X.509 certificates

Exploit on badly generated random numbers

X.509 certificates

Binds an entity to a public key using a digital signature

Used in HTTPS, Email certificates, Digital signature, SSH keys

Made using a pair of key (Private-Public), a.k.a PKI

X.509 PKI usage

What one key does		The other validates
Public key		Private key
Encrypts	⟷	Decrypts
Verifies	⟷	Signs

RSA (Rivest–Shamir–Adleman)

- Can be used in X509
- Allow for message signature, encryption and decryption

RSA Operation

Choose two prime number *(p, q)* (61, 53)

Calculate *(n = p X q)* (3233 = 61 X 53)

Choose a coprime number *(**e** where 1 < **e** < lcm*(p-1, q-1))*
(1 < **17** < lcm(60, 52))

Calculate *(**d** where 1 = (e X **d**) % lcm(p-1, q-1))*
(1 = (17 X **413**) % 780)

Tips

Use Python shell or Google to calculate

*lcm = Least Common Multiple

- Public key
  - n = 3233
  - e = 17
- Private key
  - n = 3233
  - d = 413
- Note: You can calculate everything with *p*, *q* and *e*

Focus on what matters for us

- For fast encryption-decryption the chosen exponent e is often 0x10001 or 65537
- p and q are random prime numbers and are **secret**
- n is the product of p and q and **public**
- Knowing p and q, we cracked the certificate
- It is almost impossible to compute p, q knowing n

What if *p* and *q* are not so random

- Let's choose small number for our small brains
  - *n1* = 77 ; *n2* = 35
  - Is there a way to find *p* or *q* knowing *n1* and *n2* ?

Yes! Calculate the greatest common divisor of them:

- gcd(*n1, n2)* = **7**

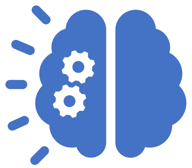
Knowing this, we assume the following

- *n1 = 7 X q ; n2 = 7 X q*
- 77 = 7 X q ⬄ q = 77 / 7 ⬄ q = 11 => **p = 7 ; q = 11**
- 35 = 7 X q ⬄ q = 35 / 7 ⬄ q = 5 => **p = 7 ; q = 5**

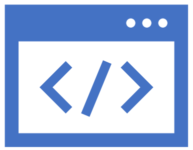
25 first prime numbers:

2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97

What to do now?



Now, we understand the flaw of having bad random prime numbers in RSA-based certificates



We can now write a program that will try to find the GCD between the *n* of a list of certificates!

1. Build a set with all *n* found in the certificates (*S*).
2. Generate all combinations of the set *S*.
3. Compute the GCD of all combinations.
4. If the GCD is greater than 1, you found a common factor.
5. Compute *q*, knowing *p* is the GCD.

Sources

- https://en.wikipedia.org/wiki/X.509#History_and_usage
- https://en.wikipedia.org/wiki/RSA_(cryptosystem)
- https://sectigo.com/resource-library/what-is-x509-certificate
- https://stackoverflow.com/questions/6098381/what-are-common-rsa-sign-exponent
- Arjen K. Lenstra1, James P. Hughes2, Maxime Augier1, Joppe W. Bos1, Thorsten Kleinjung1, and Christophe Wachter1: Ron was wrong, Whit is right