Factoring RSA keys from certified smart cards: Coppersmith in the wild

Daniel J. Bernstein, Yun-An Chang,
Chen-Mou Cheng, Li-Ping Chou,
Nadia Heninger, Tanja Lange,
Nicko van Someren

5 December 2013

Problems with non-randomness

- 2012 Heninger–Durumeric–Wustrow–Halderman (USENIX),
- 2012 Lenstra–Hughes–Augier–Bos–Kleinjung–Wachter (CRYPTO).
- Factored tens of thousands of public keys on the Internet
  ... typically keys for your home router, not for your bank.
- Why? **Many deployed devices shared RSA prime factors.**
- Most common problem: horrifyingly bad interactions between OpenSSL key generation, `/dev/urandom` seeding, entropy sources.
- Typically keys for your home router, not for your bank because those keys are usually generated by special hardware.
- The Heninger team has lots of material online at `http://factorable.net`

Nice followup student projects in data mining

1. Download all certificates of type X; extract RSA keys.
2. Check for common factors.
3. Write report that you've done the work and there are none.

Nice followup student projects in data mining

1. Download all certificates of type X; extract RSA keys.
2. Check for common factors.
3. Write report that you've done the work and there are none.

MOICA: Certificate Authority of MOI (Ministry of the Interior).
In Taiwan all citizens can get a smartcard with signing and encryption ability to

- make transactions with government agencies (property registries, national labor insurance, public safety, and immigration, file personal income taxes, update car registration,

Nice followup student projects in data mining

1. Download all certificates of type X; extract RSA keys.
2. Check for common factors.
3. Write report that you've done the work and there are none.

MOICA: Certificate Authority of MOI (Ministry of the Interior).
In Taiwan all citizens can get a smartcard with signing and encryption ability to

- make transactions with government agencies (property registries, national labor insurance, public safety, and immigration, file personal income taxes, update car registration, file grant applications),

Nice followup student projects in data mining

1. Download all certificates of type X; extract RSA keys.
2. Check for common factors.
3. Write report that you've done the work and there are none.

MOICA: Certificate Authority of MOI (Ministry of the Interior).
In Taiwan all citizens can get a smartcard with signing and encryption ability to

- make transactions with government agencies (property registries, national labor insurance, public safety, and immigration, file personal income taxes, update car registration, file grant applications),
- interact with companies (e.g. Chunghwa Telecom).
- interact with other citizens (encrypt & sign).

Taiwan Citizen Digital Certificate

- Smart cards are issued by the government.
- FIPS-140 and Common Criteria Level 4+ certified.
- RSA keys are generated on card.
- Certificates stored on national LDAP directory. This is publicly accessible to enable citizen-to-citizen and citizen-to-commerce interactions.

Certificate of Chen-Mou Cheng

```
Data: Version: 3 (0x2)
Serial Number: d7:15:33:8e:79:a7:02:11:7d:4f:25:b5:47:e8:ad:38
Signature Algorithm: sha1WithRSAEncryption
Issuer: C=TW, O=XXX
Validity
     Not Before: Feb 24 03:20:49 2012 GMT
     Not After : Feb 24 03:20:49 2017 GMT
Subject: C=TW, CN=YYY serialNumber=0000000112831644
Subject Public Key Info:
     Public Key Algorithm: rsaEncryption
Public-Key: (2048 bit) Modulus:
     00:bf:e7:7c:28:1d:c8:78:a7:13:1f:cd:2b:f7:63:
     2c:89:0a:74:ab:62:c9:1d:7c:62:eb:e8:fc:51:89:
     b3:45:0e:a4:fa:b6:06:de:b3:24:c0:da:43:44:16:
     e5:21:cd:20:f0:58:34:2a:12:f9:89:62:75:e0:55:
     8c:6f:2b:0f:44:c2:06:6c:4c:93:cc:6f:98:e4:4e:
     3a:79:d9:91:87:45:cd:85:8c:33:7f:51:83:39:a6:
     9a:60:98:e5:4a:85:c1:d1:27:bb:1e:b2:b4:e3:86:
     a3:21:cc:4c:36:08:96:90:cb:f4:7e:01:12:16:25:
     90:f2:4d:e4:11:7d:13:17:44:cb:3e:49:4a:f8:a9:
     a0:72:fc:4a:58:0b:66:a0:27:e0:84:eb:3e:f3:5d:
     5f:b4:86:1e:d2:42:a3:0e:96:7c:75:43:6a:34:3d:
     6b:96:4d:ca:f0:de:f2:bf:5c:ac:f6:41:f5:e5:bc:
     fc:95:ee:b1:f9:c1:a8:6c:82:3a:dd:60:ba:24:a1:
     eb:32:54:f7:20:51:e7:c0:95:c2:ed:56:c8:03:31:
     96:c1:b6:6f:b7:4e:c4:18:8f:50:6a:86:1b:a5:99:
     d9:3f:ad:41:00:d4:2b:e4:e7:39:08:55:7a:ff:08:
     30:9e:df:9d:65:e5:0d:13:5c:8d:a6:f8:82:0c:61:
     c8:6b
Exponent: 65537 (0x10001)

          .
          .
          .
```

This project took a slightly different turn

1. Download all certificates of type X; extract RSA keys.
2. Check for common factors.
3. Write report that you've done the work and there are none.

April 2012: downloaded all certificates from LDAP server:

- 2,300,000 1024-bit RSA public keys
- 360,000 2048-bit RSA public keys

This project took a slightly different turn

1. Download all certificates of type X; extract RSA keys.
2. Check for common factors.
3. Write report that you've done the work and there are none.

April 2012: downloaded all certificates from LDAP server:

- 2,300,000 1024-bit RSA public keys
- 360,000 2048-bit RSA public keys

HITCON 2012 (July 20–21):
Prof. Li-Ping Chou presents "Cryptanalysis in real life"
(based on work with Yun-An Chang and Chen-Mou Cheng)

Factored 103 RSA-1024 Taiwan Citizen Digital Certificates

This project took a slightly different turn

1. Download all certificates of type X; extract RSA keys.
2. Check for common factors.
3. Write report that you've done the work and there are none.

April 2012: downloaded all certificates from LDAP server:

- 2,300,000 1024-bit RSA public keys
- 360,000 2048-bit RSA public keys

HITCON 2012 (July 20–21):
Prof. Li-Ping Chou presents "Cryptanalysis in real life"
(based on work with Yun-An Chang and Chen-Mou Cheng)

Factored 103 RSA-1024 Taiwan Citizen Digital Certificates

Wrote report that some keys are factored, informed MOI.

This project took a slightly different turn

1. Download all certificates of type X; extract RSA keys.
2. Check for common factors.
3. Write report that you've done the work and there are none.

April 2012: downloaded all certificates from LDAP server:

- 2,300,000 1024-bit RSA public keys
- 360,000 2048-bit RSA public keys

HITCON 2012 (July 20–21):
Prof. Li-Ping Chou presents "Cryptanalysis in real life"
(based on work with Yun-An Chang and Chen-Mou Cheng)

Factored 103 RSA-1024 Taiwan Citizen Digital Certificates

Wrote report that some keys are factored, informed MOI.

End of story.

This project took a slightly different turn

1. Download all certificates of type X; extract RSA keys.
2. Check for common factors.
3. Write report that you've done the work and there are none.

April 2012: downloaded all certificates from LDAP server:

- 2,300,000 1024-bit RSA public keys
- 360,000 2048-bit RSA public keys

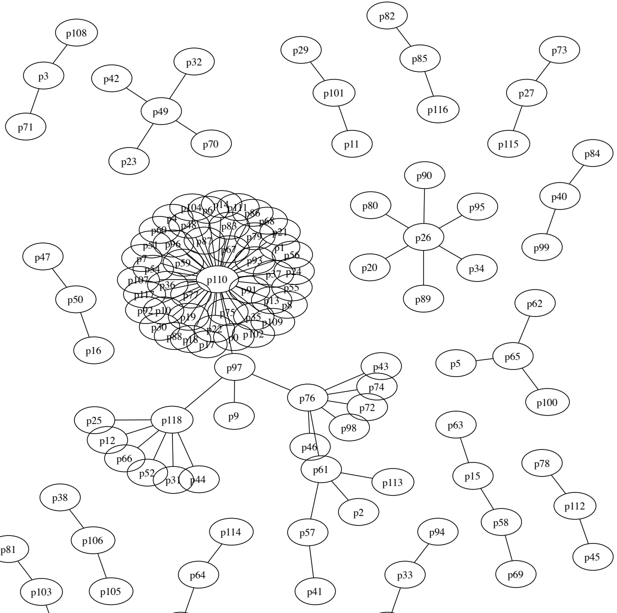
HITCON 2012 (July 20–21):
Prof. Li-Ping Chou presents "Cryptanalysis in real life"
(based on work with Yun-An Chang and Chen-Mou Cheng)

Factored 103 RSA-1024 Taiwan Citizen Digital Certificates

Wrote report that some keys are factored, informed MOI.

End of story?

January 2013: Closer look at the 119 primes



Look at the primes!

Prime factor p110 appears 46 times

```
c0000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
000000000000000000000000000002f9
```

Look at the primes!

Prime factor p110 appears 46 times

```
c0000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
000000000000000000000000000002f9
```

which is the next prime after $2^{511} + 2^{510}$.
The next most common factor, repeated 7 times, is

```
c9242492249292499249492449242492
24929249924949244924249224929249
92494924492424922492924992494924
492424922492924992494924492424e5
```

Several other factors exhibit such a pattern.

How is this pattern generated?

1100100100100100001001001001001000100100100100101001001001001001
1001001001001001010010010010010001001001001001000010010010010010
0010010010010010100100100100100110010010010010010100100100100100
0100100100100100001001001001001000100100100100101001001001001001
1001001001001001010010010010010001001001001001000010010010010010
0010010010010010100100100100100110010010010010010100100100100100
0100100100100100001001001001001000100100100100101001001001001001
1001001001001001010010010010010001001001001001000010010011100101

How is this pattern generated?

Swap every 16 bits in a 32 bit word

0010010010010010 1100100100100100 1001001001001001 0010010010010010
0100100100100100 1001001001001001 0010010010010010 0100100100100100
1001001001001001 0010010010010010 0100100100100100 1001001001001001
0010010010010010 0100100100100100 1001001001001001 0010010010010010
0100100100100100 1001001001001001 0010010010010010 0100100100100100
1001001001001001 0010010010010010 0100100100100100 1001001001001001
0010010010010010 0100100100100100 1001001001001001 0010010010010010
0100100100100100 1001001001001001 0010010011100101 0100100100100100

How is this pattern generated?

Realign

0010010010010010**11**001001001001001001001001001001001001001001
001001001001001001001001001001001001001001001001001001001001
001001001001001001001001001001001001001001001001001001001001
001001001001001001001001001001001001001001001001001001001001
001001001001001001001001001001001001001001001001001001001001
001001001001001001001001001001001001001001001001001001001001
001001001001001001001001001001001001001001001001001001001001
001001001001001001001001**1100101**0100100100100100

How is this pattern generated?

Realign

0010010010010010110010010010010010010010010010010010010010010
001001001001001001001001001001001001001001001001001001001001
001001001001001001001001001001001001001001001001001001001001
001001001001001001001001001001001001001001001001001001001001
001001001001001001001001001001001001001001001001001001001001
001001001001001001001001001001001001001001001001001001001001
001001001001001001001001001001001001001001001001001001001001
00100100100100100100100100111001010100100100100100

The 119 factors had patterns of period 1,3,5, and 7.

Prime generation

1. Choose a bit pattern of length 1, 3, 5, or 7 bits, repeat it to cover more than 512 bits, and truncate to exactly 512 bits.
2. For every 32-bit word, swap the lower and upper 16 bits.
3. Fix the most significant two bits to 11.
4. Find the next prime greater than or equal to this number.

Factoring by trial division

1. Choose a bit pattern of length 1, 3, 5, or 7 bits, repeat it to cover more than 512 bits, and truncate to exactly 512 bits.
2. For every 32-bit word, swap the lower and upper 16 bits.
3. Fix the most significant two bits to 11.
4. Find the next prime greater than or equal to this number.

Do this for any pattern:
0,1,001,010,011,100,101,110
00001,00010,00011,00100,00101,0011,00111,01000,01001,01010,. . .
00000001,0000011,0000101,0000111,0001001,. . .

Factoring by trial division

1. Choose a bit pattern of length 1, 3, 5, or 7 bits, repeat it to cover more than 512 bits, and truncate to exactly 512 bits.
2. For every 32-bit word, swap the lower and upper 16 bits.
3. Fix the most significant two bits to 11.
4. Find the next prime greater than or equal to this number.

Do this for any pattern:
0,1,001,010,011,100,101,110
00001,00010,00011,00100,00101,0011,00111,01000,01001,01010,. . .
00000001,0000011,0000101,0000111,0001001,. . .
Computing GCDs factored 105 moduli, of which 18 were new.

Factoring by trial division

1. Choose a bit pattern of length 1, 3, 5, or 7 bits, repeat it to cover more than 512 bits, and truncate to exactly 512 bits.
2. For every 32-bit word, swap the lower and upper 16 bits.
3. Fix the most significant two bits to 11.
4. Find the next prime greater than or equal to this number.

Do this for any pattern:
0,1,001,010,011,100,101,110
00001,00010,00011,00100,00101,0011,00111,01000,01001,01010,. . .
00000001,0000011,0000101,0000111,0001001,. . .
Computing GCDs factored 105 moduli, of which 18 were new.
Factored 4 more keys using patterns of length 9.

Factoring by trial division

1. Choose a bit pattern of length 1, 3, 5, or 7 bits, repeat it to cover more than 512 bits, and truncate to exactly 512 bits.
2. For every 32-bit word, swap the lower and upper 16 bits.
3. Fix the most significant two bits to 11.
4. Find the next prime greater than or equal to this number.

Do this for any pattern:
0,1,001,010,011,100,101,110
00001,00010,00011,00100,00101,0011,00111,01000,01001,01010,. . .
00000001,0000011,0000101,0000111,0001001,. . .
Computing GCDs factored 105 moduli, of which 18 were new.
Factored 4 more keys using patterns of length 9.
Second factors in moduli are also interesting . . .

Some more prime factors

```
c0000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
000000000000000000000000000101ff

c0000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000100000177
```

Some more prime factors

```
c0000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
000000000000000000000000000101ff

c0000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000100000177
```

Hypothesis: There might be more prime factors of the form

$$p = 2^{511} + 2^{510} + x$$

where x is "small".

Theorem (Coppersmith)

In polynomial time we can find the factorization of $N = pq$ *if we know the high-order* $\frac{1}{4}\log_2 N$ *bits of* p*.*

Theorem (Coppersmith)

In polynomial time we can find the factorization of $N = pq$ if we know the high-order $\frac{1}{4}\log_2 N$ bits of p.

Algorithm (Howgrave-Graham)

1. *Input $a =$ the top half of bits of p. We want r satisfying*

$$a + r = p$$

   r is a solution to the equation

$$f(x) = a + x \equiv 0 \bmod p$$

2. *Construct a lattice L of coefficients of multiples of $a + x, N$. A short vector in L corresponds to an equation Q satisfying*

$$Q(r) = 0$$

3. *Solve Q over $\mathbb{Z}$ to find r.*


LLL ALL THE
KEYS!

Factoring with Coppersmith/Howgrave-Graham

1. For all patterns a and moduli N, run LLL on

$$\begin{bmatrix} X^2 & Xa & 0 \\ 0 & X & a \\ 0 & 0 & N \end{bmatrix}$$

   to obtain a short vector $|v_1| = (X^2 q_2, X q_1, q_0)$.

2. Compute roots r_1, r_2 of $Q(x) = q_2 x^2 + q_1 x + q_0$.

3. Check if $\gcd(a + r_1, N)$ or $\gcd(a + r_2, N)$ nontrivial.

Factoring with Coppersmith/Howgrave-Graham

1. For all patterns a and moduli N, run LLL on

$$\begin{bmatrix} X^2 & Xa & 0 \\ 0 & X & a \\ 0 & 0 & N \end{bmatrix}$$

   to obtain a short vector $|v_1| = (X^2 q_2, X q_1, q_0)$.

2. Compute roots r_1, r_2 of $Q(x) = q_2 x^2 + q_1 x + q_0$.
3. Check if $\gcd(a + r_1, N)$ or $\gcd(a + r_2, N)$ nontrivial.

- Works when $r < 2^{-1/2} N^{1/6}$.
- For 1024-bit N, r as large as 170 bits.
- Factored **39 new keys** in 160 hours of computation time.

```
ffffaa55ffffffffff3cd9fe3ffff676
fffffffffffe00000000000000000000
00000000000000000000000000000000
000000000000000000000000000000009d
```

```
c000b800000000000000000000000000
00000000000000000000000000000000
00000680000000000000000000000000
00000000000000000000000000000251
```

Factoring with Bivariate Coppersmith

Search for prime factors of the form

$$p = a + 2^t x + y$$

Factoring with Bivariate Coppersmith

Search for prime factors of the form

$$p = a + 2^t x + y$$

Algorithm (Expected Algorithm)

1. *Generate lattice from multiples of* $f(x, y) = a + 2^t x + y$, N.
2. *Run LLL and take two short polynomials* $Q_1(x, y)$, $Q_2(x, y)$.
3. *Solve for* r_1, r_2 *satisfying* $Q_1(r_1, r_2) = Q_2(r_1, r_2) = 0$.
4. *Check if* $\gcd(a + 2^t r_1 + r_2, N)$ *is nontrivial.*

Factoring with Bivariate Coppersmith

Search for prime factors of the form

$$p = a + 2^t x + y$$

Algorithm (Expected Algorithm)

1. *Generate lattice from multiples of* $f(x, y) = a + 2^t x + y$*,* N*.*
2. *Run LLL and take two short polynomials* $Q_1(x, y)$*,* $Q_2(x, y)$*.*
3. *Solve for* r_1, r_2 *satisfying* $Q_1(r_1, r_2) = Q_2(r_1, r_2) = 0$*.*
4. *Check if* $\gcd(a + 2^t r_1 + r_2, N)$ *is nontrivial.*

- Analysis says 10-dimensional lattices let us solve for

$$|r_1 r_2| < N^{1/10}.$$

- For 1024-bit N, should have $|r_1 r_2| < 2^{102}$.

Tricky Details: Algebraic Dependence

- Need *two* equations $Q_1(x, y)$, $Q_2(x, y)$.
- Coefficient vectors in lattice are linearly independent, but polynomials might have algebraic relation.

Tricky Details: Algebraic Dependence

- Need *two* equations $Q_1(x, y)$, $Q_2(x, y)$.
- Coefficient vectors in lattice are linearly independent, but polynomials might have algebraic relation.

Standard Heuristic Assumption

The short vectors of the LLL-reduced basis correspond to algebraically independent polynomials.

Tricky Details: Algebraic Dependence

- Need *two* equations $Q_1(x, y)$, $Q_2(x, y)$.
- Coefficient vectors in lattice are linearly independent, but polynomials might have algebraic relation.

Standard Heuristic Assumption

The short vectors of the LLL-reduced basis correspond to algebraically independent polynomials.

This assumption **failed** in our experiments.

Tricky Details: Algebraic Dependence

- Need *two* equations $Q_1(x, y)$, $Q_2(x, y)$.
- Coefficient vectors in lattice are linearly independent, but polynomials might have algebraic relation.

Standard Heuristic Assumption

The short vectors of the LLL-reduced basis correspond to algebraically independent polynomials.

This assumption **failed** in our experiments.

- In most cases polynomials shared linear common factors

$$q_1 x + q_2 y + q_3 = 0$$

and thus had infinitely many potential solutions.

Tricky Details: Algebraic Dependence

- Need *two* equations $Q_1(x, y)$, $Q_2(x, y)$.
- Coefficient vectors in lattice are linearly independent, but polynomials might have algebraic relation.

Standard Heuristic Assumption

The short vectors of the LLL-reduced basis correspond to algebraically independent polynomials.

This assumption **failed** in our experiments.

- In most cases polynomials shared linear common factors

$$q_1 x + q_2 y + q_3 = 0$$

  and thus had infinitely many potential solutions.
- By experimenting, we learned that the *smallest* solution seemed to work.

Tricky Details: Theory vs. Practice

Solution Sizes

- Standard analysis told us algorithm should work with lattice dimension ≥ 10.
- But in practice lattice dimension 6 worked!

Patterns

- When we experimented with pattern

  $$x000\ldots000y$$

  method also found factors of form

  $$x9924\ldots4929y$$

  and other repeating patterns!

Experimental Results

dim	XY	offsets	patterns	keys factored	running time
6	2^4	5	1	104	4.3 hours
6	2^4	1	164	154	195 hours
10	2^{100}	1	1	112	2 hours
15	2^{128}	5	1	108	20 hours

11 additional keys factored.

Why are government-issued smartcards generating weak keys?

Card behavior very clearly not FIPS-compliant.

Why are government-issued smartcards generating weak keys?

Card behavior very clearly not FIPS-compliant.

Hypothesized failure:

- Hardware RNG has underlying weakness that causes failure in some situations.
- Card software not operated in FIPS mode
  $\Longrightarrow$ no testing or post-processing RNG output.

Disclosure and Response

- Disclosure to Taiwanese government in April 2012, June 2013.
- July 2012: MOICA replaced cards for GCD vulnerable certificates.
- July 2013: MOICA told us they planned to replace full "bad batch" of cards.

Disclosure and Response

August 2013: From Email to Research Team

"It took more effort than we expected to locate the affected cards. . . Now, we believe that have revoked all the problematic certificates we found and informed those affected cards holder to replace their cards. Furthermore, we are now implementing the coppersmith method based on your paper to double confirm that there are no any affected cards slipped away."

Disclosure and Response

August 2013: From Email to Research Team

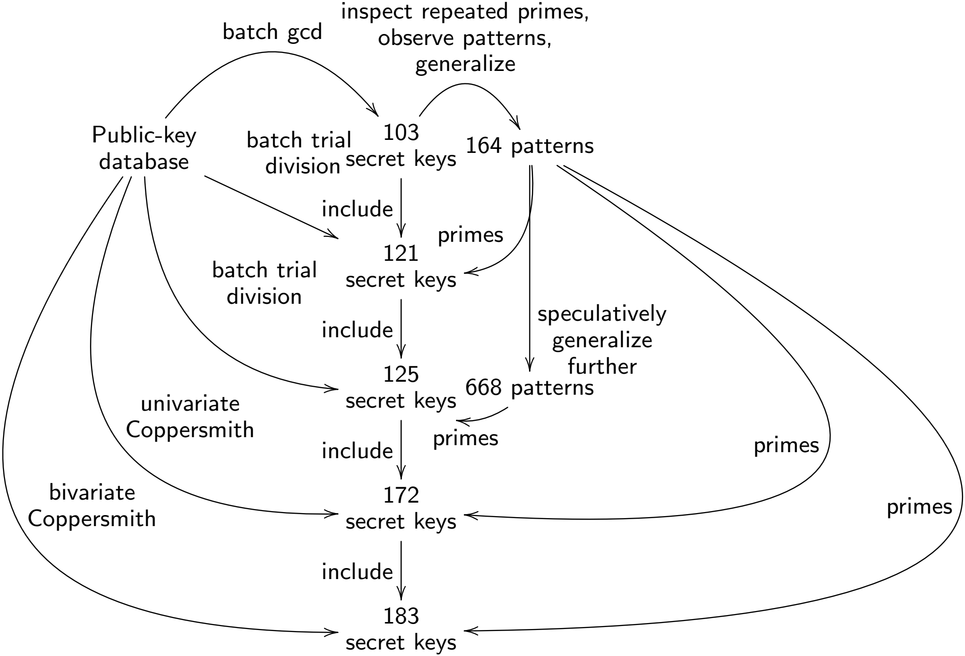
"It took more effort than we expected to locate the affected cards. . . Now, we believe that have revoked all the problematic certificates we found and informed those affected cards holder to replace their cards. Furthermore, we are now implementing the coppersmith method based on your paper to double confirm that there are no any affected cards slipped away."

September 2013: Public Press Release (In Chinese)

"Regarding the internet news about CDC weak keys and how we have dealt with this problem. . . the paper cited in the news is a result of government sponsored research. . . As a result, we have replaced all vulnerable cards in July 2012. . . So all the keys used now are safe."

Lessons

- Certification doesn't protect against usage errors.
- Hardware RNGs still need to be tested and post-processed.
- Nontrivial GCD is not the only way RSA can fail with bad RNG.



batch gcd
inspect repeated primes, observe patterns, generalize
Public-key database
batch trial division
103 secret keys
164 patterns
include
primes
121 secret keys
batch trial division
include
speculatively generalize further
125 secret keys
668 patterns
univariate Coppersmith
primes
include
primes
bivariate Coppersmith
172 secret keys
primes
include
183 secret keys