RSA Accumulator

Oct 29, 2019

Overview

- Definitions
- modulus math
- RSA Accumulator
- Hash to prime
- Efficient algorithms (Batching)
- Trusted Setup problem
- Class Group accumulators

Terminology

- **Accumulator**: “A cryptographic accumulator is a primitive that produces a short binding commitment to a set of elements together with short membership/non-membership proofs for any element in the set.”

- **Dynamic Accumulator**: “Accumulator which supports addition/deletion of elements with O(1) cost, independent of the number of accumulated elements”

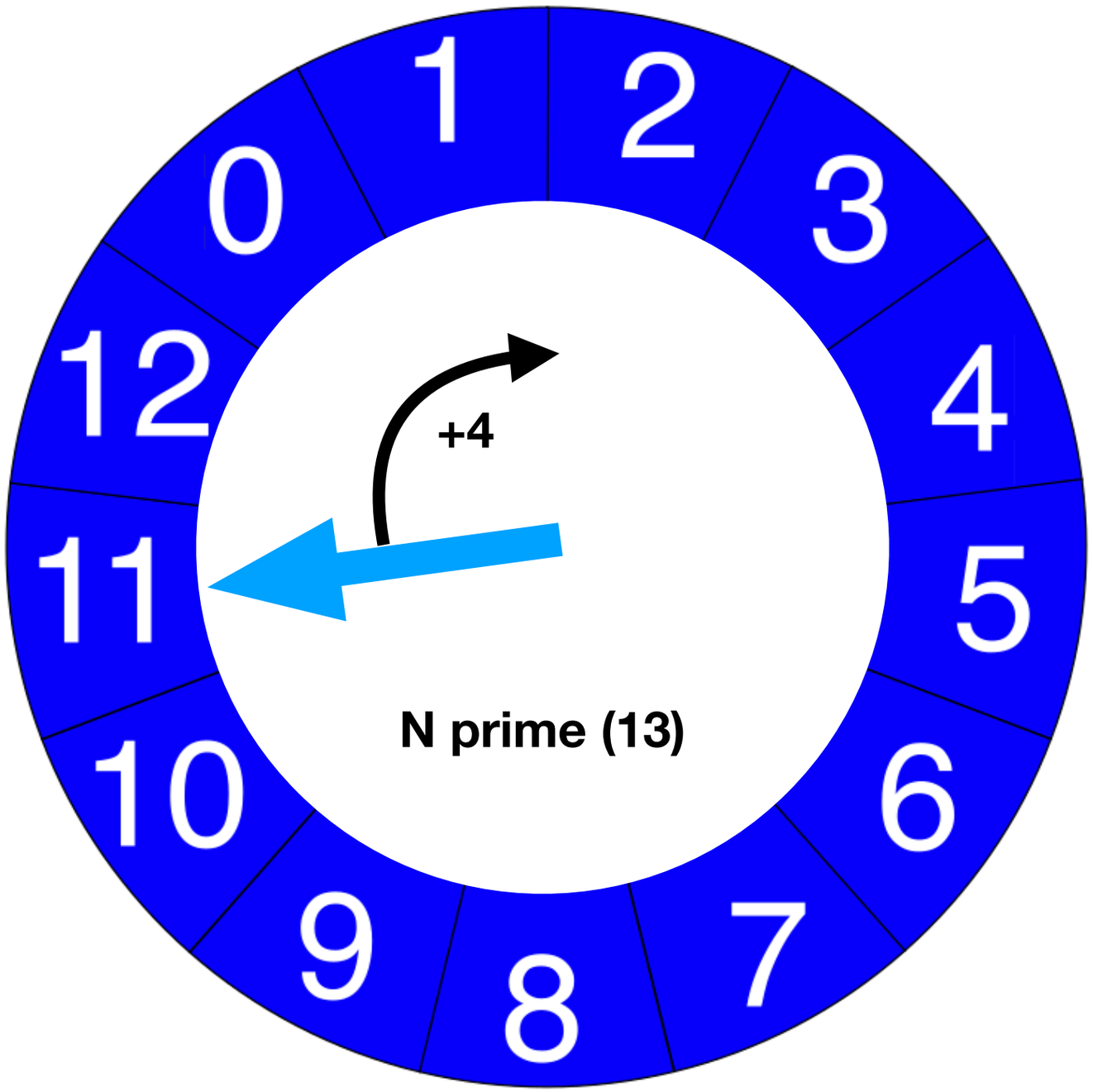
- **Universal Accumulator**: “Dynamic Accumulator which supports membership and non-membership proofs”

— D. Boneh, B. Bünz, B. Fisch,
“Batching Techniques for Accumulators with Applications to IOPs and Stateless Blockchains”, 2018

Accumulator

- What exactly do we need for an accumulator?
  - Base value (i.e. Merkle Tree root) **= Accumulator**
  - Either
    - the set of inputs (to generate a membership proof *on-the-fly* when needed)
    - or the set of membership proofs for each element **= Witness**

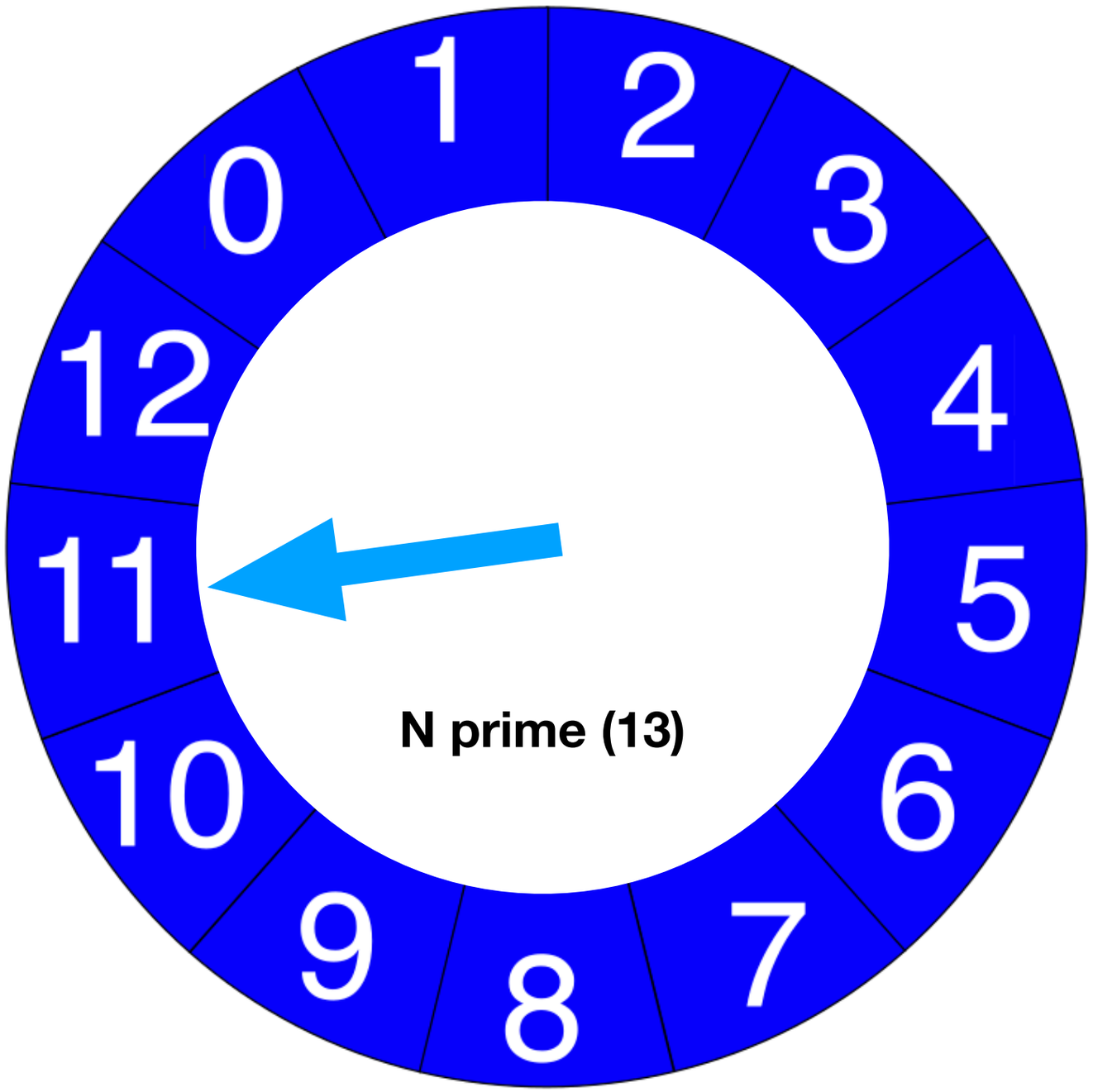
Module Math



Addition, Multiplication, etc. all well defined

$$(a+b) \mod N = \left((a \mod N) + (b \mod N) \mod N\right)$$

Module Math



A generator is an element x so that $\{x,2x,3x,\ldots\}$ produce all elements.

E.g.: $x = 4$

$\Rightarrow 4,8,12,3,7,11,2,6,10,1,5,9,0$

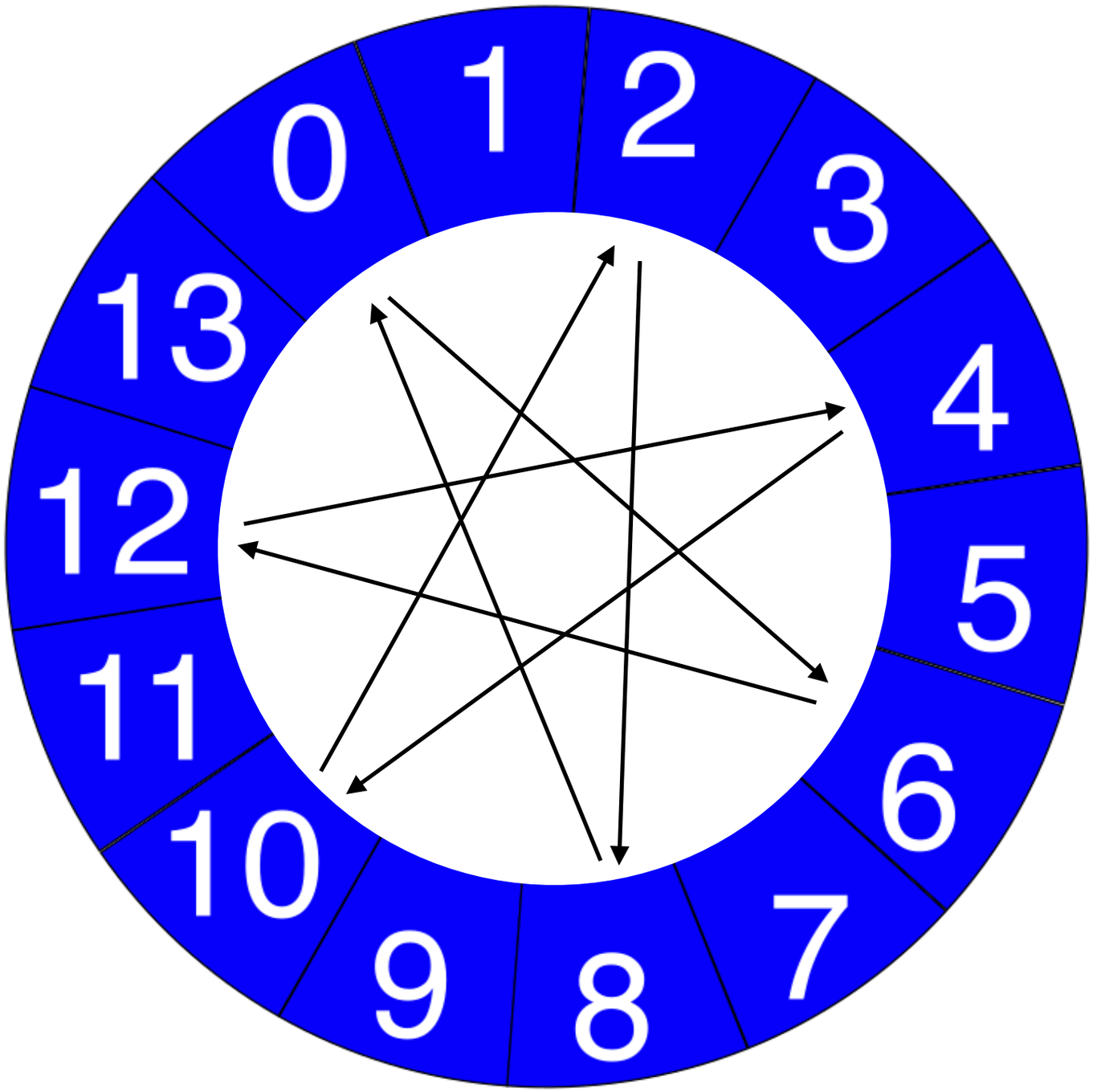
Number of generators called

$$\Phi(N)$$

If N is prime, then $\Phi(N) = N - 1$ (every number except 0 is generator)

$$(a + b) \mod N = \big((a \mod N) + (b \mod N) \mod N\big)$$

Module Math



N not prime (14)

If N is not prime, some numbers are not generators.

I.e. $x = 6$

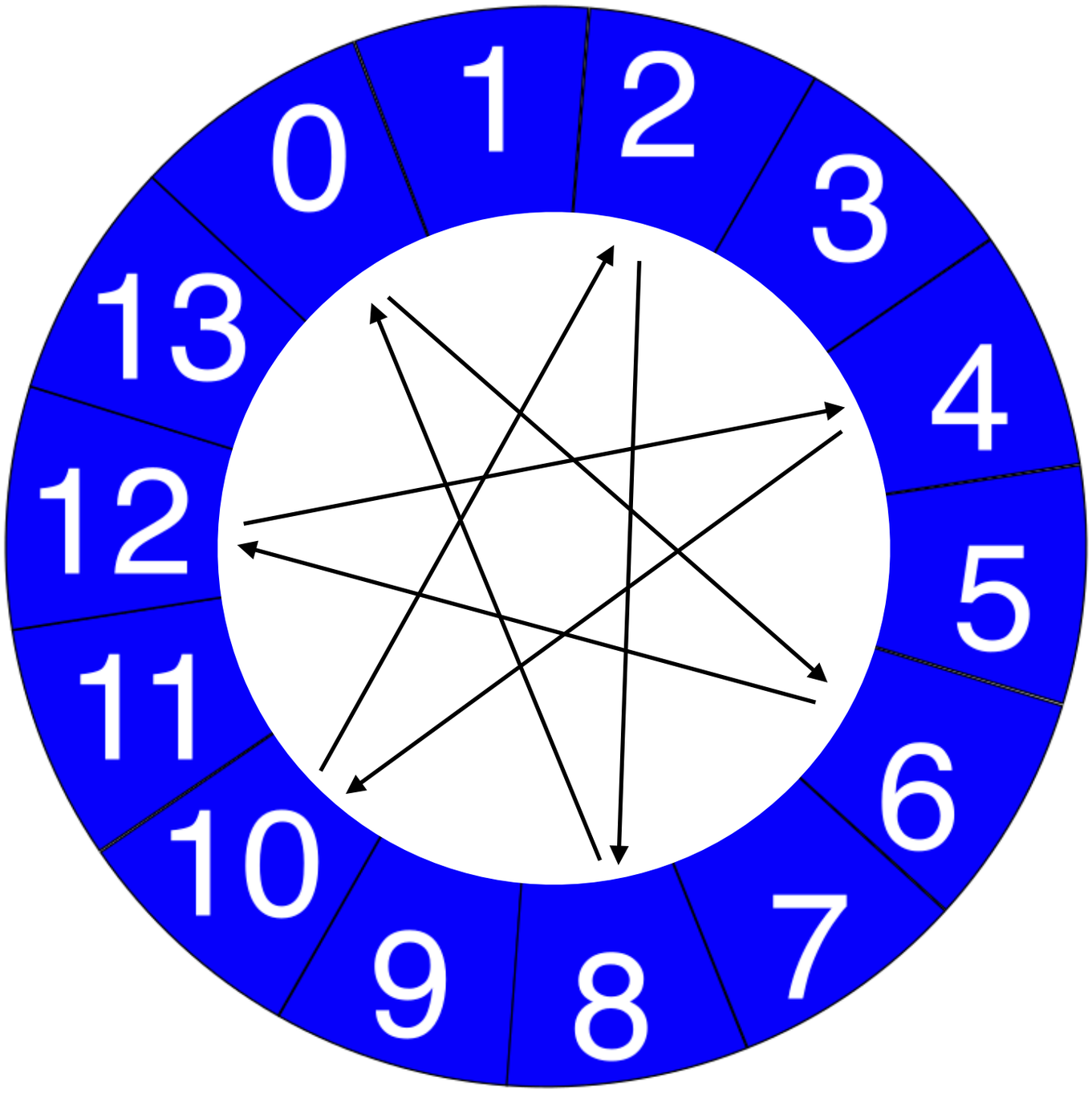
$\Rightarrow 6,12,4,10,2,8,0,6,\ldots$

(1,3,5,7,9,11,13, can not be generated)

Module Math

- If $N = pq$, with p, q prime, then the number of generators is $\Phi(N) = (p-1)(q-1)$

Module Math



N not prime (14)

$$N = 14, p = 2, q = 7, \Phi(14) = 6$$

0: {0}
1: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13}
2: {0, 2, 4, 6, 8, 10, 12}
3: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13}
4: {0, 2, 4, 6, 8, 10, 12}
5: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13}
6: {0, 2, 4, 6, 8, 10, 12}
7: {0, 7}
8: {0, 2, 4, 6, 8, 10, 12}
9: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13}
10: {0, 2, 4, 6, 8, 10, 12}
11: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13}
12: {0, 2, 4, 6, 8, 10, 12}
13: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13}

Module Math
Group with unknown order

- Assume 2 large prime numbers p, q and $n = pq$
  - It is impossible to compute p and q given n
  - Number of generators $\Phi(n) = (p-1)(q-1)$ is secret
- Do all math $\mod n$
- if $\gcd(a, n) = 1$ then $a^{\phi(n)-1} = a^{-1} \mod n$ (used is RSA crypto)
  - E.g. $3^{\phi(14)-1} = 3^{6-1} = 3^5 \mod 14 = 5 \mod 14$
  - 5 is the inverse of $3 \mod 14$, because $3 \cdot 5 = 15 = 1 \mod 14$

Module Math
Group with unknown order

- Assume 2 large prime numbers p, q and $n = pq$
- Number of generators $\Phi(n) = (p-1)(q-1)$
- $\gcd(a, n) = 1 : a^{\phi(n)-1} = a^{-1} \mod n$

- Without p, q, no $\phi(n)$
- Without $\phi(n)$, no inverse
  - and also no roots $a^{\frac{1}{x}} \mod n$

RSA Accumulator

- Using modulo math, assume we have a number $A \in \mathbb{Z}_N$

- Assume we have a hash function that creates a prime number as output $\mathscr{H}_P(\ldots)$

- Then $A' = A^{\mathscr{H}_p(\text{document})}$ is a RSA-Accumulator

RSA Acumulator

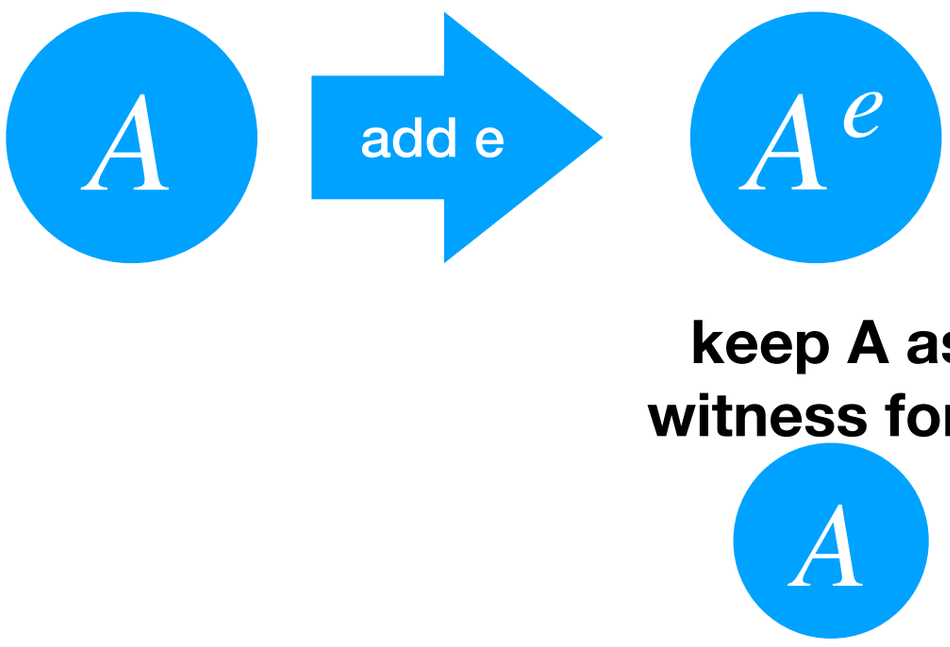
- Init: Empty accumulator $A \xleftarrow{\text{random}} \mathbb{Z}_n$
- Add an element $A_{\text{new}} = A^e \mod n$ (if e is prime)
- Witness: $A^{\frac{1}{e}}$, because $\left(A^{\frac{1}{e}}\right)^e = A$
  - The accumulator without the element is the witness
  - Verify by adding the element and check for equality

RSA Accumulator

- If the order is unknown, $A^{\frac{1}{e}}$ can not be computed
- When adding an element, keep the old accumulator as witness
    - When adding further elements, update the witnesses

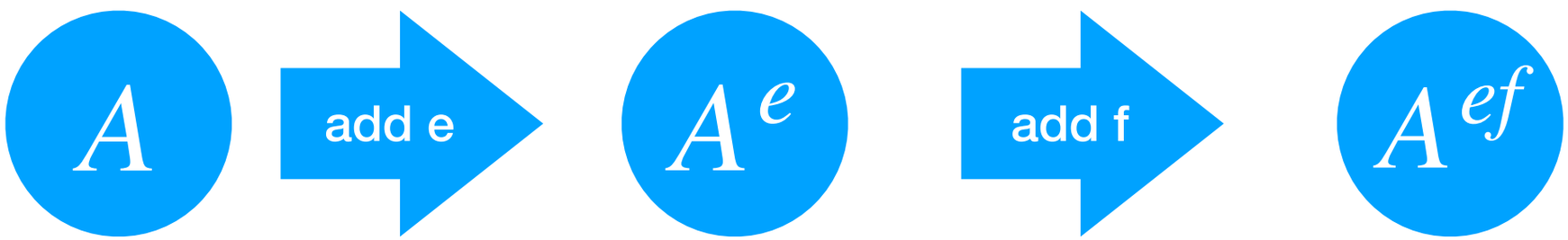
Witness

Adding element e to accumulator A

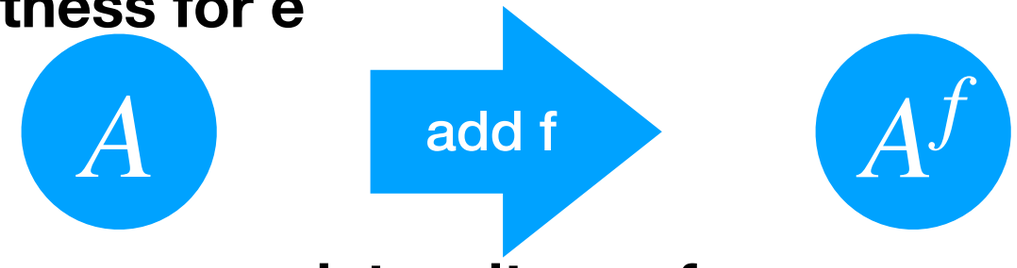


Witness

Adding element f to accumulator A'



keep A as witness for e

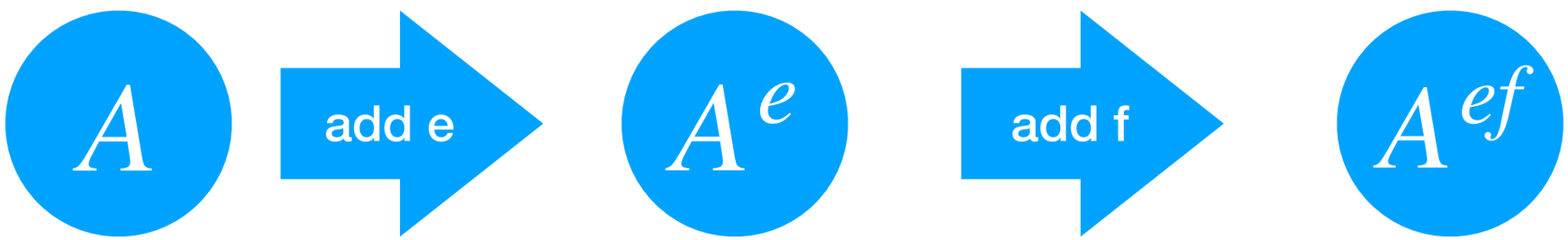


update witness for e

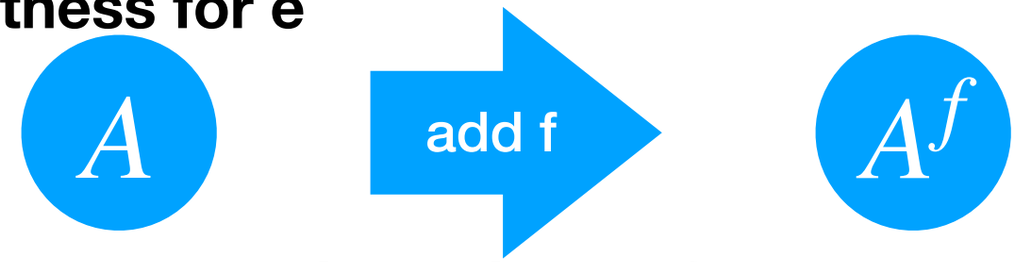
keep A^e as witness for f

Witness

Adding element f to accumulator A'



keep A as witness for e



update witness for e

keep A^e as witness for f

Verify:

$$\left(A^f\right)^e = A^{ef}$$

$$\left(A^e\right)^f = A^{ef}$$

Witnesses

- Accumulator $B = A^{e_1 \cdot e_2 \cdots e_n}$
  - B has accumulated the set $\mathcal{S} = \{e_1, e_2, \ldots, e_n\}$
  - B is a single number (2048 bits), independent of the size of the set $\mathcal{S}$
- A witness W_{e_i} for an element e_i is simply $A^{e_1 \cdots e_{i-1} e_{i+1} \cdots e_n}$
  - a single number
  - Verification via one exponentiation $\left(W_{e_i}\right)^{e_i} \stackrel{?}{=} B$

Hash to prime

- Currently we treated all elements e_i as prime numbers

- We need a hash function that produces primes

Hash to prime

- Currently we treated all elements e_i as prime numbers
- We need a hash function that produces primes
  - The output of a hash is a number
    1. Test for primality.
       - if yes → done
       - if no → hash the output once more. GOTO 1

$\mathscr{H}(e) \rightarrow \mathscr{H}(\mathscr{H}(e)) \rightarrow \mathscr{H}(\mathscr{H}(\mathscr{H}(e))) \rightarrow \ldots$ until prime

Overview so far

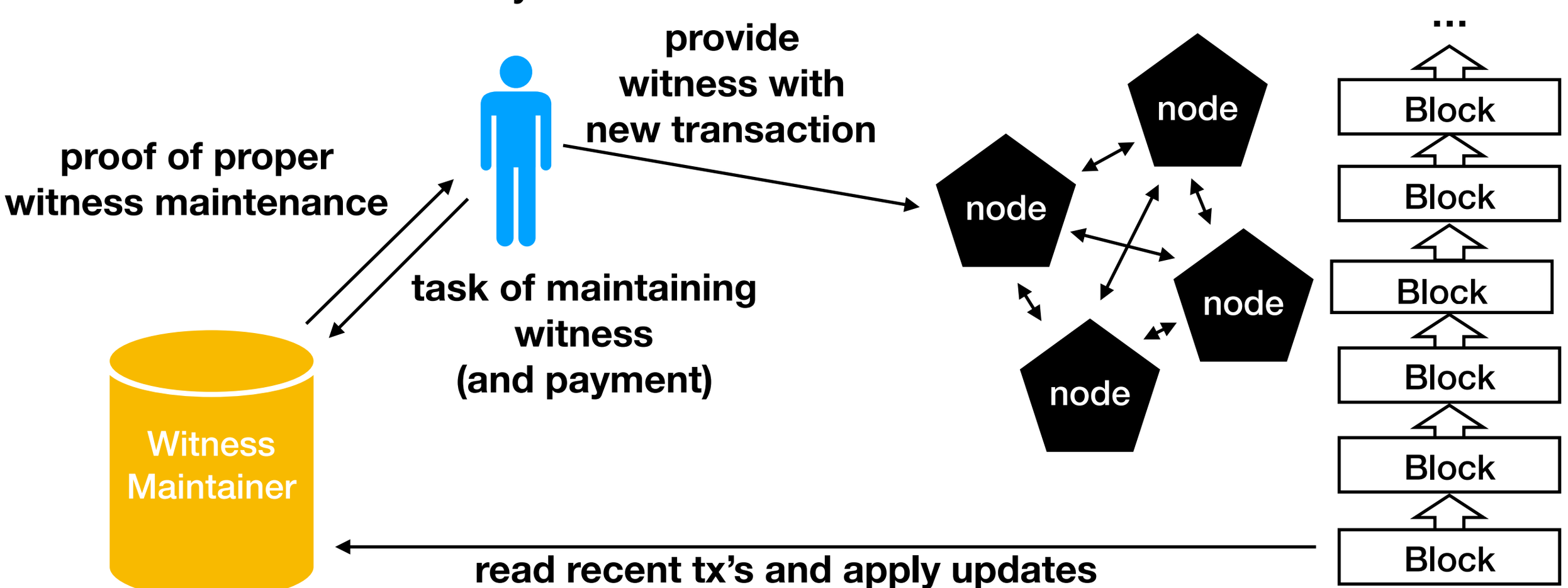
- Blockchain uses an accumulator A as summary of UTXO
  - $A = g^{x_1 \cdots x_m} \mod n$, with $n = pq$ secret primes
- Clients provide witness that their unspent tx output is available
- With every transaction
  - Clients need to update their witnesses
  - Costly and cumbersome

UTXO Replacement

- Theoretically, bitcoin could replace the UTXO set with an RSA Accumulator
- Adding the output of a new transaction: $A^{\mathcal{H}(\text{tx output})}$
- Spending: Prove membership via witness W_{txo}
  - Elements are removed, when output is spend
  - Witness itself is accumulator with the value

Outsourcing work

- A client can outsource the witness keeping to a third party
  - Batching witness maintenance should be cheap
  - A client must be confident that the witness update was done correctly



Outsourcing work

- Batching work is cheaper than individual witnesses maintenance
- Proof of correct computation
    - should be cheaper than redoing the computation

BatchAdd / BatchDel

- Assume we have
  - An accumulator A
  - a set of accumulated elements $\{x_1, x_2, \ldots\}$
    - For each element x_i a witness W_{A,x_i}
- Now, we add (delete) an element. How many operations?
  - Add: exponentiate each witness $O(n)$
  - Delete: Recreate each witness with the new set $O(n^2)$

BatchAdd / BatchDel

- Accumulator A, accumulated elements $\{x_1, x_2, \ldots\}$
- For each element x_i a witness W_{A,x_i}

- With BatchAdd / BatchDel:
  - Store set of base elements $\{x_1, x_2, \ldots\}$
  - Compute jointly $\{W_{A,x_1}, W_{A,x_2}, \ldots, W_{A,x_n}\}$ in $O(n \log(n))$
    - Per element cost of $O(\log(n))$

https://eprint.iacr.org/2018/1188.pdf

BatchAdd / BatchDel

- The function **RootFactor** takes as input a base number g and a set of elements $x_1, x_2, \ldots, x_n$ and returns the list of all witnesses $g^{x_2 x_3 \cdots c_n}, g^{x_1 x_3 x_4 \cdots x_n}, \ldots, g^{x_1 x_2 \cdots x_{n-1}}$

- Run time $O(n \log(n))$

RootFactor$(g, x_1, \ldots, x_n)$:

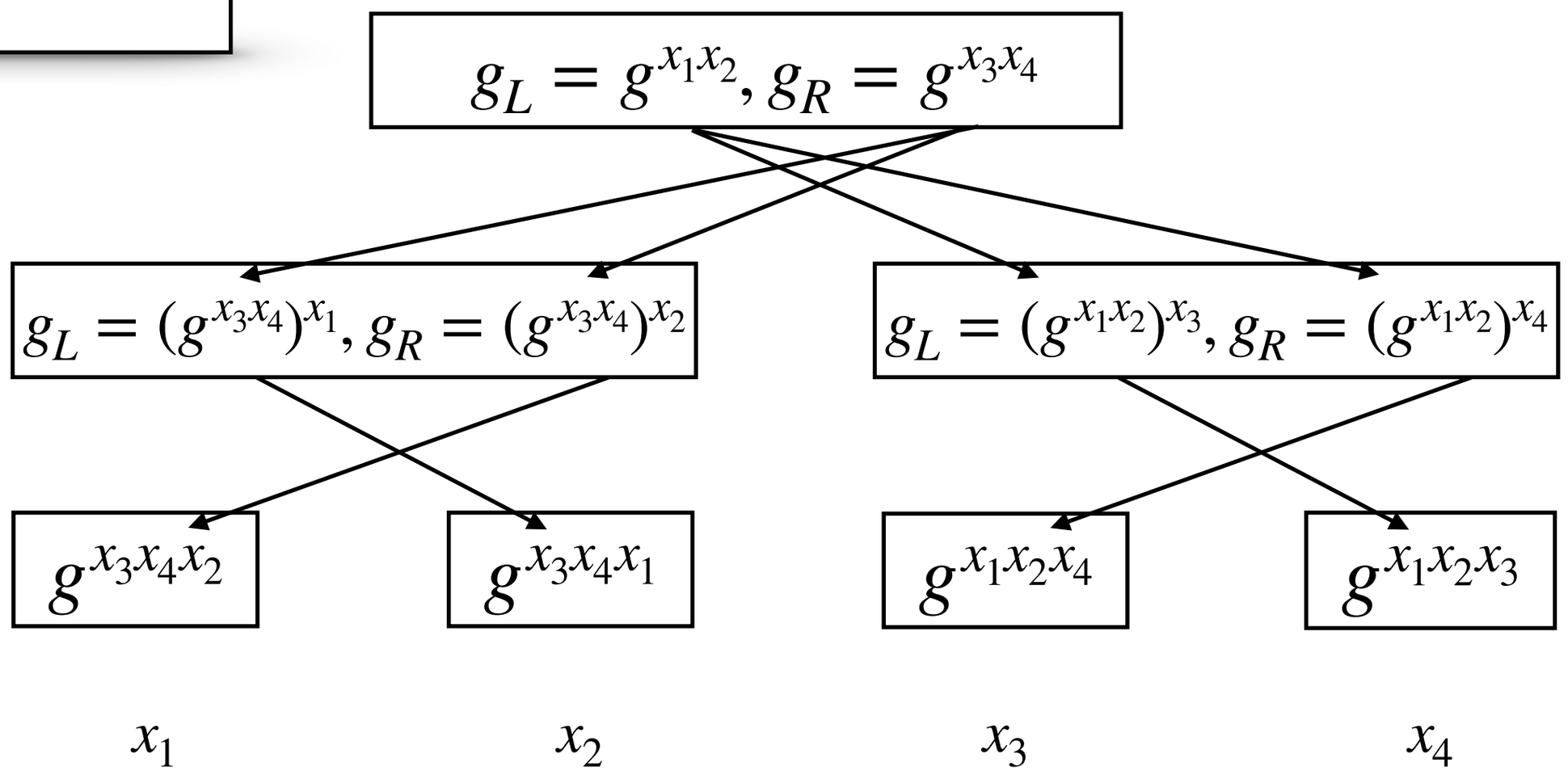
1. **if** $n = 1$ **return** g
2. $n' \leftarrow \lfloor \frac{n}{2} \rfloor$
3. $g_L \leftarrow g^{\prod_{j=1}^{n'} x_j}$
4. $g_R \leftarrow g^{\prod_{j=n'+1}^{n} x_j}$
5. $L \leftarrow$**RootFactor**$(g_R, x_1, \ldots, x_{n'})$
6. $R \leftarrow$**RootFactor**$(g_L, x_{n'+1}, \ldots, x_n)$
7. **return** $L \parallel R$

BatchAdd / BatchDel

RootFactor$(g, x_1, \ldots, x_n)$:

1. **if** $n = 1$ **return** g
2. $n' \leftarrow \lfloor \frac{n}{2} \rfloor$
3. $g_L \leftarrow g^{\prod_{j=1}^{n'} x_j}$
4. $g_R \leftarrow g^{\prod_{j=n'+1}^{n} x_j}$
5. $L \leftarrow$**RootFactor**$(g_R, x_1, \ldots, x_{n'})$
6. $R \leftarrow$**RootFactor**$(g_L, x_{n'+1}, \ldots, x_n)$
7. **return** $L \parallel R$

$$\{x_1, x_2, x_3, x_4\}$$



Proof of correct computation

- Client computes $x^* = x_1 x_2 \cdots c_m$
- Database maintainer computes $A' = A^{x^*}$ and transmits
  - A'
  - Proof of exponentiation for (x^*, A, A') so that $A' = A^{x^*}$

Proof of Exponentiation

for (x^*, A, A') so that $A' = A^{x^*}$

Prover

compute $q = \lfloor \frac{x^*}{l} \rfloor$, residue r,

so that $x^* = ql + r$

send $Q = A^q \mod n$

Verifier

send l, random prime

Compute $r = (x \mod l)$

Accept if $Q^l A^r = A' \mod n$

- $Q^l A^r = \left(A^q\right)^l A^r = A^{ql+r} = A^{x^*} = A'$
- Main work is done in computing A^q

Proof of Exponentiation

for (x^*, A, A') so that $A' = A^{x^*}$

- Verifier
  - send l, random prime. Assume $l \in 0 \ldots 2^{\lambda}$
  - receives q,
  - compute $r = (x \mod l)$
- Accept if $\left(A^q\right)^l A^r = A' \mod n$

- Computing $(x \mod l)$, Q^l, A^r is much cheaper than A^{x^*}
- Computing A^{x^*} takes λ^3 times as long as $(x \mod l)$

Trusted Setup

- p, q are toxic waste secrets
- we can use old factorization problems
  - we trust that the factors p, q have been forgotten
    - E.g. RSA Factoring Challenge

https://en.wikipedia.org/wiki/RSA_Factoring_Challenge

- Win 200000$ if you can factor

n=25195908475657893494027183240048398571429282126204032027777137836043662020707595556264018525880784406918290641249515082189298559149176184502808489120072844992687392807287776735971418347270261896375014971824691165077613379859095700097330459748808428401797429100642458691817195118746121515172654632282216869987549182422433637259085141865462043576798423387184774447920739934236584823824281198163815010674810451660377306056201619676256133844143603833904414952634432190114657544541784240209246165157233507787077498171257724679629263863563732899121548314381678998850404453640235273819513786365643912120103971228221207203 57

Class Group Accumulators

Class Group accumulators work similarly, no trusted setup

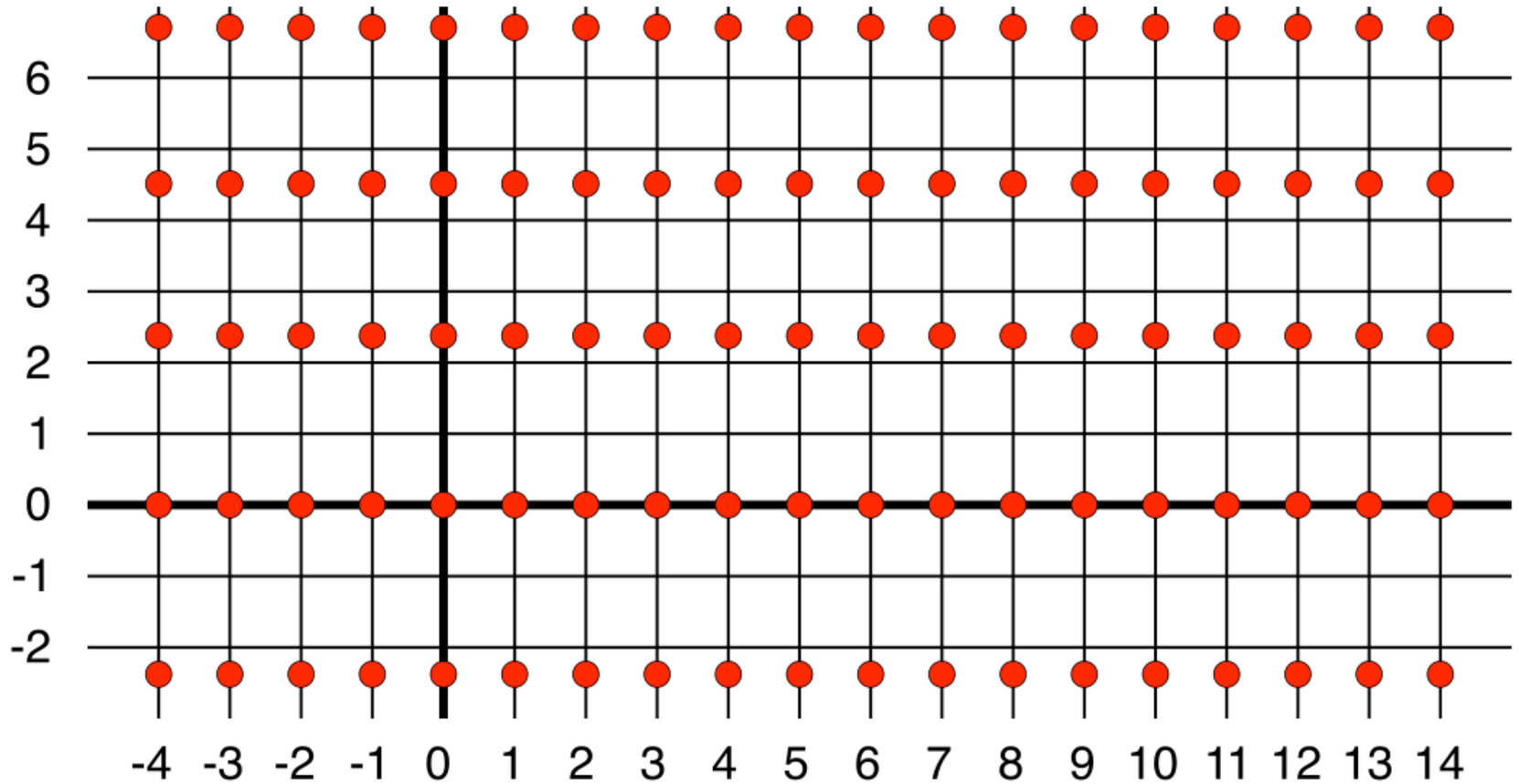
- Consider the ring of integers of a quadratic extension $\mathbb{Q}\left(\sqrt{-p}\right)$ with p large prime and $p = 3 \mod 4$
- The set of all fractions $x + y\sqrt{-p}$ were $a + b\frac{1+\sqrt{-p}}{2}$

Class Group Accumulators

Example: $\mathbb{Q}\left(\sqrt{-5}\right)$

- In this ring we observer no unique factorization

$$6 = 2 \cdot 3 = \left(1 + \sqrt{-5}\right)\left(1 - \sqrt{-5}\right)$$



Elements of $\mathbb{Q}\left(\sqrt{-5}\right)$

Class Group Accumulators

Class Group accumulators work similarly, no trusted setup

- Ideal: all numbers generated by a multiplication of a base elements $(\alpha_1, \ldots, \alpha_k) = \{c_1\alpha_1 + c_2\alpha_2 + \ldots + c_k\alpha_k\}$
  - Ex.: Every number of $\mathbb{Q}\left(\sqrt{-5}\right)$ can be generated via

$$c_1 \cdot 2 + c_2 \cdot (1 + \sqrt{-5})$$

- Principle Ideals: If the ideal is generated by a single element, i.e. $(2) = 2\mathbb{Z}$ (principle ideals of even numbers)

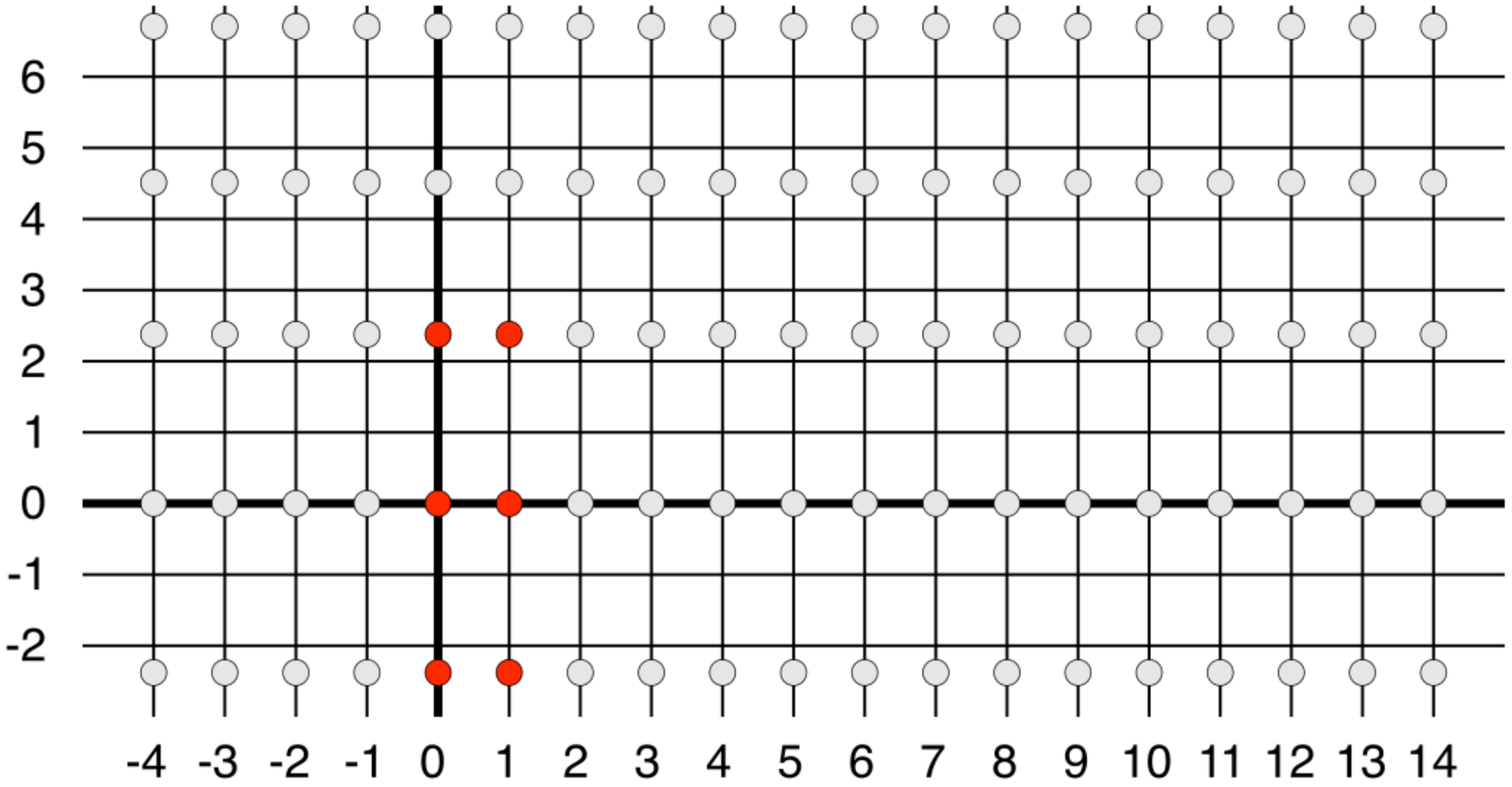
Class Group Accumulators

just like mod

- Class group : $\mathbb{Q}\left(\sqrt{-p}\right)/J$
  - where J is the subgroup of principle ideals in $\mathbb{Q}\left(\sqrt{-p}\right)$

Class Group Accumulators

- Example: $\mathbb{Q}\left(\sqrt{-5}\right), J = (2, 1 + \sqrt{-5})$



Elements of $\mathbb{Q}\left(\sqrt{-5}\right)/J$

This looks the same as the the ring $x \mod 6$

(2 subgroups of order 2 and 3)

Class Group Accumulators

- Pick a random, large integer $p = 3 \mod 4$:
  - Use the elements of the class group of $\mathbb{Q}\left(\sqrt{-p}\right)$
- Group order, n^{th} roots are believed to be hard to compute
- Exponentiation slower, otherwise everything the same

Summary

- Accumulators are can be used to squeeze a large set into a single element
    - Merkle Tree root can be seen as an accumulator
    - Even a blockchain is an accumulator
- Dynamic accumulators: adding and deleting elements
- Efficient accumulators perform adding/deleting in $O(1)$
- As a UTXO replacement, they shift the burden of tracking the UTXO set to the individual users

Remarks

- Several algorithms exist to deal with large numbers of elements
  - Naively, updating M elements requires $O(M)$ steps
    - Intelligently done, only $O(\log M)$ steps are needed
- If the prime values p, q are known, a new witness can be *invented*, since $A^{1/x}$ can be computed easily for any x
  - p, q are called *toxic waste* (trusted setup)