

Is Security on Your Nerves?



Paul Rogers
paul@knoxen.com

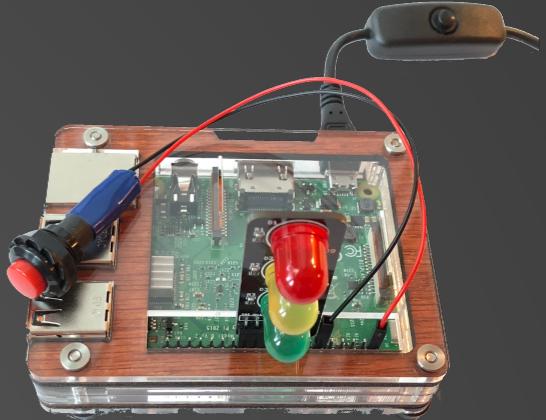
Scenario

iPad controls stop light
on an RPi3 device

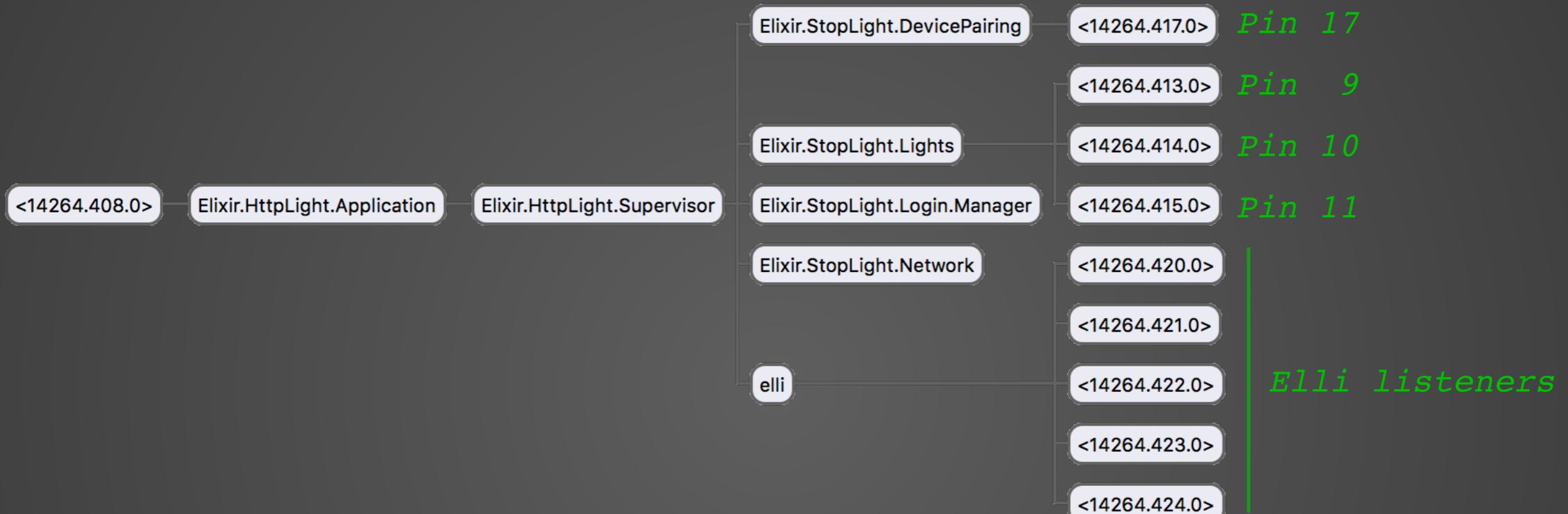


System Overview

- 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

Headers Text Hex JavaScript JSON JSON Text Raw

```
{ "password": "secret", "username": "http" }
```

Request

Action

Headers Text Hex JavaScript JSON JSON Text Raw

```
{ "action": "switch", "light": "green" }
```

Request

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

curl

HTTP Issues

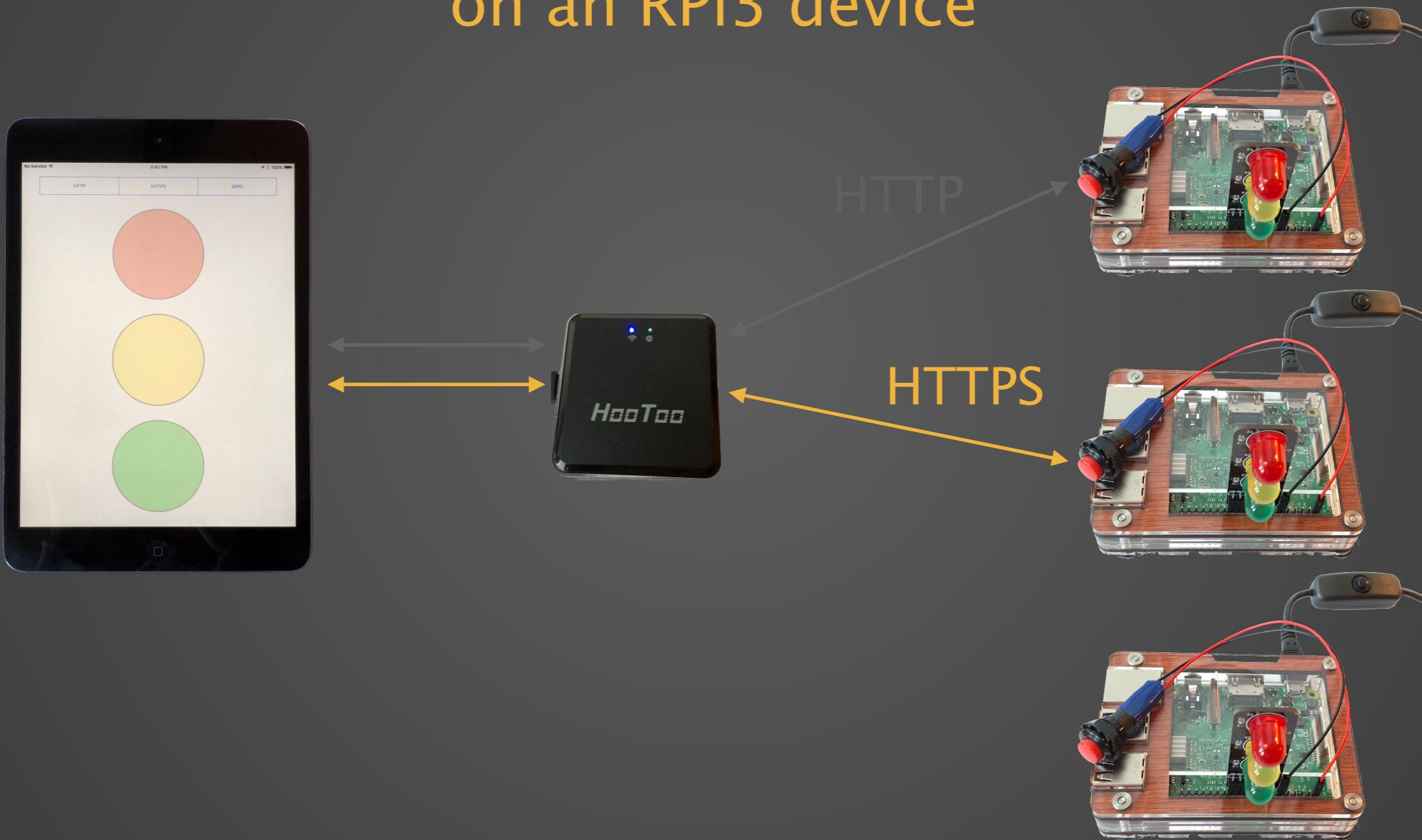
- Eve can snoop communications
 - Plaintext userid & password
 - Capture application data
- Mallory can snoop communications
 - Alter traffic
 - Manipulate API for fun and profit
- Mallory can *control* the stoplight *independently* of the iPad

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



HTTPS

10.10.10.1:4002		00000000	16 03 01 00 d5 01 00 00 d1 03 03 5a 82 04 68 cc		Z h
	<unknown>	00000010	22 b3 ec d5 66 d3 3e 90 3a 11 b7 87 9d ab 85 75	" f > :	u
	<unknown>	00000020	51 ba ff 98 35 9b 69 e3 3f 49 44 00 00 2c 00 ff	Q 5 i ?ID ,	,
	<unknown>	00000030	c0 2c c0 2b c0 24 c0 23 c0 0a c0 09 c0 08 c0 30	, + \$ #	0
	<unknown>	00000040	c0 2f c0 28 c0 27 c0 14 c0 13 c0 12 00 9d 00 9c	/ ('	
	<unknown>	00000050	00 3d 00 3c 00 35 00 2f 00 0a 01 00 00 7c 00 00	= < 5 /	
	<unknown>	00000060	00 0f 00 0d 00 00 0a 31 30 2e 31 30 2e 31 30 2e		10.10.10.
	<unknown>	00000070	31 20 20 20 20 20 20 20 17 20 10 20 20 20 20 20		
		Headers	Text	Hex	Raw
		00000000	16 03 03 00 57 02 00 00 53 03 03 00 00 02 d7 2d		W S -
		00000010	39 4a 2c 01 bf 18 57 67 ef d3 47 82 0c 59 0c 5b	9J, Wg G Y [[
		00000020	ff 9c 60 b7 a8 9a 07 38 01 65 5d 20 be 89 5b 0d	` 8 e] [
		00000030	0e e3 db 42 b2 d6 c8 dd a0 ad 69 05 dc c6 9a a8	B i	
		00000040	0a f5 6e 00 69 bb 61 de ea e7 a8 de c0 30 00 00	n i a	0
		00000050	2b 20 2b 20 22 21 20 ff 21 20 21 20 16 20 22 22 22		
		Headers	Text	Hex	Raw

Login

Request

Response

10.10.10.1:4002	00000000	16 03 01 00 d5 01 00 00 d1 03 03 5a 82 04 6e c3	Z n
<unknown>	00000010	c8 e2 3d 2a 8c 15 13 30 d3 70 bc e7 51 b6 fa 1a	=* 0 p Q
<unknown>	00000020	0c f2 01 41 ba 31 de 19 b2 7e 1d 00 00 2c 00 ff	A 1 ~ ,
<unknown>	00000030	c0 2c c0 2b c0 24 c0 23 c0 0a c0 09 c0 08 c0 30	, + \$ # 0
<unknown>	00000040	c0 2f c0 28 c0 27 c0 14 c0 13 c0 12 00 9d 00 9c	/ ('
<unknown>	00000050	00 3d 00 3c 00 35 00 2f 00 0a 01 00 00 7c 00 00	= < 5 /
<unknown>	00000060	00 0f 00 0d 00 00 0a 31 30 2e 31 30 2e 31 30 2e	10.10.10.
<unknown>	00000070	21 00 00 00 00 00 00 00 00 17 00 10 00 10 00 05 00	
	Headers	Text	Hex
	00000000	16 03 03 00 57 02 00 00 53 03 03 00 00 02 de c9	W S
	00000010	66 74 bb 09 4e f2 4d b6 8d 6a 21 ff 71 4d dc b9	ft N M j! qM
	00000020	88 19 63 0f 83 9e ba 2f 56 59 86 20 e6 53 a4 48	c /VY S H
	00000030	e7 28 38 cb 80 8e 65 81 98 2d 78 71 c6 b1 e7 45	(8 e -xq E
	00000040	21 33 84 9e e0 40 e1 0a 7a 44 65 ef c0 30 00 00	!3 @ zDe 0
	00000050	05 00 0b 00 02 01 00 ff 01 00 01 00 16 02 02 02 02	
	Headers	Text	Hex

Action

Request

Response

HTTPS with MitM

Login

The screenshot shows a browser interface with a sidebar navigation menu on the left. The menu items include device, status, status, pair, login (which is selected), status, light, light, light, and light. The main content area displays a JSON object with two fields: "password" and "username". Below the JSON object, there is a toolbar with tabs: Headers, Text, Hex, JavaScript, JSON, JSON Text, and Raw. The JSON tab is currently active.

```
{
  "password": "secret",
  "username": "https"
}
```

Request

Action

The screenshot shows a browser interface with a sidebar navigation menu on the left. The menu items include device, status, status, pair, login, status, light, light (which is selected), light, light, and light. The main content area displays a JSON object with two fields: "action" and "light". Below the JSON object, there is a toolbar with tabs: Headers, Text, Hex, JavaScript, JSON, JSON Text, and Raw. The JSON tab is currently active.

```
{
  "action": "switch",
  "light": "green"
}
```

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

Random $a = 5$

Computes $A = g^a \% N$

$$A = 2^5 \% 13$$

$$A = 6$$

$$K = B^a \% N$$

$$K = 9^5 \% 13$$

$$K = 3$$



Sara

$$b = 8$$

Random

$$B = g^b \% N$$

Computes

$$B = 2^8 \% 13$$

$$B = 9$$

$$K = A^b \% N$$

$$K = 6^8 \% 13$$

$$K = 3$$

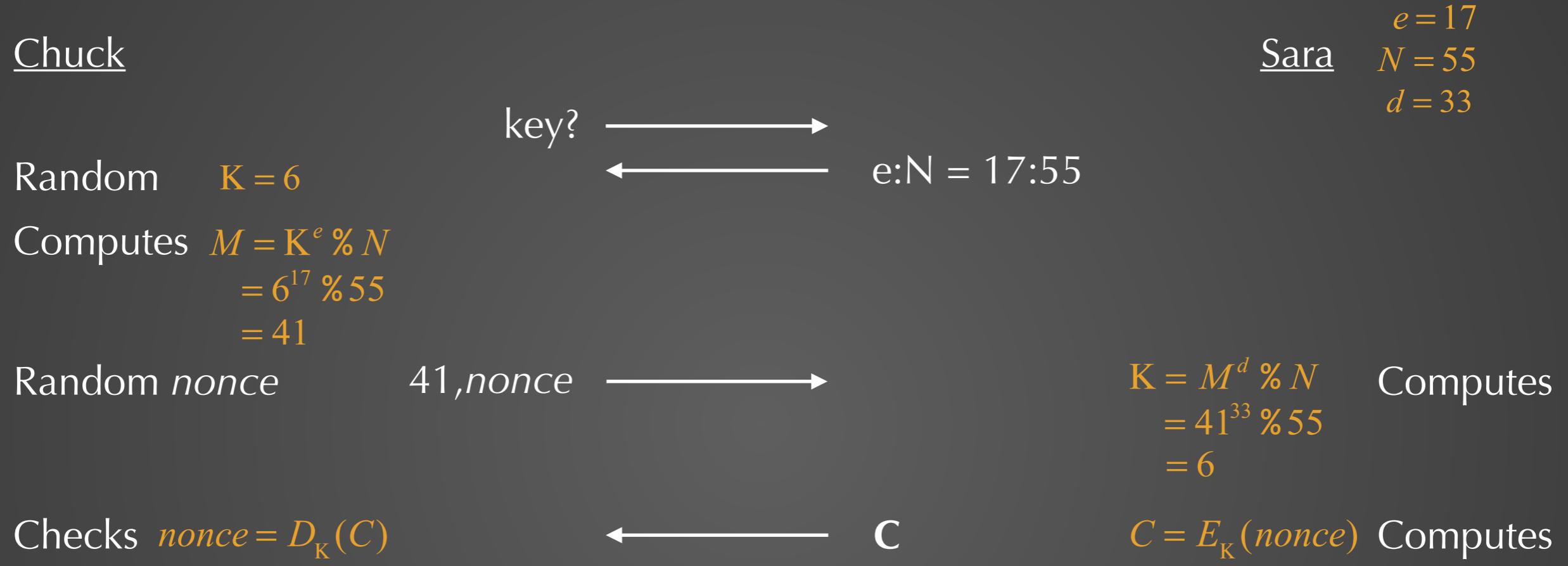
- Chuck and Sara establish $K = 3$ via *key agreement*
- If Eve knows g, N and captures 6,9 it is difficult* to calculate K
- Unauthenticated key agreement
 - Susceptible to Man-in-the-Middle (MitM) attack

* 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 $(e \cdot d) \% \varphi(N) = 1$
 $\Rightarrow (17d) \% 40 = 1 \Rightarrow d = 33$
 - Public key: $e : N = 17 : 55$; Private key: $d : N = 33 : 55$
- Why do this?
 - Because for $2 < K < N - 1$, $(K^e \% N)^d \% N = K$
 - 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 Requirements¹

Section 2.4, 19

Any transport layer encryption² **must** be layered over the cryptography³ 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 blending⁴ 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



SRPC

Login Request

http://10.10.10.3:4003

Headers	Text	Hex	Raw
00000000	ff 16 32 50 45 73 4b 67 71 79 6a 51 77 7a 4a 68	2PEsKgqyjQwzJh	
00000010	36 4f 55 56 72 62 31 63 01 fd e1 a3 fc e4 28 12	60UVrb1c (
00000020	d5 c0 c7 db e3 75 f3 1a af 73 25 5a 1e 8c 48 72	u s%Z Hr	
00000030	b9 99 84 6d f5 5a e0 46 47 a2 cc e3 ee d7 78 ca	m Z FG x	
00000040	6b 4c 33 17 5a 38 cf 5b 99 79 c1 91 9c a5 74 c6	kL3 Z8 [y t	
00000050	68 5f 6a 07 25 af 8a 57 50 52 4c 0f 3c e4 39 54	h_j % WPRL < 9T	
00000060	17 20 5d 0c d0 04 20 fc 0c d1 5c 0d 1f 2b cc 7b	m n E f	
Headers	Text	Hex	Raw
00000000	01 bd 48 2c 68 31 aa ed 1a e3 5d 92 cc a8 9f ea	H, h1]	
00000010	75 4d 26 b2 84 35 a3 fb 83 98 69 da 6d 64 48 e0	uM& 5 i mdH	
00000020	97 51 f8 50 f3 40 20 b4 f0 47 ef 5c 48 bc 85 cf	Q P @ G \H	
00000030	b9 6f 58 7e c1 4c 9e f1 d6 66 1d c2 97 d1 a9 00	oX~ L f	
00000040	7c 79 58 ba 22 bc 96 aa d6 64 83 9d af f5 03 09	yX " d	
00000050	7d 1a 9d 70 b2 6d 56 2e 10 23 1c 81 0f 19 68 e8	} p mV. # h	
00000060	60 42 7b 41 7c 42 02 40 42 50 02 40 42 50 02 40	^P^A^G^S^C^U^H	
Headers	Text	Hex	Raw

Action Request

http://10.10.10.3:4003

Headers	Text	Hex	Raw
00000000	ff 16 7a 30 7a 4b 2d 67 67 36 4c 69 50 6c 4e 49	z0zK-gg6LiPlNI	
00000010	2d 52 62 41 2d 72 35 67 01 4a 3f c6 e2 06 94 c5	-RbA-r5g J?	
00000020	92 21 cc 1d 69 f1 8d 5d 45 72 15 c2 3d 07 9b 50	! i]Er = P	
00000030	e9 5a b7 8d 94 88 97 4c e5 24 75 67 03 31 10 9b	Z L \$ug 1	
00000040	ea d9 62 43 59 4f 9a 5c 30 8c 67 7f 11 c6 33 69	bCY0 \0 g 3i	
00000050	82 9a 41 ac 69 be ec e4 05 be 34 f5 8a 41 b9 04	A i 4 A	
00000060	f0 2d cf 0d 44 0c 0f cb 26 05 b2 b0 b7 2f b0 74	- D f ? +	
Headers	Text	Hex	Raw
00000000	01 1a 35 35 47 57 d9 2a a0 b6 8c 28 e5 b7 36 e7	55GW * (6	
00000010	76 d2 a2 a3 80 6c aa 06 0b 61 10 6b 8d 36 a8 6d	v l a k 6 m	
00000020	fe 05 be 64 66 a2 c1 4e 58 dc 4b d3 68 dd 3b db	df NX K h ;	
00000030	0a 09 7f 2b b3 19 1e 85 4a 79 b3 c5 c4 5d 60 3d	+ Jy]`=	
00000040	97 4b f5 98 71 c6 68 83 76 55 25 8e 01 cb ca 42	K q h vU% B	
00000050	65 0e 62 82 68 ab 30 67 87 3a 04 39 6d 6f f3 66	e b h 0g : 9mo f	
00000060	1 01 00 07 00 00 00 00 00 00 00 00 00 00 00 00	"	
Headers	Text	Hex	Raw

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

Request

Response

Action

Request

Response

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

Random $a = 11$

Computes $A = g^a \% N$
 $= 2^{11} \% 13$
 $= 7$



Computes $K = (B - g^x)^{a+x} \% N$
 $= (9 - 2^5)^{11+5} \% 13$
 $= 3$

Computes $C_c = H(A | B | K_c)$

Checks $C_s = H(A | C_c | K_c)$

$b = 4$

$B = v + g^b \% N$
 $= 6 + 2^4 \% 13$
 $= 9$

Sara $v = 6$

Random

Computes

$K = (Av)^b \% N$
 $= (7 \cdot 6)^4 \% 13$
 $= 3$

Computes

$C_c = H(A | B | K_s)$

$C_s = H(A | C_c | K_s)$

Checks

Computes

- Chuck and Sara establish $K = 3$ 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

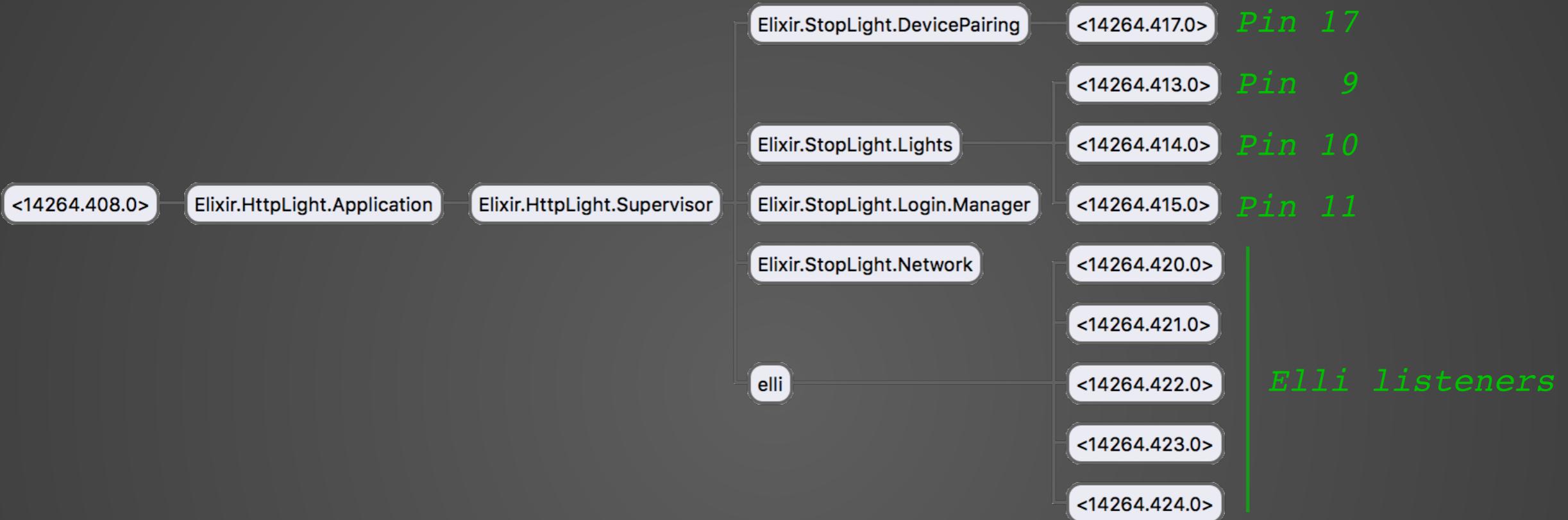
SRP & ALS

- 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 requests have same endpoint and meta-data (headers, etc.)
- Replay protection
 - Via nonces and (optional) timestamps
- Unicode normalization
 - Support credentials like $\tau\hat{\epsilon}\sigma\tau@\kappa\tilde{n}\ddot{o}\times\hat{e}n.\hat{i}\hat{\wedge}\hat{\wedge}$ / $\partial\ddot{\epsilon}\eta\Gamma\omega\Sigma K\ddot{\epsilon}$

RPi3 Application Structure



```
config :stop_light, :elli,
  port: 4003,
  stack: [
    {Srpcl Elli.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