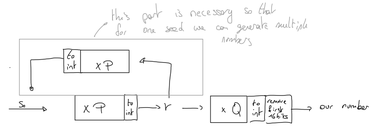
Mass surveillance

Nist's Dual EC PRNG

General Id:

↳ National Institute of standart and technology accepted to use the ideal point Elliptic curve Pseudo Random Number generator which might have a backdoor built in by NSA!

How the algorithm works



Issue

- ? is stated to be random
- Q not.
- If $Q = eP$ we could:
  1) test all 16 bit combination a st. a·x or a is on the elliptic curve. That is a small set of points
     ⇒ list of candidates for $\{0,1\}^{16} \times Q$
  2) Compute $\{0,1\}^{16} \cdot e^{-1}$ = Next state
  3) Can predict any future outputs! Check which $\{0,1\}^{16}$ gives correct next OUT!

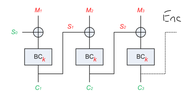
Impossible to prove because EC problem ($Q = eP$) supposed hard

Algorithm Substitution Attack

General Idea:

↳ Adversary substitutes Algorithm on a computer by $\hat{E}$ st:
- → Outputs are undistinguishable
- → Output leaks information (ex: the key of Algo enc)

Example:



- in classical E, s_0 is random
- Force $s_0 = Enc_{pk}(k)$

Note: → pk is the private key of the attacker
→ Attacker can just get k by decrypting with its sk
→ Since BC_k^{-1} is public if you know k...

Stateless SE:

" Any randomized stateless Symetric Encryption Scheme is vulnerable to an undetectable ASA attack leading to key discovery "

Proof:

- Let $F: \{0,1\}^\lambda \to \{0,1\}$ be a PRF with key $\hat{K}$ that Adversary holds.
- To leak bit K[j] encrypt as follows:
  1) choose $s \xleftarrow{\$}$ honestly
  2) $Enc_K(s, m) = C$
  3) check if $F_{\hat{K}}(C) \bmod i$ndice j = K[j]
  4) if not seed another s

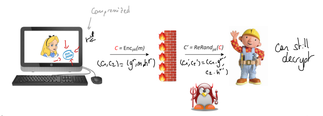
Solution: Stateful schemes

↳ have for a given scheme at most 1 ciphertext possible so that adversary cannot leak.

↳ Double counter synchronized

Crypto Reverse Firewall

General Idea (El Gamal)



⇒ Allows to solve Diffie Hellman with a contaminated computer

Big Key Cryptography

Initial Issue:

Let's say an attacker could leak l bits of the running seed (iteration (ex: via steganography)

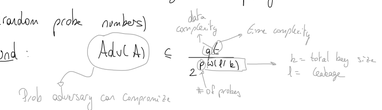
if the initial state is derived from (relatively small) key it's not good.

Note: A often have limited output throughput

Solution

Used Massive initial key and sample from it (random probe numbers)

Bound: $Adv(A) \leq \frac{q \cdot 2^l}{2^{|k|}} + \frac{(qt)^2}{2^{|k|}}$



Hardware Trojans

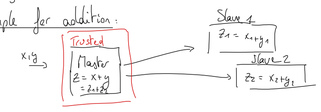
→ If hardware is compromized ... What can we do?

→ Need Prevention from:
- Time bombs (= activates after some time) → Testing
- Cheat code (= activates at certain input) → Scrambling

Scrambling

↳ Use a master that will split into random shares so that even if "code" was in input it will not be anymore!

Example for addition:



Our slaves never see the correct input

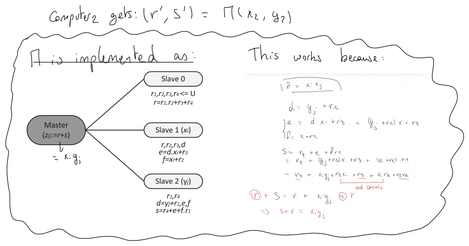
Example for multiplication

We want $z = (x_1 + x_2)(y_1 + y_2) = x_1y_1 + x_2y_1 + x_1y_2 + x_2y_2$

Computer1 gets: $(r, s) = \Pi(x_1, y_2)$ s.t. $r + s = x_1 y_2$

Computer2 gets: $(r', s') = \Pi(x_2, y_1)$

Π is implemented as:



This works because:

Computer gets: $z_1 = x_1y_1 + r + r'$
$z_2 = x_2y_2 + s + s'$

Sum is $z_1 \oplus z_2 = x_1y_1 + x_1y_2 + x_2y_1 + x_2y_2$

Note: We need the randomness. Let's have an example

If $z_1 = x_1y_1 + x_1y_2 = x_1(y_1 + y_2) = x_1y$

$z_2 = x_2(y_1 + y_2) = x_2y$

If you see $z_1 = 1$ and know $x_1 = 1$
$\Rightarrow y = 1$

So we need randomness with r and r'

Testing

For Multi Party Computation

↳ test a random number of times every circuit. Testing random otherwise trojan could just wait fixed # of time. $t_{test} \in \{0,1\}^\ell$

↳ Then run n executions (just use it)

↳ Guaranteed correctness for n cc ξ → # of sub circuits → half must work

↳ Probability of attack $\leq \left(\frac{un}{\xi}\right)^{X/2}$

~ prob that will have an issue on n runs while not declenche on more ξ tests

For generating Pseudo Random Numbers

↳ Run λ secure PRG's

↳ XOR their outputs

↳ everyone must be broken to broke:

$\left(\frac{un}{\xi}\right)^\lambda$