

650.445: Practical Cryptographic Systems

Side Channel Attacks

Instructor: Matthew Green

Housekeeping:

- A2 tonight
- Grades for A1 back

Current Events

The image is a thumbnail for a news article. It features a stylized illustration of a green hand reaching towards a grey shield-like background. The word "TLS" is printed in large, bold, black letters on the shield. To the right of the shield is the Mozilla Firefox logo, which is a stylized orange fox head and tail. Below the shield, a yellow sun-like icon with a blue ribbon is visible. At the bottom left, there's a vertical column of social media sharing icons (Facebook, Google+, LinkedIn, etc.) with a red '0' indicating no shares. A black banner across the bottom contains the text "MOZILLA PATCHING FIREFOX CERTIFICATE PINNING VULNERABILITY". At the very bottom, there's a thin horizontal bar with the author's name "by Michael Mimoso" and a Twitter link "Follow @mike_mimosa". The date "September 19, 2016, 4:03 pm" is also present.

by Michael Mimoso [Follow @mike_mimosa](#)

September 19, 2016, 4:03 pm

Current Events

[CVE-2016-6309 \(OpenSSL advisory\)](#) [Critical severity] 26th September 2016: ↑

This issue only affects OpenSSL 1.1.0a, released on 22nd September 2016. The patch applied to address CVE-2016-6307 resulted in an issue where if a message larger than approx 16k is received then the underlying buffer to store the incoming message is reallocated and moved. Unfortunately a dangling pointer to the old location is left which results in an attempt to write to the previously freed location. This is likely to result in a crash, however it could potentially lead to execution of arbitrary code. Reported by Robert Święcki (Google Security Team).

- Fixed in OpenSSL 1.1.0b (Affected 1.1.0a)

[CVE-2016-7052 \(OpenSSL advisory\)](#) [Moderate severity] 26th September 2016: ↑

This issue only affects OpenSSL 1.0.2i, released on 22nd September 2016. A bug fix which included a CRL sanity check was added to OpenSSL 1.1.0 but was omitted from OpenSSL 1.0.2i. As a result any attempt to use CRLs in OpenSSL 1.0.2i will crash with a null pointer exception. Reported by Bruce Stephens and Thomas Jakobi.

- Fixed in OpenSSL 1.0.2j (Affected 1.0.2i)

Current Events

Welcome > Blog Home > Cryptography > Mozilla Reduces Threat of Export-Grade Crypto to Firefox



f
g
in
o
t
0

MOZILLA REDUCES THREAT OF EXPORT-GRADE CRYPTO TO FIREFOX

by Michael Mimoso

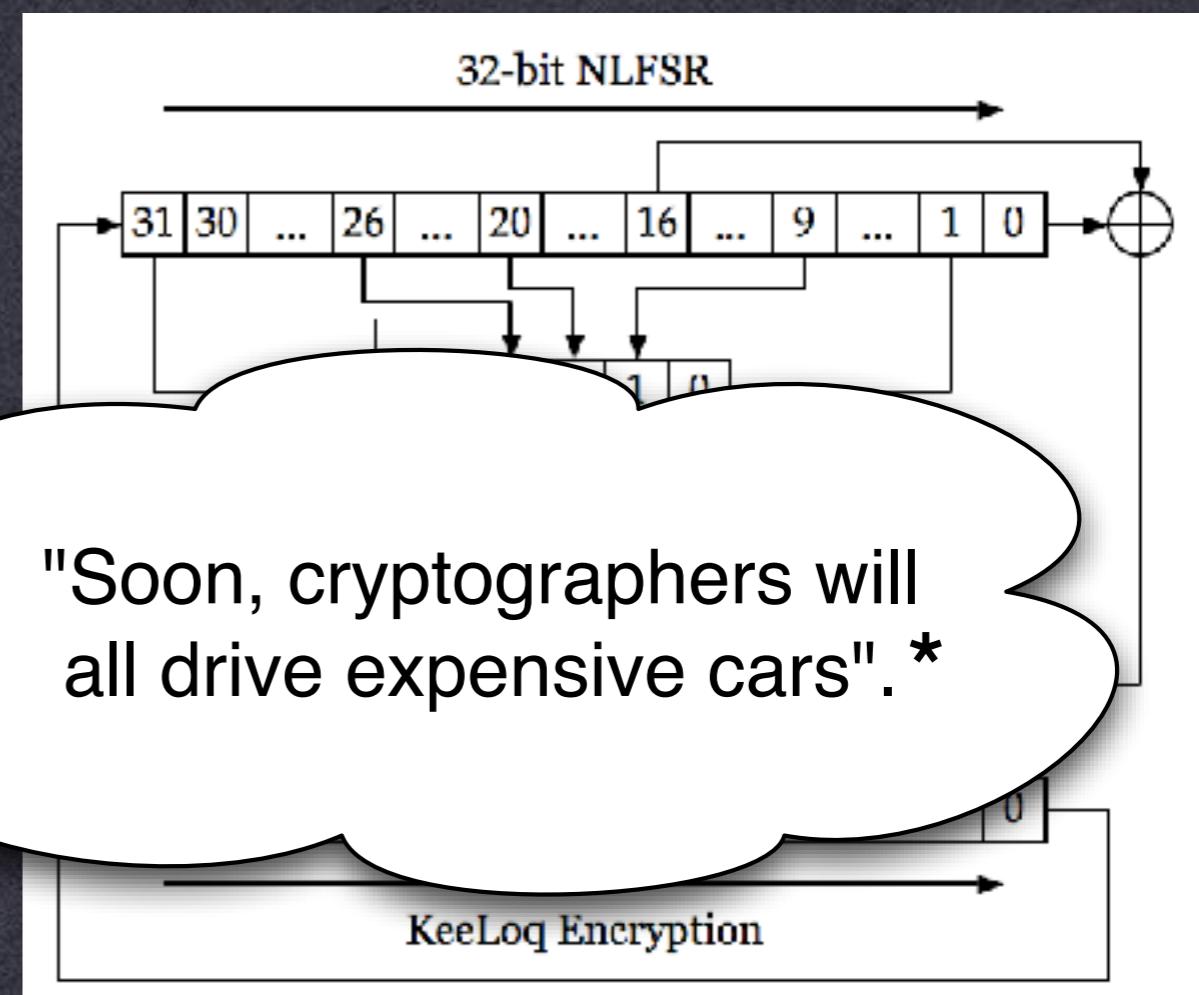
[Follow @mike_mimoso](#)

October 3, 2016 , 8:45 am

Logjam was one of several downgrade attacks discovered in the last 18 months that could theoretically allow a resourced attacker to take advantage of [lingering export-grade cryptography](#) to read and modify data over a supposedly secure connection.

KeeLoq

- Designed by Kuhn & Smit
 - Block cipher: 64-bit key, 32-bit block
 - 528 rounds
 - Mostly used for door opening
 - Direct attacks*:
 - A. 2^{16} data & 2^{51} operations
 - B. 2^{32} data & 2^{27} operations
 - C. Indesteege et al.:
(65 min to get data,
218 CPU days to process)

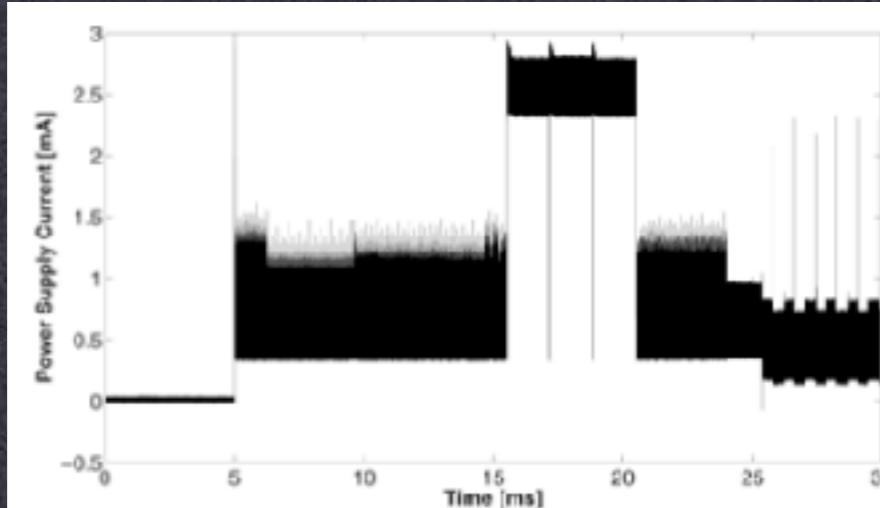


KeeLoq

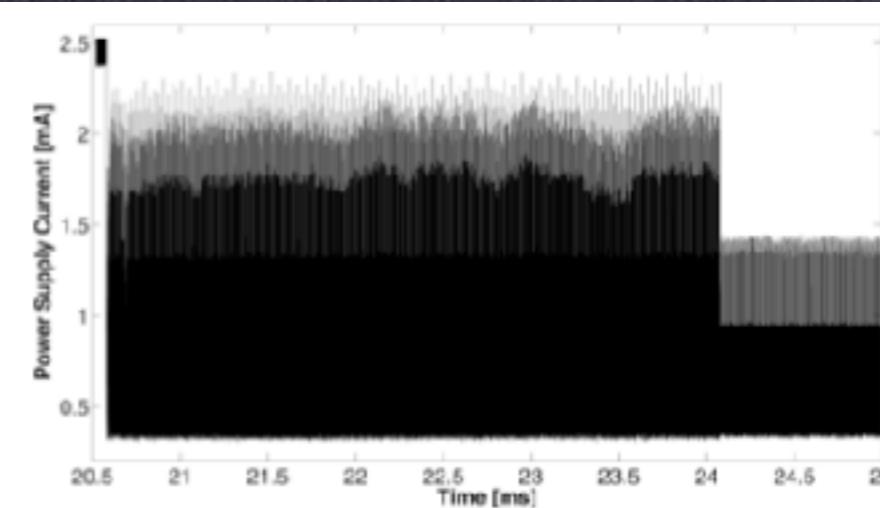
- **Newer attack, CRYPTO 2008**
 - **Requires physical access to device**
 - **Recovers secret key after 10-60 reads**
 - **How?**

KeyLoq

- CRYPTO 2008
 - Doesn't directly attack the cipher
 - Instead, measures how the cipher works in operation.
 - In this case, use power consumption

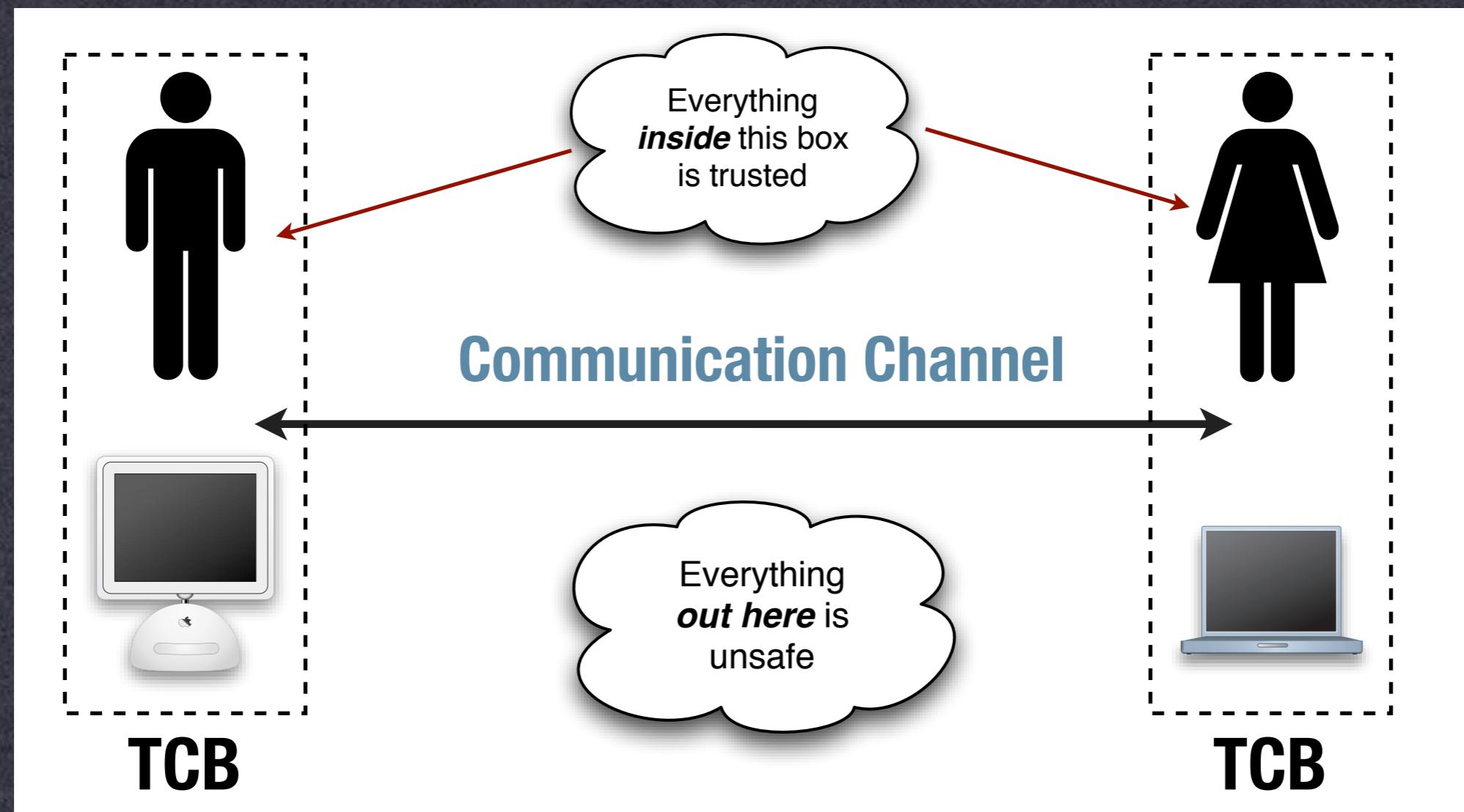


(a) From power up to start sending

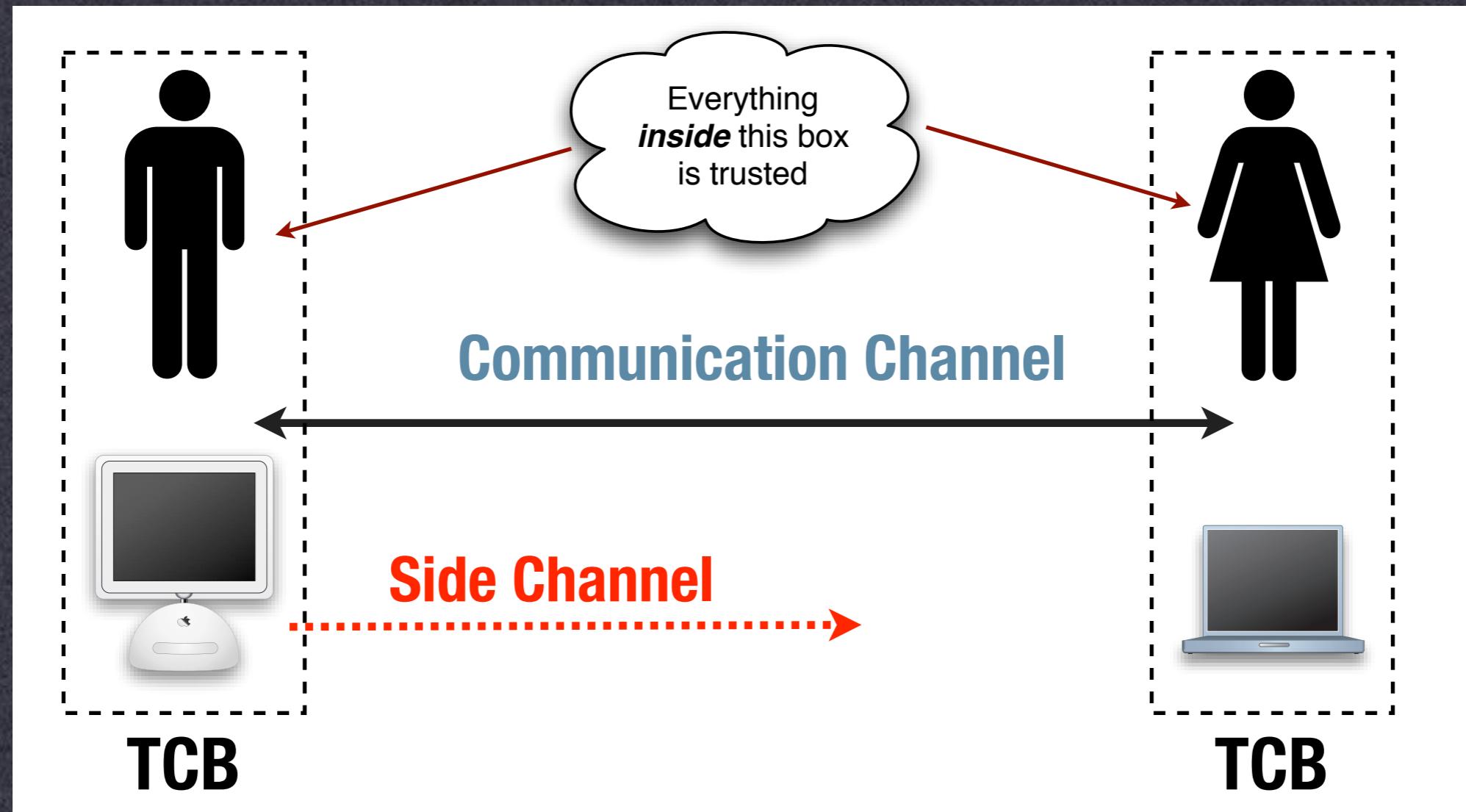


(b) Encryption part

Review

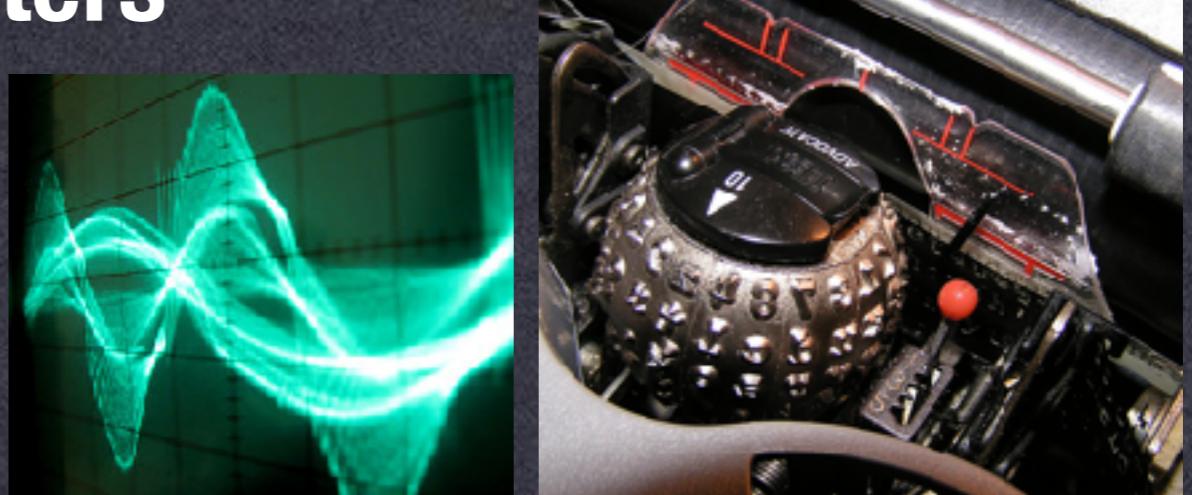


Side Channels



Side Channels

- Some history:
 - 1943: Bell engineer detects power spikes from encrypting teletype
 - 1960s: US monitors EM emissions from foreign cipher equipment
 - 1980s: USSR places eavesdropping device inside IBM Selectric typewriters



Selectric image by Flickr user relvax used under a Creative Commons license.

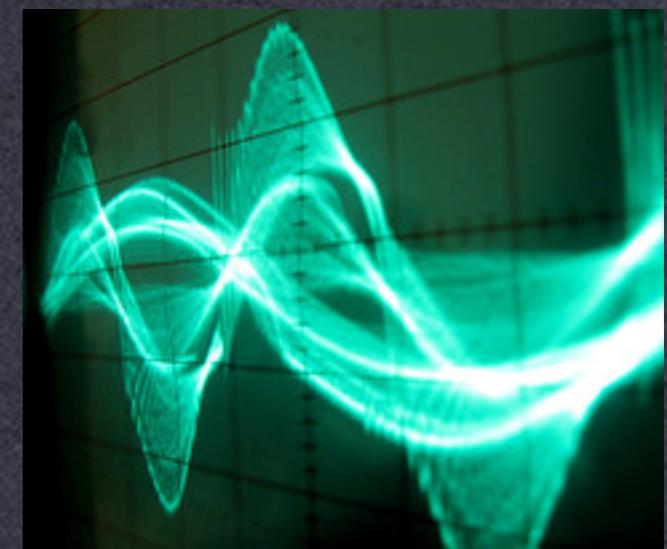
Side Channels

- Some history:
 - 1990s: Paul Kocher demonstrates timing attacks, power analysis attacks against RSA, Elgamal



Common Examples

- Timing
- Power Consumption
- RF Emissions
- Audio



RSA Cryptosystem

$$N = p \cdot q$$

$$e \cdot d \equiv 1 \pmod{(p-1)(q-1)}$$

Encryption

$$c = m^e \pmod{N}$$

Decryption

$$m = c^d \pmod{N}$$

RSA Cryptosystem

$$N = p \cdot q$$

$$e \cdot d \equiv 1 \pmod{(p-1)(q-1)}$$

Encryption

$$c = m^e \pmod{N}$$

Decryption

$$m = c^d \pmod{N}$$

$$m = \underbrace{c * c * c * \cdots * c}_{\text{d times}} \pmod{N}$$

Modular Exponentiation

$$m = c^d \bmod N$$

$$d = 1010101001110101011001001001001$$



```
modpow(c, d, N) {  
    result = 1;  
  
    for (i = 1 to |d|) {  
        if (the ith bit of d is 1) {  
            result = (result * c) mod N;  
        }  
  
        c = c2 mod N;  
    }  
  
    return result;  
}
```

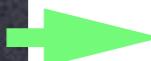
Modular Exponentiation

$$m = c^d \bmod N$$

$$d = 1010101001110101011001001001001$$



```
modpow(c, d, N) {  
  
    result = 1;  
  
    for (i = 1 to |d|) {  
        if (the ith bit of d is 1) {  
            result = (result * c) mod N;  
        }  
  
        c = c2 mod N;  
    }  
  
    return result;  
}
```



Modular Exponentiation

$$m = c^d \bmod N$$

$$d = 1010101001110101011001001001001$$



```
modpow(c, d, N) {  
  
    result = 1;  
  
    for (i = 1 to |d|) {  
        if (the ith bit of d is 1) {  
            result = (result * c) mod N;  
        }  
  
        c = c2 mod N;  
    }  
  
    return result;  
}
```



Modular Exponentiation

$$m = c^d \bmod N$$

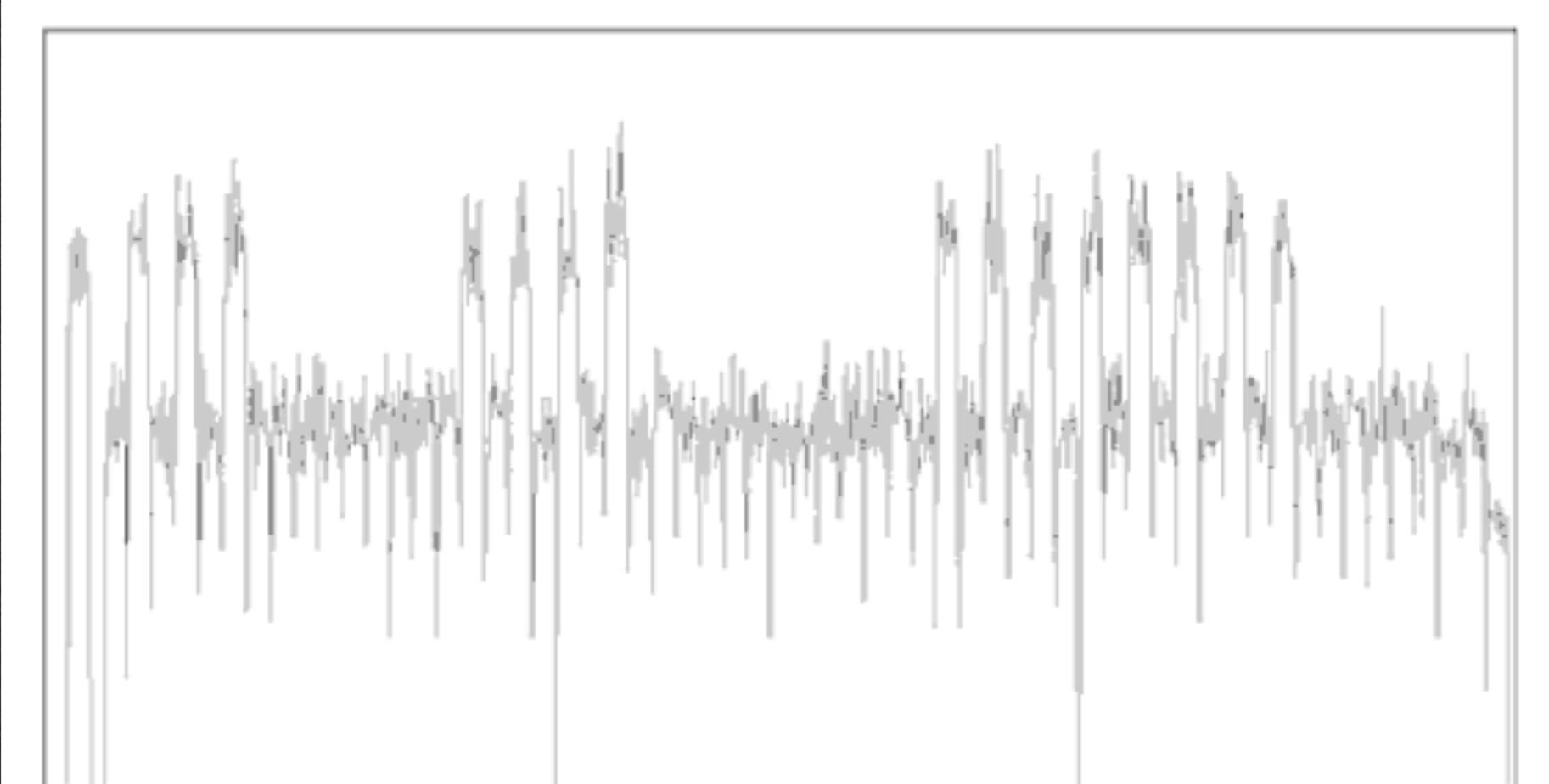
$d = 1010101001110101011001001001001$



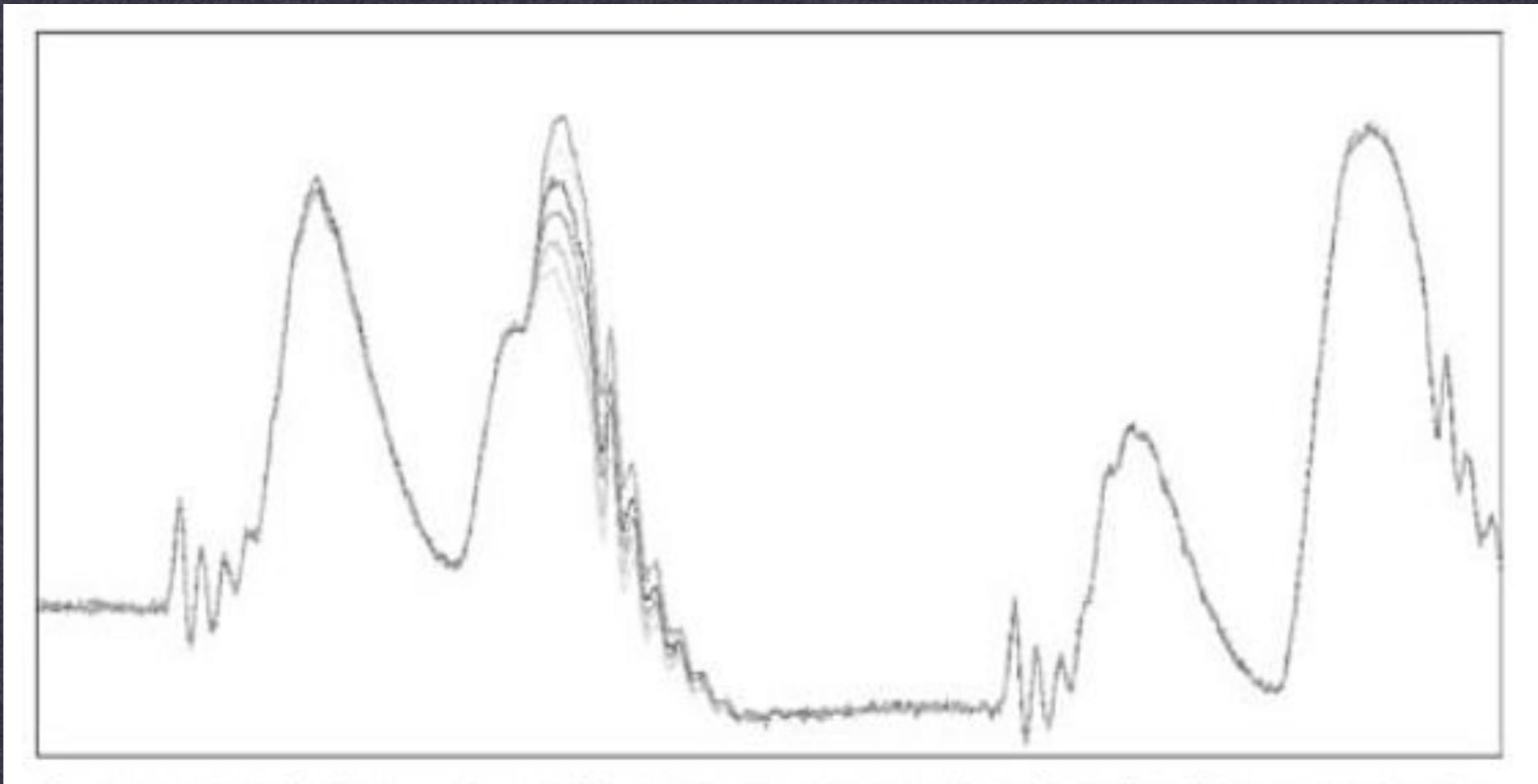
```
modpow(c, d, N) {  
  
    result = 1;  
  
    for (i = 1 to |d|) {  
        if (the ith bit of d is 1) {  
            result = (result * c) mod N; Expensive Operation  
        }  
  
        c = c2 mod N;  
    }  
  
    return result;  
}
```



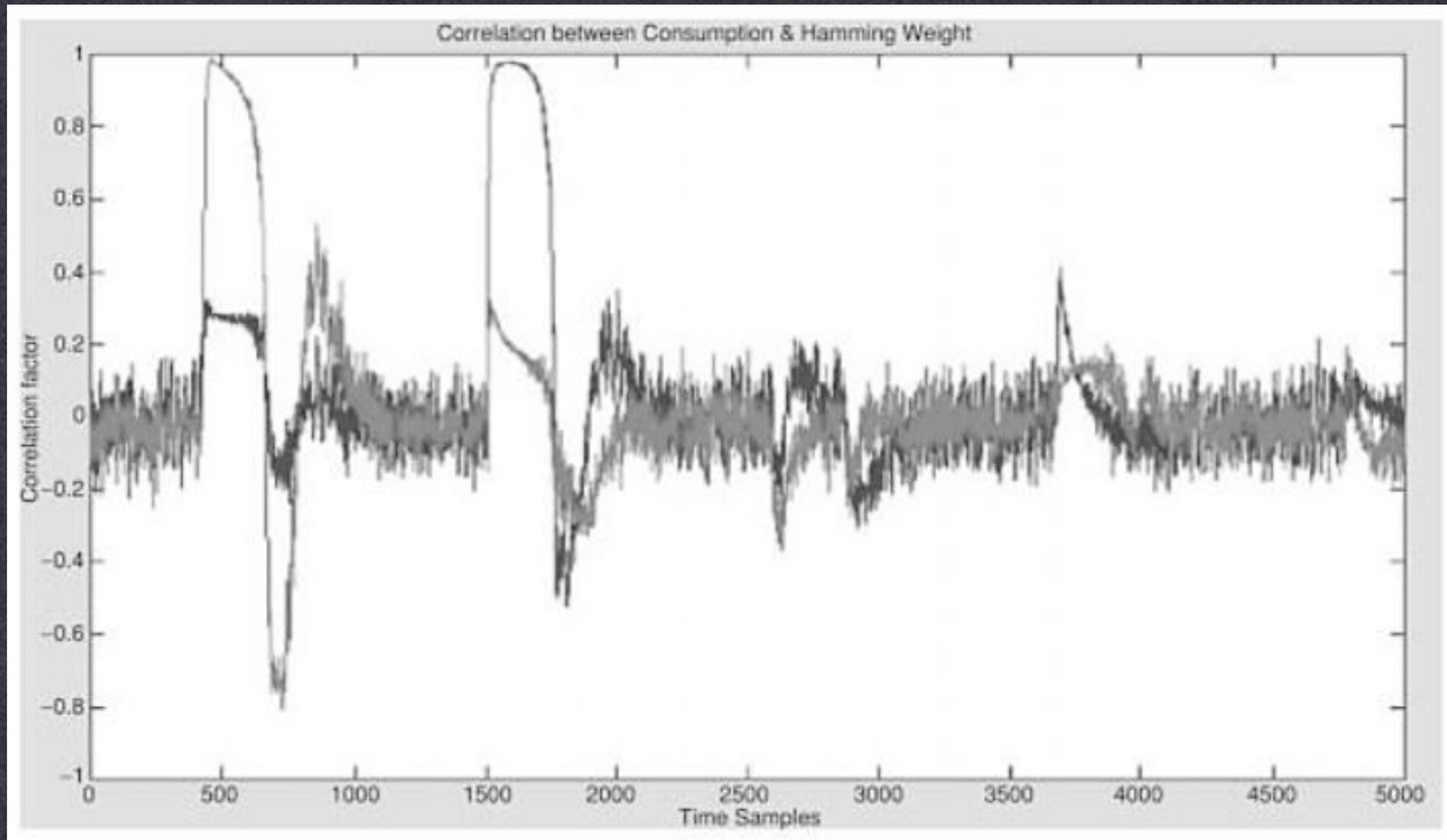
Simple Power Analysis



Differential Power Analysis



Differential Power Analysis



Modular Exponentiation

$$m = c^d \bmod N$$

$d = 1010101001110101011001001001001$



```
modpow(c, d, N) {  
    result = 1;  
  
    for (i = 1 to |d|) {  
        if (the ith bit of d is 1) {  
            result = (result * c) mod N;  
        }  
  
        c = c2 mod N;  
    }  
  
    return result;  
}
```

Expensive Operation

Modular Exponentiation

$$m = c^d \bmod N$$

$d = 1010101001110101011001001001001$



```
modpow(c, d, N) {  
  
    result = 1;  
  
    for (i = 1 to |d|) {  
        if (the ith bit of d is 1) {  
            result = (result * c) mod N;  
        }  
  
        c = c2 mod N;  
    }  
  
    return result;  
}
```

Expensive Operation

Modular Exponentiation

$$m = c^d \bmod N$$

$d = 1010101001110101011001001001001$



```
modpow(c, d, N) {  
  
    result = 1;  
  
    for (i = 1 to |d|) {  
        if (the ith bit of d is 1) {  
            result = (result * c) mod N; Expensive Operation  
        }  
  
        c = c2 mod N;  
    }  
  
    return result;  
}
```

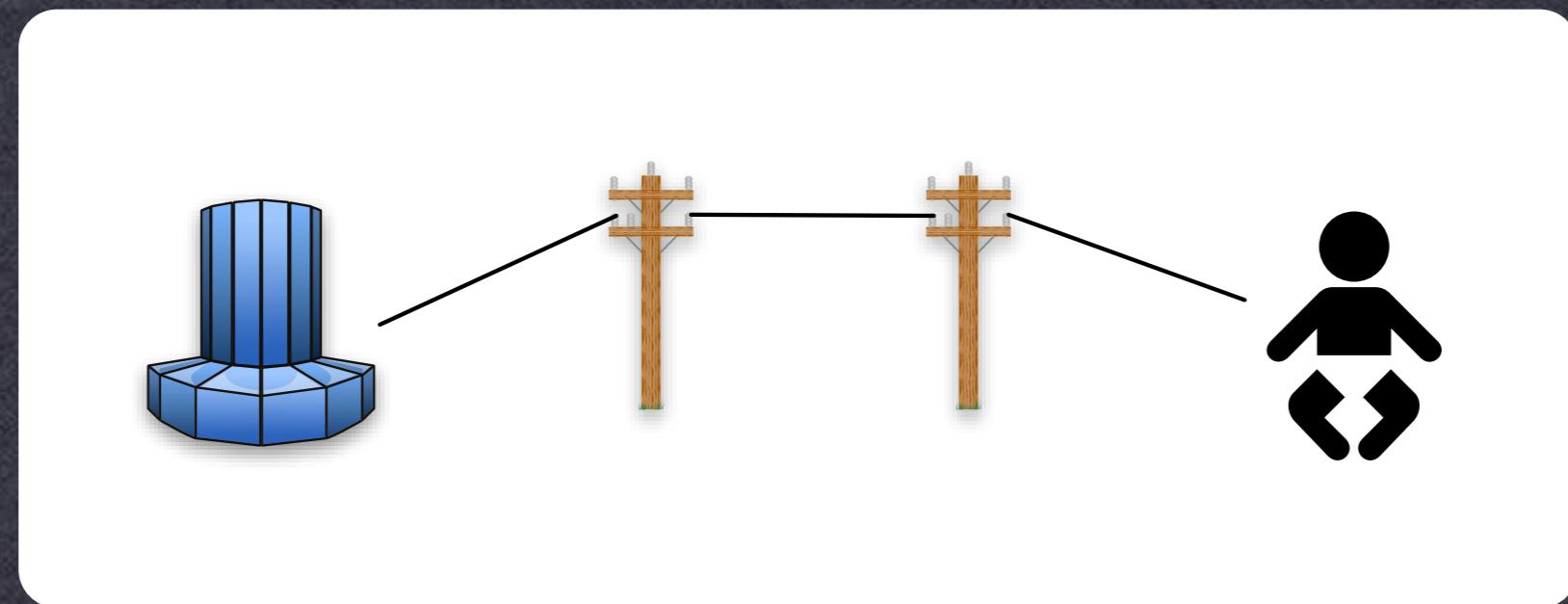


Kocher's Timing Attack

- Assume that for some values of $(a * b)$, multiplication takes unusually long
- Given the first b bits, compute intermediate value “result” up to that point
- If the next bit of $d = 1$, then calc is $(\text{result} * c)$
- If this is expected to be slow, but response is fast then the next bit of $d \neq 1$

Remote Timing Attacks

- Boneh & Brumley
 - Remote attack on RSA-CRT as implemented in OpenSSL
 - Optimization, uses knowledge of p, q



Solutions

- Quantization:
 - All operations take the same time
 - Hard to do without sacrificing performance
- Blinding:
 - Prevents attacker from selecting ciphertext (that is processed with the secret key)

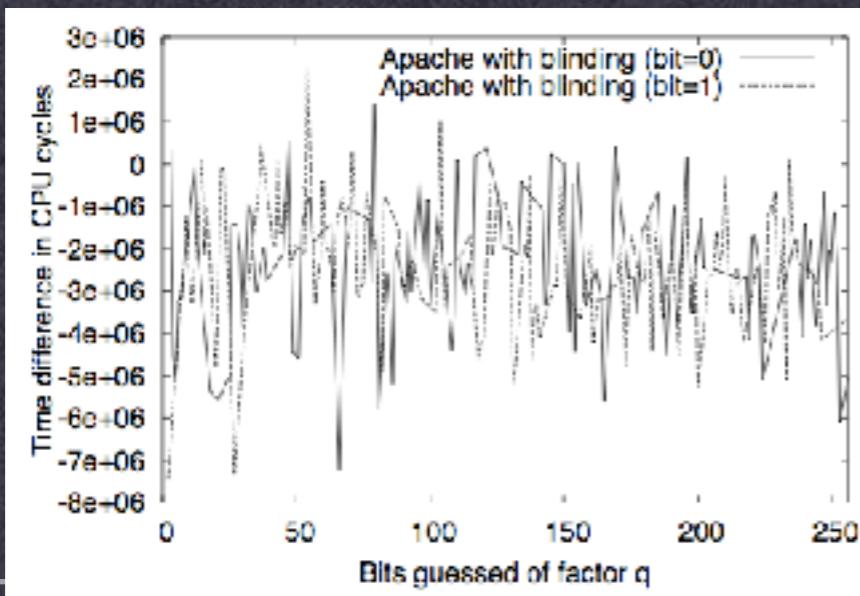
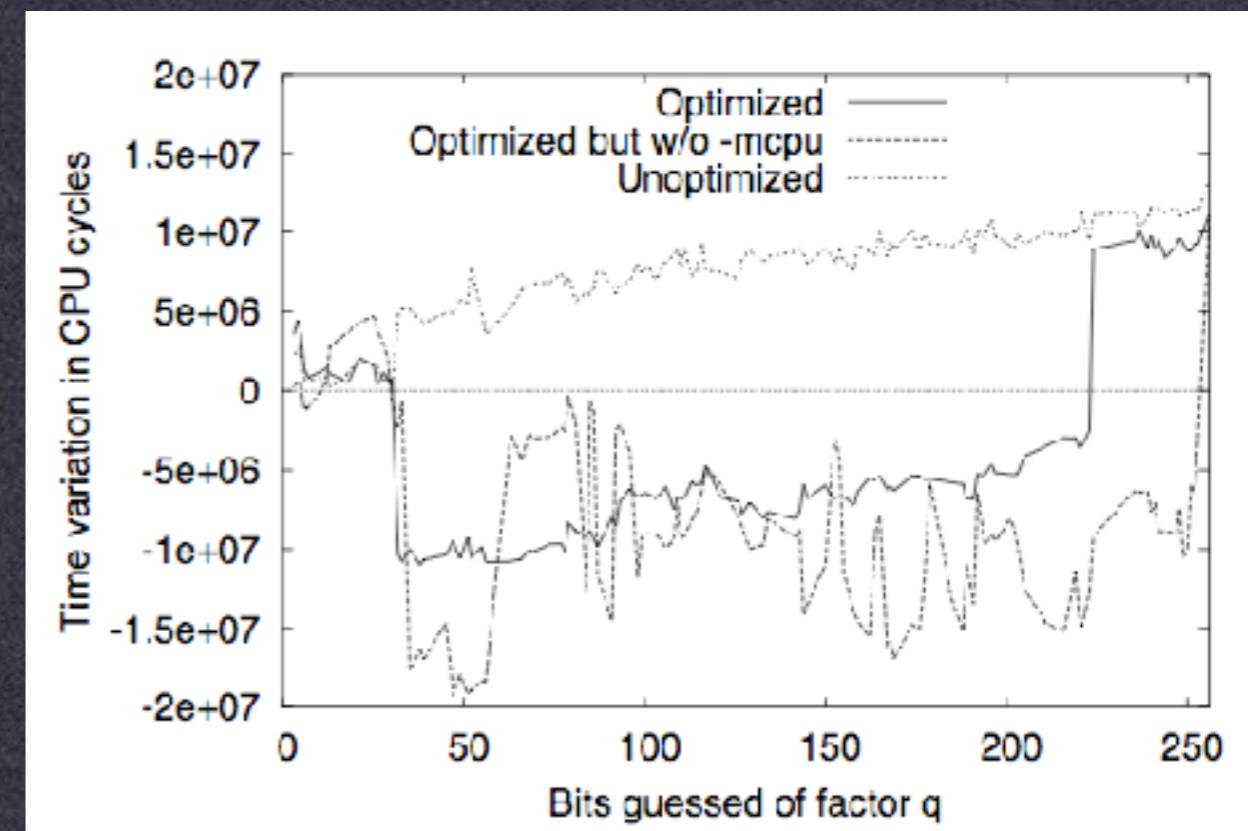
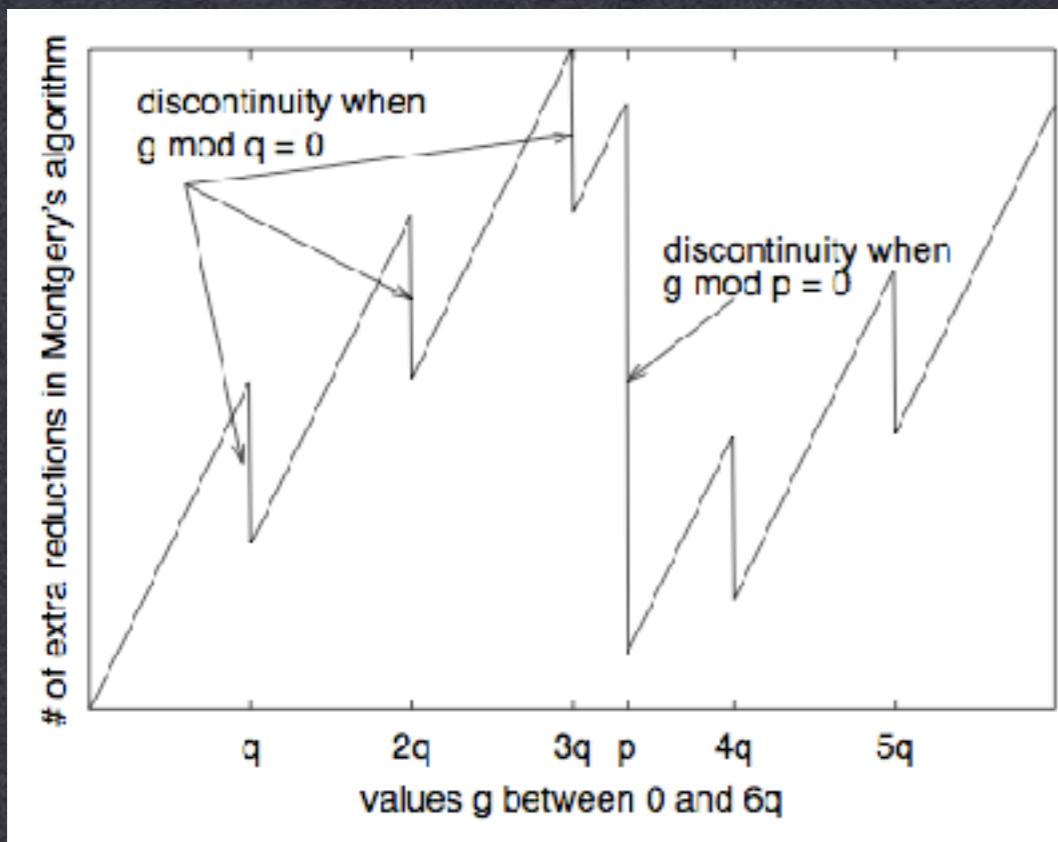


Image Source: Boneh, Brumley: Remote Timing Attacks are Practical

Remote Timing Attacks

- Boneh & Brumley
 - By repeating the timing measurements, they were able to extract a secret key after several million samples



Source: Boneh, Brumley: Remote Timing Attacks are Practical

Windowed Exponentiation

$$m = c^d \bmod N$$

$d = 1010101001110101011001001001001$

- Handles the input in larger “chunks”
 - Fixed or variable sized
 - Can speed up by pre-computing some values
 - e.g., c^2, c^3, \dots, c^{16}

Windowed Exponentiation

$$m = c^d \bmod N$$

$$d = 10101010011101010110010\boxed{01001001}$$

- Handles the input in larger “chunks”
 - Fixed or variable sized
 - Can speed up by pre-computing some values
 - e.g., c^2, c^3, \dots, c^{16}

Windowed Exponentiation

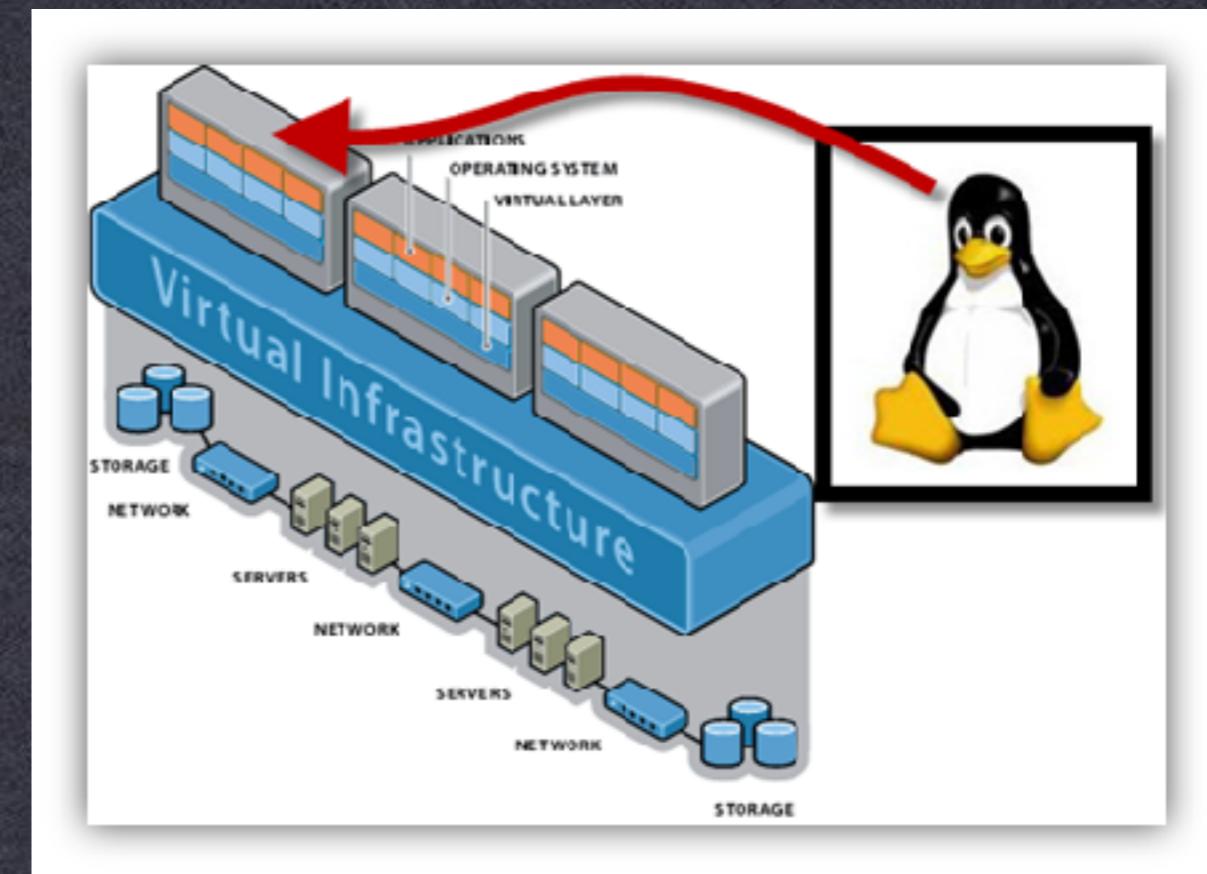
$$m = c^d \bmod N$$

$d = 1010101001110101011$ **001001001001**

- Handles the input in larger “chunks”
 - Fixed or variable sized
 - Can speed up by pre-computing some values
 - e.g., c^2, c^3, \dots, c^{16}

Cache Timing Attacks

- Observation
 - When we use pre-computed tables, this implies a memory access
 - This takes time
 - Unless the data is cached!

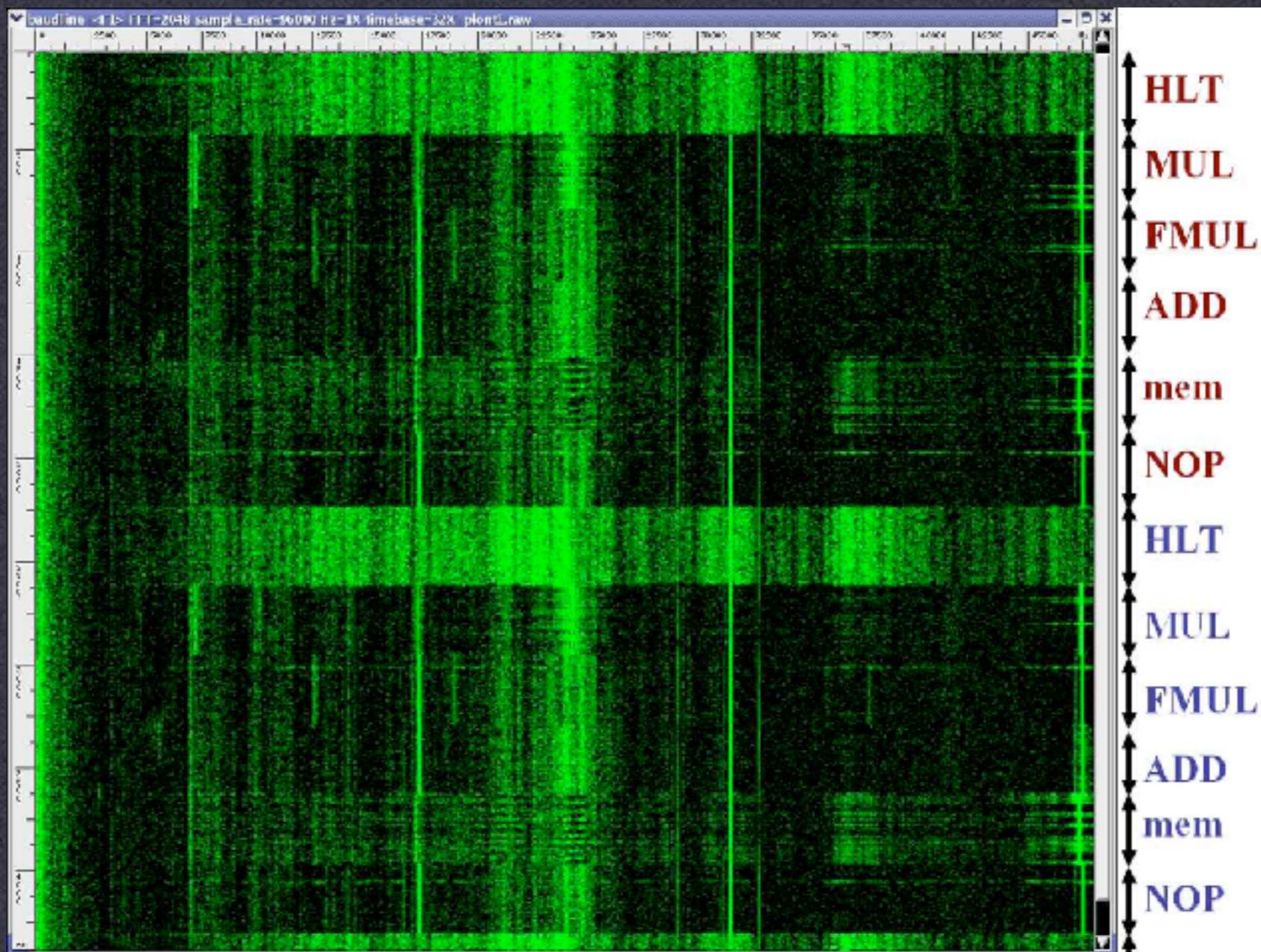


Hypervisor attacks

- Observation
 - This applies to code too!

```
SquareMult( $x, e, N$ ):  
    let  $e_n, \dots, e_1$  be the bits of  $e$   
     $y \leftarrow 1$   
    for  $i = n$  down to 1 {  
         $y \leftarrow \text{Square}(y)$  (S)  
         $y \leftarrow \text{ModReduce}(y, N)$  (R)  
        if  $e_i = 1$  then {  
             $y \leftarrow \text{Mult}(y, x)$  (M)  
             $y \leftarrow \text{ModReduce}(y, N)$  (R)  
        }  
    }  
    return  $y$ 
```

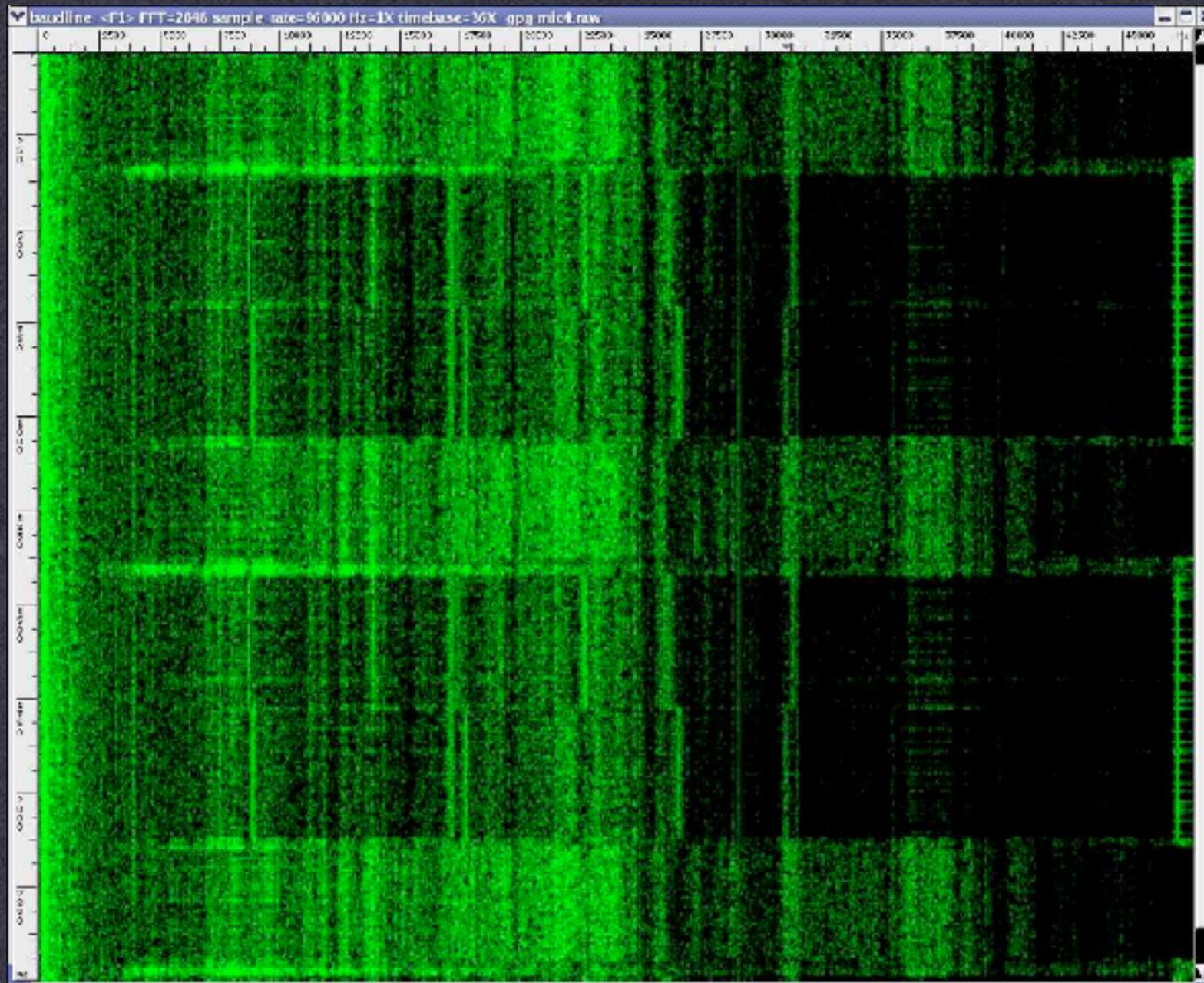
Acoustic Side Channels



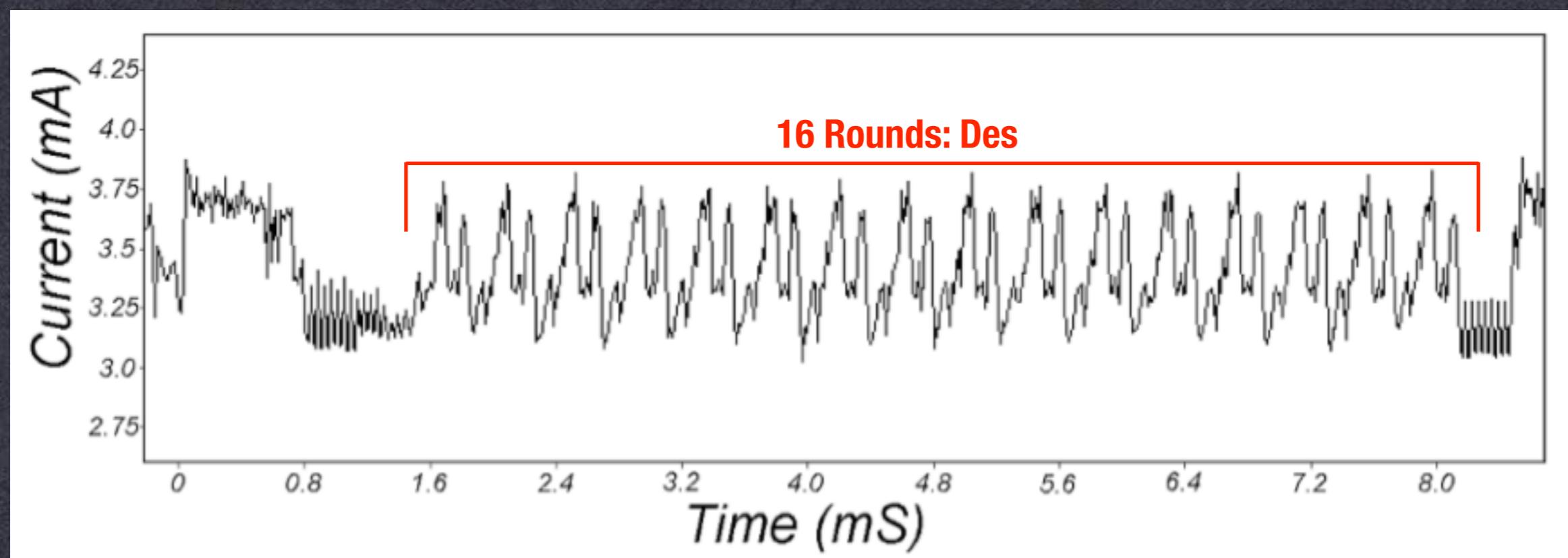
Acoustic Side-Channels

GPG RSA-CRT
signature #1:

(repeated)



Reverse Engineering



Back to KeyLoq

- KeyLoq attack has an unhappy coda:
 - Turns out that all keys for a given manufacturer are derived from the same MK
$$k = \text{pad}(ID, seed) \oplus MK$$
 - Key derivation function based on XOR :(
 - By observing 2 interactions between key/car we can derive the MK and thus steal any car