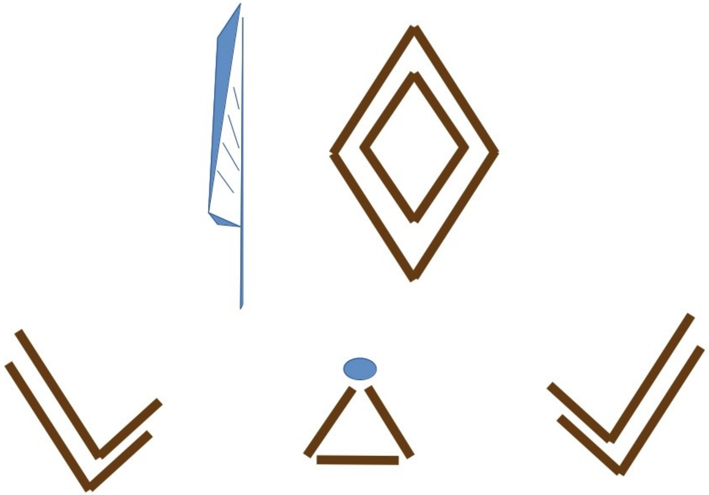
Fetu PKD



Fetu PKD

Public Key Distribution Algorithm

What is Fetu PKD?

- Fetu PKD is a Public Key Distribution Algorithm
- Fetu PKD allows 2 parties to make mathematical exchanges that allow them to compute the same secret key with which to pair with a symmetric key cipher. Attackers are not able to timely calculate the secret key values of the two parties.

Fetu PKD

- Invented in 2021 by Karl Zander
- Reference implementation in Python requiring pycryptodome

Based on Diffie-Hellman

- First published in 1976 by Whitfield Diffie and Martin Hellman
- Conceptualized by Ralph Merkle

Design Goals

- To be resistant to DLP attacks
- To have smaller key lengths than traditional Diffie-Hellman

Key Generation

- Generate 2 N bit primes, let them be N and M and let them not be equal
- Let M be the public modulus
- Let N be the public key
- Choose a integer in M between 1 and M – 1 and let that be the secret key, SK

Key Exchange Setup

- Alice and Bob generate SK, N, M
- Alice and Bob send their public modulus M to each other and agree on an M as modulus
- Alice and Bob also exchange public keys N
- Alice and Bob both generate a base key Y between 1 and M – 1 and agree upon using a single Y value

Key Exchange Setup

- Alice and Bob select a temporary key T in the space of 1 to M - 1

Key Exchange Phase 0

- Alice and Bob both raise Alice's Public Key N to their temporary secret exponent T modulo the public modulus M. Phase0 is carried on to phase1.

Key Exchange Phase 1

- Alice and Bob both raise their phase0 to Y modulo the public modulus M. They exchange phase 1.

Key Exchange Phase 2

- Alice and Bob compute phase1 raised to the temporary secret exponent T modulo M arriving at the secret modulus.

Key Exchange Phase 3

- Alice and Bob raise Bob’s Public Key N to the power of their secret key modulo the secret modulus and exchange phase3

Key Exchange Phase 4

- Alice and Bob raise each other's phase3 to their secret exponent modulo the secret modulus and arrive at the shared key

Encryption

- Encryption can be achieved by multiplying the cipher text with the secret derived in phase4
- Encryption can also be achieved by applying a hash function to the secret derived in phase4 and using the resulting hash as the symmetric secret key

Cryptanalysis

- Phase2 cryptanalysis – in-progress
- Phase4 cryptanalysis - in-progress