AARHUS UNIVERSITY

New Directions in Recovering Noisy RSA Keys (A Coding-Theoretic approach)

Kenneth G. Paterson (*Royal Holloway University*)
Antigoni Polychroniadou (*Aarhus University*)
Dale L. Sibborn (*Royal Holloway University*)

Outline

Outline

Motivation

- Side channel Information:
  power consumption, execution time, electromagnetic radiation, sounds, frequencies, temperatures, error messages, faulty outputs, visible light, memory images and cache memory gaps.

Motivation

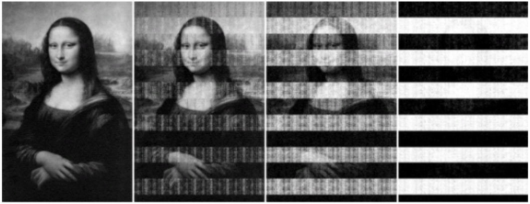
- Side channel Information:
  power consumption, execution time, electromagnetic radiation, sounds, frequencies, temperatures, error messages, faulty outputs, visible light, memory images and cache memory gaps.
- Side Channel Attacks $\leftrightarrow$ Recovering noisy secret keys.

Cold Boot Attacks

- Usenix 2008 - Halderman et al. noted that DRAMs retain their contents for a while after power is lost.

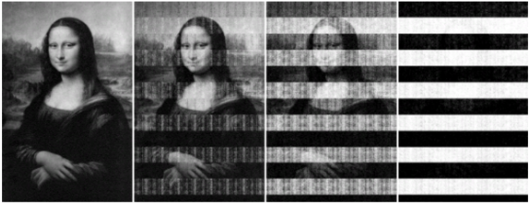
Cold Boot Attacks

- Usenix 2008 - Halderman et al. noted that DRAMs retain their contents for a while after power is lost.
- They reported longer content retention at lower temperatures. At -50°C, 99.9 % of bits were unchanged after 60 seconds.



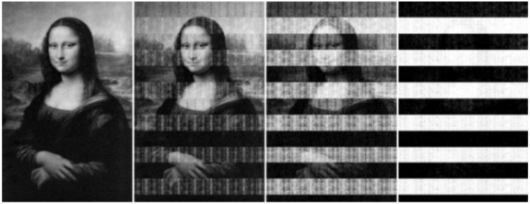
Cold Boot Attacks

- Usenix 2008 - Halderman et al. noted that DRAMs retain their contents for a while after power is lost.
- They reported longer content retention at lower temperatures. At -50°C, 99.9 % of bits were unchanged after 60 seconds.
- Bits in memory can be extracted, but they will have errors.



Cold Boot Attacks

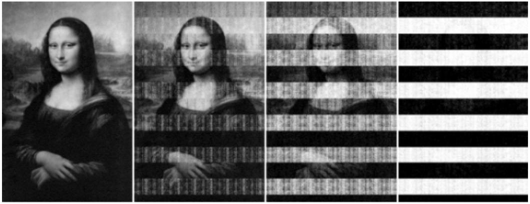
- Usenix 2008 - Halderman et al. noted that DRAMs retain their contents for a while after power is lost.
- They reported longer content retention at lower temperatures. At -50°C, 99.9 % of bits were unchanged after 60 seconds.
- Bits in memory can be extracted, but they will have errors.



- 0 bits will flip with very low probability ($< 1\%$), but 1 bits will flip with much higher probability which increases with time.

Cold Boot Attacks

- Usenix 2008 - Halderman et al. noted that DRAMs retain their contents for a while after power is lost.
- They reported longer content retention at lower temperatures. At -50°C, 99.9 % of bits were unchanged after 60 seconds.
- Bits in memory can be extracted, but they will have errors.



- 0 bits will flip with very low probability ($< 1\%$), but 1 bits will flip with much higher probability which increases with time.
- In a given region the decay is overwhelmingly either $0 \rightarrow 1$ or $1 \rightarrow 0$'

Given a noisy RSA key, is it possible to reconstruct the original key?

Rivest and Shamir (Eurocrypt 1985):
N can be factored given 2/3 of the LSBs of a prime

$1001011011010\overbrace{0001111000101101001101010}$...

Coppersmith (Eurocrypt 1996):
N can be factored given 1/2 of the MSBs of a prime

$\overbrace{10010110110100001111}000101101001101010$...

Boneh et al. (Asiacrypt 1998) :
N can be factored given 1/2 of the LSBs of a prime

$10010110110100001111\overbrace{000101101001101010}$...

Herrmann and May (Asiacrypt 2008):
N can be factored given contiguous blocks of a prime

$100\overbrace{101101}1010000111100010110\overbrace{10011}0101010$...

Given a noisy RSA key, is it possible to reconstruct the original key?

$$100\overbrace{1011}011010\overbrace{00}0111\overbrace{100}0\overbrace{101}10\overbrace{1001}10101\overbrace{010}...$$

- Heninger and Shacham (HS) Algorithm (Crypto 2009): $sk = (p, q, d, d_p, d_q)$ can be found given that some random distributed bits are known with certainty.

Given a noisy RSA key, is it possible to reconstruct the original key?

$$100\overbrace{1011}011010\overbrace{00}0111\overbrace{100}0\overbrace{101}10\overbrace{1001}10101\overbrace{010}\ldots$$

- Heninger and Shacham (HS) Algorithm (Crypto 2009): $sk = (p, q, d, d_p, d_q)$ can be found given that some random distributed bits are known with certainty.
- Henecka, May and Meurer (HMM) Algorithm (Crypto 2010): $sk = (p, q, d, d_p, d_q)$ can be found given that all the key bits are subject to errors.

Neither of the HS nor the HMM algorithm solve the motivating cold boot problem

- The HS algorithm really only applies to an idealized cold boot setting, where some bits are known for sure.

Neither of the HS nor the HMM algorithm solve the motivating cold boot problem

- The HS algorithm really only applies to an idealized cold boot setting, where some bits are known for sure.
- The HMM algorithm is designed to work for the symmetric case.

Neither of the HS nor the HMM algorithm solve the motivating cold boot problem

- The HS algorithm really only applies to an idealized cold boot setting, where some bits are known for sure.
- The HMM algorithm is designed to work for the symmetric case.
    - In the cold boot scenario, $\alpha := \Pr(0 \rightarrow 1)$ will be extremely very small, while $\beta := \Pr(1 \rightarrow 0)$ may be relatively large, and perhaps even greater than 0.5 in a very degraded case.

Contributions

- The previous algorithms do not solve their motivating cold boot problem.

Contributions

- The previous algorithms do not solve their motivating cold boot problem.
- We propose a Coding-Theoretic Approach using:

Contributions

- The previous algorithms do not solve their motivating cold boot problem.
- We propose a Coding-Theoretic Approach using:
  - channel capacity

Contributions

- The previous algorithms do not solve their motivating cold boot problem.
- We propose a Coding-Theoretic Approach using:
    - channel capacity
    - list decoding method

Contributions

- The previous algorithms do not solve their motivating cold boot problem.
- We propose a Coding-Theoretic Approach using:
  - channel capacity
  - list decoding method
  - random coding techniques

Contributions

- The previous algorithms do not solve their motivating cold boot problem.
- We propose a Coding-Theoretic Approach using:
  - channel capacity
  - list decoding method
  - random coding techniques
- We derive bounds on the performance of the previous and our new algorithm solving the cold boot problem and more...

Outline

1. Motivation
2. State of the Art
3. Our Contributions
4. Experimental Results

Coppersmith method

- Coppersmith showed how to solve a polynomial equation $f(x)$ mod N of degree k in a single variable x, as long as there is a solution smaller than $N^{1/k}$.

Coppersmith method

- Coppersmith showed how to solve a polynomial equation $f(x)$ mod N of degree k in a single variable x, as long as there is a solution smaller than $N^{1/k}$.
- The idea is to build from $f(x)$ a related polynomial $F(x)$ which still has the same solution x_0, with small coefficients.

Factoring with partial knowledge

- Let $N = pq$ and suppose we are given an approximation $\widetilde{p}$ to p.

Factoring with partial knowledge

- Let $N = pq$ and suppose we are given an approximation $\widetilde{p}$ to p.
- In other words, $p = \widetilde{p} + x_0$ where $0 \leq x_0 < X$.

Factoring with partial knowledge

- Let $N = pq$ and suppose we are given an approximation $\widetilde{p}$ to p.
- In other words, $p = \widetilde{p} + x_0$ where $0 \leq x_0 < X$.
- Coppersmith used his ideas to get an algorithm for finding p from $\widetilde{p}$.

Factoring with partial knowledge

- Let $N = pq$ and suppose we are given an approximation $\widetilde{p}$ to p.
- In other words, $p = \widetilde{p} + x_0$ where $0 \leq x_0 < X$.
- Coppersmith used his ideas to get an algorithm for finding p from $\widetilde{p}$.
- Coppersmith's original version used bivariate polynomials. We present a simpler version following work of Howgrave-Graham, Boneh, Durfee and others.

Factoring with partial knowledge

- The polynomial $f(x) = (x + \widetilde{p})$ has a small solution modulo p.

Factoring with partial knowledge

- The polynomial $f(x) = (x + \widetilde{p})$ has a small solution modulo p.
- The problem is that we don't know p, but we do know N which is a multiple of p.

Factoring with partial knowledge

- The polynomial $f(x) = (x + \widetilde{p})$ has a small solution modulo p.
- The problem is that we don't know p, but we do know N which is a multiple of p.
- The idea is to form a lattice corresponding to polynomials which have a root modulo p and to use Coppersmith method.

Example

- Let $N = 16803551$ and $\widetilde{p} = 2830$ and $X = 10$.

Example

- Let $N = 16803551$ and $\widetilde{p} = 2830$ and $X = 10$.
- Let $f(x) = (x + \widetilde{p})$.

Example

- Let $N = 16803551$ and $\widetilde{p} = 2830$ and $X = 10$.
- Let $f(x) = (x + \widetilde{p})$.
- Consider the polynomials $N, f(x), xf(x) = (x^2 + \widetilde{p}x)$ and $x^2 f(x)$.

Example

- Let $N = 16803551$ and $\widetilde{p} = 2830$ and $X = 10$.
- Let $f(x) = (x + \widetilde{p})$.
- Consider the polynomials $N, f(x), xf(x) = (x^2 + \widetilde{p}x)$ and $x^2 f(x)$.
- These all have the same small solution x_0 modulo p.

Example

- Let $N = 16803551$ and $\widetilde{p} = 2830$ and $X = 10$.
- Let $f(x) = (x + \widetilde{p})$.
- Consider the polynomials $N, f(x), xf(x) = (x^2 + \widetilde{p}x)$ and $x^2 f(x)$.
- These all have the same small solution x_0 modulo p.
- We build the lattice corresponding to these polynomials.

Example

The lattice has basis matrix:

$$\begin{pmatrix} N & 0 & 0 & 0 \\ \widetilde{p} & X & 0 & 0 \\ 0 & \widetilde{p}X & X^2 & 0 \\ 0 & 0 & \widetilde{p}X^2 & X^3 \end{pmatrix}$$

Example

- Running LLL gives the first row of the output equal to $(105, -1200, 800, 1000)$ which is of the form $(a_0, a_1X, a_2X^2, a_3X^3)$.

Example

- Running LLL gives the first row of the output equal to $(105, -1200, 800, 1000)$ which is of the form $(a_0, a_1 X, a_2 X^2, a_3 X^3)$.
- This corresponds to the polynomial $F(x) = x^3 + 8x^2 - 120x + 105$

Example

- Running LLL gives the first row of the output equal to $(105, -1200, 800, 1000)$ which is of the form $(a_0, a_1X, a_2X^2, a_3X^3)$.
- This corresponds to the polynomial $F(x) = x^3 + 8x^2 - 120x + 105$
- The polynomial has the root $x = 7$ over Z.

Example

- Running LLL gives the first row of the output equal to $(105, -1200, 800, 1000)$ which is of the form $(a_0, a_1X, a_2X^2, a_3X^3)$.
- This corresponds to the polynomial $F(x) = x^3 + 8x^2 - 120x + 105$
- The polynomial has the root $x = 7$ over Z.
- We can check that $p = \widetilde{p} + 7 = 2837$ is a factor of N.

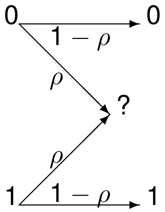
Coppersmith method

Theorem

Let $N = pq$ with $p \approx q$ and suppose we are given the high order half of the bits of p then one can factor N in polynomial time.

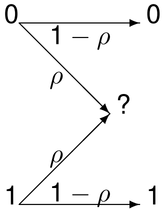
Heninger & Shacham

- Can be considered as an erasure channel.



Heninger & Shacham

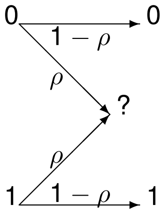
- Can be considered as an erasure channel.



- $\approx$ 73 % unknown bits can be recovered efficiently. (27 % of known bits.)

Heninger & Shacham

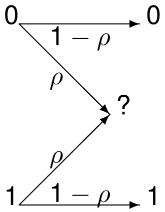
- Can be considered as an erasure channel.



- $\approx$ 73 % unknown bits can be recovered efficiently. (27 % of known bits.)
- $sk = (p, q, d, d_p, d_q)$ must satisfy certain algebraic relations.

Heninger & Shacham

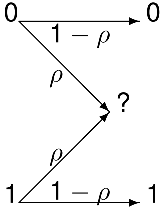
- Can be considered as an erasure channel.



- ≈ 73 % unknown bits can be recovered efficiently. (27 % of known bits.)
- $sk = (p, q, d, d_p, d_q)$ must satisfy certain algebraic relations.
  - Recovery by growing a search tree in a bit-by-bit fashion, starting with the least significant bits.

Heninger & Shacham

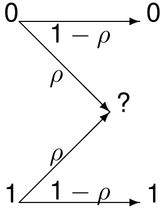
- Can be considered as an erasure channel.



- $\approx$ 73 % unknown bits can be recovered efficiently. (27 % of known bits.)
- $sk = (p, q, d, d_p, d_q)$ must satisfy certain algebraic relations.
  - Recovery by growing a search tree in a bit-by-bit fashion, starting with the least significant bits.
  - Prune the search tree removing the partial solutions which do not match with the known key bits.

Heninger & Shacham

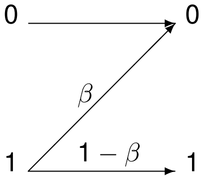
- Can be considered as an erasure channel.



- $\approx$ 73 % unknown bits can be recovered efficiently. (27 % of known bits.)
- $sk = (p, q, d, d_p, d_q)$ must satisfy certain algebraic relations.
    - Recovery by growing a search tree in a bit-by-bit fashion, starting with the least significant bits.
    - Prune the search tree removing the partial solutions which do not match with the known key bits.
- The algorithm always succeeds. However, the algorithm will blow up if only few bits are known.

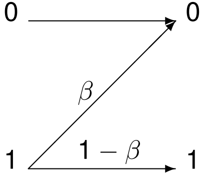
Heninger & Shacham

- Can also be considered as a Z-channel.



Heninger & Shacham

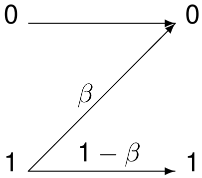
- Can also be considered as a Z-channel.



- Asymptotically the algorithm may work for $\beta < 0.46$

Heninger & Shacham

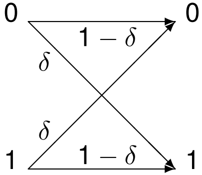
- Can also be considered as a Z-channel.



- Asymptotically the algorithm may work for $\beta < 0.46$
- Fails if there is a $0 \to 1$ flip.

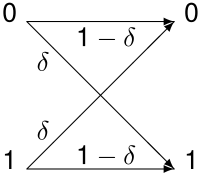
Henecka, May & Meurer

- Can be viewed as a Binary Symmetric Channel



Henecka, May & Meurer

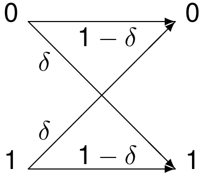
- Can be viewed as a Binary Symmetric Channel



- Algorithm theoretically is efficient for $\delta < 0.237$.

Henecka, May & Meurer

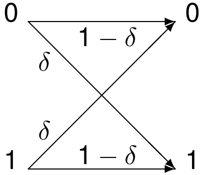
- Can be viewed as a Binary Symmetric Channel



- Algorithm theoretically is efficient for $\delta < 0.237$.
- Consider t bit-slices at a time of possible solutions to the algebraic relations.

Henecka, May & Meurer

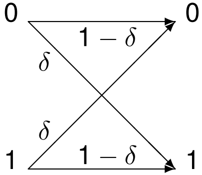
- Can be viewed as a Binary Symmetric Channel



- Algorithm theoretically is efficient for $\delta < 0.237$.
- Consider t bit-slices at a time of possible solutions to the algebraic relations.
- For the private key $sk = (p, q, d, d_p, d_q)$

Henecka, May & Meurer

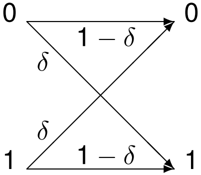
- Can be viewed as a Binary Symmetric Channel



- Algorithm theoretically is efficient for $\delta < 0.237$.
- Consider *t* bit-slices at a time of possible solutions to the algebraic relations.
- For the private key $sk = (p, q, d, d_p, d_q)$
  1. generate 2^t candidate solutions on $5t$ new private key bits for each candidate at each stage

Henecka, May & Meurer

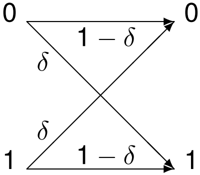
- Can be viewed as a Binary Symmetric Channel



- Algorithm theoretically is efficient for $\delta < 0.237$.
- Consider t bit-slices at a time of possible solutions to the algebraic relations.
- For the private key $sk = (p, q, d, d_p, d_q)$
  1. generate 2^t candidate solutions on $5t$ new private key bits for each candidate at each stage
  2. Compute the Hamming distance between the candidate solutions and the noisy key, keeping all candidates for which this metric is less than some carefully chosen threshold C

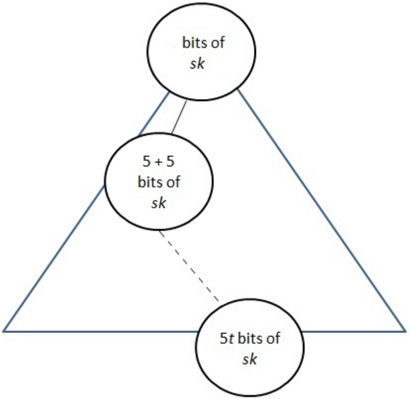
Henecka, May & Meurer

- Can be viewed as a Binary Symmetric Channel



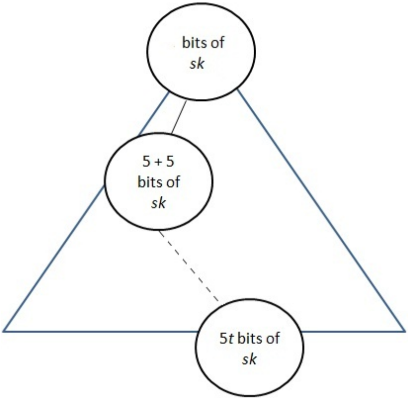
- Algorithm theoretically is efficient for $\delta < 0.237$.
- Consider t bit-slices at a time of possible solutions to the algebraic relations.
- For the private key $sk = (p, q, d, d_p, d_q)$
  1. generate 2^t candidate solutions on $5t$ new private key bits for each candidate at each stage
  2. Compute the Hamming distance between the candidate solutions and the noisy key, keeping all candidates for which this metric is less than some carefully chosen threshold C
- The algorithm will fail if the correct solution is rejected. In addition, if C is set too loosely then there is a large number of candidate solutions.

Structure of a subtree



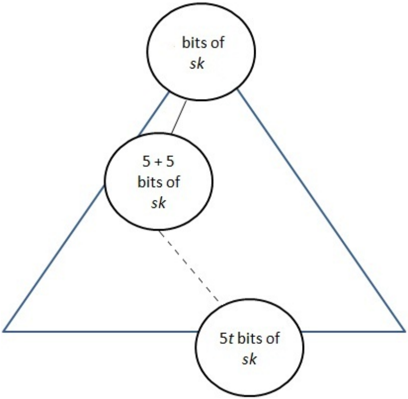
- Subtrees of depth t and prune all leaves whose Hamming distance to $sk = (p, q, d, d_p, d_q)$ is greater than C.

Structure of a subtree



- Subtrees of depth t and prune all leaves whose Hamming distance to $sk = (p, q, d, d_p, d_q)$ is greater than C.
- Each leaf contains $5t$ fresh bits of $sk = (p, q, d, d_p, d_q)$.

Structure of a subtree



- Subtrees of depth t and prune all leaves whose Hamming distance to $sk = (p, q, d, d_p, d_q)$ is greater than C.
- Each leaf contains $5t$ fresh bits of $sk = (p, q, d, d_p, d_q)$.
- Iterate for $n/(2t)$ rounds. All leaves in the last subtrees contain all $n/2$ bits of p, q, d, d_p and d_q.

Questions ?

- 0.73 (HS)?

Questions ?

- 0.73 (HS)?
- 0.237 (HMM)?

Questions ?

- 0.73 (HS)?
- 0.237 (HMM)?
- “Magic constants”?

Questions ?

- 0.73 (HS)?
- 0.237 (HMM)?
- “Magic constants”?
- Are these bounds the best possible?

Questions ?

- 0.73 (HS)?
- 0.237 (HMM)?
- “Magic constants”?
- Are these bounds the best possible?
- Is there any ultimate limit to the noise level?

Questions ?

- 0.73 (HS)?
- 0.237 (HMM)?
- “Magic constants”?
- Are these bounds the best possible?
- Is there any ultimate limit to the noise level?
- Is there any algorithm that solves the true cold boot problem?

Questions ?

- 0.73 (HS)?
- 0.237 (HMM)?
- “Magic constants”?
- Are these bounds the best possible?
- Is there any ultimate limit to the noise level?
- Is there any algorithm that solves the true cold boot problem?
- Is there any general algorithm that works in other types of side channel attack?

Questions ?

- 0.73 (HS)?
- 0.237 (HMM)?
- “Magic constants”?
- Are these bounds the best possible?
- Is there any ultimate limit to the noise level?
- Is there any algorithm that solves the true cold boot problem?
- Is there any general algorithm that works in other types of side channel attack?

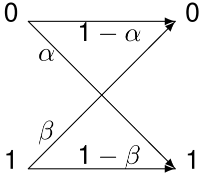
- We show how to recast the problem of noisy RSA key recovery as a problem in coding theory.

Outline

1. Motivation
2. State of the Art
3. Our Contributions
4. Experimental Results

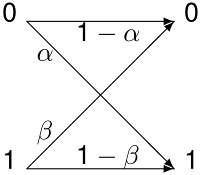
Our Contributions

- We consider the general non-symmetric channel



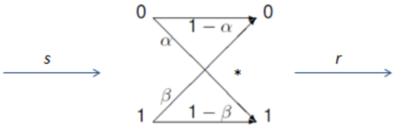
Our Contributions

- We consider the general non-symmetric channel



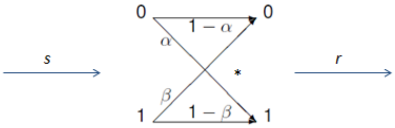
- This will allow us to model the real cold-boot scenario as well as those of Heninger & Shacham and Henecka, May & Meurer.

Channel model



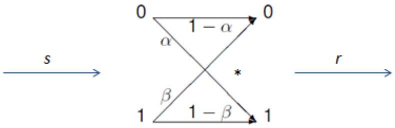
- Code $\mathcal{C}$:
  The set of 2^t candidates, with one codeword s being selected and transmitted over a noisy channel, resulting in a received word r).

Channel model



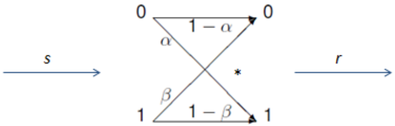
- Code $\mathcal{C}$:
  The set of 2^t candidates, with one codeword *s* being selected and transmitted over a noisy channel, resulting in a received word *r*).
- This code has rate $R \geq 1/m$, ($m = 2, 3, 5$) and lenght mt.

Channel model



- Code $\mathcal{C}$:
  The set of 2^t candidates, with one codeword s being selected and transmitted over a noisy channel, resulting in a received word r).
- This code has rate $R \geq 1/m$, ($m = 2, 3, 5$) and lenght mt.
- Decode r maximizing $\Pr(r|s)$.

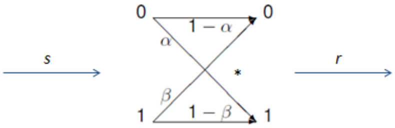
Channel model



- Code $\mathcal{C}$:
  The set of 2^t candidates, with one codeword *s* being selected and transmitted over a noisy channel, resulting in a received word *r*).
- This code has rate $R \geq 1/m$, $(m = 2, 3, 5)$ and lenght *mt*.
- Decode *r* maximizing $\Pr(r|s)$.

- * The noisy channel can also be :

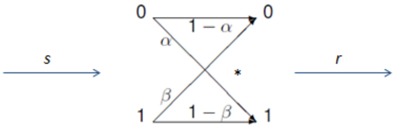
Channel model



- Code $\mathcal{C}$:
  The set of 2^t candidates, with one codeword *s* being selected and transmitted over a noisy channel, resulting in a received word *r*).
- This code has rate $R \geq 1/m$, ($m = 2, 3, 5$) and lenght mt.
- Decode *r* maximizing $\Pr(r|s)$.

- * The noisy channel can also be :
- A binary erasure channel (HS: $t = 1$).

Channel model



- Code $\mathcal{C}$:
  The set of 2^t candidates, with one codeword s being selected and transmitted over a noisy channel, resulting in a received word r).
- This code has rate $R \geq 1/m$, ($m = 2, 3, 5$) and lenght mt.
- Decode r maximizing $\Pr(r|s)$.

- * The noisy channel can also be :
- A binary erasure channel (HS: $t = 1$).
- A binary symmetric channel (HMM).

Our Coding-Theoretic viewpoint

- Derivation of upper bounds on possible error rates for all former algorithms (based on Shannon's noisy-channel coding theorem).

Our Coding-Theoretic viewpoint

- Derivation of upper bounds on possible error rates for all former algorithms (based on Shannon's noisy-channel coding theorem).
- We derive a key recovery algorithm that works for any (memoryless) binary channel.

Our Coding-Theoretic viewpoint

- Derivation of upper bounds on possible error rates for all former algorithms (based on Shannon's noisy-channel coding theorem).
- We derive a key recovery algorithm that works for any (memoryless) binary channel.
- We modify the HMM algorithm to use a likelihood statistic in place of the Hamming metric when selecting from the candidate codewords.

Our Coding-Theoretic viewpoint

- Derivation of upper bounds on possible error rates for all former algorithms (based on Shannon's noisy-channel coding theorem).
- We derive a key recovery algorithm that works for any (memoryless) binary channel.
- We modify the HMM algorithm to use a likelihood statistic in place of the Hamming metric when selecting from the candidate codewords.
- We keep the L codewords having the highest values of this likelihood statistic and reject the others(our algorithm uses maximum likelihood list decoding).

Our Coding-Theoretic viewpoint

- Derivation of upper bounds on possible error rates for all former algorithms (based on Shannon's noisy-channel coding theorem).
- We derive a key recovery algorithm that works for any (memoryless) binary channel.
- We modify the HMM algorithm to use a likelihood statistic in place of the Hamming metric when selecting from the candidate codewords.
- We keep the L codewords having the highest values of this likelihood statistic and reject the others(our algorithm uses maximum likelihood list decoding).
- We give an analysis of the success probability of our new algorithm based on the non-random nature of our code.

Our Coding-Theoretic viewpoint

- Derivation of upper bounds on possible error rates for all former algorithms (based on Shannon's noisy-channel coding theorem).
- We derive a key recovery algorithm that works for any (memoryless) binary channel.
- We modify the HMM algorithm to use a likelihood statistic in place of the Hamming metric when selecting from the candidate codewords.
- We keep the *L* codewords having the highest values of this likelihood statistic and reject the others(our algorithm uses maximum likelihood list decoding).
- We give an analysis of the success probability of our new algorithm based on the non-random nature of our code.
- Validation of our theoretic analysis through extensive experimental results.

Our Coding-Theoretic viewpoint

- Derivation of upper bounds on possible error rates for all former algorithms (based on Shannon's noisy-channel coding theorem).
- We derive a key recovery algorithm that works for any (memoryless) binary channel.
- We modify the HMM algorithm to use a likelihood statistic in place of the Hamming metric when selecting from the candidate codewords.
- We keep the L codewords having the highest values of this likelihood statistic and reject the others(our algorithm uses maximum likelihood list decoding).
- We give an analysis of the success probability of our new algorithm based on the non-random nature of our code.
- Validation of our theoretic analysis through extensive experimental results.
- The generation of our code is based on the Hensel lifting.

Initial Steps (à la HS)

- PKCS # 1 RSA key: $(N, e, p, q, d, d_p, d_q, q_p^{-1})$

Initial Steps (à la HS)

- PKCS # 1 RSA key: $(N, e, p, q, d, d_p, d_q, q_p^{-1})$
- $d_p = d \mod p - 1$ and $q_p^{-1} = q^{-1} \mod p$

Initial Steps (à la HS)

- PKCS # 1 RSA key: $(N, e, p, q, d, d_p, d_q, q_p^{-1})$
- $d_p = d \mod p - 1$ and $q_p^{-1} = q^{-1} \mod p$
- Make RSA congruences explicit:

$$\begin{aligned} N &= pq \\ ed &= k\phi(N) + 1 \\ ed_p &= k_p(p-1) + 1 \\ ed_q &= k_q(q-1) + 1 \end{aligned}$$

for some constants k, k_p and k_q.

Initial Steps (à la HS)

- k, k_p and k_q obtained via a simple algorithm (Restriction to small e).

Initial Steps (à la HS)

- k, k_p and k_q obtained via a simple algorithm (Restriction to small e).
- $$k := \left\lfloor \frac{e\tilde{d} - 1}{N + 1} \right\rfloor$$

  (trick from [Boneh, Durfee, Frankel 98])

Initial Steps (à la HS)

- k, k_p and k_q obtained via a simple algorithm (Restriction to small e).
- $$k := \left\lfloor \frac{e\tilde{d} - 1}{N + 1} \right\rfloor$$
  (trick from [Boneh, Durfee, Frankel 98])
- If e is prime we can find k_p, k_q:

$$k_p^2 - (k(N-1)+1)k_p - k \equiv 0 \mod e$$

Initial Steps (à la HS)

Define $\tau(x) := \max\{i \in \mathbb{N} : 2^i \mid x\}$ such as $2^{\tau(k_p)+1} | k_p(p-1)$, $2^{\tau(k_q)+1} | k_q(q-1)$ and $2^{\tau(k)+2} | k\phi(N)$. Then:

$$\begin{aligned} d_p &\equiv e^{-1} \mod 2^{\tau(k_p)+1} \\ d_q &\equiv e^{-1} \mod 2^{\tau(k_q)+1} \\ d &\equiv e^{-1} \mod 2^{\tau(k)+2}. \end{aligned}$$

This allows us to correct the least significant bits of d, d_p and d_q. Furthermore we can calculate *slice*(0), where we define

$$\mathit{slice}(i) := (p[i], q[i], d[i + \tau(k)], d_p[i + \tau(k_p)], d_q[i + \tau(k_q)]).$$

with $x[i]$ denoting the i-th bit of the string x.

Initial Steps (à la HS)

- Obtaining a solution (p', q', d', d'_p, d'_q) from *slice*(0) to *slice*$(i-1)$ then the bits in *slice*(i) (p, q, d, d_p, d_q) are related as follows:

$$
\begin{aligned}
p[i] + q[i] &= c1 \mod 2 \\
d[i + \tau(k)] + p[i] + q[i] &= c2 \mod 2 \\
d_p[i + \tau(k_p)] + p[i] &= c3 \mod 2 \\
d_q[i + \tau(k_q)] + q[i] &= c4 \mod 2.
\end{aligned}
$$

Because we have 4 constraints on 5 unknowns, there are exactly 2 possible solutions for *slice*(i), rather than 32.

Initial Steps (à la HS)

- Obtaining a solution (p', q', d', d'_p, d'_q) from *slice*(0) to *slice*$(i-1)$ then the bits in *slice*(i) (p, q, d, d_p, d_q) are related as follows:

$$\begin{aligned} p[i] + q[i] &= c1 \mod 2 \\ d[i+\tau(k)] + p[i] + q[i] &= c2 \mod 2 \\ d_p[i+\tau(k_p)] + p[i] &= c3 \mod 2 \\ d_q[i+\tau(k_q)] + q[i] &= c4 \mod 2. \end{aligned}$$

  Because we have 4 constraints on 5 unknowns, there are exactly 2 possible solutions for *slice*(i), rather than 32.

- Multivariate Hensel's Lemma gives values for the c_i.

Initial Steps (à la HS)

- Obtaining a solution (p', q', d', d'_p, d'_q) from *slice*(0) to *slice*$(i-1)$ then the bits in *slice*(i) (p, q, d, d_p, d_q) are related as follows:

$$\begin{aligned} p[i] + q[i] &= c1 \mod 2 \\ d[i + \tau(k)] + p[i] + q[i] &= c2 \mod 2 \\ d_p[i + \tau(k_p)] + p[i] &= c3 \mod 2 \\ d_q[i + \tau(k_q)] + q[i] &= c4 \mod 2. \end{aligned}$$

  Because we have 4 constraints on 5 unknowns, there are exactly 2 possible solutions for *slice*(i), rather than 32.
- Multivariate Hensel's Lemma gives values for the c_i.
- Previous bits give us constraints on future bits.

Initial Steps (à la HS)

- Obtaining a solution (p', q', d', d'_p, d'_q) from *slice*(0) to *slice*$(i-1)$ then the bits in *slice*(i) (p, q, d, d_p, d_q) are related as follows:

$$\begin{aligned} p[i] + q[i] &= c1 \mod 2 \\ d[i+\tau(k)] + p[i] + q[i] &= c2 \mod 2 \\ d_p[i+\tau(k_p)] + p[i] &= c3 \mod 2 \\ d_q[i+\tau(k_q)] + q[i] &= c4 \mod 2. \end{aligned}$$

  Because we have 4 constraints on 5 unknowns, there are exactly 2 possible solutions for *slice*(i), rather than 32.
- Multivariate Hensel's Lemma gives values for the c_i.
- Previous bits give us constraints on future bits.
- We perform t Hensel lifts to generate 2^t candidate partial solutions.
  Filter these according to some criterion.
  Repeat on remaining candidates.

Maximum Likelihood Approach to Filtering

- Let $M2^t$ be the candidate solutions on mt bits arising at some stage in the algorithm $s_1, \ldots, s_{M2^t}$

Maximum Likelihood Approach to Filtering

- Let $M2^t$ be the candidate solutions on mt bits arising at some stage in the algorithm $s_1, \ldots, s_{M2^t}$
- We wish to find:

$$\arg \max_{1 \le i \le M2^t} \Pr(s_i | r).$$

where r is the noisy RSA key.

Maximum Likelihood Approach to Filtering

- Let $M2^t$ be the candidate solutions on mt bits arising at some stage in the algorithm $s_1, \ldots, s_{M2^t}$
- We wish to find:

$$\arg \max_{1 \leq i \leq M2^t} \Pr(s_i | r).$$

  where r is the noisy RSA key.
- Using Bayes' theorem, this is equivalent to finding

$$\arg \max_{1 \leq i \leq M2^t} \Pr(r | s_i).$$

Maximum Likelihood Approach to Filtering

- Let $M2^t$ be the candidate solutions on mt bits arising at some stage in the algorithm $s_1, \ldots, s_{M2^t}$
- We wish to find:

$$\arg \max_{1 \leq i \leq M2^t} \Pr(s_i | r).$$

  where r is the noisy RSA key.
- Using Bayes' theorem, this is equivalent to finding

$$\arg \max_{1 \leq i \leq M2^t} \Pr(r | s_i).$$

- This can be calculated as

$$\arg \max_{1 \leq i \leq M2^t} \left((1-\alpha)^{n^i_{00}} \alpha^{n^i_{01}} (1-\beta)^{n^i_{11}} \beta^{n^i_{10}} \right)$$

Our Algorithm

Algorithm 1: Pseudo-code of our Algorithm

Data: (N, e), $\widetilde{sk} = (\tilde{p}, \tilde{q}, \tilde{d}, \tilde{d}_p, \tilde{d}_q)$ α, β.

Initialization Phase:
Find (k, k_p, k_q) given (N, e);
Find *slice*(0) given (e, k, k_p, k_q);
Create a *list* and add *slice*(0);
Lifting phase:
for stage $= 1$ ***to*** $n/(2t)$ **do**
    for i $= 1$ ***to*** L **do**
        Replace each partial solution i from the *list* with a set of 2^t candidate solutions s_i obtained by Hensel lifting;
        Pruning Phase:
        Calculate the log-likelihood $\log \Pr(r|s_i)$ for each entry s_i on list;
        Add the L entries in list having the highest log-likelihoods and delete the remainder;

Finalization Phase: Find one candidate that satisfies all the RSA equations;
Output : *sk*

Our algorithm does not quite implement ML decoding at each stage.
Our algorithm has *deterministic* polynomial running time $O(L2^t n/2t)$ and *deterministic* memory consumption $O(L2^t)$ or $(O(t + L))$.
The running time in all our experiments was $O(2^t)$ per stage rather than $O(L2^t)$ because of multi-threading.

Asymptotic Analysis of Our Algorithm

Strong Randomness Assumption

The $L2^t$ candidates s_i generated at each stage of our Algorithm are independent and uniformly random mt-bit vectors.

Asymptotic Analysis of Our Algorithm

Strong Randomness Assumption

The $L2^t$ candidates s_i generated at each stage of our Algorithm are independent and uniformly random mt-bit vectors.

- Shannon's noisy-channel coding theorem states that, as $mt \to \infty$, the use of random codes in combination with Maximum Likelihood (ML) decoding achieves arbitrarily small decoding error probability, provided that the code rate stays below the capacity of the channel.

Asymptotic Analysis of Our Algorithm

Strong Randomness Assumption

The $L2^t$ candidates s_i generated at each stage of our Algorithm are independent and uniformly random mt-bit vectors.

- Shannon's noisy-channel coding theorem states that, as $mt \to \infty$, the use of random codes in combination with Maximum Likelihood (ML) decoding achieves arbitrarily small decoding error probability, provided that the code rate stays below the capacity of the channel.
- For fixed L and m, for our code, this holds provided $1/m$ is strictly less than the capacity as $t \to \infty$.

Asymptotic Analysis of Our Algorithm

Strong Randomness Assumption

The $L2^t$ candidates s_i generated at each stage of our Algorithm are independent and uniformly random mt-bit vectors.

- Shannon's noisy-channel coding theorem states that, as $mt \to \infty$, the use of random codes in combination with Maximum Likelihood (ML) decoding achieves arbitrarily small decoding error probability, provided that the code rate stays below the capacity of the channel.
- For fixed L and m, for our code, this holds provided $1/m$ is strictly less than the capacity as $t \to \infty$.
- Apply to the maximum likelihood list decoding rule.

Asymptotic Analysis of Our Algorithm

Strong Randomness Assumption

The $L2^t$ candidates s_i generated at each stage of our Algorithm are independent and uniformly random mt-bit vectors.

- Shannon's noisy-channel coding theorem states that, as $mt \to \infty$, the use of random codes in combination with Maximum Likelihood (ML) decoding achieves arbitrarily small decoding error probability, provided that the code rate stays below the capacity of the channel.
- For fixed L and m, for our code, this holds provided $1/m$ is strictly less than the capacity as $t \to \infty$.
- Apply to the maximum likelihood list decoding rule.
- Our strong randomness assumption is not true for our code due to the Hensel lifting.

Asymptotic Analysis of Our Algorithm

- Now we give a rigorous analysis of our algorithm under reasonable assumptions in the symmetric case (where $\alpha = \beta$).

Asymptotic Analysis of Our Algorithm

- Now we give a rigorous analysis of our algorithm under reasonable assumptions in the symmetric case (where $\alpha = \beta$).
- We prove that we can achieve reliable decoding when the rate is close to the capacity bound $1 - H_2(\delta)$ imposed by the binary symmetric channel. Open problem for the non-symmetric case.

Asymptotic Analysis of Our Algorithm

- Now we give a rigorous analysis of our algorithm under reasonable assumptions in the symmetric case (where $\alpha = \beta$).
- We prove that we can achieve reliable decoding when the rate is close to the capacity bound $1 - H_2(\delta)$ imposed by the binary symmetric channel. Open problem for the non-symmetric case.

Weak Randomness Assumptions

Asymptotic Analysis of Our Algorithm

- Now we give a rigorous analysis of our algorithm under reasonable assumptions in the symmetric case (where $\alpha = \beta$).
- We prove that we can achieve reliable decoding when the rate is close to the capacity bound $1 - H_2(\delta)$ imposed by the binary symmetric channel. Open problem for the non-symmetric case.

Weak Randomness Assumptions

- The bits of all candidate solutions are uniformly distributed over $\{0, 1\}$.

Asymptotic Analysis of Our Algorithm

- Now we give a rigorous analysis of our algorithm under reasonable assumptions in the symmetric case (where $\alpha = \beta$).
- We prove that we can achieve reliable decoding when the rate is close to the capacity bound $1 - H_2(\delta)$ imposed by the binary symmetric channel. Open problem for the non-symmetric case.

Weak Randomness Assumptions

- The bits of all candidate solutions are uniformly distributed over $\{0, 1\}$.
- Leaves in the same tree are independent on the last $m(t - \ell - k)$ bits, provided the leaves have no common ancestor at depth greater than ℓ.

Asymptotic Analysis of Our Algorithm

- Now we give a rigorous analysis of our algorithm under reasonable assumptions in the symmetric case (where $\alpha = \beta$).
- We prove that we can achieve reliable decoding when the rate is close to the capacity bound $1 - H_2(\delta)$ imposed by the binary symmetric channel. Open problem for the non-symmetric case.

Weak Randomness Assumptions

- The bits of all candidate solutions are uniformly distributed over $\{0, 1\}$.
- Leaves in the same tree are independent on the last $m(t - \ell - k)$ bits, provided the leaves have no common ancestor at depth greater than ℓ.
- The bits at leaves in distinct trees are independent of each other across all mt bits.

Asymptotic Analysis of Our Algorithm

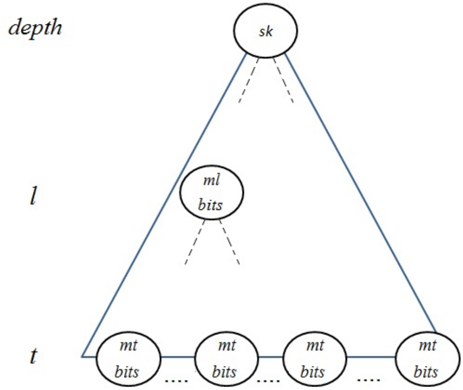
- Now we give a rigorous analysis of our algorithm under reasonable assumptions in the symmetric case (where $\alpha = \beta$).
- We prove that we can achieve reliable decoding when the rate is close to the capacity bound $1 - H_2(\delta)$ imposed by the binary symmetric channel. Open problem for the non-symmetric case.

Weak Randomness Assumptions

- The bits of all candidate solutions are uniformly distributed over $\{0, 1\}$.
- Leaves in the same tree are independent on the last $m(t - \ell - k)$ bits, provided the leaves have no common ancestor at depth greater than ℓ.
- The bits at leaves in distinct trees are independent of each other across all *mt* bits.

- The closer together in a tree two leaves are, the more correlated their bits are.

Asymptotic Analysis of Our Algorithm



The Binary Symmetric Channel

- When sk is of the form (p, q, d, d_p, d_q), the code rate is at least $1/5$ (we have 2^t codewords and length $5t$).

sk	R	δ	HMM
(p, q, d, d_p, d_q)	$1/5$	0.243	0.237
(p, q, d)	$1/3$	0.174	0.16
(p, q)	$1/2$	0.110	0.08

The Binary Symmetric Channel

- When *sk* is of the form (p, q, d, d_p, d_q), the code rate is at least $1/5$ (we have 2^t codewords and length $5t$).
- The capacity is: $C_{\mathrm{BSC}}(\delta) = 1 - H_2(\delta)$

sk	R	δ	HMM
(p, q, d, d_p, d_q)	$1/5$	0.243	0.237
(p, q, d)	$1/3$	0.174	0.16
(p, q)	$1/2$	0.110	0.08

The Binary Symmetric Channel

- When *sk* is of the form (p, q, d, d_p, d_q), the code rate is at least $1/5$ (we have 2^t codewords and length $5t$).
- The capacity is: $C_{\text{BSC}}(\delta) = 1 - H_2(\delta)$
- Applying Shannon's theorem, an algorithm that outputs a single codeword cannot reliably decode when $1 - H_2(\delta) \leq 0.2$

sk	R	δ	HMM
(p, q, d, d_p, d_q)	$1/5$	0.243	0.237
(p, q, d)	$1/3$	0.174	0.16
(p, q)	$1/2$	0.110	0.08

The Binary Symmetric Channel

- When *sk* is of the form (p, q, d, d_p, d_q), the code rate is at least $1/5$ (we have 2^t codewords and length $5t$).
- The capacity is: $C_{\text{BSC}}(\delta) = 1 - H_2(\delta)$
- Applying Shannon's theorem, an algorithm that outputs a single codeword cannot reliably decode when $1 - H_2(\delta) \leq 0.2$

Important

When $\delta \geq 0.243$ it can be shown that no algorithm can list decode using a polynomially-sized list.

sk	R	δ	HMM
(p, q, d, d_p, d_q)	$1/5$	0.243	0.237
(p, q, d)	$1/3$	0.174	0.16
(p, q)	$1/2$	0.110	0.08

The Erasure Channel

- The capacity is $1 - \rho$, where ρ is the fraction of bits erased by the channel

sk	*R*	δ	HS
(p, q, d, d_p, d_q)	$1/5$	0.8	0.73
(p, q, d)	$1/3$	0.67	0.58
(p, q)	$1/2$	0.5	0.43

The Erasure Channel

- The capacity is $1 - \rho$, where ρ is the fraction of bits erased by the channel
- The converse to Shannon's noisy channel coding theorem says that no algorithm that outputs a single codeword can reliably decode r when $1 - \rho \leq 0.2$

sk	*R*	δ	HS
(p, q, d, d_p, d_q)	$1/5$	0.8	0.73
(p, q, d)	$1/3$	0.67	0.58
(p, q)	$1/2$	0.5	0.43

The Erasure Channel

- The capacity is $1-\rho$, where ρ is the fraction of bits erased by the channel
- The converse to Shannon's noisy channel coding theorem says that no algorithm that outputs a single codeword can reliably decode r when $1-\rho \leq 0.2$

Important

For list decoding it can be shown that, on average, an exponential list of candidates will need to be considered when the code rate exceeds capacity.

sk	*R*	δ	HS
(p, q, d, d_p, d_q)	$1/5$	0.8	0.73
(p, q, d)	$1/3$	0.67	0.58
(p, q)	$1/2$	0.5	0.43

The Z-channel

The capacity is:

$$C_Z(\beta) = \log_2(1 + (1-\beta)\beta^{\frac{\beta}{1-\beta}}).$$

sk	*R*	β
(p, q, d, d_p, d_q)	$1/5$	0.666
(p, q, d)	$1/3$	0.486
(p, q)	$1/2$	0.304

The True Cold Boot Setting

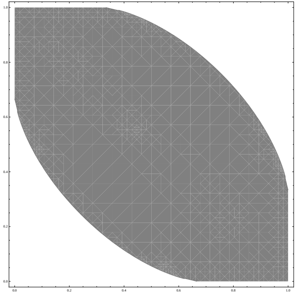


Figure: *x*-axis is α, *y*-axis is β.

- When $sk = (p, q, d)$ the capacity bound on β is 0.479.

The True Cold Boot Setting

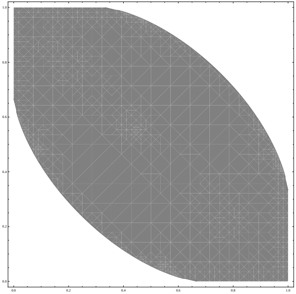


Figure: *x*-axis is α, *y*-axis is β.

- When $sk = (p, q, d)$ the capacity bound on β is 0.479.
- When $sk = (p, q)$ the capacity bound on β is 0.298.

Block-wise Partial Knowledge of p and q

- Herrmann and May (Asiacrypt 2008) used lattice-based techniques to factor N given some small number of contiguous blocks of bits in one of the primes.

Block-wise Partial Knowledge of p and q

- Herrmann and May (Asiacrypt 2008) used lattice-based techniques to factor N given some small number of contiguous blocks of bits in one of the primes.
- Works for $O(loglogN)$ blocks and they need 70% of the bits of p in total across the blocks.

Block-wise Partial Knowledge of p and q

- Herrmann and May (Asiacrypt 2008) used lattice-based techniques to factor N given some small number of contiguous blocks of bits in one of the primes.
- Works for $O(loglogN)$ blocks and they need 70% of the bits of p in total across the blocks.
- Consider the senario where the adversary is given some number of contiguous blocks of bits from both factors p and q.

Block-wise Partial Knowledge of p and q

- Herrmann and May (Asiacrypt 2008) used lattice-based techniques to factor N given some small number of contiguous blocks of bits in one of the primes.
- Works for *O*(*loglogN*) blocks and they need 70% of the bits of p in total across the blocks.
- Consider the senario where the adversary is given some number of contiguous blocks of bits from both factors p and q.
- The error rate is $\lambda/(\kappa + \lambda)$.

Block-wise Partial Knowledge of p and q

- Herrmann and May (Asiacrypt 2008) used lattice-based techniques to factor N given some small number of contiguous blocks of bits in one of the primes.
- Works for *O*(*loglogN*) blocks and they need 70% of the bits of p in total across the blocks.
- Consider the senario where the adversary is given some number of contiguous blocks of bits from both factors p and q.
- The error rate is $\lambda/(\kappa+\lambda)$.
- According to our capacity analysis $\lambda/(\kappa+\lambda) \leq 0.5$ since this is a special case of the erasure channel.

Outline

1. Motivation
2. State of the Art
3. Our Contributions
4. Experimental Results

The Erasure Channel

ρ	0.1	0.2	0.3	0.4	0.5	0.6
Success rate	1	1	1	1	1	1
Keys examined	512	512	516	527	553	627
liftings	511	511	513	520	536	593
Time per trial (s)	0.00235	0.0023	0.00232	0.00234	0.00236	0.00259

ρ	0.7	0.77	0.78	0.79	0.8
Success rate	1	1	0.98	0.77	0.4
Keys examined	971	167762	263835	923938	2875484
liftings	910	167634	263959	912849	2829735
Time per trial (s)	0.00409	0.783	1.18	4.18	13.1

Table: Experimental results for the erasure channel. Capacity bound on ρ is 0.8.

The Erasure Channel

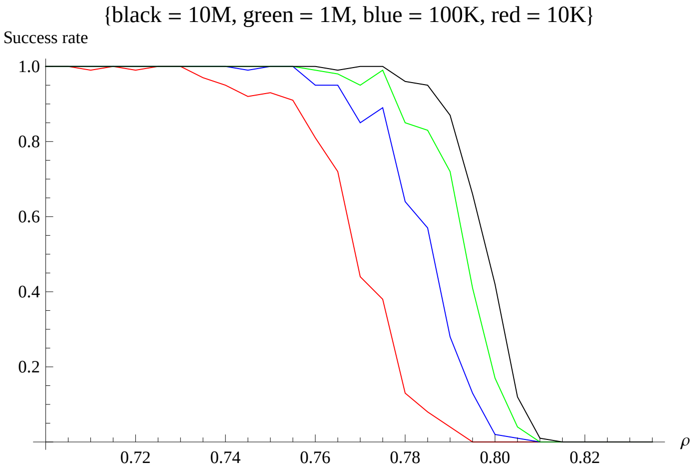
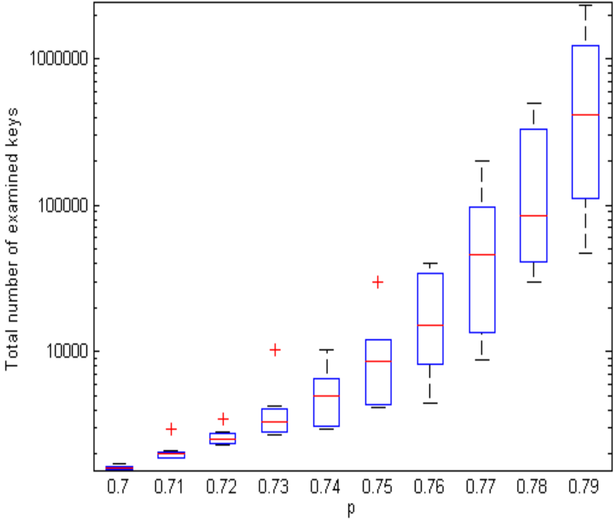


Figure: Graph showing achievable error rates based on different panic sizes.

The Erasure Channel



Partial knowledge of p and q

κ	λ	Total unknown bits	Max stack size	Keys Examined	Time (s)
2	2	510	135	31740	0.0571
4	4	508	137	44369	0.0782
6	6	508	47	15948	0.0285
8	8	504	138	172403	0.3
10	10	504	30	7942	0.014
12	12	496	48	16887	0.0292
14	14	492	61	59174	0.105
16	16	496	140	9200234	12.1
18	18	502	79	404272	0.711
20	20	484	81	1004018	1.78
22	22	496	39	25207	0.0441
24	24	496	47	134339	0.237
26	26	478	100	11521189	152

Table: Experimental results for the block-wise erasure channel with $\kappa = \lambda$.

Partial knowledge of p and q

κ	λ	Total unknown bits	Max stack size	Keys Examined	Time (s)
16	2	112	1	512	0.000907
16	4	192	1	512	0.000905
16	6	272	1	512	0.000902
16	8	336	1	512	0.000914
16	10	384	21	832	0.00141
16	12	432	39	2075	0.00345
16	14	464	108	225767	0.381
16	16	496	140	9200234	12.1
16	17	512	16	869974	1.67

Table: Experimental results for the block-wise erasure channel with $\kappa = 16$ and increasing λ.

Cold Boot Scenario

β	0.1	0.2	0.3	0.4	0.5	0.55	0.6	0.61
t	6	6	8	12	16	18	18	18
L	4	4	8	8	16	32	64	64
S.Pr	1	1	0.97	0.97	0.66	0.31	0.09	0.04

Table: Success probabilities for the true cold-boot case with $\alpha = 0.001$. Capacity bound on β is 0.658.

Cold Boot Scenario

β	0.1	0.15	0.20	0.25	0.30	0.35	0.40	0.43
t	6	10	14	16	18	18	18	18
L	4	16	16	16	16	16	32	64
$S.Pr$	0.99	0.99	0.98	0.96	0.63	0.55	0.12	0.04

Table: Success probabilities for the true cold-boot case with $\alpha = 0.001$ and $sk = (p, q, d)$. Capacity bound on β is 0.479.

β	0.05	0.1	0.15	0.20	0.26
t	10	12	16	18	18
L	8	8	16	32	64
$S.Pr$	0.95	0.83	0.68	0.29	0.06

Table: Success probabilities for the true cold-boot case with $\alpha = 0.001$ and $sk = (p, q)$. Capacity bound on β is 0.298.

Heninger & Shacham Setting

ρ	0.2	0.3	0.4	0.46	0.5	0.55	0.6	0.62	0.63
t	6	8	12	16	18	18	18	18	18
L	4	8	8	8	16	16	16	64	64
$S.Pr$	1	1	0.98	0.87	0.81	0.43	0.13	0.07	0.03

Table: Success probabilities for the idealized cold boot case ($\alpha = 0$). Capacity bound on β is 0.666.

Henecka, May & Meurer Setting

δ	0.08	0.12	0.16	0.18	0.19	0.2	0.21	0.22
t	6	10	16	18	18	18	18	18
L	4	8	32	32	32	32	32	64
$S.Pr$	1	0.93	0.84	0.60	0.38	0.20	0.08	0.04

Table: Success probabilities for the symmetric case ($(\alpha, \beta) = (\delta, \delta)$). Capacity bound on δ is 0.243.

Henecka, May & Meurer Setting

Table: Success probabilities for the symmetric case, $\alpha = \beta$.

α	0.06	0.08	0.12	0.16	0.19	0.2	0.21	0.22
HMM	0.48	0.5	0.5	0.35	0.24	0.21	-	-
ML	1	1	0.93	0.84	0.38	0.20	0.08	0.04

Summary

- We have considered a more general setting than HS and HMM.

Summary

- We have considered a more general setting than HS and HMM.
- We use the converse to Shannon's theorem, derive bounds on list decoding to establish limits on the noise levels.

Summary

- We have considered a more general setting than HS and HMM.
- We use the converse to Shannon's theorem, derive bounds on list decoding to establish limits on the noise levels.
- For practical RSA key sizes our algorithm outperforms the previous approaches.

Summary

- We have considered a more general setting than HS and HMM.
- We use the converse to Shannon's theorem, derive bounds on list decoding to establish limits on the noise levels.
- For practical RSA key sizes our algorithm outperforms the previous approaches.
- Ours is the first algorithm to solve the motivating cold-boot problem.