Information security, cryptology, and factoring

Arjen K. Lenstra
Lucent Technologies’ Bell Laboratories
Technische Universiteit Eindhoven

Outline

• Information security and cryptology
• Examples of progress in cryptology and their impact
• Integer factorization

Disclaimer & warning:
• opinions not necessarily shared by any of my employers
• ‘Theoretische Informatica’ mostly avoided

Industry view on information security

Goal is achieving ‘CIA’
• Confidentiality
• Integrity
• Availability

• not because of idealism or because industry cares about your privacy, but forced by regulations
• cheapest solution industry can get away with is the best
Cryptology
Cryptology consists of:
• Cryptography:
  design and application of data protection methods
• Cryptanalysis:
  evaluation of security of cryptographic methods
⇒ Cryptology looks crucial to achieve and maintain CIA
⇒ Cryptologists like to argue the practical importance of their field for Information Security
Rightly so: cryptology was indeed crucial to achieve the current state of affairs
But: to what extent do cryptologic ‘events’ affect real life? (as opposed to hackers, viruses, stupidities (OSs), …)
Do we, from a business point of view, need more crypto?

Cryptology in the real world
In practice: comfortable cryptocentric picture somewhat obscured by a variety of unpleasant real life issues
Just a few, in random order:
users, employees, passwords, policies & their enforcement, monitoring, auditing, access control, profit/losses, legislation, verification, liabilities, risk management implementation, legacy systems, incompetence, confusion, laws, juries, lethargy, stupidity, software, errors, hackers, operating systems, inertia, viruses, networks, public relations, public perception, conventions, standards, physical protection, …
Often argued: security is like a chain, as strong as the weakest link
It may also be argued that this chain is hidden in a mud pie, hard to find the links, to figure out if they hang together, if anyone notices or cares if it’s removed altogether:
…the mud pie will still be there…

Do we need new cryptology?
Industry point of view
Academic point of view
New cryptography?
Thanks, but no thanks, we’re fine …mostly
Of course needed: we keep churning out papers
New cryptanalysis?
We don’t really care, we keep heaping mud
Of course needed: so industry know what it gets and, never mind, sometimes we actually break something…
Bruce Schneier: Currently encryption is the strongest link we have. Everything else is worse: software, networks, people. There’s absolutely no value in taking the strongest link and making it even stronger

Does cryptologic progress have any impact?
Examples:
• symmetric cryptanalysis:
  • the Data Encryption Standard (DES)
  • Secure Hash Algorithm (SHA1)
• asymmetric cryptanalysis:
  • breaking PKCS#1
  • progress in factoring
• cryptography:
  • the rise of provable security
**The Data Encryption Standard**
- Introduced in 1977, 56 bits of security (crack in time $2^{56}$)
- Regarded with utmost suspicion, by some
- Widely used ⇒ ok to use
- 1993: probably breakable in 4 hours for US$ 1 million
- 1997: one encryption broken, in 4 months, for free
- 1998: US$ 130,000 device: breaks encryption in 4 days
- 2000: Advanced Encryption Standard (AES) announced
- 2004: NIST says (single) DES inadequate (for feds)
- 2005: DES still widely used (just do risk analysis - no incidents, yet), but new deployments (should) become rare
⇒ cryptanalysis hardly impacted course of events

**Secure Hash Algorithm (SHA1)**
- Finding $b \neq b'$ with SHA1($b$) = SHA1($b'$) must be hard
- Introduced in 1994, as last minute replacement of SHA0
- Design based on ‘public’ developments, generally liked
- August 2004: many related hashes badly broken, but:
  - until Feb 2005: SHA1 believed to offer 80-bit security, finding $b$ and $b'$ would take year on US$ 20B device
  - Feb 7, 2005, NIST: SHA1 not broken, ok until 2010
  - Feb 14, 2005: SHA1 offers at most 66 bits of security, $b$ and $b'$ in at most about a year on US$ 1M device
- Oops! But anyone really concerned? Any impact? possibly: see http://www.win.tue.nl/~bdeweger/CollidingCertificates/
- SHA0 offers at most 39 bits of security...

**Breaking PKCS#1**
- 1976-1998: (mostly) happy-go-lucky design of protocols ‘if no one can break it, it’s most likely secure’
- 1993: publication of RSA encryption standard PKCS#1 following the trusted HGL design strategy
- PKCS#1 actually deployed
- 1998: adaptive chosen ciphertext attack against PKCS #1
  - ‘broken’ from academic point of view: protocol fooled into revealing secret information without cracking the underlying problem (RSA)
  - in practice often hard to exploit
  - ‘may be the current design approach is not the right one’

**Provable security**
- took of in 1998 with Cramer/Shoup encryption scheme: reasonably practical and provably secure against attacks
- Smart marketing ploy: no relation to actual provable security
- Actual meaning is: ‘provably reducible’, getting secret information is provably as hard as solving the underlying hard problem
- Unwritten rule, strictly enforced in academia: all new protocols must be ‘provably secure’ (what about their implementation?)
- Slowly, new protocols make it to standards and products ⇒ impact on new standards & systems, barely on existing ones
Factoring
• Given a composite, how to find a non-trivial factor
  • given 15, how to find 3 or 5
  • how do you know that 15 is composite to begin with?
• what does this have to do with cryptography?

Factoring
• Given a composite, how to find a non-trivial factor
  • given 91
  • how do you know that 91 is composite?
  • what does this have to do with cryptography?
  
  because ‘Primes are in P’ (and so are composites), not only from a theoretical but also from an industrial point of view:
  Fermat’s little theorem: if \( n \) is prime, then for all integers \( a \):
  \[ n \text{ divides } a^n - a \] (i.e., \( a^n \equiv a \) modulo \( n \))
  \[ \Rightarrow \text{If an integer } a \text{ is found such that } n \text{ does not divide } a^n - a, \text{ then } n \text{ is composite (without information about } n \text{’s factors)} \]

  souped up version works ‘always’ – and, with CS101, efficiently too

  • what does this have to do with cryptography?

Factoring and cryptography (RSA)

**red is A’s secret information, green is public**
• User \( A \) selects primes \( p \) and \( q \), computes \( n = pq \), and integers \( e \) and \( d \) such that
  \[ ed = 1 + k(p - 1)(q - 1), \quad k \in \mathbb{Z} \]
• \( A \) makes \( n \) and \( e \) public, keeps \( d \) secret
  (may throw \( p \) and \( q \) away)
• To encrypt message \( m \) intended for \( A \):
  \[ E(m) = m^e \text{ mod } n \]
• No one can make sense of \( E(m) \), except \( A \):
  \[ E(m)^d = (m^e)^d \text{ mod } n = (m^{1+k(p-1)(q-1)}) \text{ mod } n = m \]
  because \( m^{p-1} = 1 \text{ mod } p \), \( m^{q-1} = 1 \text{ mod } q \), and \( n = pq \)

How to select the modulus in RSA?
• RSA can be broken if the modulus \( n \) can be factored (and who knows in how many other ways)
• RSA is efficient if the modulus \( n \) is small
  \[ \Rightarrow \text{Try to select the modulus as small as possible in such a way that the modulus cannot be factored} \]
• Need to know what size numbers cannot be factored, now, and in the foreseeable future

  Same as familiar ‘practical relevance’ argument for xxx: mostly bogus, xxx addicts do it because they like it
A ‘recent’ history of integer factorization & results

<table>
<thead>
<tr>
<th>year</th>
<th>factorization event</th>
<th>most wanted</th>
<th>RSA length</th>
</tr>
</thead>
<tbody>
<tr>
<td>1969</td>
<td>Invention of CFRAC:</td>
<td>$F_7 = 2^{128} + 1$</td>
<td>4 digits = 3.32 bits</td>
</tr>
<tr>
<td>1970</td>
<td>Factorization of $F_7$</td>
<td>$F_8 = 2^{256} + 1$</td>
<td>80-129 digits</td>
</tr>
<tr>
<td>1976</td>
<td>Invention of RSA</td>
<td>$F_9 = 384-512$ bits</td>
<td></td>
</tr>
<tr>
<td>1977</td>
<td>Invention of linear sieve:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1978</td>
<td>1979, almost factorization of $F_9$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1980</td>
<td>Pollard-$\rho$ factorization of $F_8$</td>
<td>$F_9 = 2^{312} + 1$</td>
<td></td>
</tr>
<tr>
<td>1981-3</td>
<td>Development of quadratic sieve (QS)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1985</td>
<td>Invention of elliptic curve method (ECM),</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1988</td>
<td>Factorization of $F_{10}$</td>
<td>155 digits</td>
<td></td>
</tr>
<tr>
<td>1989</td>
<td>Factorization by internet QS</td>
<td>512-768 bits</td>
<td></td>
</tr>
<tr>
<td>1990</td>
<td>Pollard invents special number field sieve</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1994</td>
<td>Factorization of 129-digit modulus by QS</td>
<td>768-1024 bits</td>
<td></td>
</tr>
<tr>
<td>1999</td>
<td>Factorization of 512-bit modulus by NFS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2003</td>
<td>1024-bit RSA moduli are widely used, and still recommended</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Problem: since 1989 nothing seems to be happening!

More examples of things that did not happen:

• 1994, integers can quickly be factored on a quantum computer but no one knows how to build one

• 1999, TWINKLE opto-electronic device to factor 512-bit moduli estimates a bit too optimistic (device never actually built)

• 2001, Bernstein’s factoring circuits: 1536 bits for cost of 512 bits based on a neat accounting trick (sparked new research)

• 2003, TWIRL hardware siever: 1024 bits in a year for US$1-10M somewhat challenging design (unlikely that it will be built)

• 2005, SHARK hardware siever: 1024 bits in a year for < US$200M conservative design and estimates

Pollard’s mnemonic for $F_9$ factorization

In 1990 we found a 49-digit prime factor of $F_9$:

745560282564788420833739573620045491878366342657

which can easily be memorized as

MASSIVE TEAM BROKE NINTH FERMAT!

It factored as three primes, June fifteen (forenoon) nineteen nine oh. Actually one can explain the algorithm quite quickly and easily, er . . Well, space here precludes a detailed account - candidly, the big double search was done by Number Field Sieving (Periods (full stops) and exclamation marks denote single zeros. Two dots denote double zero. Other punctuation is ignored.)

Factoring algorithms

Special purpose methods

- Take advantage of special properties of factor $p$ to be found

  Examples:
  - Trial division, Pollard-$\rho$ (find small $p$)
  - Pollard-$\rho$–1 (find $p$ such that $p$–1 has small factors)
  - Elliptic curve method (ECM) (find small $p$)

General purpose methods

- Cannot take advantage of any properties of $p$

  Examples: All based on same, apparently wrong, approach
  - CFRAC, Dixon’s algorithm
  - Linear sieve, Quadratic sieve
  - Number field sieve (NFS) ← Relevant for RSA
**Intermezzo on runtimes**

Trial division takes time $n^{1/2}$, Pollard-$\rho$ time $n^{1/4}$ (worst case)

Because $n^4 = (e^{1/24})^4$ this is called exponential-time (very bad)

rewriting $(e^{1/24})^4$ as: $\exp(k(\ln n)4/(\ln \ln n)^4)$

Anything in between is called subexponential-time

Halfway point: $\exp(k(\ln n)^{1/2}(\ln \ln n)^{1/2})$ (not good: bad)

is runtime of CFRAC, Dixon, linear&quadratic sieve, ECM

and the best we could do until 1989

RSA’s dream destroyed by Pollard’s NFS:

runtime $\exp(k(\ln n)^{1/3}(\ln \ln n)^{2/3})$

still subexponential-time (bad, but not so bad)

Factoring on quantum computer takes time $(\ln n)^k$ for constant $k$

$(\ln n)^k$ is called polynomial-time (good, if $k$ is decent)

rewrite $(\ln n)^k$ as: $\exp(k(\ln n)^{1}(\ln \ln n)^{1})$

**How to factor numbers?**

- We have no clue
- Try to write $n$ as $x^2 - y^2 = (x - y)(x + y)$
  - Example: $n = 91 = 100^2 - 3^2$
- More generally: try to find integers $x \neq y$ such that
  $x^2 = y^2 \mod n$

If $n$ divides $x^2 - y^2$, then $n$ divides $(x - y)(x + y)$, so

$n = \gcd(x - y, n) \cdot \gcd(x + y, n)$

may be a non-trivial factorization (and computing gcd’s is easy)

**How to solve $x^2 \equiv y^2 \mod n$?**

1. Collect integers $v$ such that $v^2 \mod n$
   - ‘satisfies a milder condition than being a square’
   - ‘relation collection’ or ‘sieving’ step
2. Look at the product of some of the $v^2$’s such that
   - ‘the product of the milder conditions is also a square’
   - ‘matrix’ step
   - In theory two steps equally hard
   - In practice:
     - Sieving step takes more time, but anyone can help,
       it’s fault tolerant, just wait until it’s done
     - Matrix step needs large computer, all bits critical

**Example: $n = 143$**

1. Define ‘milder condition than being a square’ as:
   - ‘factor into primes $\leq 5$’

Notice that $143 = 12^2 - 7^2 \mod (12 - 1)(12 + 1) = 11-13$

⇒ collect integers $v$ such that $v^2 \mod 143$ has factors 2, 3, 5 only

Use Dixon’s algorithm: pick $v$’s at random and hope for the best

Pick $v = 17$: $17^2 = 289 = 3 + 2\cdot 143 = 3 \mod 143 = 2^2\cdot 3^1\cdot 5^0$, good!

$18^2 = 3 + 17 + 18 \mod 143 = 38 = 2\cdot 19$, bad

$19^2 = 38 + 18 + 19 \mod 143 = 75 = 2^3\cdot 3^1\cdot 5^2$, good!
Example: $n = 143$

1. Define ‘milder condition than being a square’ as:
   ‘factor into primes $\leq 5$’
   $\Rightarrow$ collect integers $v$ such that $v^2 \mod 143$ has factors 2, 3, 5 only

Use Dixon’s algorithm: pick $v$’s at random and hope for the best

Pick $v = 17$:

$17^2 = 289 = 3 + 2 \cdot 143 = 2^0 \cdot 3^1 \cdot 5^0$, good!  
$18^2 = 3 + 17 + 18 \mod 143 = 38 = 2 \cdot 19$, bad  
$19^2 = 38 + 18 + 19 \mod 143 = 75 = 2^0 \cdot 3^1 \cdot 5^2$, good!

2. Look at exponent vectors $(0,1,0)$ and $(0,1,2)$ of the good ones:
   Their sum is $(0,2,2)$, all even numbers
   $\Rightarrow (17 \cdot 19)^2 \equiv 2^0 \cdot 3^2 \cdot 5^2$
   $\Rightarrow x \equiv 37 - 15 \mod 143$, $y \equiv 20 \cdot 31 \cdot 51 \mod 143$

   $143 = \gcd(37 - 15, 143) \cdot \gcd(37 + 15, 143) = 11 \cdot 13$

Making $v^2 \mod n$ smaller

Random $v$’s: $v^2 \mod n$ has same order magnitude as $n$,  
$\Rightarrow$ How to generate the $v$’s such that $v^2 \mod n$ is smaller?

• Let $a/b$, be $ith$ continued fraction convergent to $\sqrt{n}$:
  $v = a$, $v^2 \mod n = a^2 - nb^2 \equiv 2^\alpha \mod n$: CFRAC

• Small $i,j$: $g(i,j) = (i+j [\sqrt{n}]/(j+i [\sqrt{n}])$: $g(i,j)-n \approx (i+j)/n$  
  $p|g(i,j) \Rightarrow p|(i+k,g,j+k,g)$ is sievable: linear sieve

• To make $g(i,j)$ a square, take $i = j$: quadratic sieve

• $n = f_0 \cdots f_m \cdots \mod \cdots = f(m)$ for some $m \approx n^{1/(d+1)}$,  
  $Q(\alpha) = Q(\alpha)/(f(\alpha))$: $a - bm \mod n$ is sievable modulo $n$

Refinements

• Generate $v$’s such that $v^2 \mod n$ is ‘smaller’  
  (so $v^2 \mod n$ has a higher smoothness probability)

Upto and including QS: residues to be tested are $n^{\Theta(1)}$

Number Field Sieve: residues to be tested are $n^{1/2}$

• Generate $v$’s so they can be tested simultaneously  
  (sieving)

or

• Test smoothness using fast non-sieving method

More in general

1. Define ‘milder condition than being a square’ as:
   ‘factor into first $\pi(B)$ primes, i.e., the primes $\leq B$’
   $\Rightarrow$ collect $v$ such that $v^2 \mod n$ is $B$-smooth’

As soon as set $V$ of good $v$’s satisfies $#V > \pi(B)$:
   exponent vectors linearly dependent modulo 2  
   $\Rightarrow$ a right combination of the $v$’s exists

2. Find dependencies modulo 2 in $#V \times \pi(B)$ matrix,
   each new dependency produces a new pair $x,y$

Making $v^2 \mod n$ smaller

Random $v$’s: $v^2 \mod n$ has same order magnitude as $n$,  
$\Rightarrow$ How to generate the $v$’s such that $v^2 \mod n$ is smaller?

• Let $a/b$, be $ith$ continued fraction convergent to $\sqrt{n}$:
  $v = a$, $v^2 \mod n = a^2 - nb^2 \equiv 2^\alpha \mod n$: CFRAC

• Small $i,j$: $g(i,j) = (i+j [\sqrt{n}]/(j+i [\sqrt{n}])$: $g(i,j)-n \approx (i+j)/n$  
  $p|g(i,j) \Rightarrow p|(i+k,g,j+k,g)$ is sievable: linear sieve

• To make $g(i,j)$ a square, take $i = j$: quadratic sieve

• $n = f_0 \cdots f_m \cdots \mod \cdots = f(m)$ for some $m \approx n^{1/(d+1)}$,  
  $Q(\alpha) = Q(\alpha)/(f(\alpha))$: $a - bm \mod n$ is sievable modulo $n$

factor $a - bm$ in $Z$: $\approx n^{1/(d+1)}$ (small $a, b$), sievable
factor $a - b\alpha$ in $Z[\alpha]$ ‘as’ $b^\alpha(a/b) \approx n^{1/(d+1)}$, sievable
with $d^3 \approx (\log n)/(\log \log n)$ all ‘residues’ $\approx n^{1/2}$: NFS
NFS factorization of 512-bit $n$, 1999

- Two bounds $B_1$ and $B_2$, each about $2^{24}$
- Total number of ‘primes’ about 2 million
- Relation collection about 8 years on 1GHz laptop (or 10 minutes on US$10K$ TWIRL device)
- Due to large primes: matrix about $6.7M \times 6.7M$ with on average about 63 non-zeros per row
- Matrix step in 10 days on Cray C916 (required 2Gigabyte RAM)

Current record 576 bits, soon 640 bits, mostly achieved by throwing more time at it

Factoring conclusion

- Practical factoring impact so far:
  - Bad PR: 512-bit product line discontinued in 1990
  - Despite attempts: no dent in 1024-bit RSA security
  - General purpose factoring is stuck, since 1970, in the Morrison-Brillhart approach
  - Severely running out of steam
  - Needed: entirely new, fresh approach to factoring

- Practical question: does modulus length have to be divisible by 32?