In [1]:
```python
# Styling notebook
from IPython.core.display import HTML
def css_styling():
    styles = open("./styles/custom.css", "r").read()
    return HTML(styles)
css_styling()
```

Out[1]:

Key Exchange (Diffie-Hellman)

Alice and Bob publicly agree to use numbers $p = 23$ and $g = 5$ (fixed here)
Alice: chooses secret integer a, sends Bob $A = g^a \bmod p$
Bob: chooses secret integer b, sends Alice $B = g^b \bmod p$
Alice: computes $s = B^a \bmod p \;\leftarrow$ Shared secret
Bob: computes $s = A^b \bmod p \;\leftarrow$ Shared secret

Why? From Bob's side: $A^b \bmod p = g^{ab} \bmod p = g^{ba} \bmod p = B^a \bmod p$

In [1]:
```python
import random
from Modular import fastExpMod

# Alice and Bob publicly agree to use numbers p = 23 and g = 5.
g = 5
p = 23
print('Alice and Bob agree on g =',g,'and p =',p)

# Alice: chooses secret integer a, sends Bob A = g^a mod p
a = random.randrange(3,100)
A = fastExpMod(g,a,p)
print('Alice sends A:',A)

# Bob: chooses secret integer b, sends Alice B = g^b mod p
b = random.randrange(3,100)
B = fastExpMod(g,b,p)
print('Bob sends B:',B)

# Alice: computes s = B^a mod p
sAlice = fastExpMod(B,a,p)
print('Alice knows',sAlice)

# Bob: computes s = A^b mod p
sBob = fastExpMod(A,b,p)
print('Bob knows',sBob)
```

```
267
Alice and Bob agree on g = 5 and p = 23
Alice sends A: 19
Bob sends B: 6
Alice knows 8
Bob knows 8
```

Breaking Diffie-Hellman

Scenarion: An eavesdropper can only hear g, p, A, B. This is how the secret s can be recomputed from that:

1. Solve (for b) the *Discrete Logarithm Problem* $B = g^b \bmod p$ (or symmetric for $A = g^a \bmod p$) That's not as easy as it looks - we have a solution that uses brute force in the [Modular Notebook (Modular.ipynb#dlog)](Modular.ipynb#dlog).
2. Compute $s = A^b \bmod p$ with the b you just found

In [3]:
```python
from Modular import fastExpMod, dLog

def breakDH(g,p,A,B) :
    return fastExpMod(A,dLog(g,p,B),p)
```

In [4]:
```
breakDH(5,23,19,6)
```
Out[4]:
```
8
```