Last Part on Go

Answer some questions

1. NTRU - in 17 stage to for NIST quantum proof encryption.
2. What is an API? Why are web3/dApps important? What is the big picture?

Goroutines

Go routes allow you to create parallel running code.

```
package main

import (
	"fmt"
	"sync"
)

var wg sync.WaitGroup

func f(from string) {
	wg.Add(1)
	defer wg.Done()
	for i := 0; i < 3; i++ {
		fmt.Printf("%s: %v\n", from, i)
	}
}

func main() {


	f("direct")

	go f("goroutine")

	for i := 0; i < 10; i++ {
		// fmt.Printf("AT: %s\n", godebug.LF())
		wg.Add(1)
		go func(msg string) {
			// fmt.Printf("AT: %s\n", godebug.LF())
			defer wg.Done()
			fmt.Printf("%s\n", msg)
		}(fmt.Sprintf(" I am %d ", i))
	}
	// fmt.Printf("AT: %s\n", godebug.LF())
```

```
        wg.Wait()
        fmt.Printf("All Done\n")
}
```

Go Interfaces

Two uses for interfaces (Actually more than 2 but 2 primary uses).

1. Variable parameter list functions.
2. Interfaces to sets of functions.

Variable parameter list functions.

```
func vexample(a int, b ...interface{}) {
        for pos, bVal := range b {
                switch v := bVal.(type) {
                case int:
                        fmt.Printf("It's an int, %d at %d\n", v, pos)
                case []int:
                        fmt.Printf("It's a slice of int\n")
                default:
                        fmt.Printf("It's a something else\n")
                }
        }
}
```

Interfaces to sets of functions.

```
type InterfaceSpecType interface {
        DoFirstThing(p1 int, p2 int) error
        DoSomethingElse() error
}

type ImplementationType struct {
        AA int
        BB int
}

var _ InterfaceSpecType = (*ImplementationType)(nil)

func NewImplementationType() InterfaceSpecType {
        return &ImplementationType{
                AA: 1,
                BB: 2,
        }
}

func (xy *ImplementationType) DoFirstThing(p1 int, p2 int) error {
        // ... do something ...
        return nil
}

func (xy *ImplementationType) DoSomethingElse() error {
        // ... do something ...
        return nil
}

func Demo() {

        var dd InterfaceSpecType
        dd = NewImplementationType()
        _ = dd.DoSomethingElse()
}
```

Go Channels

We will come back to this later.

Go Weaknesses

What are the limitations of using Go

1. No objects - Use interfaces instead. No inheritance.
2. No generics - Use templates and code instead.
3. No error handling - Just return errors.

Go 2.0 is coming in 1.5 years. Go's design team commitment is 100% backward compatibility - it will be able to correctly compile go 1.0 code without change to the language.