# Information Security Lab: Module 4

## Lab Session Week 2: Lattice Reduction

Kenny Paterson, Jan Gilcher, Kien Tuong Truong

# Your tasks

- For this week you have the following tasks
  - Run lattice attacks using partial knowledge of nonces.
  - Exploit side channels on broken implementations of Schnorr.

# Task 0 and 1: Lattice Attack

- For both tasks, we give you a server.py which will answer a limited number of signature queries.
- Each reply will also send you some MSB of the nonce.
- Goal: Recover the key using lattice techniques and forge a signature.
- You have seen (Lecture/Exercise) how to do this for ECDSA, now do it for Schnorr.
- **DO NOT USE FPYLLL FOR YOUR LATTICE REDUCTIONS!**

# Task 2: Timing Attacks on Schnorr (E2E-Version)

- By this point you have already managed:
  - How to recover nonce MSB from timing (Lab 1)
  - How to use knowledge about MSB to recover the secret key (previous tasks)
- Let’s combine this for a more realistic attack!
- Similar server to last week
- Forge a signature!

# Sagemath Vectors

- Provides vectors over some ring

```
vector([1,2,3,4]) # (1,2,3,4)
vector(Zmod(3), [1,2,3,4]) # (1,2,0,1)
# works with any 1-dimensional iterable
vector(i^2 for i in range(3)) # (0,1,4)
vector([[1,2],[3,4]]) # throws an error!
# parentheses needed when more than 1 arg
vector(Zmod(3), (i^2 for i in range(3))) # (0,1,1)
```

# Sagemath Vectors

```
# Integers are cast down to rings
vector([1,Zmod(3)(2),3,4]) # (1,2,0,1)
# other modular rings might not!
vector([1,Zmod(3)(2),Zmod(5)(3),4]) # Throws an error!
vector(ZZ,[1,Zmod(3)(2),Zmod(5)(3),4]) # (1,2,3,4)
# empty vectors can be created via a degree argument
vector(Zmod(3),4) # (0,0,0,0)
# degree argument can be used for error checks
vector(Zmod(3),2,[1,2,3]) # Throws an error!
vector(Zmod(3),3,[1,2,3]) # OK! (1,2,0)
```

# Sagemath Matrix

- Similar to Vector, just 2 dimensional

```
                  # [1 2]
matrix([[1,2],[3,4]]) # [3 4]

                              # [1 2]
matrix(Zmod(3), [[1,2],[3,4]]) # [0 1]

             # [0 0]
matrix(ZZ,2) # [0 0]

               # [0 0 0]
matrix(ZZ,2,3) # [0 0 0]
```

# Sagemath Matrix

```
                        # [1 2]
matrix(ZZ,2,[1,2,3,4]) # [3 4]
#alternatively:
v=vector(ZZ, [1,2,3,4])
            # [1 2]
matrix(2,v) # [3 4]
                              # [1 2 3]
matrix(ZZ,2,3,[1,2,3,4,5,6]) # [4 5 6]
matrix(ZZ,2,3,[1,2,3,4]) # Throws an Error!
```

# Sagemath Matrix

```
v1=vector([1,2])
v2=vector([3,4])
                # [1 2]
matrix([v1,v2]) # [3 4]
                # [1 0]
matrix.diag(v1) # [0 2]
                      # [1 0]
matrix.identity(ZZ,2) # [0 1]
# functions are also allowed to initialize values:
                                             # [1 1/2 1/3]
matrix(QQ, 2,3, lambda x,y: (x+1)/(y+1)) # [2 1 2/3]
```

# Sagemath Matrix

```
                                         # [5 0]
A = matrix.identity(ZZ,2) * 5 # [0 5]
B = A*vector([1,2]) # (5, 10) <- this is a vector!
                                               # [5  0|0]
                                               # [0  5|0]
                                               # [----+-]
matrix.block([[A,0],[matrix(B),2]]) # [5 10|2]
# 0 represents any 0 block matrix.
# Other integers are diagonal matrices
```

# Sagemath LLL

- Dense Rational and Integer Matrix object have an LLL method:

```
                                                 # [0 0 0]
                                                 # [0 0 0]
matrix(ZZ,3,lambda x,y: (x+1)*(y+1)).LLL() # [1 2 3]
                                                 # [0   0   0]
                                                 # [0   0   0]
matrix(QQ,3,lambda x,y: (x+1)/(y+1)).LLL() # [1 1/2 1/3]
```

# HELP!?

- If you are really confused you can always conjure the docs
- `help(matrix)` results in:

IPython: home/gilcherj

```
Help on cython_function_or_method in module sage.matrix.constructor:

matrix(*args, **kwds)
    matrix(*args, **kwds)
    File: sage/matrix/constructor.pyx (starting at line 21)

        Create a matrix.

        This implements the ``matrix`` constructor::

            sage: matrix([[1,2],[3,4]])
            [1 2]
            [3 4]

        It also contains methods to create special types of matrices, see
        ``matrix.[tab]`` for more options. For example::

            sage: matrix.identity(2)
            [1 0]
            [0 1]

        INPUT:

:
```

# A few tips

- This weeks exercise contains important information for choosing M for Kannan's Embedding.
- If you choose M well the same solution should work for Task 0 & 1.
- For Task 2:
    - You have to pick a few parameters yourself here.
    - You can use Tasks 1.2 and 2.1 as guideline, but better ones might exist!
    - running the server locally requires a few (reversible) changes of settings on your pc. Check the Lab Sheet for more information.