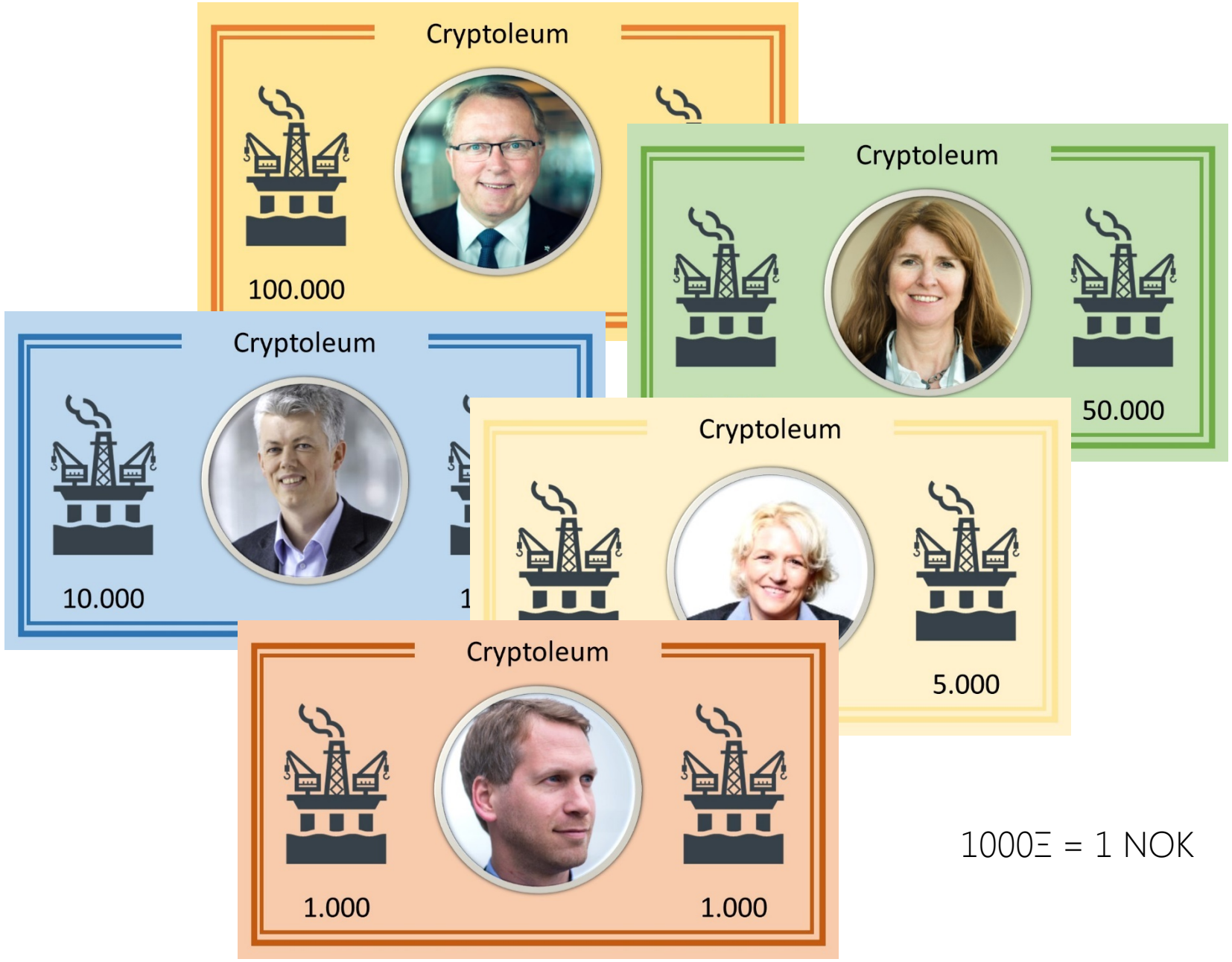
```
Transaction 1 verified by Miner with Alice signature 230 and Bob signature 228
Transaction 2 verified by Miner with David signature 46 and Charlie signature 35
Transaction 3 verified by Miner with Eve signature 178 and Alice signature 191
Transaction 4 verified by Miner with Bob signature 241 and Charlie signature 253
Miner signs block with hash 84 and nonce 19 to signature 122
Block signature verified by Alice
Block nonce verified
```

Practical demo in paper and code

Practical exercise


Cryptoleum
100.000
Cryptoleum
50.000
Cryptoleum
10.000
Cryptoleum
5.000
Cryptoleum
1.000
1.000

1000Ξ = 1 NOK

2 miners

5 wallets

(Alice, Bob, Charlie, David, Eva)

Transaction 1

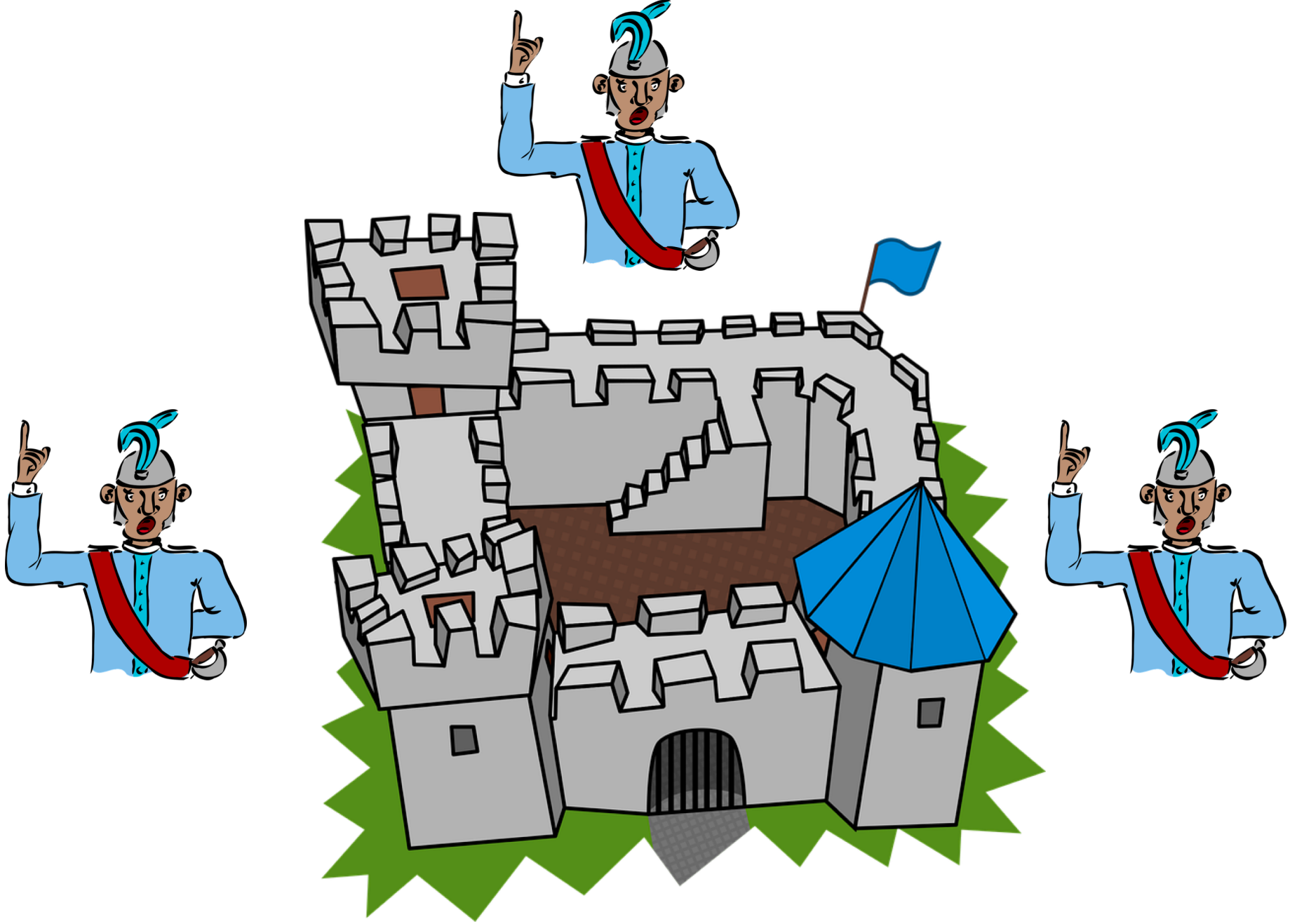
Alice pays Bob 5000Ξ

Kryptoleum transaction

Transaction number	1
Date	31.08.2017 12:27:33
Sender	Alice
Recipient	Bob
Amount	5000
Digital signature	3

Signature: Digit sum (#) + mod(count ([A…Z]);3)
Digit sum: 46 => 10 => 1
Count([A..Z])=8, mod(8;3)=2
Signature = 1+2 = 3

Consensus



Transaction 2

David pays Charlie 15000Ξ

Kryptoleum transaction

Transaction number	2
Date	31.08.2017 12:29:05
Sender	David
Recipient	Charlie
Amount	15000
Digital signature	4

Signature: Digit sum (#) + mod(count ([A…Z]);3)
Digit sum: 49 => 13 => 4
Count([A..Z])=12, mod(12;3)=0
Signature = 4 + 0 = 4

Transaction 3

Eva pays Alice 125000Ξ

Kryptoleum transaction

Transaction number	3
Date	31.08.2017 12:31:45
Sender	Eva
Recipient	Alice
Amount	125000
Digital signature	6

Signature: Digit sum (#) + mod(count ([A…Z]);3)
Digit sum: 49 => 13 => 4
Count([A..Z])=8, mod(8;3)=2
Signature = 4 + 2 = 4

Transaction 4

Bob pays Charlie 37000Ξ

Kryptoleum transaction

Transaction number	4
Date	31.08.2017 12:32:17
Sender	Bob
Recipient	Charlie
Amount	37000
Digital signature	8

Signature: Digit sum (#) + mod(count ([A…Z]);3)
Digit sum: 52 => 7
Count([A..Z])=10, mod(10;3)=1
Signature = 7 + 1 = 8

From transactions to block

Kryptoleum block

Block number	15
Date	31.08.2017 12:33:12
Previous block	14
Previous block signature	8
Transaction numbers	1,2,3,4
Total amount	182000
Digital signature	2
Proof-of-work	

Signature:
Mod (
Digit sum (transaction sig) +
Digit sum (total amount) +
Digit sum (block number) +
Digit sum (transaction #);10)

Digit sum (sig) = 3+4+6+8 =>
21 => 3
Digit sum ($) = 11 => 2
Digit sum (blck #) = 6
Digit sum (trn #) = 10 => 1
Mod (3 + 2 + 6 + 1; 10) = 2

Proof of work

Find the number that together with the block signature opens the lock

Kryptoleum block

Block number	15
Date	31.08.2017 12:33:12
Previous block	14
Previous block signature	8
Transaction numbers	1,2,3,4
Total amount	182000
Digital signature	2
Proof-of-work	4

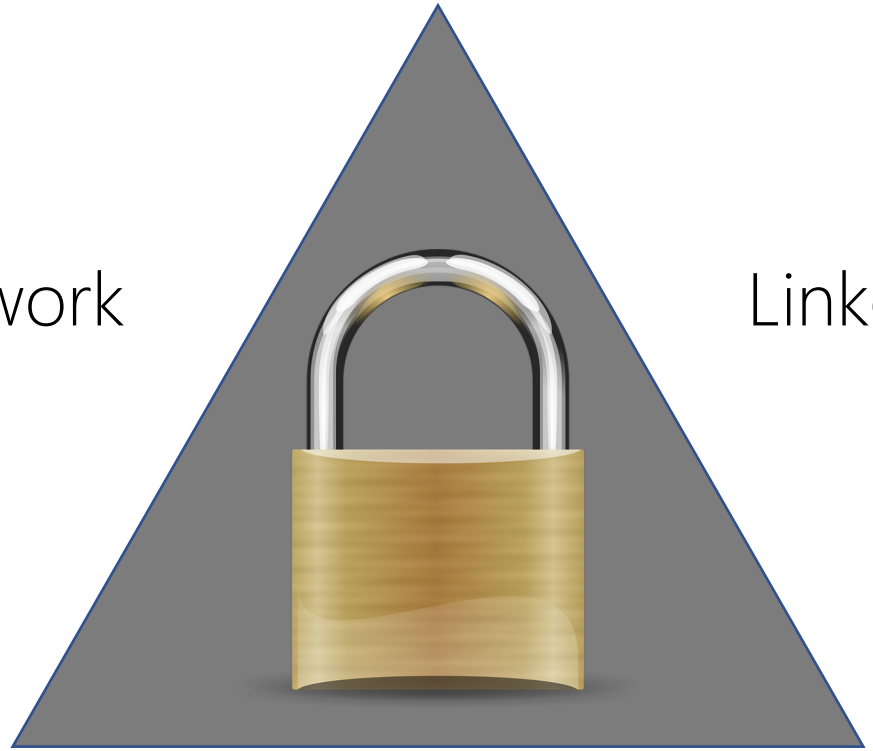
Digit sum (2, 4) = 6

Let’s do this in code

Blockchain security fundamentals

Follow the code
https://github.com/hwes/BlockchainCourse

Proof-of-work



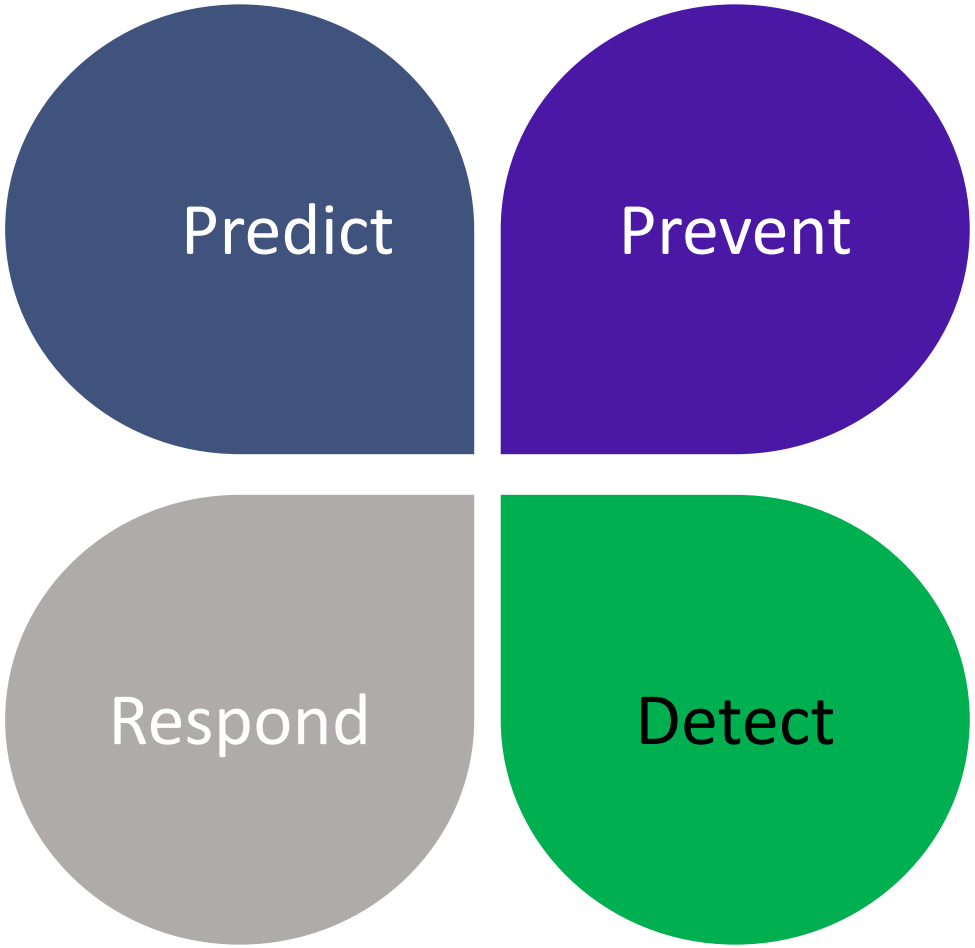
Linked chain

Consensus

Resilience

Structure

Automation


Predict
Prevent
Respond
Detect

Public key cryptography

Challenge

How can I exchange information
with someone I have never met?

One way functions

A one way function is a mathematical function
that is easy to do in one direction but
“impossible” to do in the other direction

Modular arithmetic

Clock arithmetic
6 AM + 8 hours => 2 PM
6+8 = (14 mod 12) = 2

x	1	2	3	4	5	6
3^x	3	9	27	81	243	729
$3^x \pmod 7$	3	2	6	4	5	1

$453^x \pmod{21\,997} = 5787$, x=?

ModularLarge.py

Diffie-Hellman

Use the function $Y^x \pmod P$
1: Call Alice calls Bob. Agree on Y and P, e.g. 7 and 11

2: Choose a number A (e.g. 3)
3: Put A into the one-way function $7^A \pmod{11}$
$\alpha = 7^3 \pmod{11} = 343 \pmod{11} = 2$
4: Call Bob and tell him α

2: Choose a number B (e.g. 6)
3: Put B into the one-way function $7^B \pmod{11}$
$\beta = 7^6 \pmod{11} = 117\,649 \pmod{11} = 4$
4: Call Alice and tell her β

Eve intercepts the number 2 and 4

5: Calculate $\beta^A \pmod{11} = 64 \pmod{11} = 9$

5: Calculate $\alpha^B \pmod{11} = 64 \pmod{11} = 9$

9 is the key

DiffieHellman.py

Run time for increasing A and B

A and B	Run Diffie Hellman	Break Diffie Hellman
Less than 10	0.09 ms	0.09 ms
Less than 50	0.09 ms	3.31 ms
Less than 100	0.09 ms	16.42 ms
Less than 500	0.09 ms	1300 ms
Less than 1000	0.09 ms	12861 ms

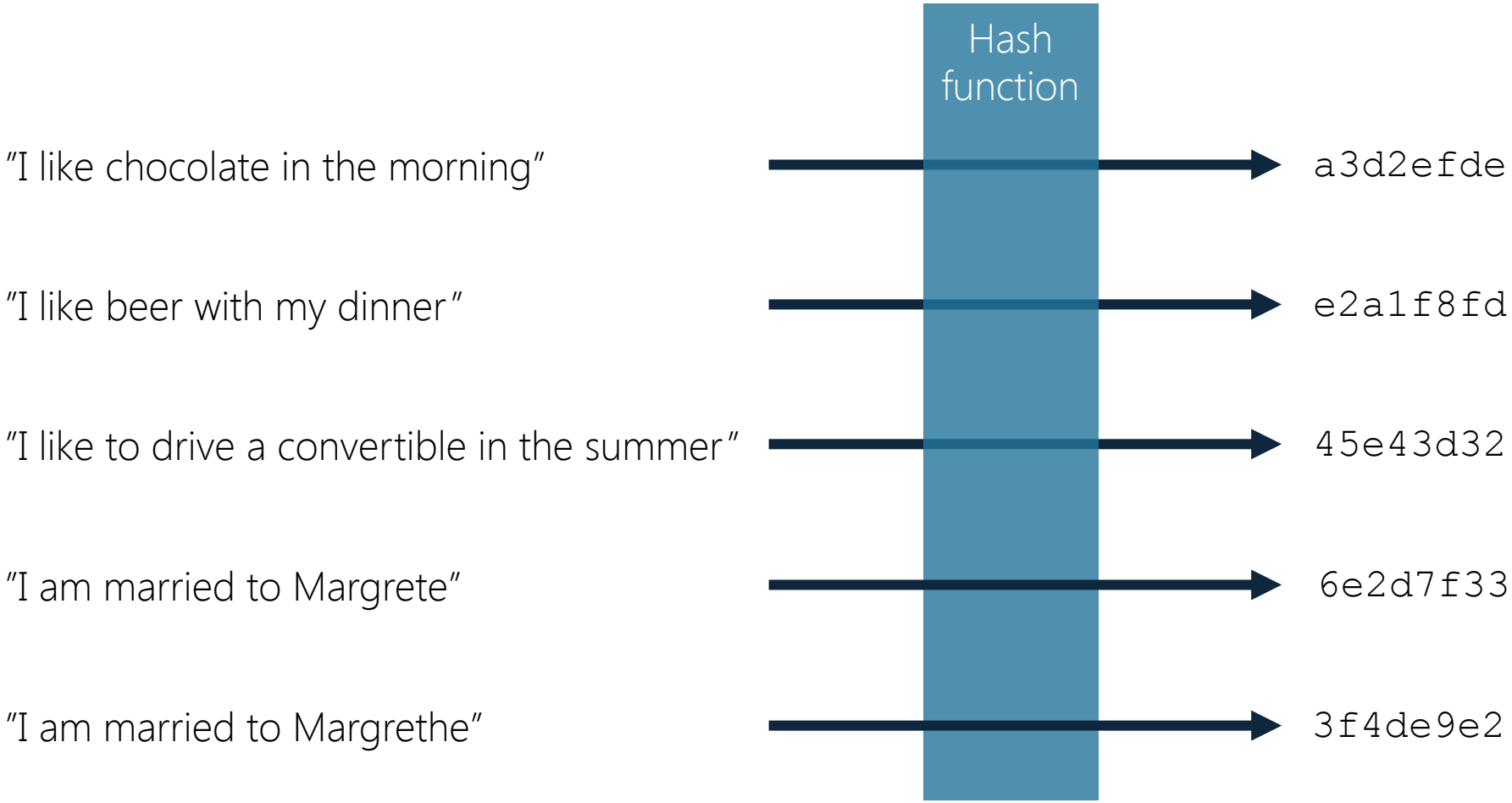
What happens when A and B becomes really large?

DiffieHellman.py & DHBreak.py

Public key cryptography is the foundation of “all” security on the internet. The algorithms are much more complex, but the principles remain the same.

Hashing algorithms

Hashing functions are one-way functions that for any input text maps to a fixed length text, where the fixed length text is “unique” and dependent on the input text


Hash function
"I like chocolate in the morning"
"I like beer with my dinner"
"I like to drive a convertible in the summer"
"I am married to Margrete"
"I am married to Margrethe"
a3d2efde
e2a1f8fd
45e43d32
6e2d7f33
3f4de9e2

Pearson 8-bit hash algorithm

Using the message C and the permutation table T

```
h := 0
for each c in C loop
    h := T[ h xor c ]
end loop
return h
```

Pearson.py & HashTest.py

The nonce

A nonce is a number (or text) that is added to a string so that the computed hash result satisfies certain properties

NonceTest.py

Bringing it all together

- Hashing is used to give each transaction a unique identity
- Alice, Bob, Charlie, David and Eve signs the hash using public key cryptography, creating a transaction signature for the Miner to verify
- The Miner take a group of transactions to create a block
- The miner must find a nonce so that the hash from block and the nonce satisfies certain properties
- The Miner signs the block including the nonce for Alice to verify

Never,
ever,
under any circumstance,
roll your own crypto