Cold Boot Attacks in the Discrete Logarithm Setting

B. Poettering [1] & D. L. Sibborn [2]

[1]Ruhr University of Bochum

[2]Royal Holloway, University of London

October, 2015

Outline of the talk

Cold Boot Attacks

- Usenix 2008 - Halderman et al. noted that DRAMs retain their contents for a while after power is lost.
- Bits in memory can be extracted (but it requires physical access to the machine).
  1. The attacker can insert a flash drive to the target machine,
  2. the attacker turns off the machine,
  3. the computer is restarted and the memory contents are copied to the flash drive.
- Unfortunately, the extracted bits will have errors.

Cold Boot Attacks

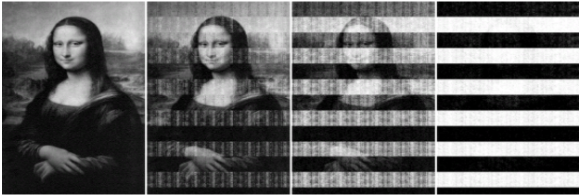
- The number of errors depends on a number of things.
- The machine: newer machines lose data quicker.
- The temperature: bits decay quicker at higher temperatures.
- The amount of time since power was lost: less time results in fewer errors.

Degradation of bits

- At room temperature, some machines erase all data within 2.5 seconds. Others require 35 seconds.
- At temperatures of $-50°\mathrm{C}$ (via the use of compressed air) all machines retained at least 99.9% of data after 60 seconds without power.
- Cooling memory chips with liquid nitrogen resulted in only 0.17% of bits degrading after 60 minutes without power.

Cold Boot Attacks

- Portions of memory are either a $1 \to 0$ region or $0 \to 1$.
- In a $1 \to 0$ region, 0 bits will always flip with very low probability (<1%), but 1 bits will flip with much higher probability.



Cold Boot Attacks

- Why is this a problem?
- Secrets will be stored in memory.
- If we can recover a noisy memory image, it might be possible to recover private keys.

Important Question

Given a noisy key obtained from a cold boot attack, how can we recover the original key?

Previous Approaches

- This question has been addressed many times before.
- Most cold boot attacks consider the reconstruction of RSA private keys.
- There are attacks against symmetric schemes such as DES and AES.
- There is only one paper that discusses cold boot attacks in the discrete logarithm setting.

Cold Boot Attacks for Discrete Logarithm Keys

- Cold boot attacks usually exploit redundancy in the private key's in-memory representation.
- E.g. in practice RSA private keys contain the parameters $(p, q, d, d_p, d_q, q_p^{-1})$ instead of just d.
- For previous DL cold boot attacks, the authors (Lee et al.) assumed there was no redundancy in the key.

Lee et al.'s Approach

- Algorithm in a nutshell:
  Let n be the length of the key, and let δ be the maximum probability that a bit flips.

  For $i = 0$ to $\lfloor n\delta \rfloor$:
  1. Assume the key has i errors,
  2. attempt to recover the key using a modified 'splitting system' algorithm,
  3. if the key is found, output it.

Problems with this approach

- It is not much better than a brute-force search.
- The algorithm assumes we know an upper bound for the number of errors.
- The algorithm is designed to work for symmetric errors (i.e., $\mathbb{P}(1 \to 0) = \mathbb{P}(0 \to 1)$), but this does not reflect the behaviour of a cold boot attack.

Improving the approach

- There are several ways we can improve key-recovery techniques in the DL setting.
- The most obvious way is to find redundant representations of keys.

Important Question

Are there any discrete logarithm implementations that contain redundant information about the private key?

Non-Adjacent Forms (NAFs)

- The simplest NAF re-encodes a scalar $x \in \{0, 1\}^{\ell}$ as a string $x' \in \{0, 1, -1\}^{\ell+1}$.
- Binary expansion: $7 = 2^2 + 2^1 + 2^0 = 111_2$.
- Alternatively $7 = 2^3 – 2^0$, so $\mathrm{NAF}(111_2) = 1\ \ 0\ \ 0\ \ -1$.
- The NAF is designed to reduce the number of additions.
- For elliptic curves, subtractions are as efficient as additions.
- The NAF is more efficient than the standard double-and-add algorithm.

NAFs

- A generalised and modified version of this NAF is used for OpenSSL elliptic curve implementations.
- The generalised NAF has *width w*. This means there is at most one non-zero digit in any string of length w (and digits are any odd number between -$2^{w-1}+1$ and $2^{w-1}-1$).
- The modified version of the NAF may alter the $\mathrm{w}+1$ most significant digits of the NAF (to increase efficiency).

In-memory representation of NAFs

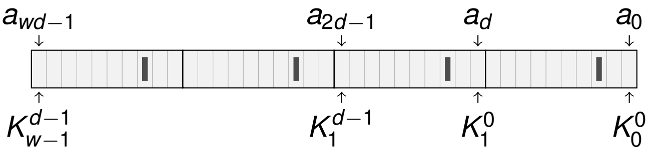
- In OpenSSL, each digit of the NAF is represented as a byte in memory.
- The digits are represented using two-complement arithmetic.
- For example,
    - $-3 \rightarrow 11111101$
    - $-1 \rightarrow 11111111$
    - $0 \rightarrow 00000000$
    - $1 \rightarrow 00000001$
    - $3 \rightarrow 00000011.$

Comb-Based Methods

- Comb methods are designed to reduce the number of multiplications.
- They require some pre-computation that depends on a fixed base point.
- Basic combs are a re-ordering of the bits.

Basic Comb

- The basic comb has parameters w and d.
- Consider a bit string, a, which has wd bits (prepend zeros, if necessary).
- The string a is rearranged into d blocks of length w, called K^i, for $i \in \{0, \dots, d-1\}$.
- Let K^i_j denote the jth bit of K^i, then $K^i_j = a_{i+jd}$.



Basic Comb

- For a point P, the value aP is computed by evaluating a sum over the K^i values.
- The basic comb is vulnerable to power analysis techniques, since $K^i = \underline{0}$ with probability 2^{-w}.
- When $K^i = \underline{0}$, the addition of this zero vector is easily identifiable.

PolarSSL Comb

- PolarSSL employs a modified comb technique.
- The modifications are designed to prevent the previous power analysis attacks.
- The output of the PolarSSL comb is $(\sigma^d, K^d, \sigma^{d-1}, K^{d-1}, \ldots, \sigma^0, K^0)$.
- The K^i are always odd in the PolarSSL comb, which prevents $K^i = \underline{0}$.
- The σ values are either 1 or -1, to denote whether K^i is positive or negative.

PolarSSL Comb

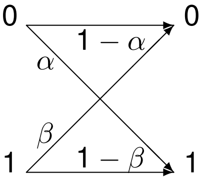
- For the PolarSSL comb, we have $w \in \{2, \ldots, 7\}$.
- Recall that each K^i has length w. Hence, each pair (σ^i, K^i) can be stored in a byte.
- For σ, $-1 \mapsto 1$ and $1 \mapsto 0$.
- The K^i values are unchanged.
- We store (σ^i, K^i) as "σ^i, padding, K^i".
- Example: $w = 3$ and $(\sigma, K) = (-1, (1, 0, 1))$. The in-memory representation as a byte is 10000101.

Attack Model

- Neither OpenSSL nor PolarSSL explicitly states that the original private key should be discarded.
- Hence, both the original key and its re-encoding (NAF or comb) will be contained in memory, at least for some time.
- We assume an adversary has mounted a cold boot attack and obtains noisy versions of the key and its re-encoding.

Attack Model

- We assume the adversary knows α and β, where bits degrade according to the following channel:



- We may estimate α and β by comparing public values with the degraded public values that were in memory.

The Reconstruction Technique

- The (textbook) NAF is constructed by starting from the least significant bits.
- i.e., for the simplest NAF, the least t signed digits only rely on knowledge of the least $t+1$ bits of the bit string.
- For example, take the integer 7:

$$
\begin{array}{rcr}
\text{partial bit string} & : & \text{partial NAF} \\
1\ \ 1 & \rightarrow & -1 \\
1\ \ 1\ \ 1 & \rightarrow & 0\ \ -1 \\
0\ \ 1\ \ 1\ \ 1 & \rightarrow & 0\ \ 0\ \ -1 \\
0\ \ 0\ \ 1\ \ 1\ \ 1 & \rightarrow & 1\ \ 0\ \ 0\ \ -1
\end{array}
$$

- Comb encodings have a similar property.

The Reconstruction Technique

- Our reconstruction procedure will consider partial solutions for the private key (across a small section of bits).
- For each candidate we can compute a partial re-encoding (NAF/comb).
- We compare these candidate solutions (and their re-encodings) against the noisy information.
- We keep a (possibly large) list of candidates for which the ‘correlation’ is ‘good’. Candidates with bad correlation are discarded.
- We then consider candidate solutions across a new section of bits, and repeat the procedure.

The Reconstruction Technique (Example for NAFs)

- Suppose we consider 2 bits at a time. We begin like this:

candidate, x	partial-NAF(x)	Correlation
0 0	0	bad
0 1	1	bad
1 0	0	bad
1 1	-1	good

- The second stage would then look like this:

candidate, x	partial-NAF(x)	Correlation
0 0 1 1	1 0 -1	bad
0 1 1 1	0 0 -1	good
1 0 1 1	1 0 -1	bad
1 1 1 1	0 0 -1	good

The Reconstruction Technique

- This process would repeat until the candidate solutions are all of equal size to the private key.
- We can then compare each remaining candidate solution against the public key $Q = aP$.
- If $xP = Q$ for any candidate x, the algorithm outputs x as the private key. Otherwise the algorithm fails.
- A similar technique applies to our comb reconstruction procedure.
- Note, our actual OpenSSL reconstruction differs slightly from the description given here (please see the paper!).

How Do We Measure Correlation?

- How is the correlation measured? However you like.
- We could use Hamming distance, Maximum-Likelihood, ...
- We could measure the correlation of all bits, or only the newly-added bits, ...
- But, we chose to use a multinomial test because it provides us with a neat theoretical analysis of success.

Multinomial Distributions

- Multinomial distributions are a generalisation of binomial distributions.
- Multinomial distributions have k mutually exclusive events.
- Each of the k events has probability $p_i > 0$, and $\sum_{i=1}^{k} p_i = 1$.

Multinomial Distributions

- Consider a bowl of sweets from which we sample at random (with replacement):

- Suppose we have four colours, with $\mathbb{P}(\text{red}) = 0.4$, $\mathbb{P}(\text{blue}) = 0.3$, $\mathbb{P}(\text{yellow}) = 0.2$, $\mathbb{P}(\text{green}) = 0.1$.
- If we pick 10 sweets randomly, what is the probability of picking:
  - 5 red, 2 blue, 2 yellow, 1 green?
  - 1 red, 6 blue, 1 yellow, 2 green?
- The multinomial distribution tells us the probability of any combination.

Multinomial Test

- Suppose we observe a set of values (say 6 red, 1 blue, 2 yellow, and 1 green).
- Suppose we believe that $(p_1, p_2, p_3, p_4) = (0.5, 0.2, 0.2, 0.1)$.
- How can we be confident that the observed values were chosen according to the probabilities p_1, p_2, p_3 and p_4?
- There are several methods, but we chose to use the multinomial test.

Multinomial Test Statistic

- Suppose we sample N items, with each item belonging to one of k distinct categories.
- Let x_i be the number of sampled items that belong to category i.
- If we hypothesise that each category has probability p_i, then we define

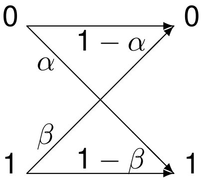
$$\mathrm{LR} = \sum_{i=0}^{k} x_i \ln\left(\frac{p_i N}{x_i}\right).$$

Multinomial Test Statistic

- Asymptotically, we have $-2\mathrm{LR} \to \chi^2_{k-1}$ whenever the observed values follow the hypothesised distribution.
- Therefore $\mathbb{P}(-2\mathrm{LR} < C) \to \mathbb{P}(\chi^2_{k-1} < C)$.
- This allows us to set an appropriate confidence interval to decide whether to reject the hypothesis.
- i.e., if we are happy to reject the correct hypothesis with probability 0.05, we set C such that $\mathbb{P}(\chi^2_{k-1} < C) = 0.95$.
- Computing C is easy.

Multinomial Test

- How does this help us?
- Recall that our algorithm measures the ‘correlation’ between our candidate key and the noisy bits.
- Recall that in a cold boot attack the bits will degrade according to the following channel:



Multinomial Test

- Hence, there are four possible bit-pairs.
- These are: $0 \to 0$, $0 \to 1$, $1 \to 0$ and $1 \to 1$.
- These four pairs can be viewed as the colours red, blue, green and yellow of the previous example.
- If we let p_b denote the probability of a b-bit appearing in the original key (together with the re-encoding), then:
  - $\mathbb{P}(0 \to 0) = p_0(1-\alpha)$,
  - $\mathbb{P}(0 \to 1) = p_0\alpha$,
  - $\mathbb{P}(1 \to 0) = p_1\beta$,
  - $\mathbb{P}(1 \to 1) = p_1(1-\beta)$.

Multinomial Test

- For each candidate solution, we perform a multinomial test.
- If the candidate's degradation is consistent with the probability vector $(p_0(1-\alpha), p_0\alpha, p_1\beta, p_1(1-\beta))$, it is kept.
- Otherwise, the algorithm discards the candidate.
- The user can specify his own confidence interval for the multinomial test.
- This allows the user to recover the private key with an arbitrary success (with a trade-off between running-time).
- N.B. This test also works in the RSA setting (and others!).

Estimating p_0 and p_1

- We have not yet addressed how to set the values of p_0 and p_1.
- One option is to estimate these values by using knowledge of the asymptotic distribution of bits of the NAF or comb.
- However, given the small sample sizes, the asymptotic estimates may not be very good or useful.
- Instead, we perform two multinomial tests: one for the 0 bits of the candidate key, and one for the 1s.

Example

- Consider the following example.

Candidate key:	01001010100010111 . . .
Noisy memory:	11100001100100001 . . .

- We parse the candidate key into 1s and 0s.

Candidate key:	000000000	11111111
Noisy memory:	110010010	10010001

- Now we test the 1s and 0s separately, which avoids the need to estimate p_0 and p_1.

Why Not Maximum-Likelihood?

- At Asiacrypt 2012, Paterson et al. showed that Maximum-Likelihood (ML) decoding is very successful and quick to recover RSA keys from a cold boot attack.
- Why, then, do we not use ML decoding?
- Firstly, the ML algorithm does not have a rigorous theoretical analysis of success, whereas the multinomial test does.
- Secondly, the ML algorithm benefits from several advantages that are inherent in the RSA recovery procedure.

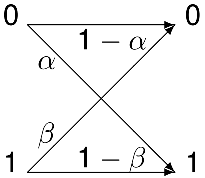
Experiments

- We will shortly see some of our experimental results.
- For each experiment we degraded 100 keys (each of length 160 bits).
- We then used our algorithm to attempt to recover the original keys.

OpenSSL (NAF) Experiments

- For these experiments we set $\alpha = 0.001$. (N.B. There are several extra parameters to the algorithm that are not displayed here.)

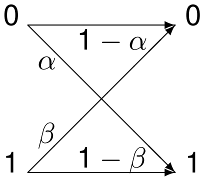
β	0.1	0.15	0.2	0.25	0.3
Predicted Success	0.15	0.15	0.02	0.01	0.01
Success	0.17	0.2	0.07	0.06	0.04



PolarSSL (comb) Experiments

- For these experiments we set $\alpha = 0.001$. (N.B. There are several extra parameters to the algorithm that are not displayed here.)

β	0.01	0.03	0.06	0.08	0.1
Predicted Success	0.73	0.17	0.04	0.01	0.01
Success	0.81	0.6	0.55	0.37	0.08



Predicted Success vs Actual Success

- There is sometimes a big discrepancy between the predicted success and the observed success!
- The predicted success is based on the chi-squared distribution.
- Recall that the distribution of the multinomial test converges to the chi-squared distribution.
- For small sample sizes, the convergence is poor.
- Due to the probabilities used in our model (i.e., $\alpha = 0.001$), the chi-squared test is providing a lower bound on the success of our algorithm.

$0 \to 1$ vs $1 \to 0$ region

- Recall that portions of memory are either $0 \to 1$ or $1 \to 0$.
- In previous cold boot attacks, the targeted private keys have an (approximately) uniform distribution of 1 bits and 0 bits.
- Hence, the key-recovery algorithms work equally well in each region.
- For the PolarSSL comb, there are slightly more 1s than 0s, but this will make a negligible difference to the algorithm.

$0 \to 1$ vs $1 \to 0$

- For the OpenSSL NAF, there are many more 0s than 1s.
- Theoretically, the success of the algorithm is independent of whether we are in a $0 \to 1$ or $1 \to 0$ region.
- However, in practice the success will be affected (because different regions will result in different rates of convergence to the chi-squared statistic).
- It will also affect the running time of the algorithm.
- In a $0 \to 1$ region, the running-time will be much longer.

Open Problems

- Bound the running-time of the algorithm.
- Bound the probability of a Type II error for the multinomial test.
    - This requires assumptions regarding the distribution of incorrect solutions.
    - In the RSA setting there is a conjecture regarding this distribution, and this would allow us to bound the running-time of the algorithm (but not the running-time of our DL algorithm).

Conclusions

- We have proposed practical key-recovery algorithms against OpenSSL and PolarSSL elliptic curve implementations.
- Our algorithms allow keys to be recovered with a user-chosen success rate (at the expense of running-time).
- The statistical test we use can be implemented with other key-recovery algorithms in other settings, such as RSA.
- Our paper provides the first exposition of the PolarSSL encoding in the cryptographic literature.