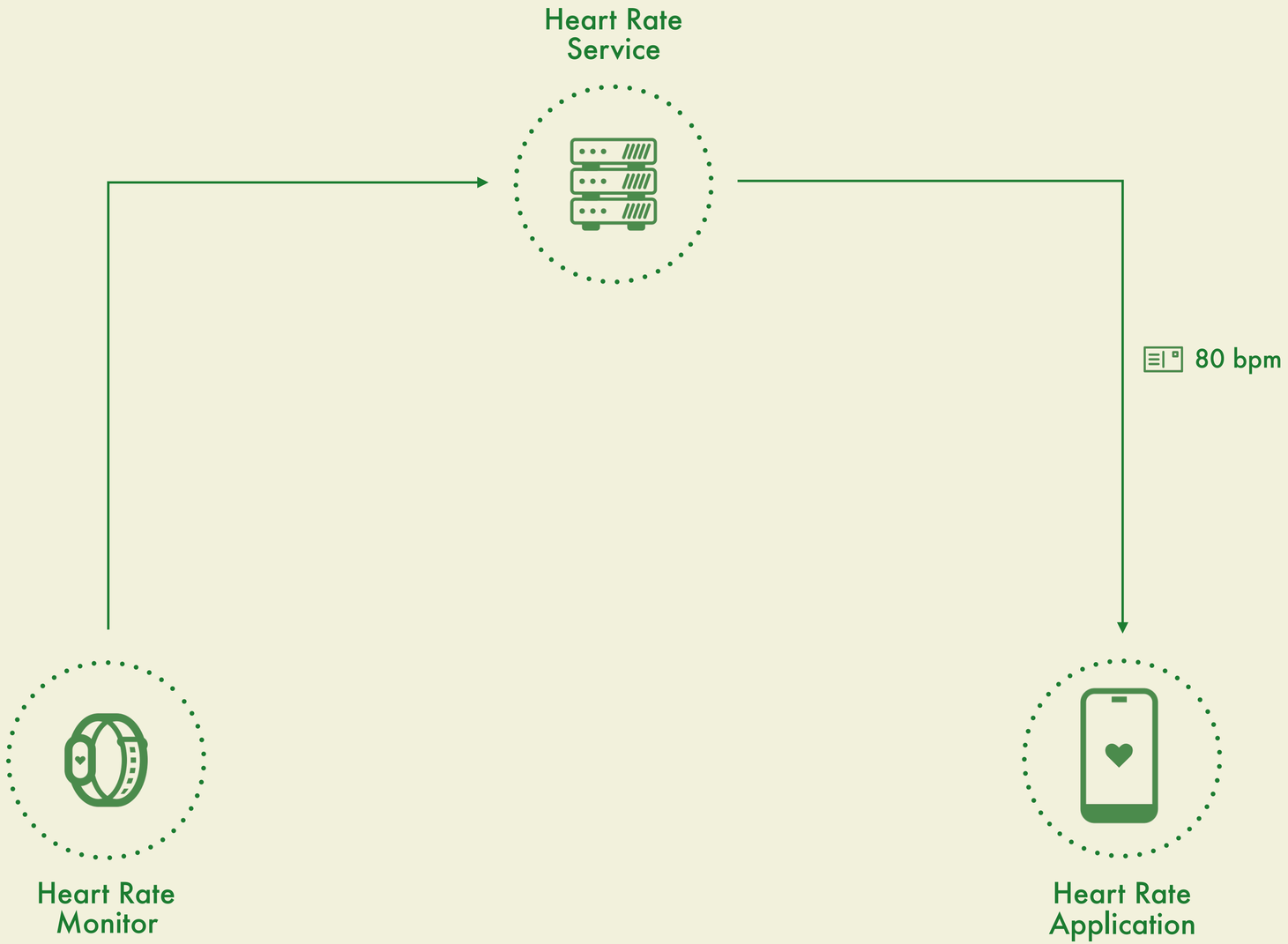
Heart Rate
Monitor

Heart Rate
Application

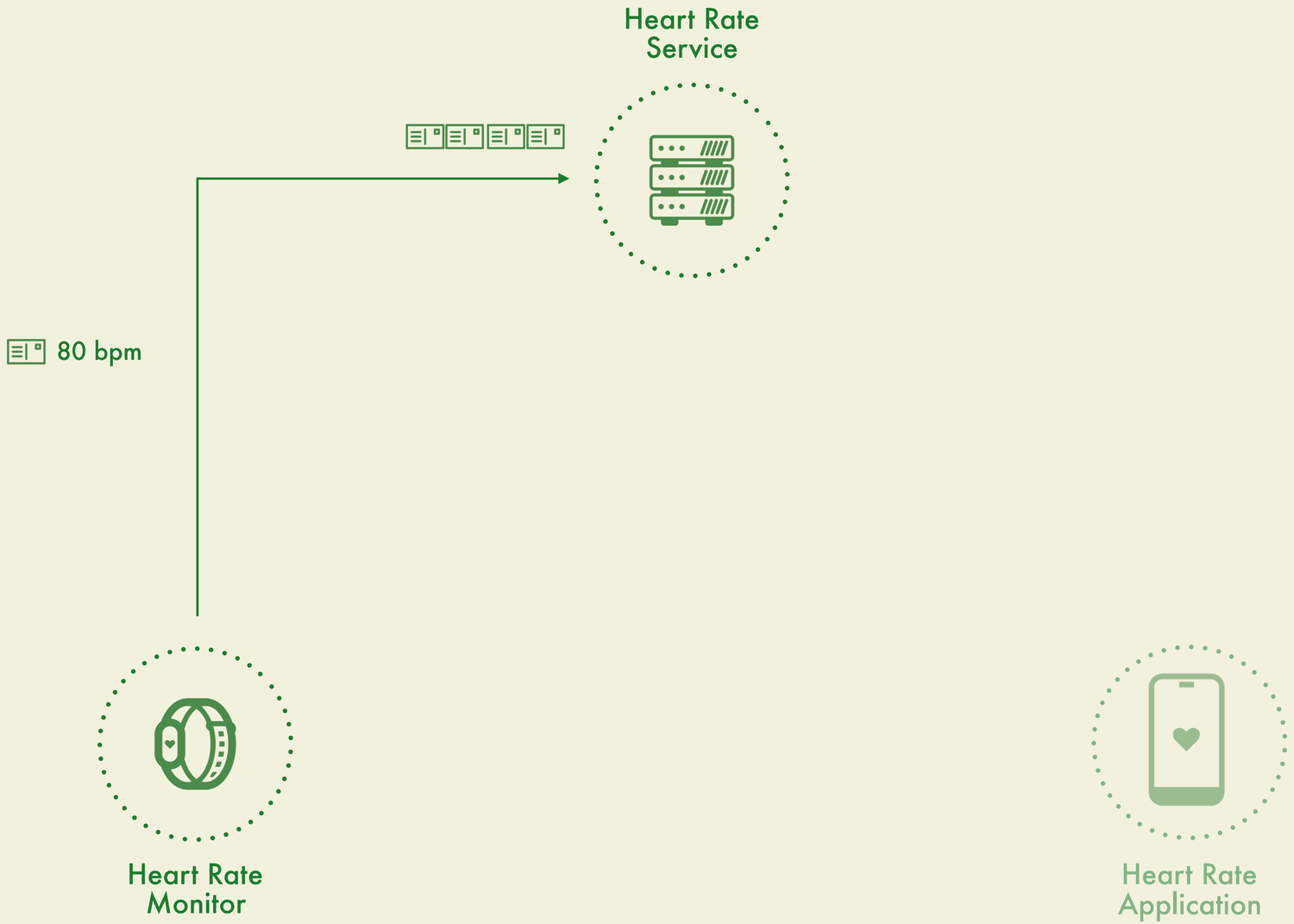
Typically you would setup a web service to deliver heart rate readings to a phone.
Since there is no direct route from the device to the phone.



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.

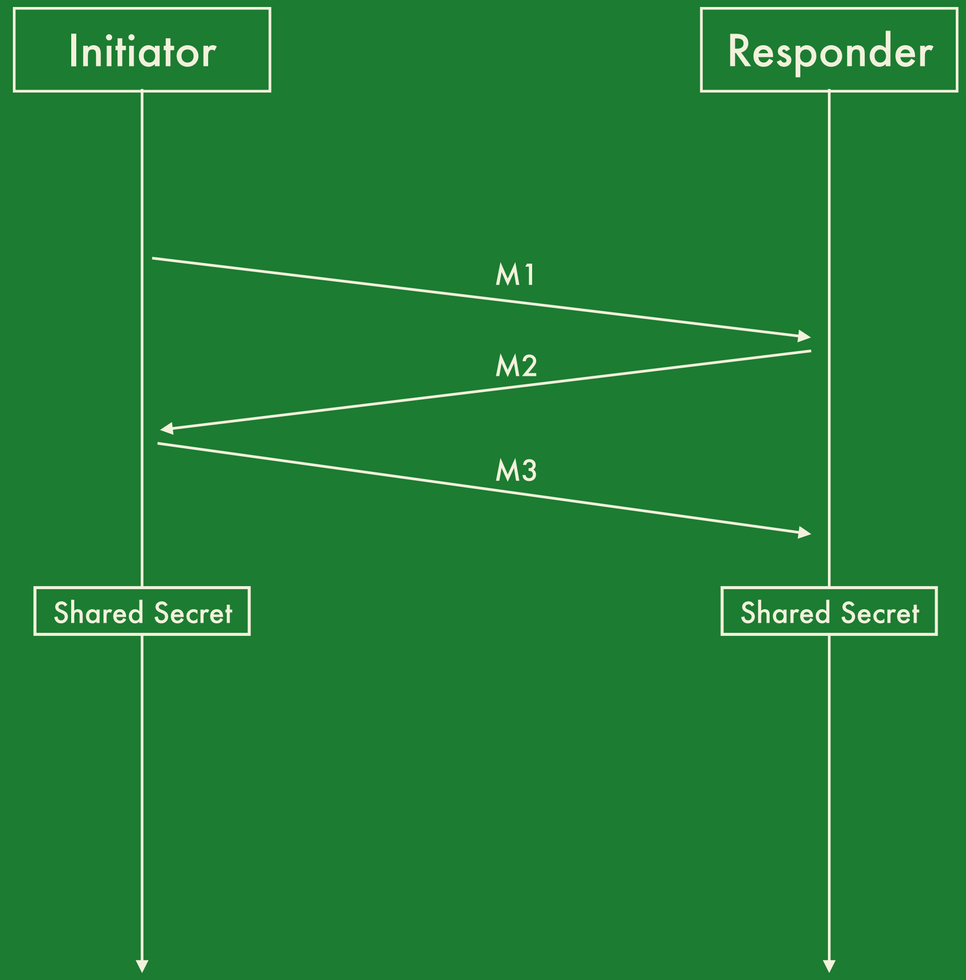
The degree of resistance to encountering an unfortunate event.

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.

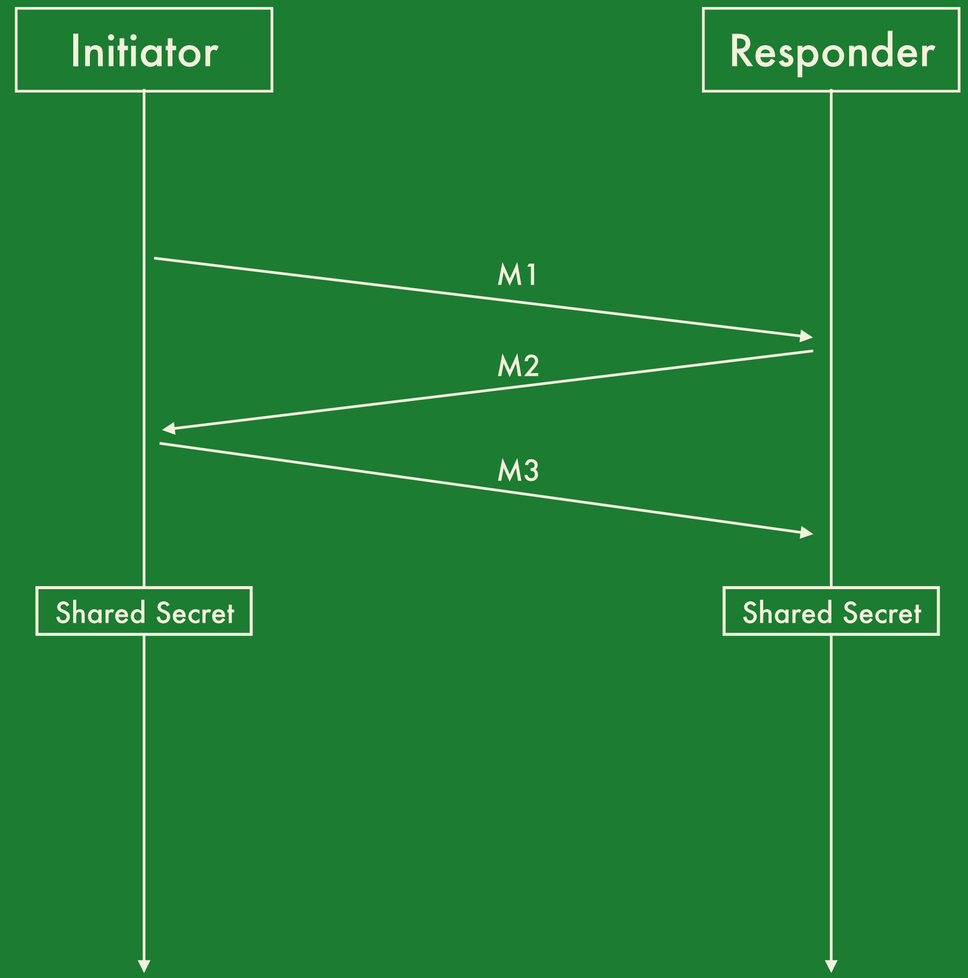
Secure Channels.



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.

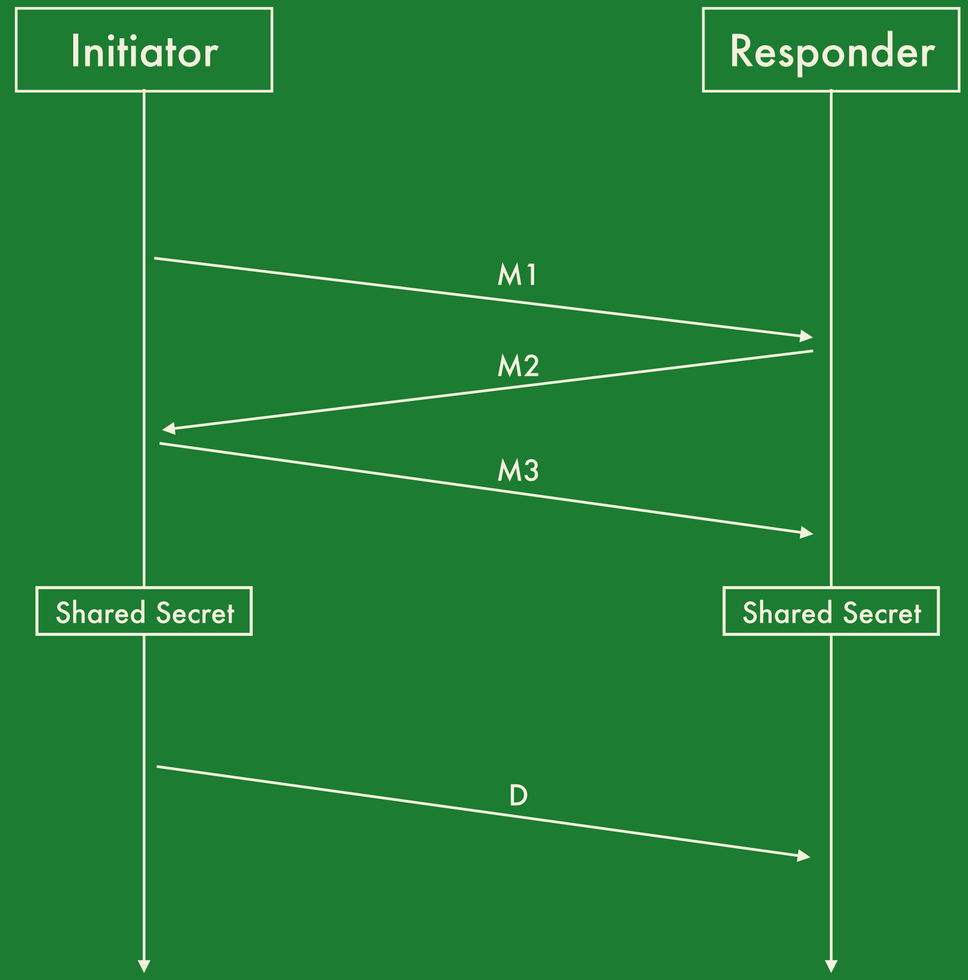


Authenticated Key Exchange

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

Authenticated Encryption with Authenticated Data

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



We need Secure Channels even when we aren't looking for confidentiality/encryption.

	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.

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? 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, GCM, GCM-SIV ...
9. Authenticated Key Exchange? SigMa like or Noise like?
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?

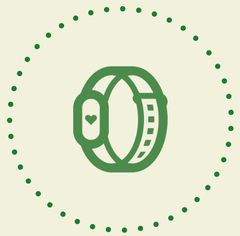
Many transport protocols, that are commonly used within IoT systems,
provide some notion of a secure channel.

However, such secure channels implementations are usually:

- Tightly coupled with transport layer protocol connections.
- Hard to configure correctly.
- Inefficient in resource usage (code size, memory, compute, network).
- Poorly implemented black boxes that cannot be audited.
- Quality of their design can vary in many subtle ways (cryptographic choices from previous slide)
- Require expensive licenses.

  ...

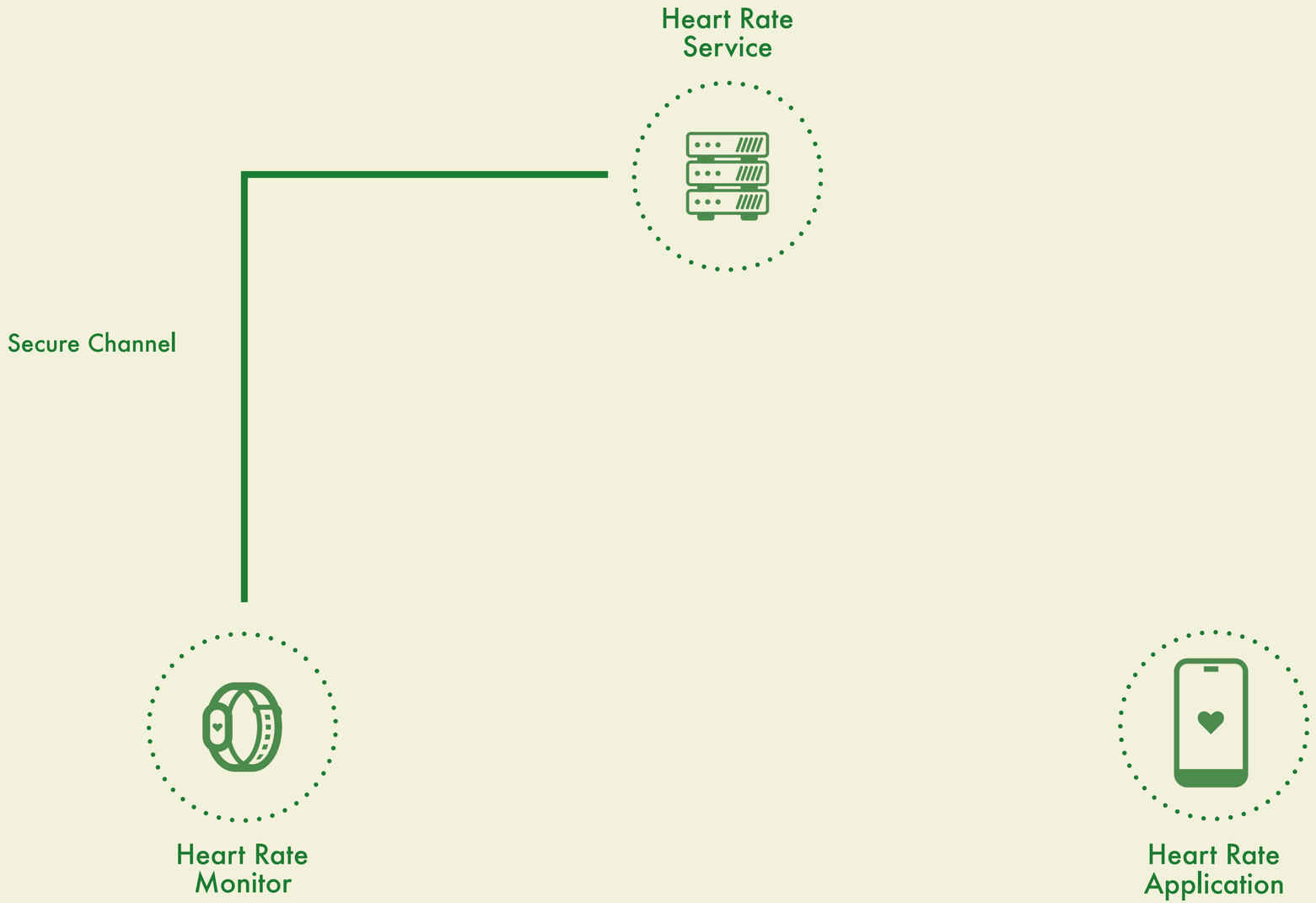
Heart Rate
Service



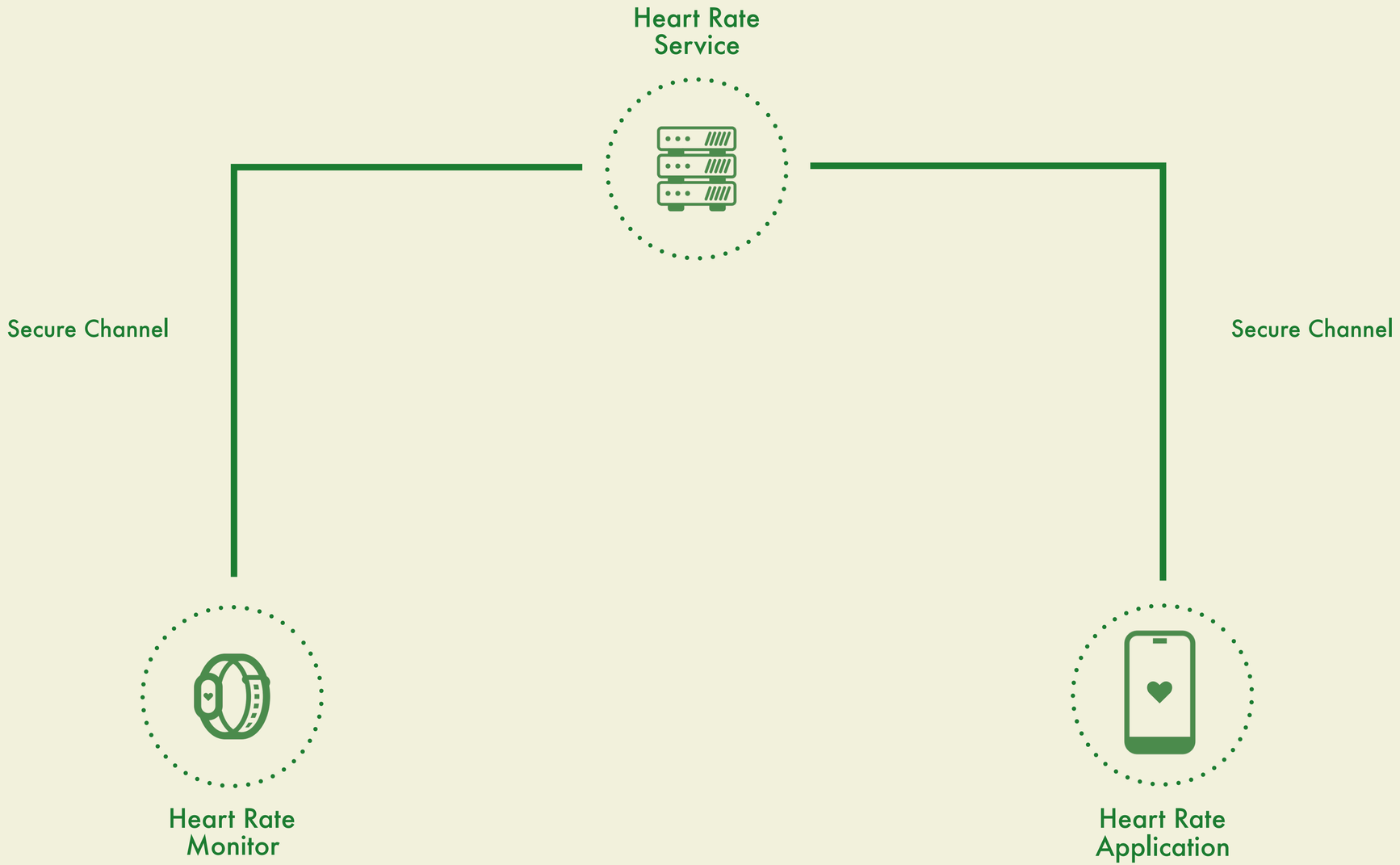
Heart Rate
Monitor

Heart Rate
Application

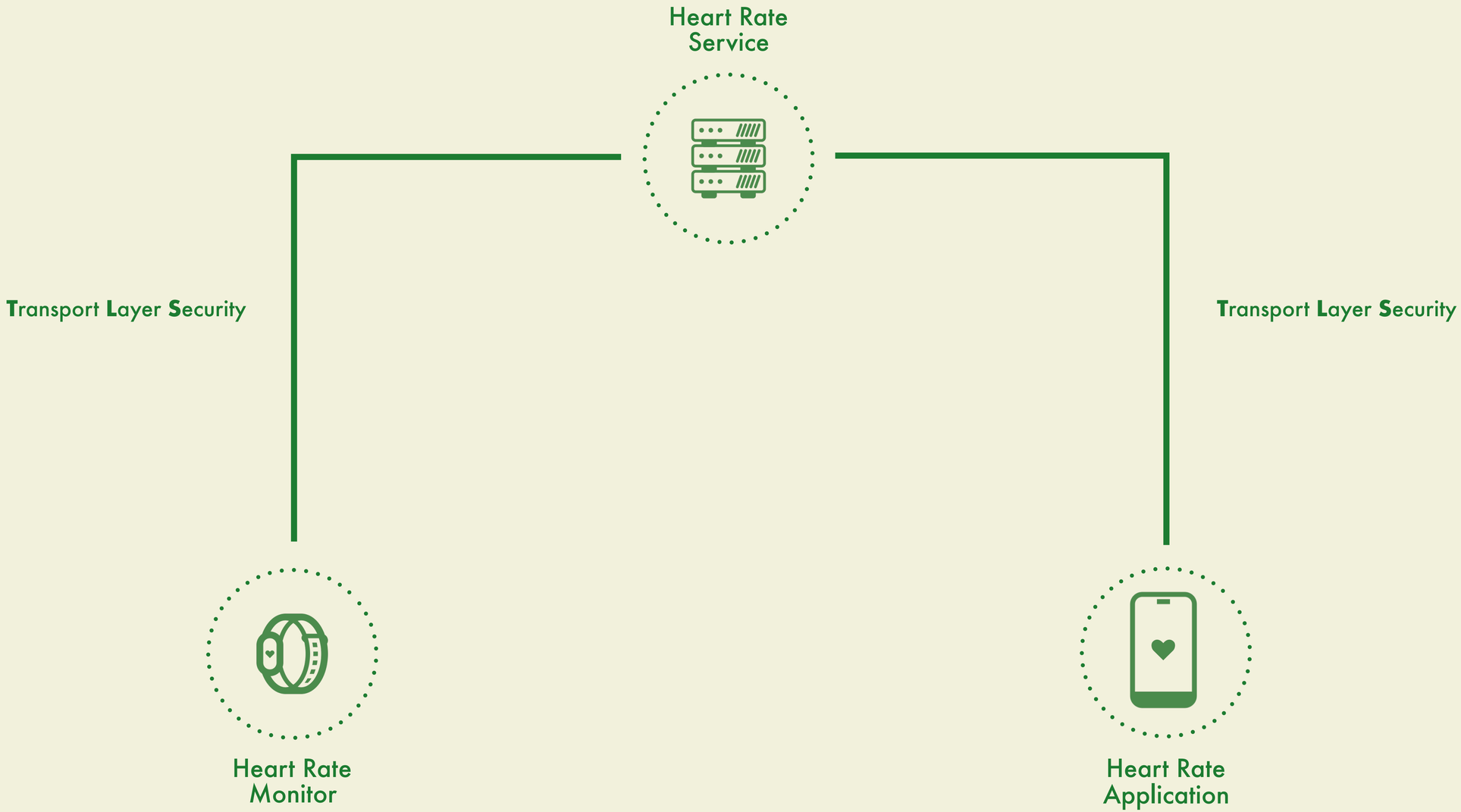
Coming back to our heart rate solution, for secure communication ...



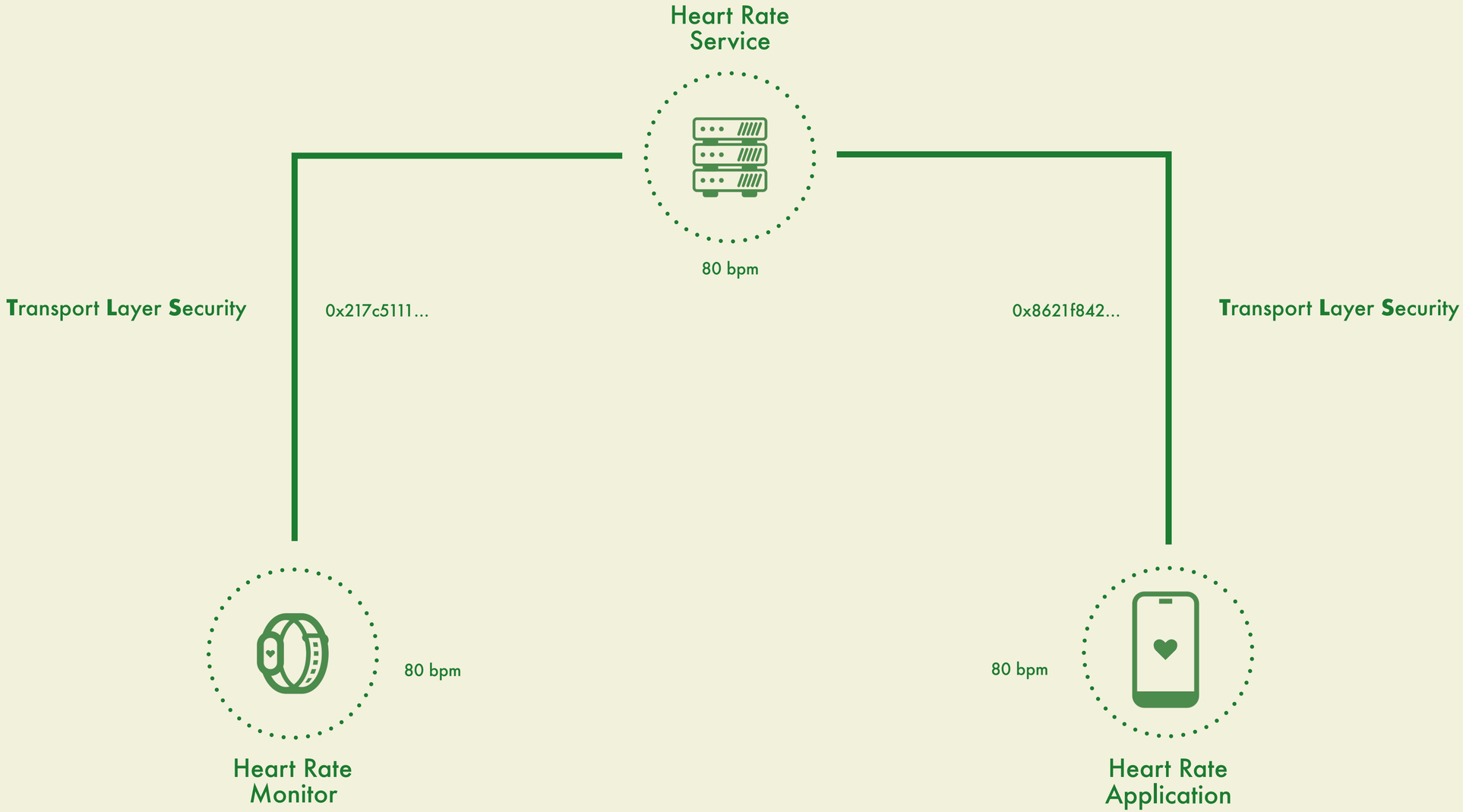
We setup a secure channel between the monitor and the service.



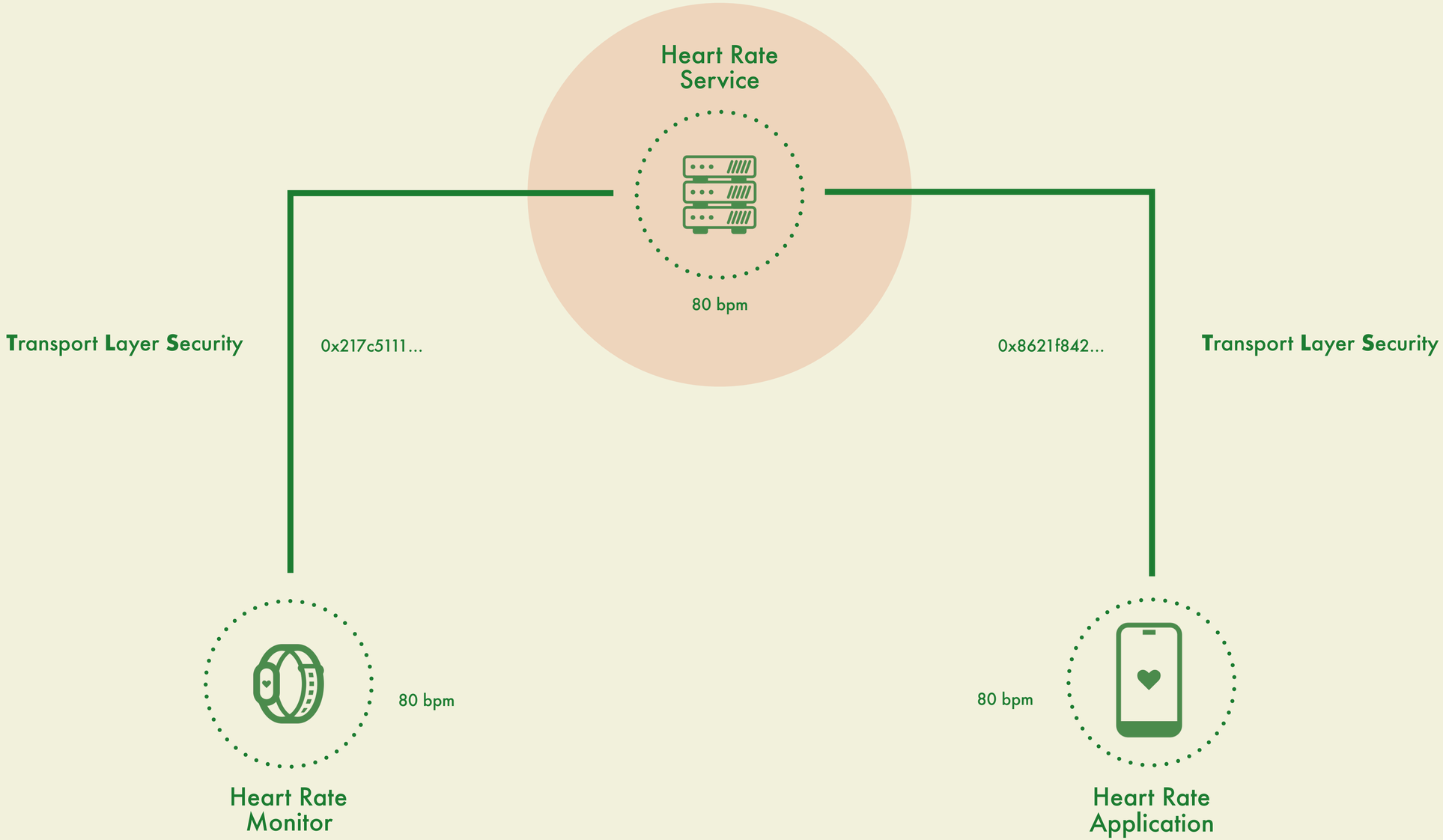
And another secure channel between the phone and the service.



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



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.

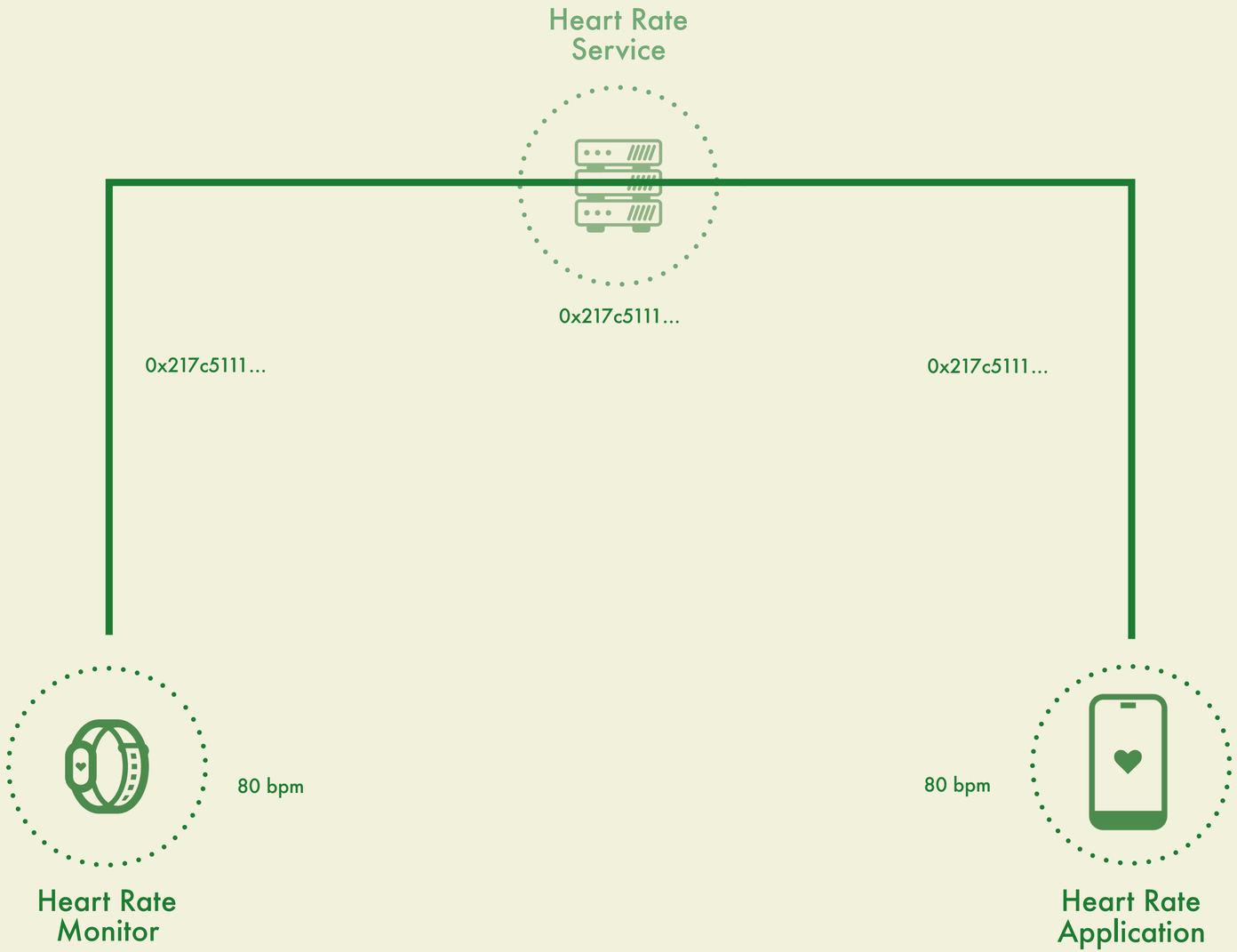
"Every program and every privileged user of the system should operate using the least amount of privilege necessary to complete the job."

– 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 ...).

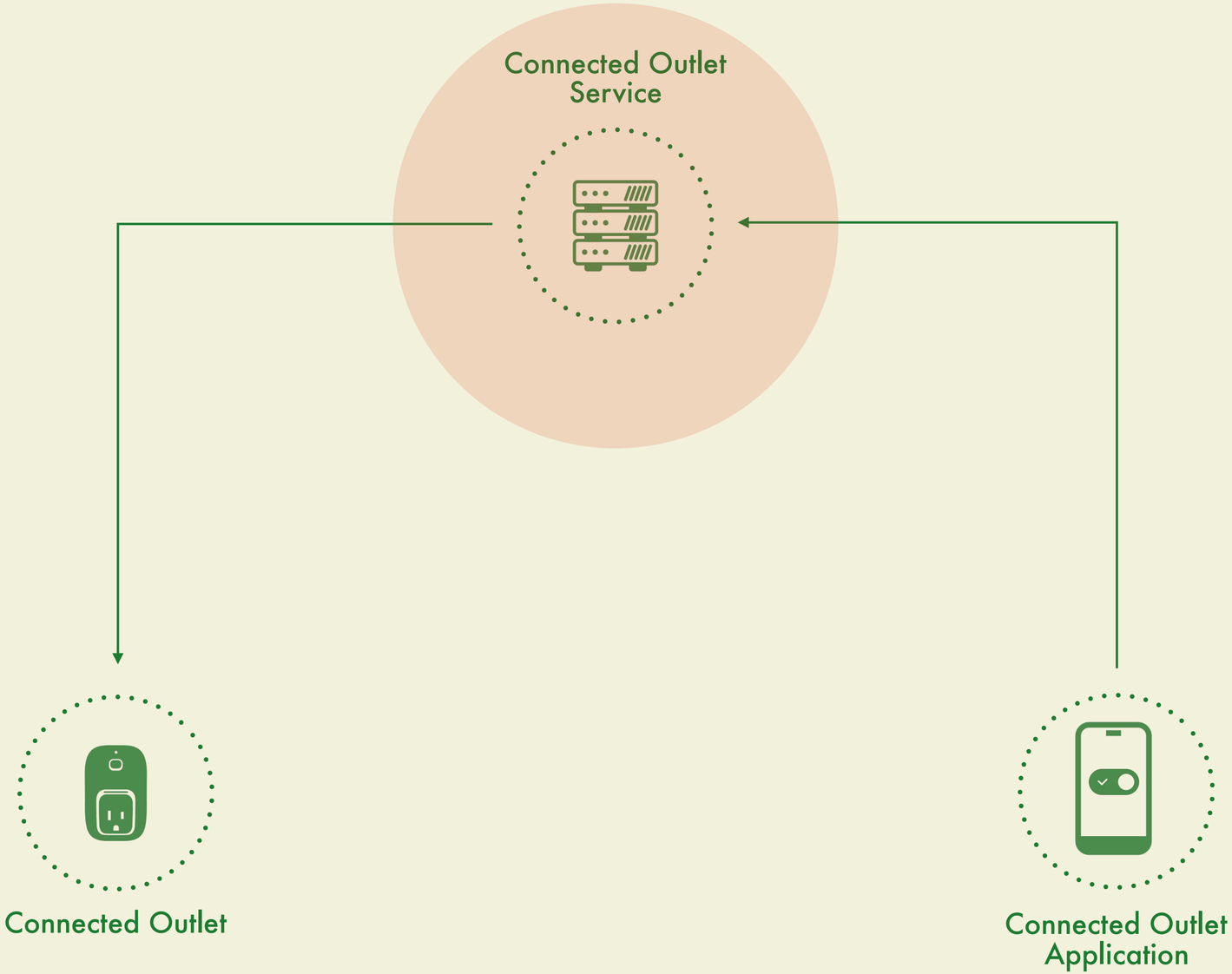
Privacy.

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

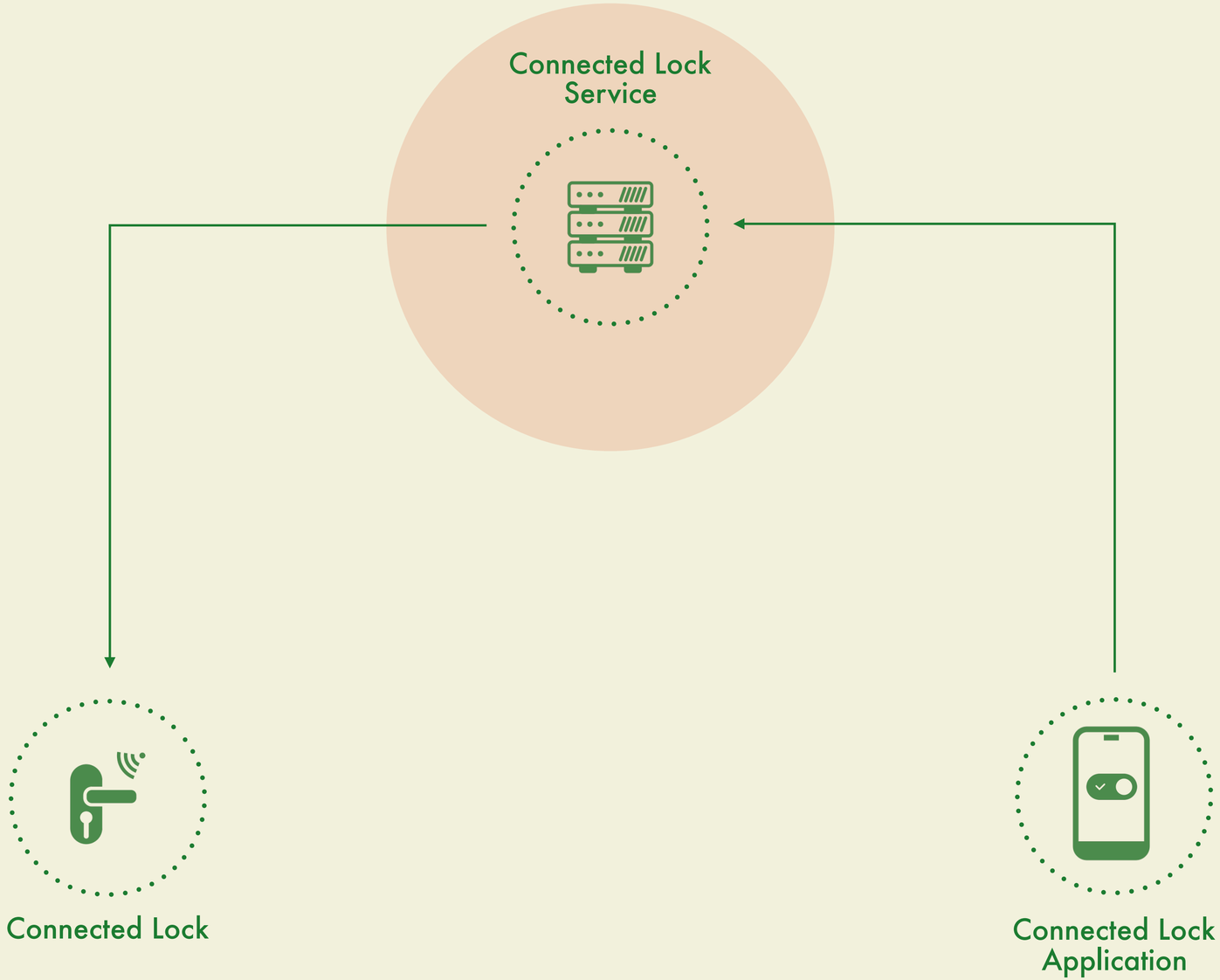


If, instead, we **decouple** the secure channel protocol from the transport connections, we could have an end-to-end secure and private channel.

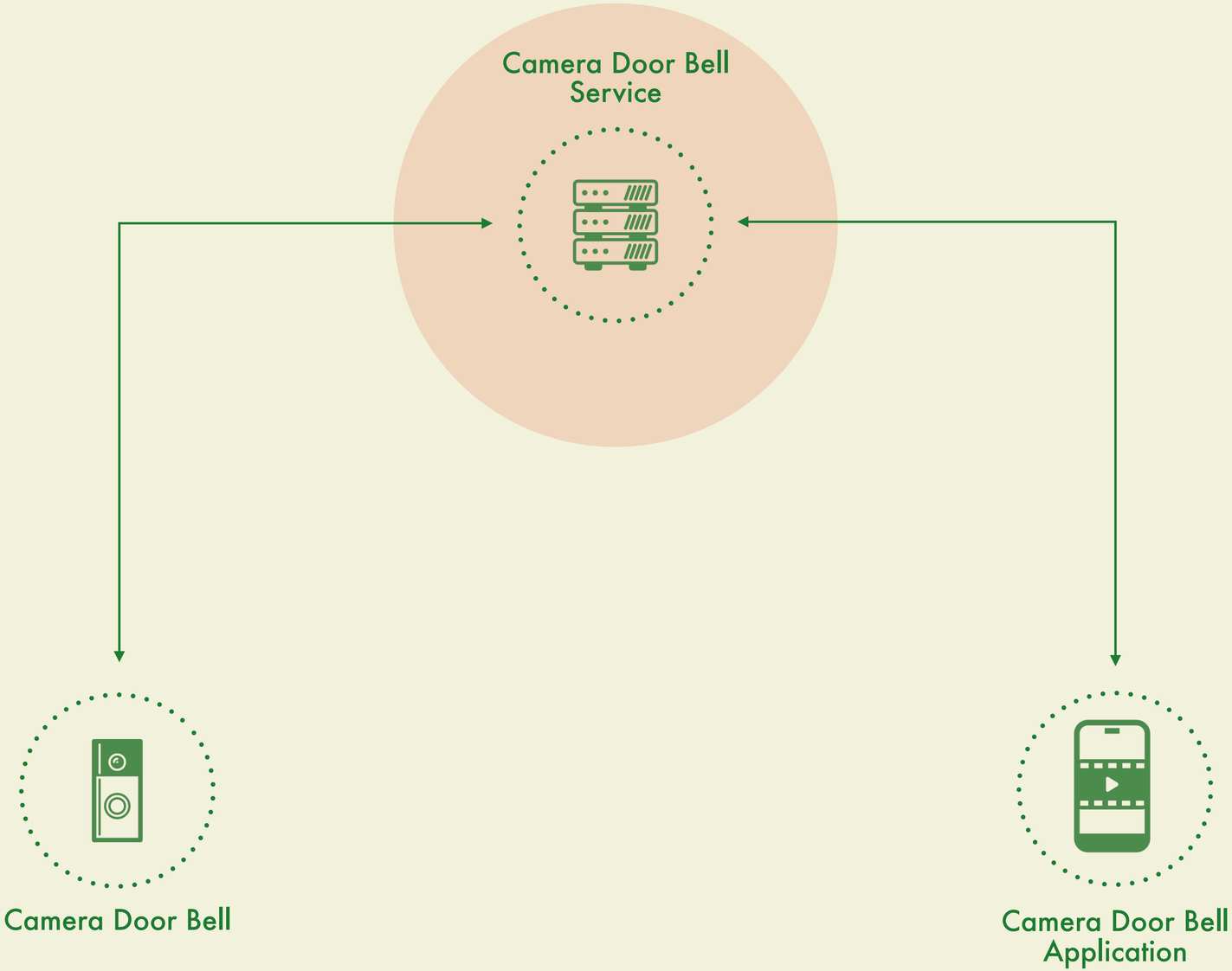
Lots of connected devices only need an internet service for routing and caching …



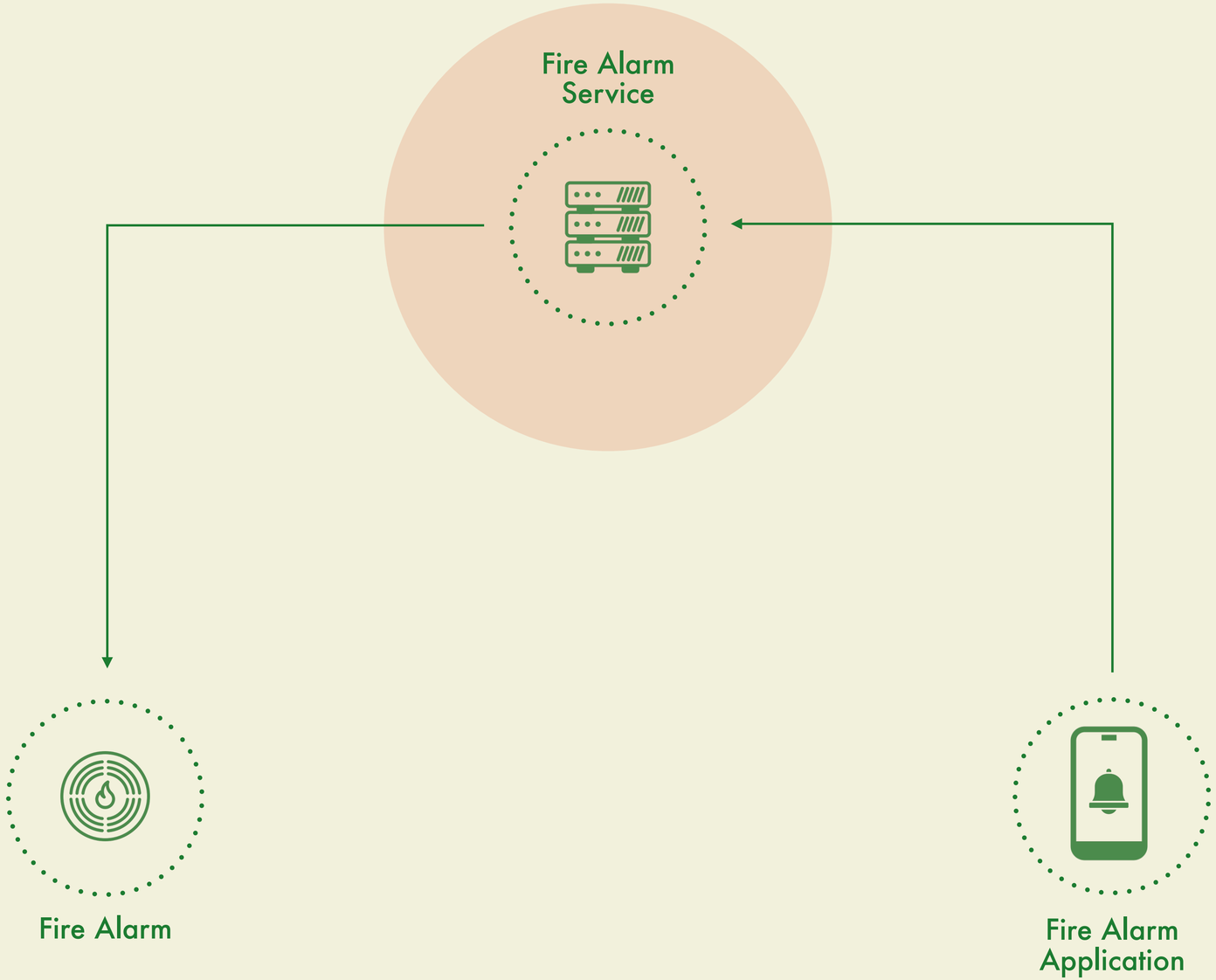
Route on/off instructions.



Route open/close instructions.

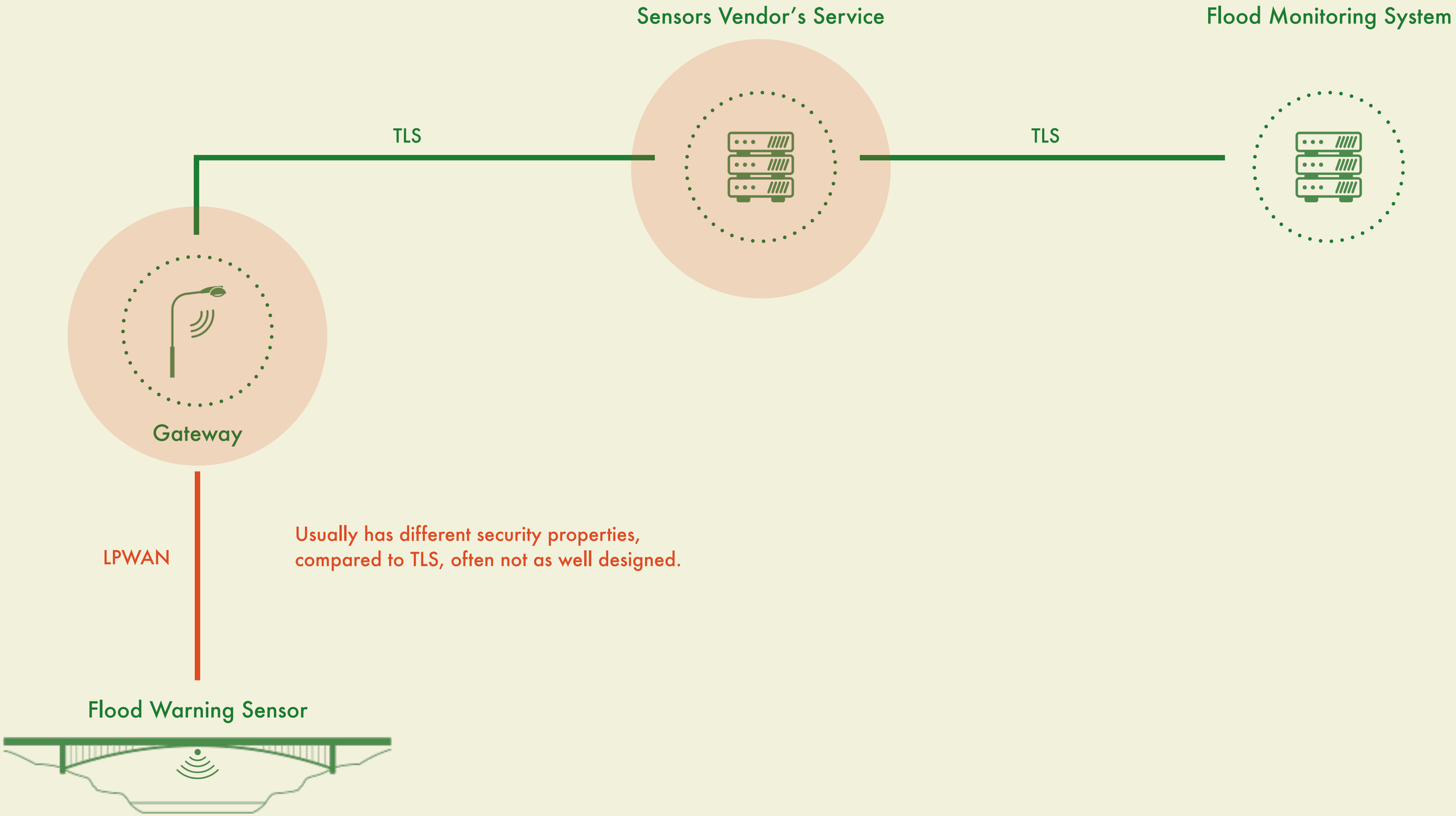


Route/Cache sensor data, alerts and videos.

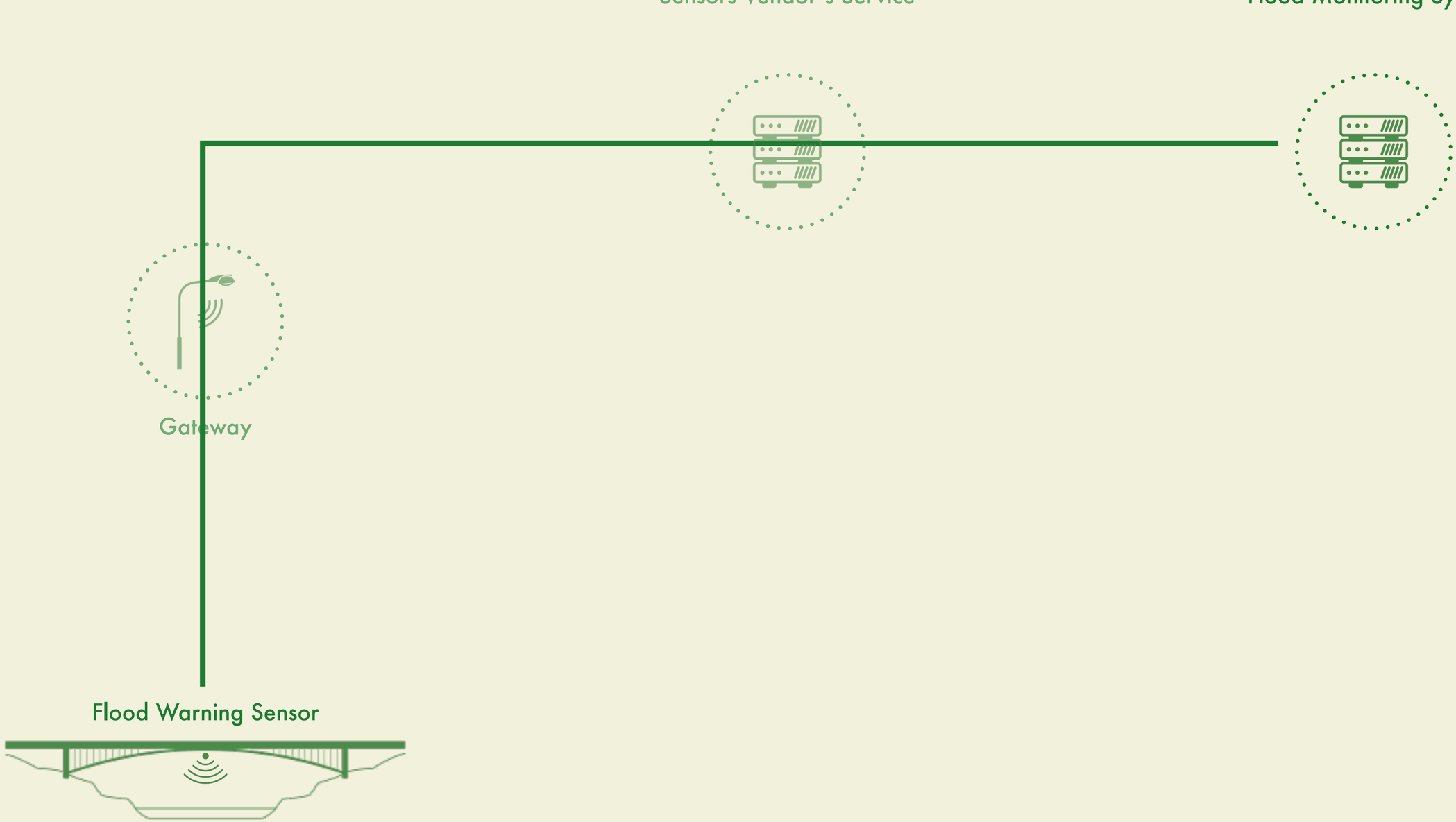


Route Sensor Alerts

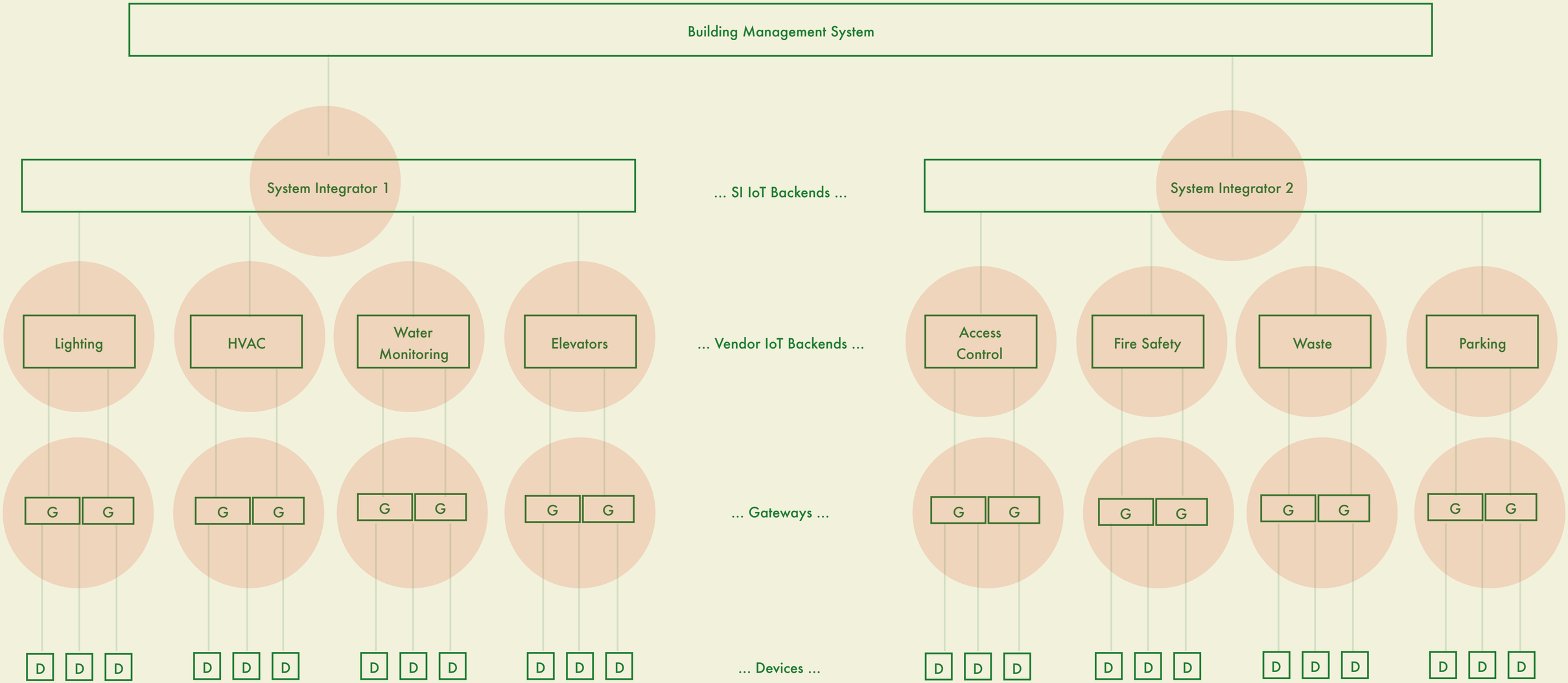
Many, many more … all of which can be built without exposing user/application data to the Internet (including the operator of the system).



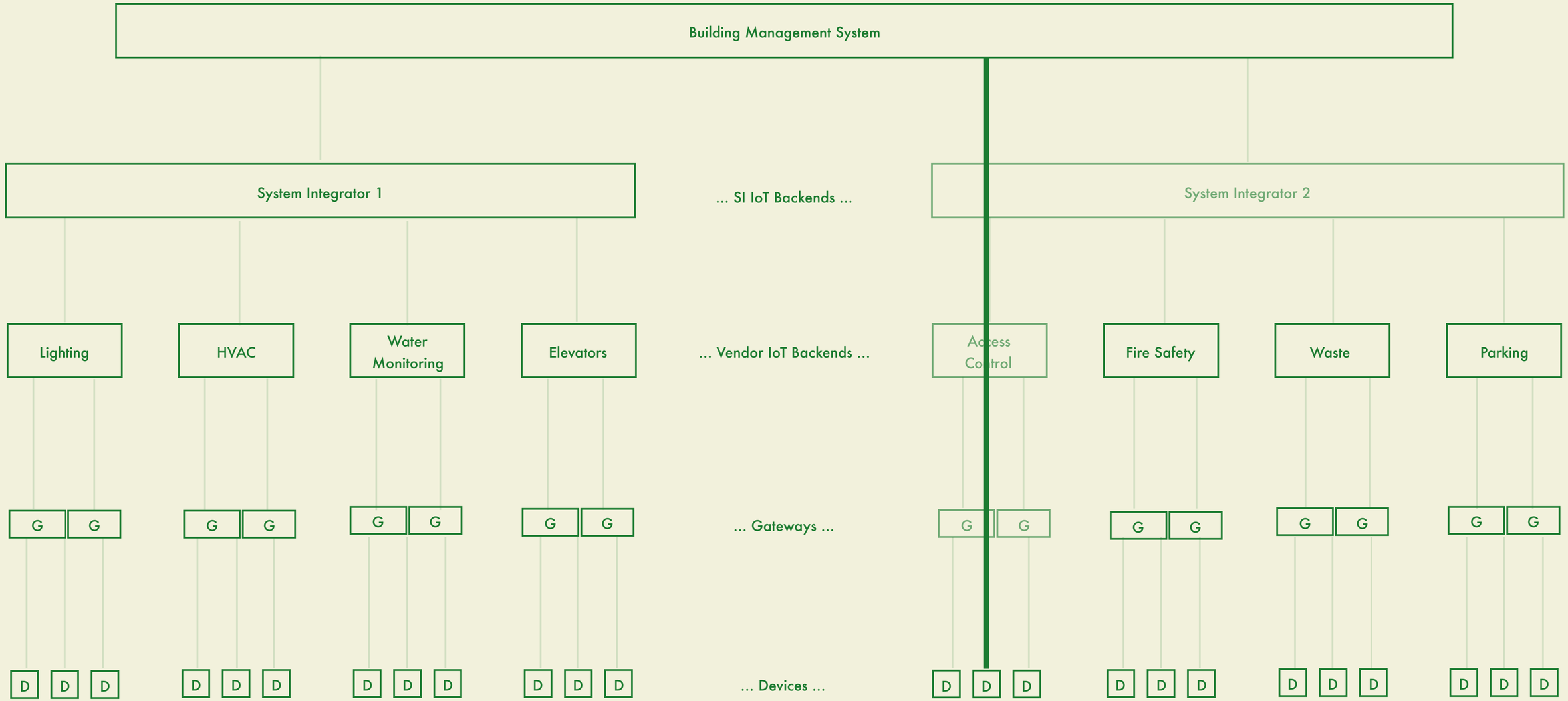
Various protocols have various different secure channel designs.



A secure channel that is decoupled from the transport layer connections.
The gateway and sensor vendor shouldn't be exposed to application data.



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 of ambient authority.

The Internet of Things needs:

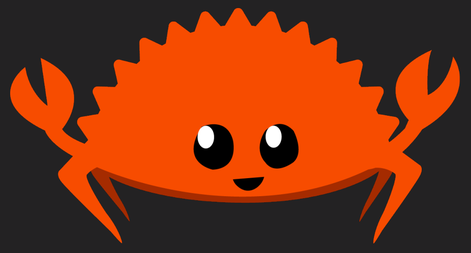
Secure Messaging.

Communication using messages traveling over end-to-end authenticated, encrypted, transport agnostic, secure, and private channels.

Ockam

We’re building multi-language, open source, libraries and tools that makes it easy to add secure messaging to IoT systems.

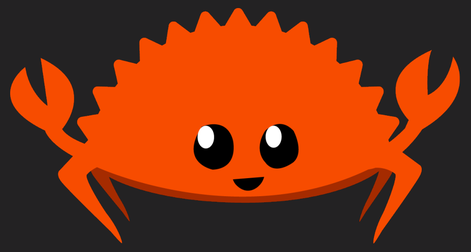
github.com/ockam-network



Zero Cost Abstractions

Vault, an abstraction over cryptographic hardware that provides a common set of sane building blocks that we can use to design secure channels and other higher level protocols.

Transport, an abstraction over various transport layer protocols that exposes consistent behavior that we can rely on for secure channels and protocols like routing.



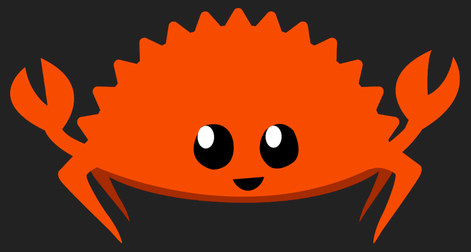
C Interoperability

Ease of calling C libraries from Rust.

Ease of calling Rust libraries from C

Ease of integrating with external build tools.

Ease of shipping libraries that can be seamlessly called from C code and used by proprietary toolchains

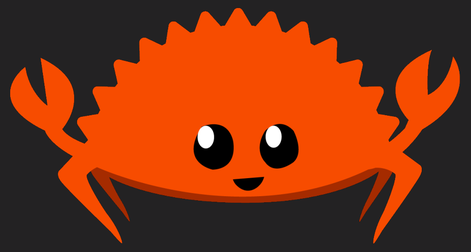


Erlang/Elixir Interoperability

Our Messaging and Routing infrastructure is being built in Erlang/Elixir.

Rust integrates nicely with the BEAM virtual machine, this will allow us implement our core cryptographic code once and share between embedded systems and cloud servers.

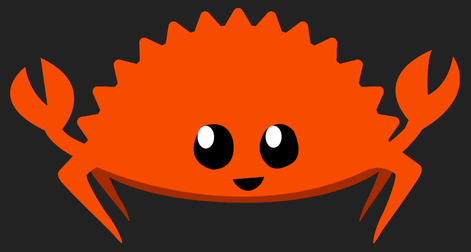
Rust NIFs (via rustler) can never crash the BEAM.



Cryptography

Rust Crypto, Dalek-Cryptography etc. are already really good and maturing fast.

They give us small, simple, well designed building blocks for our protocols.



Portability

embedded-hal and growing hardware support.

Mrinal Wadhwa

twitter.com/mrinal

Image Credits

https://www.behance.net/gallery/42774743/Rustacean

https://www.rustacean.net/