

# Is Security on Your Nerves?



Paul Rogers

*[paul@knoxen.com](mailto:paul@knoxen.com)*

# Scenario

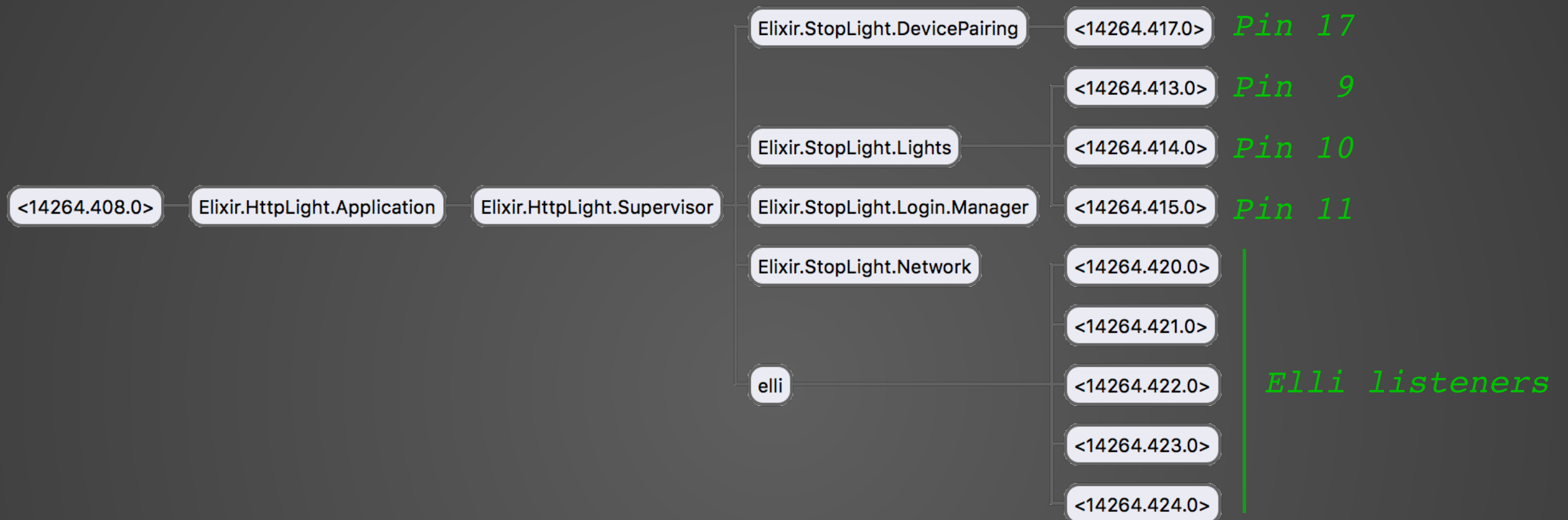
iPad controls stop light  
on an RPi3 device



# 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, []}
  ]
```

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

# HTTP

http://10.10.10.2:4001

- device
  - status
  - status
  - pair
  - login
  - status
  - light
  - light
  - light
  - light

Login

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

Headers Text Hex JavaScript JSON JSON Text Raw

```
{  "status": "ok"}  
```

Headers Text Hex JavaScript JSON JSON Text Raw

Request

Response

http://10.10.10.2:4001

- device
  - status
  - status
  - pair
  - login
  - status
  - light
  - light
  - light

Action

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

Headers Text Hex JavaScript JSON JSON Text Raw

```
{  "on": "green"}  
```

Headers Text Hex JavaScript JSON JSON Text Raw

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







# HTTPS with MitM

https://10.10.10.1:4002

- device
  - status
  - status
  - pair
  - login**
  - status
  - light
  - light
  - light
  - light

Login

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

Headers Text Hex JavaScript JSON JSON Text Raw

Request

Response

https://10.10.10.1:4002

- device
  - status
  - status
  - pair
  - login
  - status
  - light
  - light**
  - light
  - light

Action

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

Headers Text Hex JavaScript JSON JSON Text Raw

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

Random  $b = 8$   
Computes  $B = g^b \% N$   
 $B = 2^8 \% 13$   
 $B = 9$

$K = A^b \% N$   
 $K = 6^8 \% 13$   
 $K = 3$



- Chuck and Sara *establish*  $K = 3$  via *key agreement*
- If Eve knows  $g, N$  and captures 6,9 it is difficult\* to calculate  $K$
- Anonymous 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

# RSA (simplified)

# Chuck

Sara  $e = 17$   
 $N = 55$   
 $d = 33$

Random       $K = 6$

key?  $\longleftrightarrow$  e:N = 17:55

Computes  $M = K^e \% N$   
 $= 6^{17} \% 55$   
 $= 41$

Random *nonce*      41,*nonce*       $\longrightarrow$

$$\begin{aligned} K &= M^d \% N \\ &= 41^{33} \% 55 \\ &= 6 \end{aligned}$$

Checks  $\text{nonce} = D_K(C)$  ← C

$C = E_K(\text{nonce})$  Computes

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



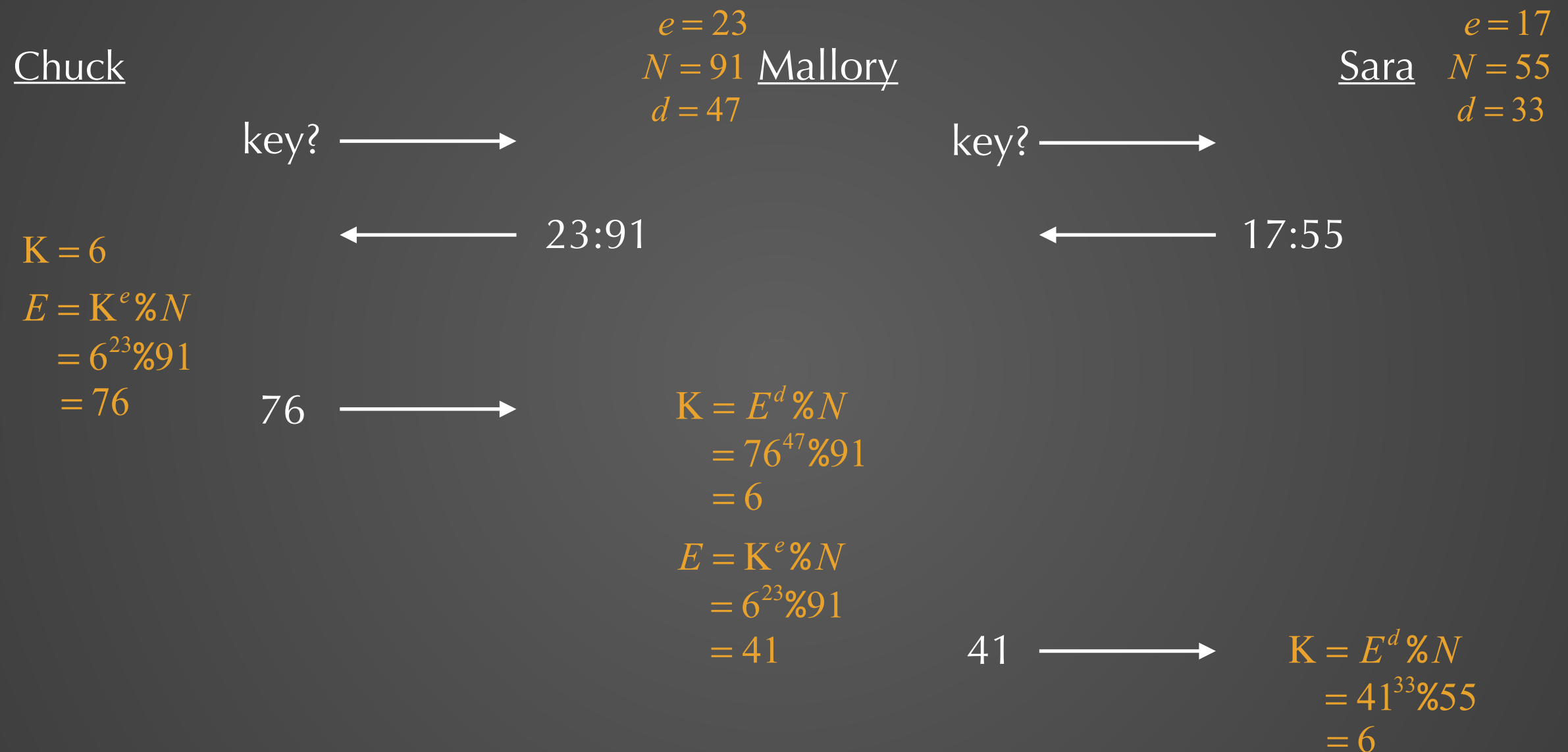
# Structure

- $e = 17, d = 33, N = 55$  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<sup>1</sup>

Section 2.4, 19

Any transport layer encryption<sup>2</sup> **must** be layered over the cryptography<sup>3</sup> 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<sup>4</sup> 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	http://10.10.10.3:4003	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 6d 0c d0 04 20 f0 0c d1 60 0d 46 2b 0c 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

Request

Response

Action	http://10.10.10.3:4003	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 8d 44 0c 0f cb 26 05 b2 b0 b7 2f b0 74	- D 6 2 +
	/	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

Request

Response

```
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

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

Id, 7  $\xrightarrow{\hspace{1cm}}$   
 $\xleftarrow{\hspace{1cm}}$  9

$C_c$   $\xrightarrow{\hspace{1cm}}$   
 $\xleftarrow{\hspace{1cm}}$   $C_s$

Sara  $v = 6$

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

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

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

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

- 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
  - All request have same endpoint and meta-data (headers, etc.)
- Replay protection
  - Via nonces and (optional) timestamps
- Unicode normalization
  - Support credentials like τêστ@κñöxên.î🧐 / ∂ïηΓωΣK¥

# 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*