Hardware Cracker for Integer Factorization
Designs, Costs and Feasibility

Colin Stahlke

EDIZONE GmbH, Bonn, Germany
stahlke@edizone.de

Outline of the talk

- RSA, factoring, algorithms, ideas
- Running times, what does this mean?
- How secure are 128 bit ?
- RSA with large bit sizes?
- A question about economy
- What are the costs to break RSA?
- Hardware design and Moore's Law
- Three ideas for the SHARK machine
- Conclusions

Outline of the talk

- RSA, factoring, algorithms, ideas
- Running times, what does this mean?
- How secure are 128 bit ?
- RSA with large bit sizes?
- A question about economy
- What are the costs to break RSA?
- Hardware design and Moore's Law
- Three ideas for the SHARK machine
- Conclusions

RSA and Factoring

To break RSA it suffices to factor the used modulus N.

$N=pq$

p, q extremely large primes

Best knowledge of today:

Apparently breaking RSA is as hard as factoring N.

How safe is RSA?

Some Modern Factoring Algorithms

- **N**umber **F**ield **S**ieve
  - due to John Pollard for special numbers in 1988 (SNFS)
  - for general numbers due to Josef Buhler, Hendrik Lenstra and Carl Pomerance in 1990 (GNFS)
  - see A.K. Lenstra and H.W. Lenstra (eds.), The development of the number field sieve, Springer LNM 1554, Springer, 1993

  - note here that GNFS is currently also the best method for attacking the DL problem in $\mathbf{F}_p^*$ of comparable bit sizes

- **Q**uadratic **S**ieve (QS) and

  Multiple **P**olynomial **Q**uadratic **S**ieve (MPQS)

- **E**lliptic **C**urve **M**ethod (ECM), 1985

Basic Idea behind sieving

Find enough of certain congruences mod N within a sieving step

and finally solve a huge sparse system of linear equations over $\mathbf{F}_2$.

- QS: start with one quadratic polynomial
- MPQS: use many polynomials yielding smaller sieving areas per polynomial
- Number field sieve: polynomials of higher degree involved, sieving in algebraic number fields

Outline of the talk

- RSA, factoring, algorithms, ideas
- Running times, what does this mean?
- How secure are 128 bit ?
- RSA with large bit sizes?
- A question about economy
- What are the costs to break RSA?
- Hardware design and Moore's Law
- Three ideas for the SHARK machine
- Conclusions

Running times

$t, c \in \mathbb{R}, \; N \to \infty$

$$L_N[t, c] = e^{(c+o(1))(\log N)^t (\log\log N)^{1-t}}$$

Expected running times based on heuristics

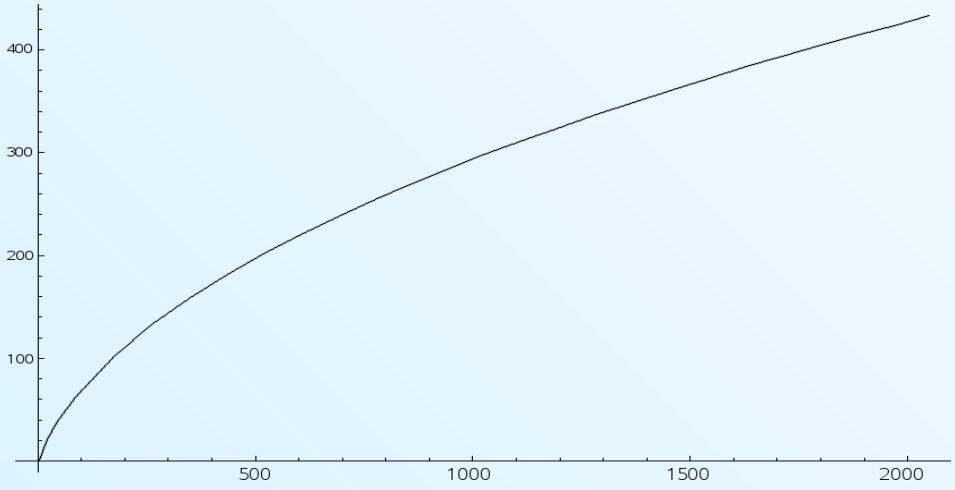
- GNFS: $L_N[1/3, (64/9)^{1/3}]$ and for SNFS: $L_N[1/3, (32/9)^{1/3}]$
- QS and MPQS: $L_N[1/2, 1]$
- ECM: $L_p[1/2, \text{sqrt}(1/2)]$ and for p of size $\sqrt{N}$ get $L_N[1/2,1]$ as for QS

What does this mean?

- Coppersmith's variant of GNFS has the best asymptotics.
- Until several years ago it was not clear whether GNFS or MPQS would be best in practice.

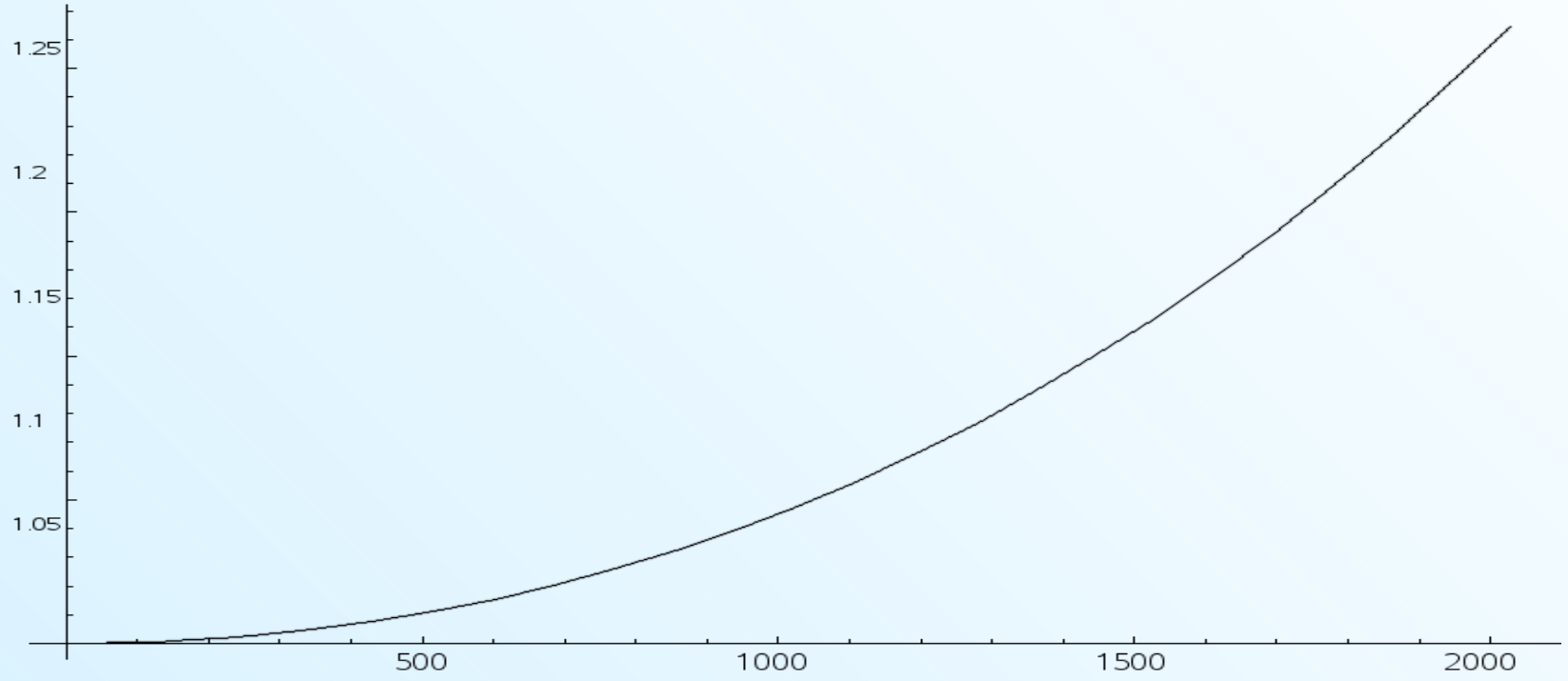
Example: $\widetilde{L_N}[1/3,(64/9)^{1/3}] = e^{((64/9)^{1/3}+\tilde{o}(N))(\log N)^{1/3}(\log\log N)^{2/3}}$

$\tilde{o}(N) = 0$

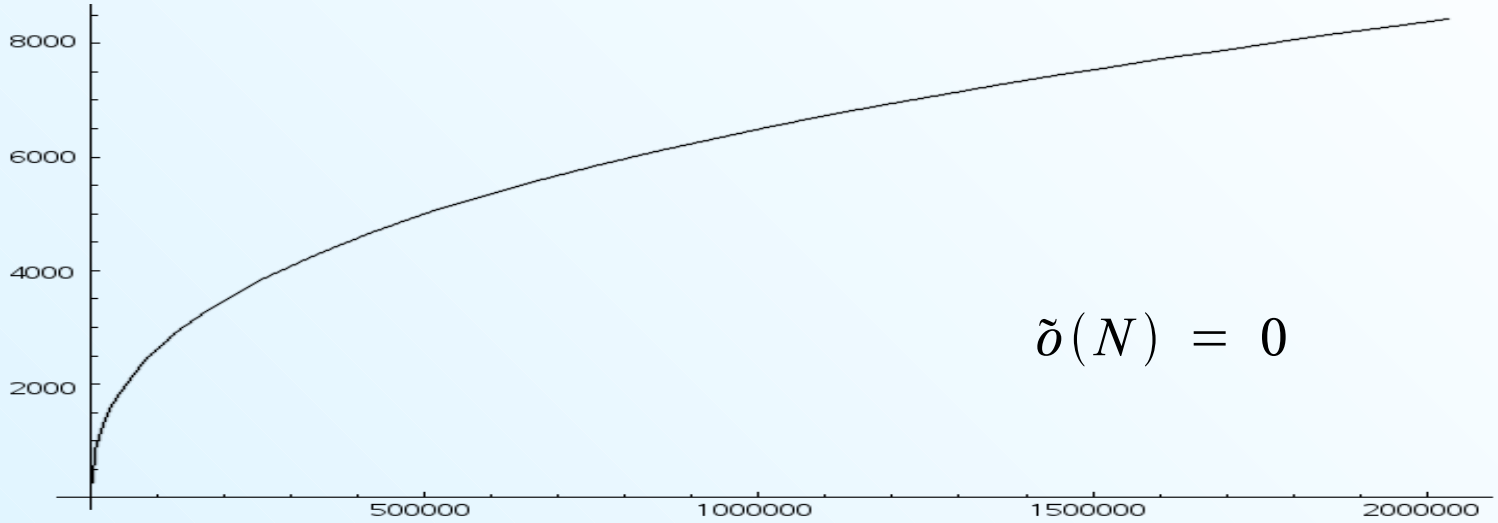


What does this mean?

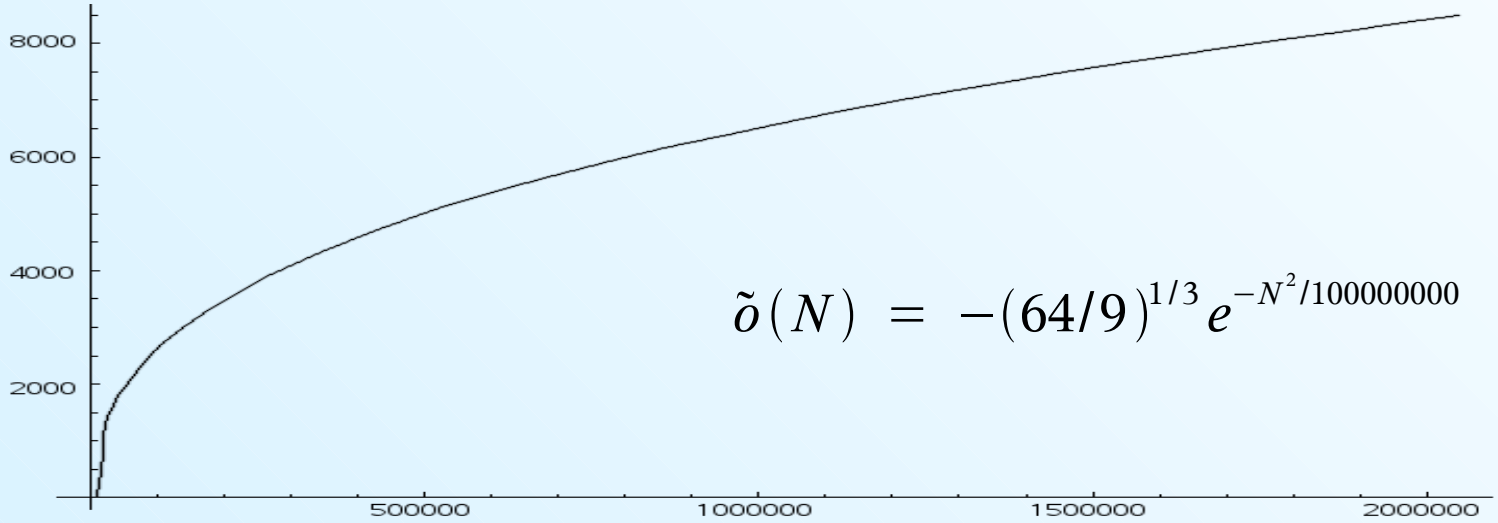
$$\tilde{o}(N) = -(64/9)^{1/3} e^{-N^2/100000000}$$



What does this mean?



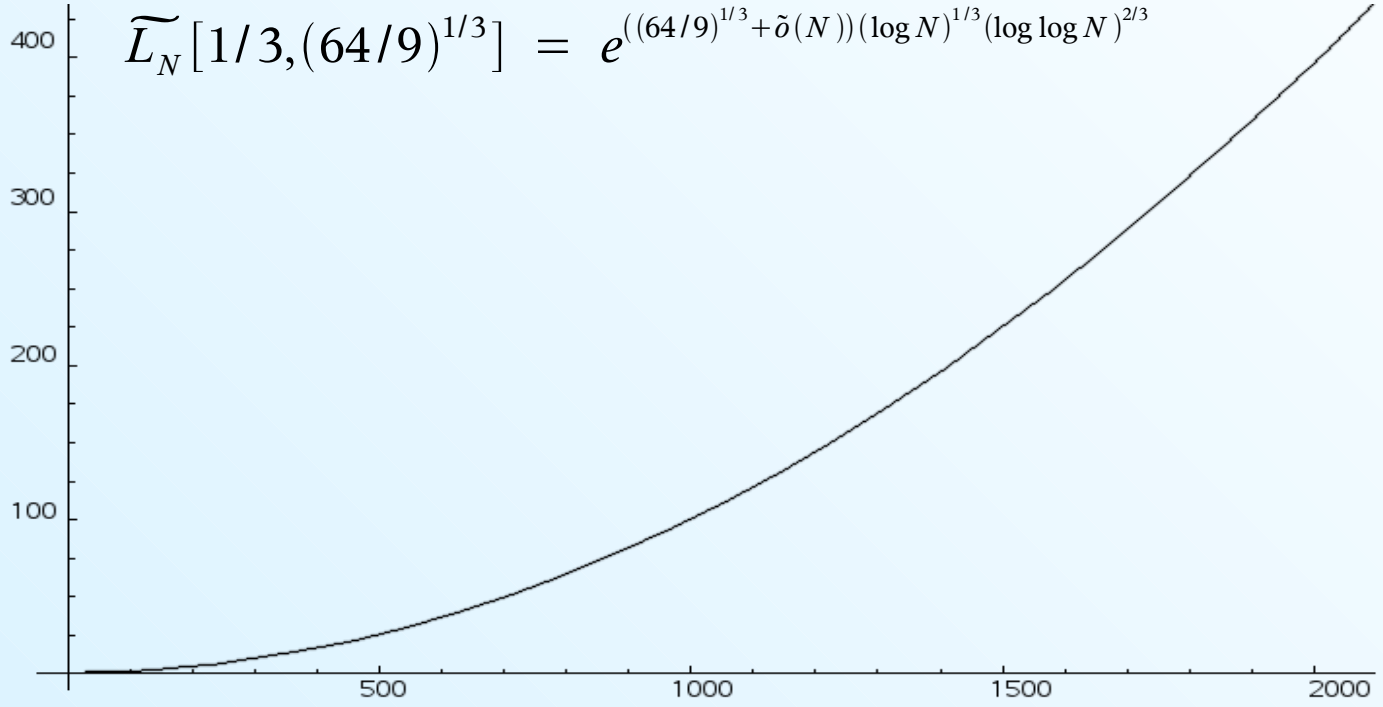
$$\tilde{o}(N) = 0$$



$$\tilde{o}(N) = -(64/9)^{1/3} e^{-N^2/100000000}$$

What does this mean?

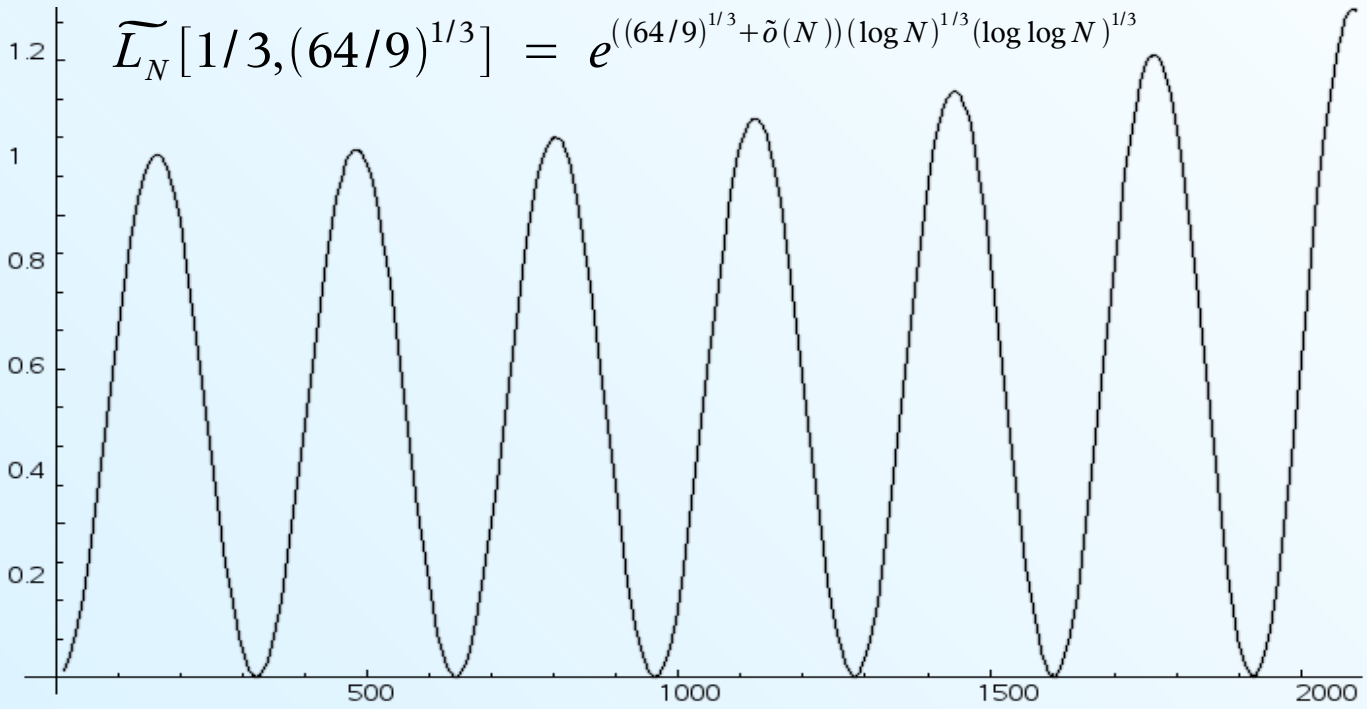
$$\tilde{o}(N) = e^{-N^2/100000000}(-(64/9)^{1/3} + \log(N^2/10000)/((\log N)^{1/3}(\log\log N)^{2/3}))$$



This looks almost like the function $N^2/10000$

What does this mean?

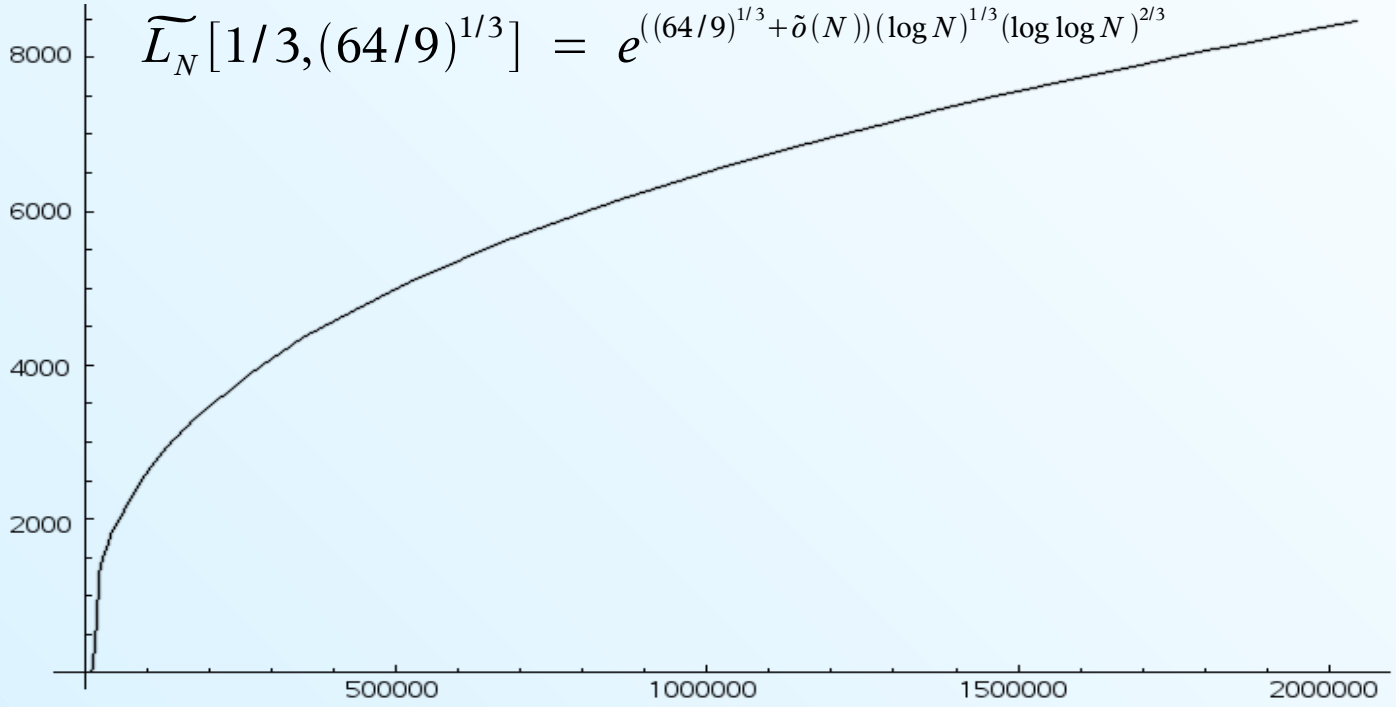
$$\tilde{o}(N) = e^{-N^2/100000000}(-(64/9)^{1/3} + \log(\sin(N/100)^2)/((\log N)^{1/3}(\log\log N)^{2/3}))$$



Now it looks almost like the function $\sin(N/100)^2$

What does this mean?

$$\tilde{o}(N) = e^{-N^2/100000000}(-(64/9)^{1/3} + \log(\sin(N/100)^2)/((\log N)^{1/3}(\log\log N)^{2/3}))$$



Now it looks almost as if $\tilde{o}(N) = 0$

What does this mean?

- Asymptotics says nothing about the behaviour of a function at finite places.
- Adi Shamir: Constants are important!
- Asymptotic complexity does not help much to estimate the security of RSA.

How secure is RSA?

Outline of the talk

- RSA, factoring, algorithms, ideas
- Running times, what does this mean?
- How secure are 128 bit ?
- RSA with large bit sizes?
- A question about economy
- What are the costs to break RSA?
- Hardware design and Moore's Law
- Three ideas for the SHARK machine
- Conclusions

How secure are 128 bit?

Imagine a perfect symmetric algorithm that can only be broken by brute force.

DES 56 bit can be broken, suppose within 1 second for at least 1 USD (we need a safety margin!).

$$2^{128} = 2^{25}\,2^{56}\,2^{47}$$

We need at least 100,000,000,000,000 USD to break a perfect 128 bit symmetric cipher within a year.

SAFE!

How secure are 80 bit?

Imagine a perfect symmetric algorithm that can only be broken by brute force.

DES 56 bit can be broken, suppose within 1 second for at least 1 USD (we need a safety margin!).

$$2^{80} = 2^{25} 2^{56} 2^{-1}$$

We need at least 0.50 USD to break a perfect 80 bit symmetric cipher within a year.

- We need more sophisticated methods for RSA.
- Asymmetric crypto with usual bit lengths seems less secure than symmetric crypto.

Outline of the talk

- RSA, factoring, algorithms, ideas
- Running times, what does this mean?
- How secure are 128 bit ?
- RSA with large bit sizes?
- A question about economy
- What are the costs to break RSA?
- Hardware design and Moore's Law
- Three ideas for the SHARK machine
- Conclusions

RSA with large bit sizes?

Use of asymmetric crypto in:

computers, home electronic, cellphones, PDAs, cameras,
access control, sensor networks, satellites, cars and other vehicles,
chipcards and RFIDs

RSA with large bit sizes?

- To estimate the security of RSA is to estimate the costs in terms of time and money to break it.
- There are estimates between 1 Mio. and 200 Mio. USD for breaking RSA 1024 within one year.
- The costs for small devices depend directly on the bit length, thus on security.
- We need a trade off between security and costs.

How secure is RSA?
What are the costs to break RSA?

Outline of the talk

- RSA, factoring, algorithms, ideas
- Running times, what does this mean?
- How secure are 128 bit ?
- RSA with large bit sizes?
- A question about economy
- What are the costs to break RSA?
- Hardware design and Moore's Law
- Three ideas for the SHARK machine
- Conclusions

A question about economy

- For economical reasons RSA cannot be made as secure as symmetric ciphers in most applications.
- It is crucial to give a precise estimate for the security of RSA in order to use it in common applications, because:
  - Breaking the cipher can lead to great economical loss.
  - Choosing a bit size with a too large safety margin wastes a lot of money and makes many applications impossible.
- This is true for all asymmetric ciphers.

A question about economy

Suppose you can break RSA with PCs for 10,000,000,000 USD.

Different strategies:

- Buy PCs for 10,000,000,000 USD.
- Use hacked PCs in the internet (rent them for 1 USD a day) and cut costs by a factor of 100.
- Write your own trojan horse and use spare processing time (scalability?).
- Write a cool game for Sony's Playstation 3 that uses little resources and sieves in the background.

A question about economy

Is RSA 1024 bit more secure than RSA 900 bit?

Different strategies:

- Build a machine that breaks RSA 900 and use it once.
- Refinancing: Build a machine that breaks RSA 1024 and break several RSA 1024 bit keys for other people that pay you. Then use it to break RSA 900.
  Drawback: If each key needs a year to be broken, breaking RSA 900 takes several years.

A question about economy

Observations:

- A key is less secure if it protects a greater value.
- The value of a key is different for different attackers.

For each crypto application we need to answer several questions:

- What are the values to be protected?
- Is it enough to break one root key or one key for every delivered application?
- Who are the potential attackers?
- What would be their revenue?
- Will they attack the application or do they have better oportunities?
- What are the costs to break the key?

Outline of the talk

- RSA, factoring, algorithms, ideas
- Running times, what does this mean?
- How secure are 128 bit ?
- RSA with large bit sizes?
- A question about economy
- What are the costs to break RSA?
- Hardware design and Moore's Law
- Three ideas for the SHARK machine
- Conclusions

What are the costs to break RSA?

Assumption:

- The costs are the costs to build and run some kind of machine that breaks RSA (excluding development costs).

First question:

- What is the smallest RSA key length that cannot be broken?

How to find the answer:

- Design a machine that can break the key length, which is
  - just too expensive to be built and
  - optimal in terms of costs

What are the costs to break RSA?

Second question:

- What is the smallest RSA key length that cannot be broken within 40 years?

How to find the answer:

- Design a machine using future technology that can break the key length, which is
  - just too expensive to be built and
  - optimal in terms of costs

Even if today's designs seem adventurous, such designs are needed to answer the fundamental question about the security of today's applications using RSA.

What are the costs to break RSA?

Observations about the time consuming parts of code breaking machines:

- Specialized hardware is cheaper than general purpose hardware.
- The time area product of ASICs must be low.
- Memory is rather expensive. Of-the-shelf RAM is cheapest.
- Computation is rather cheap.
- Communication is expensive.

What are the costs to break RSA?

Proposal for a roadmap:

- Find the best mathematical algorithm to factor an RSA modulus.
- Create a rough design of an implementation in ASICs.
- Minimize communication. Extensively use local computations.
- Minimize use of memory, especially of fast access memory.
- Try to concentrate large amounts of memory at few places in order to replace it by of-the-shelf RAM.
- Go back to the design of the mathematical algorithm to adapt it to the new constraints, computation capabilities and adjust the parameters.

Outline of the talk

- RSA, factoring, algorithms, ideas
- Running times, what does this mean?
- How secure are 128 bit ?
- RSA with large bit sizes?
- A question about economy
- What are the costs to break RSA?
- Hardware design and Moore's Law
- Three ideas for the SHARK machine
- Conclusions

Hardware design and Moore's Law

- Moore's Law: computing power doubles every 18 month
- valid since 40 years
- hardware crackers 1000 times cheaper after 15 years
- sieving step for RSA 1024 bit:
  - TWIRL (2003, 1 Mio. USD) used “futuristic” 90 nm technology, today 65 nm is in use
  - SHARK (2005, 200 Mio. USD) much cheaper today
- 40 more years for Moore's Law?

Hardware design and Moore's Law

Limits for Moore's Law

- 4 nm technology: Silicon layer will be 10 atoms thick,
  doping will lead to unpredictable transistor characteristics.

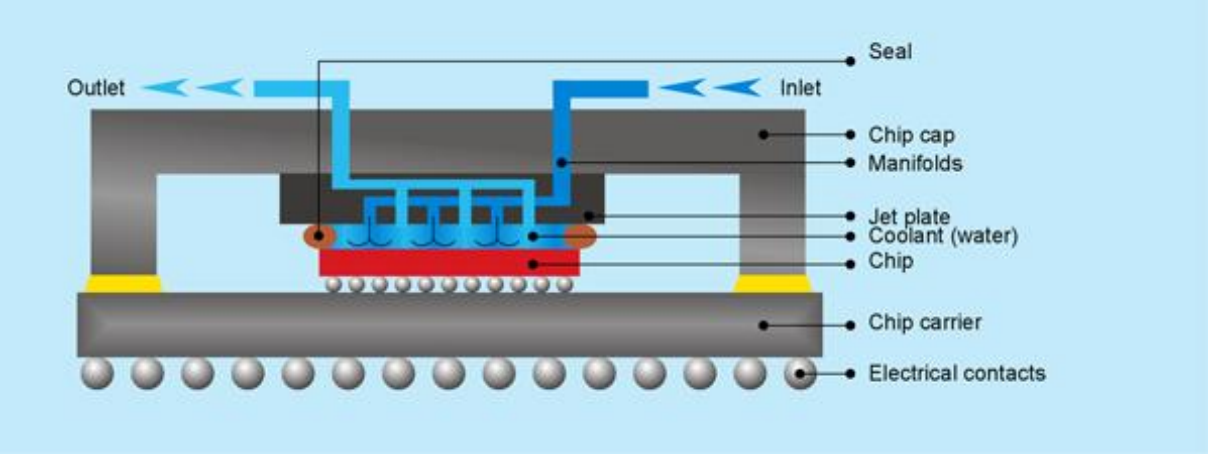
- Larger chips have smaller yield.
  The yield dropped from the 386 processor to the Pentium processor
  from 71% to 9%.

- The chips produce more and more heat, e.g. 100 Watt / cm².
  Nobody has an idea how to build a cooling for more than
  200 Watt / cm², except ...

Hardware design and Moore's Law

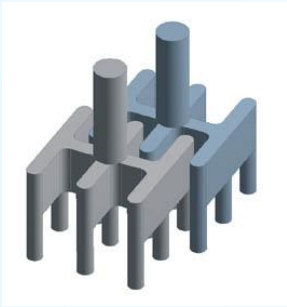
The IBM Zurich Research Laboratory

The technique employs a distributed return architecture with alternating inlets and outlets to squirt small amounts of water onto the chip and suck them off again, The 50,000 channels are 30-50 micrometers wide and made with microtechnology (MEMS). With the JAC a cooling performance of up to 370 W/cm2 was demonstrated with water as coolant.



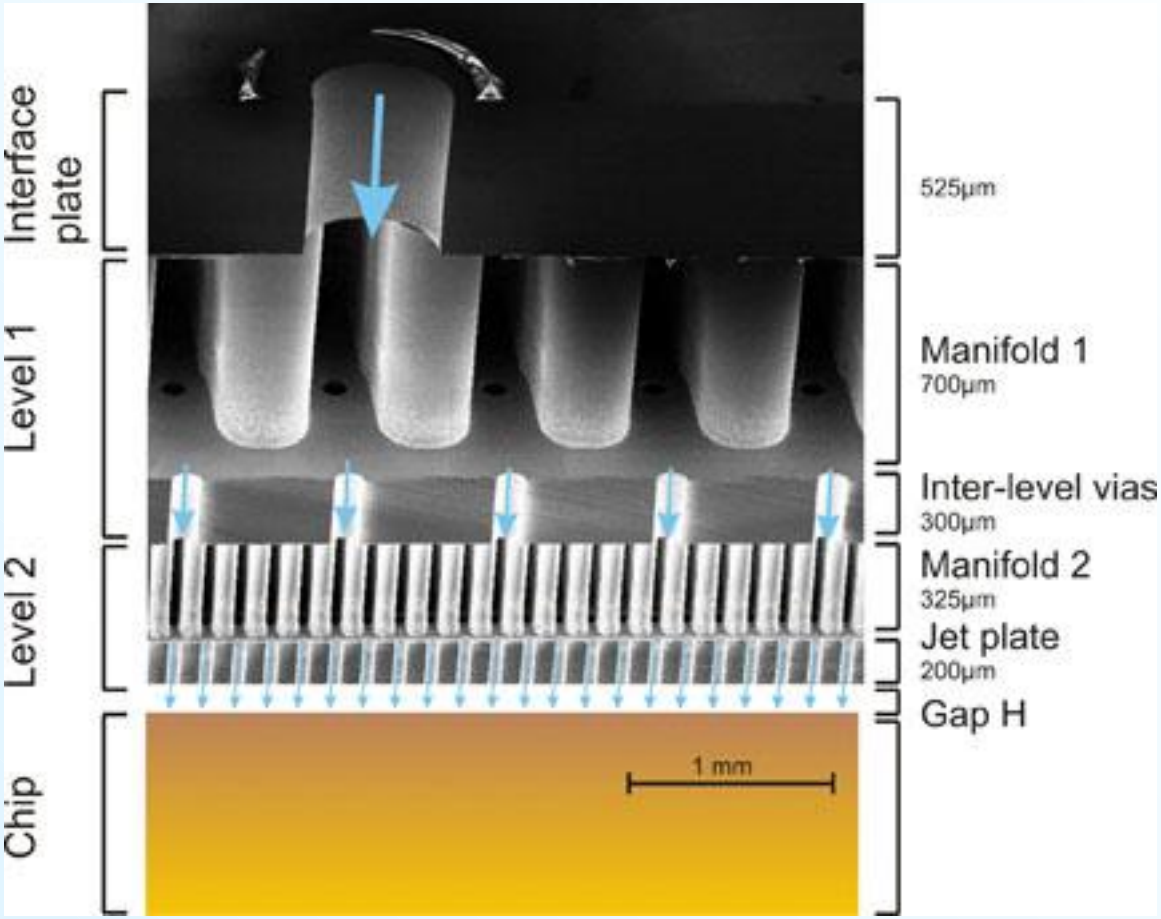
Representation of one inlet and outlet tree, similar to the human vascular system.

cools up to 370 Watt /cm²



Hardware design and Moore's Law

Electron microscope image showing a cross section through the jets and four hierarchical layers of manifolds (jets).



Outline of the talk

- RSA, factoring, algorithms, ideas
- Running times, what does this mean?
- How secure are 128 bit ?
- RSA with large bit sizes?
- A question about economy
- What are the costs to break RSA?
- Hardware design and Moore's Law
- Three ideas for the SHARK machine
- Conclusions

Three ideas for the SHARK machine

The General Number Field Sieve (GNFS) is an algorithm to factor integers.

Most expensive steps:

- sieving step
- matrix step (or linear algebra step)

Actual record: GNFS in software factorized a 663 bit RSA modulus.

SHARK is a design for a machine that performs the sieving step of the GNFS for a 1024 bit RSA modulus within one year.

Three ideas for the SHARK machine

Some costly properties of GNFS:

- GNFS for 1024 bit is huge. Squeezing the whole GNFS in one large ASIC would lead to a dramatic drop in the yield.
- GNFS needs tremendous amounts of memory. Memory should be collected in larger chunks in order to use of-the-shelf RAM.
- GNFS needs a lot of sorting data and communication. Sorting data and communication should be reduced and organized in a clever way.

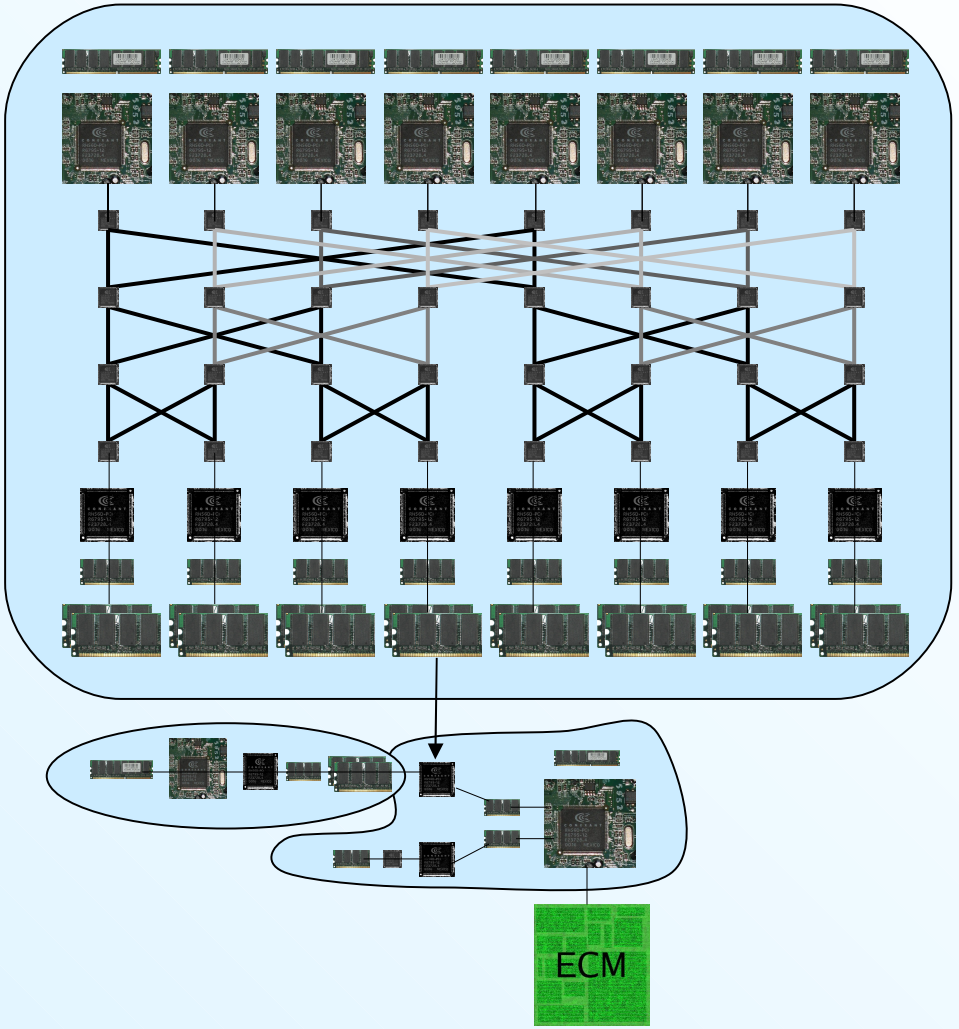
Three ideas for the SHARK machine

First idea:

Perform different types of computations at different locations in the machine using many small ASICs.

Second idea:

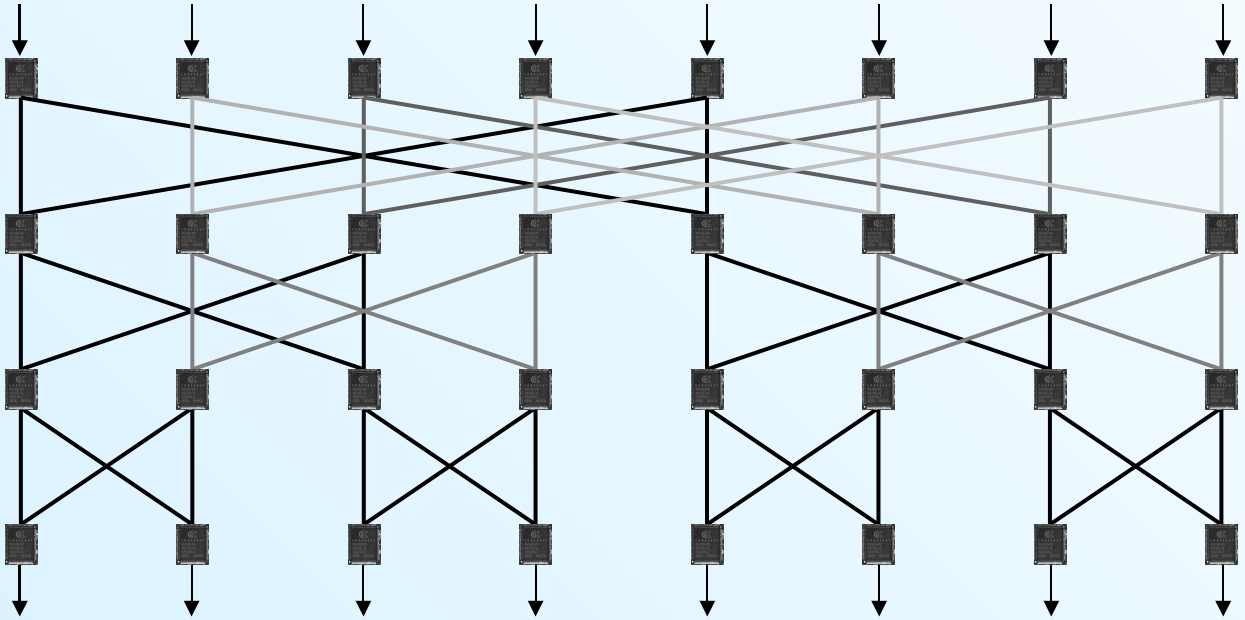
This partition should respect the memory needs. Equip each ASIC with its own of-the-shelf RAM.



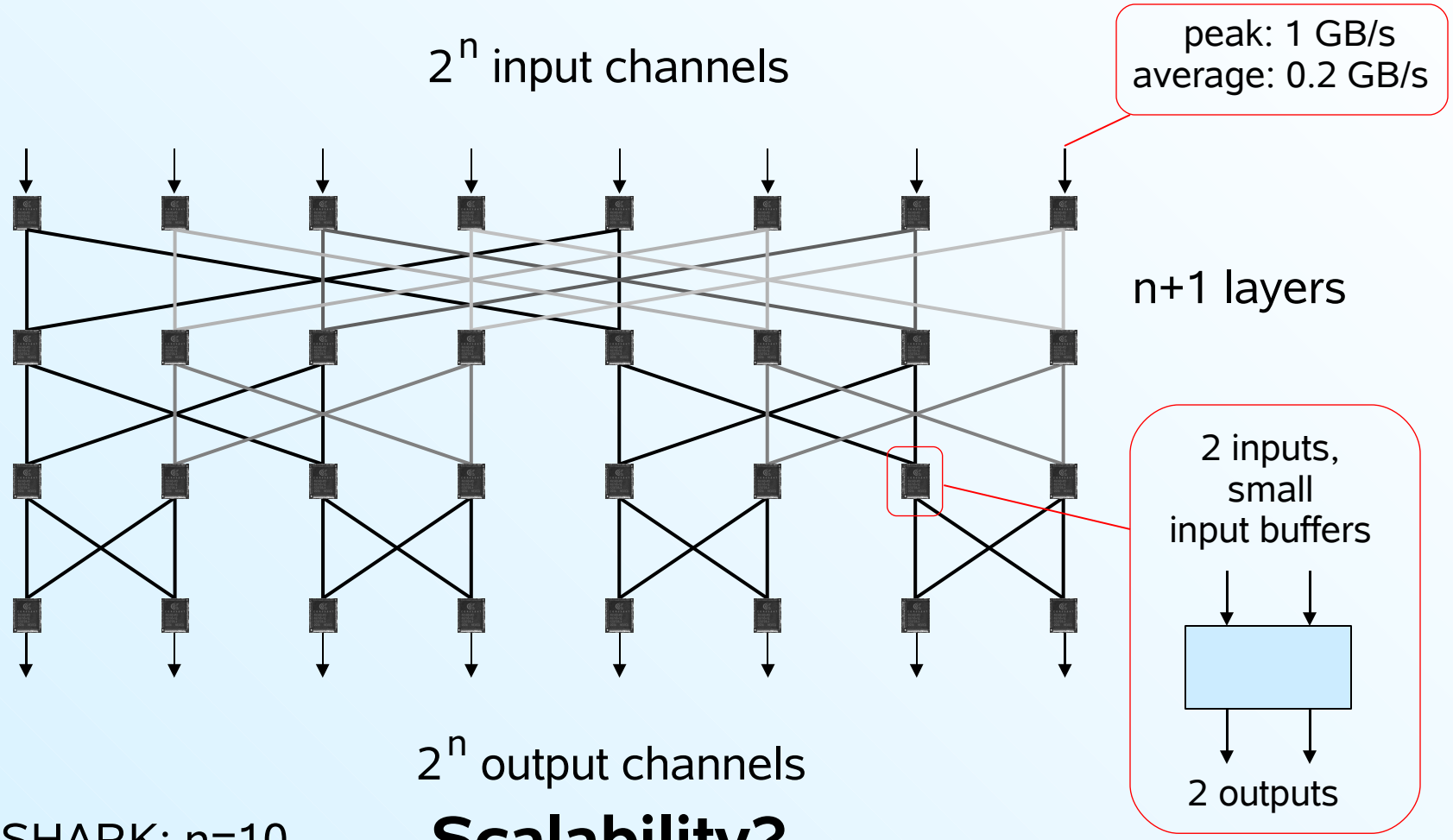
Three ideas for the SHARK machine

Third idea: This partition should respect the sorting and communication needs. All major communication should be one-way, such that the communication speed is not important, only the throughput. Sorting is only needed for the large primes of the factor base and is performed by the

butterfly transport system.



Three ideas for the SHARK machine



SHARK: n=10

Scalability?

Conclusions

- The safety margin for asymmetric cryptography in today's products is small.

- Economy badly needs accurate estimations about this safety margin, i.e. about the costs to break the asymmetric cipher for every used key length.

- Estimates can only be achieved by creating and analysing designs for hardware crackers.

- This analysis needs
  - Mathematics,
  - Hardware Engineering,
  - economical facts,

  and a lot of common sense to bring everything together.