ElG-6) $r = g^k \bmod p$

$x \equiv k^{-1}(H + a_A r) \pmod{p-1}$

If Alice uses H_1 and H_2 but the same k both days, some things are kept constant, so

$$x_1 \equiv k^{-1}(H_1 + a_A r) \pmod{p-1}$$
$$x_2 \equiv k^{-1}(H_2 + a_A r) \pmod{p-1}$$

$$x_1 - x_2 \equiv k^{-1}(H_1 - H_2) \pmod{p-1}$$

To find a_A, Eve can first find k since x_1, x_2, H_1, H_2 are known, and then use either x_1, H_1, k or x_2, H_2, k to solve for a_A in either one of the equations.

ECDSA-1) The check is

$$G * H + a_A G * kG[1] = kG * x$$

```
? ellpow(E,G,H)
%15 = [Mod(35634253512680661292, 100000000000000000039), Mod(77324529282921925367, 100000000000000000039)]
? ellpow(E,aAG,kG[1])
%16 = [Mod(41228830649142682590, 100000000000000000039), Mod(36578933883955767227, 100000000000000000039)]
? elladd(E,%15,%16)
%17 = [Mod(19543389628484684932, 100000000000000000039), Mod(99444274481452187725, 100000000000000000039)]
? lift(%17)
%18 = [19543389628484684932, 99444274481452187725]
? ellpow(E,kG,x)
%19 = [Mod(19543389628484684932, 100000000000000000039), Mod(99444274481452187725, 100000000000000000039)]
? lift(%19)
%20 = [19543389628484684932, 99444274481452187725]
? right=%18
%21 = [19543389628484684932, 99444274481452187725]
? left=%20
%22 = [19543389628484684932, 99444274481452187725]
? \l
   log = 0 (off)
   [logfile was "pari.log"]
```

ElG-7) i) From $kx \equiv H + a_A r \pmod{p-1}$, Freddy can solve for a_A since it is the only unknown.

ii) It's hard to brute force all exponents to find a_A.

iv) Freddy has two unknowns, a_A and k, and therefore have infinite a_p's and k's that fit the equation

v) It's hard to brute force exponents

vi) Freddy cannot solve for a_p or k if both are unknown since there are infinite solutions.

Cert-1) The first two steps are important in that the browser verifies the autheticity of the certificate. One solution is to brute force all signature values raised to $e_{ver} \bmod n_{ver}$ until it equals the hash value.