Ensuring High-Quality Randomness in Cryptographic Key Generation

Henry Corrigan-Gibbs, Wendy Mu, Dan Boneh - *Stanford*

Bryan Ford - *Yale*

20th ACM Conference on Computer and Communications Security
6 November 2013

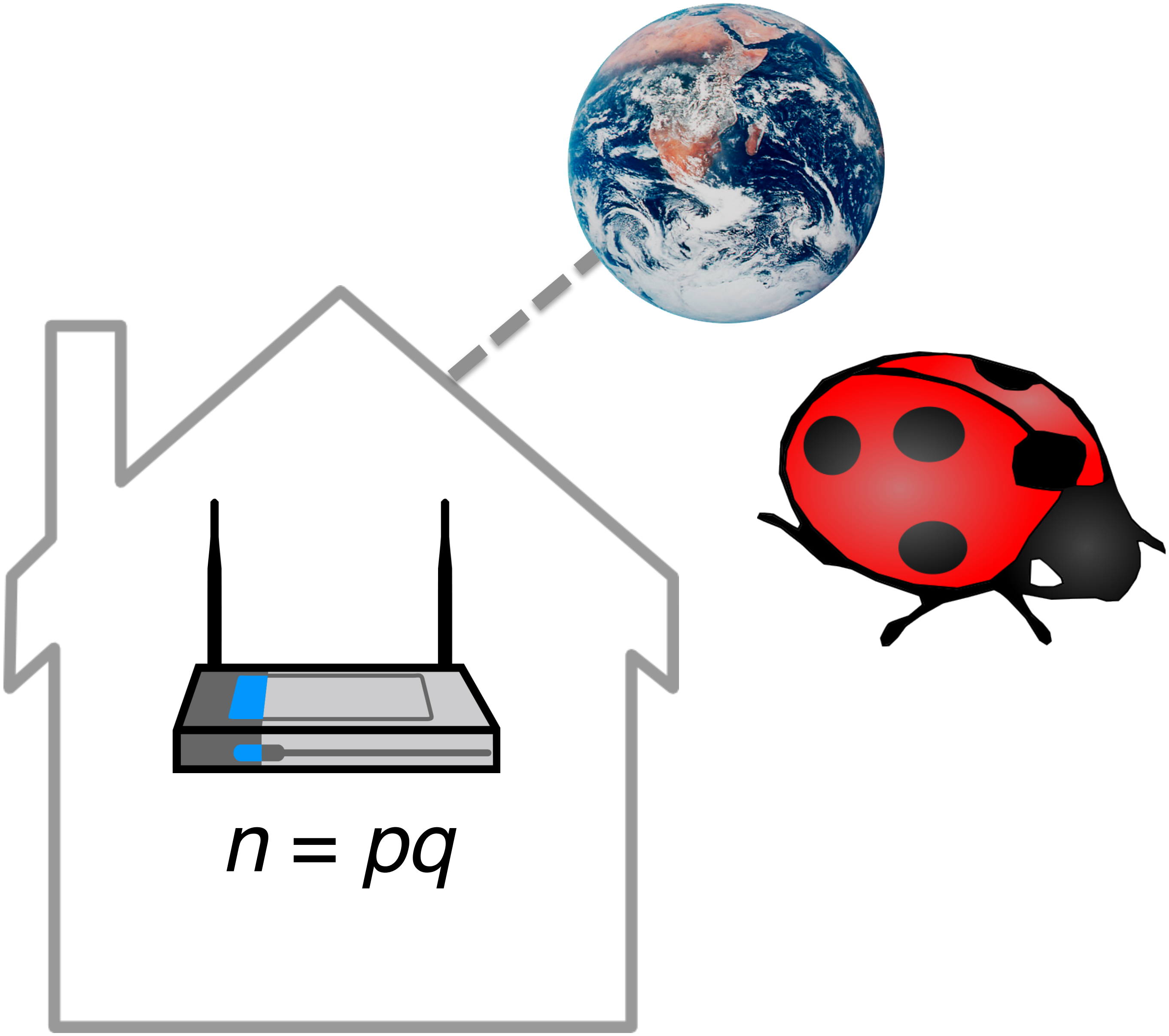
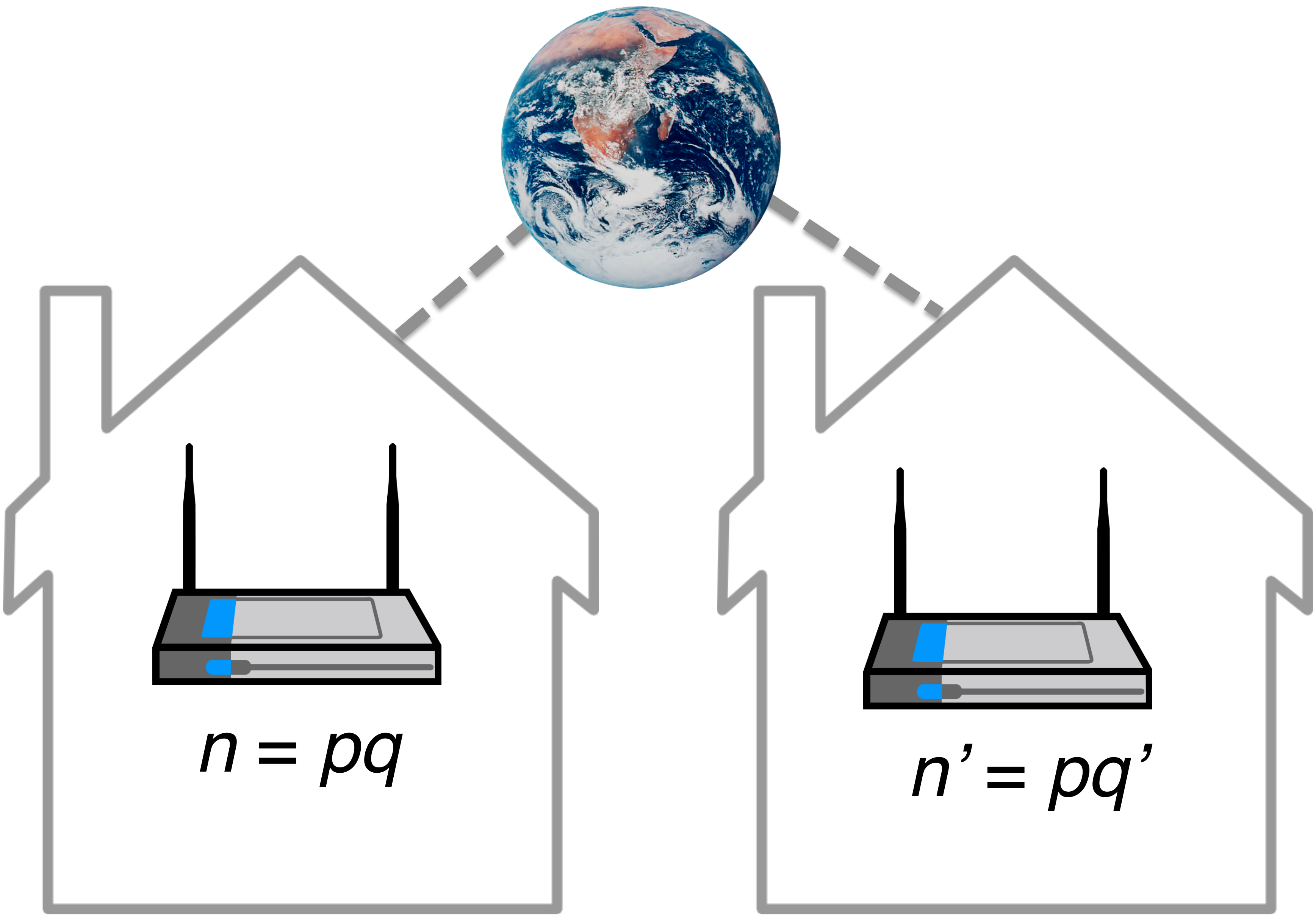
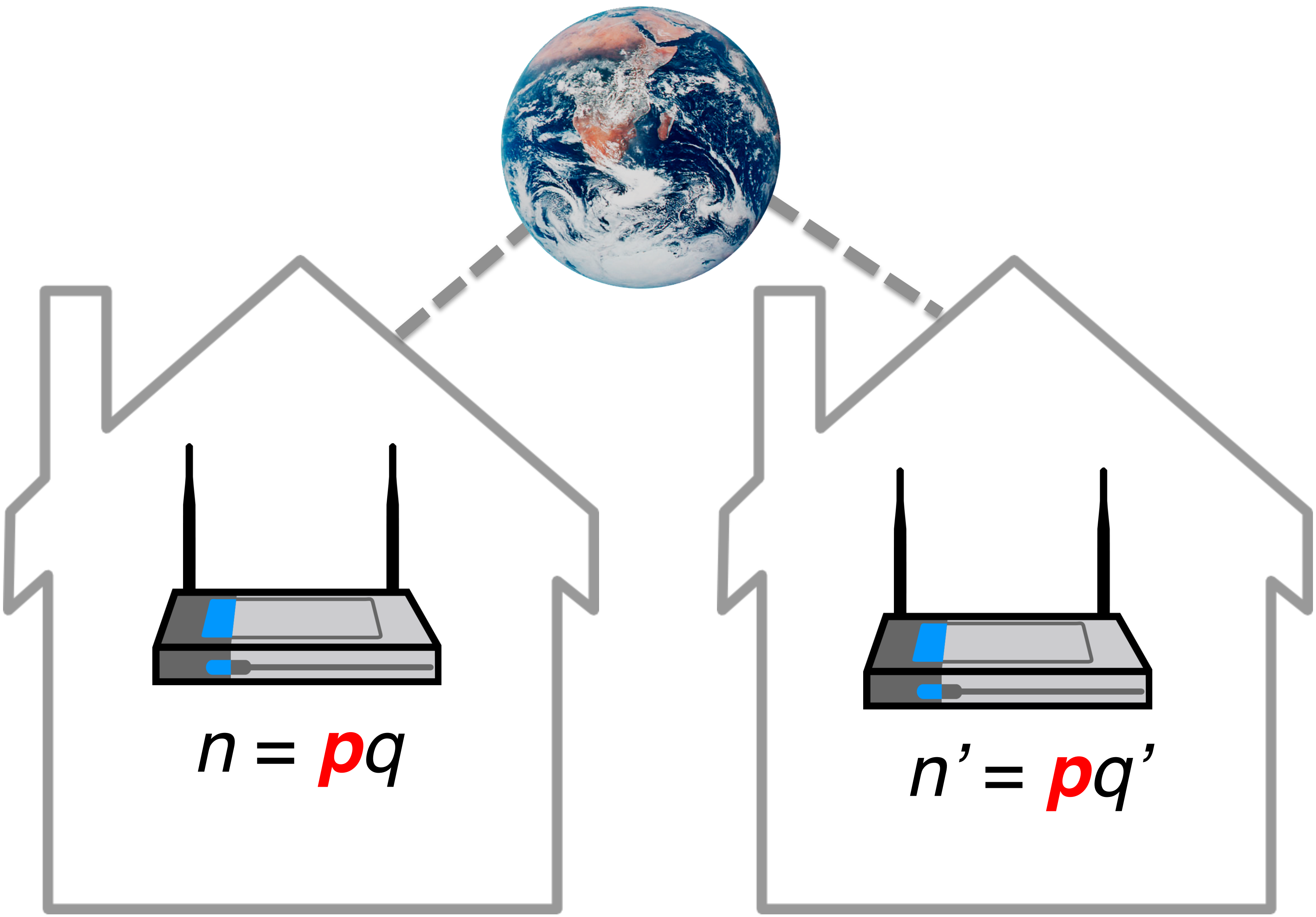


Image courtesy NASA Johnson Space Center

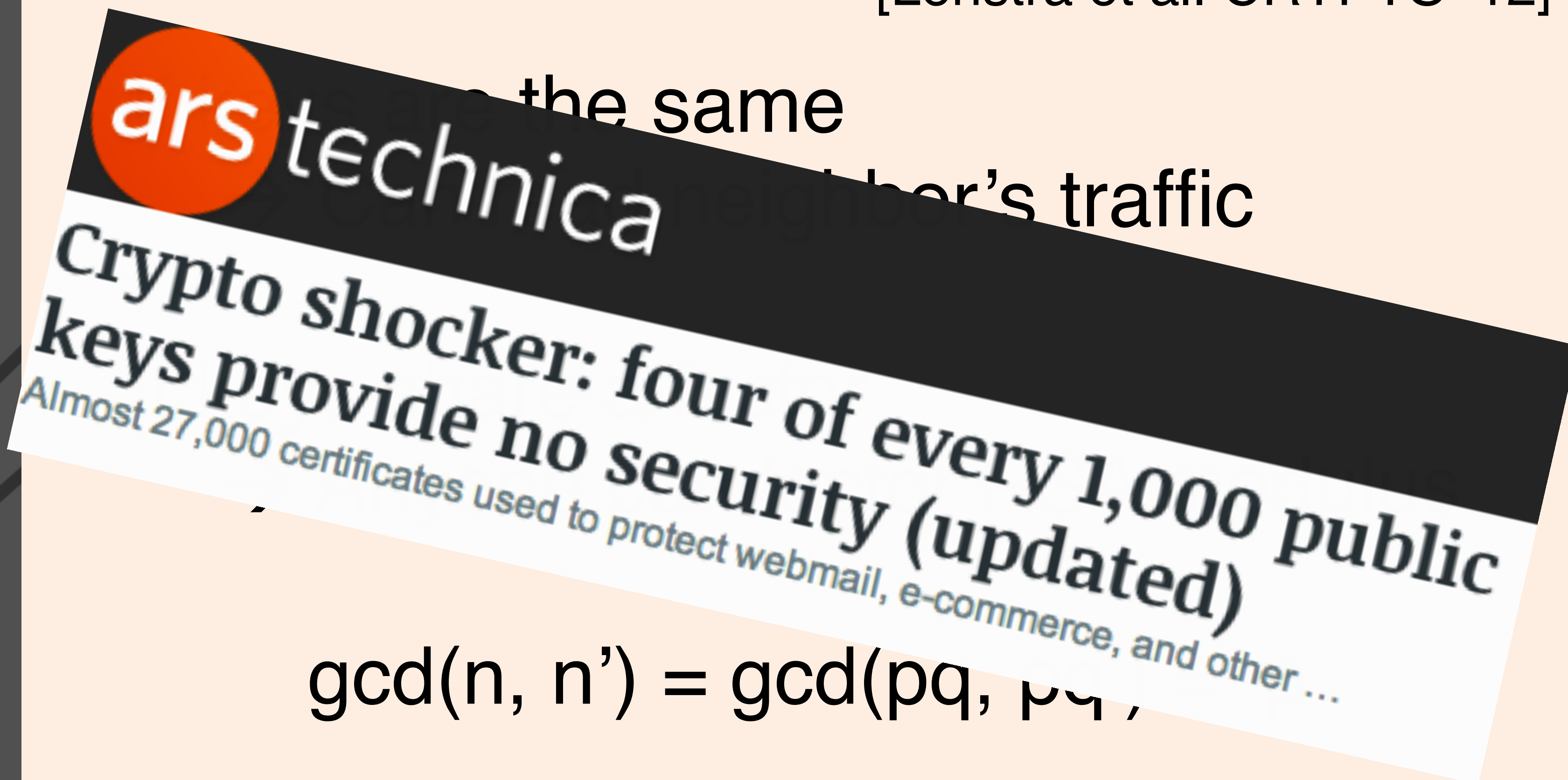


n = pq
n' = pq'



n = pq
n' = pq'

[Heninger et al. USENIX Sec '12]
[Lenstra et al. CRYPTO '12]



ars technica

Crypto shocker: four of every 1,000 public keys provide no security (updated)

Almost 27,000 certificates used to protect webmail, e-commerce, and other ...

...s are the same

...neighbor's traffic

gcd(n, n') = gcd(pq, pq'...

100,000s of vulnerable keys

Common Failure Modes

1) App never reads strong random values

```
bytes = Hash(time(), get_pid(), pwd);
```

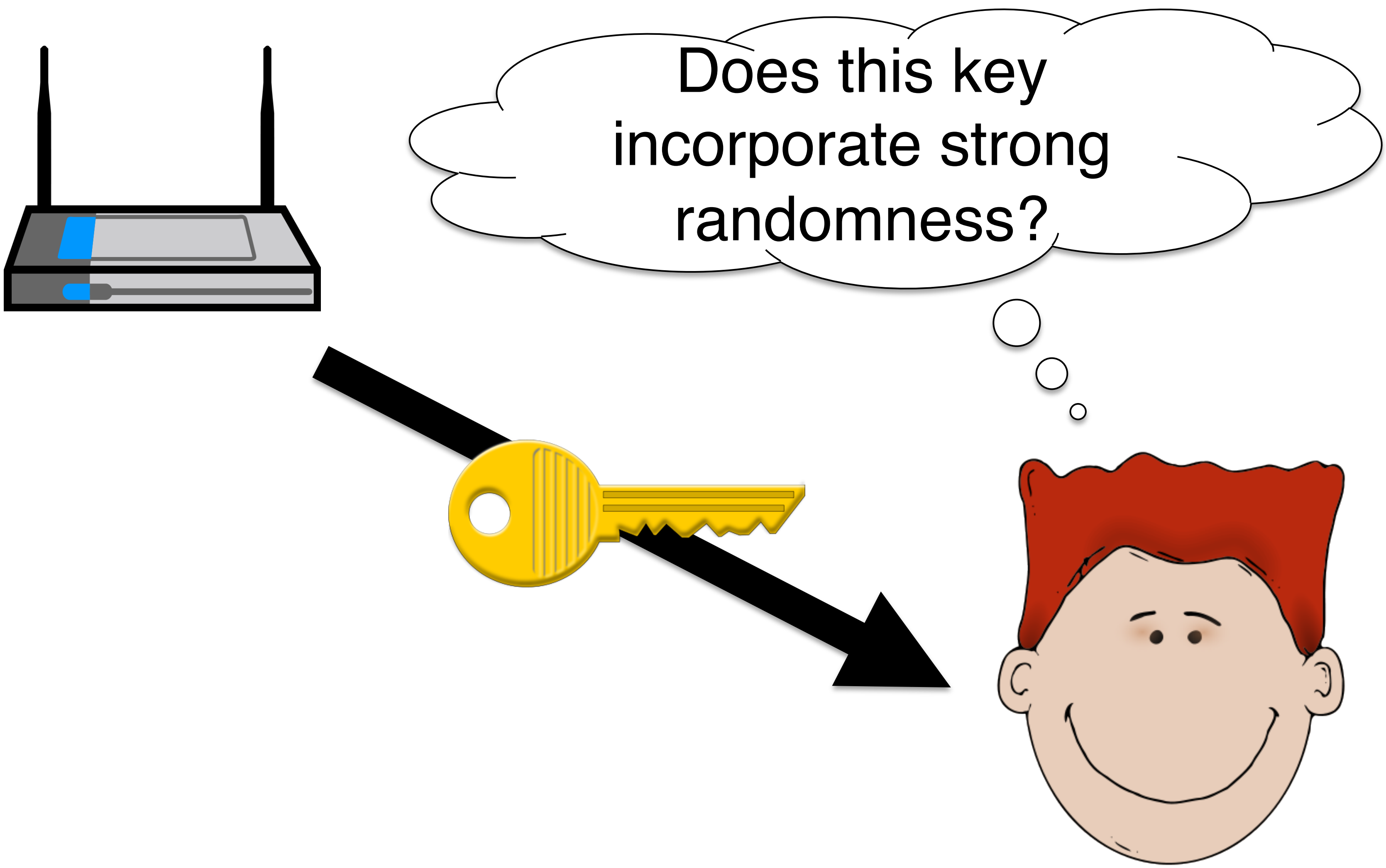
[CVE-2001-0950, CVE-2001-1467, CVE-2005-3087, CVE-2006-1378, CVE-2008-0141, CVE-2008-2108, CVE-2009-3278]

2) App misuses random values

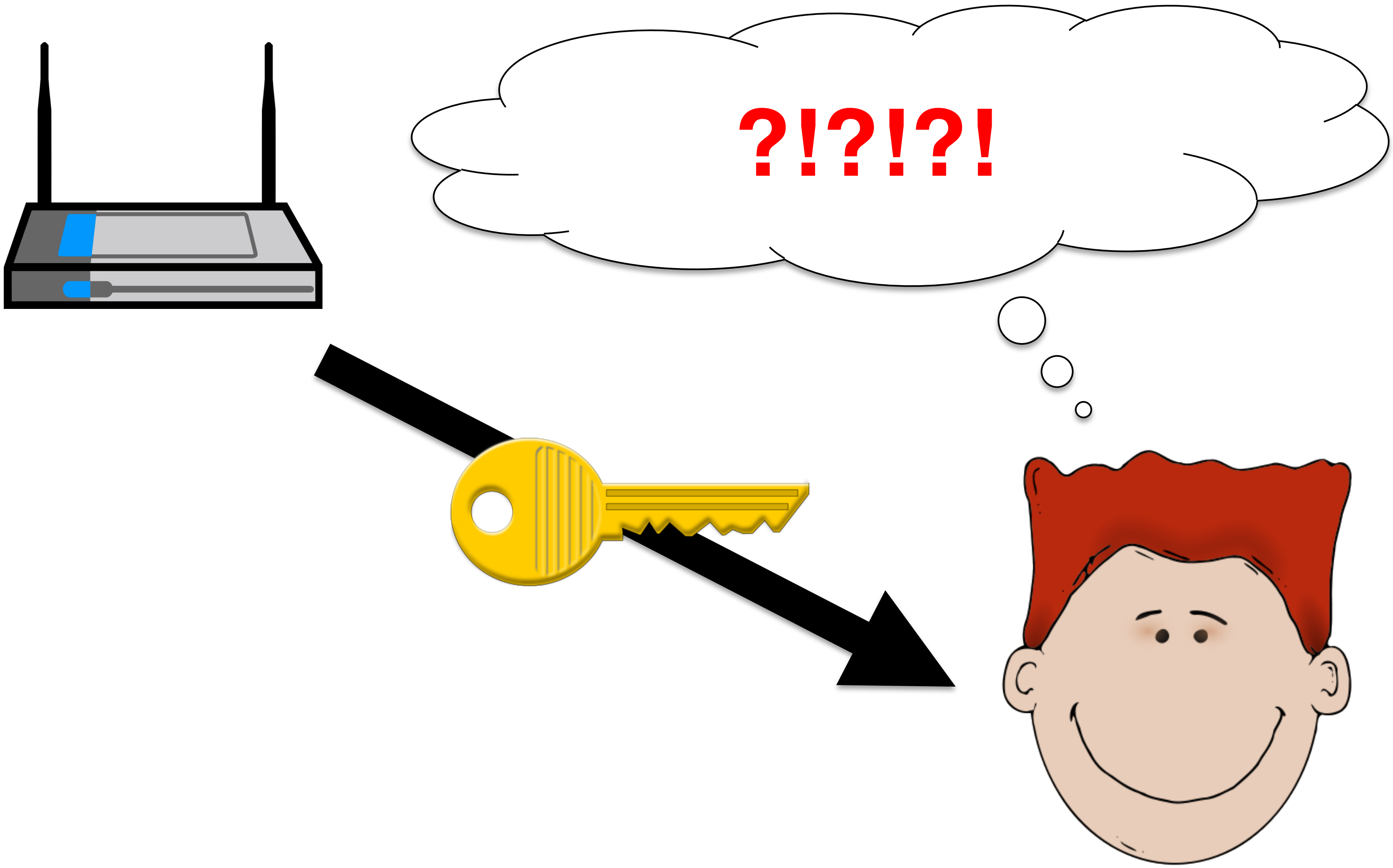
```
bytes = read_block('/dev/random');
// ...
bytes = Hash(time());
```

[CVE-2001-1141, CVE-2003-1376, CVE-2008-0166, CVE-2011-3599]

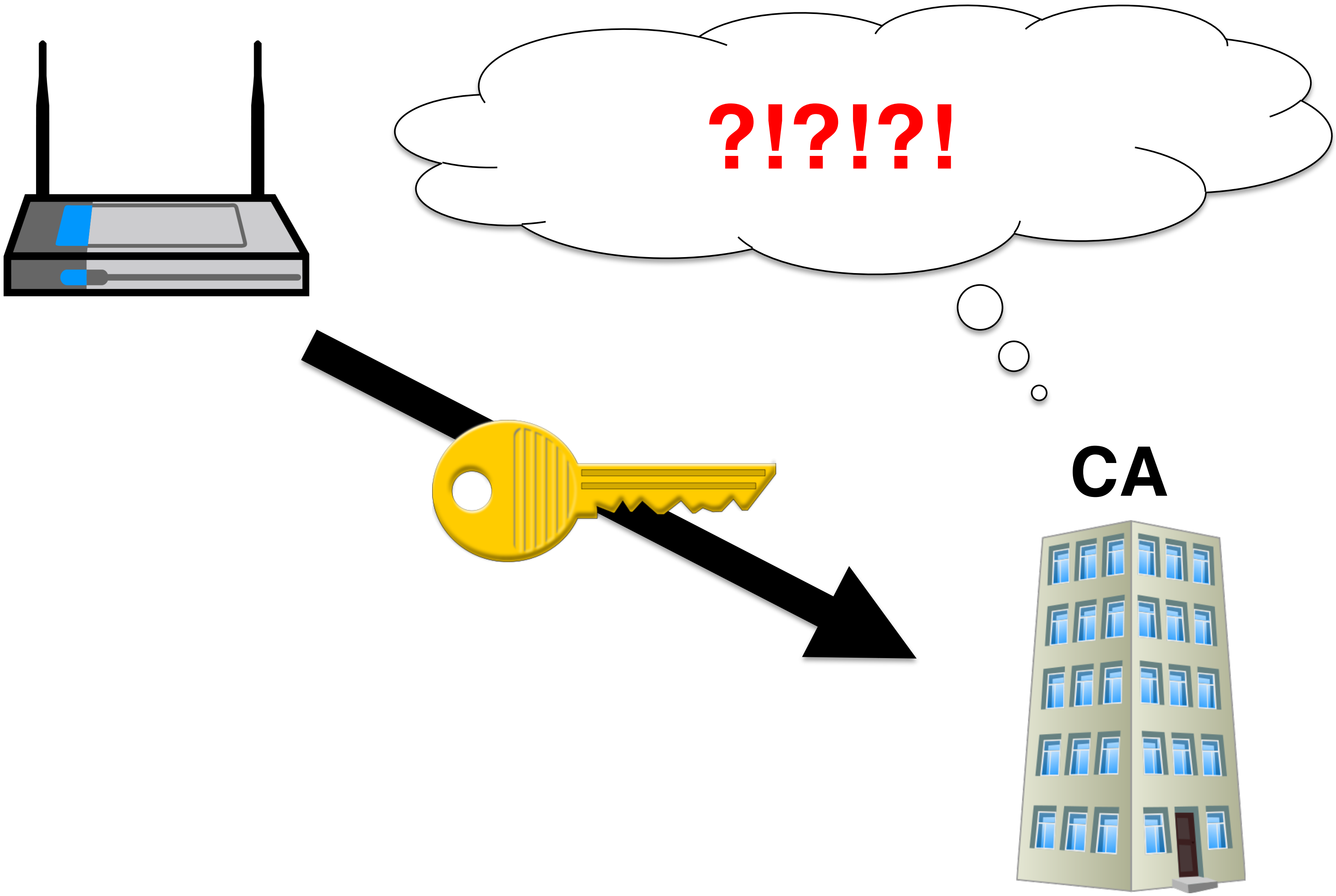
State of the art



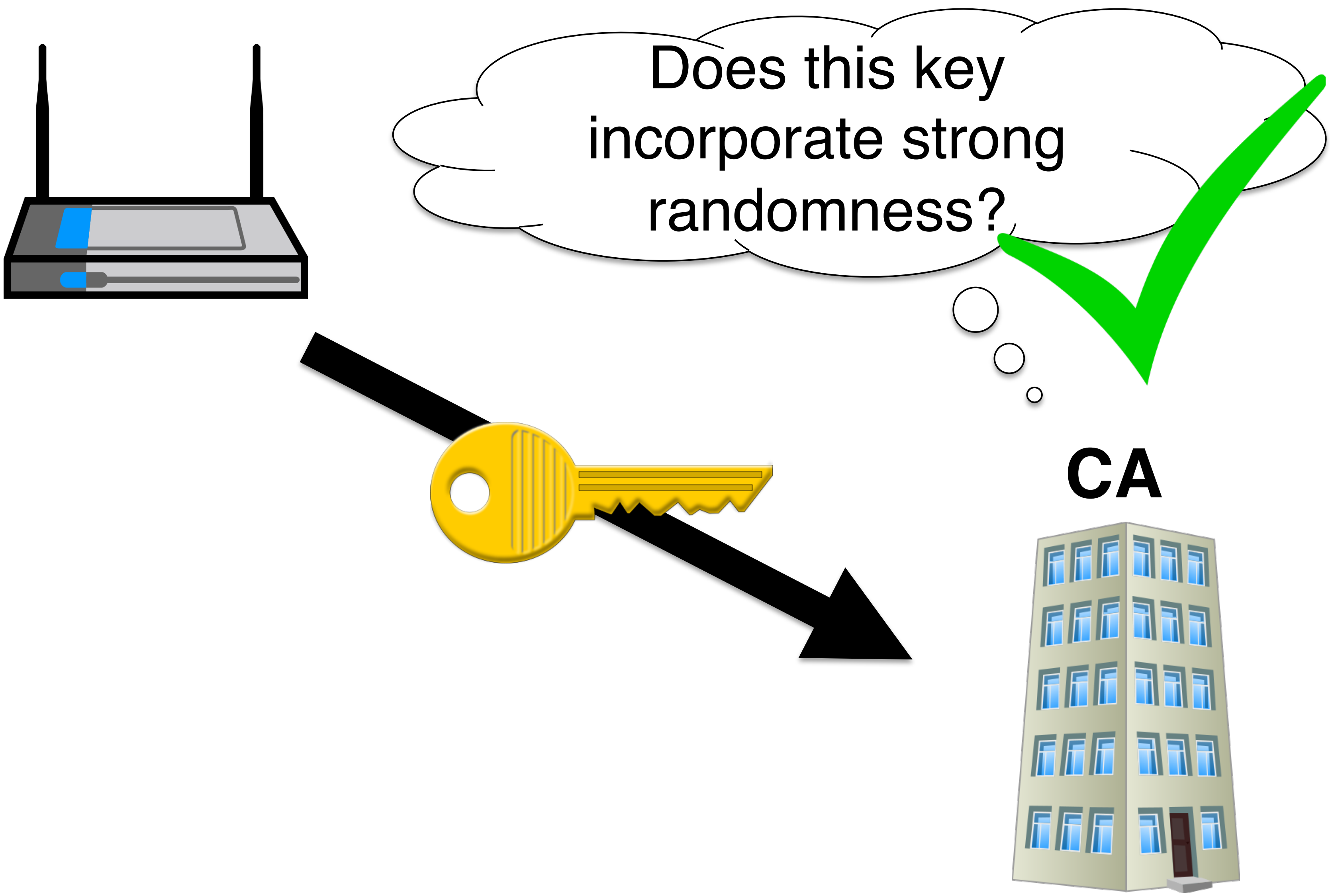
State of the art

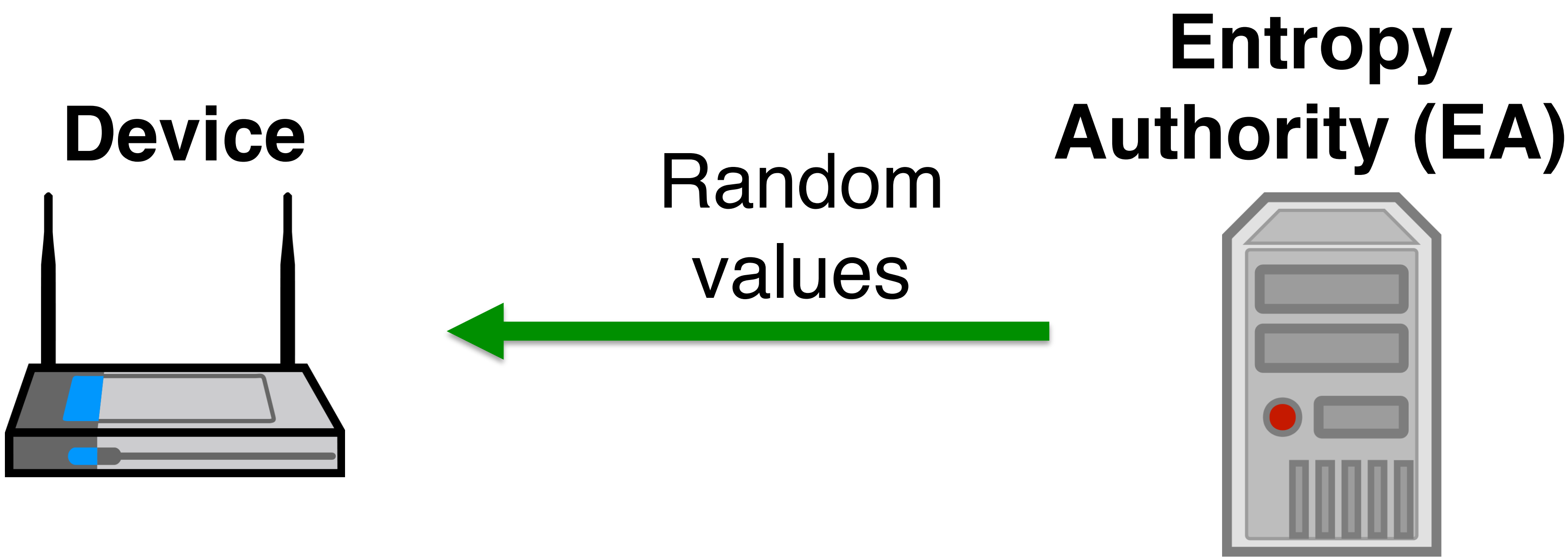


State of the art

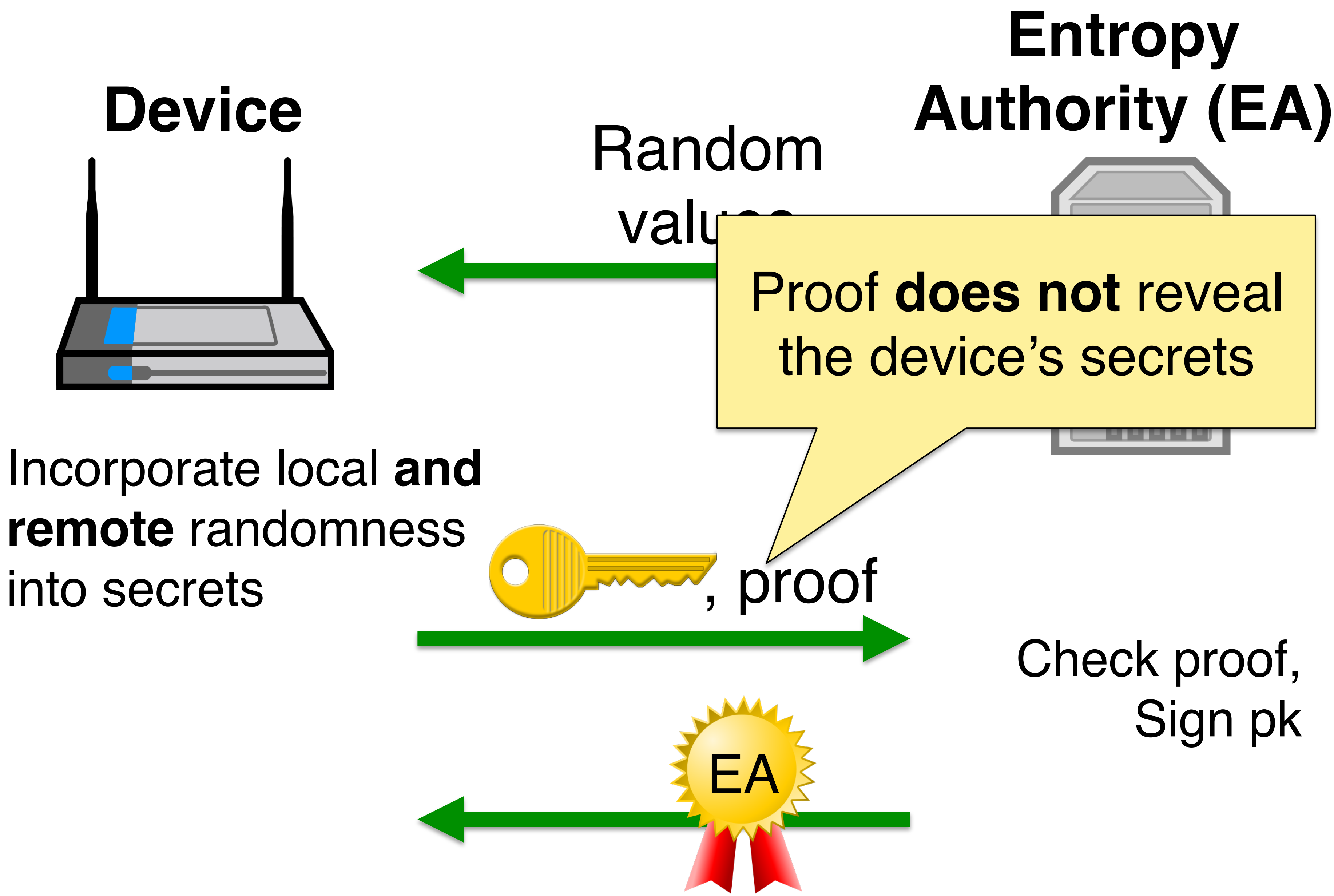


Goal

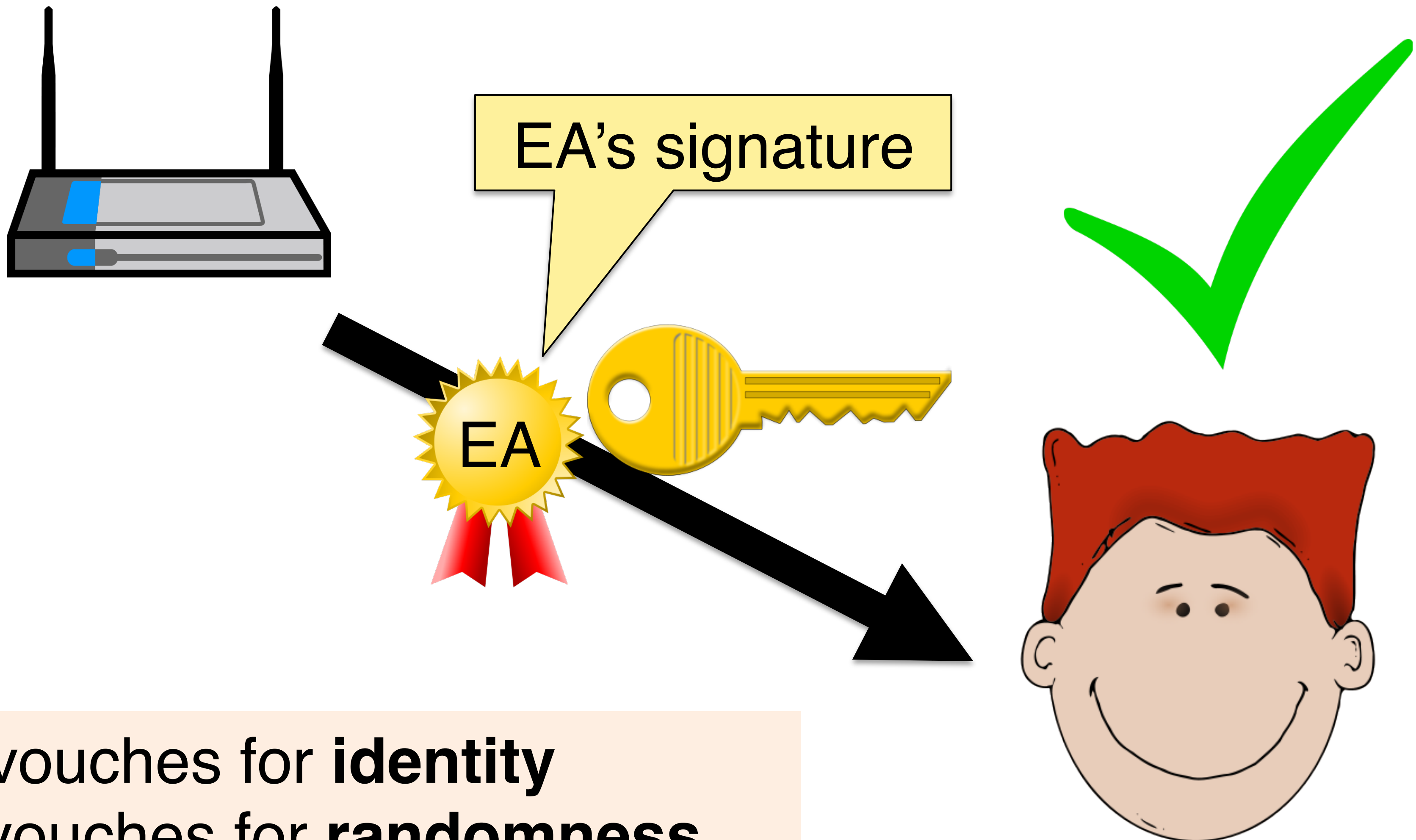





```
bytes = read_from_EA();
// ...
// ...
bytes = Hash(time());
```


Device
Entropy Authority (EA)
Random
valu
Proof **does not** reveal the device's secrets
Incorporate local **and remote** randomness into secrets
, proof
Check proof,
Sign pk
EA

Goal



CA vouches for **identity**
EA vouches for **randomness**

System Goals

1) Output public key is as "random" as possible

```
key_entropy =
  max(device_entropy, ea_entropy);
```

2) If device uses strong randomness source, it is no worse off by running the protocol
   - Does not leak secrets to EA

Outline

- **Motivation**
- Threat Model
- Protocol
- Evaluation

Outline

- Motivation
- **Threat Model**
- Protocol
- Evaluation

Threat model

Adversary: can eavesdrop on everything except for a one-time “set-up phase”

Device: tries to generate correctly formed key drawn from **distribution with low min-entropy**

- Device is correct otherwise

Captures many real-world randomness threats

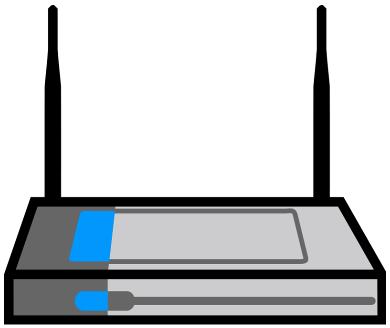
Preliminaries

- Homomorphic commitments [Pedersen, Crypto '91]
  - Commitment to x: $C(x) = g^x h^r$
  - Given C(x) and C(y), can compute C(x+y)

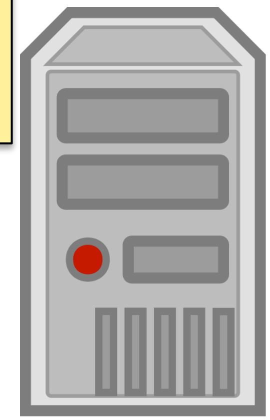
- ZK proof of knowledge for multiplication
  - Given C(x), C(y), z, prove in zero knowledge that $z = xy \bmod Q$
  - `bool Verify(`π, C(x), C(y), `z)`

Outline

- Motivation
- Threat Model
- **Protocol**
- Evaluation



Prevents device from setting $x = -x'$



$C(x), C(y)$ →

← x',

$C(p) = C(x)g^{x'+\delta_x}$

Prevents device from setting $\delta_x = -x'$

Find δs to make

$p = x+x'+\delta_x$

$q = y+y'+\delta_y$

RSA primes.

π = `Prove`(n=pq).

n, δ_x, δ_y →

Check δs are small

`Verify`(π, C(p), C(q), n),

sign public key

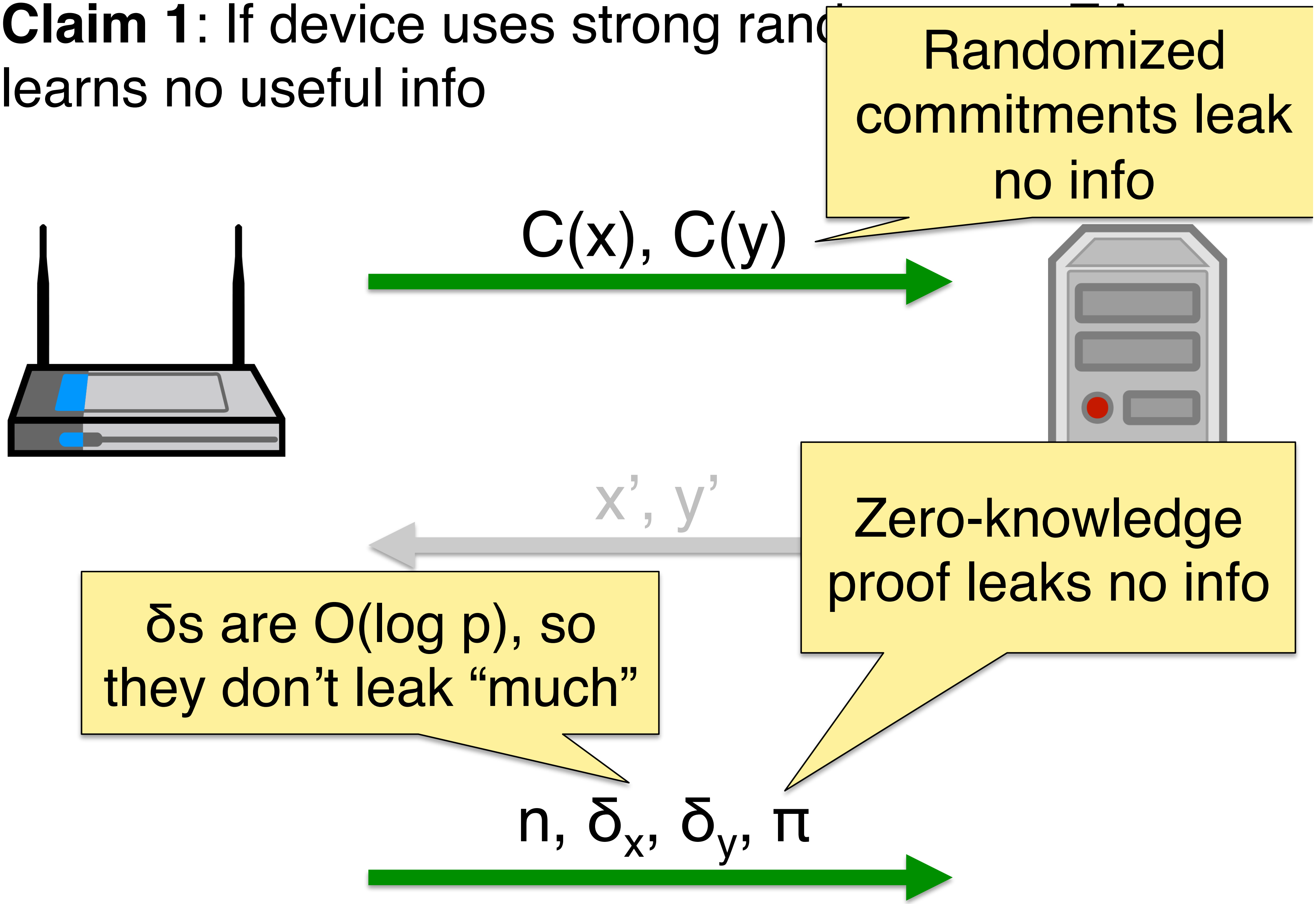
← $\sigma(n)$

Security Properties

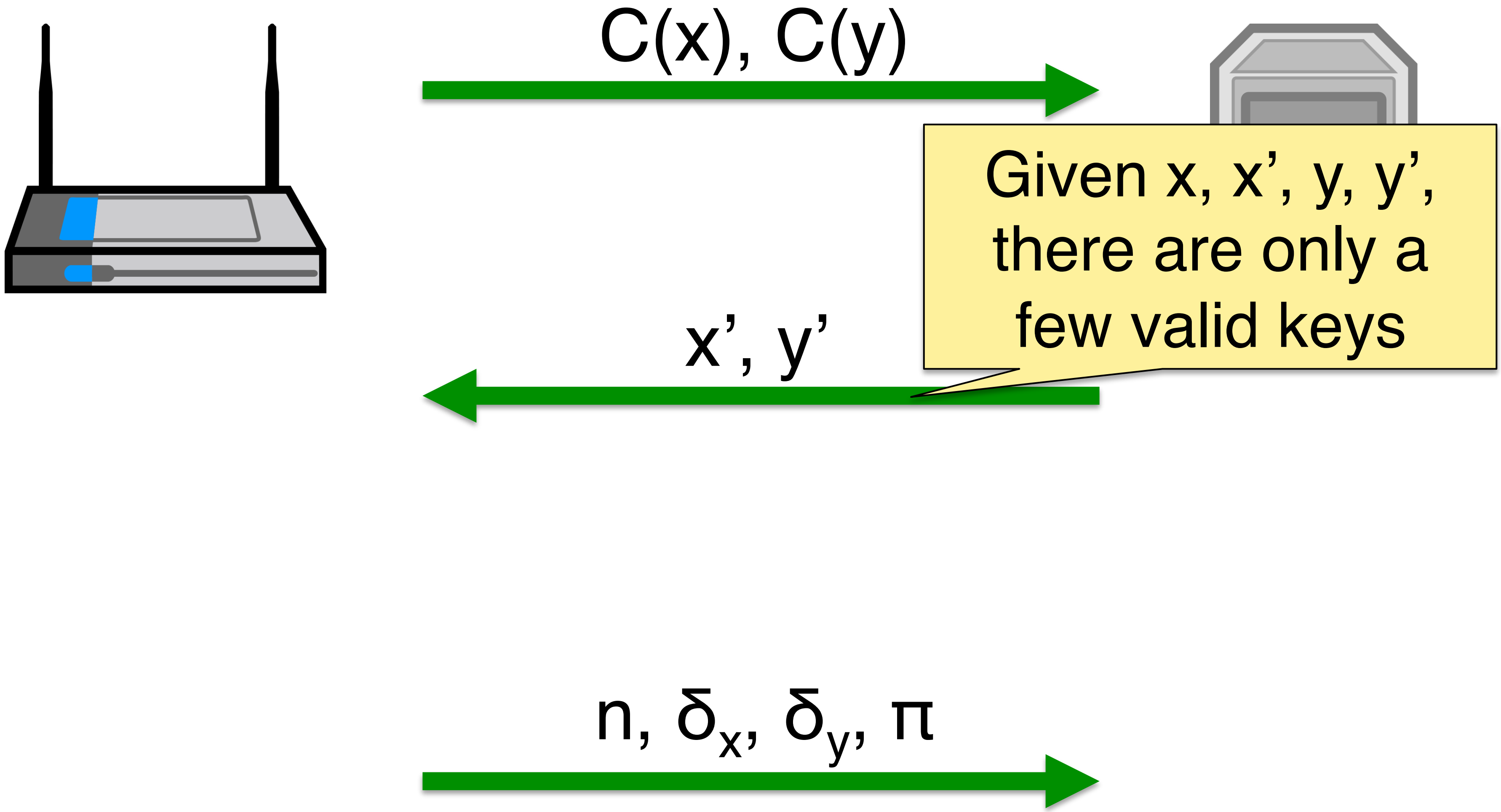
- If the device uses strong randomness
  → EA learns no useful info about the secrets
- If the device uses strong randomness OR the EA is correct:
  → Device ends up with a strong key

Even if the EA is untrustworthy, device is better off running the protocol

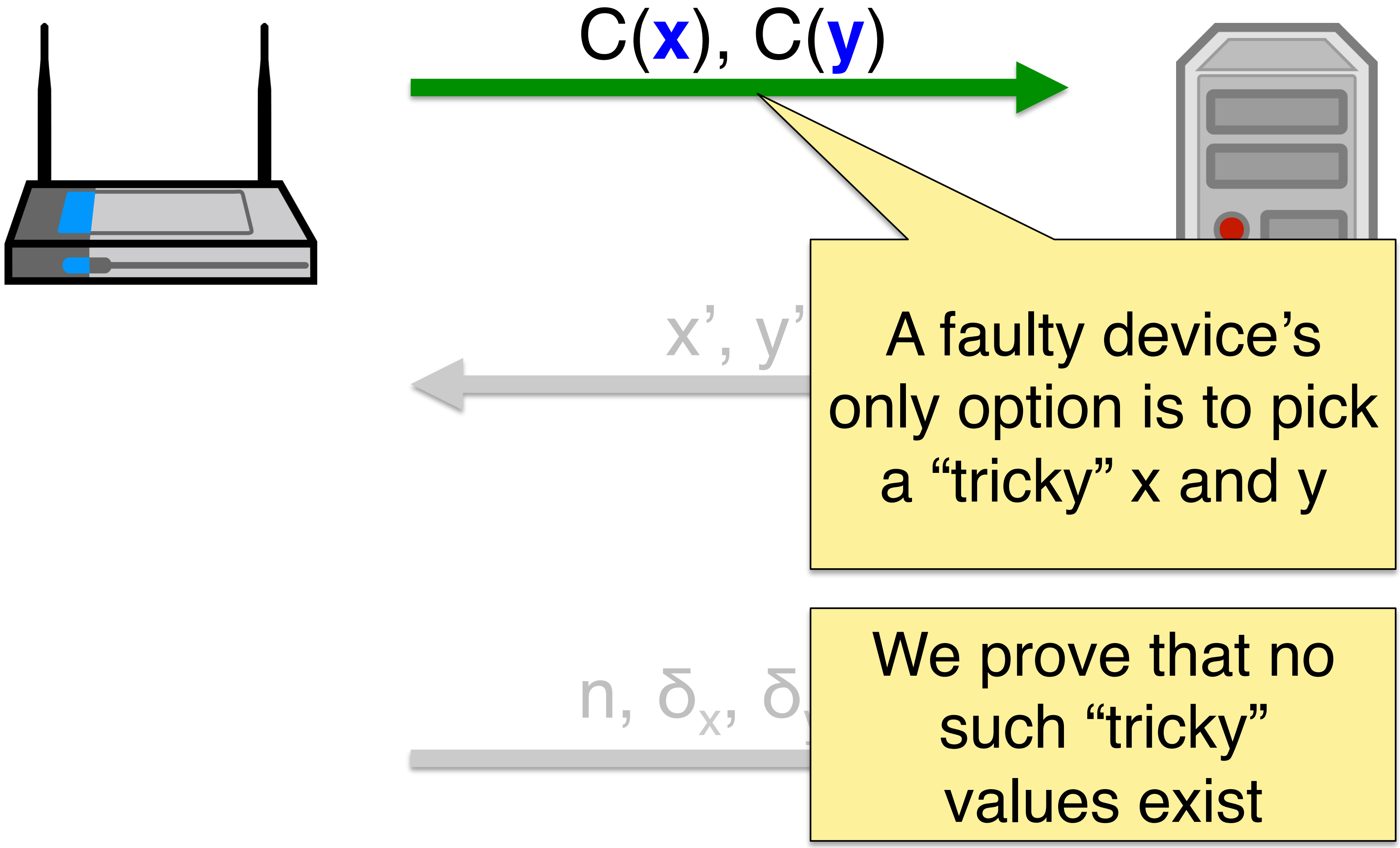
Claim 1: If device uses strong rand
learns no useful info



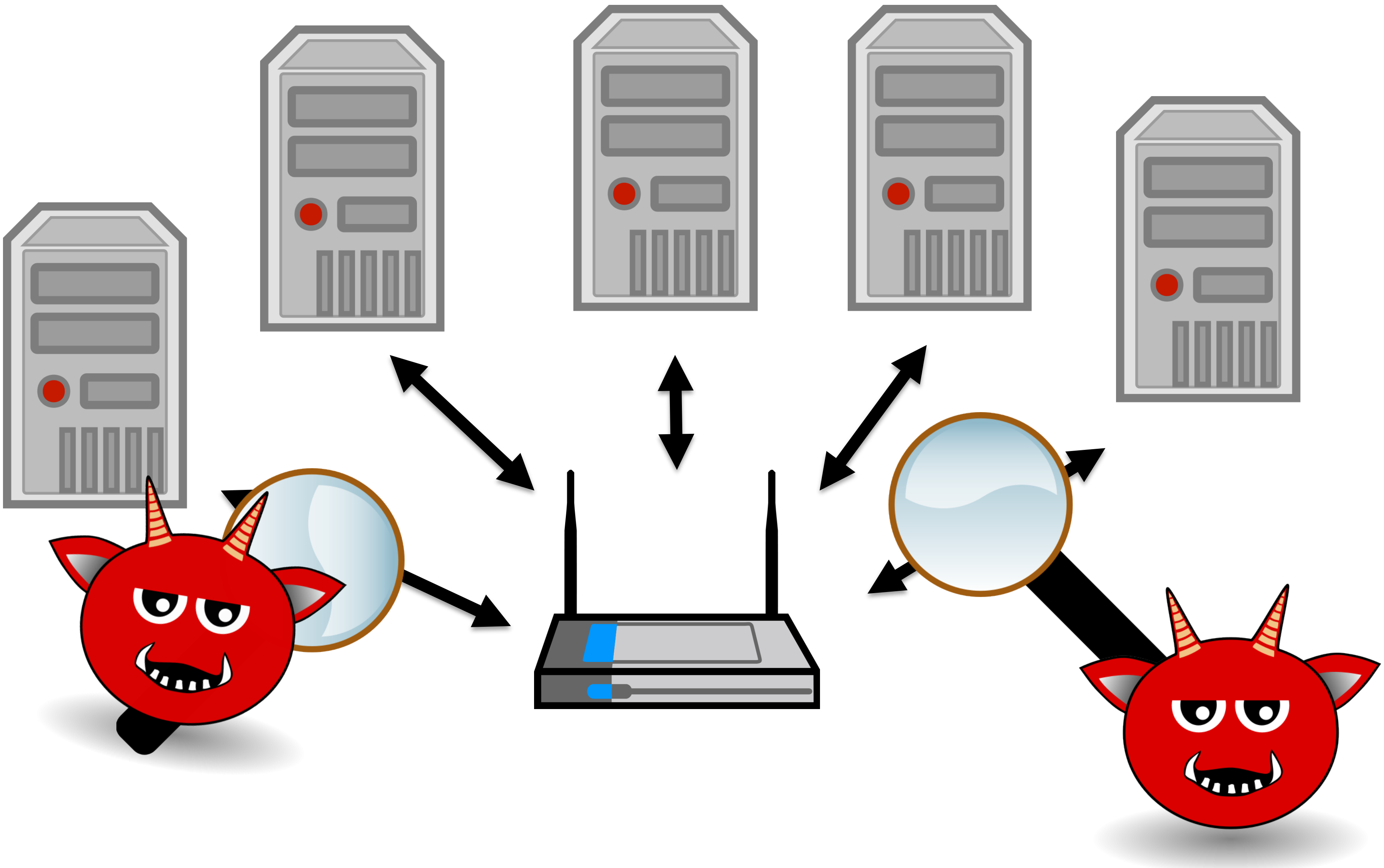
Claim 2: Correct EA will *never* sign a key sampled from a distribution with low min-entropy



Claim 2: Correct EA will *never* sign a key sampled from a distribution with low min-entropy



Multiple Entropy Authorities



Outline

- Motivation
- Threat Model
- Protocol
- **Evaluation**

Key Generation Time

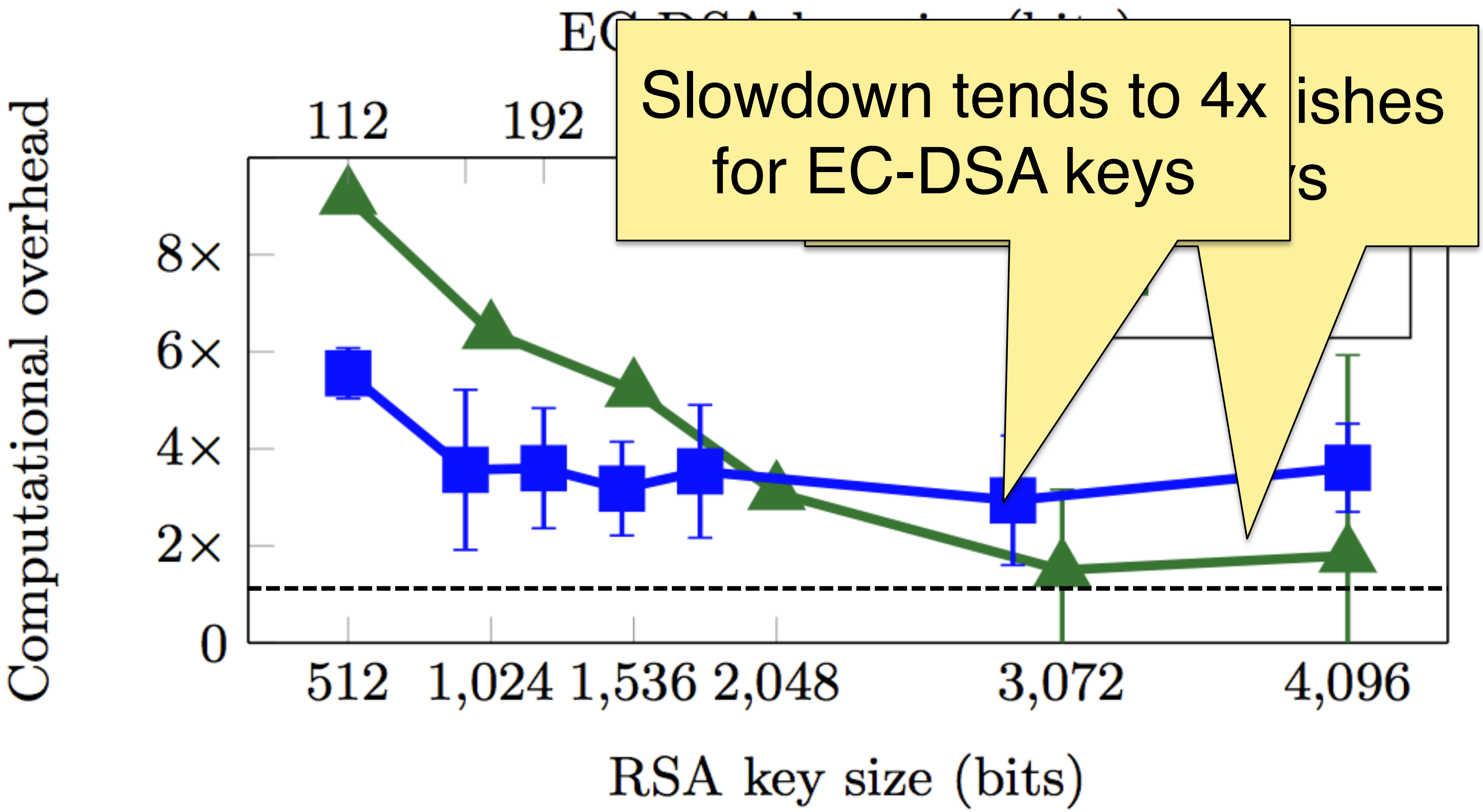
- Wall-clock time (in seconds) to generate a key on a Linksys E2500-NP home router.
- EA is modern Linux server 5

Less than 2x
SA

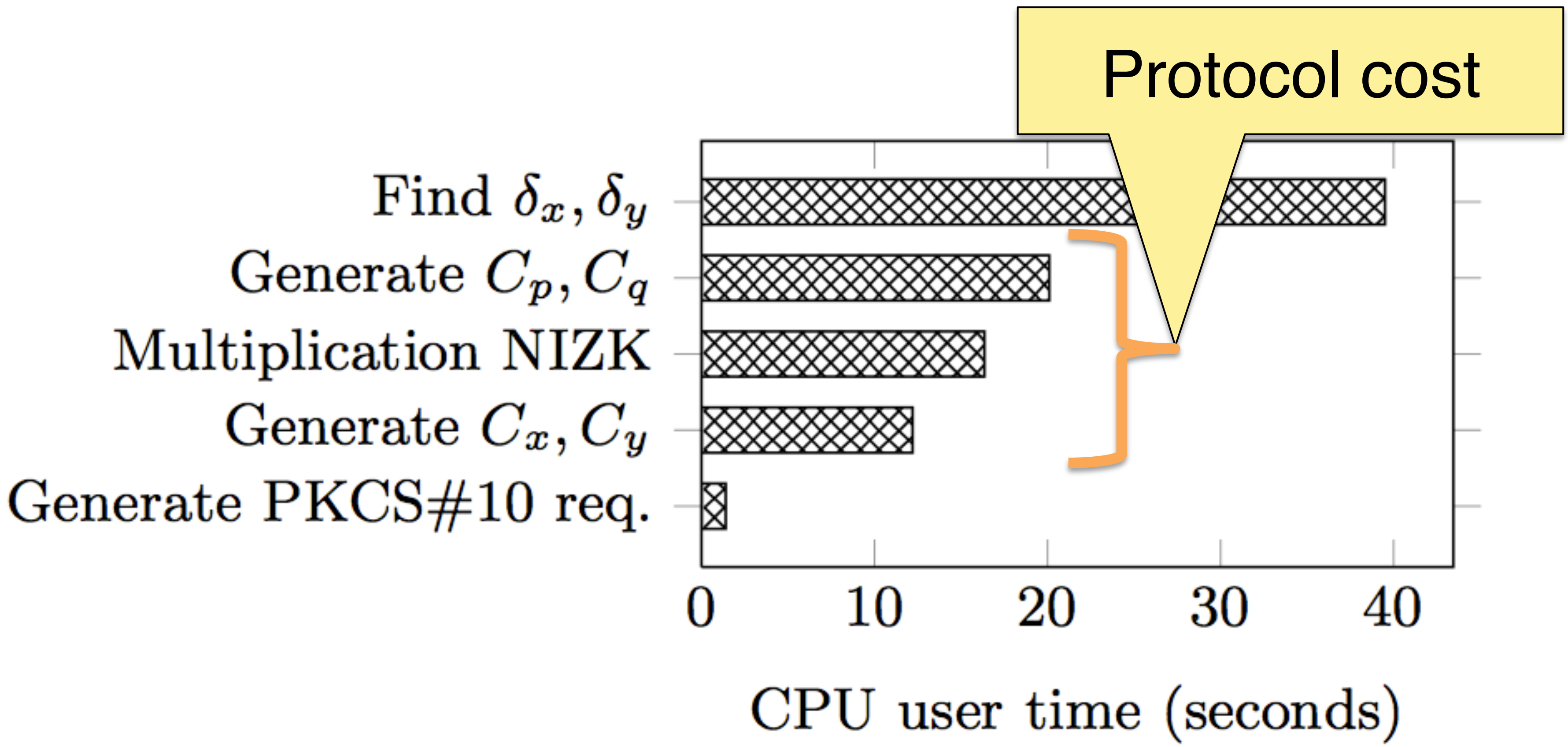
Less than 2 seconds for EC-DSA

	No proto	Proto	rnet	Slowdown
RSA 2048	59.16	96.93	[illegible].57	1.7x
EC-DSA 224	0.45	0.84	1.61	3.6x

Computational Overhead



Computational Bottlenecks

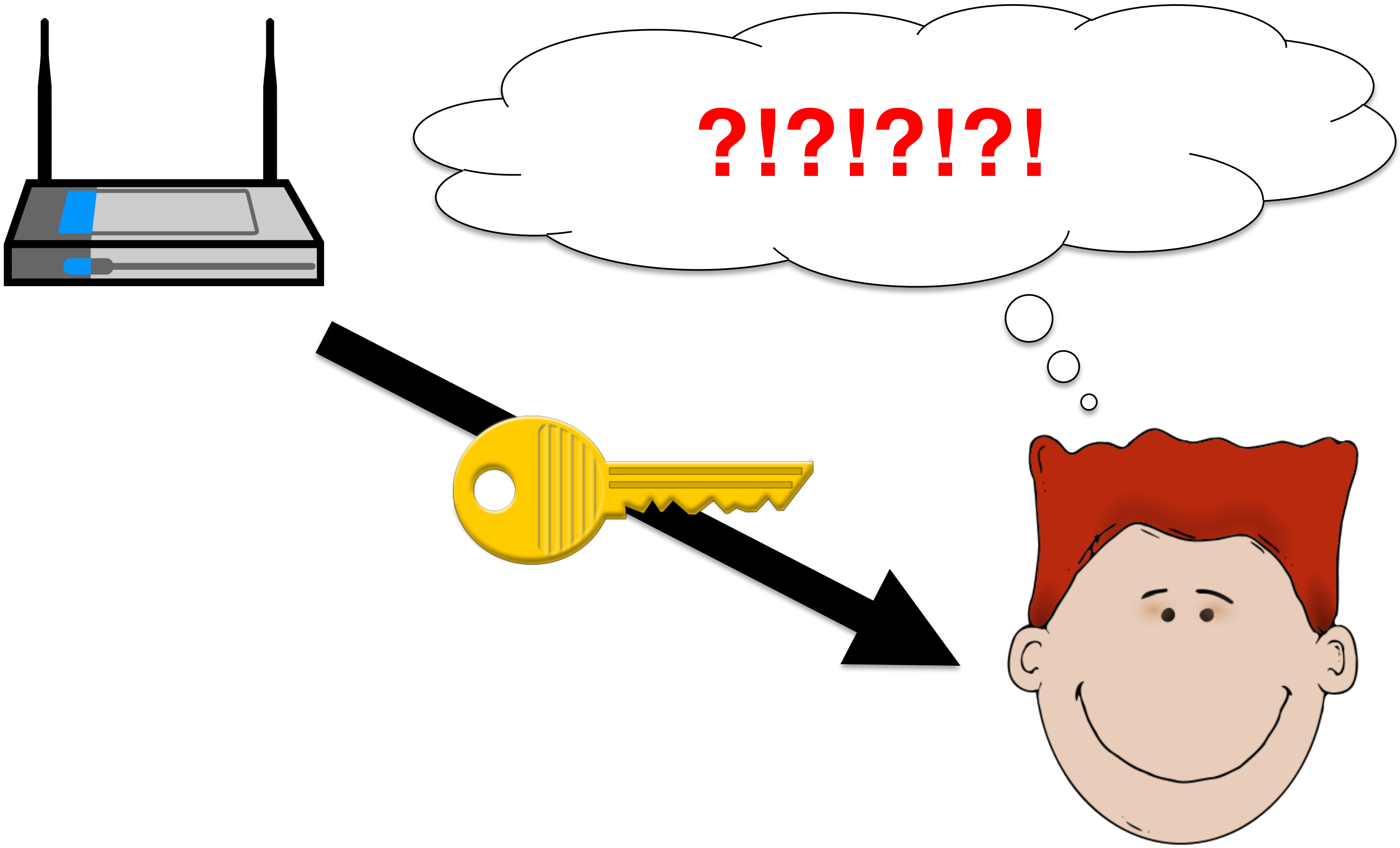


RSA-2048 key on Linksys E2500-NP home router

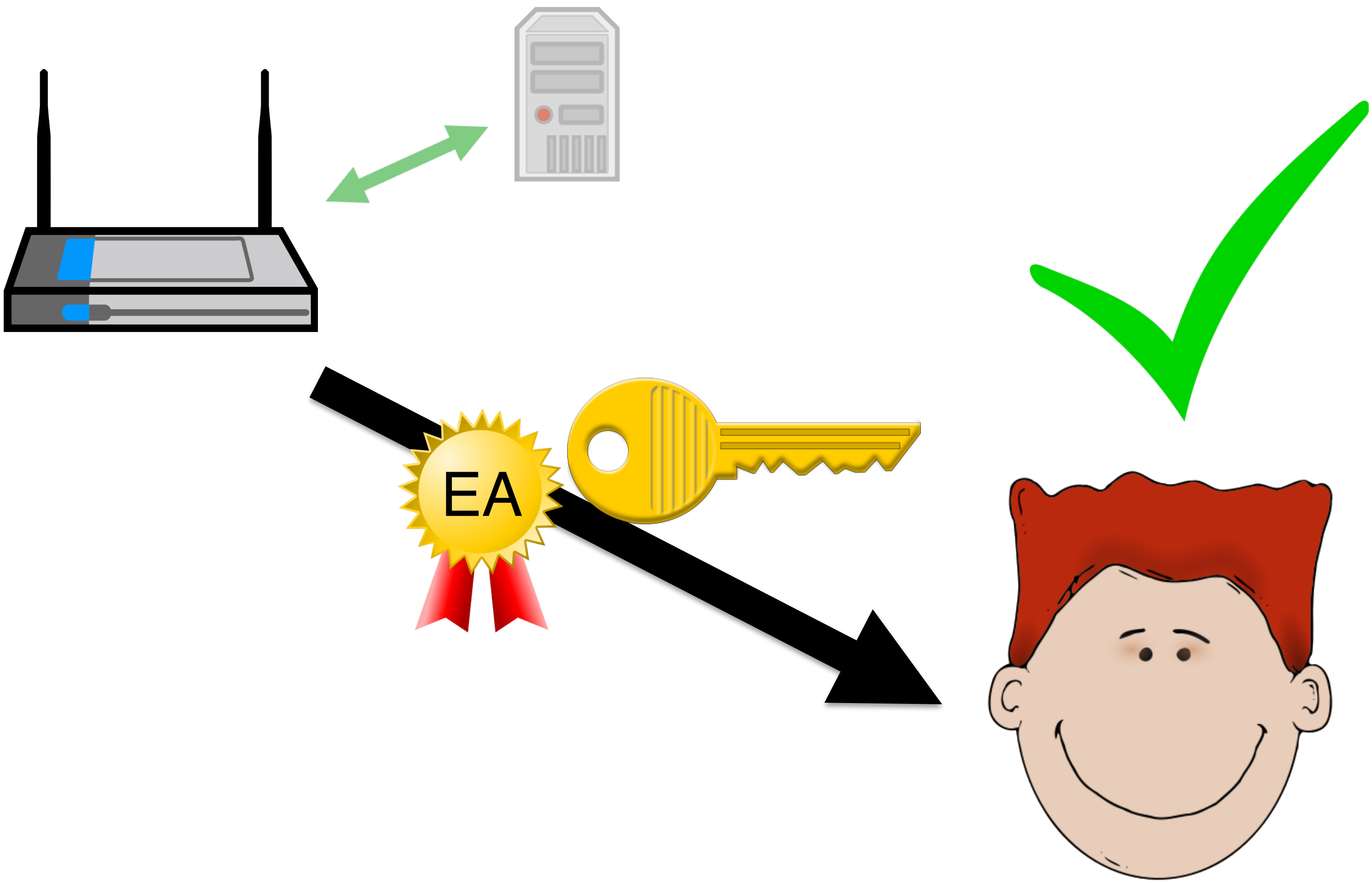
Related Work

- Hedged PKC [Bellare et al., ASIACRYPT '09]
- Better random values
  - Hardware RNG
  - Entropics [Mowery et al. Oakland '13]
- Juels-Guajardo Protocol [PKC '02]
  - Defends against *kleptography*
  - Requires heavier primitives
    (24x more big exponentiations)

Today



With our protocol



Conclusion

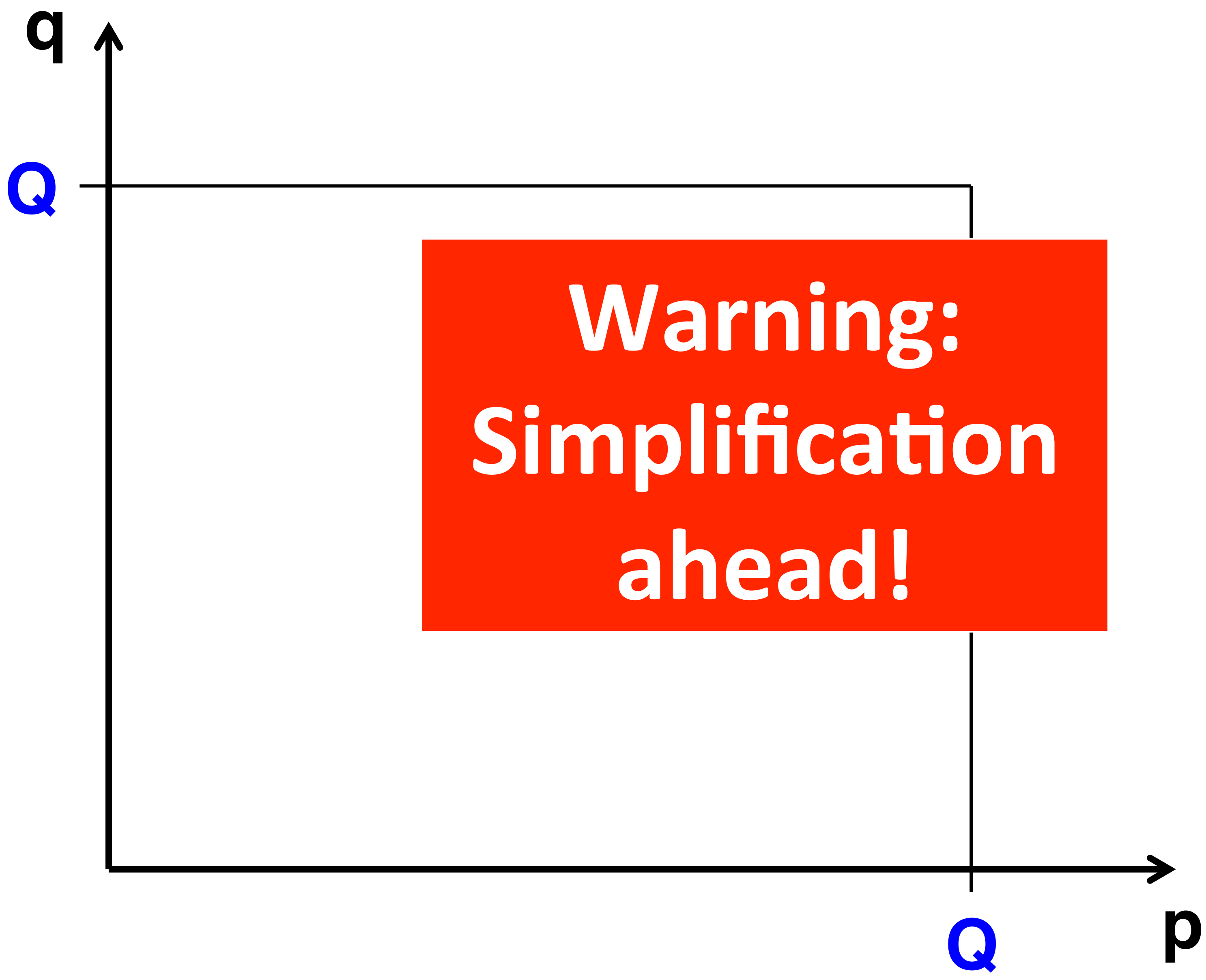
- Using “entropy authorities” is a practical way to prevent weak cryptographic keys
- Other parts of the stack need help too
  - Signing nonces, ASLR, DNS source ports, …
- Interested in running an entropy authority? Let’s talk!

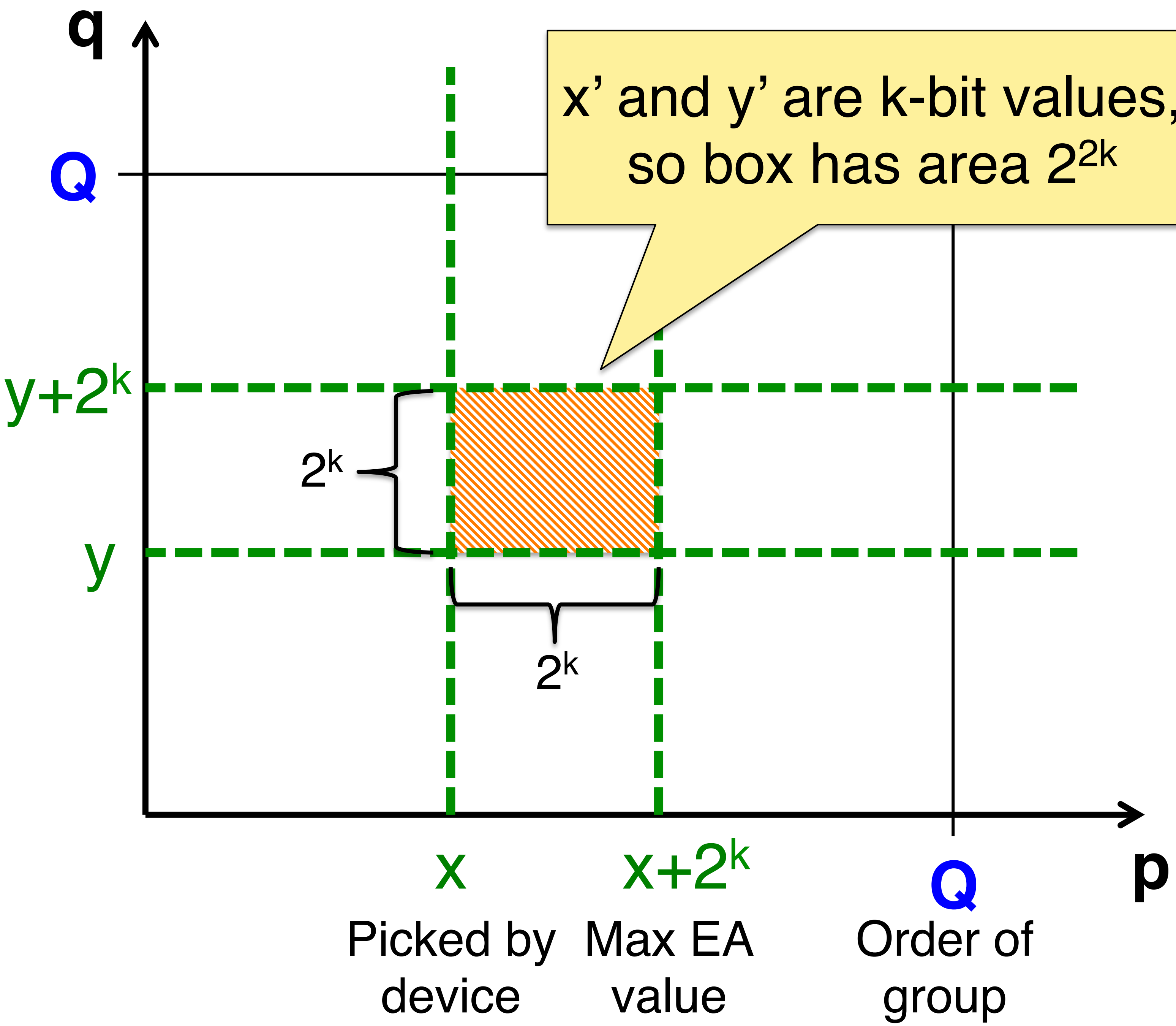
Questions?

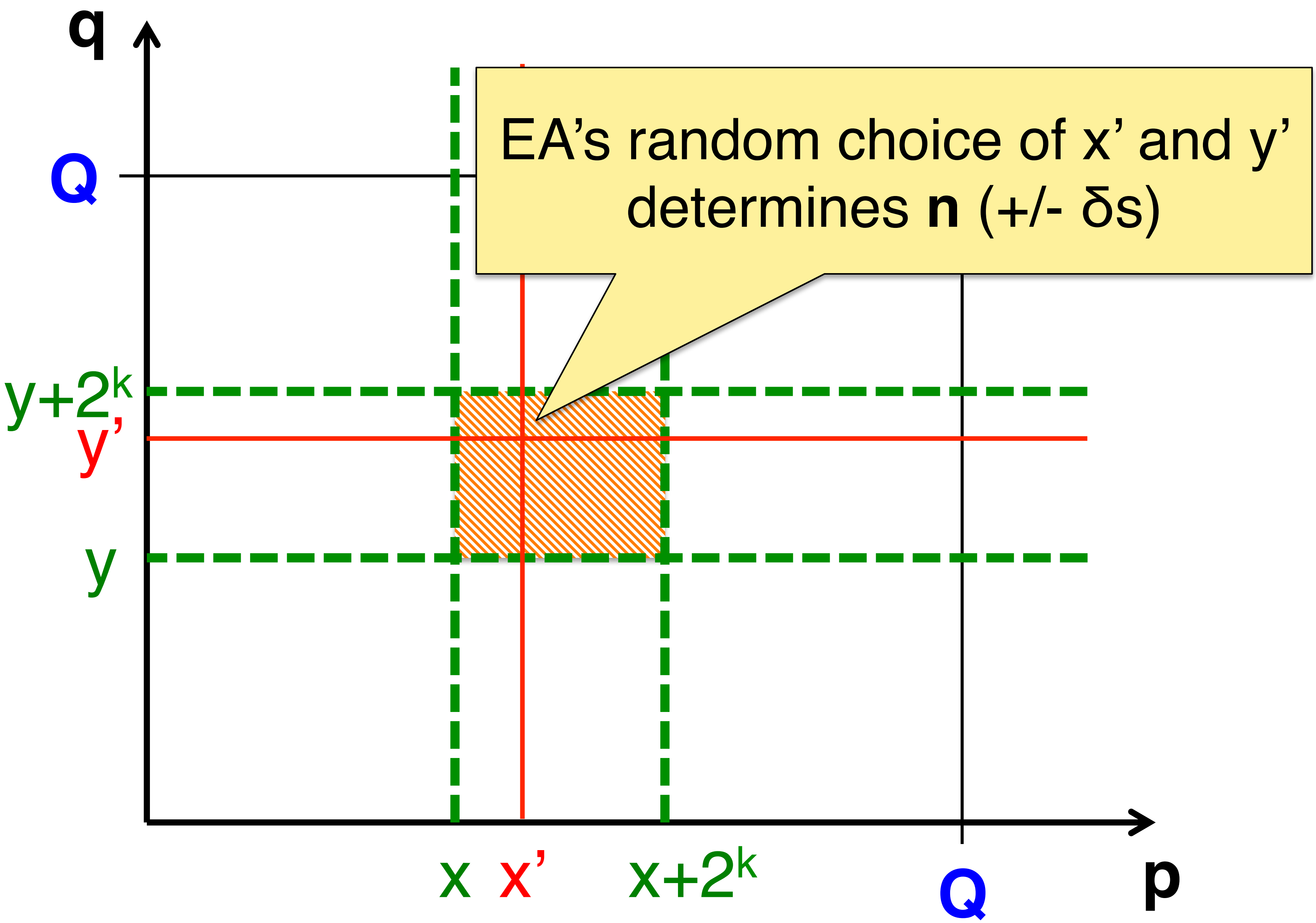
Henry Corrigan-Gibbs
henrycg@stanford.edu
http://github.com/henrycg/earand

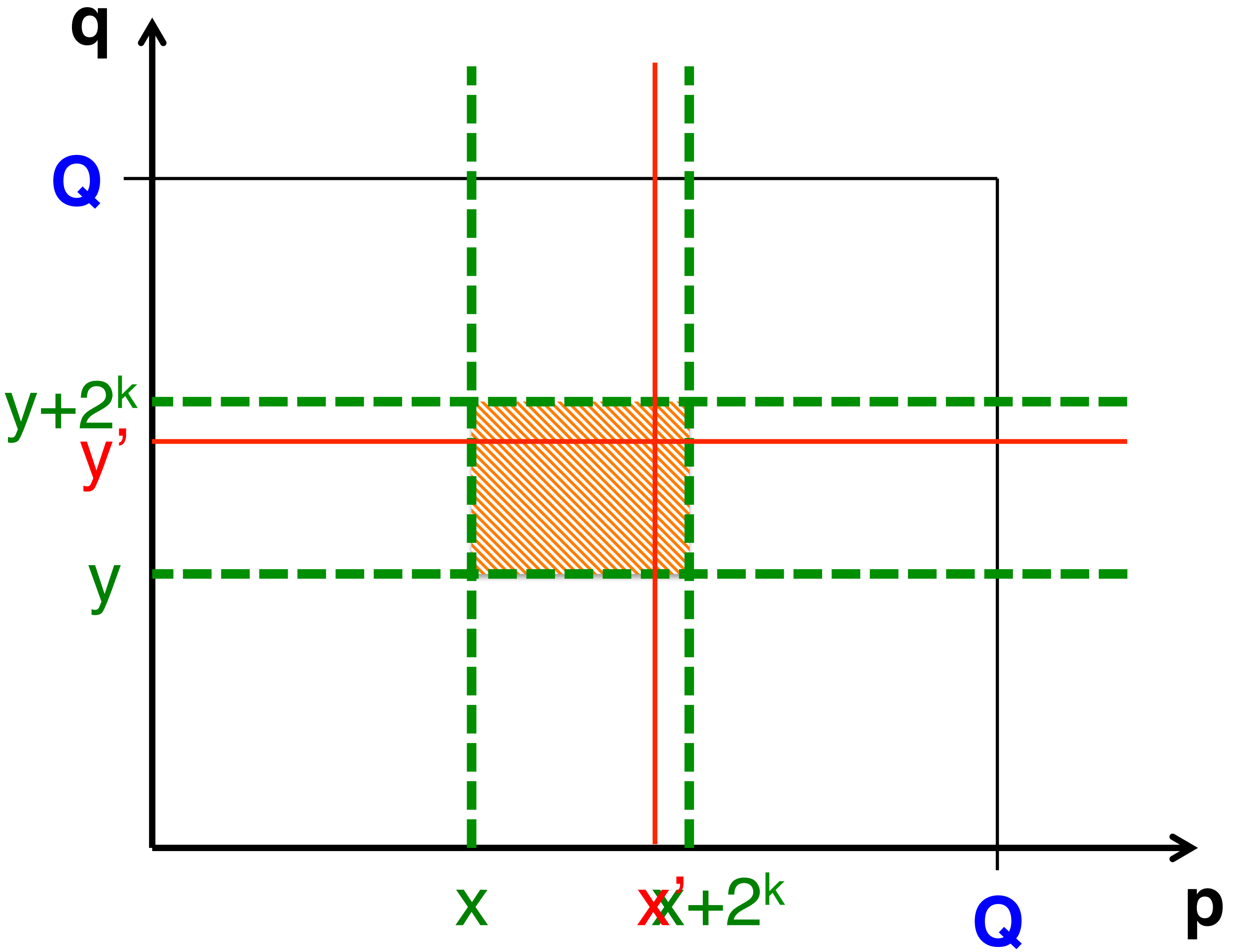
Thanks to David Wolinsky,
Ewa Syta, Phil Levis, Suman Jana,
and Zooko Wilcox-O'Hearn.

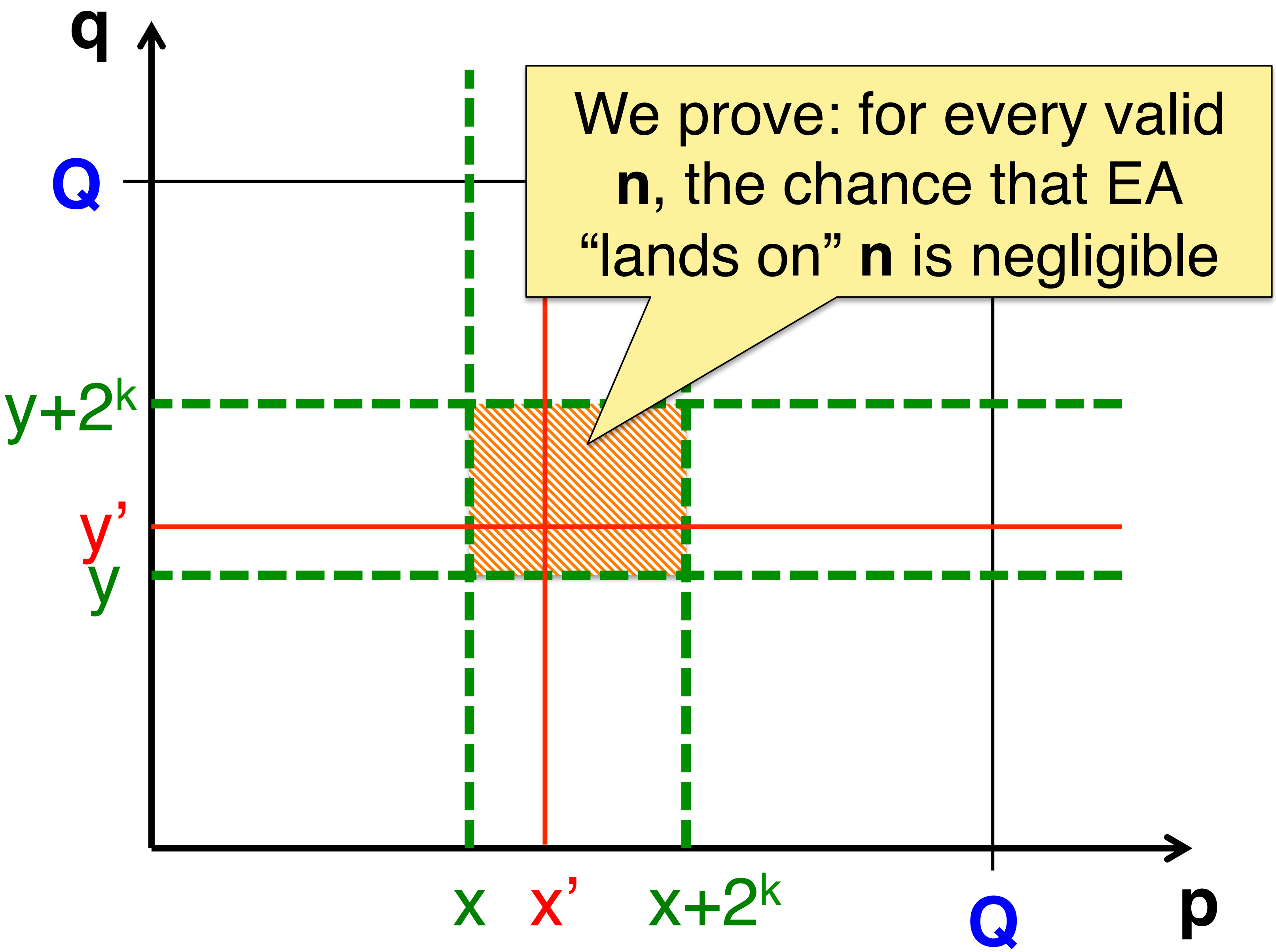
[redacted]

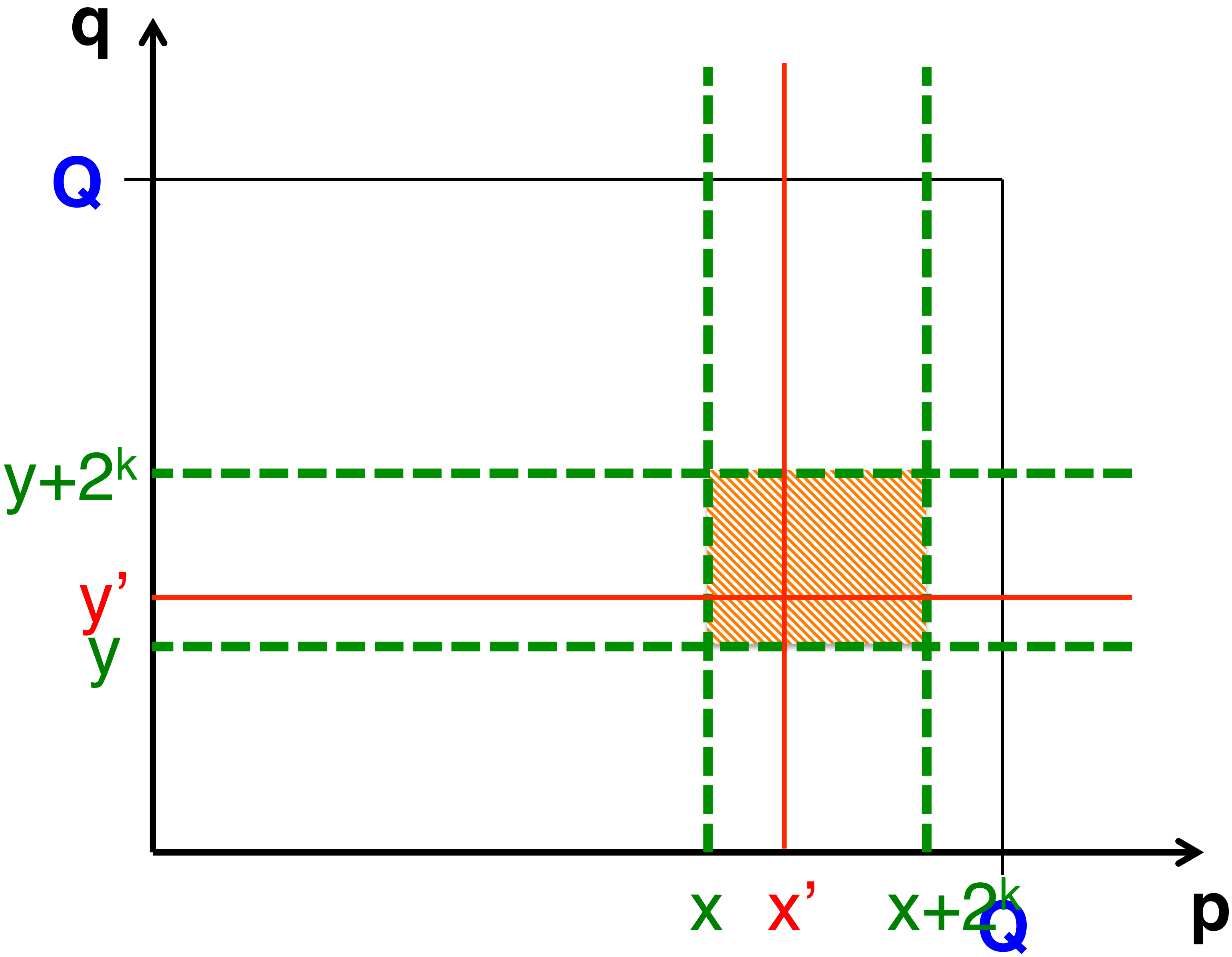

q
Q
Warning:
Simplification
ahead!
Q
p

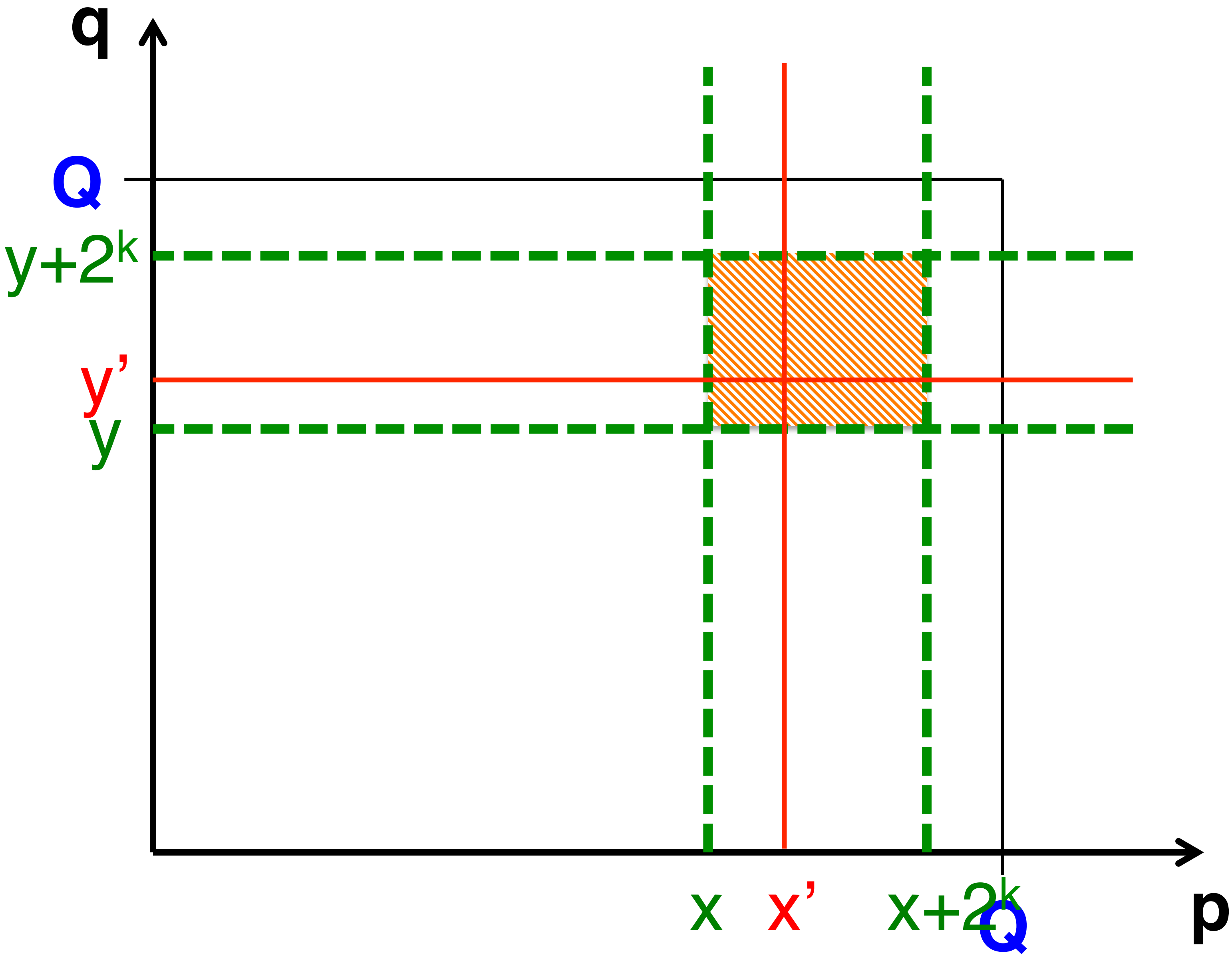

q
Q
x' and y' are k-bit values,
so box has area 2^{2k}
$y+2^k$
2^k
y
2^k
x
$x+2^k$
Q
p
Picked by device
Max EA value
Order of group

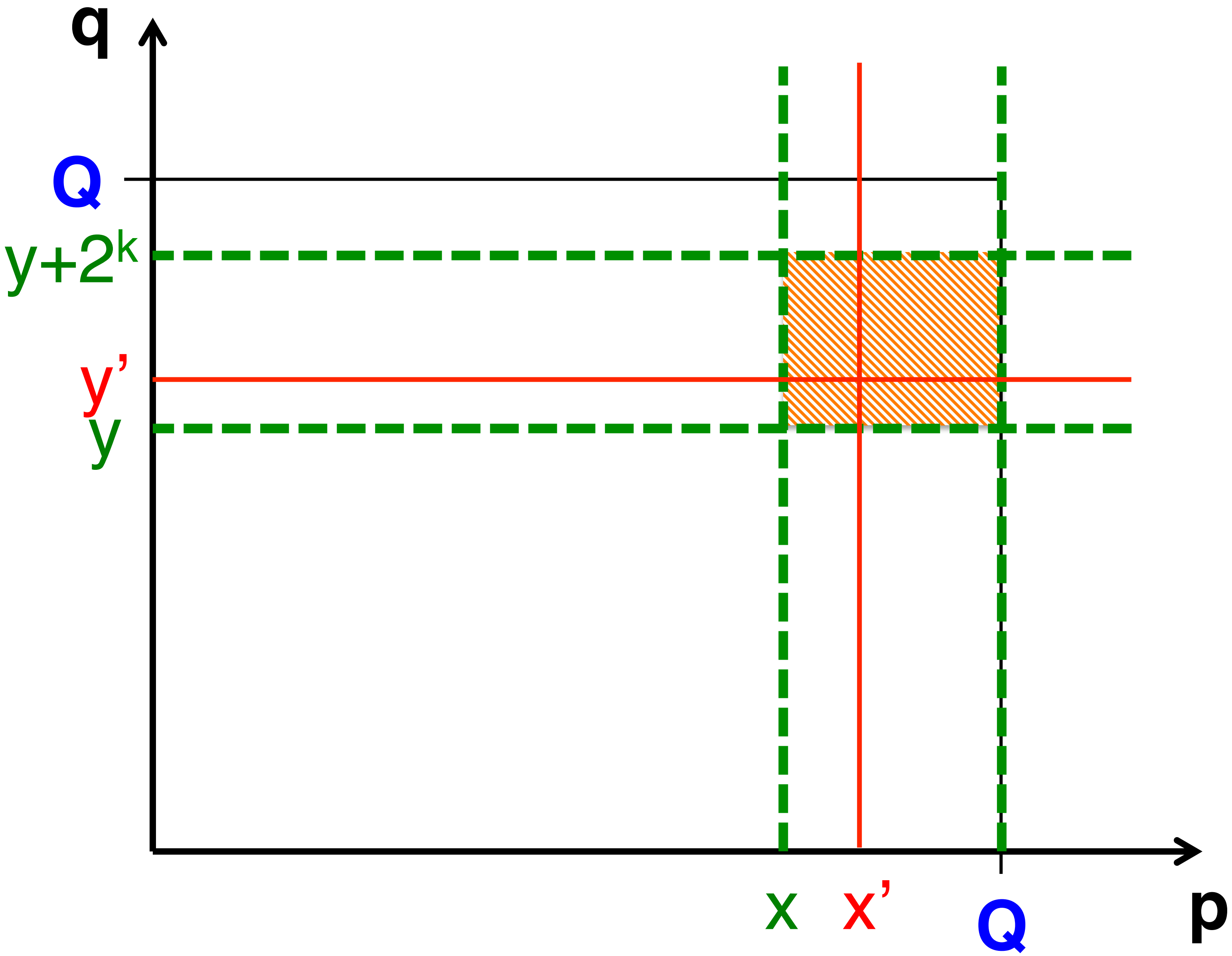

EA's random choice of x' and y' determines **n** (+/- δs)
q
Q
y+2^k
y'
y
x
x'
x+2^k
Q
p

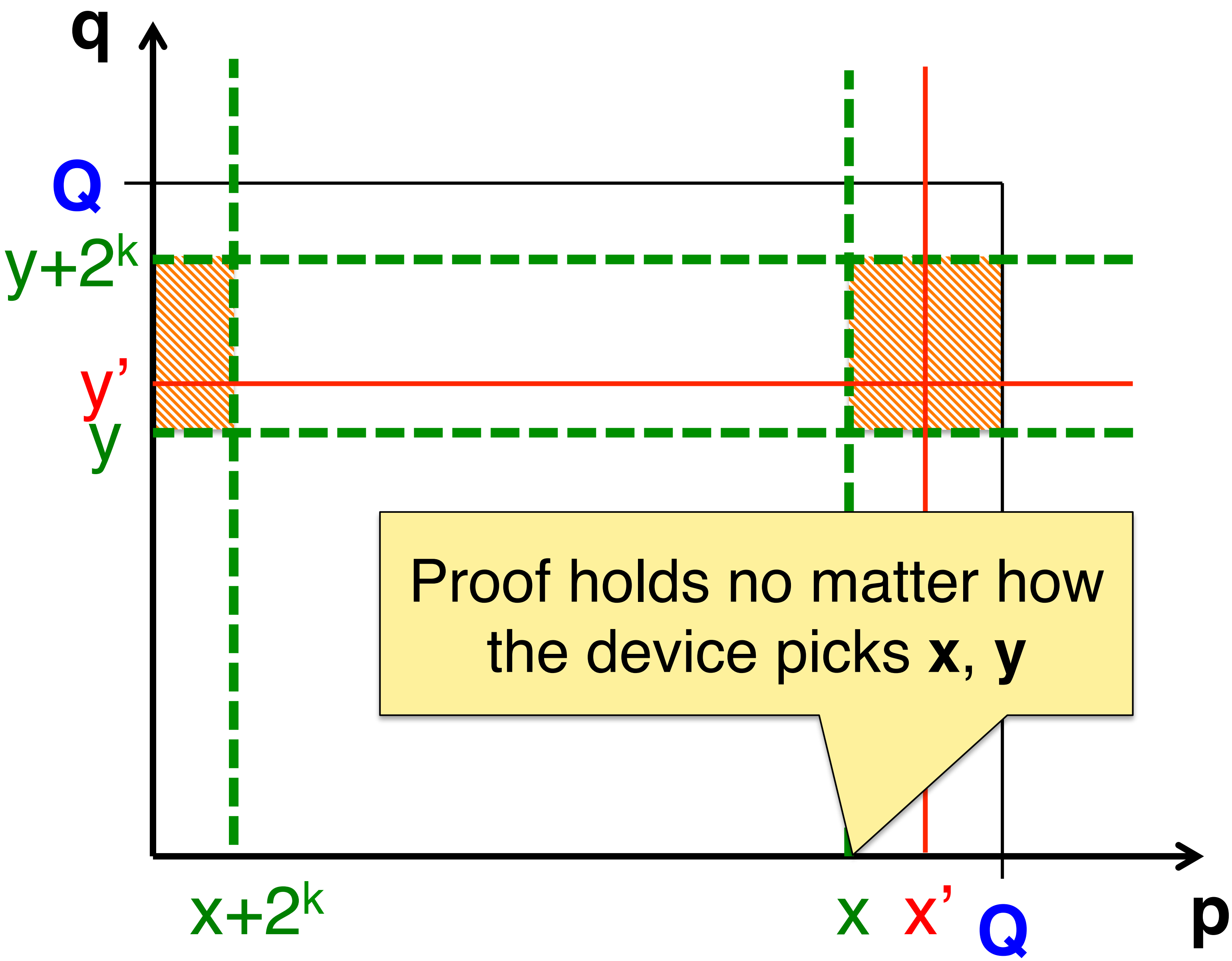

q
Q
y+2k
y'
y
x
x'
x+2k
Q
p


q
Q
We prove: for every valid **n**, the chance that EA "lands on" **n** is negligible
y+2^k
y'
y
x
x'
x+2^k
Q
p

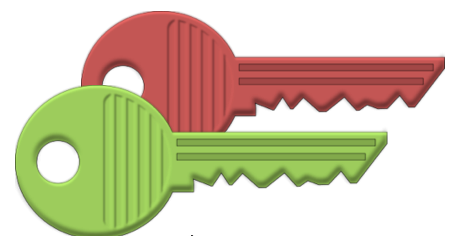

q
Q
y+2^k
y'
y
x
x'
x+2^k
Q
p


q
Q
y+2^k
y'
y
x
x'
x+2^k
Q
p


q
Q
y+2^k
y'
y
x
x'
Q
p


q
Q
y+2^k
y'
y
Proof holds no matter how the device picks **x**, **y**
x+2^k
x
x'
Q
p

Our Goal

	Device	
Entropy Authority	Strong randomness	Weak randomness
Strong randomness		
Weak randomness / Malicious		

Reveals device's secrets to EA only