RSA accumulators

Can we reduce the proof size?

- So far all the methods we have seen have proof size at least logarithmic
- Can we reduce the proof size?
- Yes!
- By changing the cryptographic primitive
- Are we loosing anything?

RSA Accumulator

- Exponential accumulation of elements:

$$A = a^{x_1 x_2 \cdots x_n} \bmod N$$

  - $N = pq$ is an RSA modulus
  - a and N are relatively prime
  - Only the client knows p and q, and thus $\phi(N) = (p-1)(q-1)$
  - Each x_i is prime
- The basis is the accumulation A
- Proof of membership of x_i (witness):

$$A_i = a^{x_1 \cdots x_{i-1} x_{i+1} \cdots x_n} \bmod N$$

- Verification:
  - Test $A = A_i^{x_i} \bmod N$
- [Benaloh de Mare]

Accumulator as a Hash Function

- Quasi-commutative hash function

$$h(h(a, x_1), x_2) = h(h(a, x_2), x_1)$$

- Exponential accumulation yields quasi-commutative hash function

$$h(a, x) = a^x \bmod N$$

- Witness verification as hash computation

$$A = A_i^{x_i} \bmod N = h(A_i, x_i)$$

- Collision resistance
  - Given a, x, y difficult to find a' such that

$$h(a, x) = h(a', y)$$

Security

- Why should elements be prime?
  - Witness can be computed for factors of elements
- Why should the factorization of N be kept secret?

Security based on strong RSA assumption:

- Given a modulus *N* of unknown factorization and a base g, it is infeasible to find some e-th root of g mod N.
- How do we prove security based on the above assumption?

Efficiency

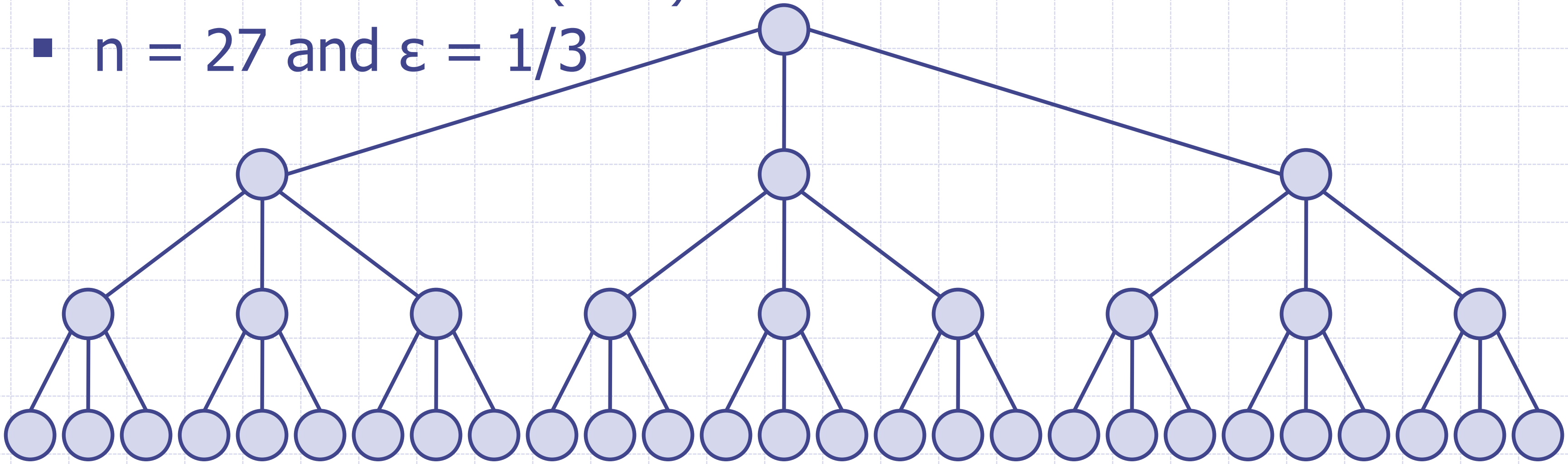
- Size of proof: O(1)
- Time to compute the proof: O(n)
- Time to verify: O(1)
- Time to update: O(1)

OR (precomputed witnesses)

- Size of proof: O(1)
- Time to compute the proof: O(1)
- Time to verify: O(1)
- Time to update: O(n)

Can we make the costs sublinear?

- Set $X = \{x_1, x_2, \ldots, x_n\}$ of elements to authenticate
- Constant $0 < \varepsilon < 1$
- We build a tree $T(\varepsilon)$ on top of the elements:
  - The leaves store $x_1, x_2, \ldots, x_n$
  - The tree has $O(1/\varepsilon)$ levels
  - Every node has $O(n^{\varepsilon})$ children
  - Level i contains $O(n^{1-i\varepsilon})$ nodes
- $n = 27$ and $\varepsilon = 1/3$

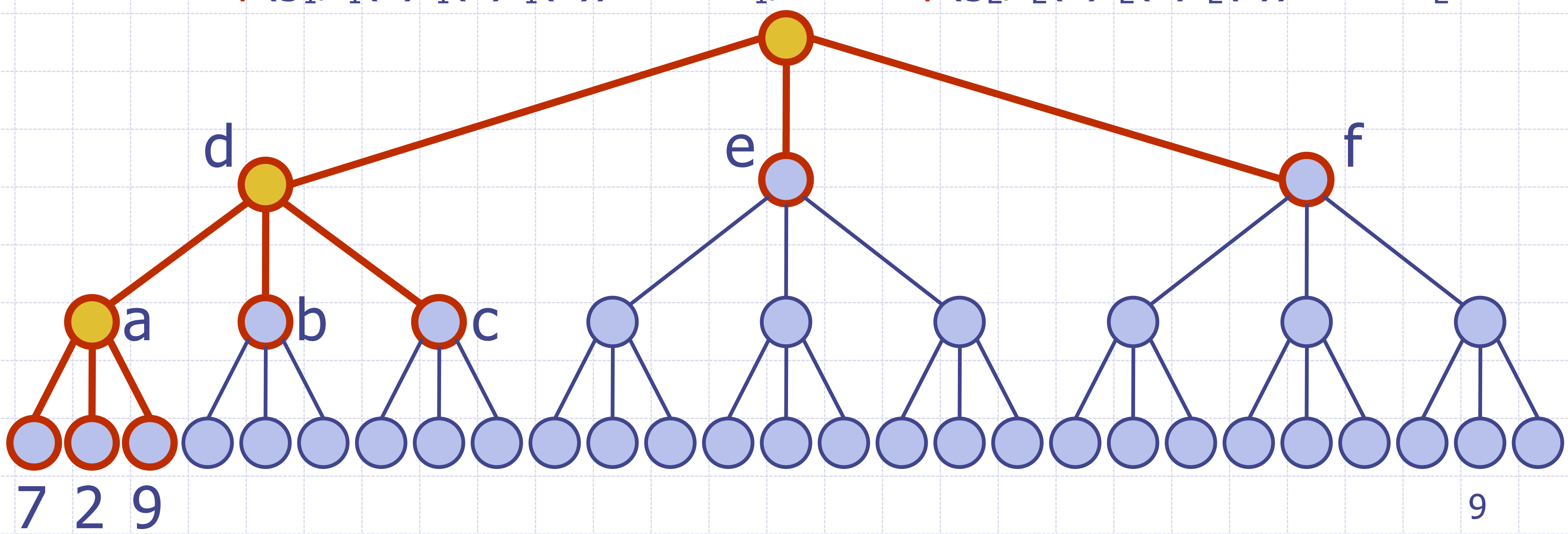


Accumulation tree: digests

- For each level i of the tree
  - N_i: RSA modulus; $g_i \in QR_{N_i}$
  - $r_i(x)$: prime representative of x at level i
  - RSA digest of a node v with children $v_1, v_2, \ldots, v_t$

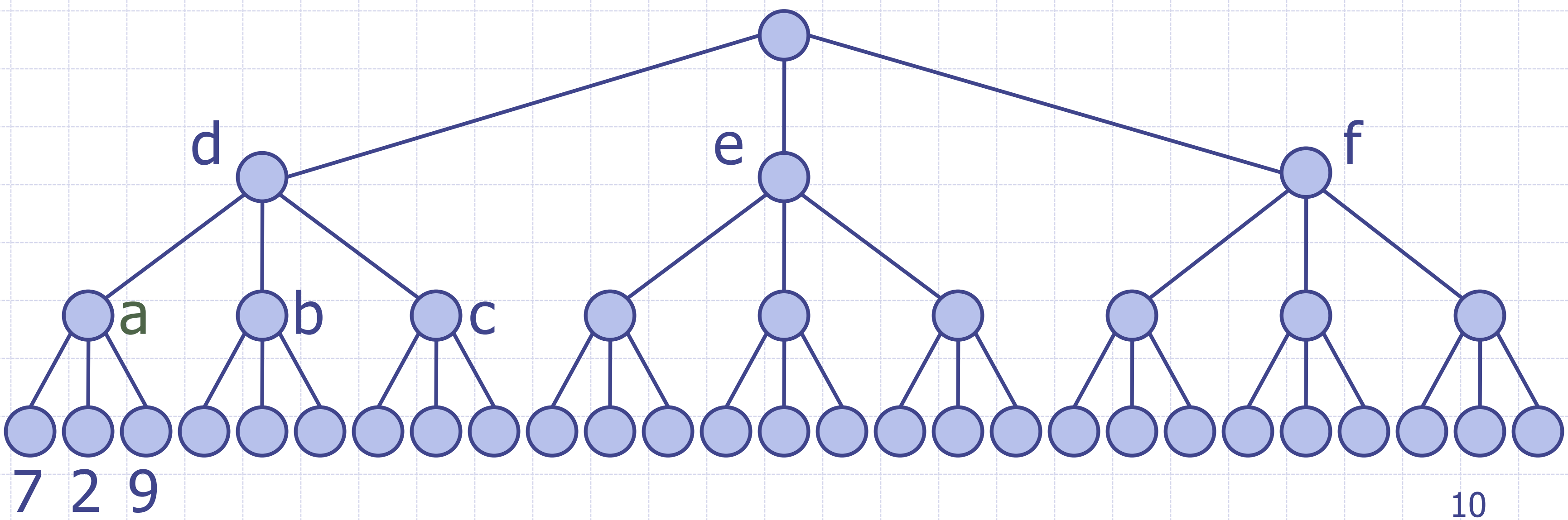
$$d(v) = \mathrm{exp}(g_i, r_i(v_1) r_i(v_2) \ldots r_i(v_t)) \bmod N_i$$

- The digest of set X is the RSA digest of the root
- $a = \mathrm{exp}(g_1, r_1(7) r_1(2) r_1(9)) \bmod N_1$, $d = \mathrm{exp}(g_2, r_2(a) r_2(b) r_2(c)) \bmod N_2$



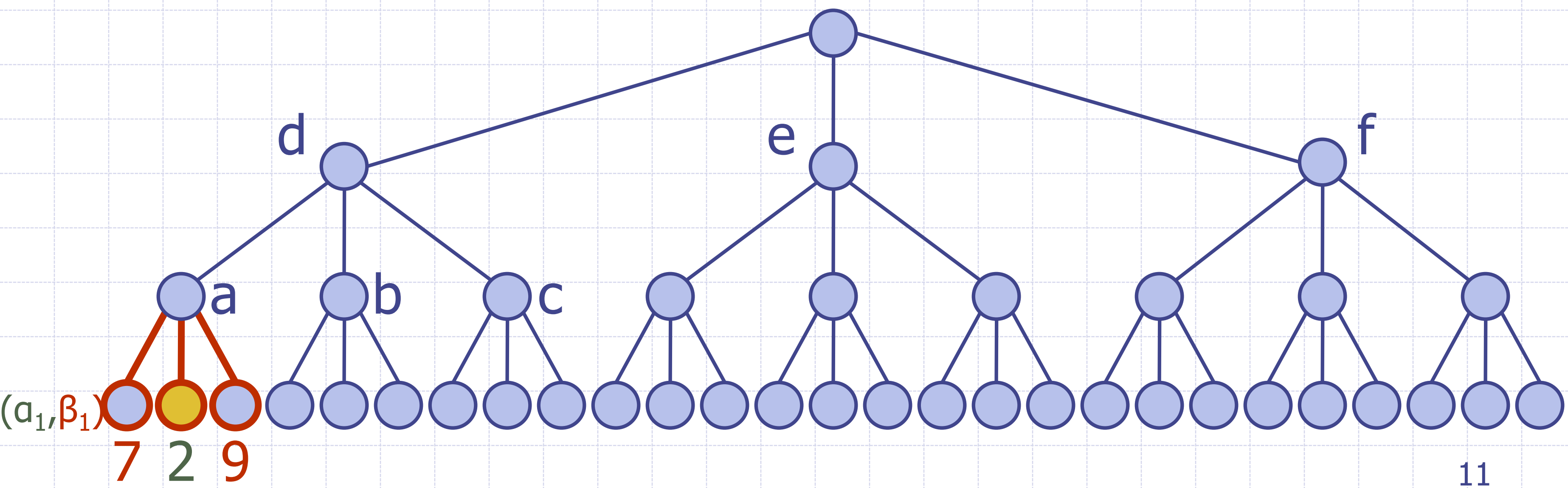
Example: query for element 2

	α (prime)	β (branch witness)



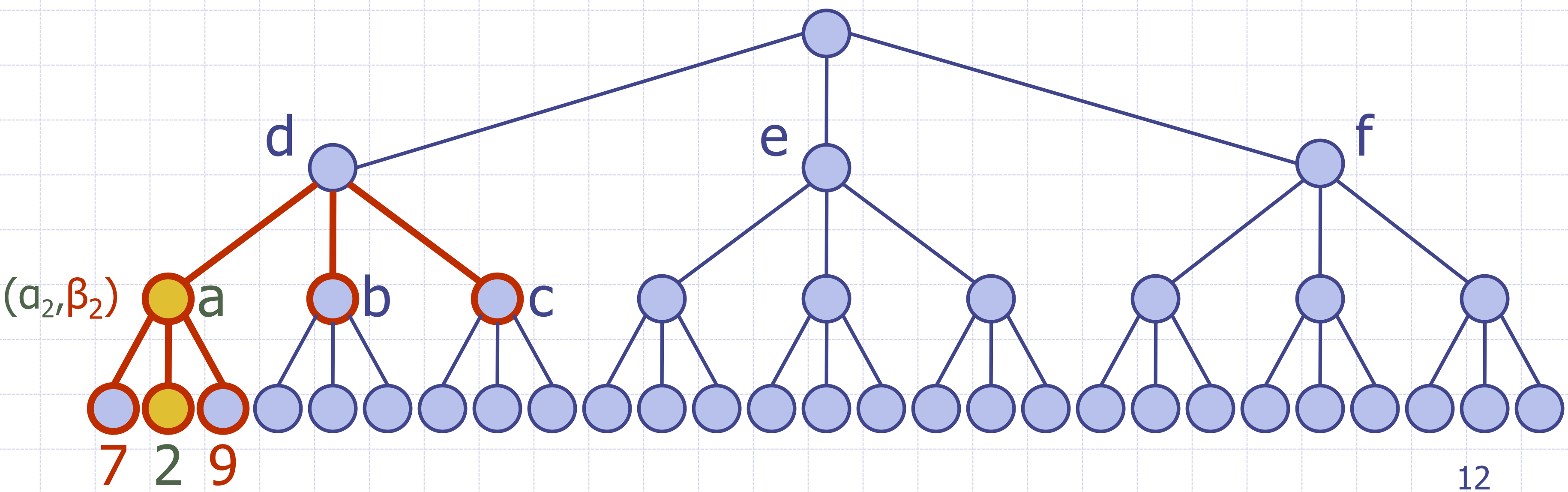
Query: level 1

	α (prime)	β (branch witness)
1	$\alpha_1 = r_1(2)$	$\beta_1 = \exp(g_1, r_1(7) r_1(9)) \bmod N_1$



Query: level 2

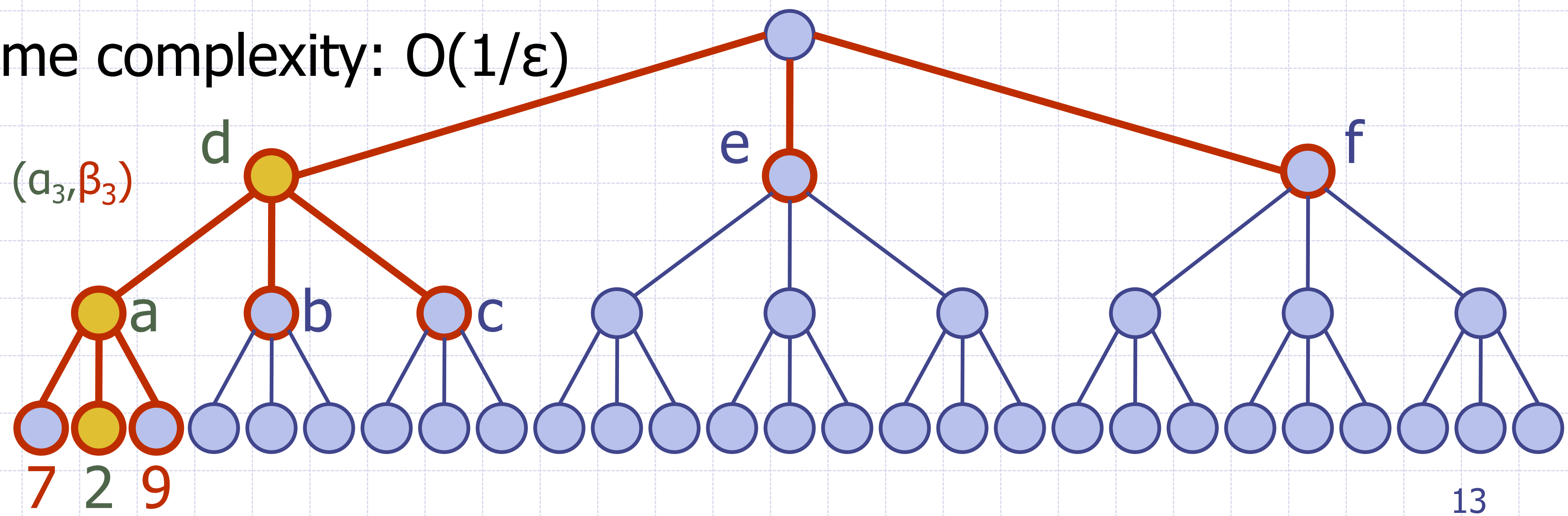
	α (prime)	β (branch witness)
1	$\alpha_1 = r_1(2)$	$\beta_1 = \exp(g_1, r_1(7) r_1(9)) \bmod N_1$
2	$\alpha_2 = r_2(a)$	$\beta_2 = \exp(g_2, r_2(b) r_2(c)) \bmod N_2$



Query: level 3

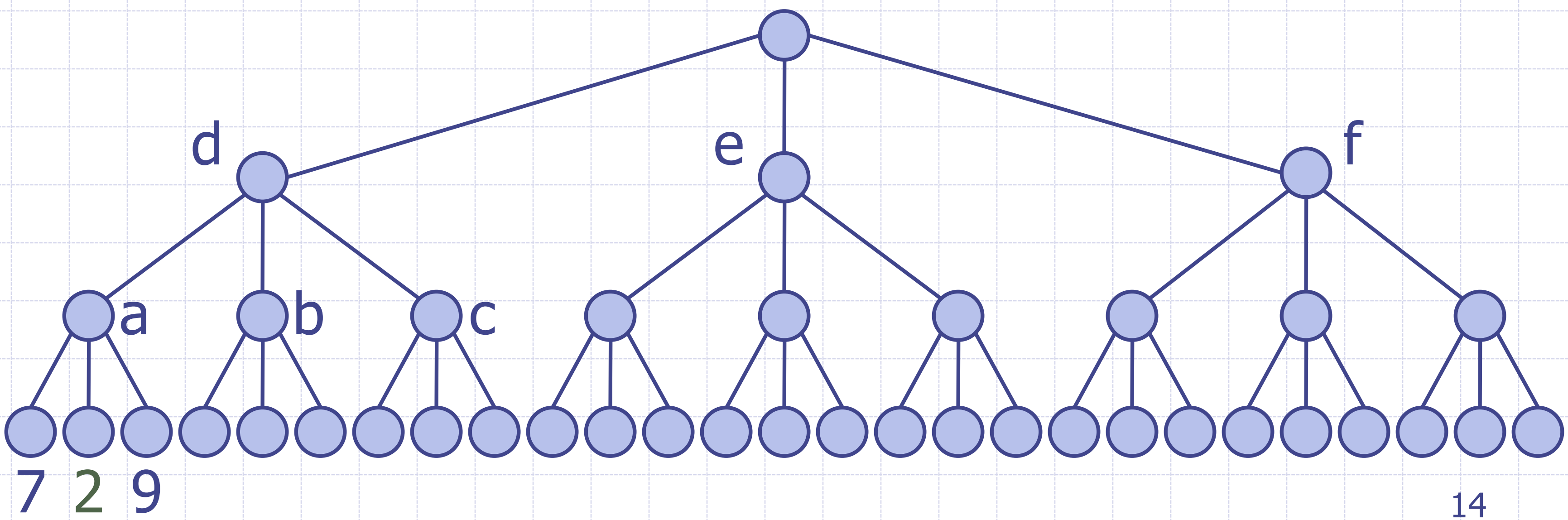
	α (prime)	β (branch witness)
1	$\alpha_1=r_1(2)$	$\beta_1=\exp(g_1,r_1(7)r_1(9)) \bmod N_1$
2	$\alpha_2=r_2(a)$	$\beta_2=\exp(g_2,r_2(b)r_2(c)) \bmod N_2$
3	$\alpha_3=r_3(d)$	$\beta_3=\exp(g_3,r_3(e)r_3(f)) \bmod N_3$

Time complexity: $O(1/\varepsilon)$



Example: verification of element 2

	α (prime)	β (branch witness)
1	$\alpha_1 = r_1(2)$	$\beta_1 = \exp(g_1, r_1(7) r_1(9)) \bmod N_1$
2	$\alpha_2 = r_2(a)$	$\beta_2 = \exp(g_2, r_2(b) r_2(c)) \bmod N_2$
3	$\alpha_3 = r_3(d)$	$\beta_3 = \exp(g_3, r_3(e) r_3(f)) \bmod N_3$

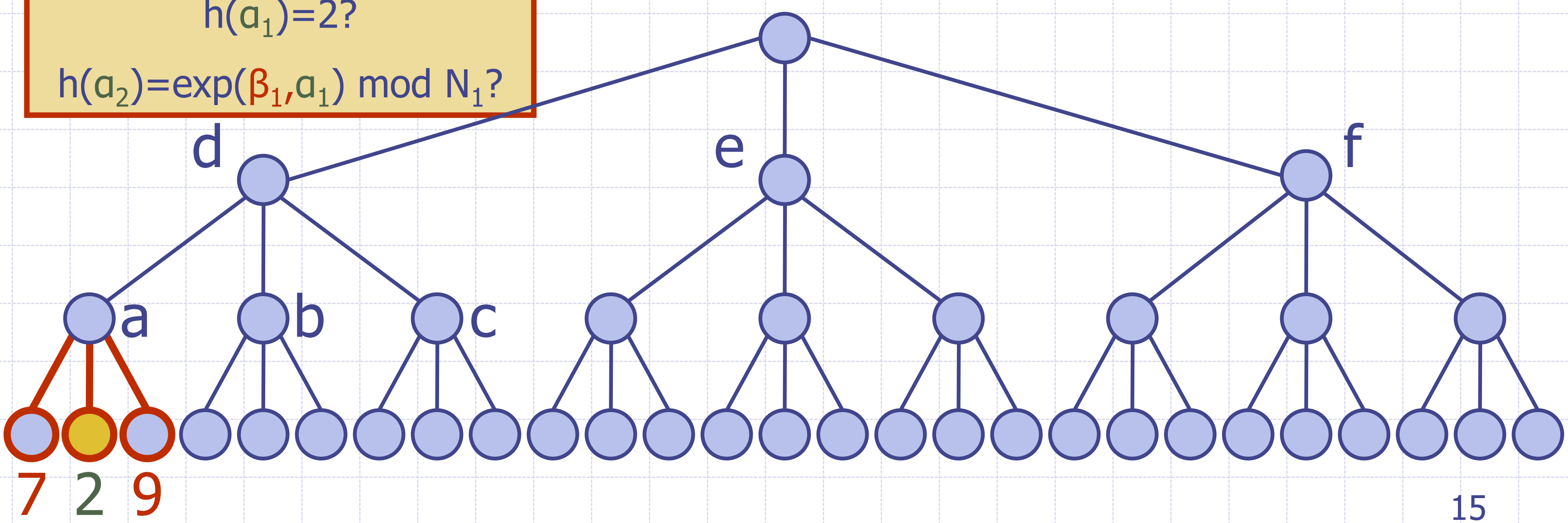


Verification: level 1

	α (prime)	β (branch witness)
1	$\alpha_1=r_1(2)$	$\beta_1= \exp(g_1,r_1(7)r_1(9)) \bmod N_1$
2	$\alpha_2=r_2(a)$	$\beta_2= \exp(g_2,r_2(b)r_2(c)) \bmod N_2$
3	$\alpha_3=r_3(d)$	$\beta_3= \exp(g_3,r_3(e)r_3(f)) \bmod N_3$

$h(\alpha_1)=2$?

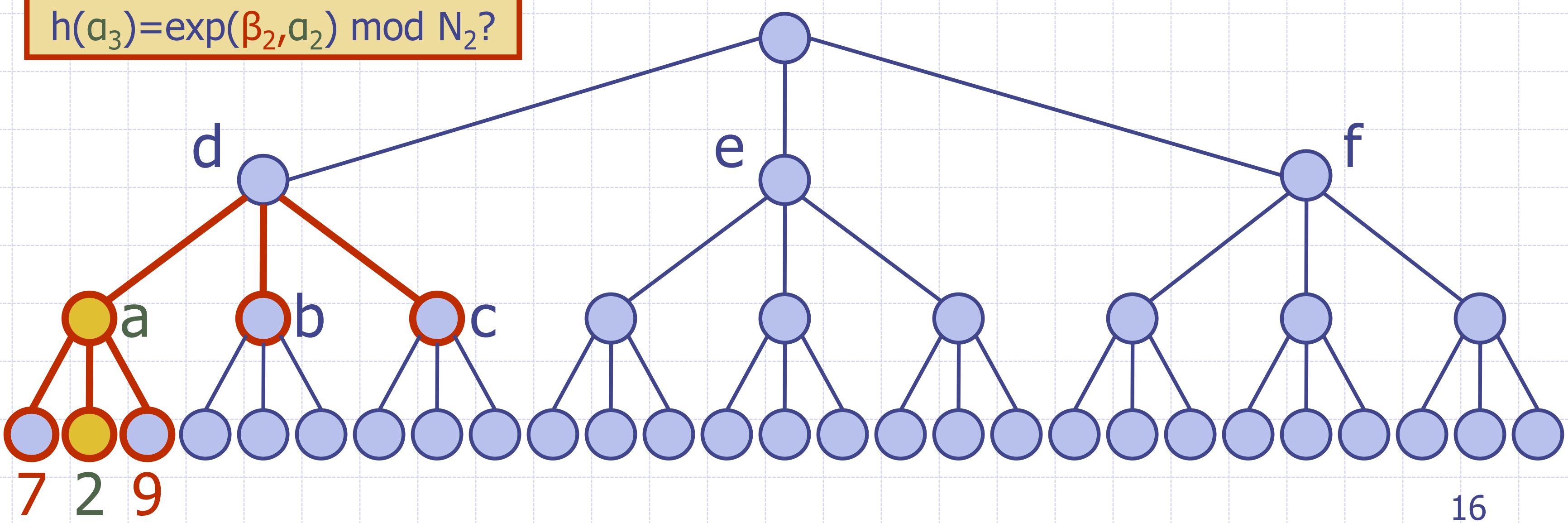
$h(\alpha_2)=\exp(\beta_1,\alpha_1) \bmod N_1$?



Verification: level 2

	α (prime)	β (branch witness)
1	$\alpha_1 = r_1(2)$	$\beta_1 = \exp(g_1, r_1(7) r_1(9)) \bmod N_1$
2	$\alpha_2 = r_2(a)$	$\beta_2 = \exp(g_2, r_2(b) r_2(c)) \bmod N_2$
3	$\alpha_3 = r_3(d)$	$\beta_3 = \exp(g_3, r_3(e) r_3(f)) \bmod N_3$

$h(\alpha_3) = \exp(\beta_2, \alpha_2) \bmod N_2$?

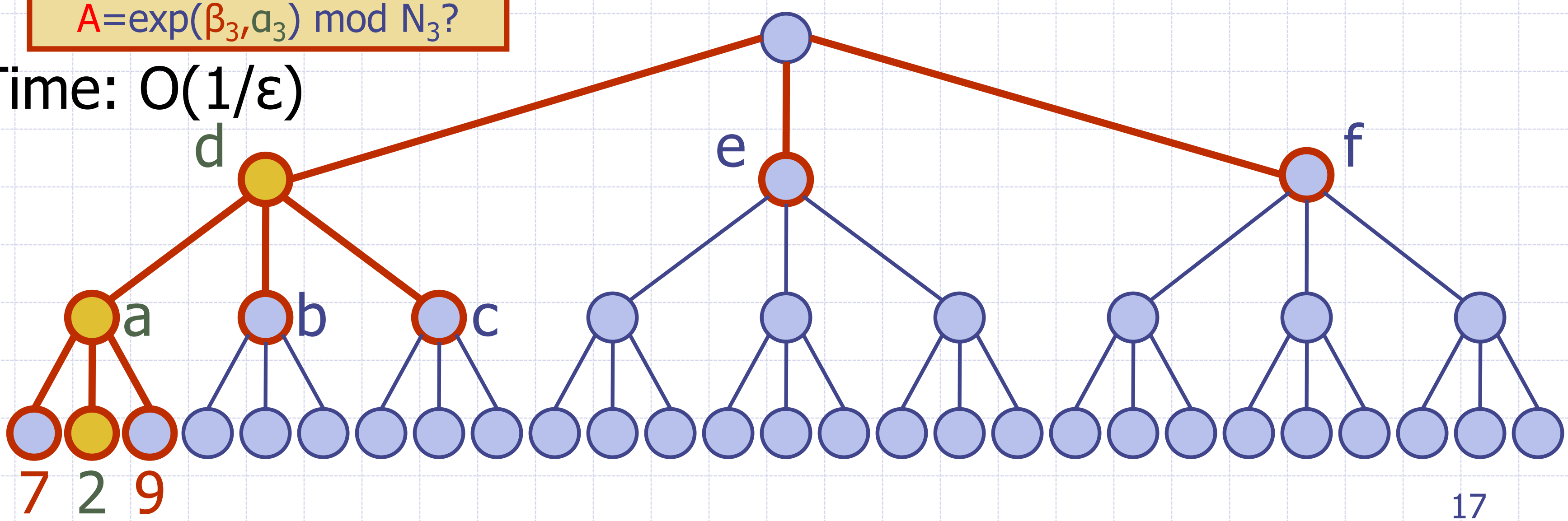


Verification: level 3

	α (prime)	β (branch witness)
1	$\alpha_1 = r_1(2)$	$\beta_1 = \exp(g_1, r_1(7) r_1(9)) \bmod N_1$
2	$\alpha_2 = r_2(a)$	$\beta_2 = \exp(g_2, r_2(b) r_2(c)) \bmod N_2$
3	$\alpha_3 = r_3(d)$	$\beta_3 = \exp(g_3, r_3(e) r_3(f)) \bmod N_3$

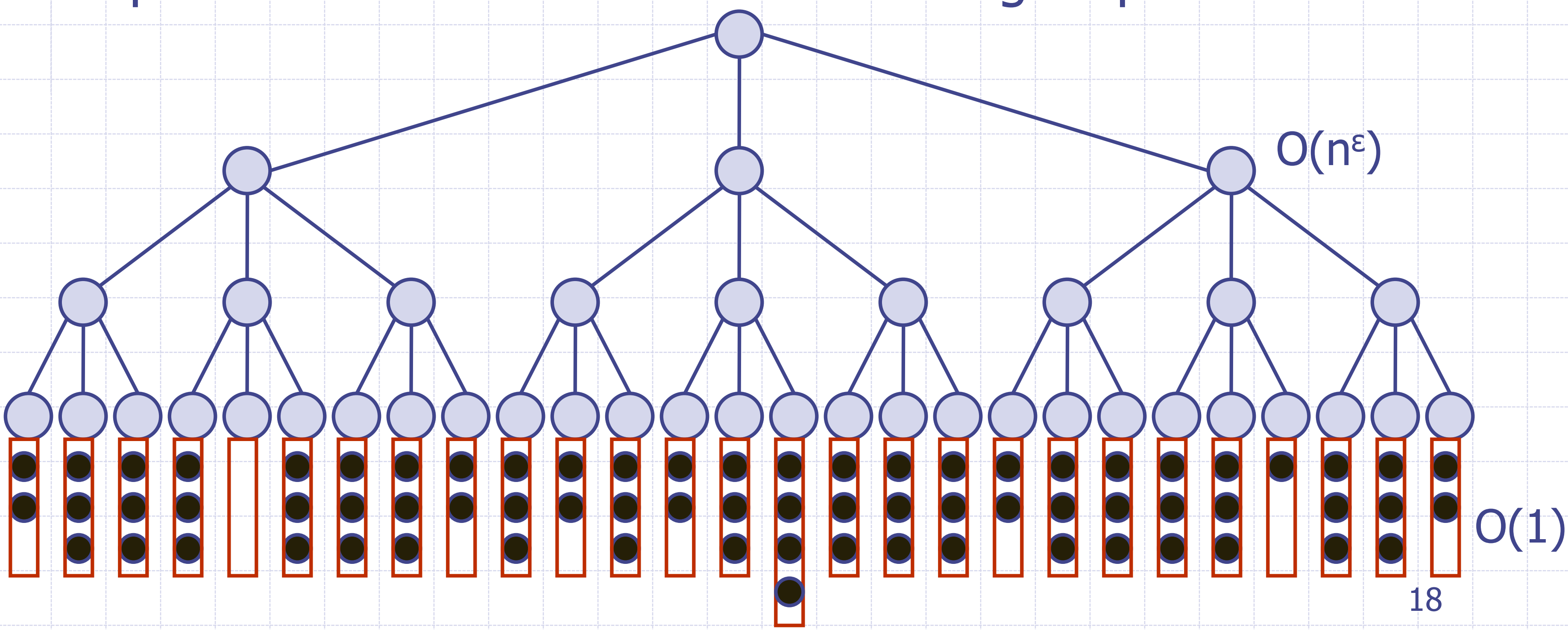
$A = \exp(\beta_3, \alpha_3) \bmod N_3$?

Time: O(1/ε)



Dynamic dictionaries

- Hash function
- O(n) buckets
- Expected O(1) elements per bucket
- Pick $0<\varepsilon<1$
- Tree $T(\varepsilon)$ on top of the buckets
- One digest per bucket

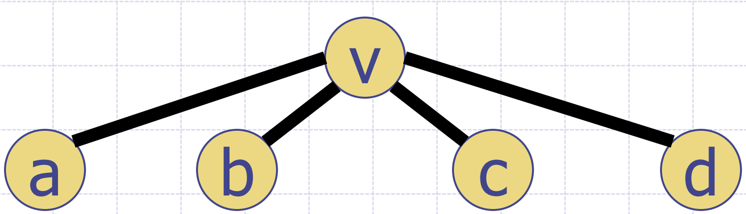


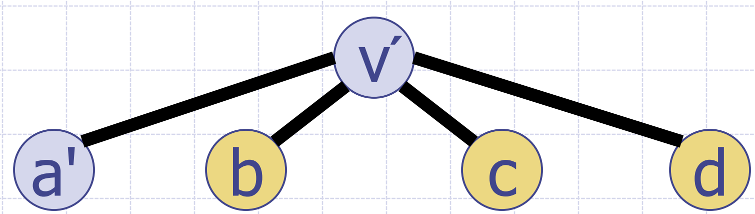
Queries and updates

- With pre-computed witnesses we can support queries in expected O(1) time
- How do you update witnesses?

a	exp(g_i,bcd) mod N_i
b	exp(g_i,acd) mod N_i
c	exp(g_i,abd) mod N_i
d	exp(g_i,abc) mod N_i

a’	exp(g_i,bcd) mod N_i
b	exp(g_i,a’cd) mod N_i
c	exp(g_i,a’bd) mod N_i
d	exp(g_i,a’bc) mod N_i





Updating witnesses

- Suppose a node has m children
- We want to update all m witnesses during an update
- $\phi(N)$ is **not** known to the untrusted server

Algorithm	**Complexity**
brute force	$O(m^2)$
divide and conquer, [Sander+ (00)]	$O(m \log m)$

- Since a node has n^ε children, the update time is $O(n^\varepsilon \log n)$, which can be reduced to $O(n^\varepsilon)$
- Result is expected amortized
  - Bound on size of buckets is expected
  - The hash table has to be rebuilt periodically

Witness on the fly

- With no pre-computed witnesses we can support updates in O(1) time
- Receive the new digests from the source
- Compute the witness explicitly in $O(n^{\varepsilon})$ time

v	exp(g_i,abcd) mod N_i

v′	exp(g_i,a′bcd) mod N_i