Using Bleichenbacher's Solution to the Hidden Number Problem to Attack Nonce Leaks in 384-Bit ECDSA.

Elke De Mulder, Michael Hutter, Mark E. Marson, Peter Pearson

CRYPTOGRAPHY™ RESEARCH
a division of Rambus

IAIK TU Graz

Overview of the talk

- The device and implementation being evaluated
- Previous work attacking nonce leaks
- Details of Bleichenbacher's attack
- Results
- Further research suggestions

The device and implementation being evaluated

- The device
  - BasicCard Family of Smart Cards: ZeitControl (German)
  - Card has built in curves from ECC-Brainpool

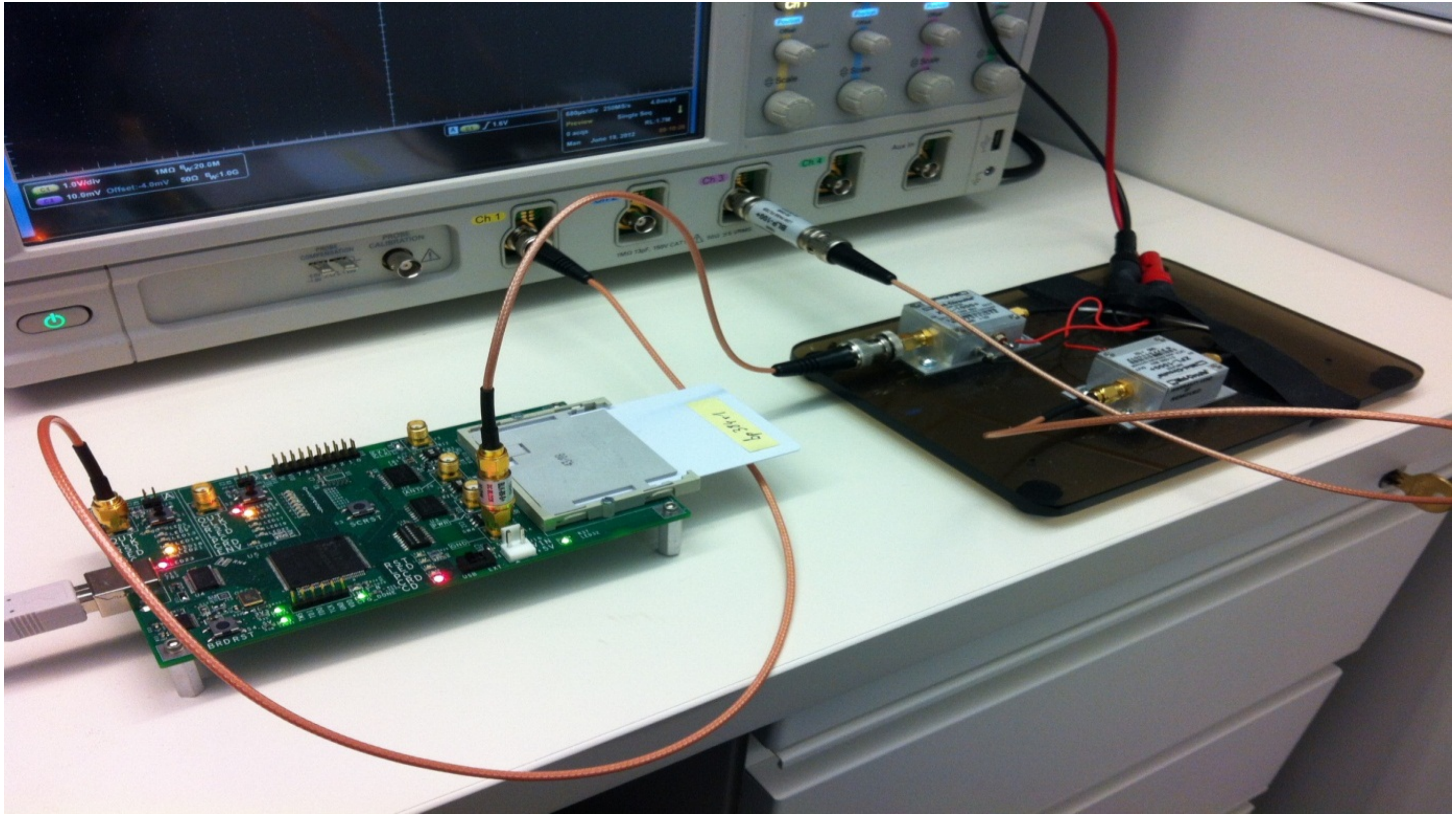
- The Implementation
  - ECDSA over a 384-bit prime field curve, BrainpoolP384r1
  - Values in Montgomery representation for efficient arithmetic
  - Curve points represented in Jacobian projective coordinates
  - Scalar multiplications computed on the curve twist BrainpoolP384t1
  - Scalar multiplication uses the signed comb method with 7 teeth
  - 64 pre-computed points in memory for point additions; points for subtractions computed on the fly

The device and implementation being evaluated

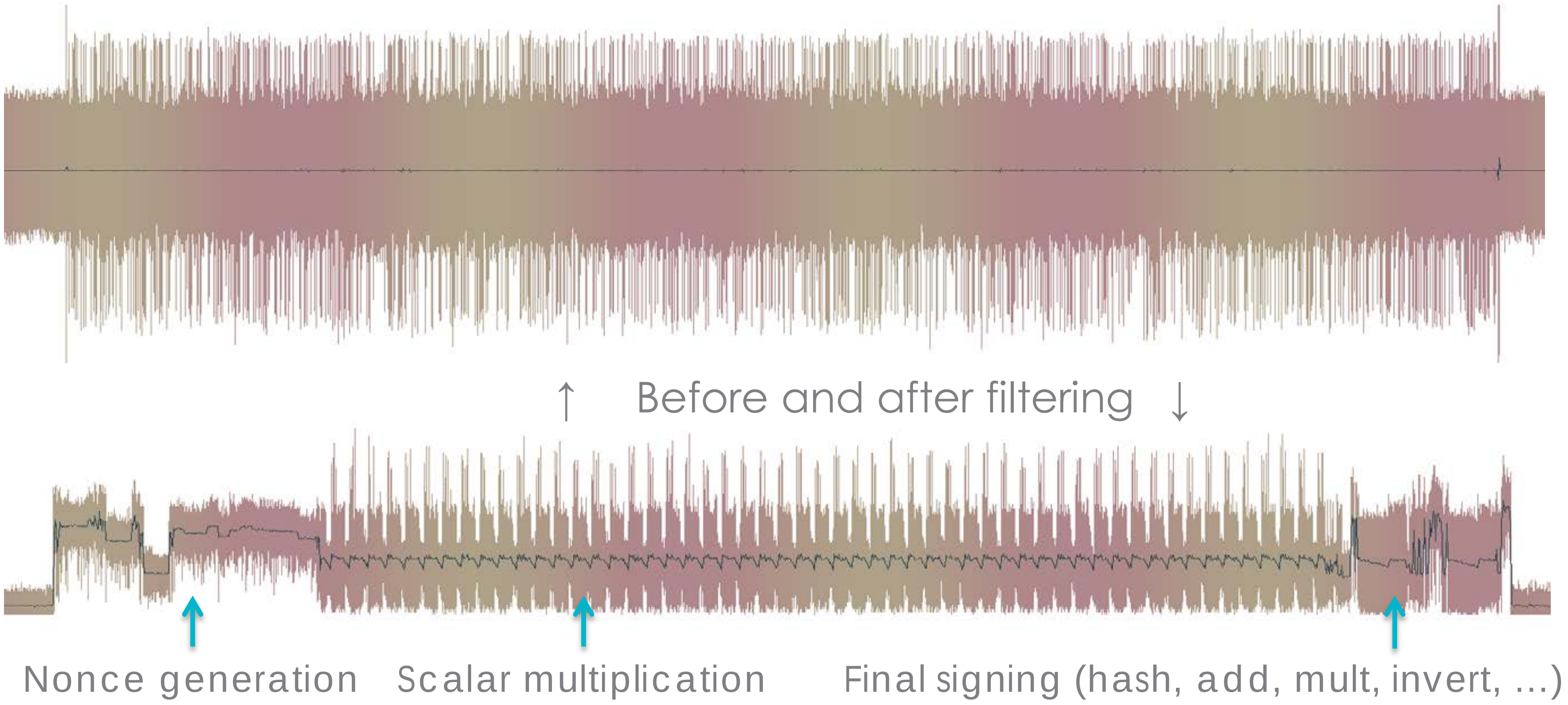
- Let E be an elliptic curve defined over $\mathbf{F}_p$ and G an element of order q ($q * G = \mathbf{O}$) in E. Let H denote the hash of the message m to be signed.

- **ECDSA Signature Generation**
  - Generate a random secret nonce K, $0 < K < q$, and computes $K * G = (u,v)$
  - Compute $r = u \bmod q$
  - Compute $s = K^{-1} * (H + r*x) \bmod q$
  - Signature of m is (r, s)

The device and implementation being evaluated



The device and implementation being evaluated



Analyzing the modular inversion of the nonce

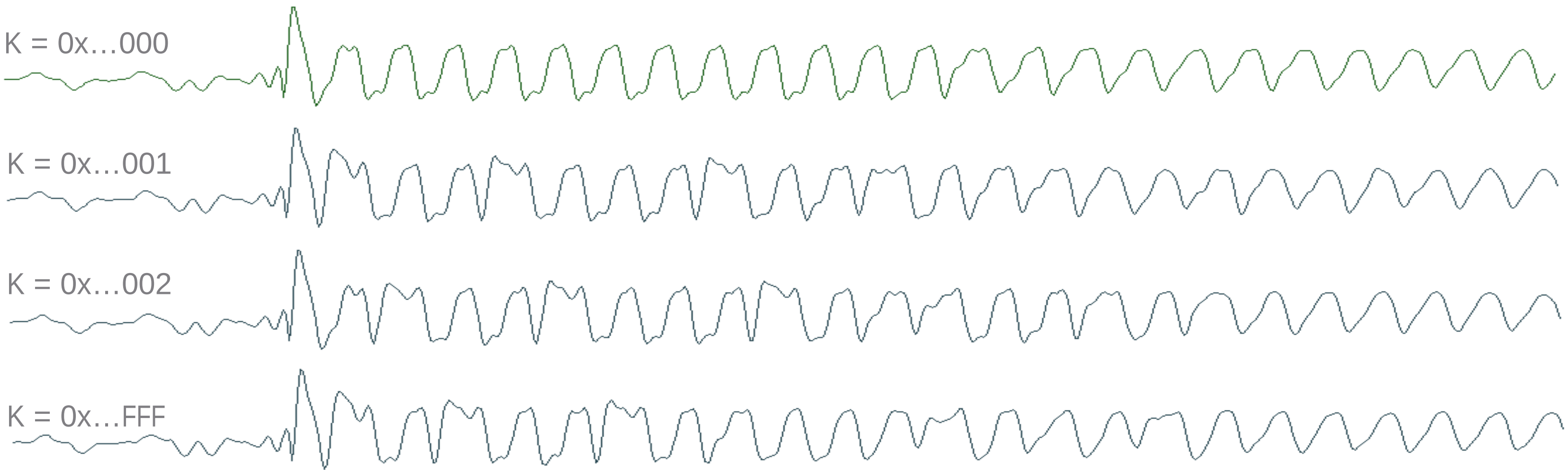
- We used DPA to identify the inversion algorithm
- Binary inversion algorithm
  - Due to R. Lórencz (CHES 2002)
  - Modified M. Penk algorithm

Input: $a \in [1, p-1]$ and p

Output: $r \in [1, p-1]$ and k, where $r = a^{-1} \bmod p$ and $n \le k \le 2n$

```
u := p, v := a, r := 0, s := 1
k := 0
while (v > 0)
    if (u is even) then
        if (r is even) then
            u := u/2, r := r/2, k := k + 1
        else
            u := u/2, r := (r + p)/2, k := k + 1
    else if (v is even) then
        if (s is even) then
            v := v/2, s := s/2, k := k + 1
        else
            v := v/2, s := (s + p)/2, k := k + 1
    else x := (u - v)
        if (x > 0) then
            u := x, r := r - s
            if (r < 0) then
                r := r + p
        else
            v := -x, s := s - r
            if (s < 0) then
                s := s + p
if (r > p) then
    r := r - p
if (r < 0) then
    r := r + p
return r and k.
```

Analyzing the modular inversion of the nonce



- Developed templates based on the low-order bits of the nonces
- The template attack recovered the low-order 7 bits of each nonce reliably

Previous work attacking nonce leaks

- Most previous nonce leak attacks are based on lattice methods
  - Boneh and Venkatesan, CRYPTO ’96
  - Howgrave-Graham and Smart, 2001
  - Nguyen and Shparlinski, 2002 & 2003
  - Naccache et al., 2005
  - Liu and Nguyen, 2013
- Wanted to try a different approach: Bleichenbacher’s method
  - Introduced in 2000 at an IEEE P1363 Working Group meeting
  - Used to attack the PRNG in DSA
- Largely undocumented
  - We had to fill in many of the details of the attack
  - Many in the crypto community are unaware of the method

Mapping nonce leaks to a hidden number problem

- $s_j = K_j^{-1}(H_j + r_j x) \; mod \; q$ (second half of ECDSA signature)
- Given the low-order b bits of each K_j

$$K_j = 2^b K_{j,hi} + K_{j,lo}$$

- We can rearrange the signature equation above

$$k_j = h_j + c_j x + \alpha_j q$$

Hidden Number Problem

where

$$k_j = K_{j,hi} - \lfloor q_{b+1} \rfloor$$

Unknown but biased

$$h_j = 2^b (s_j^{-1} H_j - K_{j,lo}) - \lfloor q_{b+1} \rfloor \; mod \; q$$

Known

$$c_j = 2^{-b} s_j^{-1} r_j \; mod \; q$$

Known

- Goal of the Hidden Number Problem is to find the secret x

Bleichenbacher's attack: The bias formula

- For random variable X over $\mathbb{Z}/q\mathbb{Z}$, the bias is defined as

$$B_q(X) = E(e^{(\frac{2\pi i X}{q})})$$

- For a set of points $V = [v_0, v_1, \ldots, v_{L-1}]$ in $[0, \ldots, q-1]$ the sampled bias is defined as

$$B_q(V) = \frac{1}{L}\sum_{j=0}^{L-1} e^{\left(\frac{2\pi i v_j}{q}\right)}$$

Bleichenbacher's attack: The bias formula

- Let $0 < T \leq q$, and suppose X is uniformly distributed over $\left[-\frac{T-1}{2}, \ldots, \frac{T-1}{2}\right]$. Then:
  - $B_q(X + X') = B_q(X)B_q(X')$ for independent X and X'
  - $B_q(X) = \frac{\frac{1}{T}\sin\left(\frac{\pi T}{q}\right)}{\sin\left(\frac{\pi}{q}\right)}$. Hence $0 \leq B_q(X) \leq 1$
  - If X is uniformly distributed over $[0 \ldots q-1]$, then $B_q(X) = 0$
- Example biases for $R = \frac{T}{q} = 2^{-b}$, for large q, are shown below

b	1	2	3	4	5	6	7	8
$B_q(X)$	0.637	0.900	0.974	0.994	0.998	0.9995985	0.9998996	0.9999749

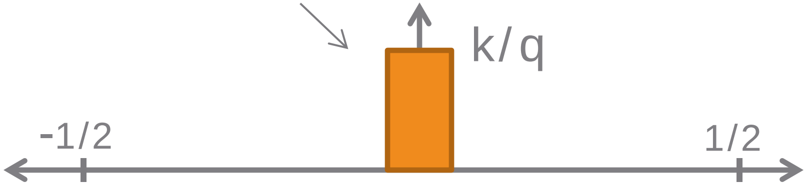
Bleichenbacher's attack: The bias formula

- For each $w \in [0, \dots, q-1]$, define the set of points $V_w = \{h_j + c_j w \bmod q\}$ for $j = [0, \dots, L-1]$

- Then the sampled bias of V_w is:

$$B_q(w) = \frac{1}{L}\sum_{j=0}^{L-1} e^{2\pi i (h_j + c_j w)/q}$$

$$= \sum_{t=0}^{q-1} \left(\frac{1}{L} \sum_{\{j|c_j=t\}} \boxed{e^{2\pi i k_j/q}} \right) e^{2\pi i t(w-x)/q}$$

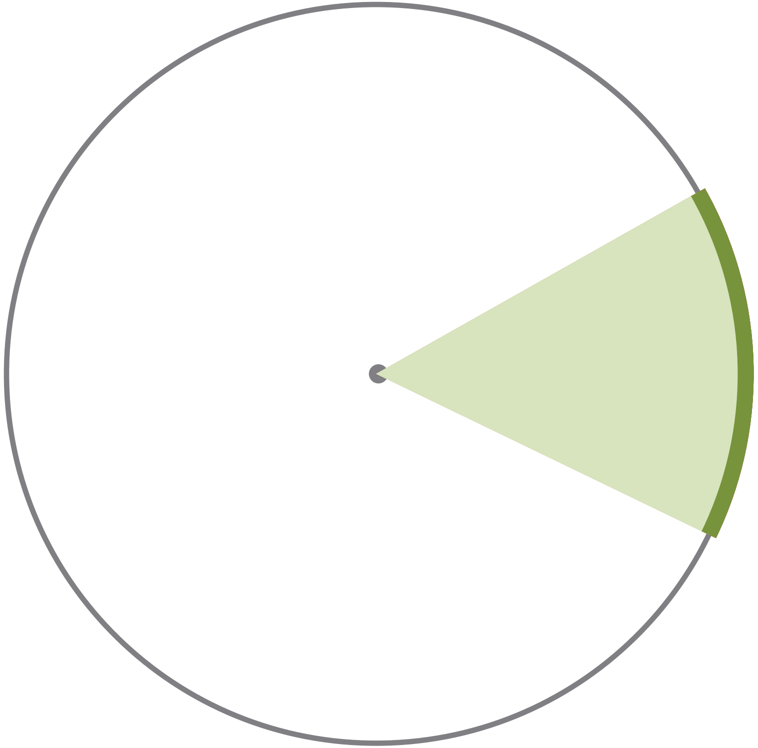


Bleichenbacher's attack: Why does it work?

$w = x$

$$t = 0$$
$$t = 1$$
$$t = 2$$
$$\cdots$$
$$t = q - 1$$

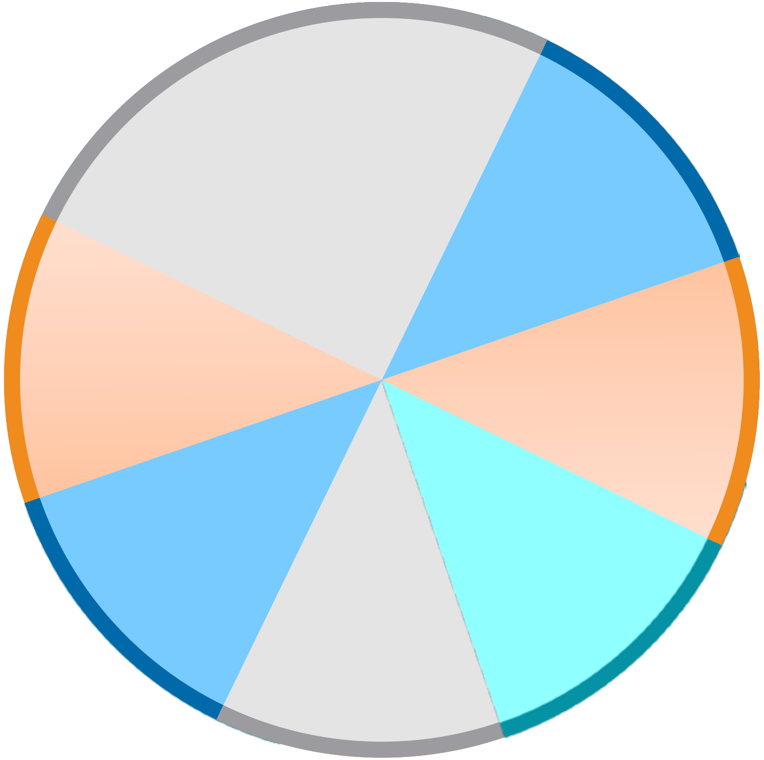
$$\boxed{\sum_{t=0}^{q-1}}\left(\frac{1}{L}\sum_{\{j|c_j=t\}}\boxed{e^{2\pi i k_j/q}}\right)e^{2\pi i t\boxed{(w-x)}/q}$$



Bleichenbacher's attack: Why does it work?

$w \neq x$

$$t = 0$$
$$t = 1$$
$$t = 2$$
$$\cdots$$
$$t = q - 1$$



$$\boxed{\sum_{t=0}^{q-1}} \left(\frac{1}{L} \sum_{\{j \mid c_j = t\}} \boxed{e^{2\pi i k_j / q}} \right) e^{2\pi i t \boxed{(w-x)}/q}$$

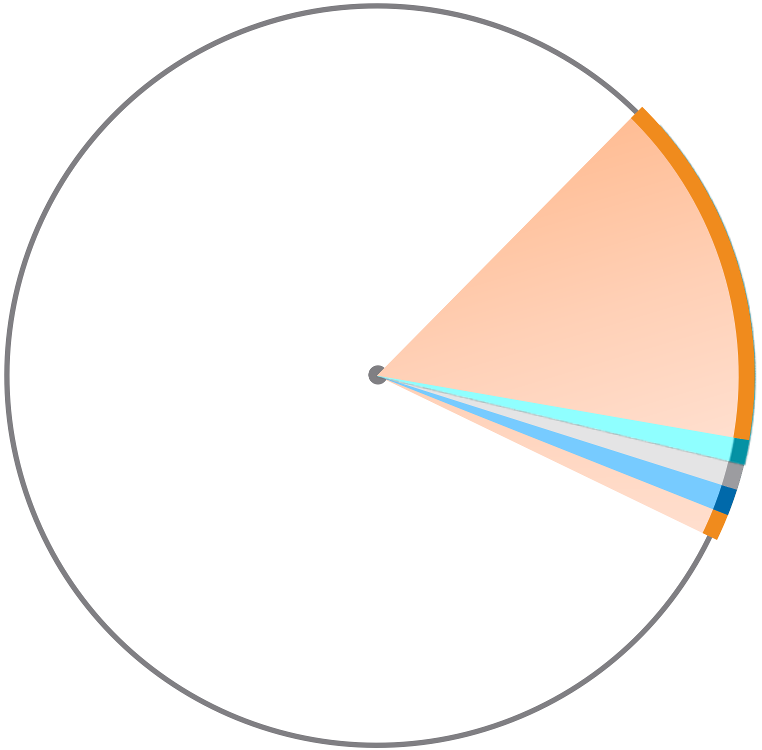
Bleichenbacher's attack: Bounding the c's

- The bias gives us a way to score putative solutions w to our hidden number problem
  - The correct x is will have bias close to one
  - All other w's will have bias close to zero
- Problem: q has 384 bits
  - Far too large to exhaust over all the w's
- Recall we are computing the sampled biases of $V_w = \{h_j + c_j w \bmod q\}$
- Bleichenbacher's insight was that if all the c's are bounded and much smaller than q, then the w's near x will also have large biases
  - This allows us to find to find approximations to x by searching over a much sparser set of w's

Bleichenbacher's attack: Bounding the c's

w close to x

$$
\begin{array}{l}
t = 0 \\
t = 1 \\
t = 2 \\
\quad \cdots \\
t = C \\
t = C + 1 \\
t = C + 2 \\
\quad \cdots \\
t = q - 1
\end{array}
$$



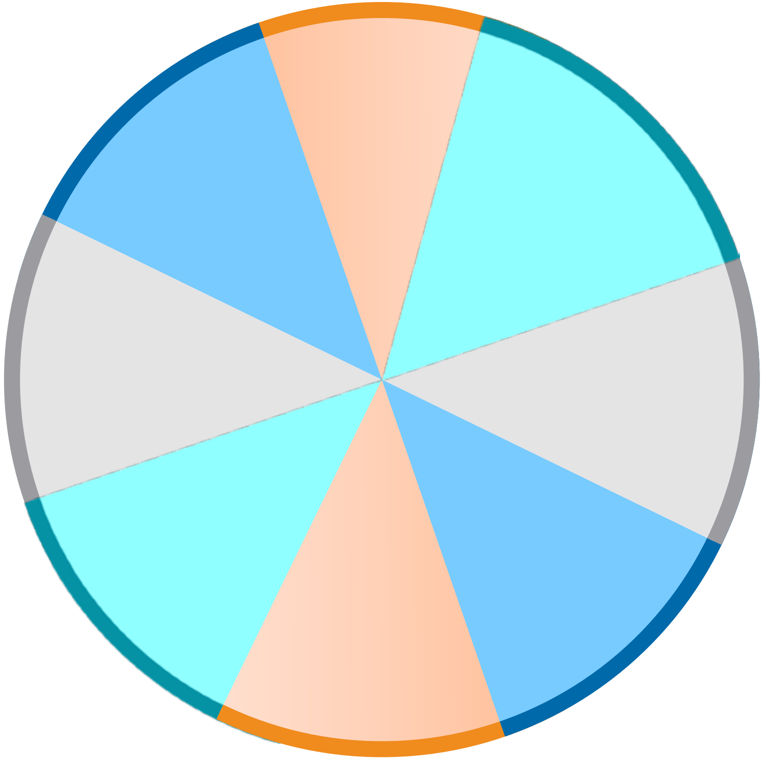
$$\boxed{\sum_{t=0}^{q-1}} \left(\frac{1}{L} \sum_{\{j \mid c_j = t\}} \boxed{e^{2\pi i k_j / q}} \right) e^{2\pi i t \boxed{(w-x)}/q}$$

Bleichenbacher's attack: Bounding the c's

w not close to x

$$t = 0$$
$$t = 1$$
$$t = 2$$
$$\cdots$$
$$t = C$$
$$t = C + 1$$
$$t = C + 2$$
$$\cdots$$
$$t = q - 1$$

$$\boxed{\sum_{t=0}^{q-1}} \left(\frac{1}{L} \sum_{\{j|c_j=t\}} \boxed{e^{2\pi i k_j/q}} \right) e^{2\pi i t \boxed{(w-x)}/q}$$



Attack algorithm for bounded c's

- Let's find an approximation to x by searching over $n = 2^N$ equally spaced w's in $[0, \ldots, q-1]$.
- Bound the c's by $C = n/2$
  - $C = nq/2^{u+1}$ if u bits of x remain unknown
- Let $w_m = mq/n$ for $m \in [0, \ldots, n-1]$
  - $w_m = 2^u m/n$ if u bits of x remain unknown
- Placing w_m in the bias equation gives

$$B_q(w_m) = \sum_{t=0}^{n-1} Z_t e^{2\pi i t m/n} \quad \text{where} \quad Z_t = \sum_{\{j|c_j=t\}} e^{2\pi i h_j/q}$$

- This is the inverse FFT of $Z = (Z_0, \ldots, Z_{n-1})$

Attack algorithm for bounded c's

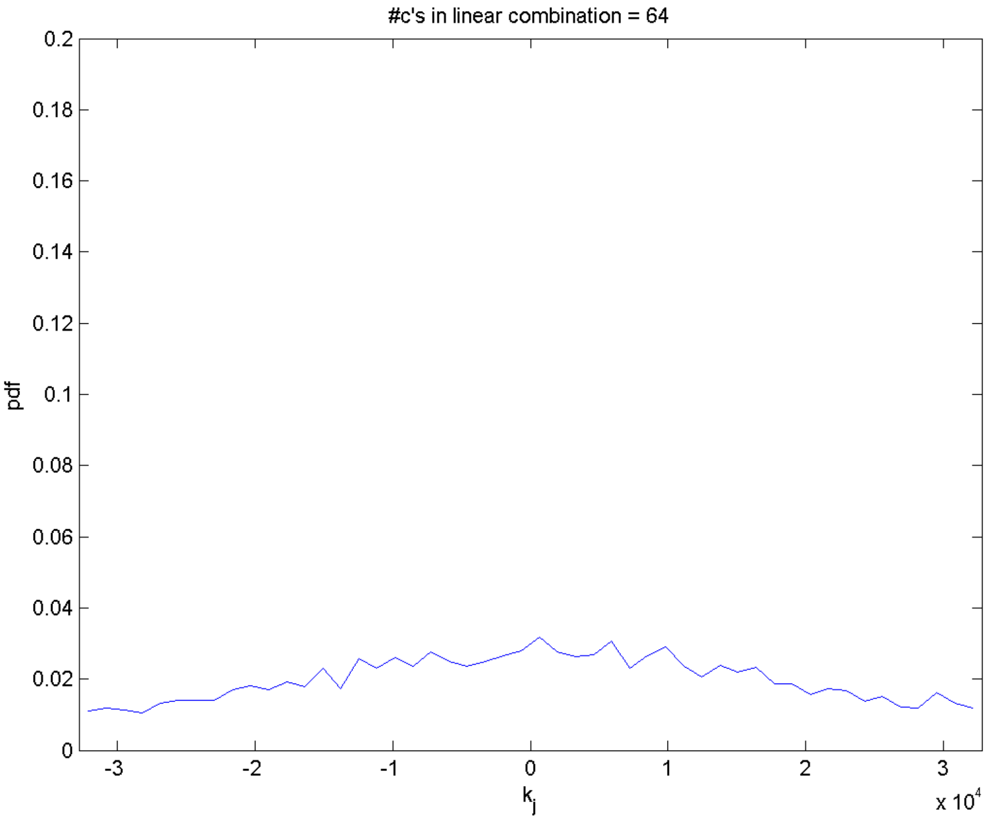
Attack is an iterative process with $n = 2^N$-point FFT.

1. Zero the vector Z and then loop over all L pairs (c_j, h_j)
   i. Add each $e^{\left(\frac{2\pi i h_j}{q}\right)}$ to the appropriate Z_t , namely $t = c_j$.
2. Compute the inverse FFT of Z and find the m for which $B_q(w_m)$ is maximal
3. The most significant N bits of x are $msb_N(x) = msb_N(\frac{mq}{n})$
   i. If u bits of x remain unknown, the next block of bits of x recovered is $msb_N(\frac{2^u m}{n})$
4. Absorb the recovered bits of x into the h_j's. Adjust the bound on the c_j's . Repeat steps 1-3 to recover the next block of bits of x.

Practical Issues: How to reduce the c's

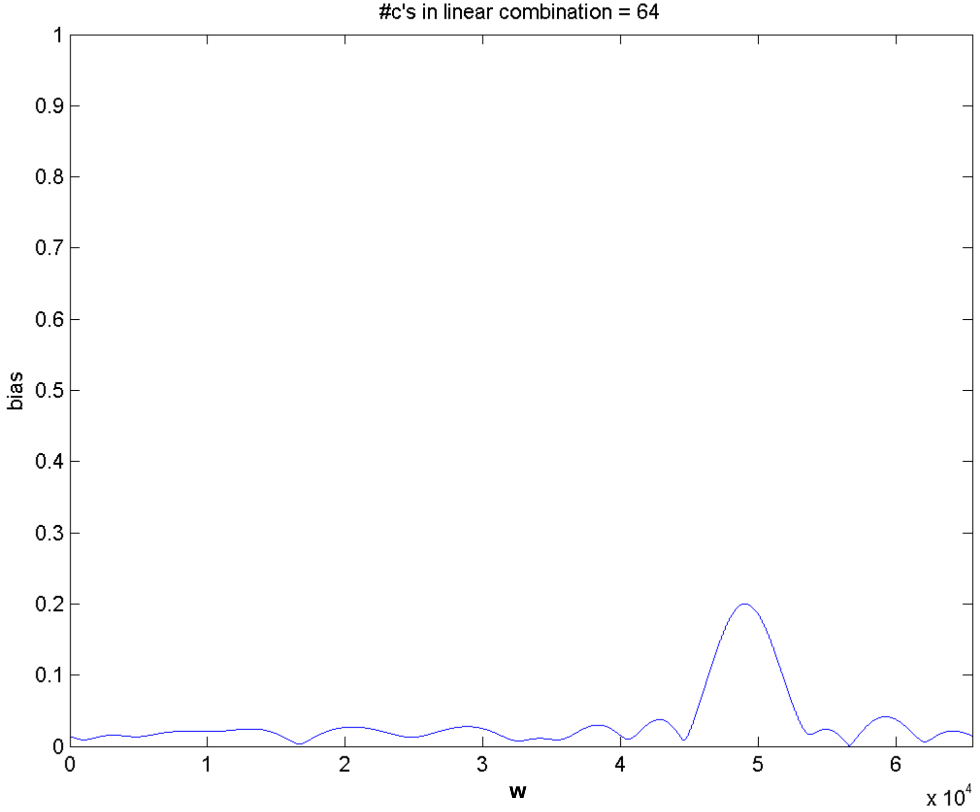
- The c's will not be nicely bounded as required for the attack
- Need to find linear combinations of the c's which are appropriately bounded
- Take corresponding linear combinations of the h's as well
- This will broaden the peak of the bias at the cost of attenuating it
  - The goal is to broaden the peak so that the required number of w_m is small enough to exhaust over, without flattening the peak so much that the bias disappears

Practical Issues: How to reduce the c's



- Effect on the pdf of the k_j as the c_j (and corresponding h_j) are linearly combined
- Number of c's in each linear combination: 64
- Number of bits in the reduced c's: 4

Practical Issues: How to reduce the c's



- Effect on the biases of the $V_w = \{h_j + c_j w \bmod q\}$ as the c_j (and corresponding h_j) are linearly combined

- Number of c's in each linear combination: 64

- Number of bits in the reduced c's: 4

Practical Issues: How to reduce the c's

- Bleichenbacher originally used millions of signatures and a clever sort and subtract algorithm to reduce the c's
  - We only had around 4000 signatures available
- We used BKZ to reduce the range of the c's
- The L_1 and L_∞ norms of the coefficients must be small enough to avoid attenuating the bias too much
- We had to find a lot of parameters heuristically
  - $\max(L_1), \max(L_\infty)$ norms
  - Good BKZ parameters for the lattices

Practical Issues: Implementation

- The attack is an iterative process
  - Use BKZ to reduce the c's
  - Compute the inverse FFT of the reduced points to get an improved approximation to x
- Discarded a few of the lower order bits of x recovered in each iteration
  - Results of the inverse FFT can be off by a few bits
  - Rounded current approximation of x for next round
- Kept a short list of top scoring candidates from each iteration

Results

- BKZ phase
  - Used BKZ to compute 3000 reduced c's from 4000 original signatures
  - BKZ dimension = 129, block size of 20, and weight of 2^{25}
  - Bound of 2^{28} for the c's during the first iteration
  - Coefficient bounds: $L_1 = 325$, $L_\infty = 8$
  - Each reduction took 2 minutes and returned on average 2 usable reduced c's
- 2^{28}-point inverse FFT phase
  - One inverse FFT took 30 seconds
- We successfully attacked a 5-bit leak, a 4-bit leak would be possible using 500,000 reduced c's with smaller L_1 and L_∞ norms
- Using SVP and CVP lattice methods we successfully attacked a 4-bit leak twice in 583 trials over a range of 100-200 points per lattice.

Further research suggestions

- Although we performed worse than standard attacks, we believe there is a lot of room for improvement
- Balance the work between BKZ and FFT phases
  - Guess some high-order bits of x and keep a list of possible candidates for a few iterations
  - This makes the first iteration (the hardest one) easier
  - With enough points the list will prune quickly
- Better range reduction of the c's is the key
  - Improved BKZ implementations such as BKZ 2.0, perhaps using the L_∞ norm as the metric instead of the usual L_2 norm
  - Other strategies?

Questions

