NTRU Key Exchange

based on a posting of Lars Luthman on the Cryptography mailinglist on 05/17/2014

The search for a Post-Quantum Diffie-Hellman replacement

Diffie-Hellman

- Alice
- generate a,g,p
- A=g^d mod p
- send(g,p,A)

- get(B)
- K=B^a mod p

- Bob
- generate b

- get(g,p,A)
- B=g^b mod p
- send(B)
- K=A^b mod p

NTRU Key Exchange

- Alice
- pubA, privA, randA
- send(pubA)
- get(pubB)
- eA=enc(rA,pubB)
- send(eA)
- get(eB)
- rB=dec(eB,privA)
- K=hash(rA,rB)

- Bob
- pubB,privB,randB
- send(pubB)
- get(pubA)
- eB=enc(rB,pubB)
- send(eB)
- get(eA)
- rA=dec(eA,privB)
- K=hash(rA,rB)

Requirements

- K cannot be computed with the knowledge of all sent data
  - Fulfilled, since only pubA, pubB, eA and eB are sent, given that the public-key encryption is secure
- None of the parties can choose the resulting key by choosing the input parameters (your communication peer cannot force you to communicate with a bad key)

Randomness

- The security of Random Number Generators is a tough problem
  - It is impossible to proof that any given random number generator is secure
- The history of broken random number generators is long
  - Therefore it has to be assumed that any given Random Number Generator might be insecure, resulting in insecure keys

Requirements

- One important requirement for a Key Exchange algorithm is that
- if any of the parties follows the protocol
- and at least one of the parties has a good random number generator
- that the party can trust that the resulting key will be secure

- If you follow the protocol
- you will get a secure key EVEN IF your own random number generator is broken
- or if the random number generator of the other party is broken
- And even if both random number generators are partly broken, there is a chance that you will get a secure key

Potential problems

- In the original DH algorithm, Alice leaks the randomness that was used to generate g and p to a passive attacker. Bob does not leak any randomness.
- NTRU key exchange does not leak random numbers to a passive attacker
- In NTRU Key exchange, an active attacker can get rA from Alice and rB from Bob leaked.

Hash security and Race condition

- It was suggested that XOR could be a sufficient Hash algorithm
- But there is a race condition between Alice and Bob.
- If Bob sends eB before Alice sends eA, then Alice can decrypt rB, and generate/choose rA, which would then be hashed together with rB

Race condition

- If Alice would set rA to rB, then in the case of XOR, this would result in
- K:=rA x rB = 0

- The original DH algorithm does not have such a weakness

Potential solutions

- A strong hash algorithm (SHA-384) should be used, preferrably in a HMAC way.
- Alice and Bob should do a Bit-commitment of rA and rB and send that together with the initial pubA and pubB handshake.

Another problem

- It could be argued that Diffie-Hellman itself is not an encryption algorithm
- But for this Key-Agreement protocol, we need an encryption algorithm.
- Some people unfortunately don´t like encryption algorithms

Performance

- DH requires 2 communications:
  - 1. A->B: g,p,A B knows K
  - 2. A<-B: B A knows K
- NTRU Key Exchange requires 3 communications:
  - 1. A->B: pubA
  - 2. A<-B: pubB, eB A knows K
  - 3. A->B: eA B knows K
  - This means likely more latency

Alternatives

- NTRU-KE: A Lattice-based Public Key Exchange Protocol
  - by Xinyu Lei and Xiaofeng Liao
  - https://eprint.iacr.org/2013/718.pdf

Similarities with NTRU-KE

– it comes to the same conclusion that 3 messages are necessary due to the public key structure of NTRU

Advantages of NTRU KE

- NTRU KE might be a bit more efficient than this proposal

Advantages of NTRU Key Exchange

- NTRU KE invents ist own NTRU-ENCRYPT inversion problem and NTRU-ENCRYPT assumption, which are not necessary here
- NTRU Key Exchange reuses the security properties of NTRU and can also easily be used with different Public Key Encryption algorithms instead of NTRU.
- NTRU KE is more complex and likely needs more complex code