“Ooh Aah... Just a Little Bit”: A small amount of side channel can go a long way

Naomi Benger Joop van de Pol Nigel Smart
Yuval Yarom

Outline

- Background
  - ECDSA
  - wNAF scalar multiplication
  - Hidden Number Problem
- The FLUSH+RELOAD Technique
- Attacking OpenSSL ECDSA

ECDSA

Signer has a private key $1<\alpha<q-1$ and a public key $Q=[\alpha]G$

1. Compute $h=\mathrm{Hash}(m)$
2. Randomly select an ephemeral key $1<k<q$
3. Compute $(x,y)=[k]G$
4. Take $r=x \bmod q$; If $r=0$ repeat from 2
5. Take $s=(h+r\cdot\alpha)/k \bmod q$; if $s=0$ repeat from 2
6. (r,s) is the signature

Note that $$k=(r/s)\times a+(h/s) \bmod q$$

wNAF Form

To compute $[d]G$, first write d in wNAF form:

$$d = \sum_{i=0}^{n-1} d_i 2^i \text{ for } d_i \in \{0, \pm 1, \pm 3, \ldots, \pm(2^w - 1)\}$$

Such that if $d_i \neq 0$ then $d_{i+1} = \ldots = d_{i+w+1} = 0$.

Scalar Multiplication with wNAF form

Precompute $\{\pm G, \pm[3]G, \ldots, \pm[2^w\text{-}1]G\}$

x=0
for i=n-1 **downto** 0
  x = Double(x)
  if ($d_i \neq 0$) **then**
    x = Add(x, $[d_i]G$)
  end
end
return x

The Hidden Number Problem

Suppose we know numbers t_i, u_i such that

$$\left| \alpha t_i - u_i \right|_q < q / 2^z$$

The Hidden Number Problem

Suppose we know numbers t_i, u_i such that

$$\left|\alpha t_i - u_i\right|_q < q/2^z$$

We can construct a lattice

$$\begin{pmatrix} 2^z \cdot q & & & \\ & \ddots & & \\ & & 2^z \cdot q & \\ 2^z \cdot t_1 & \cdots & 2^z \cdot t_d & 1 \end{pmatrix}$$

And a vector

$$\left(2^z \cdot u_1, \ldots, 2^z \cdot u_d, 0\right)$$

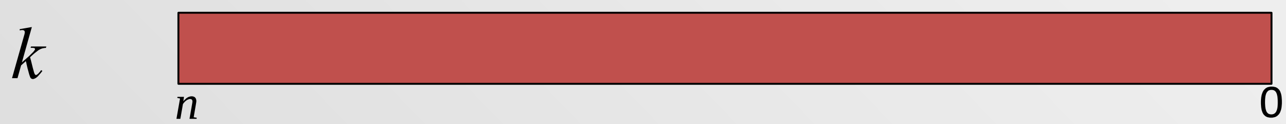
Which is very close to a lattice vector that depends on α.

HNP and ECDSA

Recall that $k = (r/s) \times a + (h/s) \bmod q$

We want $\left| a t_i - u_i \right|_q < q / 2^z$

In terms of k:



Or in terms of t_i and u_i:

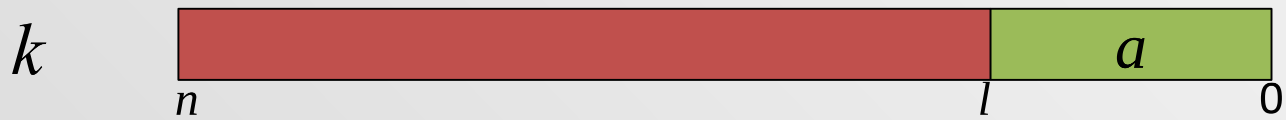
$$\left| (r/s) \times a + (h/s) \right|_q = k$$

HNP and ECDSA

Recall that $k = (r/s) \times a + (h/s) \bmod q$

We want $\left| a t_i - u_i \right|_q < q/2^z$

In terms of k:



Or in terms of t_i and u_i:

$$\left| (r/s) \times a - (-(h/s)) \right|_q < q$$

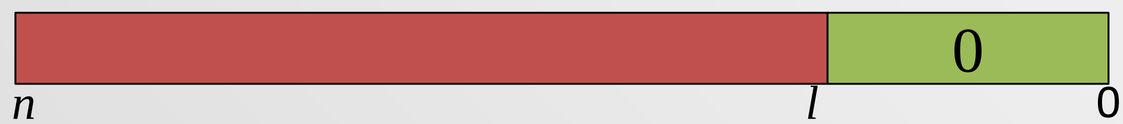
HNP and ECDSA

Recall that $k = (r/s) \times \alpha + (h/s) \bmod q$

We want $\left| \alpha t_i - u_i \right|_q < q / 2^z$

In terms of k:

$k - a$



Or in terms of t_i and u_i:

$$\left| (r/s) \times \alpha - \left(a - (h/s)\right) \right|_q < q$$

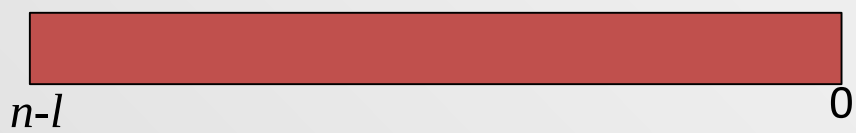
HNP and ECDSA

Recall that $k = (r/s) \cdot \alpha + (h/s) \bmod q$

We want $\left|\alpha t_i - u_i\right|_q < q/2^z$

In terms of k:

$$(k - a)/2^l$$



Or in terms of t_i and u_i:

$$\left|\left((r/s) \cdot \alpha - \left(a - (h/s)\right)\right)/2^l\right|_q < q/2^l$$

HNP and ECDSA

Recall that $k = (r/s)\times\alpha + (h/s) \bmod q$

We want $\left|\alpha t_i - u_i\right|_q < q/2^z$

In terms of k:

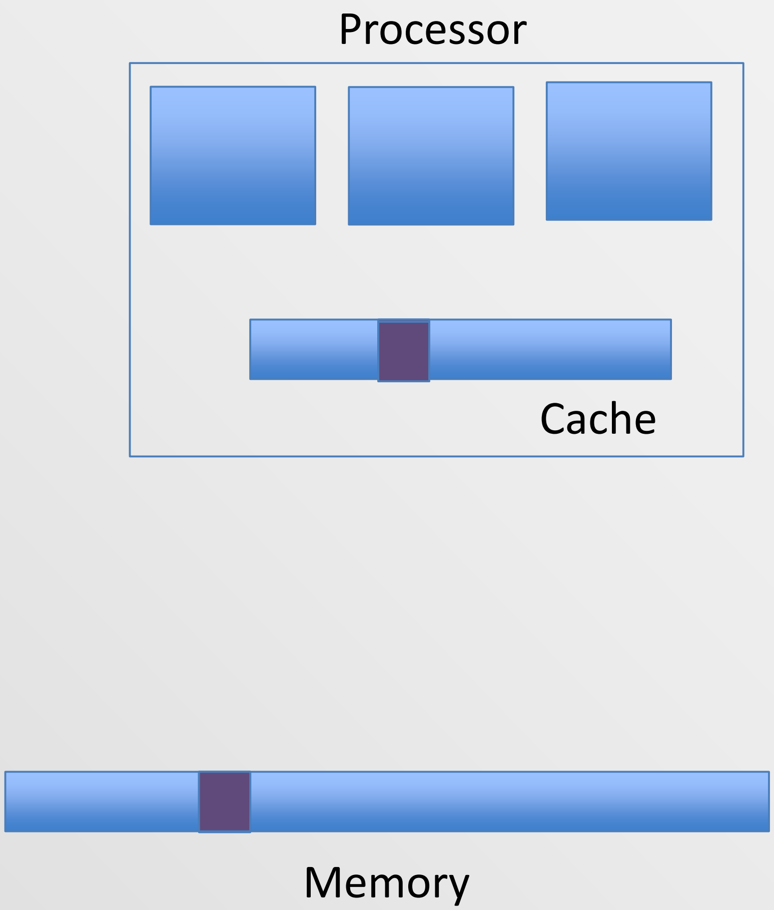
$$\left|(k - a - q/2)/2^l\right|_q < q/2^{l+1}$$

Or in terms of t_i and u_i:

$$\left|\left((r/s)\times\alpha - \left(a - (h/s) + q/2\right)\right)/2^l\right|_q < q/2^{l+1}$$

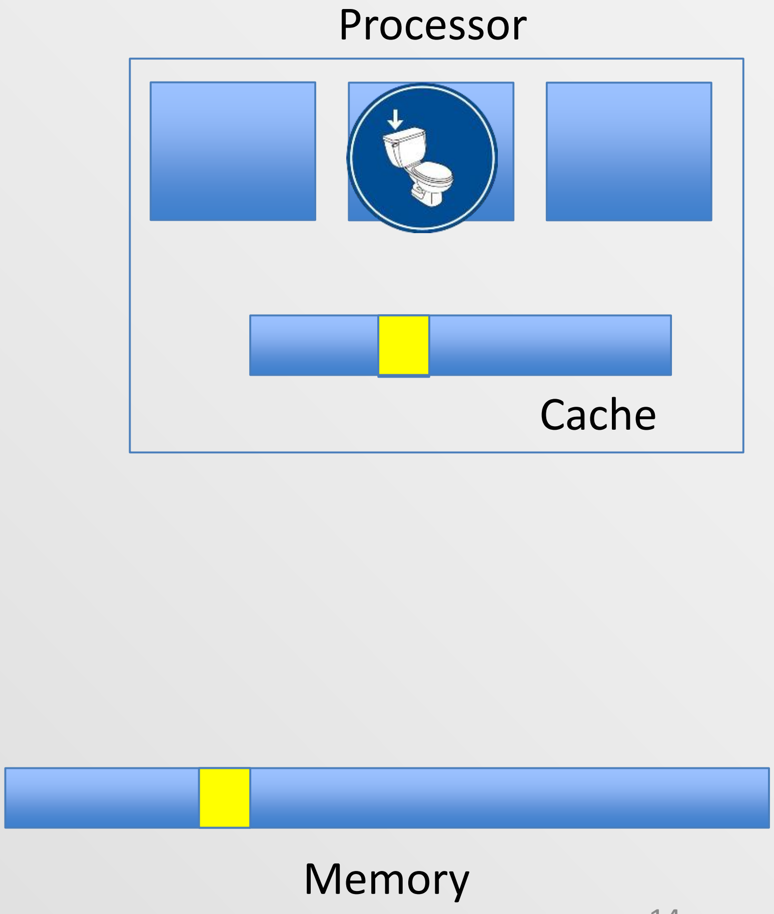
The X86 Cache

- Memory is slower than the processor
- The cache utilises locality to bridge the gap
  - Divides memory into *lines*
  - Stores recently used lines
- Shared caches improve performance for multi-core processors



Cache Consistency

- Memory and cache can be in inconsistent states
  - Rare, but possible
- Solution: Flushing the cache contents
  - Ensures that the next load is served from the memory

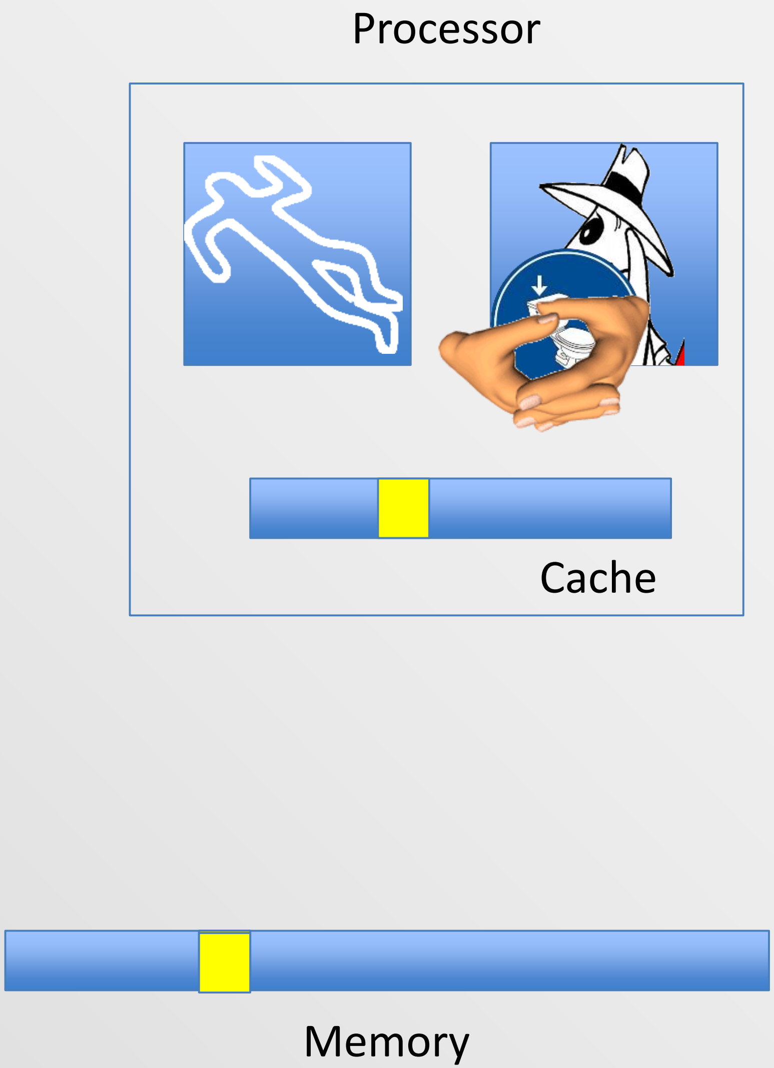


The FLUSH+RELOAD Technique

- Exploits cache behaviour to leak information on victim access to shared memory.
  - Shared text segments
  - Shared libraries
  - Memory de-duplication
- Spy monitors victim’s access to shared code
  - Spy can determine what victim does
  - Spy can infer the data the victim operates on

FLUSH+RELOAD

- **FLUSH** memory line
- Wait a bit
- Measure time to **RELOAD** line
  - slow-> no access
  - fast-> access
- Repeat



Attacking OpenSSL wNAF

- Achieve sharing of the victim code
- Use FLUSH+RELOAD to recover the double and add chain of the wNAF calculation
- Divide time into slots of 1200 cycles (about 0.4μs)
- In each slot, probe a memory line in the code of the Double and Add functions.

Sample Trace

Raw:

D| | | |D|D| | |D| | | |A| | | |D| | |D| | | |D| | |D| | |A|A| | |D| | |D| | |
|D| | |D| | | |D| | |D| | |A| | | |D| | |D|D| | |D| | |D| | | |D| |A|A| | |D
| | |D|D| | |D| | |D| | |A| | | |D| | |D| | |D| | |D| | |D| | |A|A| | | |D| |
|D| | |D| | | |D| |A|A| | |D| | | |D| | |D| | |D| | |A| | |D| | |D| | |D|D|
| |D| | |D| | |A| | | |D| | |D| | |D| | |D|D| | |D| | |D| | |D| | |A| | |D|D
| | |D| |…

Processed:

DDDADDDDADDDDDDADDDDDADDDDADDDDDADDDDADDDDADDDDDADDD
DDDDADDDDADDDDADDDDDADDDDADDDDDDADDDDDDADDDDADDDDDAD
DDDADDDDADDDDADDDDDADDDDDADDDDADDDDADDDDADDDDADDDDDAD
DDDADDDDDDDADDDDDADDDDADDDDDDADDDDADDDDDDADDDDDDADDDD
ADDDDDADDDDDADDDDDADDDDDADDDDXDDDADDDDADDDDADDDDADDD
DDADDDDADDDDDDADDDDDADDDDADDDDDDADDDDDDADDDDADD

Using the LSBs

The trace: `DDDADDDDADDDDD…DDDDDDADDDDD`**ADD**

Reveals 3 LSBs (100). A different trace might reveal fewer bits. How do we deal with that?

$$\begin{pmatrix} 2^z \cdot q & & & \\ & \ddots & & \\ & & 2^z \cdot q & \\ 2^z \cdot t_1 & \cdots & 2^z \cdot t_d & 1 \end{pmatrix}$$

$$\left(2^z \cdot u_1, \ldots, 2^z \cdot u_d, 0\right)$$

Using the LSBs

The trace: `DDDADDDDADDDDD…DDDDDDADDDD`**ADD**

Reveals 3 LSBs (100). A different trace might reveal fewer bits. How do we deal with that?

$$\begin{pmatrix} 2^{z_1}\cdot q & & & \\ & \ddots & & \\ & & 2^{z_d}\cdot q & \\ 2^{z_1}\cdot t_1 & \cdots & 2^{z_d}\cdot t_d & 1 \end{pmatrix}$$

$$\left(2^{z_1}\cdot u_1, \ldots, 2^{z_d}\cdot u_d, 0\right)$$

We vary the z per (t_i, u_i) tuple.

Results

- Previous: Liu and Nguyen 2013 – 160 bit key, 100 signatures, 2 known bits

Results

- Previous: Liu and Nguyen 2013 – 160 bit key, 100 signatures, 2 known bits
- Our results: against secp256k1

Expected # Sigs	d	Time (s)	Success Prob.	Time / Prob.
200	100	611.13	.035	17460
220	110	79.67	.020	3933
240	60	2.68	.005	536
260	65	2.26	.055	41
280	70	4.46	.295	15
300	75	13.54	.530	26

Summary

- FLUSH+RELOAD extracts the double-and-add chains with almost no errors
- We can use a variable number of bits in the lattice attack
- We can break a 256 bit key by obtaining less than 256 signatures.