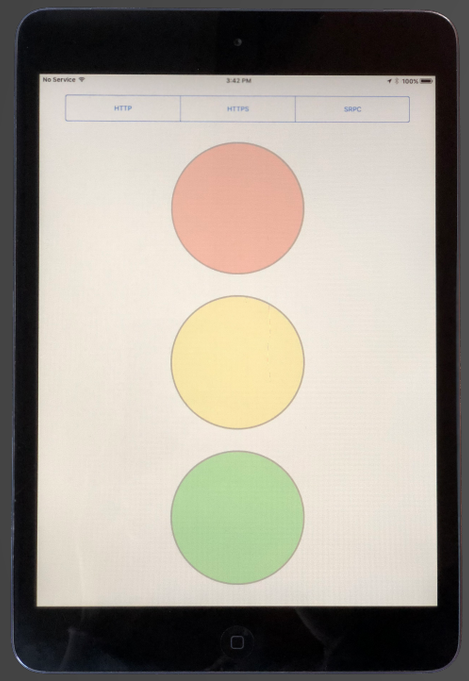
Is Security on Your Nerves?

Paul Rogers

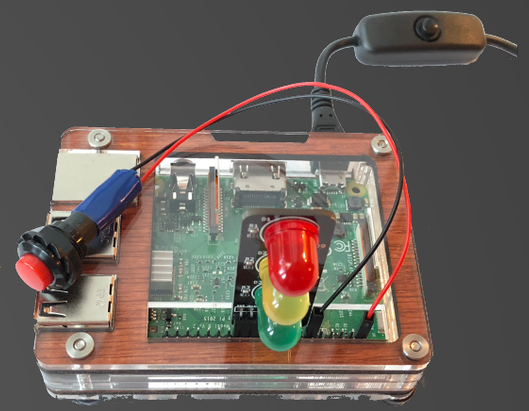
paul@knoxen.com

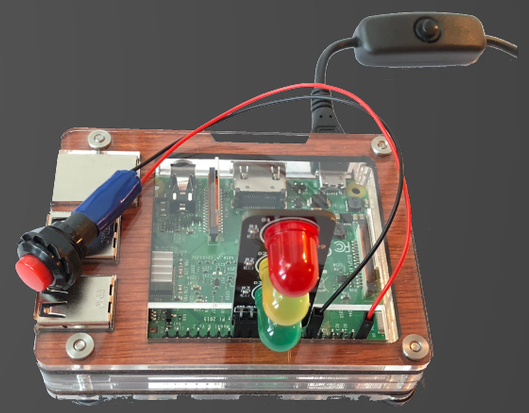
Scenario

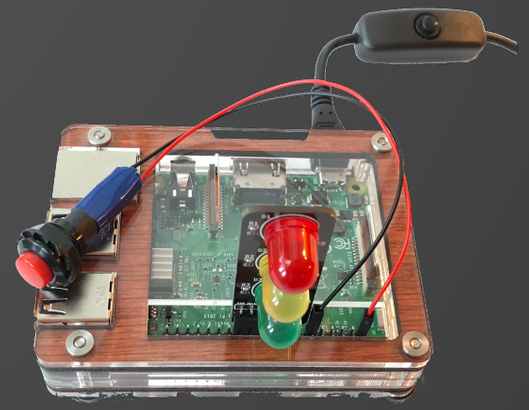
iPad controls stop light on an RPi3 device



HTTP

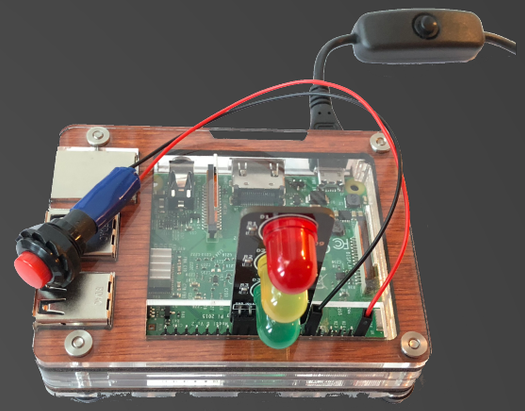


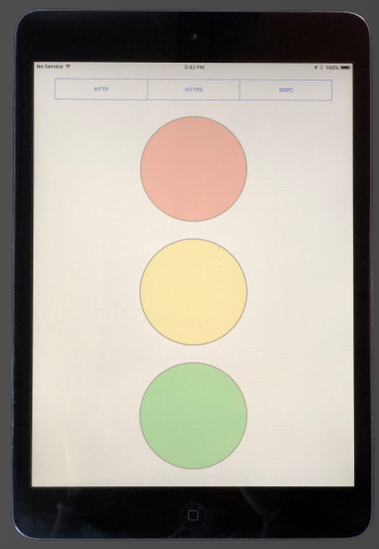




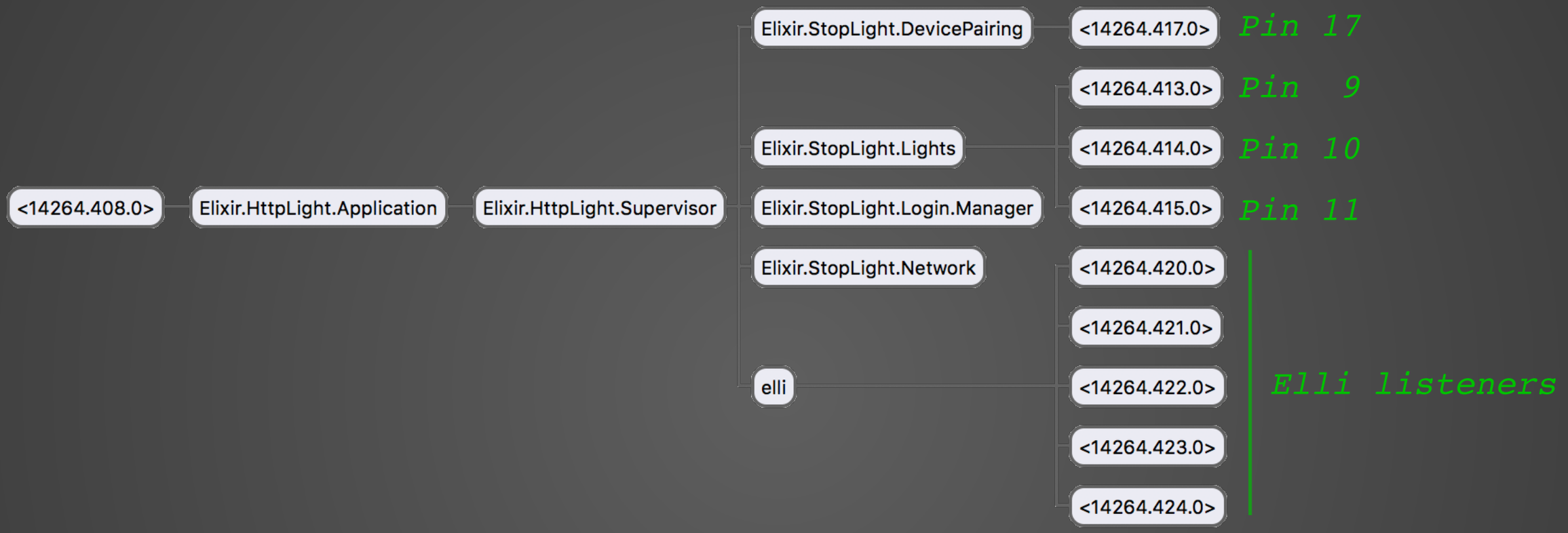
System Overview

- RPi3 Elixir Nerves
  - ElixirALE - Stoplight and momentary switch
  - Elli - HTTP interface
- iPad
  - iOS app responding to tap gestures
- Interaction
  - RPi3 is placed in “pairing” mode
  - iPad pairs with RPi3
  - iPad logs into the RPi3
  - iPad controls RPi3 stoplight via JSON API



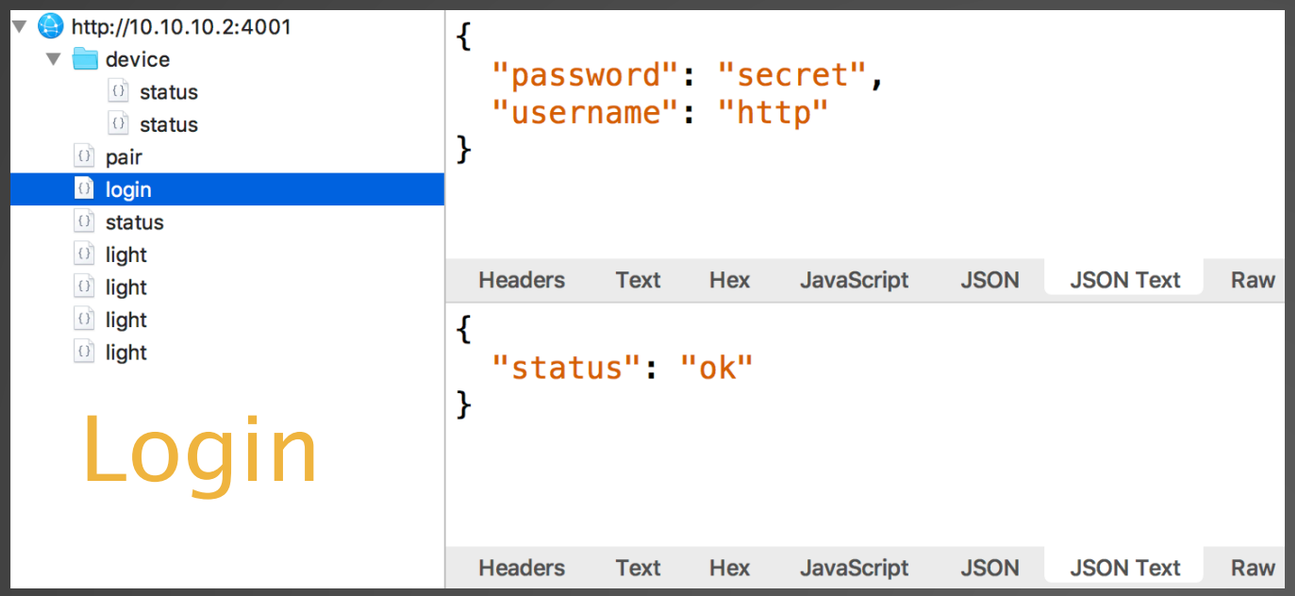


RPi3 Application Structure




```
config :stop_light, :elli,
  port: 4001,
  stack: [
    {StopLight.Elli.StatusHandler, []},
    {HttpLight.Elli.LoginHandler, []},
    {StopLight.Elli.LightsHandler, []}
  ]
```

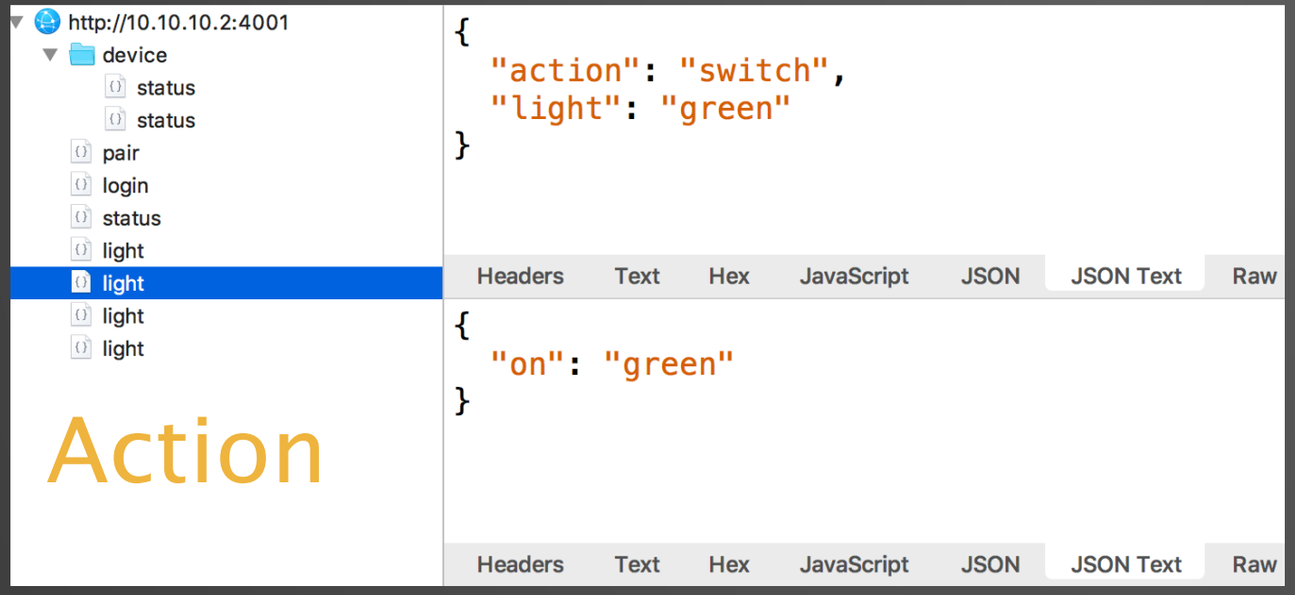
HTTP



Login

Request

Response



Action

Request

Response

```
curl -d '{"action":"switch","light":"red"}' 'http://10.10.10.2:4001/light'
```

curl

HTTP Issues

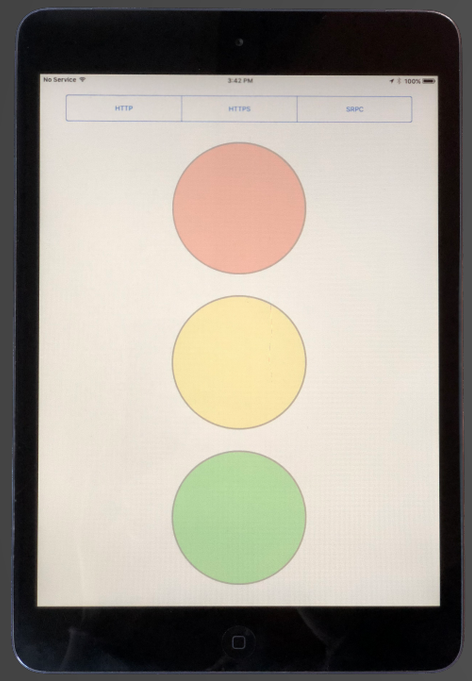
- Eve can snoop communications
  - Plaintext userid & password
  - Capture application data
- Mallory can snoop communications
  - Alter traffic
  - Manipulate API for fun and profit
- Mallory can *control* the stoplight *independently* of the iPad

Add Security

- Encrypt Communications
  - Use symmetric key encryption
    - How do we get the key on the iPad and RPi3?
- Pre-Shared Key
  - Significant issues (forward secrecy, key refresh & maintenance)
- Dynamic Key Establishment
  - Asymmetric scheme to establish a symmetric key
  - Symmetric encryption to provide secrecy

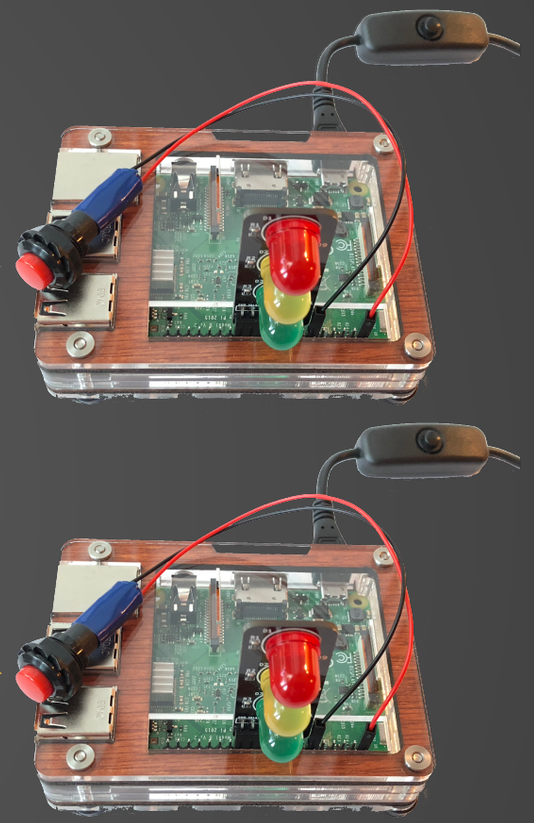
Scenario

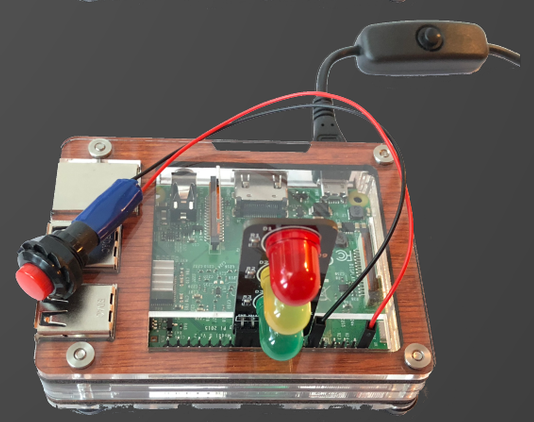
iPad controls stop light on an RPi3 device



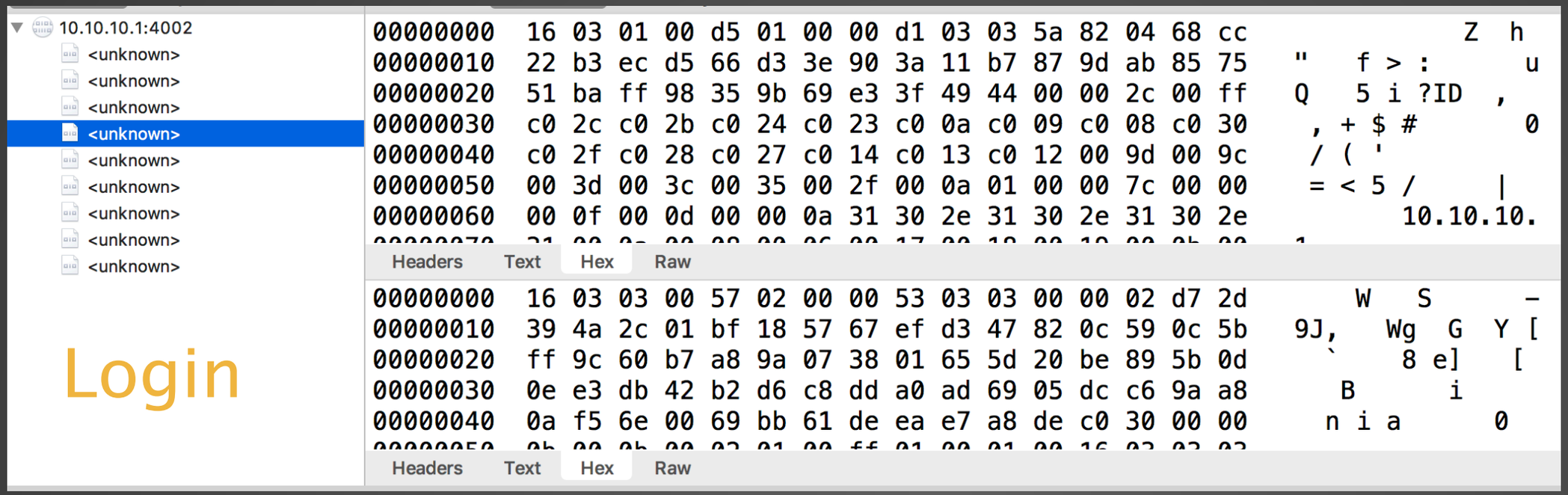
HTTP

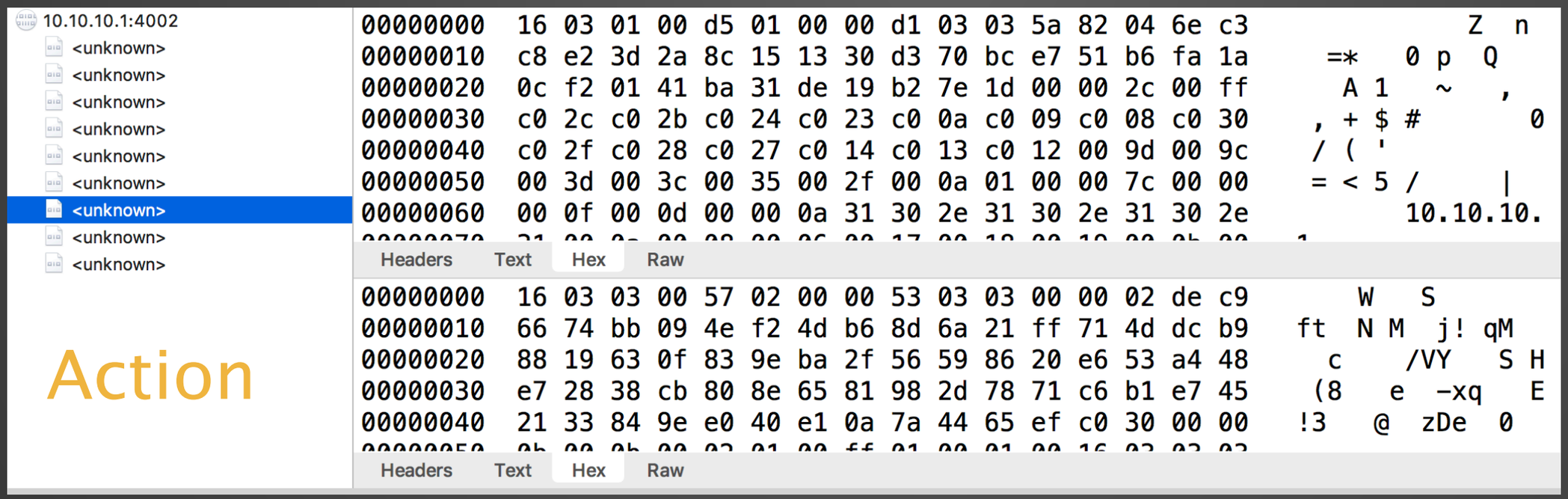
HTTPS



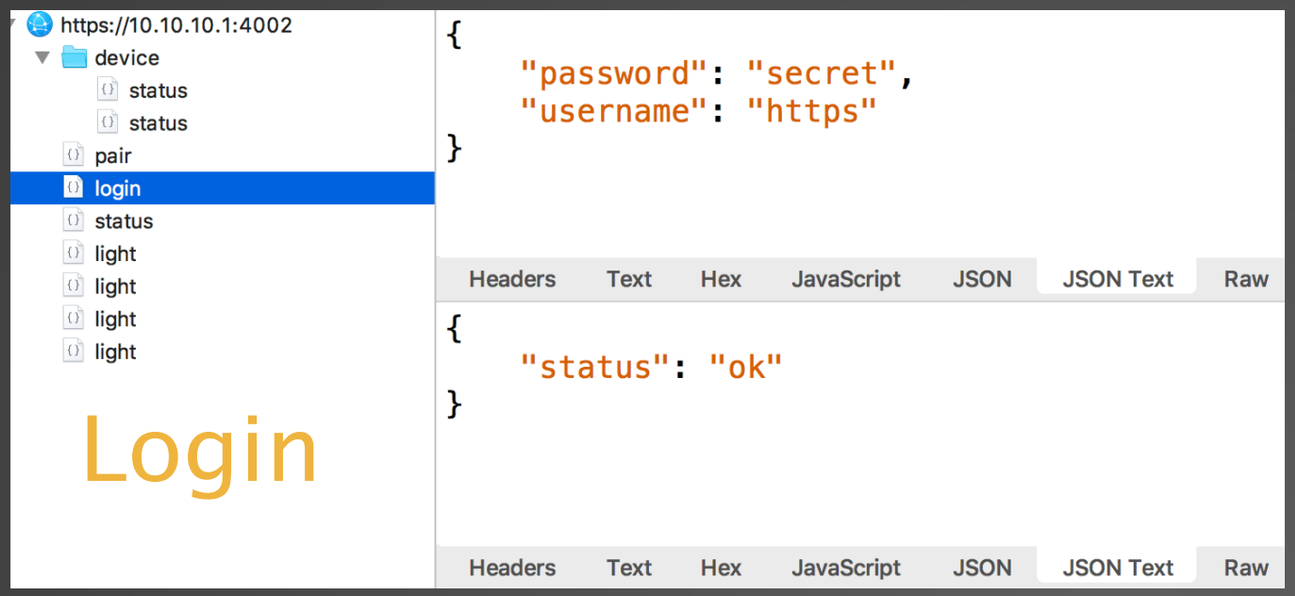


HTTPS





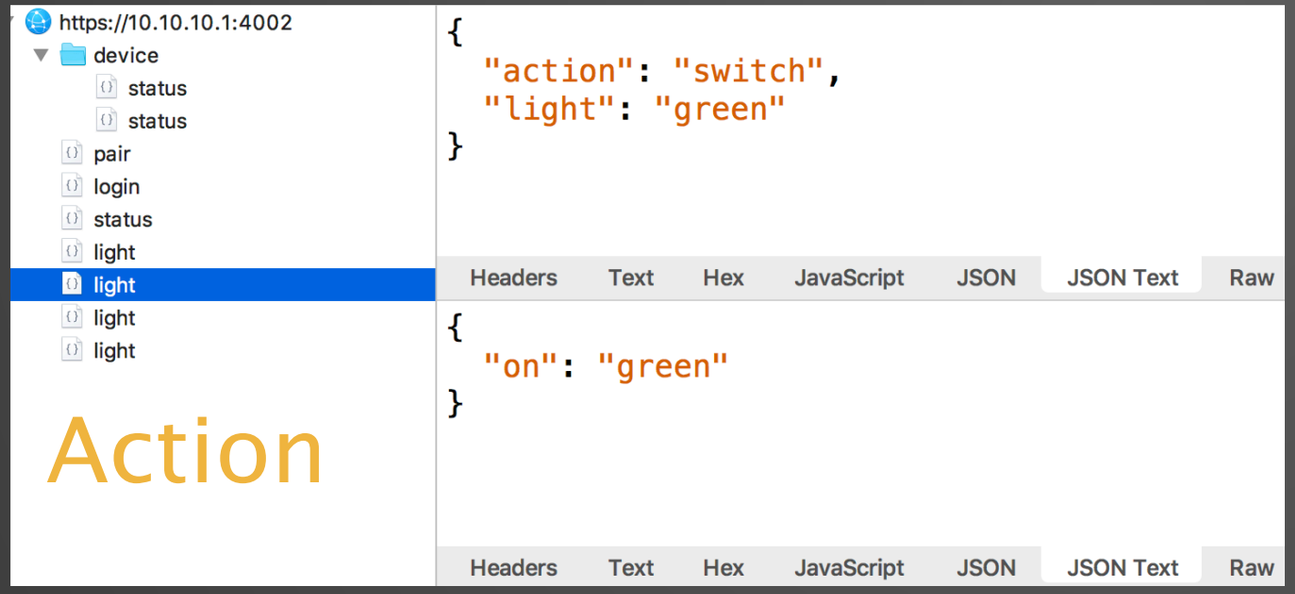
HTTPS with MitM



Login

Request

Response



Action

Request

Response

```
curl -k -d '{"action":"switch","light":"red"}' 'https://10.10.10.1:4001/light'
```

curl

HTTPS Issues

- Mallory can snoop* iPad traffic through legitimate use
  - Manipulate API for fun and profit
- Mallory can *control* the stoplight independently of the iPad
- MitM can snoop communications
  - Plaintext userid & password
  - Capture application data
  - Alter traffic

* Even if the iPad app uses public key pinning

Diffie–Hellman

Chuck and Sara agree to use $g = 2,\ N = 13$ (public)

Chuck			Sara
Random	$a = 5$		$b = 8$ Random
Computes	$A = g^a \% N$		$B = g^b \% N$ Computes
	$A = 2^5 \% 13$		$B = 2^8 \% 13$
	$A = 6$		$B = 9$
		6 →	
		← 9	
	$K = B^a \% N$		$K = A^b \% N$
	$K = 9^5 \% 13$		$K = 6^8 \% 13$
	$K = 3$		$K = 3$

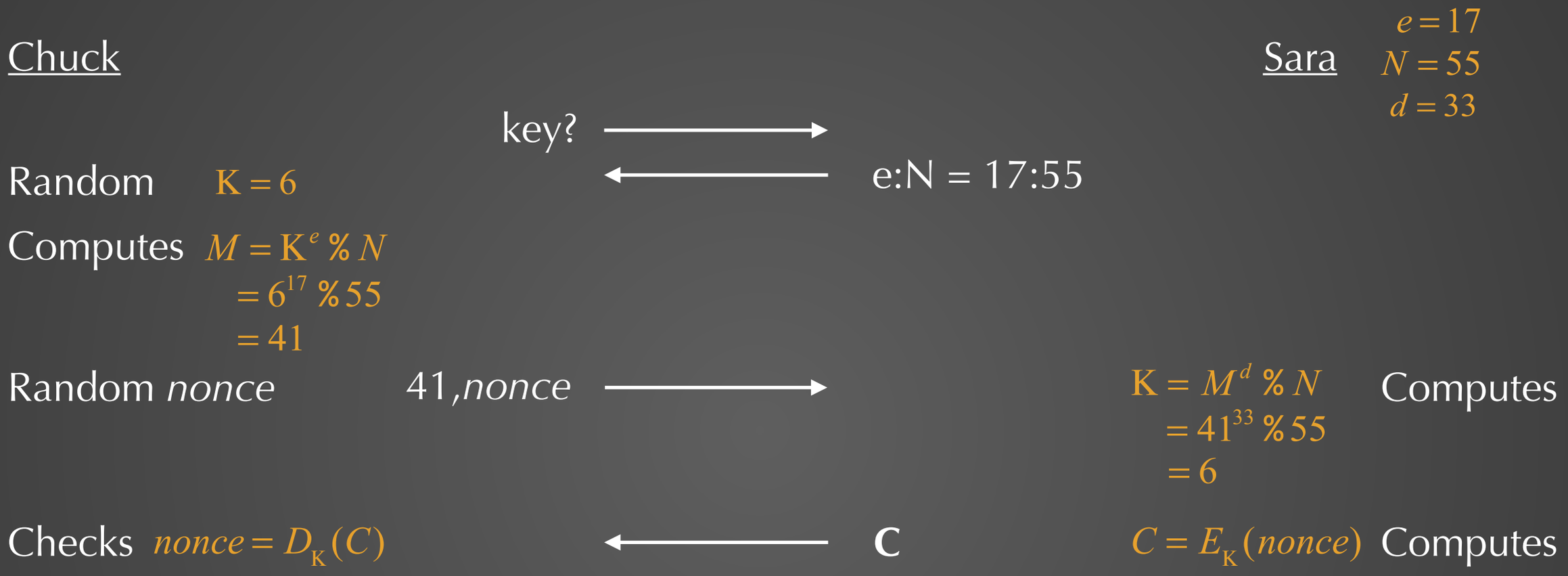
- Chuck and Sara *establish* $K = 3$ via *key agreement*
- If Eve knows g, N and captures 6,9 it is difficult* to calculate K
- Unauthenticated key agreement
  - Susceptible to Man-in-the-Middle (MitM) attack

* The DH problem is linked to the discrete logarithm problem

RSA (simplified)

- Sara creates a key pair
  - Chooses $p = 5, q = 11 \Rightarrow N = pq = 55$
    & $\varphi(N) = (p-1)(q-1) = 40$
  - Chooses $e = 17$, finds d where $(e \cdot d)\ \%\ \varphi(N) = 1$
    $\Rightarrow (17d)\ \%\ 40 = 1 \quad \Rightarrow d = 33$
    - Public key: $e : N = 17 : 55$; Private key: $d : N = 33 : 55$
- Why do this?
  - Because for $2 < \mathrm{K} < N - 1$, $\left(\mathrm{K}^e\ \%N\right)^d\ \%N = \mathrm{K}$
    - Exponentiating by e is encryption; exponentiating by d is decryption
  - This scheme is computationally expensive (i.e. slow)
    - Use for key establishment (and digital signatures)

RSA (simplified)



- Chuck and Sara *establish* $K = 6$ via *key transport*
  - They could use a DH *key agreement* instead

Structure

- $e = 17,\ d = 33,\ N = 55$ represents a binding relationship through

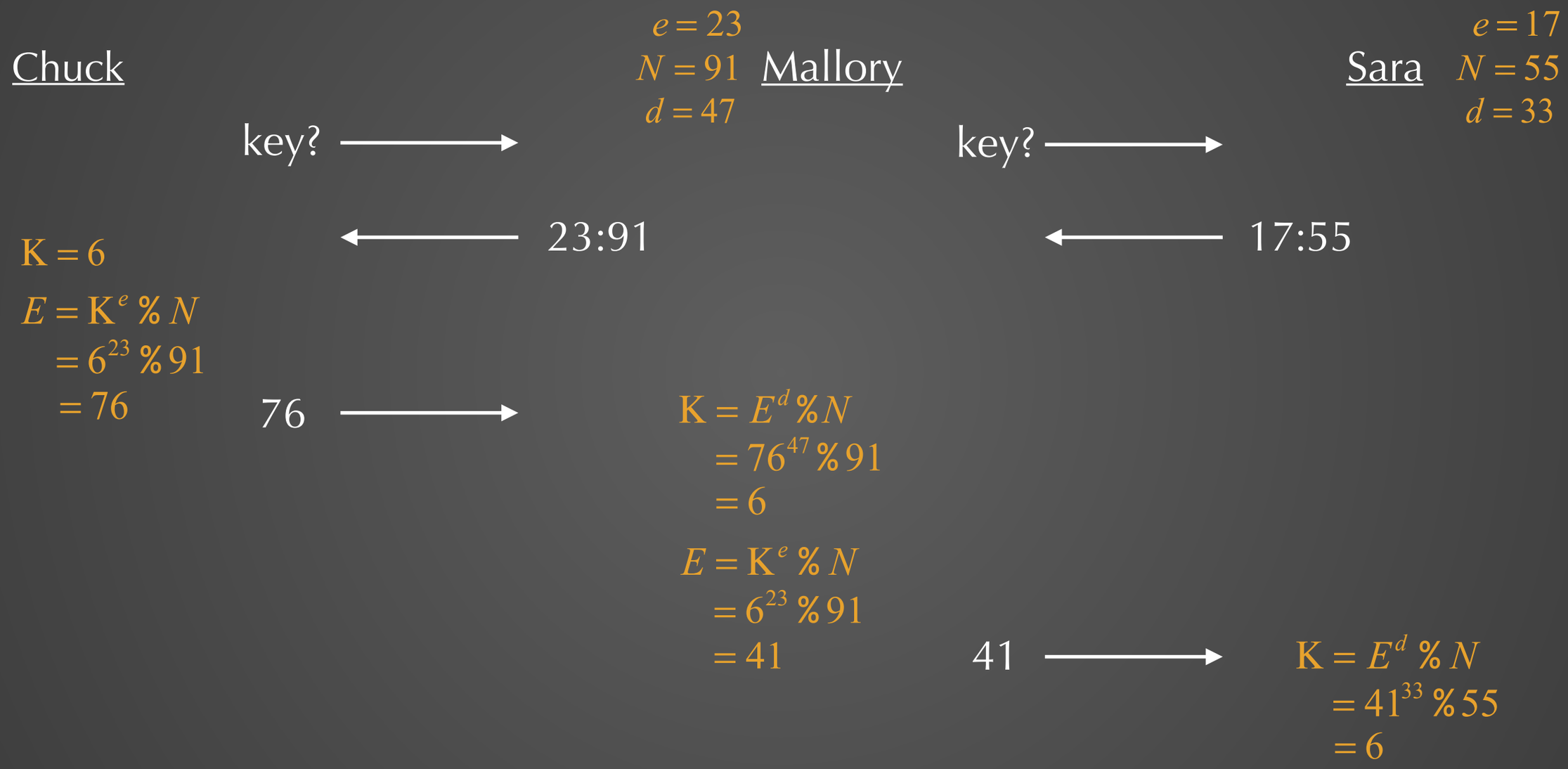
$$(e \cdot d)\ \%\ \varphi(N) = 1$$

- Given $e : N$ it is difficult* to determine d
- Chuck trusts $e : N$ belongs to Sara
  - Either explicitly or via transfer of trust
- Sara trusts she want to “talk” to Chuck
- Open system
  - Multiple entity trust model
  - No single control of who “talks” to whom

* The RSA problem is linked to the integer factorization problem

Man-in-the-Middle

Mallory creates a key pair: $p = 7,\ q = 13,\ N = 91,\ e = 23,\ d = 47$



- Mallory knows the value of the Chuck and Sara secret

CIA Document

Network Operations Division Cryptographic Requirements[1]

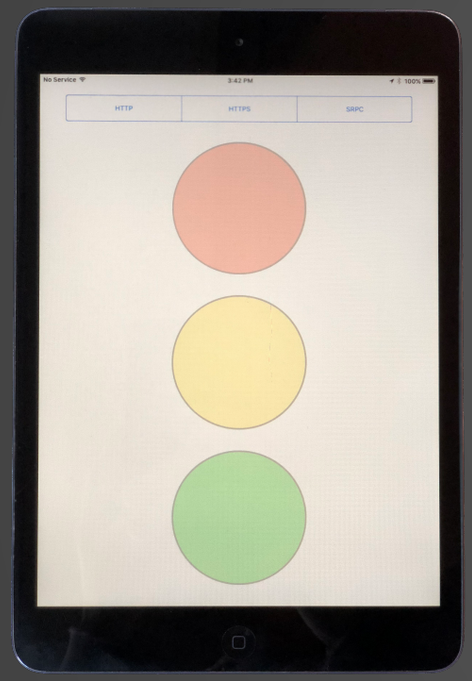
Section 2.4, 19

Any transport layer encryption[2] **must** be layered over the cryptography[3] discussed in this document. Because this outer layer may be decrypted by an attacker (e.g., SSL Man-in-the-Middle) any transport encryption **must** be used for traffic blending[4] only and not for secrecy.

1. Top-secret classified document obtained by Wikileaks through Shadow Brokers
2. TLS (e.g. HTTPS)
3. Application level security
4. Request metadata leaks no information

Scenario

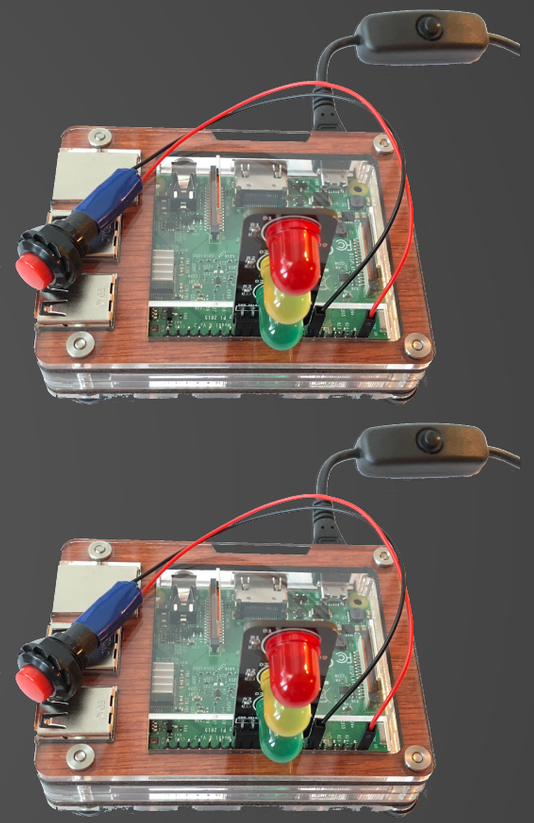
iPad controls stop light on an RPi3 device

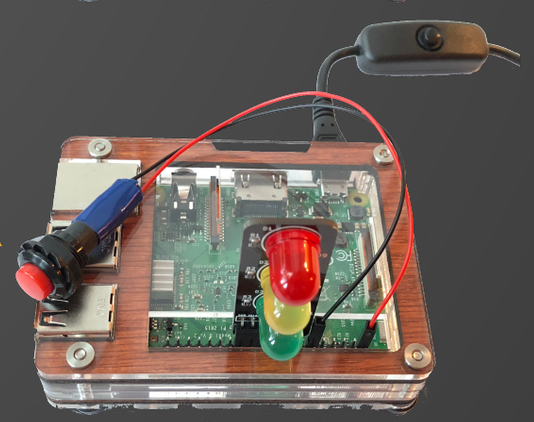


HTTP

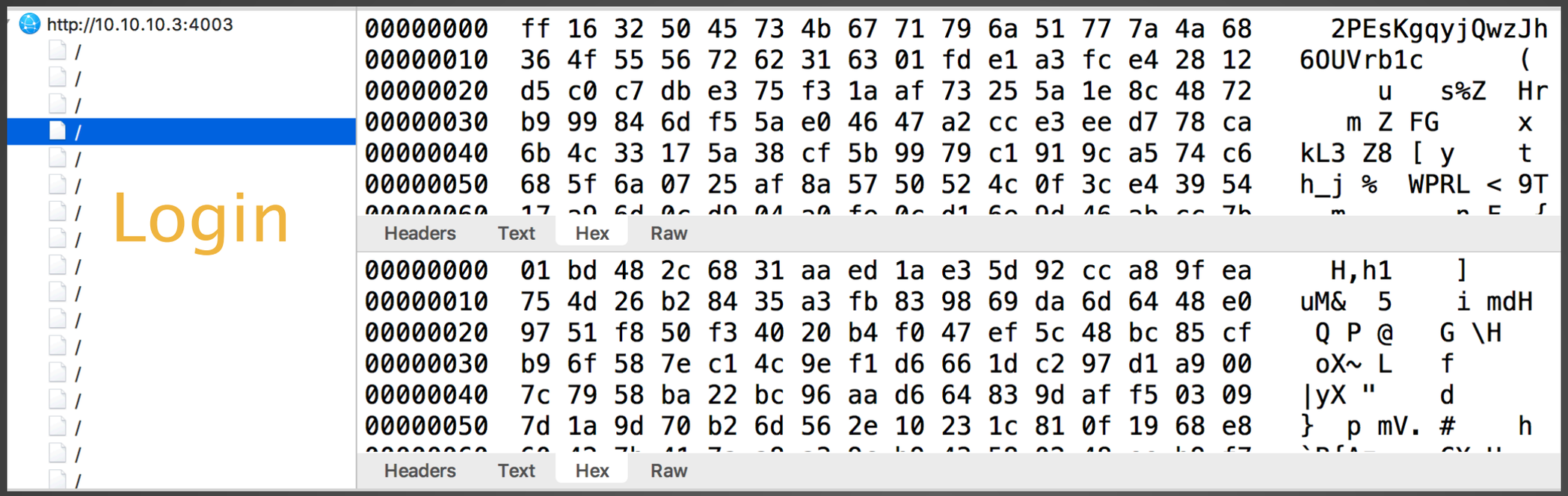
HTTPS

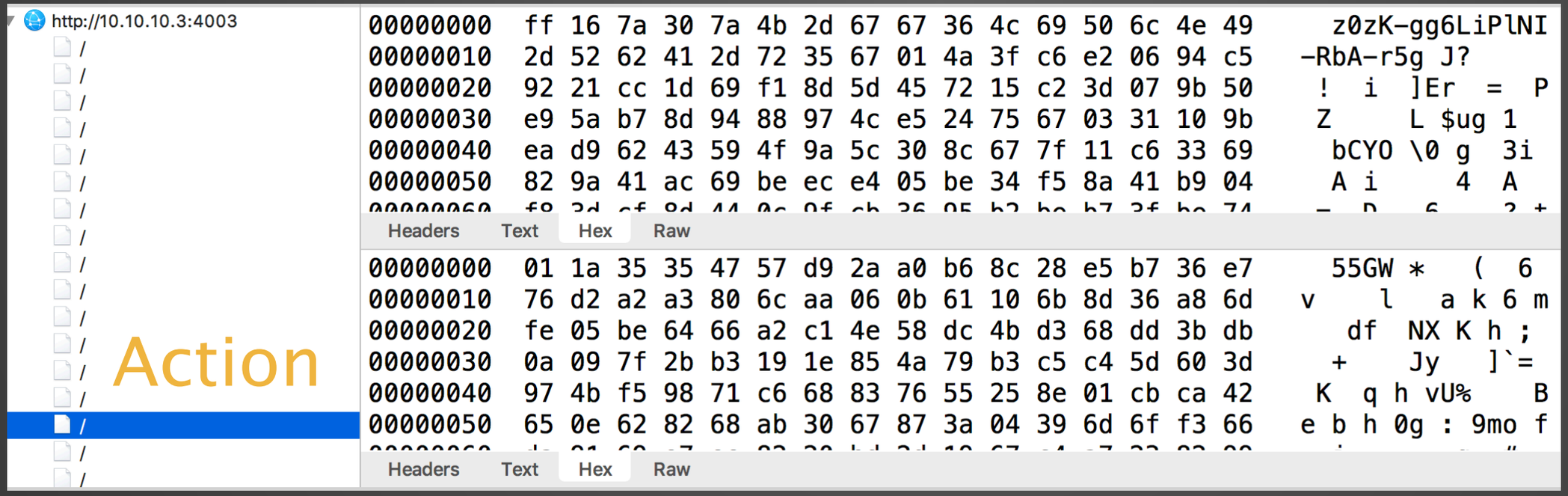
SRPC





SRPC






```
curl -d '{"action":"switch","light":"red"}' 'http://10.10.10.3:4001/light'
```

~~curl~~

SRP (simplified)

Chuck and Sara agree to use $g = 2, N = 13$ (public)

- Chuck chooses $x = 5$ and calculates $v = g^x \,\%\, N = 2^5 \,\%\, 13 = 6$
  - Chuck keeps $x = 5$ (secret) and gives $v = 6$ to Sara (verifier)

$x = 5$ Chuck			Sara $v = 6$
Random	$a = 11$		$b = 4$ Random
Computes	$A = g^a \,\%\, N = 2^{11} \,\%\, 13 = 7$	Id,7 →	
		← 9	$B = v + g^b \,\%\, N = 6 + 2^4 \,\%\, 13 = 9$ Computes
Computes	$\mathrm{K} = (B - g^x)^{a+x} \,\%\, N = (9 - 2^5)^{11+5} \,\%\, 13 = 3$		$\mathrm{K} = (Av)^b \,\%\, N = (7 \cdot 6)^4 \,\%\, 13 = 3$ Computes
Computes	$C_c = H(A \mid B \mid \mathrm{K}_c)$	Cc →	$C_c = H(A \mid B \mid \mathrm{K}_s)$ Checks
Checks	$C_s = H(A \mid C_c \mid \mathrm{K}_c)$	← Cs	$C_s = H(A \mid C_c \mid \mathrm{K}_s)$ Computes

- Chuck and Sara *establish* $\mathrm{K} = 3$ via *key agreement*

Structure

- $x = 5, v = 6, g = 2, N = 13$ represents a binding relationship through

$$v = g^x \,\%\, N$$

- Given $v : g : N$ it is difficult* to determine x
- Chuck and Sara *only trust each other*
  - Each holds a participating piece of the binding relationship
- Closed system
  - Single entity trust model
  - Explicit control over who “talks” to whom

* The SRP problem is linked to the Diffie-Hellman problem which is linked to the discrete logarithm problem

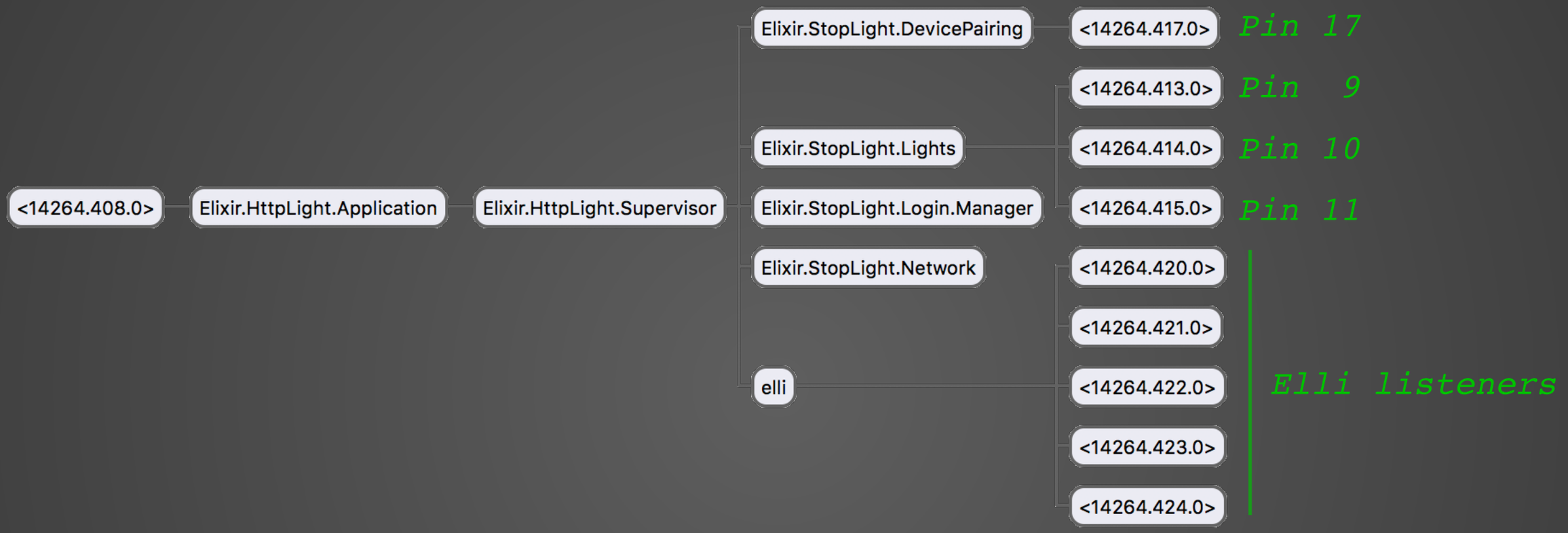
SRP & ALS

- Mutual client/server authentication
  - Ensures each “side” is communicating with a desired party
- Mutual user authentication
  - Ensures application is communicating on behalf of specific user
- Authentication via zero knowledge proof
  - Password never leaves client device
- Authentication and encryption in same layer
- Single entity trust mode
- Security orthogonal to API

SRPC

- Ephemeral keys with flexible session key management
    - Client initiated key refresh (manual or auto)
    - Auto-reconnect on first stale request (timed out session)
- Client-side key stretching
- Traffic blending (transport independent)
    - All request have same endpoint and meta-data (headers, etc.)
- Replay protection
    - Via nonces and (optional) timestamps
- Unicode normalization
    - Support credentials like τệ𝜎𝝉@𝛋ñöxện.î🤓 / ∂ïη𝛤𝜔Σ𝑲¥

RPi3 Application Structure




```
config :stop_light, :elli,
  port: 4003,
  stack: [
    {SrpcElli.ElliHandler, []},
    {StopLight.Elli.StatusHandler, []},
    {HttpLight.Elli.LoginHandler, []},
    {StopLight.Elli.LightsHandler, []}
  ]
```

Key Exchange Options

- PAKE
  - SRP is an augmented PAKE
  - A slew of others
- Install server public key in client (and vice-versa)
  - Needham-Schroeder
  - Station-to-Station
  - A slew of others
- A slew of others

Is Security on Your Nerves?

Demo: GitHub Knoxen/SrpcWorld

Presentation: GitHub Knoxen/SecurityNerves

Paul Rogers

paul@knoxen.com