Stratum 0

Hackerspace Braunschweig

- RSS

Search

Navigate…

- Blog
- Archives
- Writeups
- Calendar
- Wiki

Hack.lu 2013: Marvin is plain-Jane

Oct 26th, 2013 by comawill & tsuro & spq

Hey mister super-duper robo-dabster. We need you to tell us, what **Marvin is**!

What we know:

Marvin is
using brainpool p256r1.
His friend is called meneze or something. Or was it van-stone?

What we heard:

(23372093078317551665216159139784413411806753229249201681647388827754827452856
: 1)
71164450240897430648972143714791734771985061339722673162401654668605658194656
12951693517100633909800921421096074083332346613461419370069191654560064909824
What we need to know:

What **Marvin is**

The description made it very clear that the used crypto algorithm is Menezes-Vanstone. And if you are a bit into elliptic curve cryptography it's obvious that if you know one part of the plain text (especially x_1) you are able to calculate the other one.

1. you know y_1 and x_1
2. therefore you know $c_1 = y_1 \cdot x_1^{-1}$
3. with c_1 you are able to calculate both possible values for c_2
4. with c_2 you know $x_2 = y_2 \cdot c_2^{-1}$
5. done

Due to the fact that “Marvin is” was highlighted that much in the description, we guessed that it was the first plaintext.

```
#!/usr/bin/env python
def txt(istr):
        return int(istr.encode("hex"),16)
x1 = txt("Marvin is")
y1 = 71164450240897430648972143714791734771985061339722673162401654668605658194656
y2 = 12951693517100633909800921421096074083332346613461419370069191654560064909824
p = 0xA9FB57DBA1EEA9BC3E660A909D838D726E3BF623D52620282013481D1F6E5377
A = 0x7D5A0975FC2C3057EEF67530417AFFE7FB8055C126DC5C6CE94A4B44F330B5D9
B = 0x26DC5C6CE94A4B44F330B5D9BBD77CBF958416295CF7E1CE6BCCDC18FF8C07B6

# from: http://eli.thegreenplace.net/2009/03/07/computing-modular-square-roots-in-python/
def modular_sqrt(a, p):
        """ Find a quadratic residue (mod p) of 'a'. p
        must be an odd prime.

        Solve the congruence of the form:
                x^2 = a (mod p)
        And returns x. Note that p - x is also a root.

        0 is returned is no square root exists for
        these a and p.

        The Tonelli-Shanks algorithm is used (except
        for some simple cases in which the solution
        is known from an identity). This algorithm
        runs in polynomial time (unless the
        generalized Riemann hypothesis is false).
        """
        # Simple cases
        #
        if legendre_symbol(a, p) != 1:
                return 0
        elif a == 0:
                return 0
        elif p == 2:
                return p
        elif p % 4 == 3:
                return pow(a, (p + 1) / 4, p)

        # Partition p-1 to s * 2^e for an odd s (i.e.
        # reduce all the powers of 2 from p-1)
        #
        s = p - 1
        e = 0
        while s % 2 == 0:
                s /= 2
                e += 1

        # Find some 'n' with a legendre symbol n|p = -1.
        # Shouldn't take long.
        #
        n = 2
        while legendre_symbol(n, p) != -1:
                n += 1

        # Here be dragons!
        # Read the paper "Square roots from 1; 24, 51,
        # 10 to Dan Shanks" by Ezra Brown for more
        # information
        #

        # x is a guess of the square root that gets better
        # with each iteration.
        # b is the "fudge factor" - by how much we're off
        # with the guess. The invariant x^2 = ab (mod p)
        # is maintained throughout the loop.
        # g is used for successive powers of n to update
        # both a and b
        # r is the exponent - decreases with each update
        #
        x = pow(a, (s + 1) / 2, p)
        b = pow(a, s, p)
        g = pow(n, s, p)
        r = e

        while True:
                t = b
                m = 0
                for m in xrange(r):
                        if t == 1:
                                break
                        t = pow(t, 2, p)

                if m == 0:
                        return x

                gs = pow(g, 2 ** (r - m - 1), p)
                g = (gs * gs) % p
                x = (x * gs) % p
                b = (b * g) % p
                r = m

# from: http://stackoverflow.com/a/9758173
def legendre_symbol(a, p):
        """ Compute the Legendre symbol a|p using
                Euler's criterion. p is a prime, a is
                relatively prime to p (if p divides
                a, then a|p = 0)

                Returns 1 if a has a square root modulo
                p, -1 otherwise.
        """
        ls = pow(a, (p - 1) / 2, p)
        return -1 if ls == p - 1 else ls


def egcd(a, b):
        if a == 0:
                return (b, 0, 1)
        else:
                g, y, x = egcd(b % a, a)
                return (g, x - (b // a) * y, y)

def modinv(a, m):
        g, x, y = egcd(a, m)
        if g != 1:
                raise Exception('modular inverse does not exist')
        else:
                return x % m

def gety(x):
        y = modular_sqrt(x**3 + A * x + B, p) % p
        y2 = -y % p
        return y,y2

def hextotext(nbr):
        s = hex(nbr)[2:-1]
        if len(s) % 2 ==1:
                s = "0"+s
        return s.decode("hex")


x1_inv = modinv(x1, p)
c1 = (y1 * x1_inv) % p
c2_1, c2_2 = gety(c1)

print repr(hextotext(y2*modinv(c2_1, p)  % p))
print repr(hextotext(y2*modinv(c2_2, p)  % p))
```

Posted by comawill & tsuro & spq Oct 26th, 2013 Categories: crypto, ctf, english, hack.lu13, writeup

« Zeitabgleich: CCCAC Hack.lu 2013: Robot Plans »

Like this post

Be the first to comment...

Recent Posts

- Stratumnews Vol. 19
- Stratumnews Vol. 18
- Stratumnews Vol. 17
- Stratumnews Vol. 16
- Replacing a Husqvarna Viking Designer Topaz 30 Needle Holder

Categories

- english (30)
- ctf (25)
- writeup (24)
- german (17)
- hack.lu13 (10)
- CSAW2013 (6)
- asisCTF13 (4)
- stratumnews (4)
- crypto (4)
- howto (3)

Impressum