Blockchains in IOT

(taking it to the edge)

Chester Rebeiro

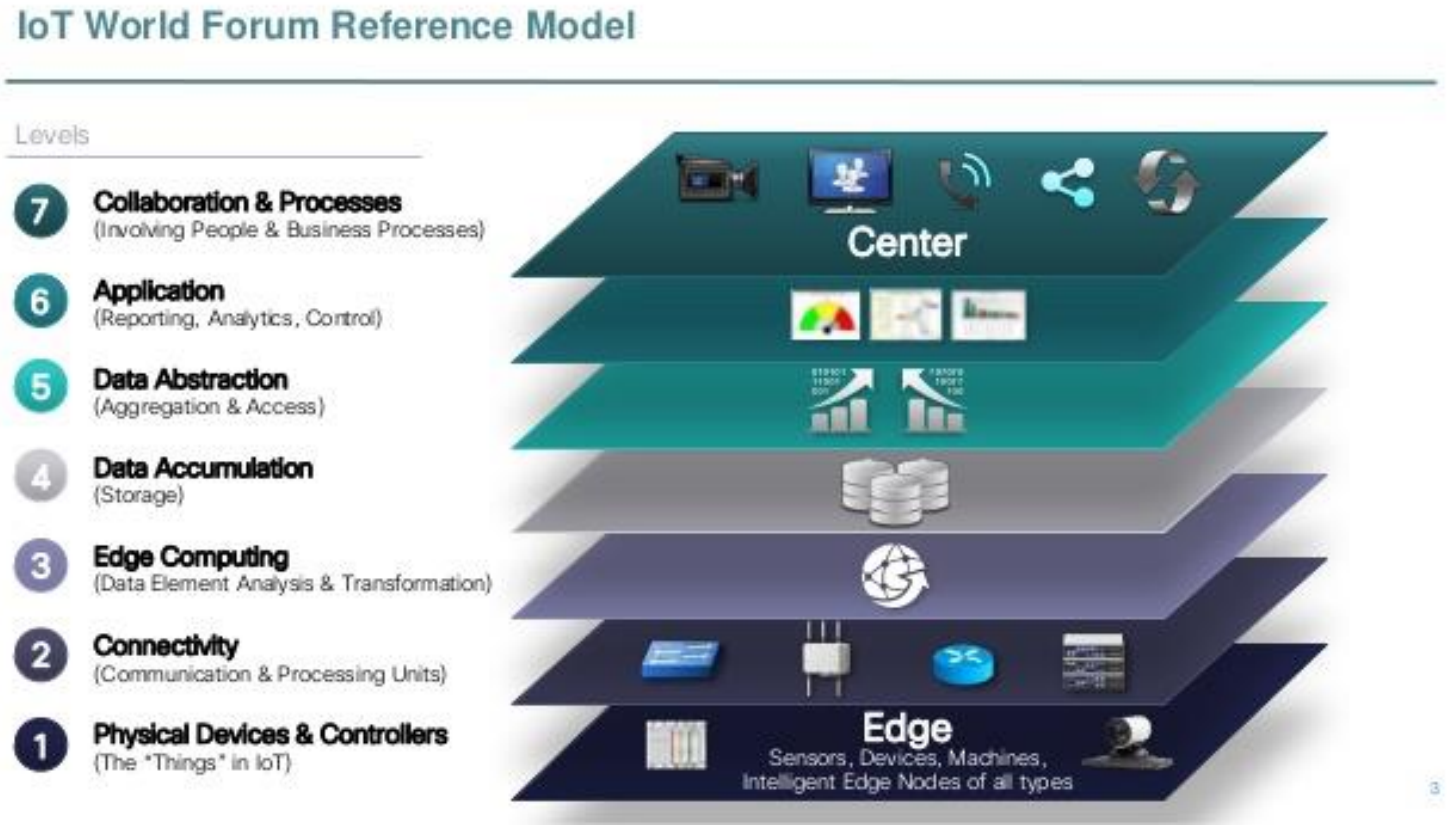
IIT Madras


THE INTERNET of
THINGS

Continuous monitoring of various parameters from a person's health
to environment conditions such as pollution, energy consumption, etc.

Actions taken to control parameters

IoT Reference Model



Edge Devices

1000s of them expected to be deployed

Low power (solar or battery powered)
Small footprint
Connected to sensors and actuators

Expected to operate 24 x 7 almost unmanned

24x7 these devices will be continuously pumping data into the system, which may influence the way cities operate

Will affect us in multiple ways, and we may not even know that they exist.

Threats

Need to guarantee data is well managed and data is legitimate

Edge devices are cheap.

- An adversary may insert a large number of these devices into the system.
- Devices may be cloned.

The threat and promise

Need to guarantee data is well managed and data is legitimate

Two technologies that will help in a big way

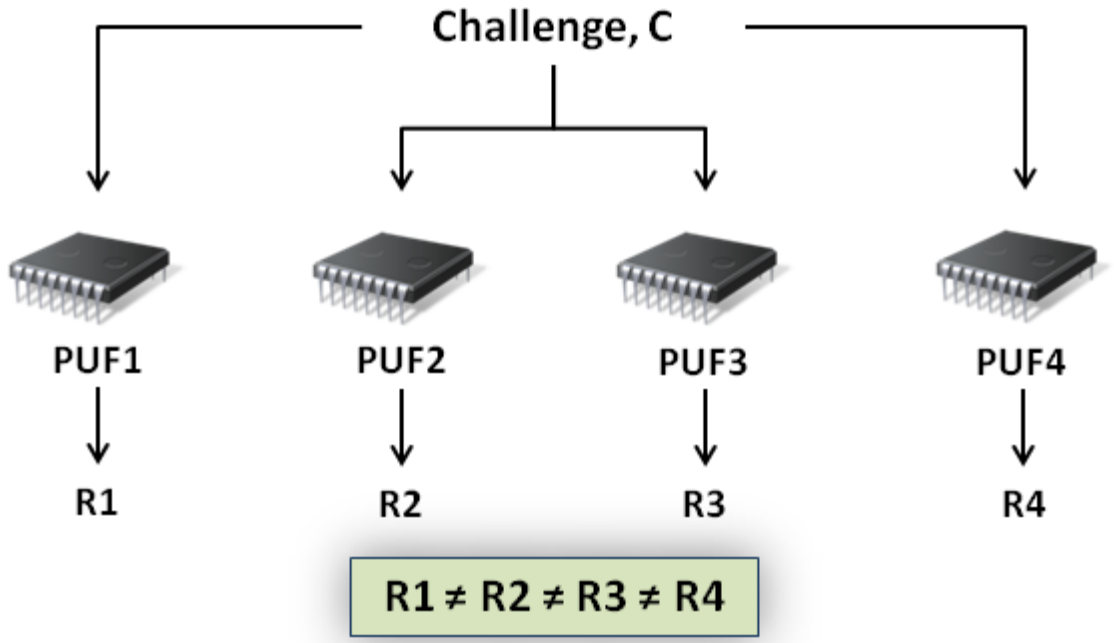
Blockchains:

- Trust
- Record of ownership
- Transparency
- Decentralization

Physically Unclonable Functions (PUFs)

- device level authentication

Physically Unclonable Functions (PUFs)

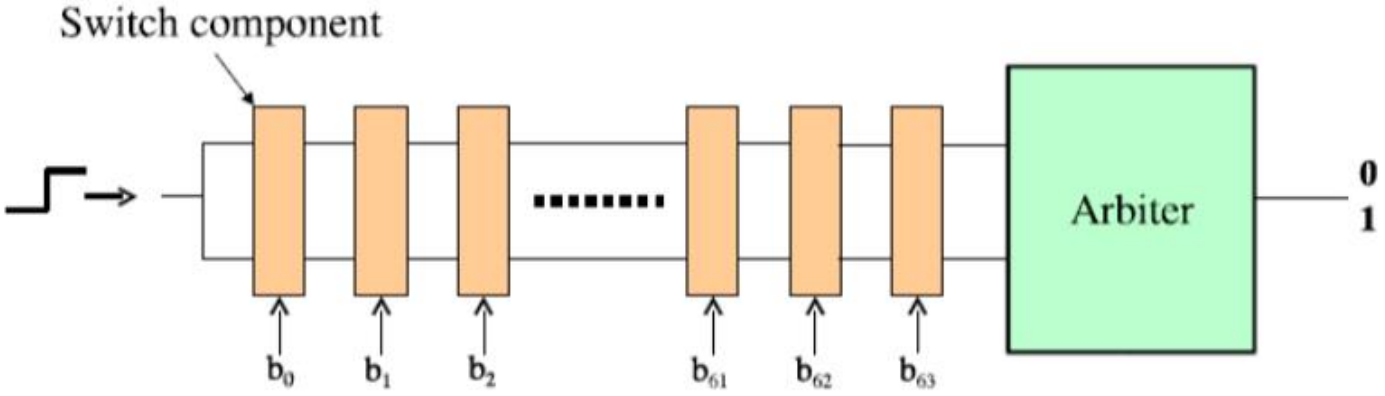


A function whose output depends on the input as well as the device executing it.

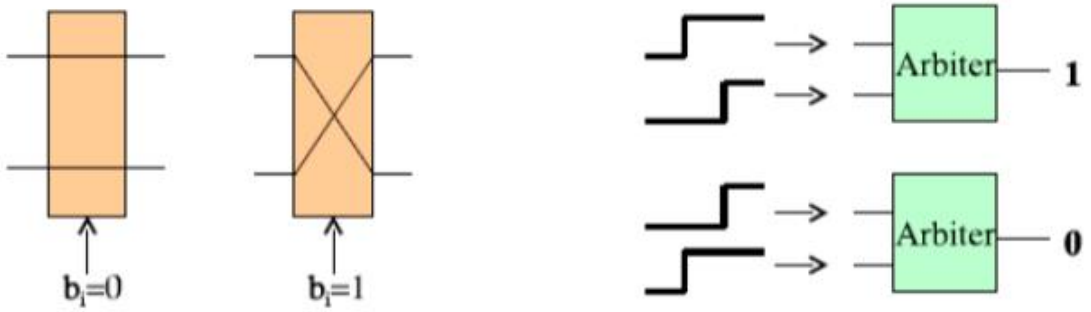
eg. SRAM PUF, Ring Oscillator PUF, Butterfly PUF, Arbiter PUF

Vulnerable to machine learning attacks

Arbiter PUF



Arbiter-based PUF circuit



Enabling Blockchains on Edge Devices

Crypto Primitives

Merkle Tree

Achieves
- consistency proofs
- Audit proofs
- data synchronization

Makes use of **hash functions**

Digital Signatures

Achieves
- authenticity
- non-repudiation
- integrity

Makes use of hash functions and elliptic curves

The minimum requirement for an edge device is to implement a hash Function and compute a digital signature

Cryptographic Hash Functions

Hash functions provide unique digests with high probability.
Even a small change in **M** will result in a new digest

SHA256("short sentence")

0x 0acdf28f4e8b00b399d89ca51f07fef34708e729ae15e85429c5b0f403295cc9

SHA256("The quick brown fox jumps over the lazy **dog**")

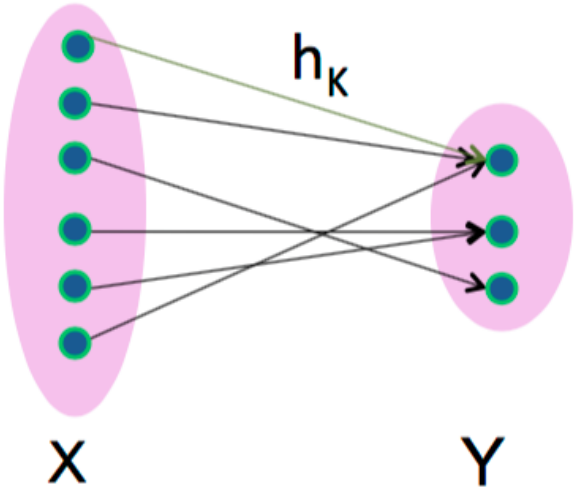
0x d7a8fbb307d7809469ca9abcb0082e4f8d5651e46d3cdb762d02d0bf37c9e592

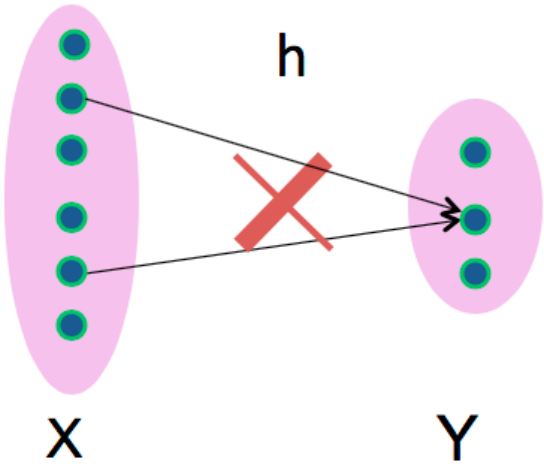
SHA256("The quick brown fox jumps over the lazy **dog.**")

(extra period added)

0x ef537f25c895bfa782526529a9b63d97aa631564d5d789c2b765448c8635fb6c

Cryptographic Hash Functions

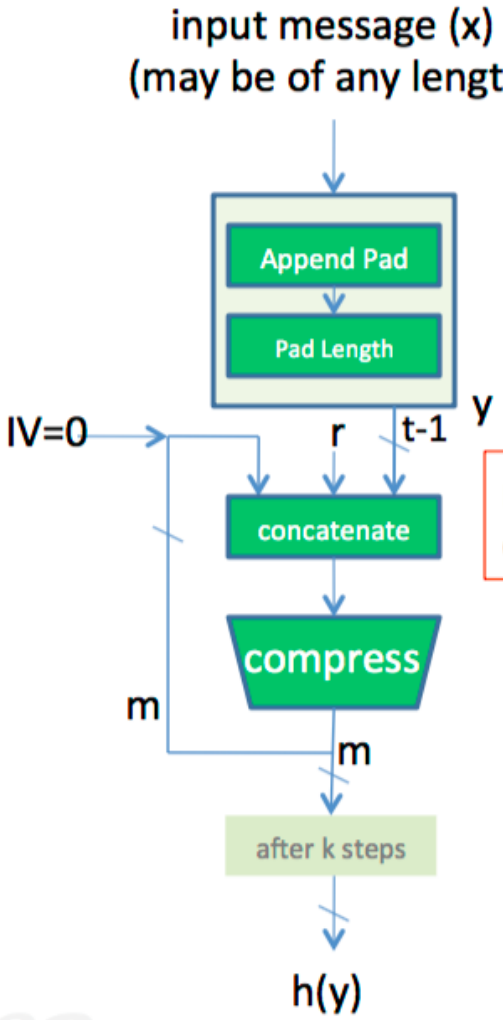




Collision resistance

X may be an infinite set
Y is a finite set (for example having size 2^256)

Merkle Damgard Construction



$$h : \{0,1\}^{m+t} \rightarrow \{0,1\}^{m}$$

$$X = \bigcup_{i=m+t+1}^{\infty} \{0,1\}^{i}$$

r=0 for the first iteration
else r=1

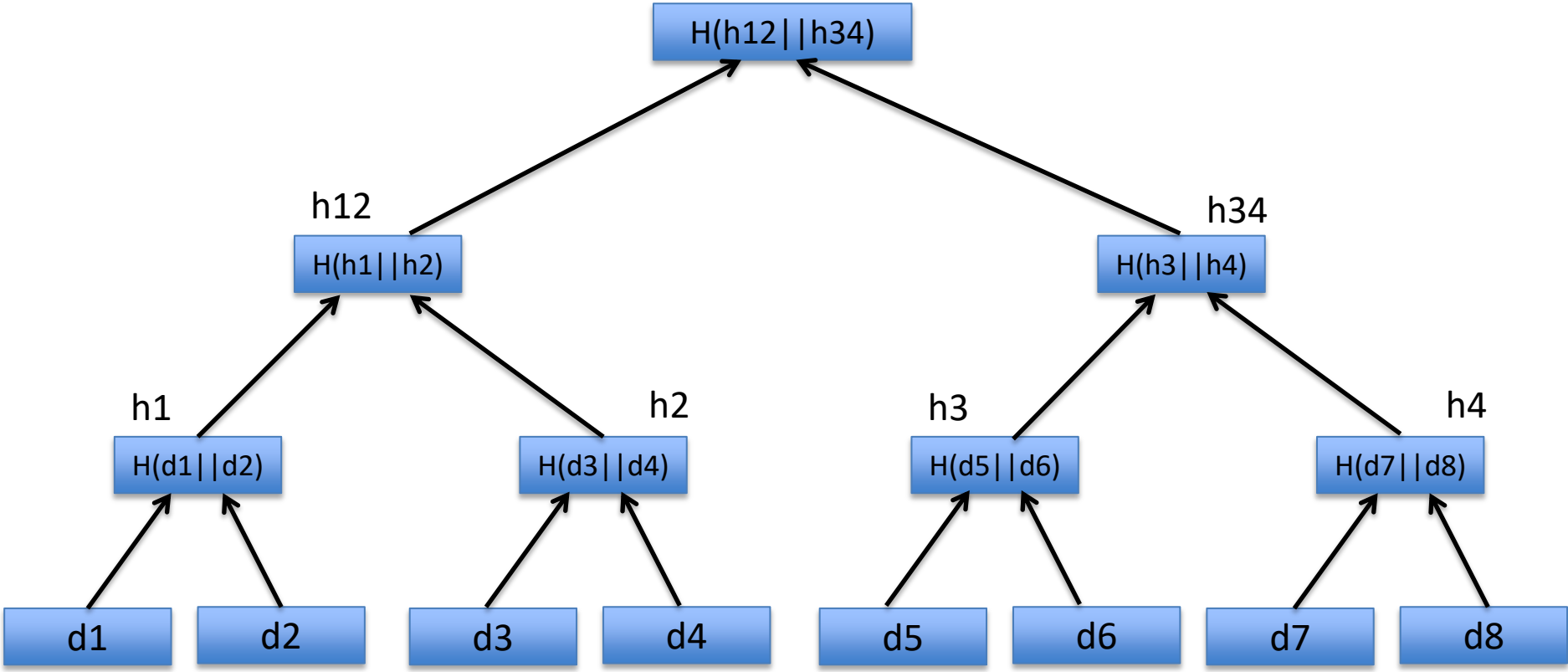
Itrated hash function construction
That uses a compress function h

If h is collision resistant then the Merkle Damgard construction is collision resistant

Hash Functions on Edge Devices

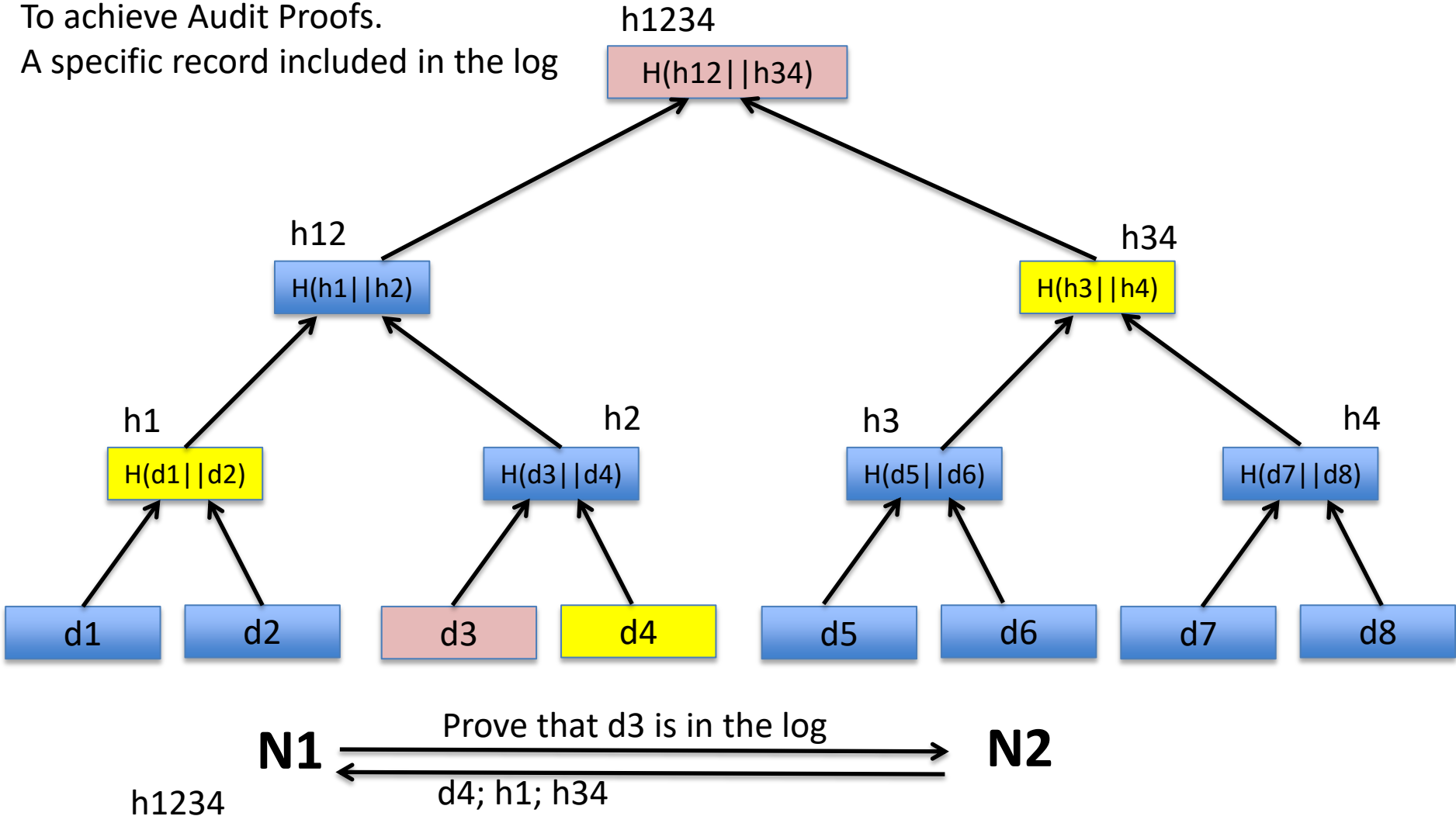
- Simple operations that repeat a large number of times
  - Mod addition, Ex-or, shift operations, etc.
- Memory footprint low
- Computation may be low
  - Continuously repeating may be costly

Merkle Tree


H(h12||h34)
h12
H(h1||h2)
h34
H(h3||h4)
h1
H(d1||d2)
h2
H(d3||d4)
h3
H(d5||d6)
h4
H(d7||d8)
d1
d2
d3
d4
d5
d6
d7
d8

Merkle Tree : Audit Proofs

To achieve Audit Proofs.
A specific record included in the log

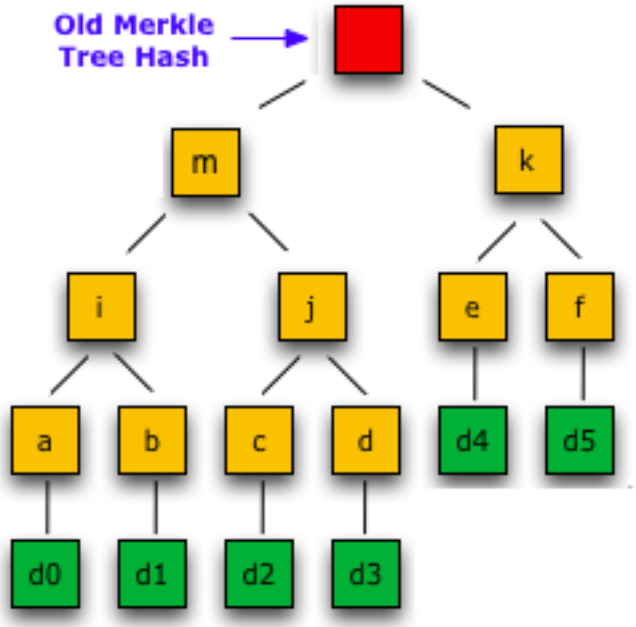


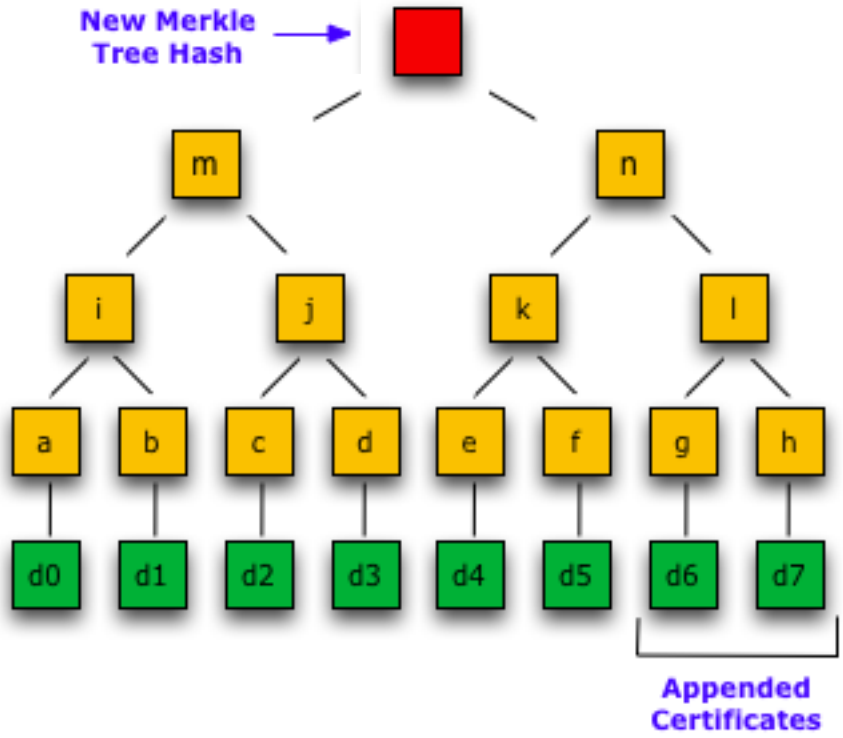
Merkle Trees : Consistency Proofs

- Consistency proof : Lets you verify that a later version of the log includes
  - everything present in an earlier version
  - And in the same order

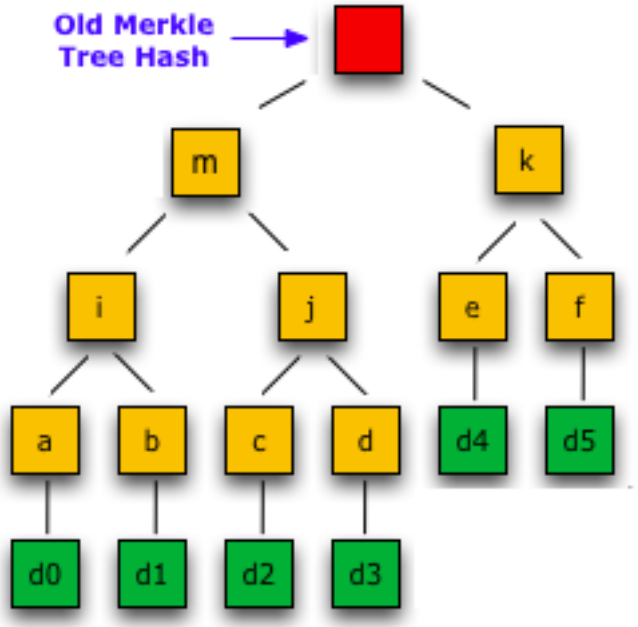
- A proof that a log is consistent would mean that
  - No certificates have been backdated
  - Or inserted into the log
  - No certificates have been modified

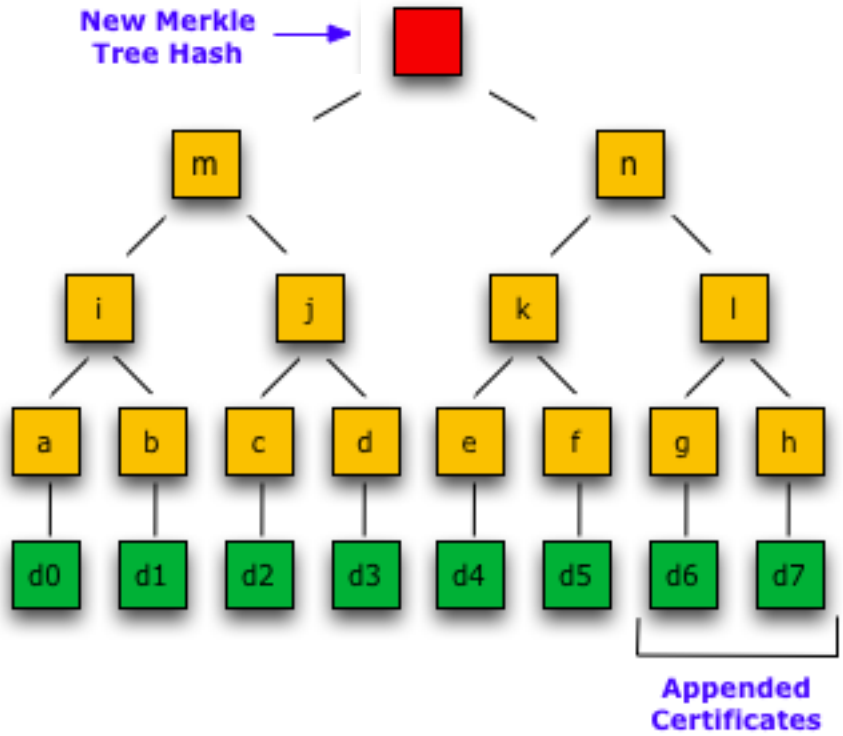
Merkle Trees : Consistency Proofs





Merkle Trees : Consistency Proofs



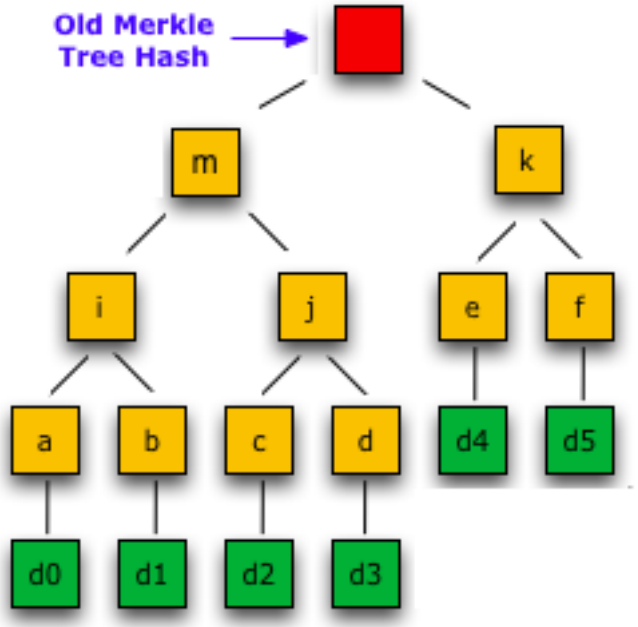


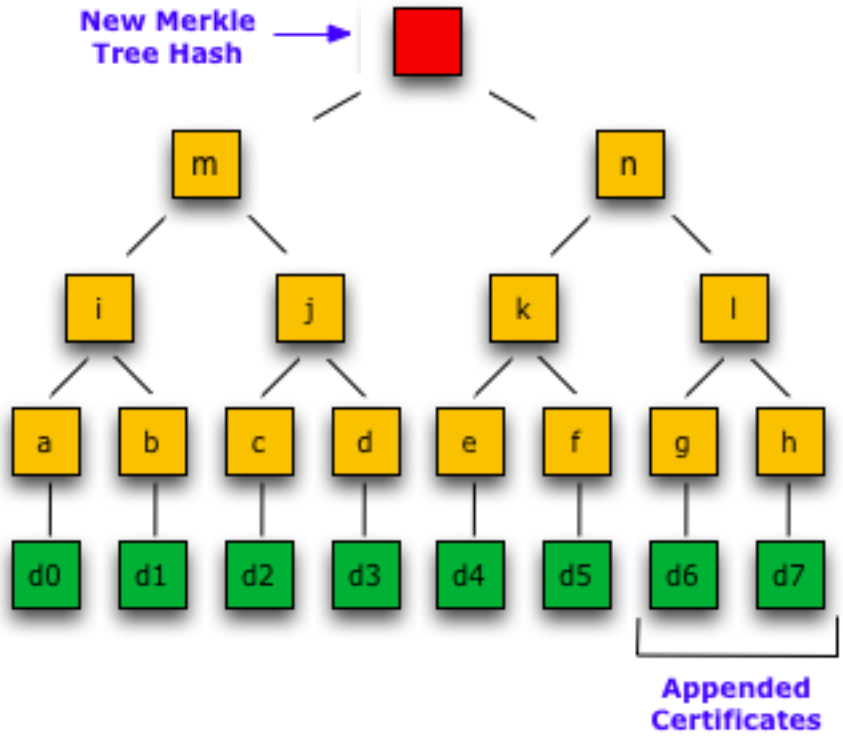
Old hash
New Hash

N1 ⟶ Prove that the old logs are still present ⟶ **N2**

⟵ m; k

Merkle Trees : Consistency Proofs





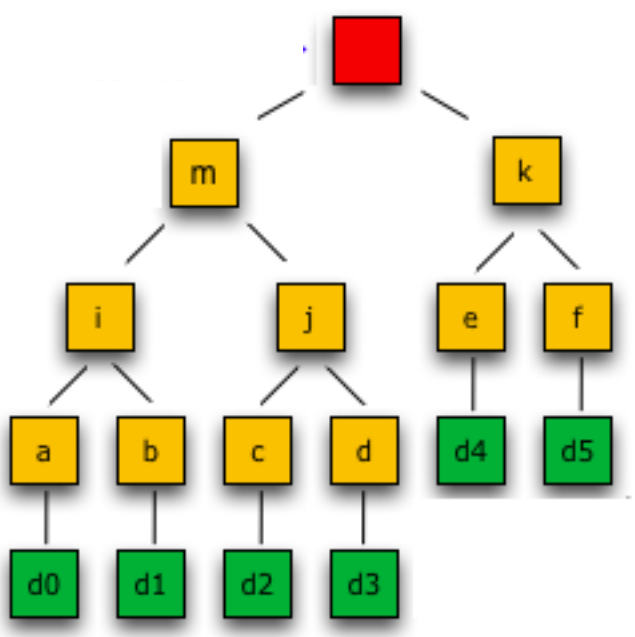
Old hash
New Hash

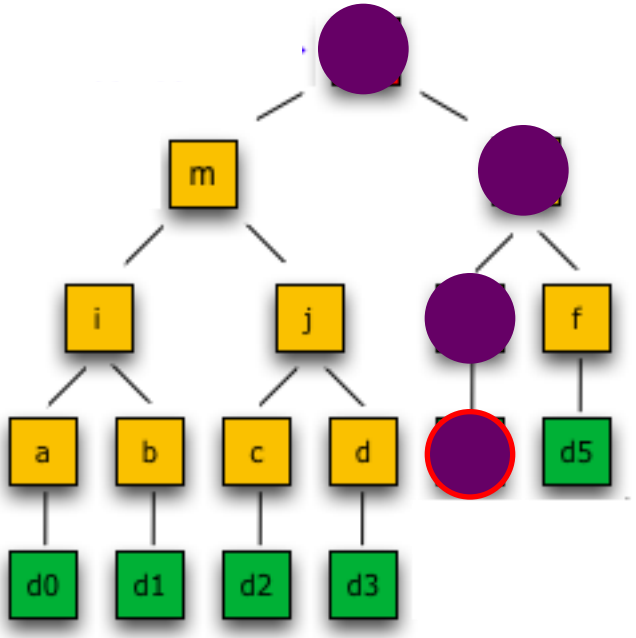
Ordering : Prove that the new entries are strictly appended to the old log

N1 → **N2**

m; k; l

Merkle Trees : Data Synchronization





Traverse from root to leaf towards

Merkle Trees on Edge Devices

- Memory limits

The size of a block in a bitcoin block chain is 1MB

The RAM size in TI's IOT Processor MSP 430 is 66 KB

a mismatch obviously!

Digital Signatures

Alice

Alice's Private Key

sig_K

y = digital signature

Everyone Else

TRUE / FALSE

(x, y)

Message

x = "Attack at Dawn!!"

unsecure channel

ver_K

Alice's Public Key

Signing Function

$y = sig_a(x)$

Input : Message (x) and Alice's private key
Output: Digital Signature of Message

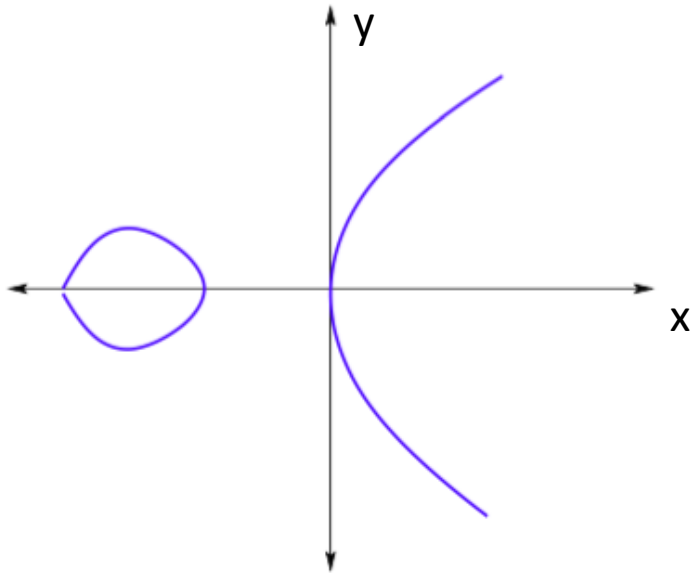
Verifying Function

$ver_b(x, y)$

Input : digital signature, message
Output : true or false
true if signature valid
false otherwise

Elliptic Curve Cryptography

Weierstrass Equation $y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$



x and y and constants all belong to a field of say rational numbers, complex numbers, finite fields (F_p) or Galois Fields ($GF(2^n)$).

Scalar Multiplication

- Given a point P on the curve and a scalar k, scalar multiplication is defined as
  $Q = kP = P + P + P + P \ldots$ (k times)

- Given kP it is easy to compute Q
- Given Q and P it is difficult to find what k is

Computing kP is easy

Double and Add Multiplier

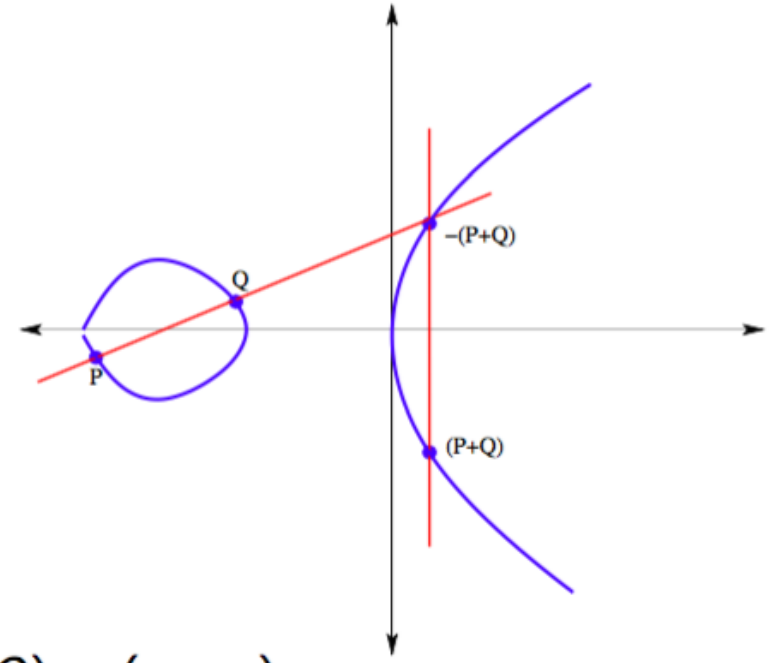
$22P$

$22 = (10110)_2$

bit	Double/Add	Value
1	-	P
0	D	2P
1	D&A	$4P + P = 5P$
1	D&A	$10P + P = 11P$
0	D	22P

Point Operations

- Point Addition



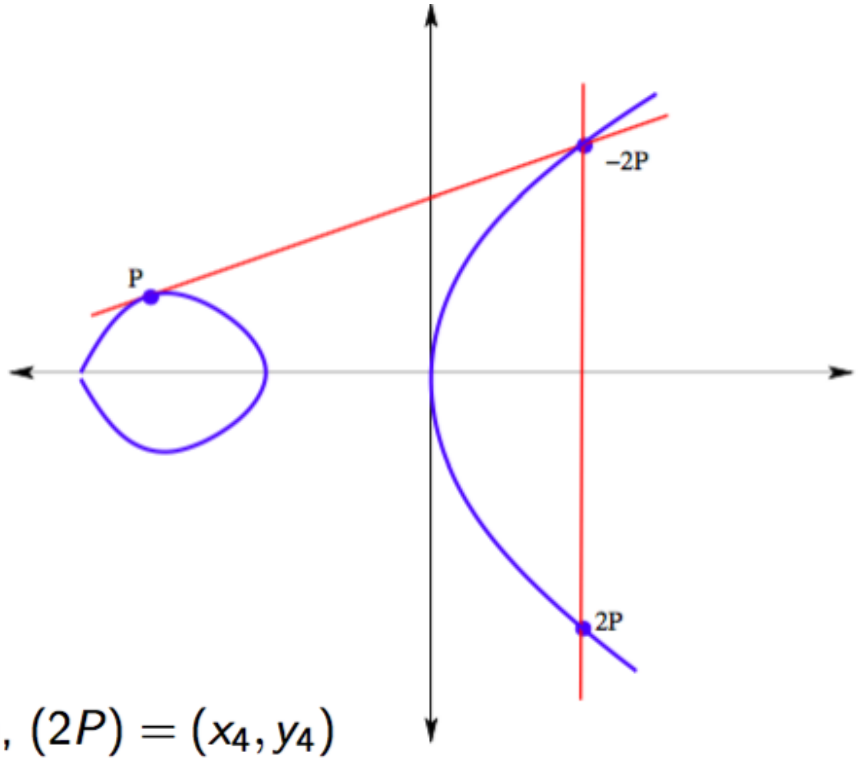
- Equation for Point Addition
  $P = (x_1, y_1)$, $Q = (x_2, y_2)$ and $(P + Q) = (x_3, y_3)$

$$x_3 = \lambda^2 + \lambda + x_1 + x_2 + a$$
$$y_3 = \lambda(x_1 + x_3) + x_3 + y_1$$

where $\lambda = (y_1 + y_2)/(x_1 + x_2)$.

Point Operations

- Point Doubling



- Equation for Point Doubling, $P = (x_1, y_1)$, $(2P) = (x_4, y_4)$

$$x_3 = \lambda^2 + \lambda + a = {x_1}^2 + \frac{b}{{x_1}^2}$$

$$y_3 = {x_1}^2 + \lambda x_3 + x_3$$

where $\lambda = x_1 + (y_1/x_1)$.

ECC on Edge Devices

- 256 bit multi-precision operations to be performed
- Significant memory requirements
- Computationally intensive
  (20million clock cycles to compute a single digital signature on an AVR micro-controller (approx 1.25 seconds))

The way ahead

- Leverage the light-weight properties of PUFs to authenticate
- Light weight crypto
- Batch processing
- More powerful edge devices
  (cost factor and power consumption)
- Move the crypto up the hierarchy at the cost of security