Ron was Wrong, Whit is Right

Sanity checks on the Internet PKI

Arjen K. Lenstra [1] James P. Hughes [2]
Maxime Augier [1] Joppe Bos [1] Thorsten Kleinjung [1]
Christophe Wachter [1]

[1] Ecole Polytechnique Federale de Lausanne, LACAL, Station 14, CH-1015 Lausanne

[2] Self, Palo Alto, CA, USA

April 5, 2012

Project goal

- Public key cryptography is widely used and subject to much scrutiny.
- But what about the implementations, and operational conditions ?
- Public keys are public. Let's collect as many as possible and perform some sanity checks.

Key sources

- PGP Keyservers: 5.4 M keys
- EFF SSL Observatory: 6.2M, then 7.2M keys (two datasets)
- Various other X509 collection projects
- Missing: SSH

Key statistics

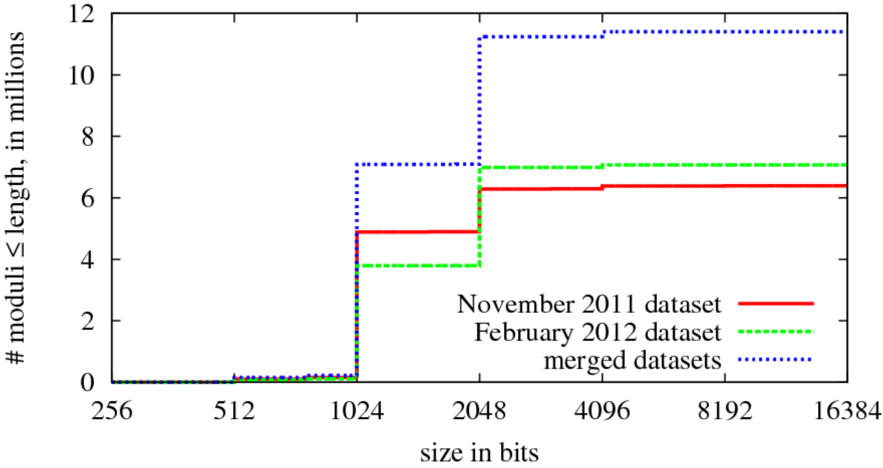
- Total *individual* PGP keys and subkeys collected: 5'481'332

ElGamal	2'546'752
DSA	2'536'959
RSA	397'621

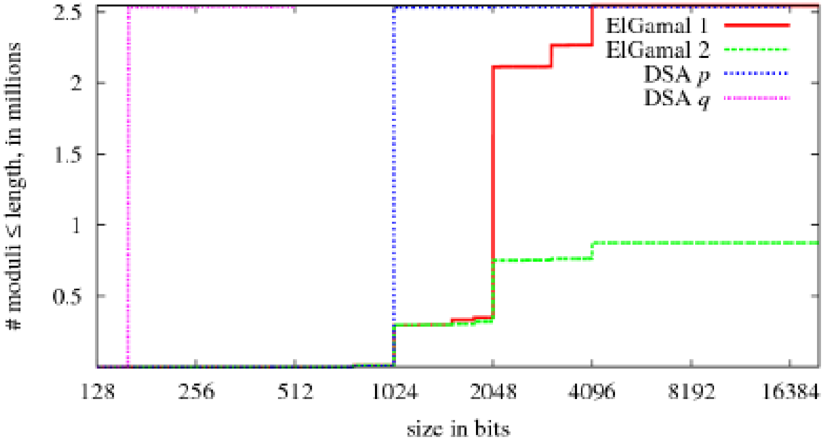
- Total distinct x509 keys: 6'185'372

RSA	6M
DSA	141
ECDSA	1

RSA Moduli sizes



ElGamal/DSA sizes



X509 Hash algorithms

SHA1 *	4.8M
MD5	1.35M
SHA256 *	5.3k
SHA512 *	525
MD2	122
GOST	30
SHA384 *	24
MD4	14
RIPEMD160	9

- 47.6%: Expiration date later than 2011
- 33.4%: above + SHA1 or better

1. Key Collection
2. RSA
   - Exponents
   - Moduli
   - Mutually Factorable keys
3. ElGamal, DSA, ECDSA
   - ElGamal
   - DSA
   - ECDSA
4. Conclusion

RSA reminder

- Pick p,q primes
- Compute $n = pq$
- Compute $\phi(n) = (p-1)(q-1)$
- Pick e coprime with $\phi(n)$
- Compute d $|ed \equiv 1 \pmod{\phi(n)}$
- $c \equiv m^e \pmod{n}, m' \equiv c^d \pmod{n}$
- $m' \equiv m^{ed} \equiv m^1 \pmod{n}$

What would obviously be wrong ?

- $e = 1$: ROT26
- e even: extremely hard to decrypt
- Suspiciously large/random e

What would obviously be wrong ?

- $e = 1$: ROT26 (8 occurrences)
- e even: extremely hard to decrypt (2 occurrences)
- Suspiciously large/random e (2 occurrences)

How did this happen ?

```
int_to_bignum(int e) {
        BIGNUM *big = BN_new();
        for (i=0; i < sizeof(exp); ++i)
                if (exp & (1 << i))
                        BN_set_bit(big, i);
```

How did this happen ?

- Small e are safe if m is properly padded
- Special small e means fast encryption/verification

How did this happen ?

- Small e are safe if m is properly padded
- Special small e means fast encryption/verification
- e and d are interchangeable in key generation

How did this happen ?

- Small e are safe if m is properly padded
- Special small e means fast encryption/verification
- e and d are interchangeable in key generation
- Hey, let's pick a special d to make decryption/signing fast !

Exponents distribution

X.509		PGP	
e	%	e	%
65537	98.49	65537	48.85
17	0.76	17	39.5
3	0.38	41	7.57
35	0.14	19	2.48
5	0.12	257	0.39
other	0.1	other	0.6

Duplicate moduli

We look for identical moduli in different certificates

- 4.3% (240k) of the X.509 certificates have non-unique moduli
- Among these, 30k (0.5%) Debian moduli
- Legitimate cases: same owner with different expiration dates
- Sometimes no obvious relation between them

Duplicate moduli

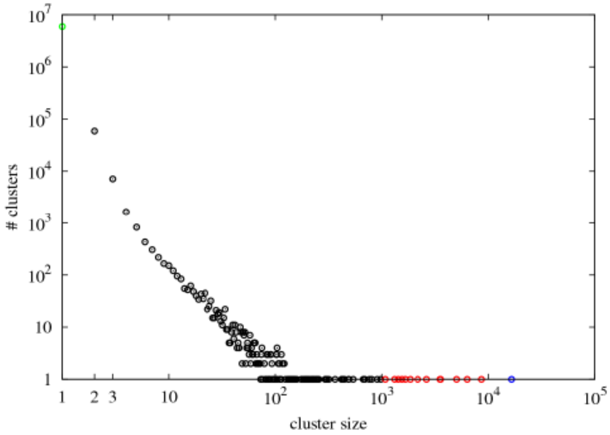
We look for identical moduli in different certificates

- 4.3% (240k) of the X.509 certificates have non-unique moduli
- Among these, 30k (0.5%) Debian moduli
- Legitimate cases: same owner with different expiration dates
- Sometimes no obvious relation between them

And in PGP keys

- 59 non-unique moduli

Moduli clusters



Broken moduli

Simple checks on the moduli:

- Prime n: $\phi(n) = n - 1$
- Small factors in n
  - Even n
  - Copy-paste mistakes
- Fermat method

Broken moduli

Simple checks on the moduli:

- Prime n: $\phi(n) = n - 1$ (2 occurrences)
- Small factors in n (171 occurrences)
  - Even n (68 occurrences)
  - Copy-paste mistakes (9 occurrences)
- Fermat method (0 occurrences)

Actual fun stuff

- User 1 picks $n_1 = p_1 q_1$
- User 2 picks $n_2 = p_2 q_2$

Actual fun stuff

- User 1 picks $n_1 = p_1 q_1$
- User 2 picks $n_2 = p_2 q_2$
- What if $p_1 = p_2 = p$?

Actual fun stuff

- User 1 picks $n_1 = p_1 q_1$
- User 2 picks $n_2 = p_2 q_2$
- What if $p_1 = p_2 = p$?
- Anyone can compute $gcd(n_1, n_2) = p$, and recover p, q_1, q_2.
- Thus, these moduli offer no security.

Checking for colliding keys

- GCD is fast ($O(N \log^2 N)$).
- However, colliding keys requires to check pairs. Naively doing so ($O(n^2)$) is unworkable.
- However, well-known solutions exist. We used a LCM-tree while checking each node for a non-trivial GCD, which runs close to $O(n \log n)$.

Collision shapes

Consider the undirected graph where vertices are prime numbers, and edges are existing keys.

- Ideally, this would be a forest of disconnected edges.
- In practice: 1995 components of more than 1 edge (1st run).
- 1988 are trees of depth one (single magic prime per tree)
- Total 27k/31k total (0.4%) keys compromised

How did this happen ?

```
seed_my_rng();
PRIME p = random_prime();
PRIME q = random_prime();
```

How did this happen ?

```
seed_my_rng();
PRIME p = random_prime();
PRIME q = random_prime();

void seed_my_rng() {
    state = 4; //chosen by fair dice roll
}
```

How did this happen ?

```
seed_my_rng();
PRIME p = random_prime();
//just to be safe
seed_my_rng_again()
PRIME q = random_prime();
```

How did this happen ?

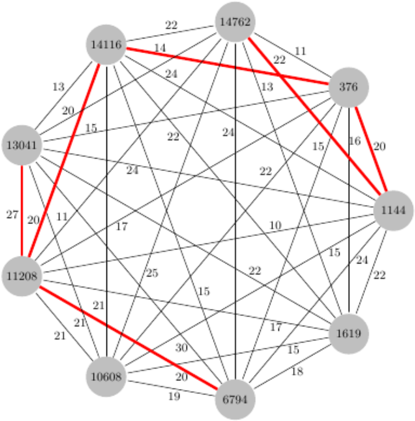
```
PRIME random_prime() {
    PRIME p;
    do {
        p = random_number();
    } while (!prime(p)); // I Feel Lucky !
    return p;
}
```

Suspect 1: Network device manufacturer X

- Bunch of X509 self-signed certificates
- Suspiciously similar generation date, Jan 01 20XX 00:00:0Y
- A single magic p in common for several thousand certs
- Duplicate keys on seemingly unrelated devices. *The moduli are correlated with the generation times*

Suspect 2: Embedded management device Y

- Fully connected set of *9* distinct primes
- Each modulus occurs in many keys with unrelated owners.



Disclosure

"Why not set up an online service to check weak keys ?"

Disclosure

"Why not set up an online service to check weak keys ?"

Disclosure

"Why not set up an online service to check weak keys ?"

Disclosure

"Why not set up an online service to check weak keys ?"

- A key may be flawed and we won't have any idea until another key comes along.
- Telling that a key is weak implicitely compromises another one.
- Our data sources are public. Allowing anyone to test the keys would let an attacker filter out the weak keys and massively speed up the attack.

1. Key Collection
2. RSA
   - Exponents
   - Moduli
   - Mutually Factorable keys
3. ElGamal, DSA, ECDSA
   - ElGamal
   - DSA
   - ECDSA
4. Conclusion

ElGamal reminder

- Pick a prime p
- Pick $g \in (\mathbb{Z}/p\mathbb{Z})^*$
- Pick x
- Compute $y \equiv g^x \pmod{p}$

Basic checks

Total keys: 2.55M

- p not prime: 82 occurrences
- p not safe prime: 34.4%
- g does not generate $(\mathbb{Z}/p\mathbb{Z})^*$: at least 16.4%
- $y \notin \langle g \rangle$: 33 occurrences
- a few suspicious y values

DSA reminder

- Pick primes p,q with $q|(p-1)$
- Pick g such that $\langle g \rangle$ is of order q.
- Pick secret key x $0 \leq x < q$
- Compute $y \equiv g^y \pmod{p}$

Basic tests

- p not prime
- q not prime
- q does not divide $p-1$
- g not of order q
- y not of order q
- $x < 2^{12}$
- Copy-paste errors again ?

Basic tests

- p not prime (12 occurrences)
- q not prime (2 occurrences)
- q does not divide $p-1$ (10 occurrences)
- g not of order q (no occurrences)
- y not of order q (42 occurrences)
- $x < 2^{12}$ (no occurrences)
- Copy-paste errors again ?

ECDSA

- Only one key (no ssh keys yet)
- Only a dozen of signatures

Conclusions

- Most of the observed PKI appears to be working as intended
- 20k RSA moduli have been factored in 30k certs (0.5% of the collection)
- Requiring multiple secrets may have non-obvious pitfalls (Ron vs Whit thing)
- Can we trust nonce randomness where it is critical ?

Conclusions

- Most of the observed PKI appears to be working as intended
- 20k RSA moduli have been factored in 30k certs (0.5% of the collection)
- Requiring multiple secrets may have non-obvious pitfalls (Ron vs Whit thing)
- Can we trust nonce randomness where it is critical ?

and NOT (as it has been misreported):

- That this makes RSA itself broken or inferior
- That electronic commerce will collapse