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

September 16, 2013

Problems with non-randomness

- 2012 Heninger–Durumeric–Wustrow–Halderman,
- 2012 Lenstra–Hughes–Augier–Bos–Kleinjung–Wachter.
- 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 prime factors.**
- Most common problem: horrifyingly bad interactions between OpenSSL key generation, `/dev/urandom` seeding, entropy sources.
- 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.

This started as such a student project on a very nice system: MOICA: Certificate Authoritiy of MOI (Ministry of the Interior). In Taiwan all citizens can get a smartcard with signing and encryption ability to

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

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.
- About 3,002,000 certificates (all using RSA keys) 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

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 Taiwan Citizen Digital Certificates
(out of 2.26 million keys with 1024 bits).

This project took a slightly different turn

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 Taiwan Citizen Digital Certificates
(out of 2.26 million keys with 1024 bits).

Wrote report that some keys are factored, informed MOI.

This project took a slightly different turn

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 Taiwan Citizen Digital Certificates
(out of 2.26 million keys with 1024 bits).

Wrote report that some keys are factored, informed MOI.

End of story.

This project took a slightly different turn

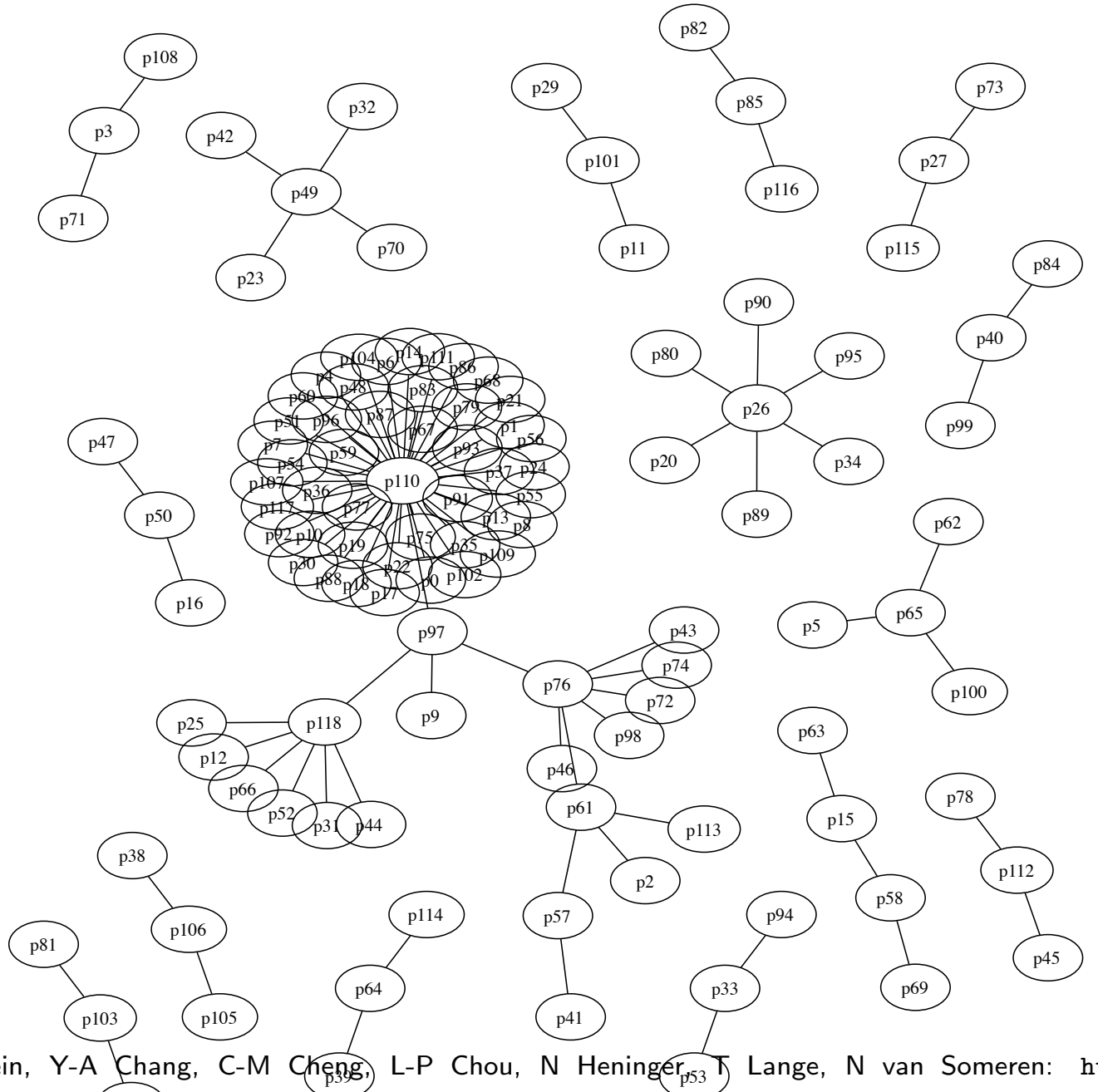
HITCON 2012 (July 20–21):
Prof. Li-Ping Chou presents "Cryptanalysis in real life"
(based on work with Yun-An Chang and Chen-Mou Cheng)

Factored 103 Taiwan Citizen Digital Certificates
(out of 2.26 million keys with 1024 bits).

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

```
c000000000000000000000000000000
000000000000000000000000000000
000000000000000000000000000000
0000000000000000000000000002f9
```

Look at the primes!

Prime factor p110 appears 46 times

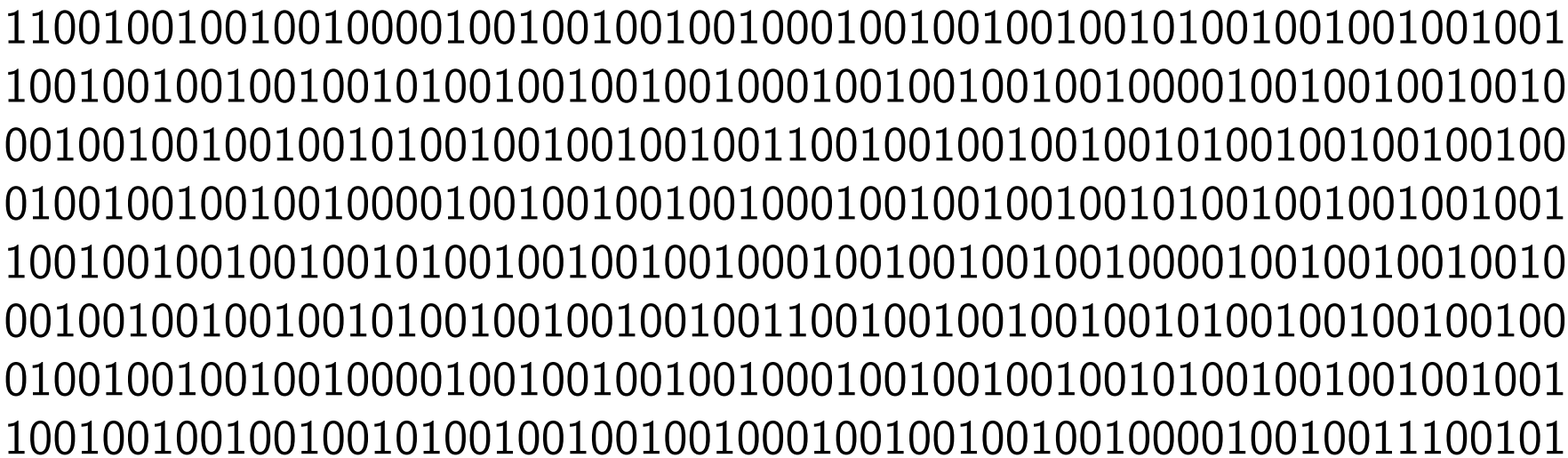
c000000000000000000000000000000
000000000000000000000000000000
000000000000000000000000000000
0000000000000000000000000002f9

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

c924249224929249924949244924249 2
249292499249492449242492249292 49
924949244924249224929249924949 24
492424922492924992494924492424e5

Several other factors exhibit such a pattern.

How is this pattern generated?



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

00100100100100101100100100100100100100100100100100100100100
00100100100100100100100100100100100100100100100100100100100
00100100100100100100100100100100100100100100100100100100100
00100100100100100100100100100100100100100100100100100100100
00100100100100100100100100100100100100100100100100100100100
00100100100100100100100100100100100100100100100100100100100
00100100100100100100100100100100100100100100100100100100100
00100100100100100100100100100100100100100100100100100100100
00100100100100100100100100110010101001001001001001001001001

How is this pattern generated?

Realign

00100100100100101100100100100100100100100100100100100100100100
00100100100100100100100100100100100100100100100100100100100100
00100100100100100100100100100100100100100100100100100100100100
00100100100100100100100100100100100100100100100100100100100100
00100100100100100100100100100100100100100100100100100100100100
00100100100100100100100100100100100100100100100100100100100100
00100100100100100100100100100100100100100100100100100100100100
00100100100100100100100100100100100100100100100100100100100100
00100100100100100100100100110010101001001001001001001001001001

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.

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.

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

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,...

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

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.

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

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.

Patterns do not find all factors

These primes

c000000000000000000000000000000
000000000000000000000000000000
000000000000000000000000000000
00000000000000000000000002030b

c000000000000000000000000000000
000000000000000000000000000000
000000000000000000000000000000
00000000000000000000100000177

were found via GCDs, but not from the patterns.

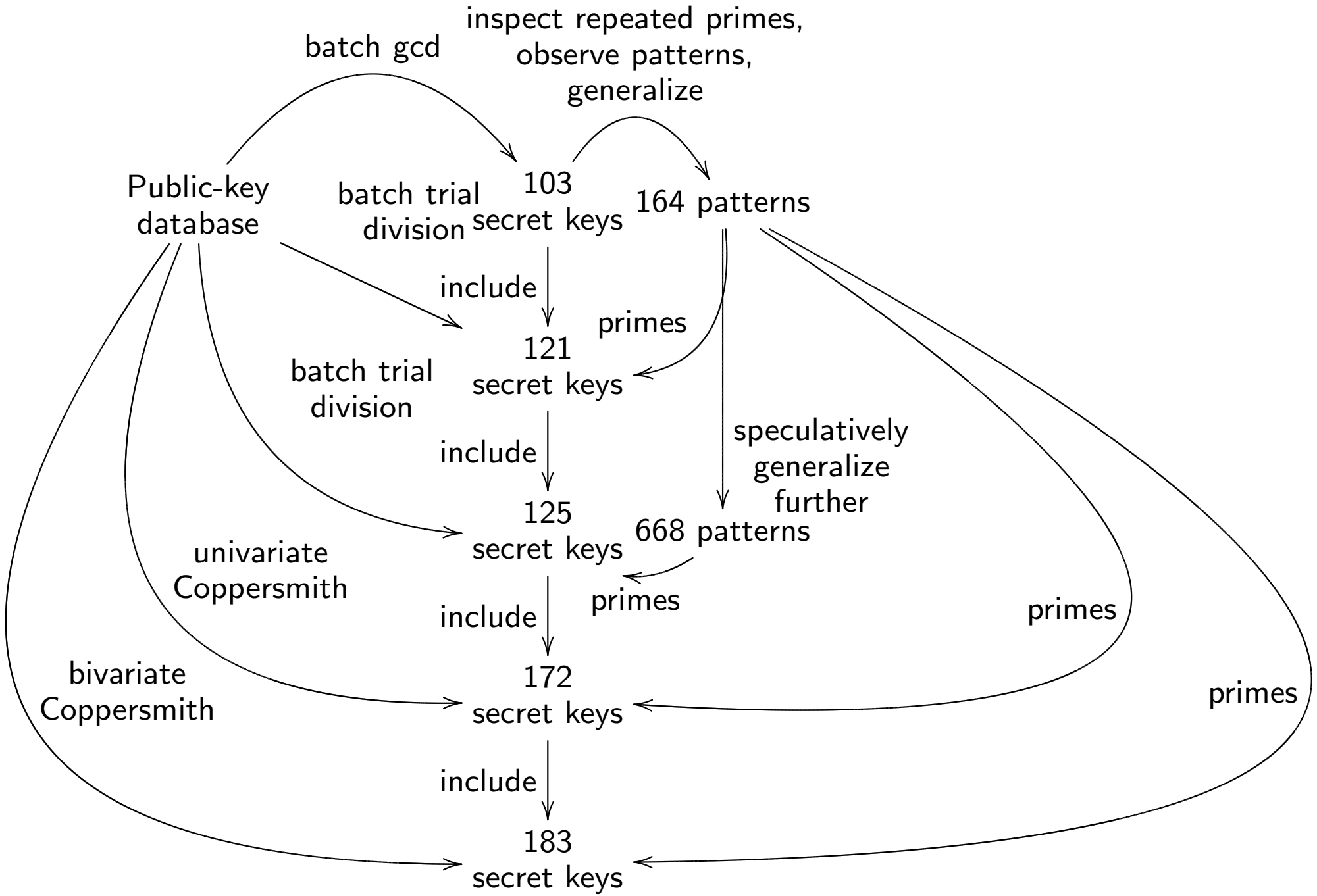
Patterns do not find all factors

These primes

c000000000000000000000000000000
000000000000000000000000000000
000000000000000000000000000000
00000000000000000000000002030b

c000000000000000000000000000000
000000000000000000000000000000
000000000000000000000000000000
00000000000000000000100000177

were found via GCDs, but not from the patterns.
Looks like base pattern 0 with some bits flipped.


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

Why are government-issued smartcards generating weak keys?

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 ring oscillator gets stuck in some conditions or does not output quickly enough.
- Card software not post-processing RNG output.

Important Lesson:

- Nontrivial GCD is not the only way RSA can fail with bad RNG.

Future work:

- Breaking RSA-1024 with Fermat factoring.

Future work:

- Breaking RSA-1024 with Fermat factoring.
- Breaking RSA-1024 using Adi Shamir's secret database of all primes.

Future work:

- Breaking RSA-1024 with Fermat factoring.
- Breaking RSA-1024 using Adi Shamir's secret database of all primes.
- Breaking RSA-1024 using $1024 = 2 * 2 * 2 * 2 * 2 * 2 * 2 * 2 * 2 * 2$.

Future work:

- Breaking RSA-1024 with Fermat factoring.
- Breaking RSA-1024 using Adi Shamir's secret database of all primes.
- Breaking RSA-1024 using $1024 = 2 * 2 * 2 * 2 * 2 * 2 * 2 * 2 * 2 * 2$.
- Breaking RSA-1024 using Intel's new RDRAND_NSAKEY instruction.