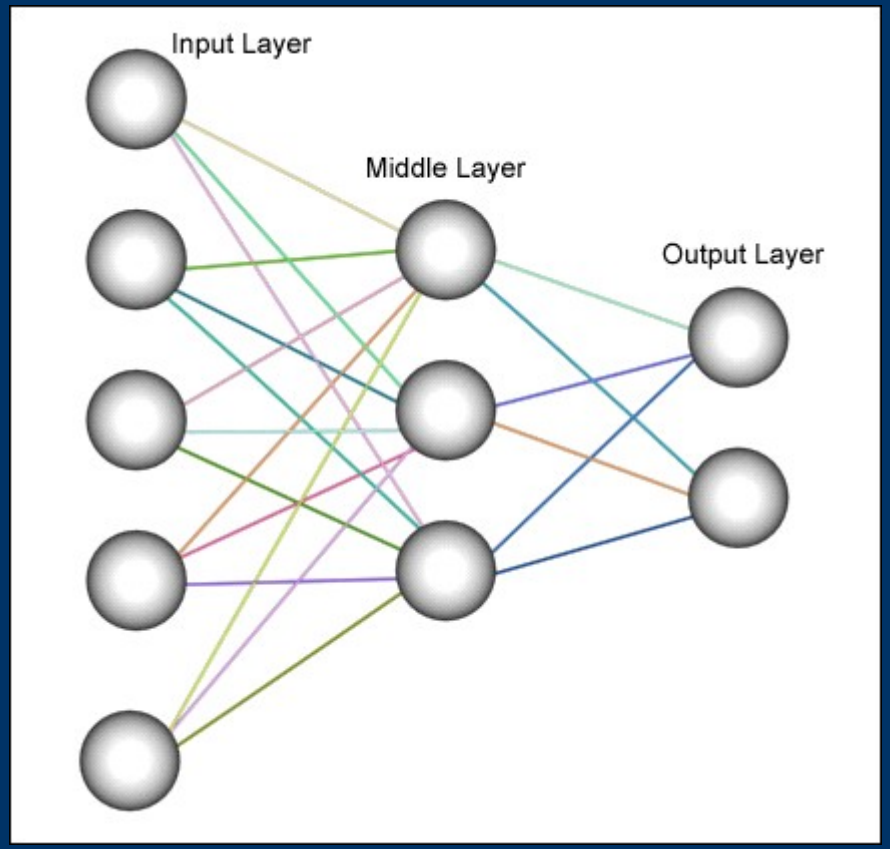
Neural Key Exchange

Presented by: Jessica Lowell
10 December 2009
CS 6750

What is a neural network?

- Simple processing elements which can exhibit complex global behavior through connections and parameters
- Input, output, and hidden nodes
- Interested in learning a function from inputs to output



Source: GameDev.net

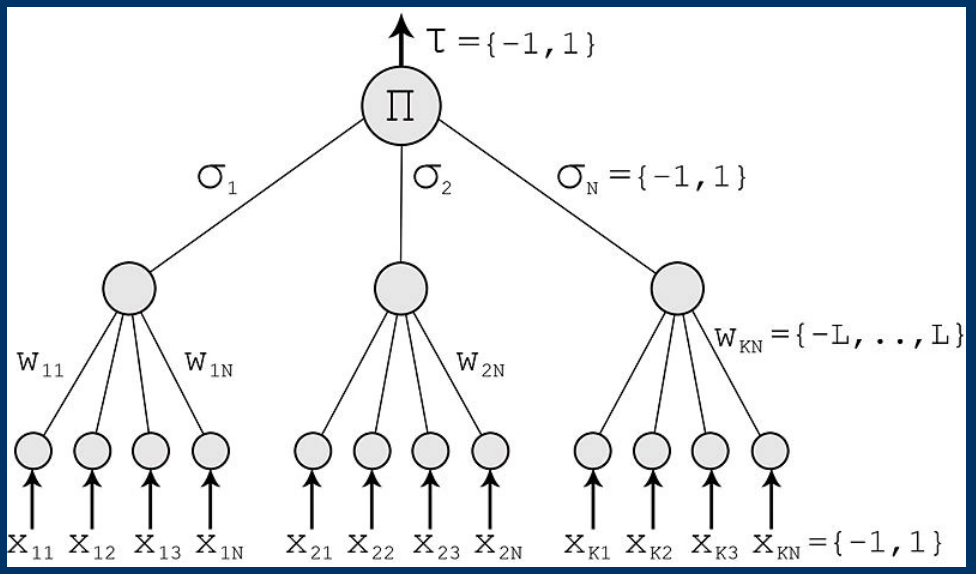
Neural networks in cryptography

- First used for DES cryptanalysis (Dourlens, 1995)
- Decryption (Shihab, 2006)
- Pseudo-random number generation (Karras & Zorkadis, 2003)
- Neural key exchange

The last of these is what we are looking at!

Tree-parity machines

- Type of multilayer feedforward network
- One output, K hidden neurons, K*N inputs, inputs are binary
- Output of each hidden neuron is sum of all multiplications of input neurons, weights
- Binary output value



Source: Wikimedia Commons

Synchronization

- The basis of neural key exchange is synchronization of the weights of tree parity machines.
- Similar to synchronization of chaotic oscillators in chaos communications
- We want to synchronize the weights of the TPMs to establish a key!

Kanter-Kinzel-Kanter: Three main ingredients

- Knowledge of output does not uniquely determine internal representation, so observer cannot tell which weight vector was updated (hidden units)
- Tree parity machine
- Bounded weights

What does this mean?

Observer cannot recover initial weight vectors from the knowledge of time-dependent synchronized keys.

Kanter-Kinzel-Kanter: The protocol

1. Initialize random weight values for each party's tree parity machine.
2. Do until synchronization is achieved:
    1. Generate random input vector X
    2. Compute the hidden neuron values
    3. Compute the output neuron value
    4. Compare the output values of the tree parity machines

        If outputs are different, go to 2.1

        If outputs are the same, apply a learning rule (e.g. Hebbian, Anti-Hebbian, Random Walk) to the weights

Why this is really intriguing

- Low complexity (linear with size of the network)
- Not based on number theory
  - Could potentially give rise to faster key exchange
  - Algorithms based on number theory are potentially vulnerable to having their operations inverted by quantum computers – neural algorithms possibly more secure

Can it be brute-forced?

- Attacker would have to test all possible keys
- $(2L+1)^{KN}$ possibilities
- For a reasonable value of N, is is impossible with today's computer power.

Can attacker fight fire with fire?

Can attacker learn with own tree parity machine?

- Kanter, Kinzel, Kanter observed empirically that attacker synchronized less quickly than Alice and Bob
- Attacker is less likely to make coordinated move with either of the parties than they are to make coordinated move with each other (Klimov et al, 2002)

So what's the catch?

Klimov, Mityaguine, and Shamir found three unusual attacks to which neural key exchange was vulnerable:

1. Geometric attack
2. Genetic attack
3. Probabilistic analysis

Vulnerability: Geometric attack

- Based on the geometric interpretation of the action of a perceptron

The procedure:

- If output A != output B, the attacker doesn't update output C
- If output A = output B and output A = output C, attacker updates using the learning rule
- Otherwise, attacker uses geometry-based formula to update

Vulnerability: Genetic attack

- A biologically-inspired attack for a biologically-inspired cryptosystem
- Simulates a population of tree parity machines trained with the same inputs as those of the two parties
- “At each stage...networks whose outputs mimic those of the two parties breed and multiply, while unsuccessful networks die.”

Vulnerability: Probabilistic analysis

- Easier to predict the position of a bounded point in a random walk after several moves, than to guess its original position
- The attacker does not know which perceptrons are updated in each round (the moves are unknown)
- Attack uses dynamic programming to calculate the probabilities of particular outputs using probability distribution

Fixes

- Authentication (Volkmer & Schaumburg, 2004)
- Addition of feedback mechanism (Prabakaran et al, 2008)
- One party sending erroneous output bits which the other party can predict and remove (Allam & Abbas, 2009)

Other improvements on the original protocol

- “Public channel cryptography by synchronization of neural networks and chaotic maps” (Mislovaty et al, 2003)
- “Neural cryptography with feedback” (Ruttor et al, 2004)

Conclusion

Neural key exchange is promising against conventional attacks and quantum computing, but needs work against some unconventional attacks.