Digital Signatures from Strong RSA without Prime Generation

David Cash
Rafael Dowsley
Eike Kiltz

Digital Signatures

Digital signatures are one of mostly deployed cryptographic primitives.

Digital Signatures

Digital signatures are one of mostly deployed cryptographic primitives.

The proved-secure schemes with the best security guarantees are not nearly as efficient as the (unbroken) schemes used in practice.

Digital Signatures

Digital signatures are one of mostly deployed cryptographic primitives.

The proved-secure schemes with the best security guarantees are not nearly as efficient as the (unbroken) schemes used in practice.

The best provable security evidence for the most practical schemes are in the random oracle model (ROM). For instance, the RSA full domain hash [BR96]:

$$Sign(sk=(n,d), m) = H(m)^{d} \bmod n$$

Digital Signatures

Digital signatures are one of mostly deployed cryptographic primitives.

The proved-secure schemes with the best security guarantees are not nearly as efficient as the (unbroken) schemes used in practice.

The best provable security evidence for the most practical schemes are in the random oracle model (ROM). For instance, the RSA full domain hash [BR96]:

$$Sign(sk=(n,d), m) = H(m)^{d} \bmod n$$

Problem: In practice non-random hash functions like SHA-256 are used, which implies some theoretical limitations of these results [CGH98,DOP05].

RSA-based Signatures in the Standard Model

There is a line of work on designing provable secure, efficient signatures based on the (strong) RSA problem in the standard model. The aim is to decrease the efficiency gap between these solutions and the ones in the ROM.

RSA-based Signatures in the Standard Model

There is a line of work on designing provable secure, efficient signatures based on the (strong) RSA problem in the standard model. The aim is to decrease the efficiency gap between these solutions and the ones in the ROM.

In common: the signing algorithm generates primes.

RSA-based Signatures in the Standard Model

There is a line of work on designing provable secure, efficient signatures based on the (strong) RSA problem in the standard model. The aim is to decrease the efficiency gap between these solutions and the ones in the ROM.

In common: the signing algorithm generates primes.

Generating primes is expensive and it is not an intrinsic step for the signing algorithm, thus it is desirable to avoid it.

Our Work

Strong RSA-based signature without prime generation for signing.

Our Work

Strong RSA-based signature without prime generation for signing.

Public key and signature sizes competitive with prior schemes, the verification is much slower.

Our Work

Strong RSA-based signature without prime generation for signing.

Public key and signature sizes competitive with prior schemes, the verification is much slower.

Not competitive with ROM solutions.

Our Work

Strong RSA-based signature without prime generation for signing.

Public key and signature sizes competitive with prior schemes, the verification is much slower.

Not competitive with ROM solutions.

Conceptual contribution towards the goal of practical schemes from conservative hardness assumptions without random oracles.

Strong RSA Problem and Signatures

Challenger

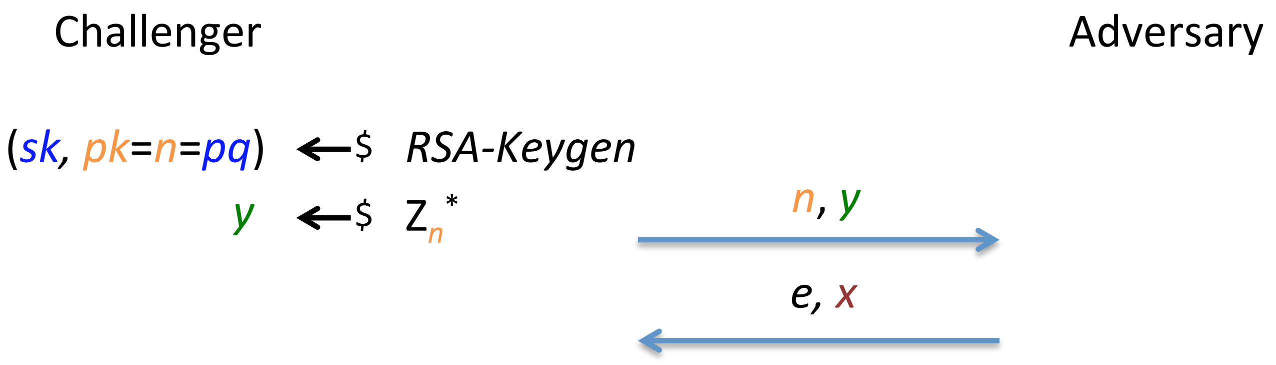
Adversary

$(sk, pk = n = pq) \leftarrow\$ \ \mathit{RSA\text{-}Keygen}$

$y \leftarrow\$ \ Z_n^*$

n, y

Strong RSA Problem and Signatures



Challenger

Adversary

$(sk, pk=n=pq) \leftarrow\$ \ \mathit{RSA\text{-}Keygen}$

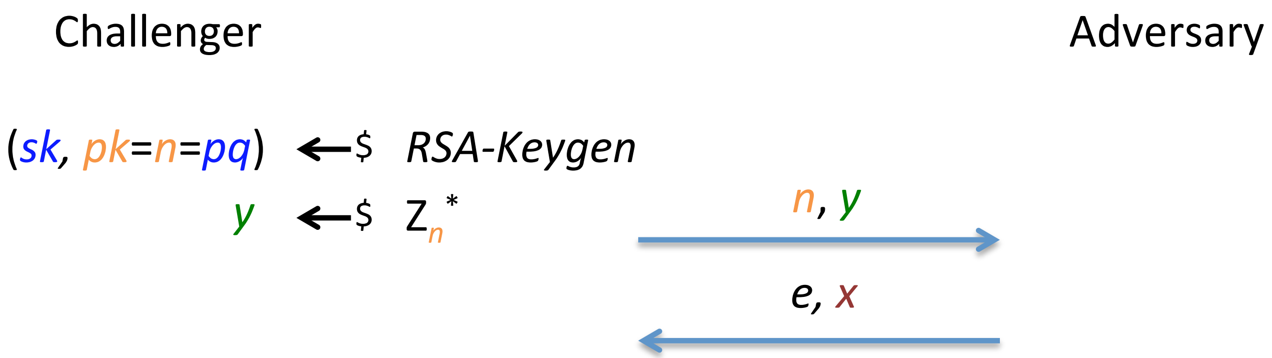
$y \leftarrow\$ \ Z_n^*$

n, y

e, x

Wins if $x^e=y \bmod n$ and $e>1$.

Strong RSA Problem and Signatures



Challenger

Adversary

$(sk, pk=n=pq) \leftarrow\$ \; \mathit{RSA\text{-}Keygen}$

$y \leftarrow\$ \; Z_n^*$

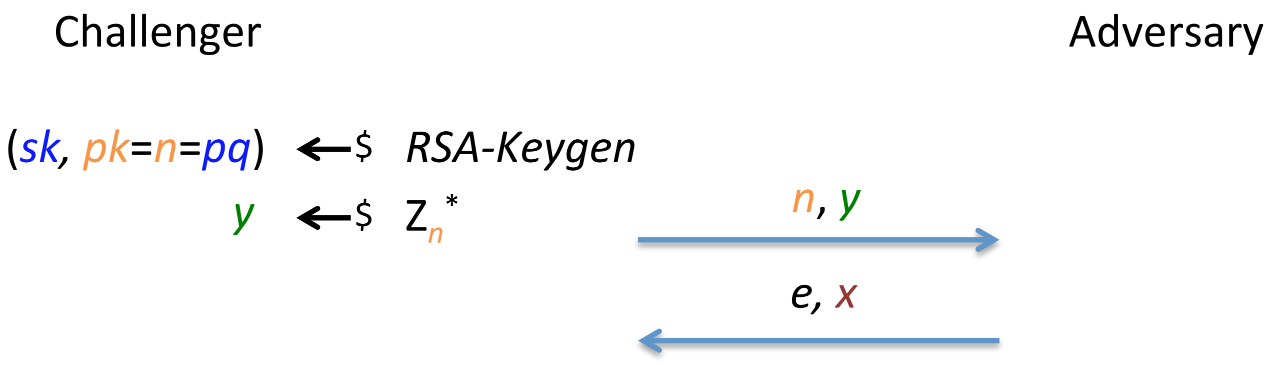
n, y

e, x

Wins if $x^e=y \bmod n$ and $e>1$.

Natural approach for embedding it into digital signatures: the signature is (e, x) where x is the e-th root of a value y that depends on the message.

Strong RSA Problem and Signatures



Challenger

Adversary

$(sk, pk=n=pq) \leftarrow\$ \ \textit{RSA-Keygen}$

$y \leftarrow\$ \ Z_n^*$

n, y

e, x

Wins if $x^e=y \bmod n$ and $e>1$.

Natural approach for embedding it into digital signatures: the signature is (e, x) where x is the e-th root of a value y that depends on the message.

In order to apply known techniques that prevent an adversary from assembling several signatures into a new signature, e is typically required to be prime or a product of primes.

Current Schemes

Many existing schemes [CS99,F03,HK08,Z01,Z03,CL04] use

$$Sign(sk, m) = (H(m)^{1/e} \text{ mod } n, e)$$

where e is a random prime and is H some (algebraic) hash function.

Current Schemes

Many existing schemes [CS99,F03,HK08,Z01,Z03,CL04] use

$$Sign(sk, m) = (H(m)^{1/e} \bmod n, e)$$

where e is a random prime and is H some (algebraic) hash function.

Other schemes [GHR99,HW09,HJK11] based on the strong/standard RSA problem use

$$Sign(sk, m) = g^{1/\prod h_i(m)} \bmod n$$

where h_i are independent hash functions that hash into primes.

Our Scheme

Uses the prefix signing technique of Hohenberger-Waters [HW09].

Our Scheme

Uses the prefix signing technique of Hohenberger-Waters [HW09].

Let p and q be large safe primes and F a be pseudorandom function that outputs random odds numbers.

Our Scheme

Uses the prefix signing technique of Hohenberger-Waters [HW09].

Let p and q be large safe primes and F a be pseudorandom function that outputs random odds numbers.

pk = (n,g,k) where n=pq, g is a random element of Z_n^* and k is a key for F

Our Scheme

Uses the prefix signing technique of Hohenberger-Waters [HW09].

Let p and q be large safe primes and F a be pseudorandom function that outputs random odds numbers.

$pk = (n,g,k)$ where $n=pq$, g is a random element of Z_n^* and k is a key for F

$sk = (p,q,pk)$

Our Scheme

Uses the prefix signing technique of Hohenberger-Waters [HW09].

Let p and q be large safe primes and F a be pseudorandom function that outputs random odds numbers.

$pk = (n,g,k)$ where $n=pq$, g is a random element of Z_n^* and k is a key for F

$sk = (p,q,pk)$

The signature on a message m of L-bits is

$$Sign(sk, m) = g^{1/e} \bmod n$$

$$e = \prod_{i=1}^{L} F(k,m[1 \ldots i]) \prod_{i=1}^{u} F(k,m||i)$$

Our Scheme

To verify a signature σ on a message m first compute

$$e = \prod_{i=1}^{L} F(k, m[1 \ldots i]) \prod_{i=1}^{u} F(k, m || i)$$

Accept if

$$\sigma^{e} = g \bmod n$$

Our Scheme

To verify a signature σ on a message m first compute

$$e = \prod_{i=1}^{L} F(k, m[1 \ldots i]) \prod_{i=1}^{u} F(k, m || i)$$

Accept if

$$\sigma^e = g \bmod n$$

We show that this scheme is secure against weak chosen message attacks.

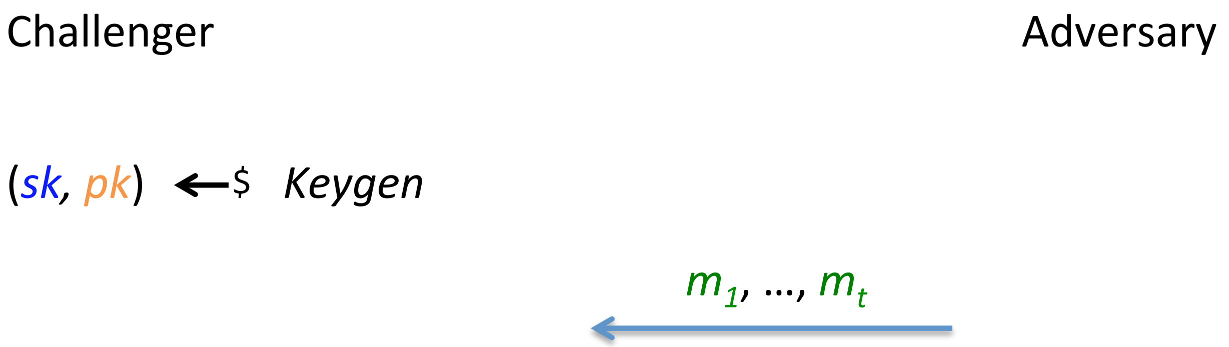
Weak CMA Security

Challenger

Adversary

$(sk, pk) \xleftarrow{\$} \mathit{Keygen}$

Weak CMA Security



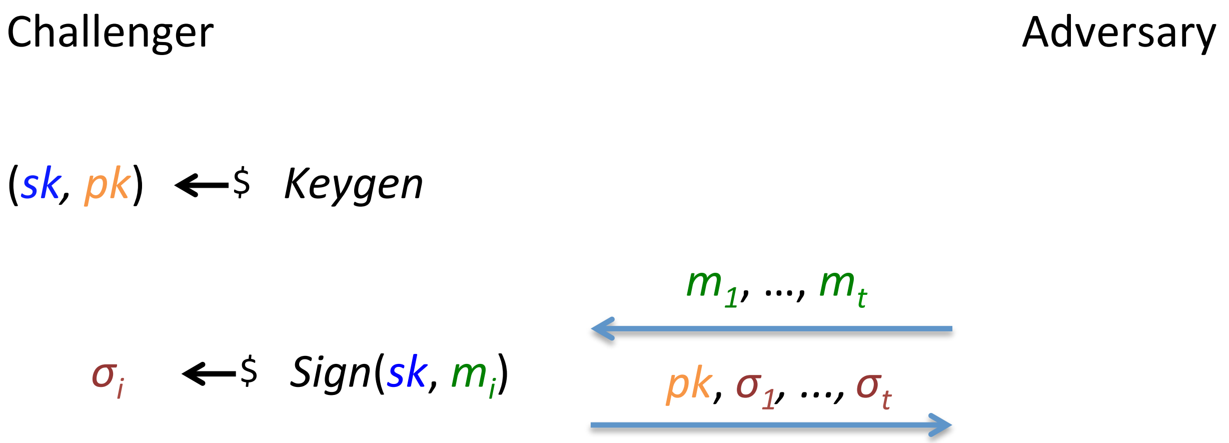
Challenger

Adversary

$(sk, pk) \leftarrow\$ \; Keygen$

$m_1, \ldots, m_t$

Weak CMA Security



Challenger

Adversary

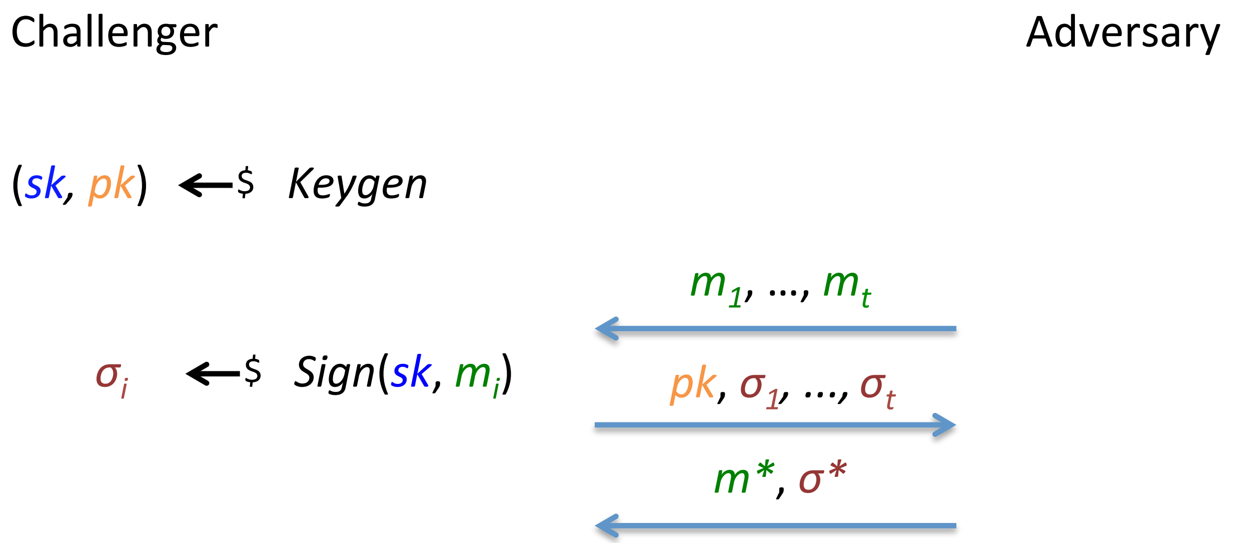
$(sk, pk) \leftarrow\$ \ \mathit{Keygen}$

$m_1, \ldots, m_t$

$\sigma_i \leftarrow\$ \ \mathit{Sign}(sk, m_i)$

$pk, \sigma_1, \ldots, \sigma_t$

Weak CMA Security



Challenger

Adversary

$(sk, pk) \leftarrow\$ \ \mathit{Keygen}$

$m_1, \ldots, m_t$

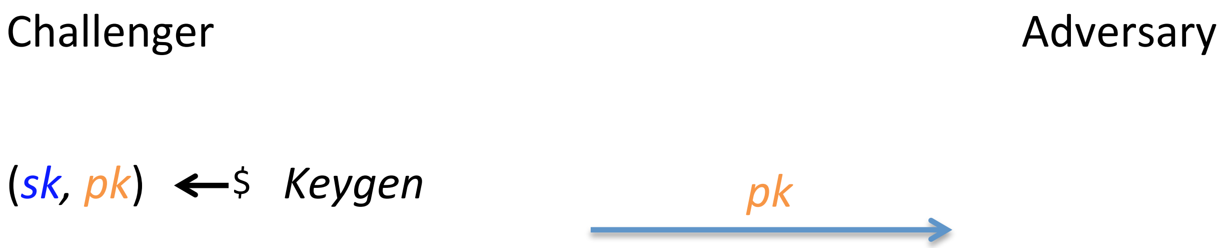
$\sigma_i \leftarrow\$ \ \mathit{Sign}(sk, m_i)$

$pk, \sigma_1, \ldots, \sigma_t$

m^*, σ^*

Wins if $\mathit{Verify}(pk, m^*, \sigma^*)$=1 and m^* was not queried.

CMA Security



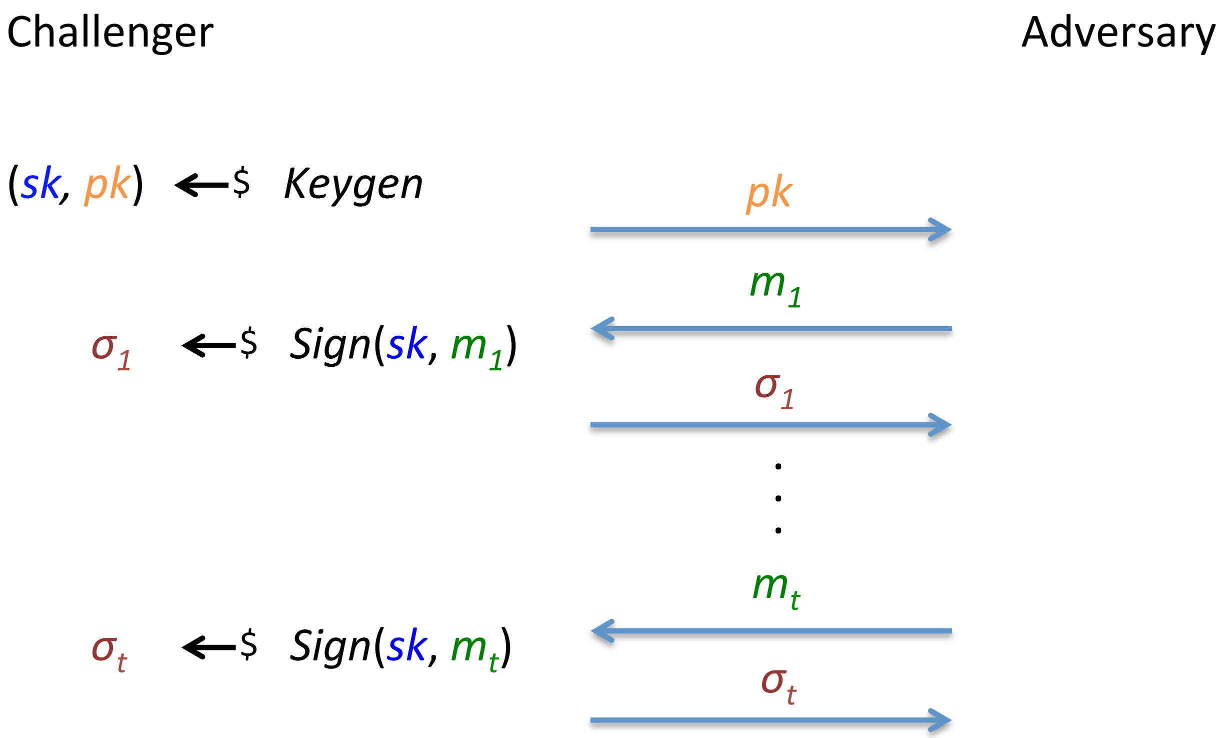
Challenger

Adversary

$(sk, pk) \leftarrow \$ \ \mathit{Keygen}$

pk

CMA Security



Challenger

Adversary

$(sk, pk) \xleftarrow{\$} Keygen$

pk

m_1

$\sigma_1 \xleftarrow{\$} Sign(sk, m_1)$

σ_1

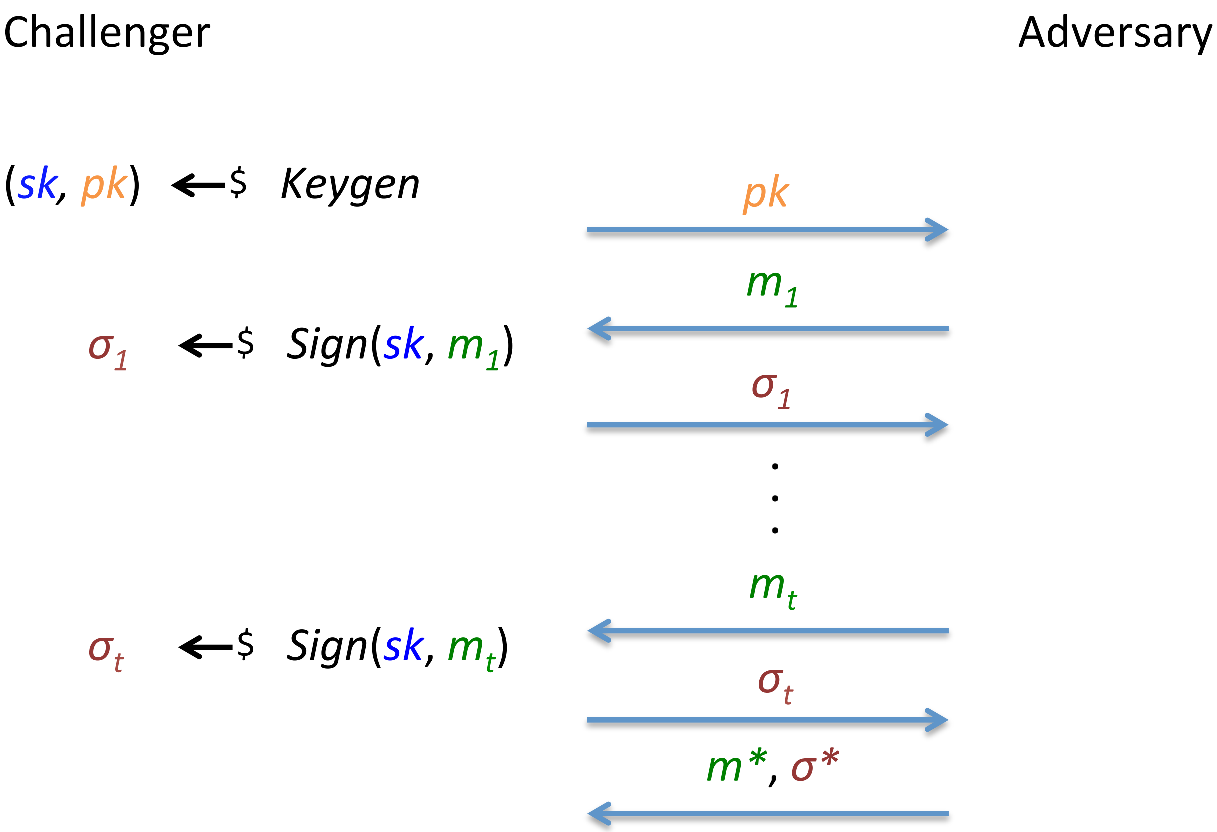
$\vdots$

m_t

$\sigma_t \xleftarrow{\$} Sign(sk, m_t)$

σ_t

CMA Security



Challenger

Adversary

$(sk, pk) \leftarrow\$ \ \mathit{Keygen}$

pk →

m_1 ←

$\sigma_1 \leftarrow\$ \ \mathit{Sign}(sk, m_1)$

σ_1 →

⋮

m_t ←

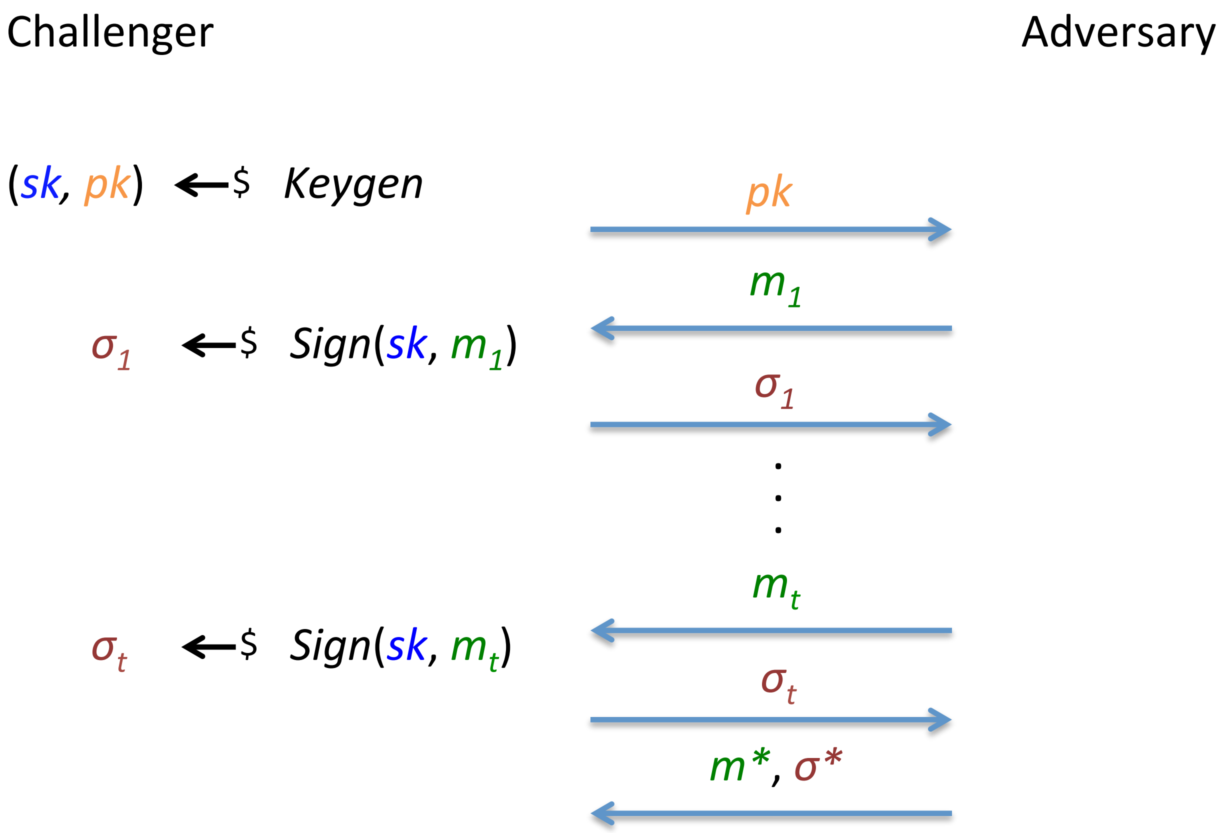
$\sigma_t \leftarrow\$ \ \mathit{Sign}(sk, m_t)$

σ_t →

m^*, σ^* ←

Wins if *Verify*(pk, m^*, σ^*)=1 and m^* was not queried.

CMA Security



Challenger

Adversary

$(sk, pk) \xleftarrow{\$} \mathit{Keygen}$

pk

m_1

$\sigma_1 \xleftarrow{\$} \mathit{Sign}(sk, m_1)$

σ_1

$\vdots$

m_t

$\sigma_t \xleftarrow{\$} \mathit{Sign}(sk, m_t)$

σ_t

m^*, σ^*

Wins if $\mathit{Verify}(pk, m^*, \sigma^*)=1$ and m^* was not queried.

A chameleon hash function can be used to get from weak CMA to CMA [KR00,HW09].

Prefix Signing Technique [HW09]

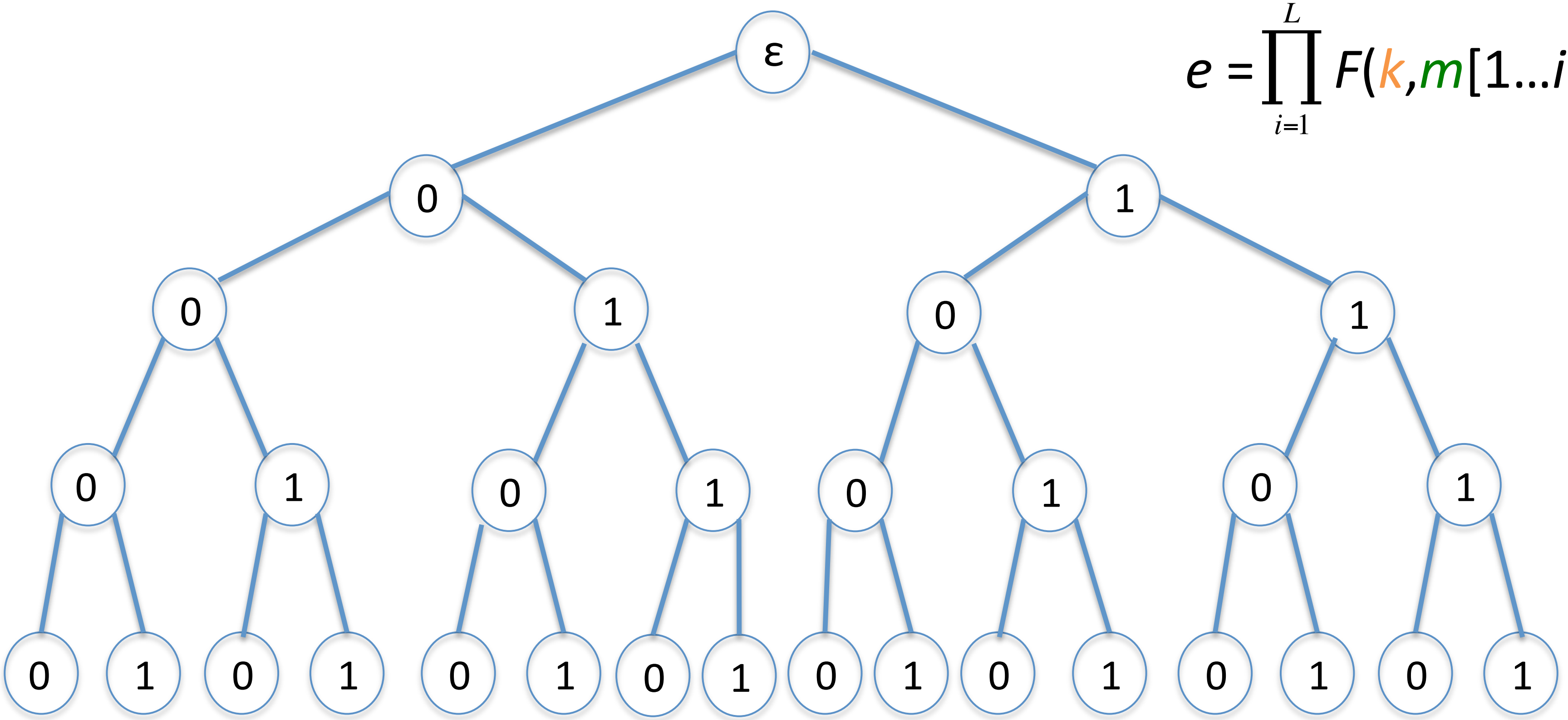
$$Sign(sk, m) = g^{1/e} \bmod n$$

$$e = \prod_{i=1}^{L} F(k, m[1 \ldots i])$$

Prefix Signing Technique [HW09]

$$Sign(sk, m) = g^{1/e} \bmod n$$

$$e = \prod_{i=1}^{L} F(k, m[1\ldots i])$$

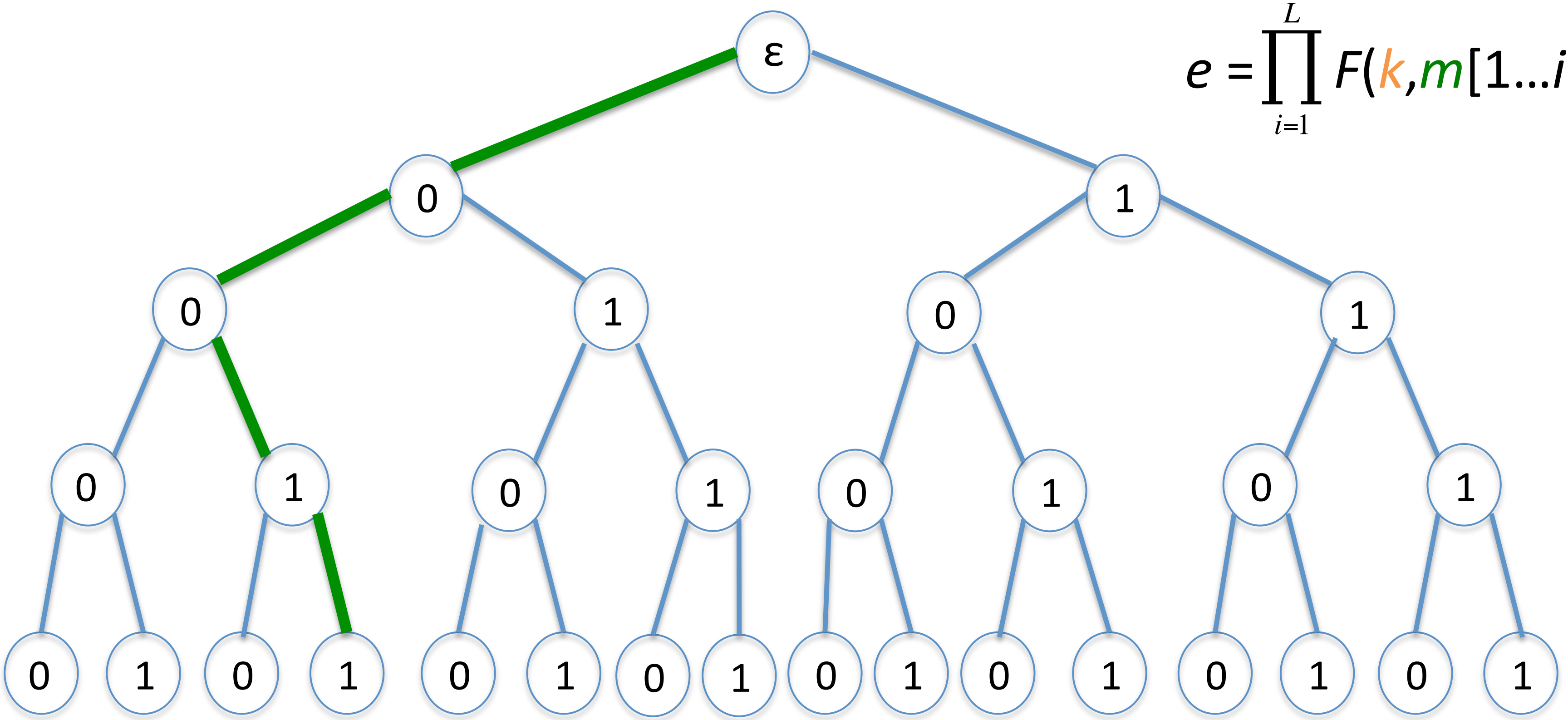


Each branch has an associated number that is determined by *F*.

Prefix Signing Technique [HW09]

$$Sign(sk, m) = g^{1/e} \bmod n$$

$$e = \prod_{i=1}^{L} F(k, m[1\ldots i])$$



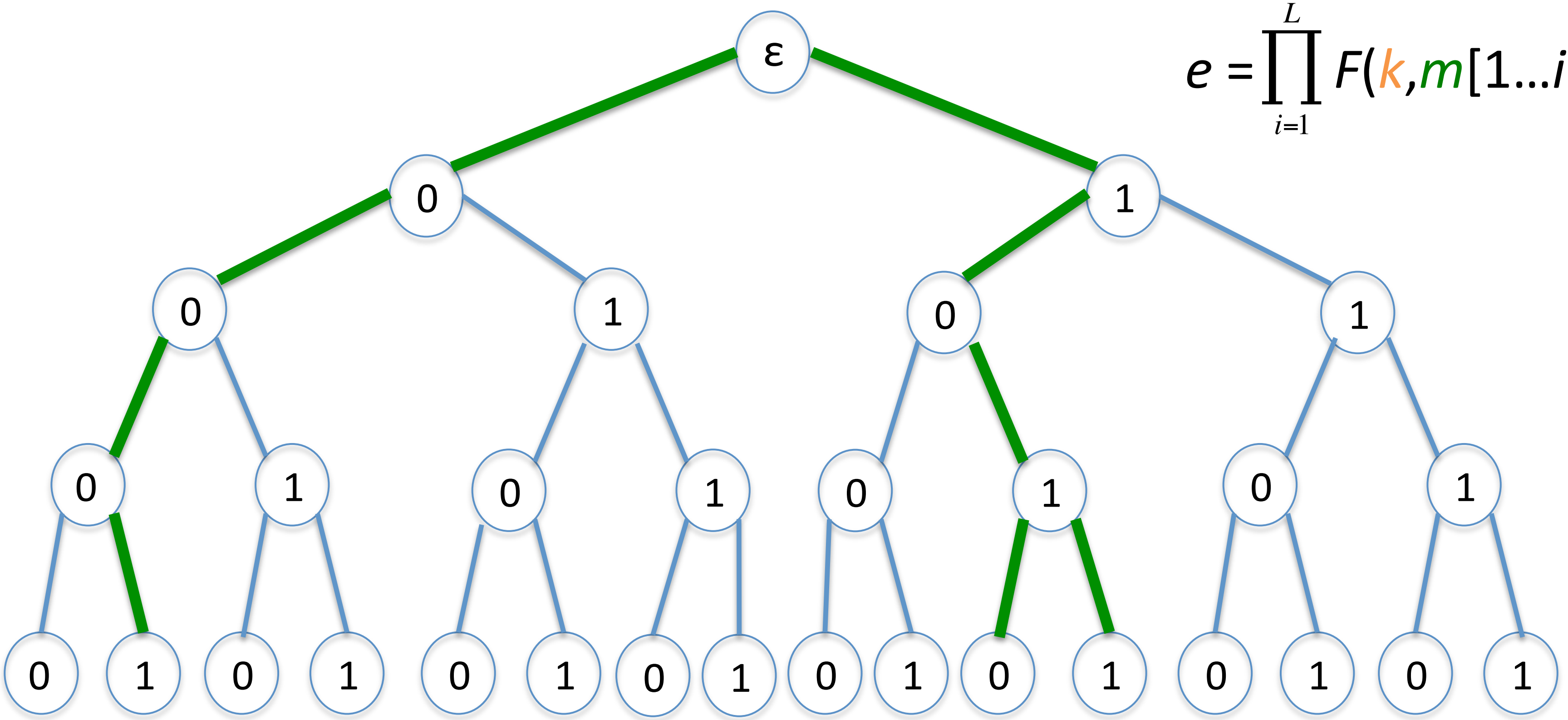
Each branch has an associated number that is determined by *F*.

To sign a message *m*=0011, for instance, compute *e* as the product of the numbers associated with the branches in green.

Prefix Signing Technique [HW09]

$$Sign(sk, m) = g^{1/e} \bmod n$$

$$e = \prod_{i=1}^{L} F(k, m[1 \ldots i])$$

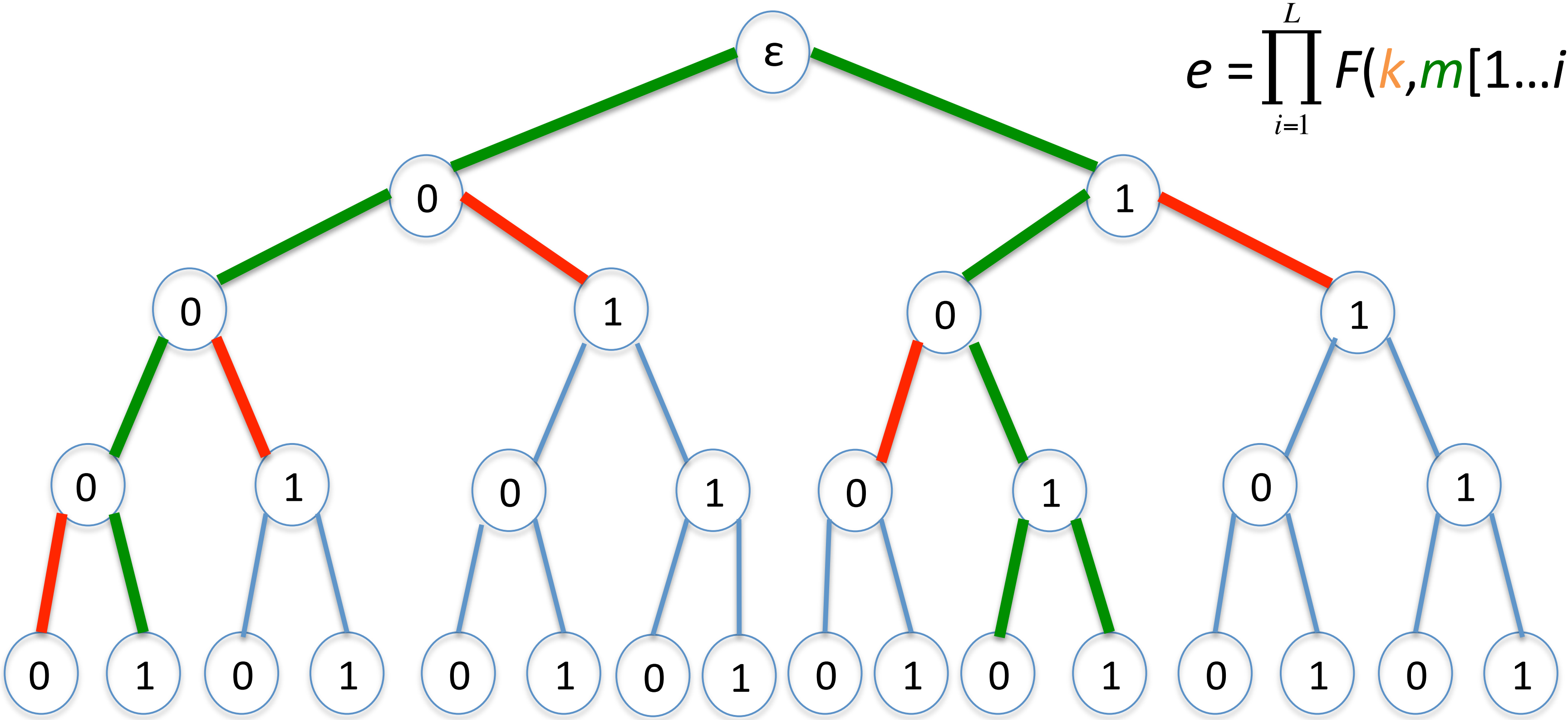


Let the sub-tree in green be the branches associated with the parallel signing query of the adversary.

Prefix Signing Technique [HW09]

$$Sign(sk, m) = g^{1/e} \bmod n$$

$$e = \prod_{i=1}^{L} F(k, m[1 \ldots i])$$



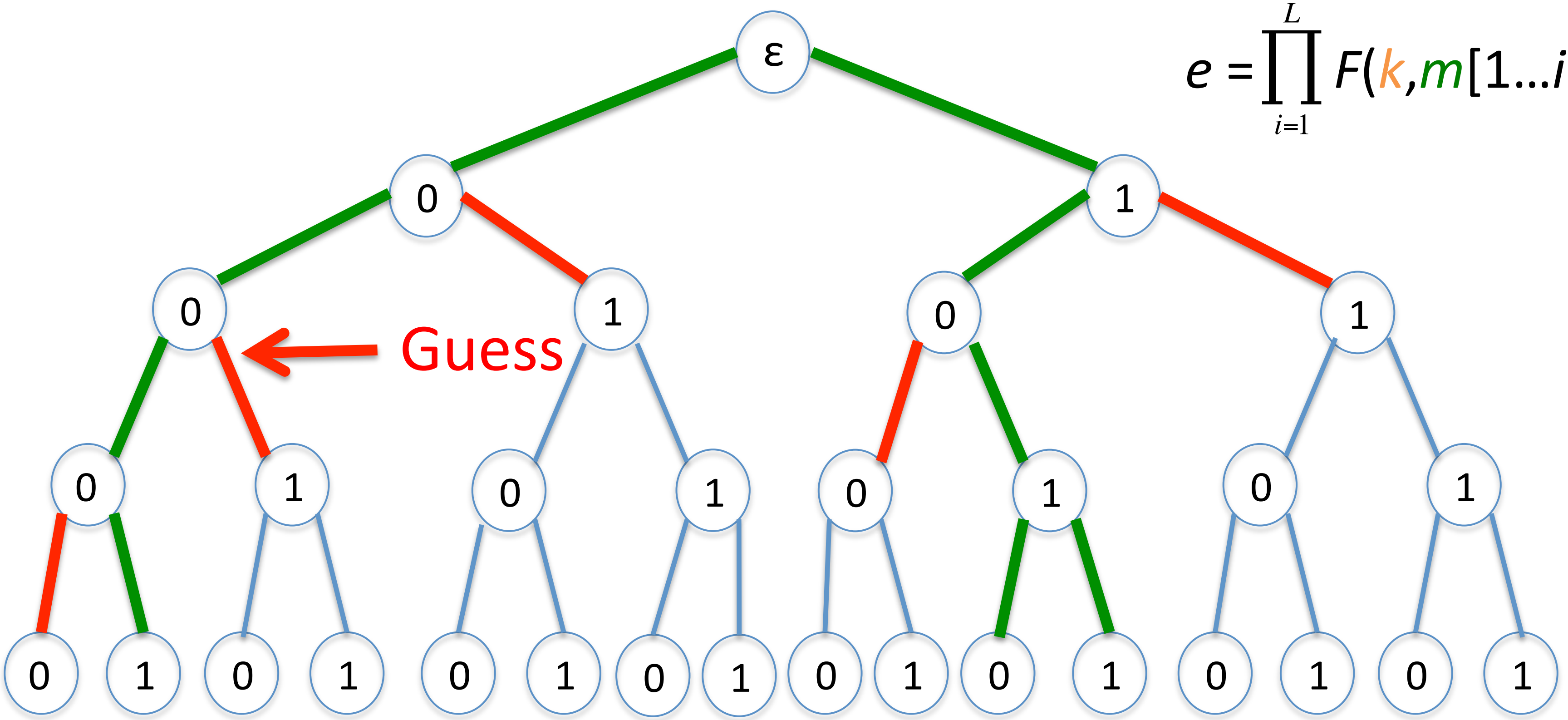
Let the sub-tree in green be the branches associated with the parallel signing query of the adversary.

Exit branch: non-green, but only has green branches on path from root until it.

Prefix Signing Technique [HW09]

$$Sign(sk, m) = g^{1/e} \bmod n$$

$$e = \prod_{i=1}^{L} F(k, m[1 \ldots i])$$



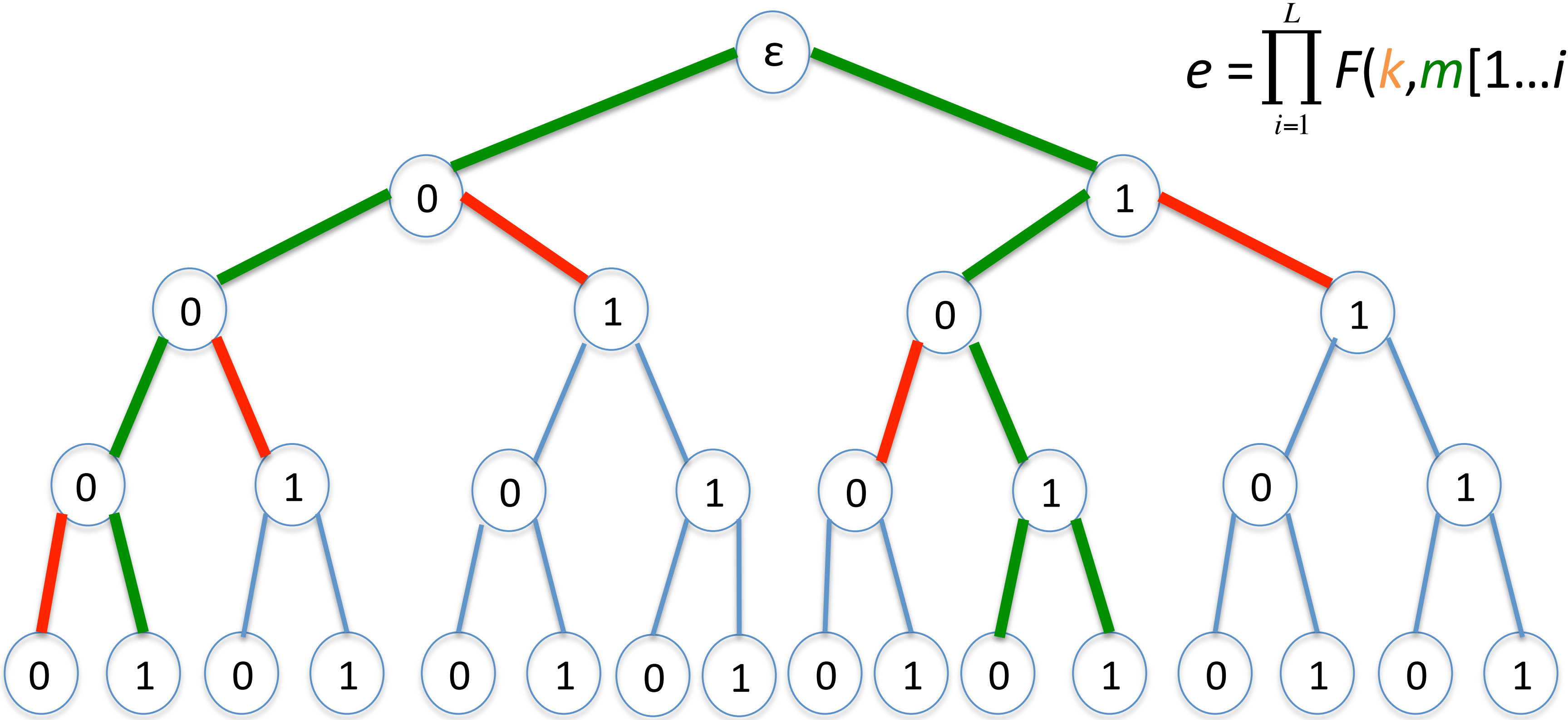
Let the sub-tree in green be the branches associated with the parallel signing query of the adversary.

Exit branch: non-green, but only has green branches on path from root until it.

Prefix Signing Technique [HW09]

$$Sign(sk, m) = g^{1/e} \bmod n$$

$$e = \prod_{i=1}^{L} F(k, m[1...i])$$

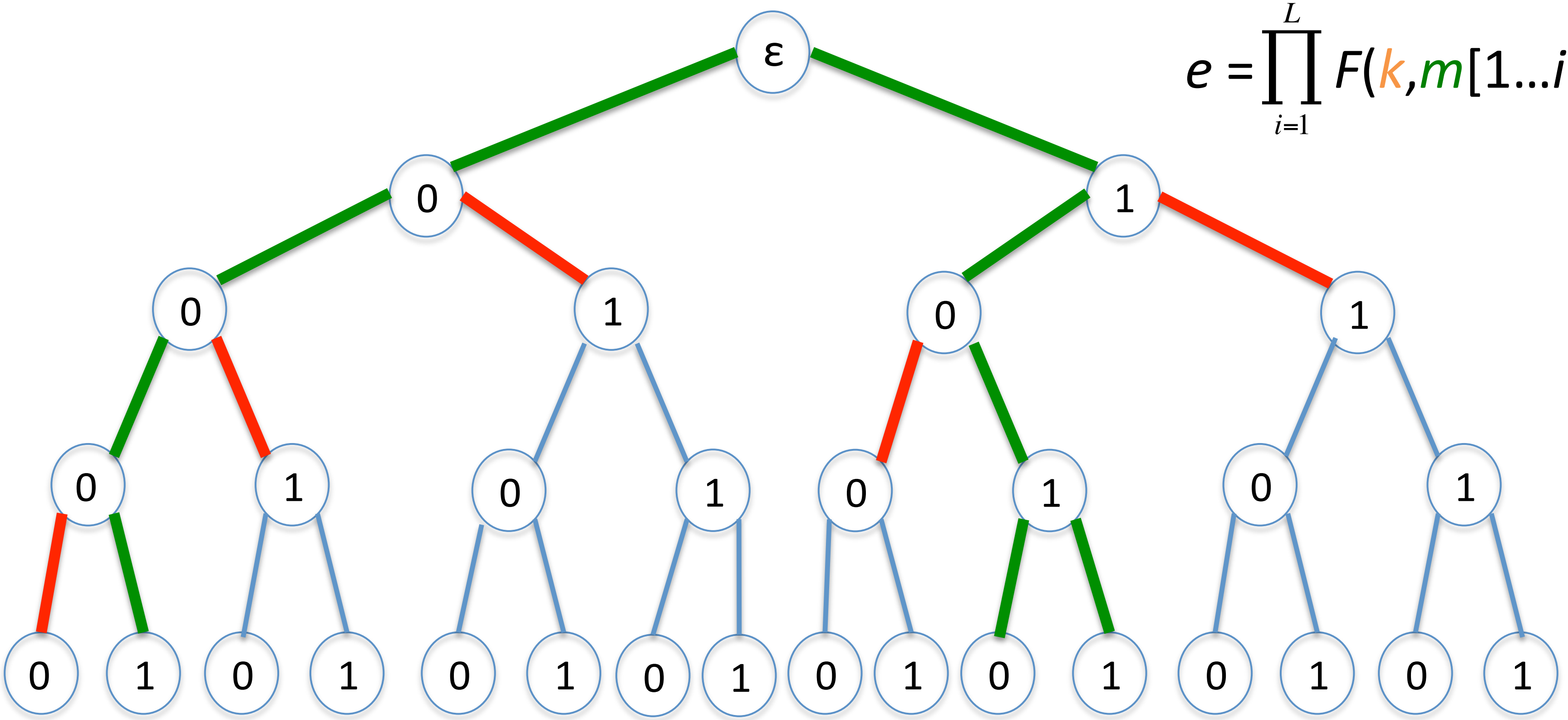


We also use this structure, but with *F* that outputs random odd numbers and based on the strong RSA problem.

Prefix Signing Technique [HW09]

$$Sign(sk, m) = g^{1/e} \bmod n$$

$$e = \prod_{i=1}^{L} F(k, m[1 \ldots i])$$



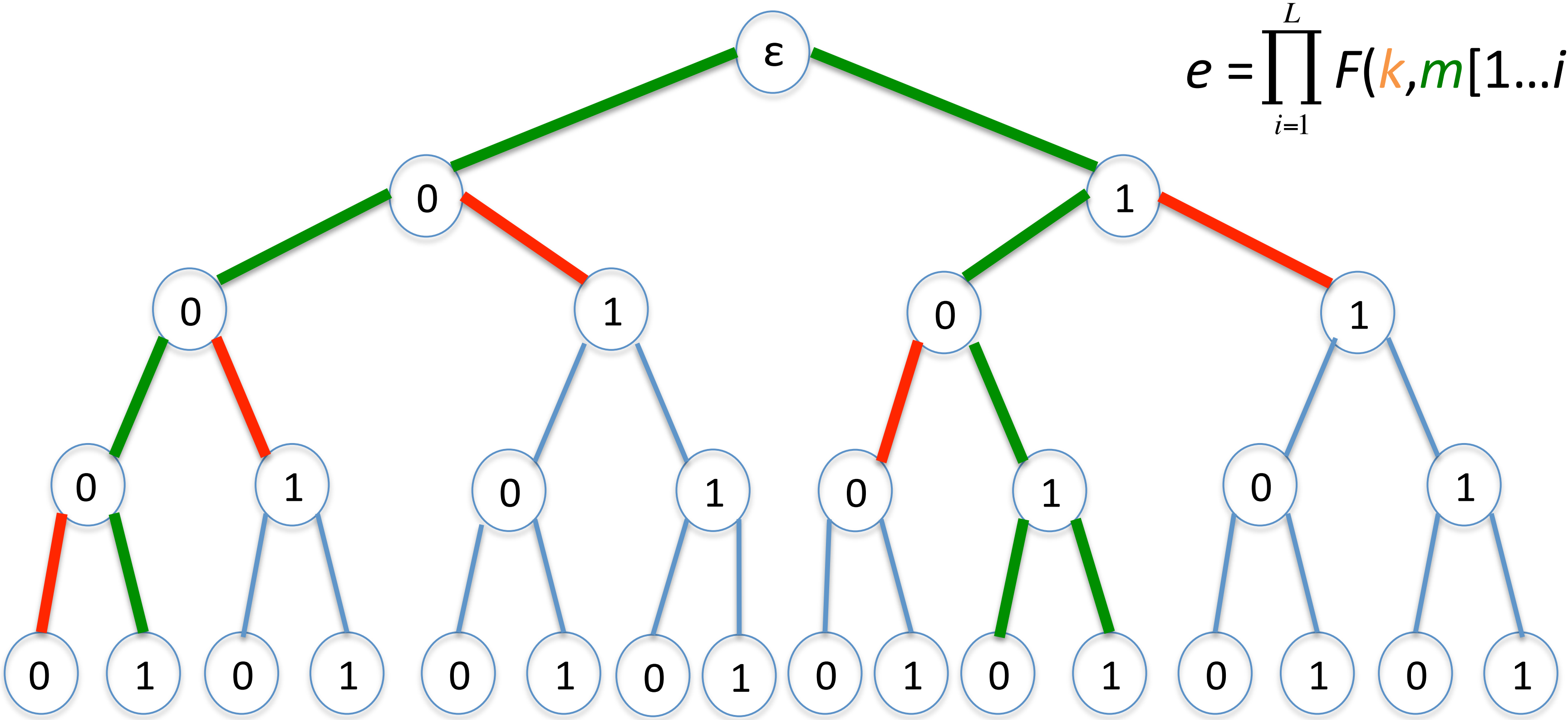
We also use this structure, but with *F* that outputs random odd numbers and based on the strong RSA problem.

We do not try to guess the exit branch.

Naïve Approach

$$Sign(sk, m) = g^{1/e} \bmod n$$

$$e = \prod_{i=1}^{L} F(k, m[1 \ldots i])$$

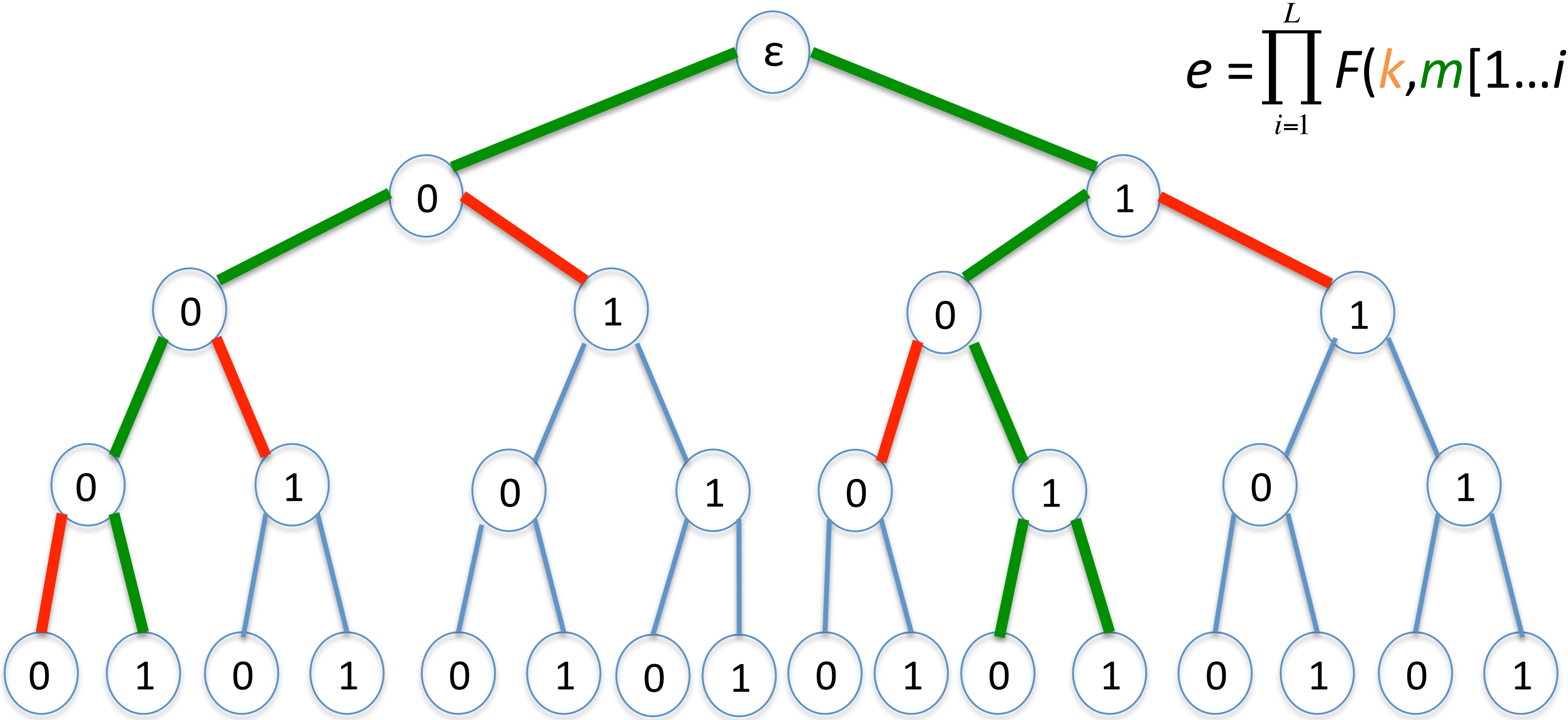


Let α be a sufficiently large integer and set the outputs of F to be large enough so that they are α-smooth with negligible probability only.

Naïve Approach

$$Sign(sk, m) = g^{1/e} \bmod n$$

$$e = \prod_{i=1}^{L} F(k, m[1 \ldots i])$$



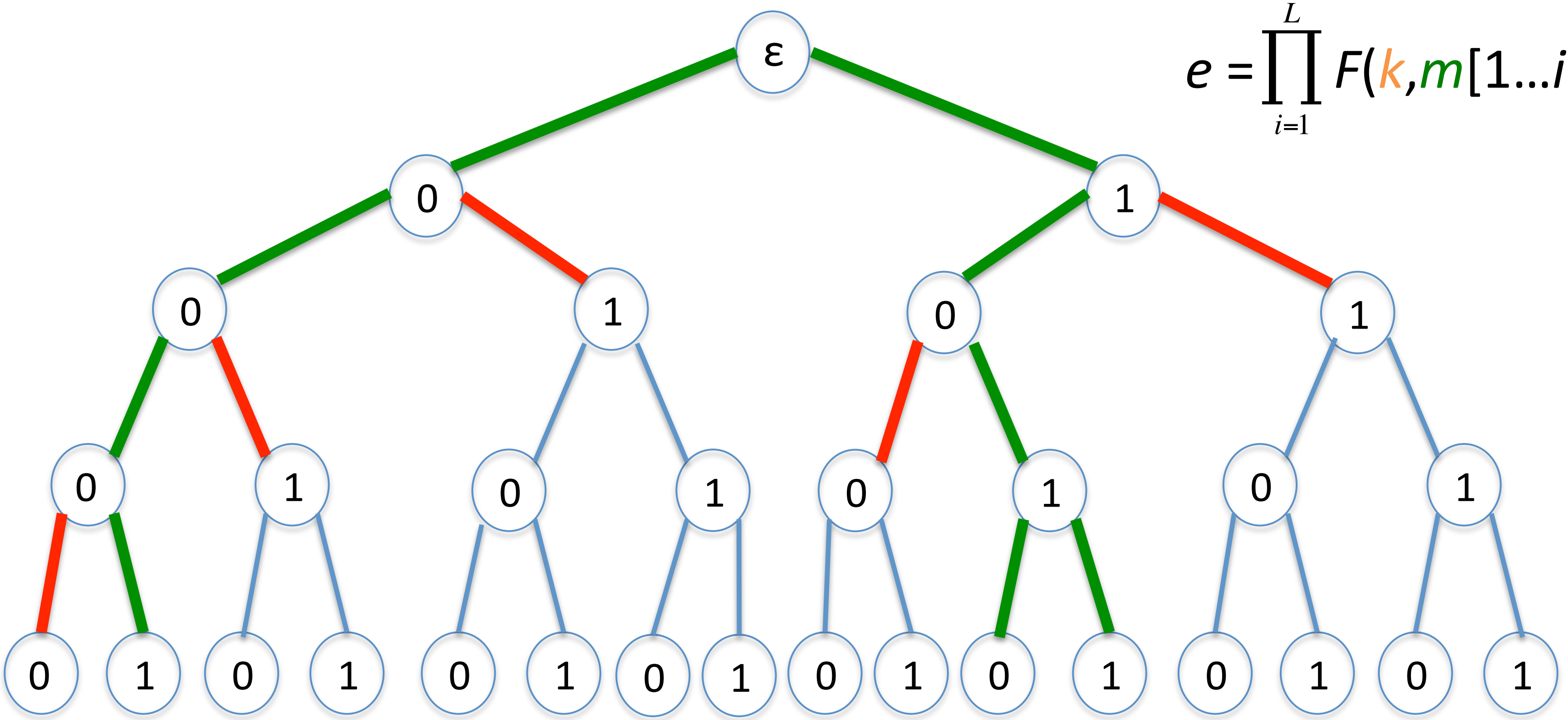
Let α be a sufficiently large integer and set the outputs of F to be large enough so that they are α-smooth with negligible probability only.

With overwhelming probability all exit branches are good. Very bad parameters.

Our Scheme

$$Sign(sk, m) = g^{1/e} \bmod n$$

$$e = \prod_{i=1}^{L} F(k, m[1\ldots i])$$

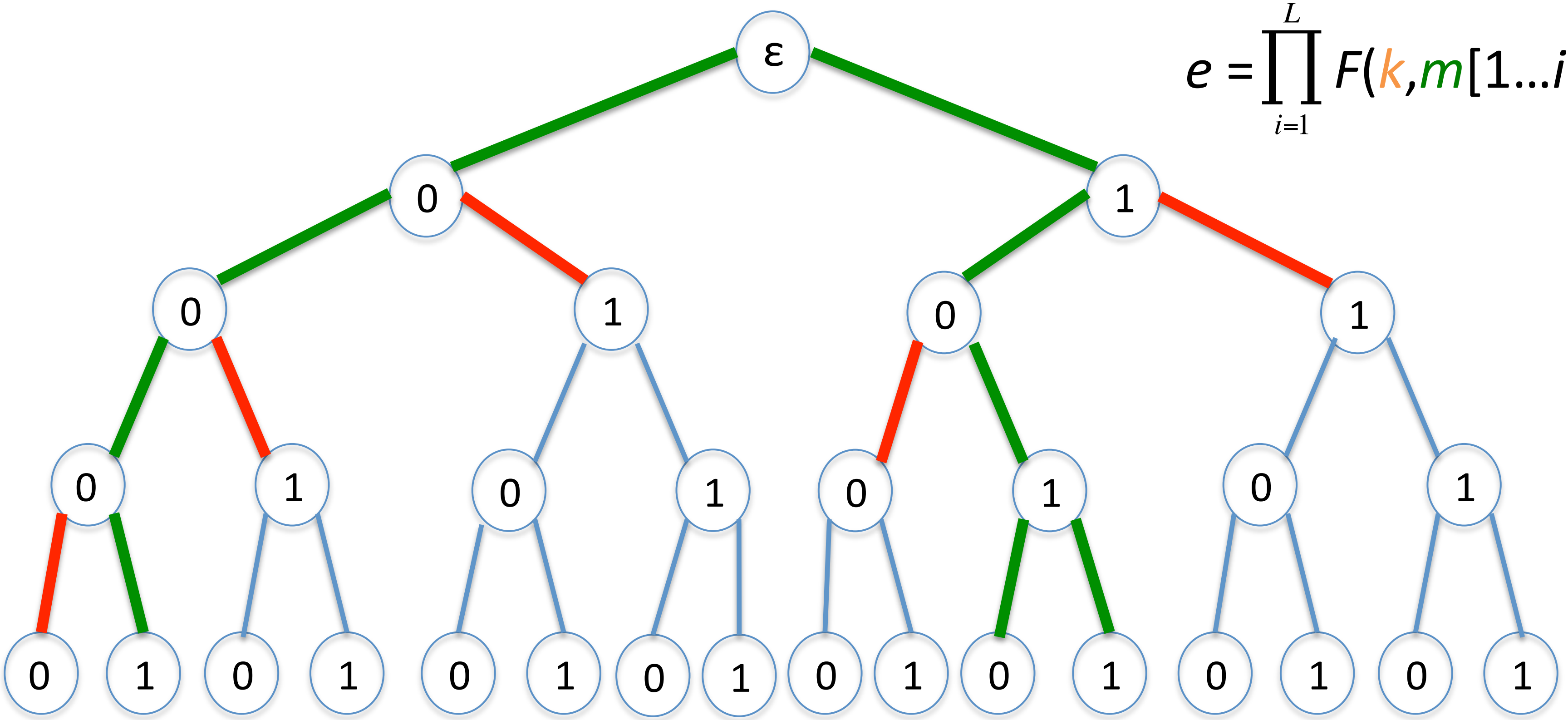


Only require the numbers to be non-α-smooth with constant probability.

Proof Idea

$$Sign(sk, m) = g^{1/e} \bmod n$$

$$e = \prod_{i=1}^{L} F(k, m[1 \ldots i])$$

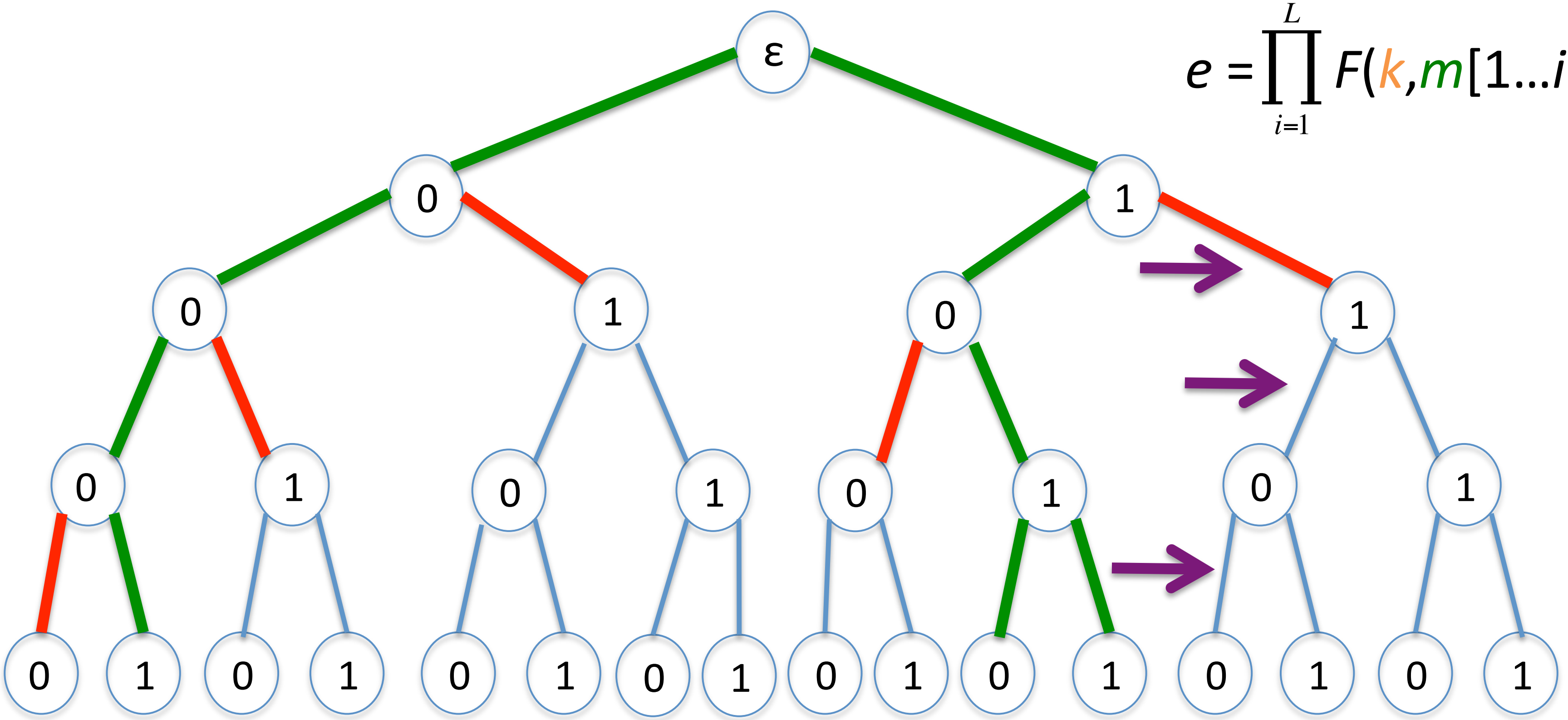


For every possible valid forgery, there should be a non-α-smooth number in its root to message path that is not in the sub-tree of queried messages.

Proof Idea

$$Sign(sk, m) = g^{1/e} \bmod n$$

$$e = \prod_{i=1}^{L} F(k, m[1 \ldots i])$$

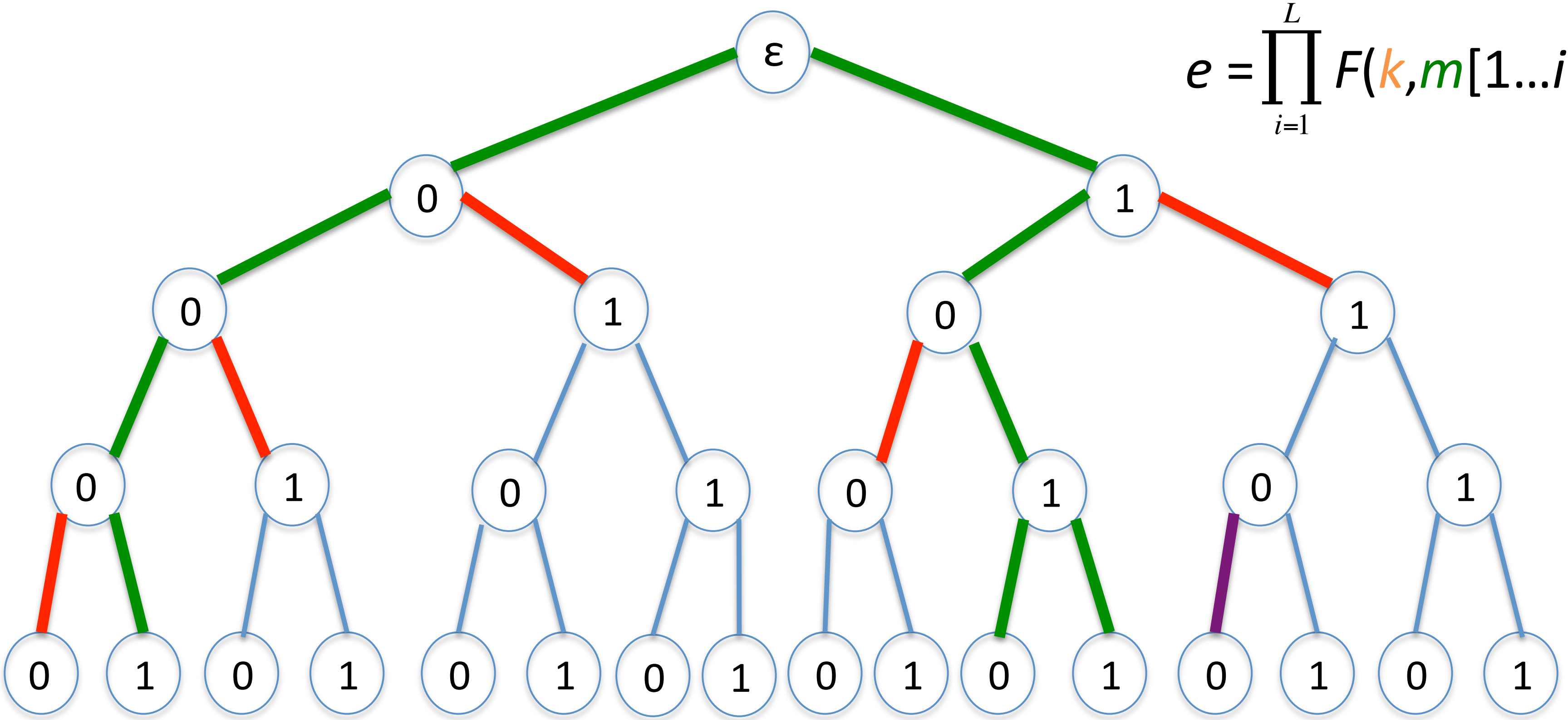


For every possible valid forgery, there should be a non-α-smooth number in its root to message path that is not in the sub-tree of queried messages.

Proof Idea

$$Sign(sk, m) = g^{1/e} \bmod n$$

$$e = \prod_{i=1}^{L} F(k, m[1 \ldots i])$$

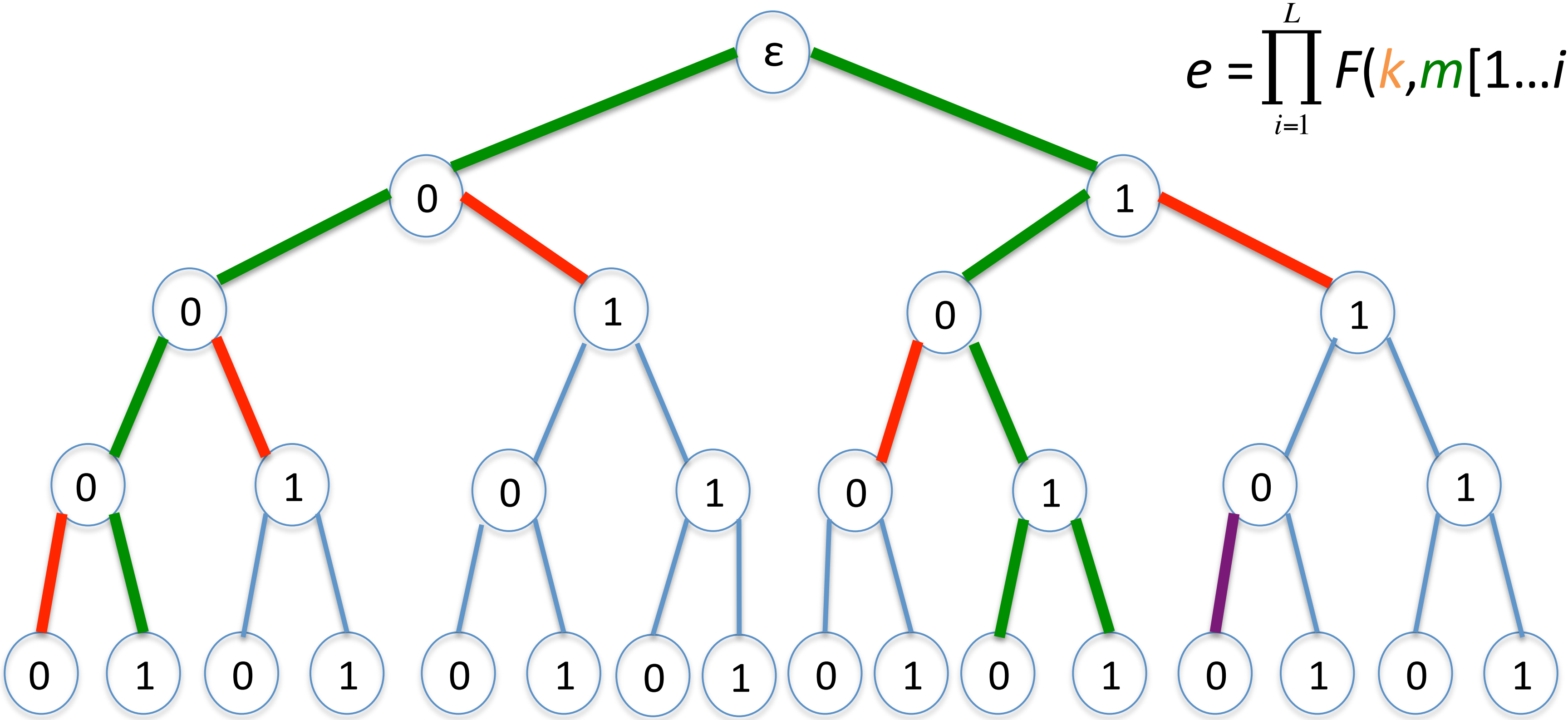


For every possible valid forgery, there should be a non-α-smooth number in its root to message path that is not in the sub-tree of queried messages.

Proof Idea

$$Sign(sk, m) = g^{1/e} \bmod n$$

$$e = \prod_{i=1}^{L} F(k, m[1 \ldots i])$$

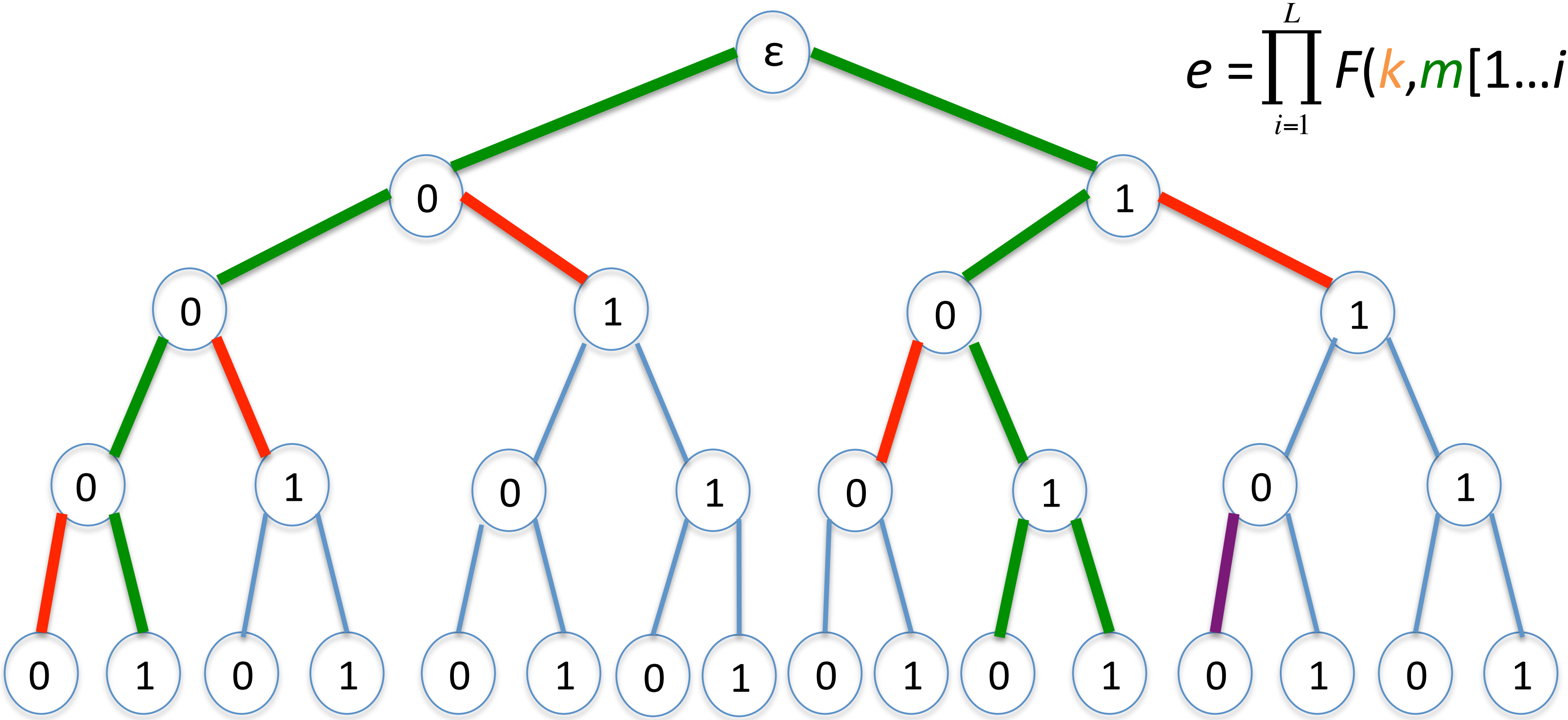


Adapt analysis technique from Gennaro et al. [GHR99] to analyze the probability that a non-smooth number divides the product of some random numbers.

Proof Idea

$$Sign(sk, m) = g^{1/e} \bmod n$$

$$e = \prod_{i=1}^{L} F(k, m[1 \ldots i])$$



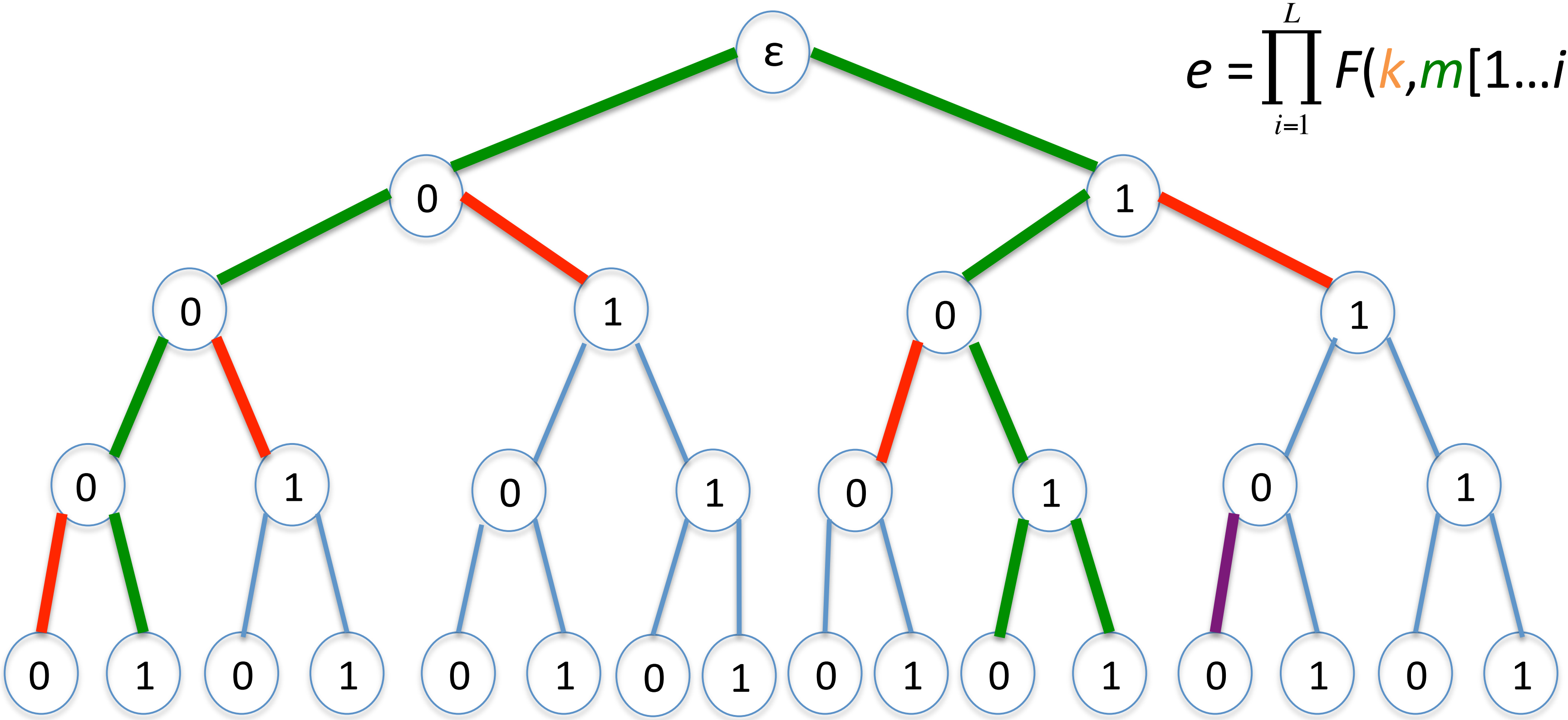
Adapt analysis technique from Gennaro et al. [GHR99] to analyze the probability that a non-smooth number divides the product of some random numbers.

If the number associated with the branch in **purple** does not divide the product of the ones in **green**, then it is possible to solve the strong RSA problem.

Proof Idea

$$Sign(sk, m) = g^{1/e} \bmod n$$

$$e = \prod_{i=1}^{L} F(k, m[1...i])$$



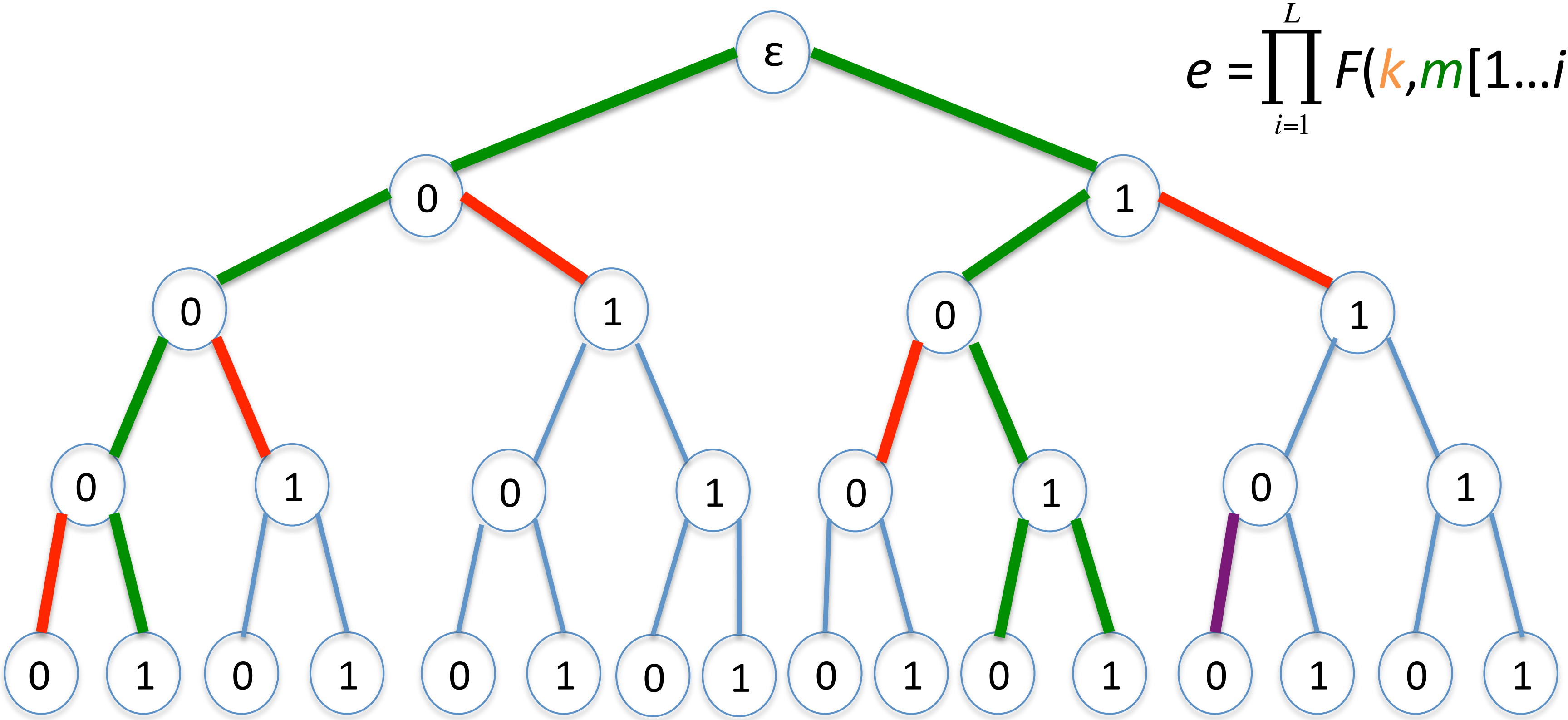
Given an strong RSA instance (n, y) and the signature query, compute g backwards:

$$g = y^{\prod \text{green numbers}} \bmod n$$

Proof Idea

$$Sign(sk, m) = g^{1/e} \bmod n$$

$$e = \prod_{i=1}^{L} F(k, m[1 \ldots i])$$



Given a forged signature (m^*, σ^*), it is possible to solve the strong RSA problem if

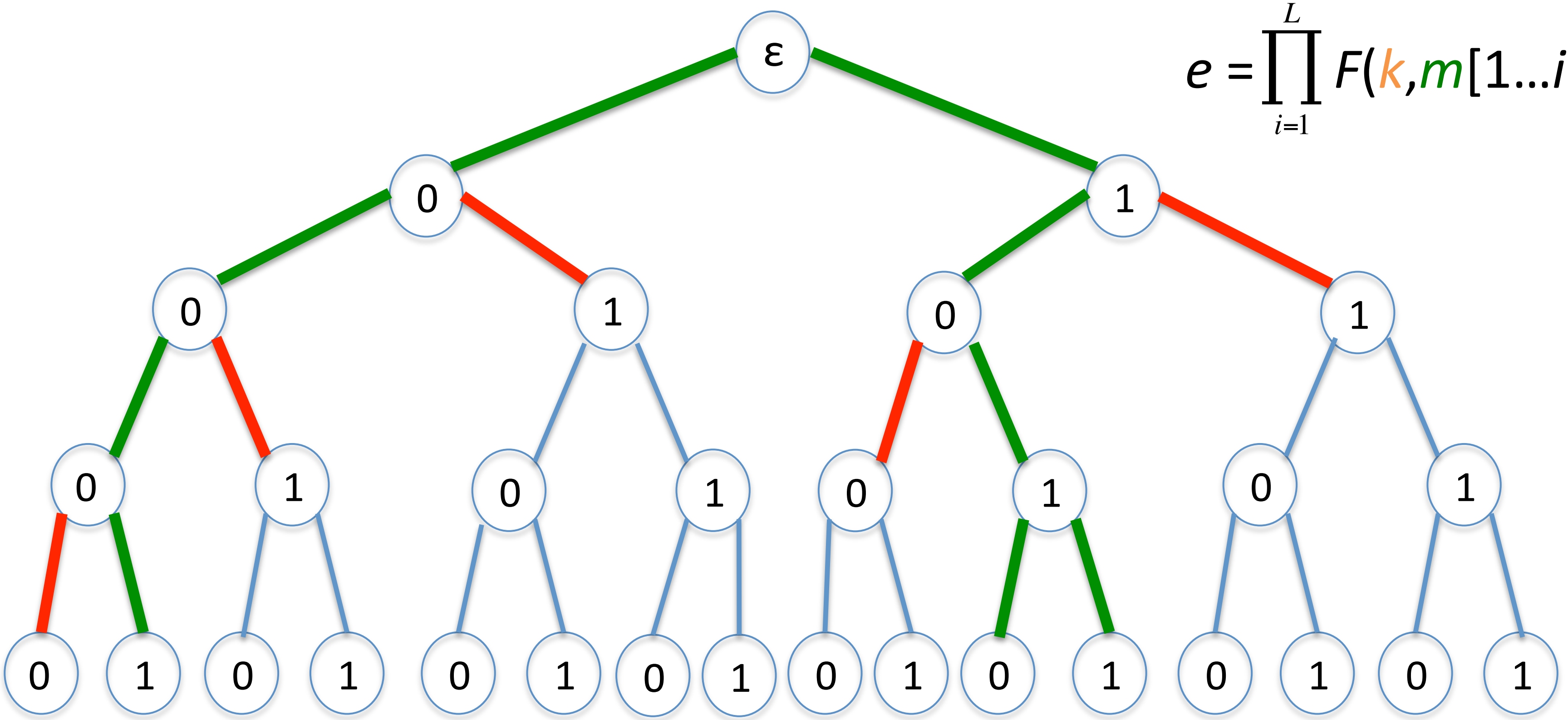
$$\gcd(e^*, \prod \textit{green numbers}) > 1$$

$$e^* = \prod F(k, m^*[1 \ldots i])$$

Smoothness Analysis

$$Sign(sk, m) = g^{1/e} \bmod n$$

$$e = \prod_{i=1}^{L} F(k, m[1 \ldots i])$$

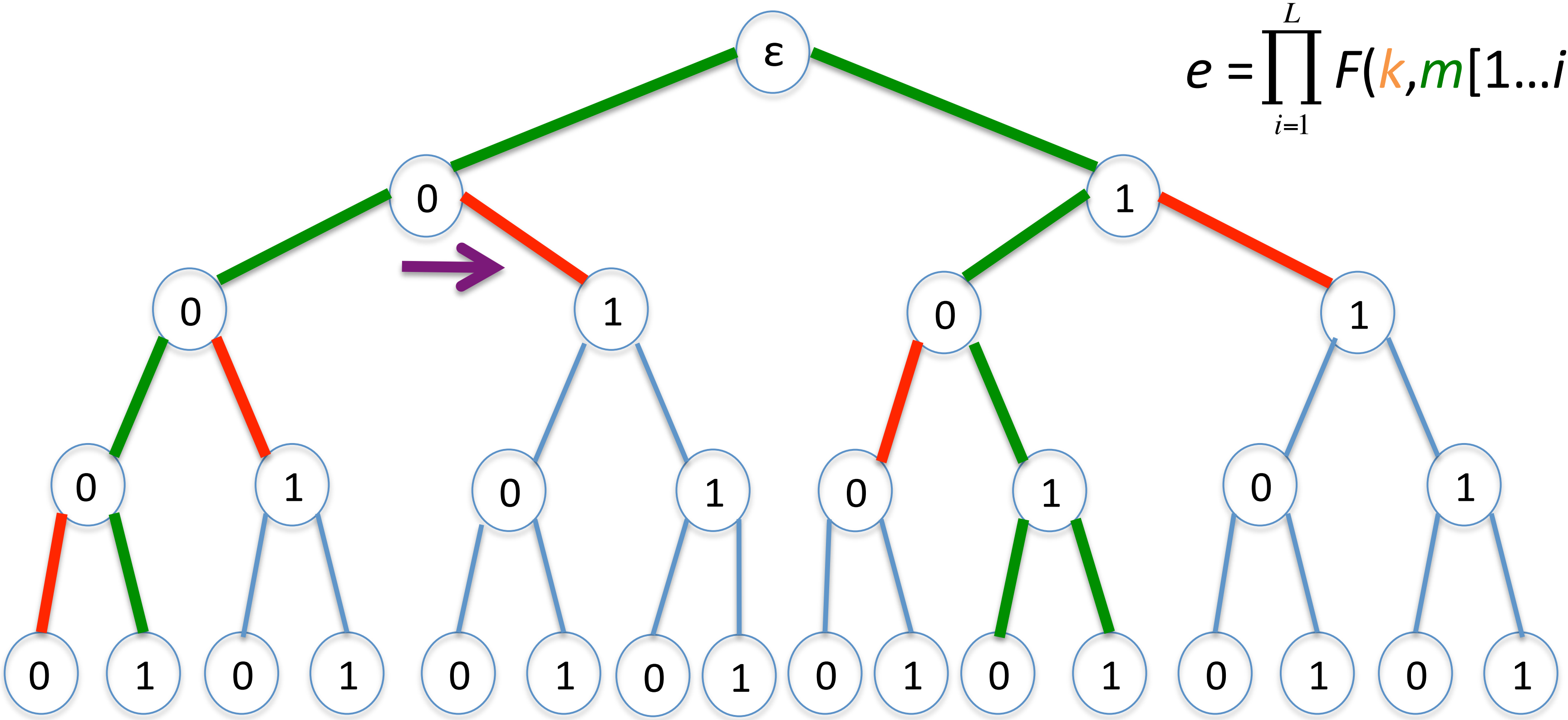


Start analysis with the exit branches. If one of them is smooth, analyze its children and repeat this process recursively.

Smoothness Analysis

$$Sign(sk, m) = g^{1/e} \bmod n$$

$$e = \prod_{i=1}^{L} F(k, m[1 \ldots i])$$

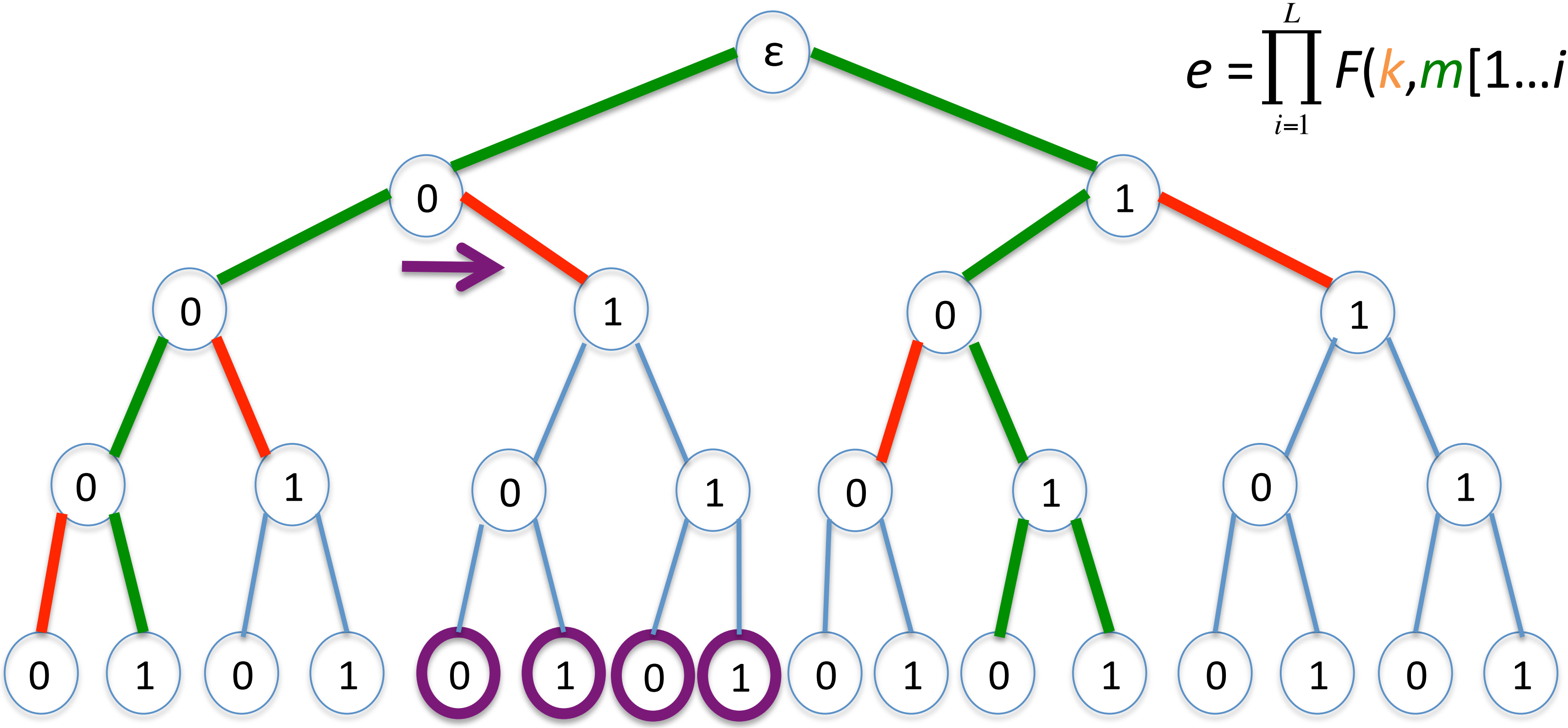


Start analysis with the exit branches. If one of them is smooth, analyze its children and repeat this process recursively.

Smoothness Analysis

$$Sign(sk, m) = g^{1/e} \bmod n$$

$$e = \prod_{i=1}^{L} F(k, m[1...i])$$

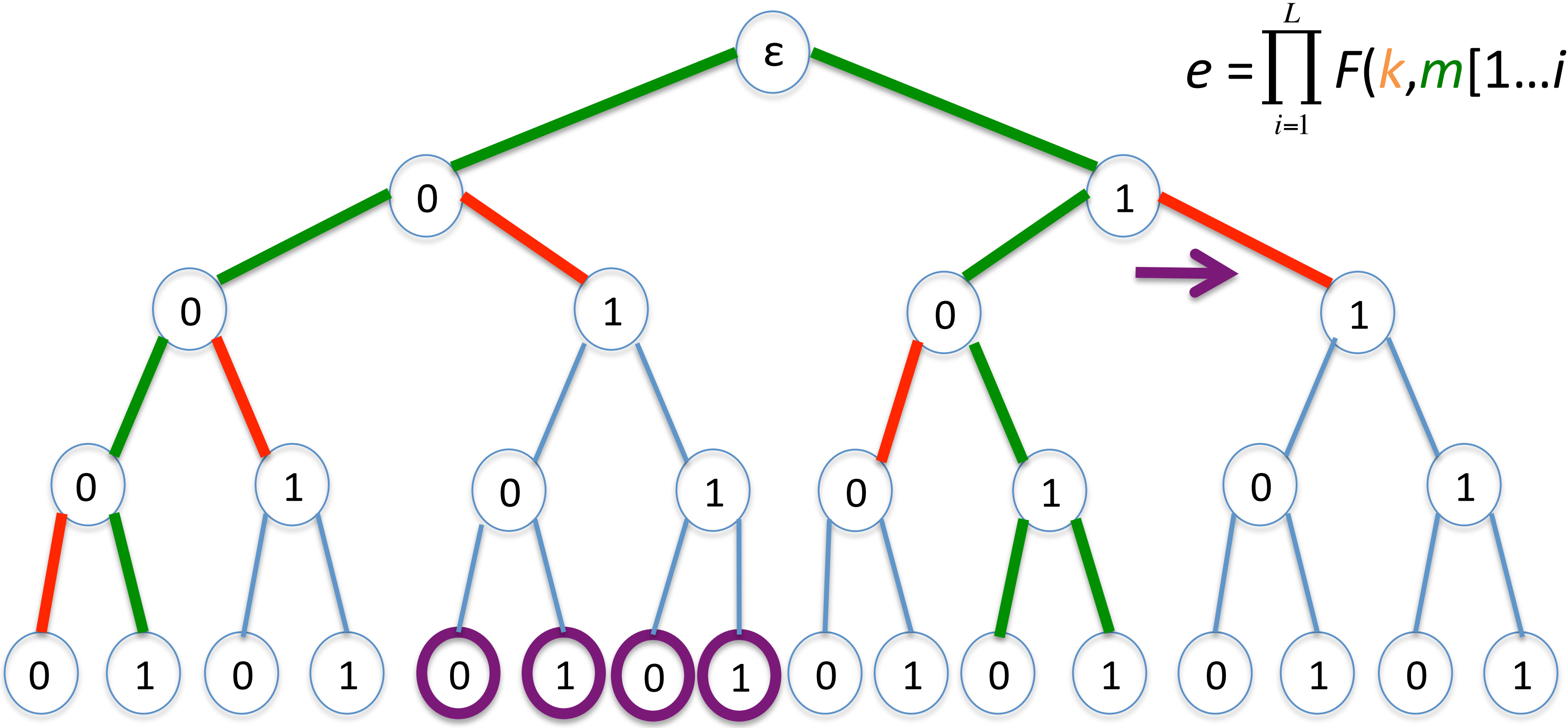


Start analysis with the exit branches. If one of them is smooth, analyze its children and repeat this process recursively.

Smoothness Analysis

$$Sign(sk, m) = g^{1/e} \bmod n$$

$$e = \prod_{i=1}^{L} F(k, m[1 \ldots i])$$

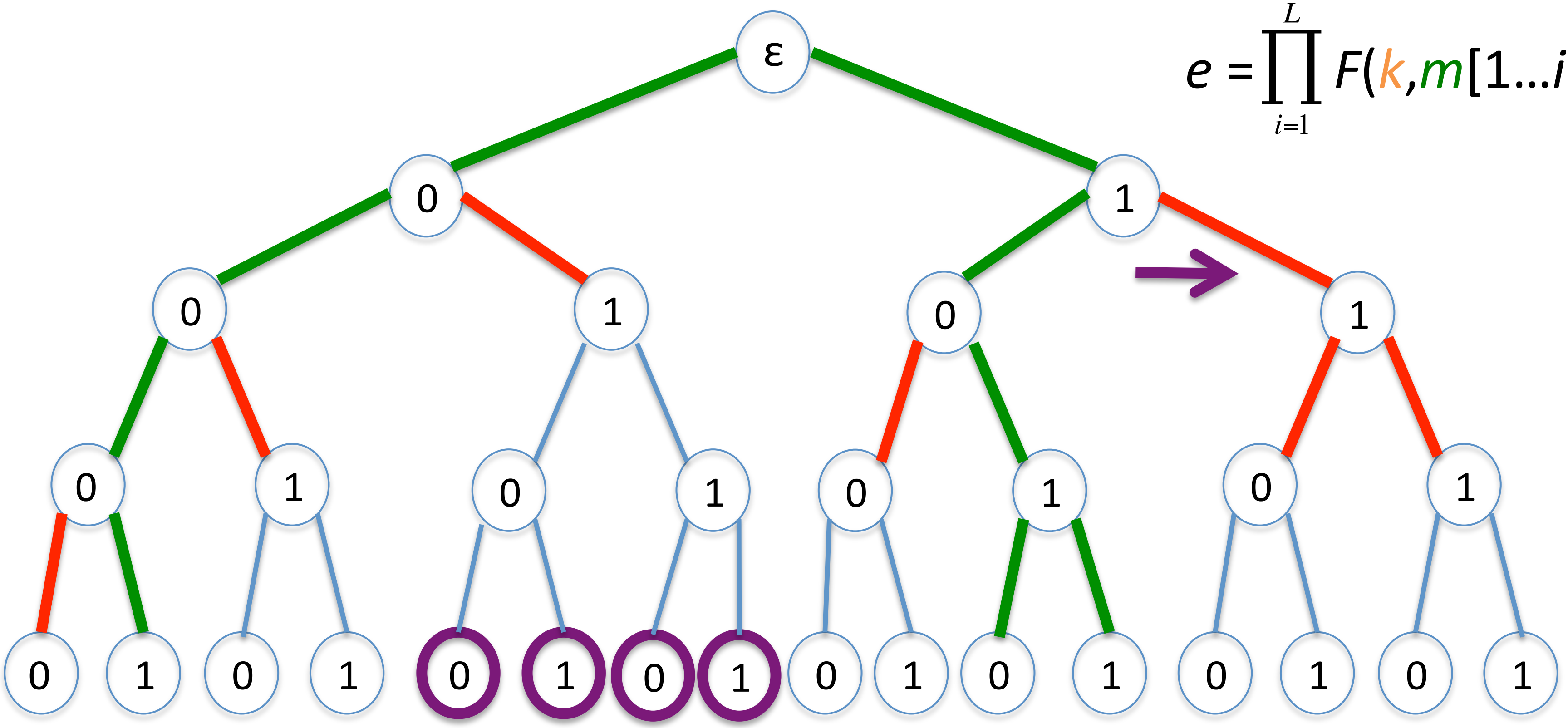


Start analysis with the exit branches. If one of them is smooth, analyze its children and repeat this process recursively.

Smoothness Analysis

$$Sign(sk, m) = g^{1/e} \bmod n$$

$$e = \prod_{i=1}^{L} F(k, m[1 \ldots i])$$

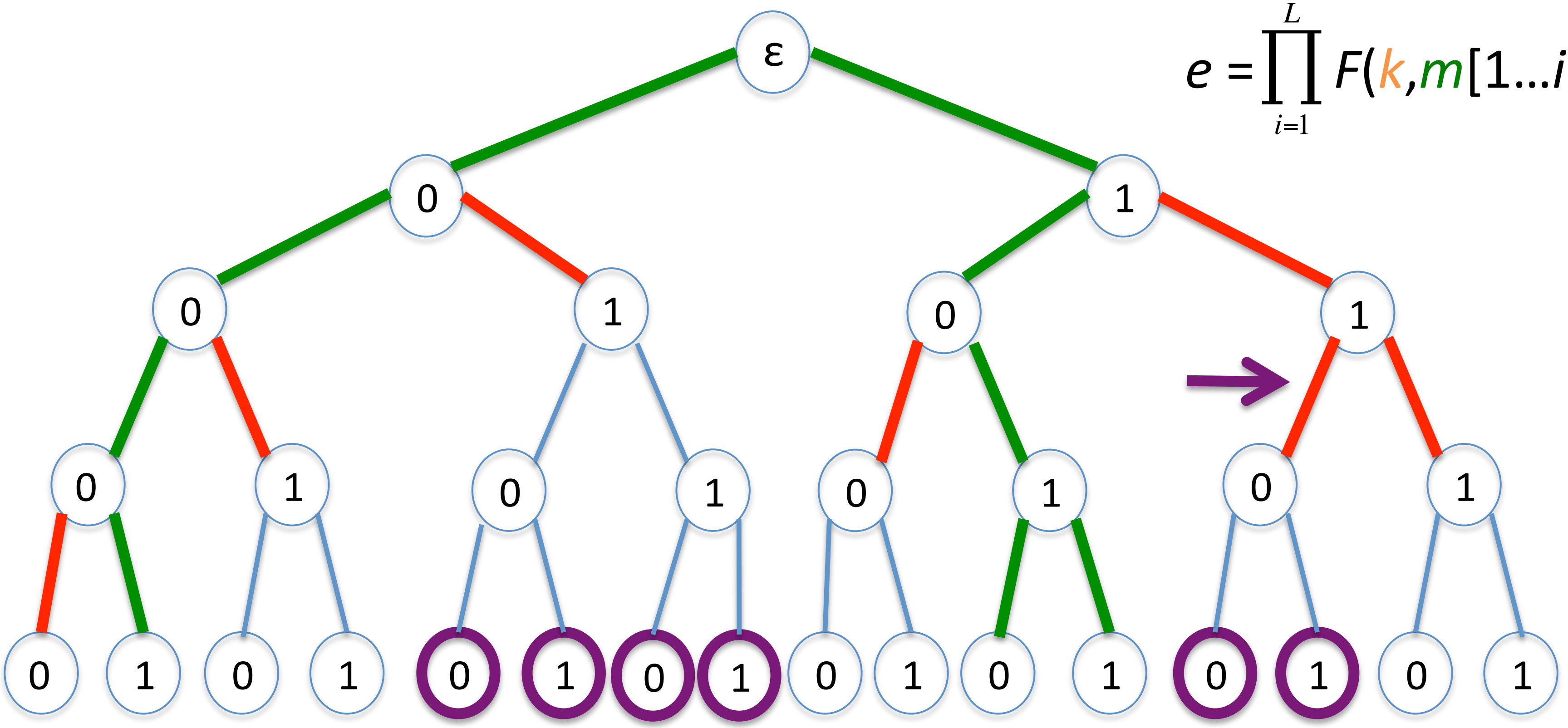


Start analysis with the exit branches. If one of them is smooth, analyze its children and repeat this process recursively.

Smoothness Analysis

$$Sign(sk, m) = g^{1/e} \bmod n$$

$$e = \prod_{i=1}^{L} F(k, m[1 \ldots i])$$

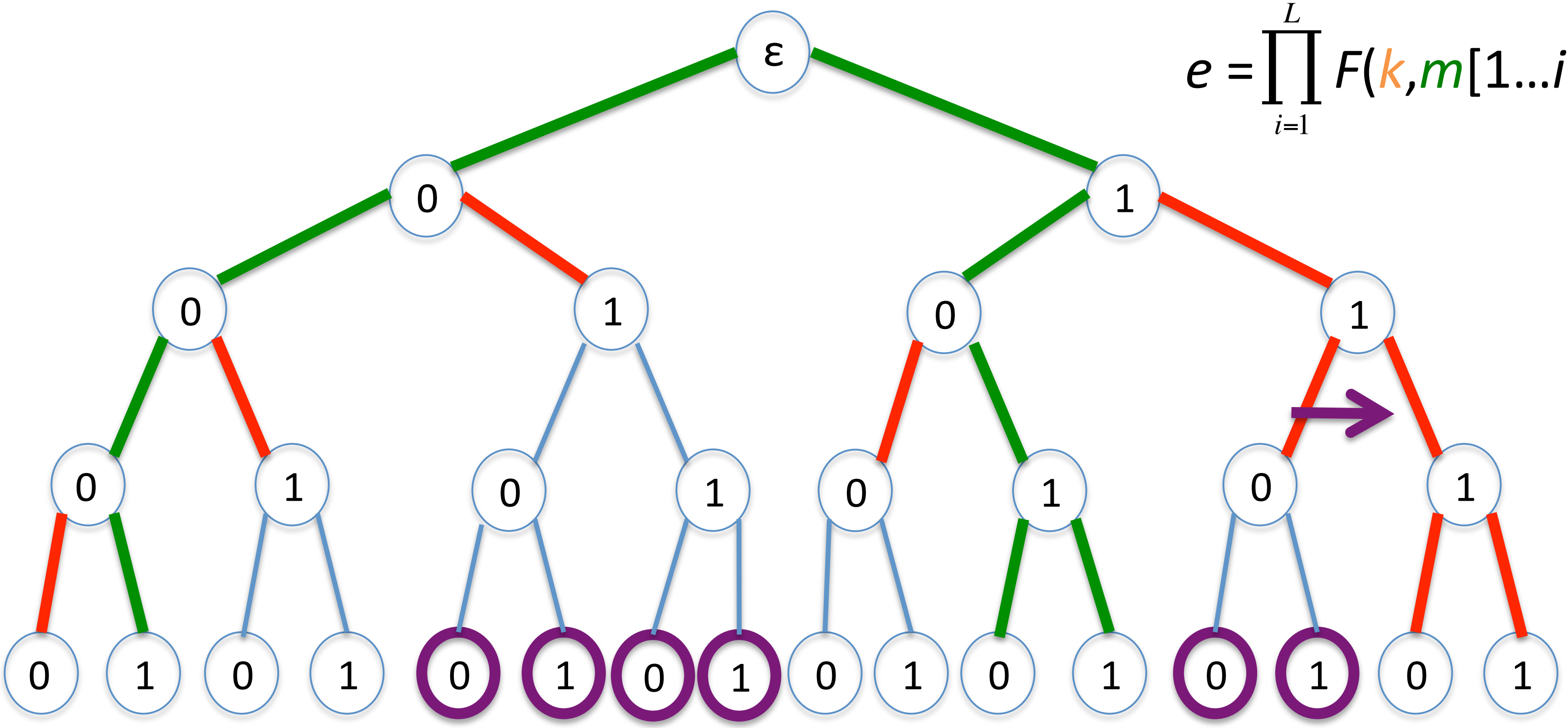


Start analysis with the exit branches. If one of them is smooth, analyze its children and repeat this process recursively.

Smoothness Analysis

$$Sign(sk, m) = g^{1/e} \bmod n$$

$$e = \prod_{i=1}^{L} F(k, m[1 \ldots i])$$

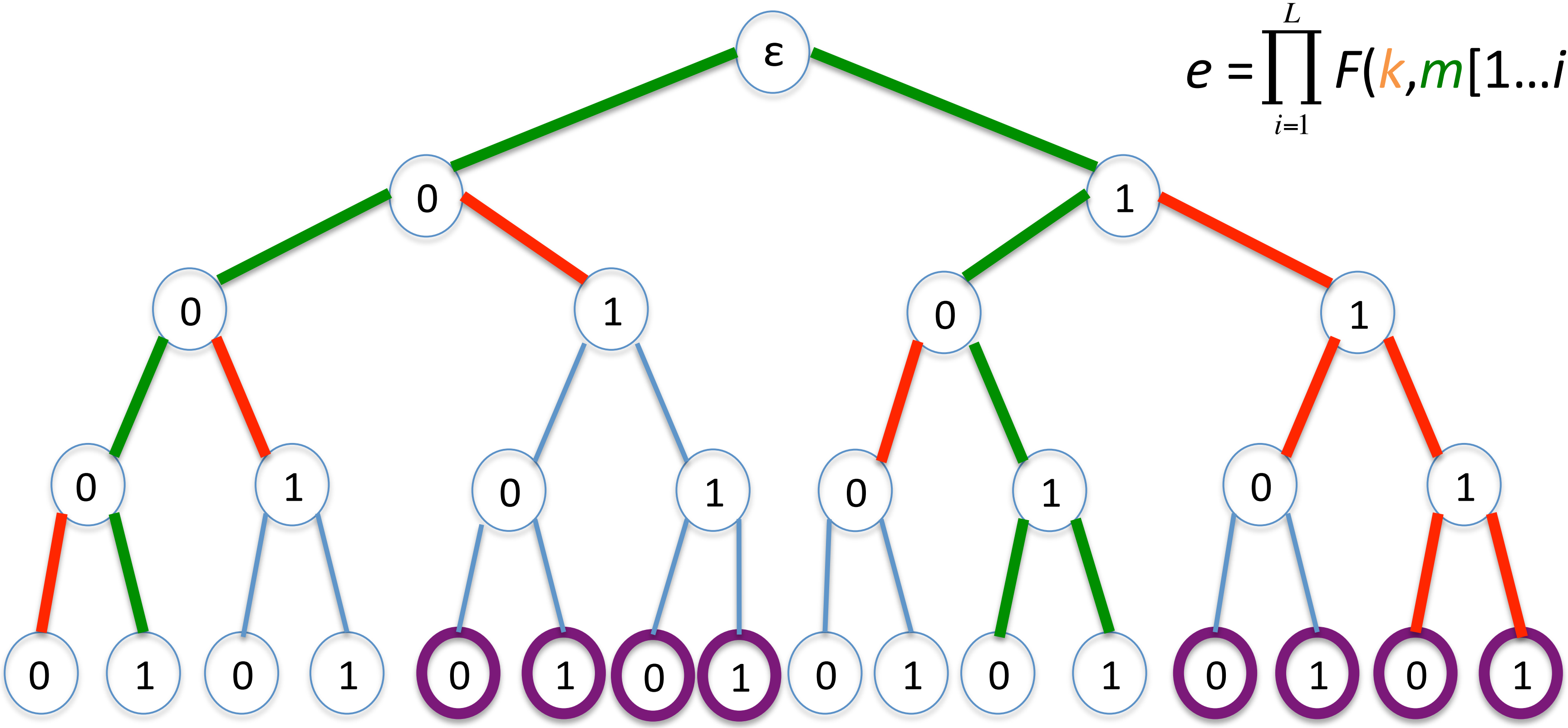


Start analysis with the exit branches. If one of them is smooth, analyze its children and repeat this process recursively.

Smoothness Analysis

$$Sign(sk, m) = g^{1/e} \bmod n$$

$$e = \prod_{i=1}^{L} F(k, m[1 \ldots i])$$

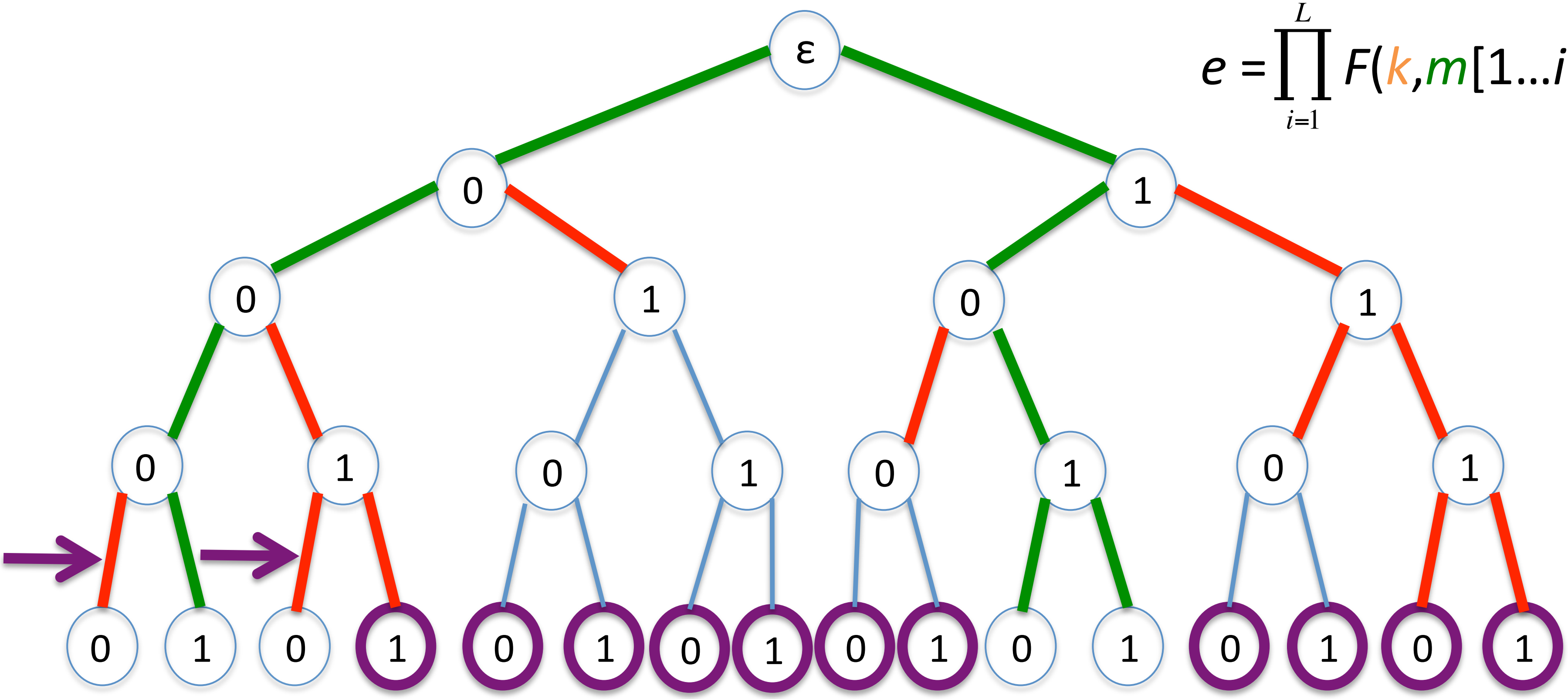


Start analysis with the exit branches. If one of them is smooth, analyze its children and repeat this process recursively.

Smoothness Analysis

$$Sign(sk, m) = g^{1/e} \bmod n$$

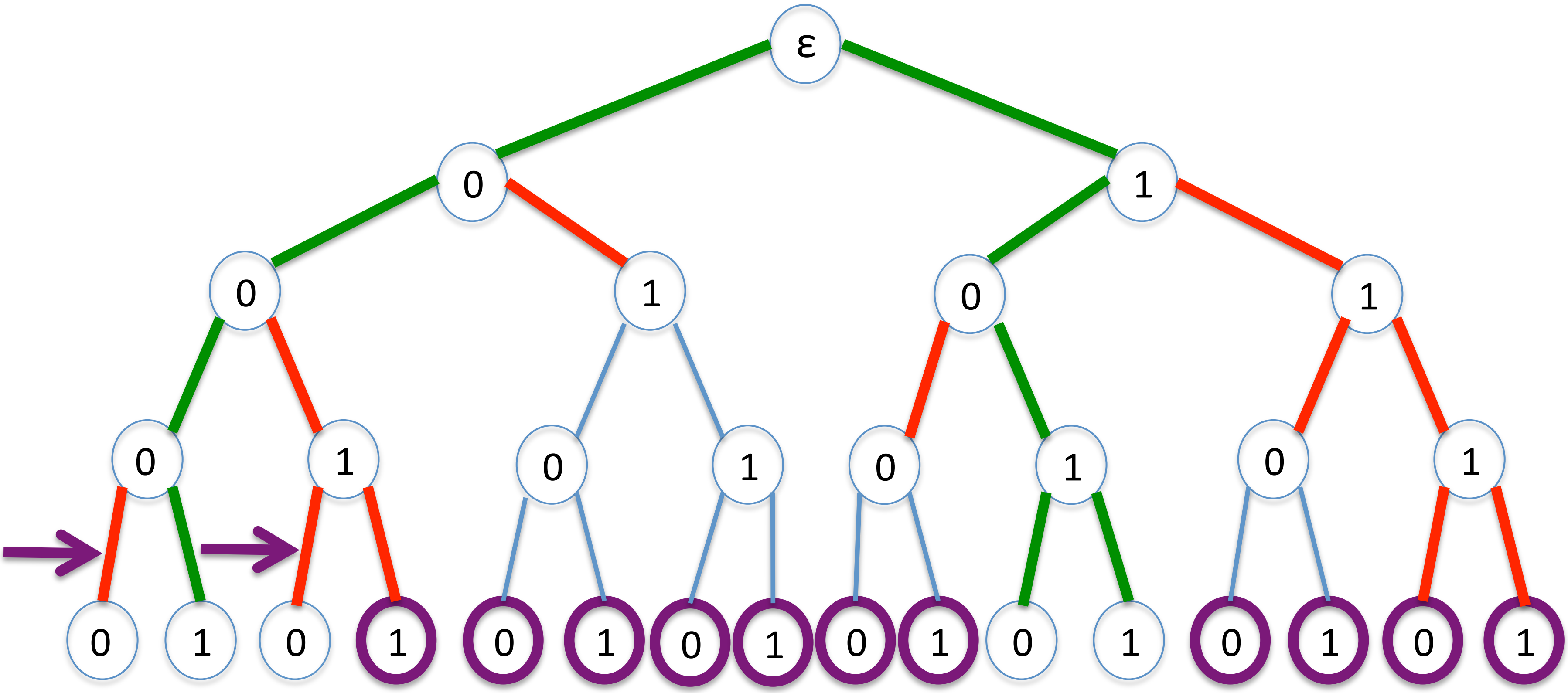
$$e = \prod_{i=1}^{L} F(k, m[1 \ldots i])$$



Start analysis with the exit branches. If one of them is smooth, analyze its children and repeat this process recursively.

Problem: How to deal with exit branches in the last level/non-finished recursions.

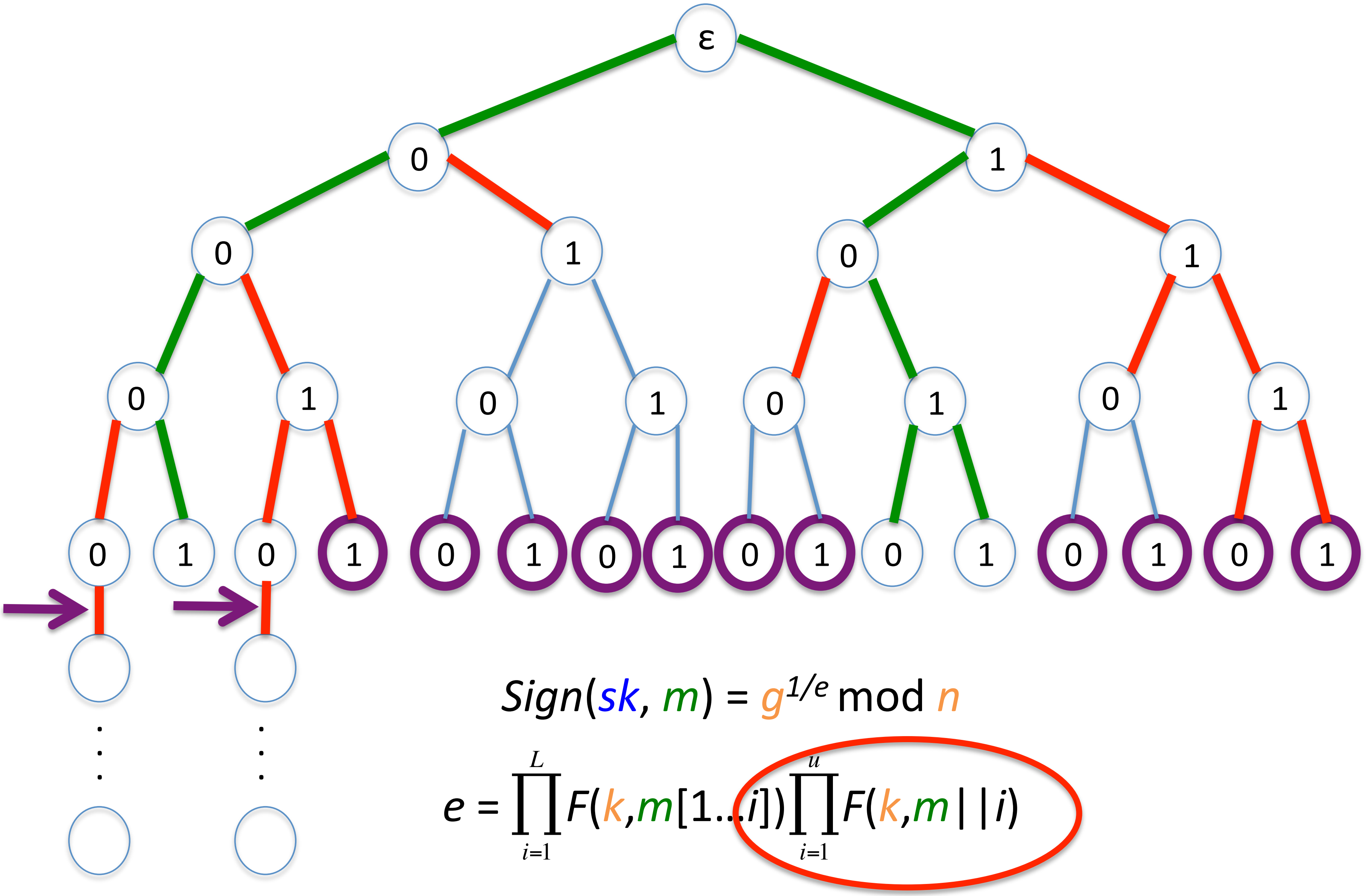
Smoothness Analysis



$$Sign(sk, m) = g^{1/e} \bmod n$$

$$e = \prod_{i=1}^{L} F(k, m[1..i]) \prod_{i=1}^{u} F(k, m || i)$$

Smoothness Analysis

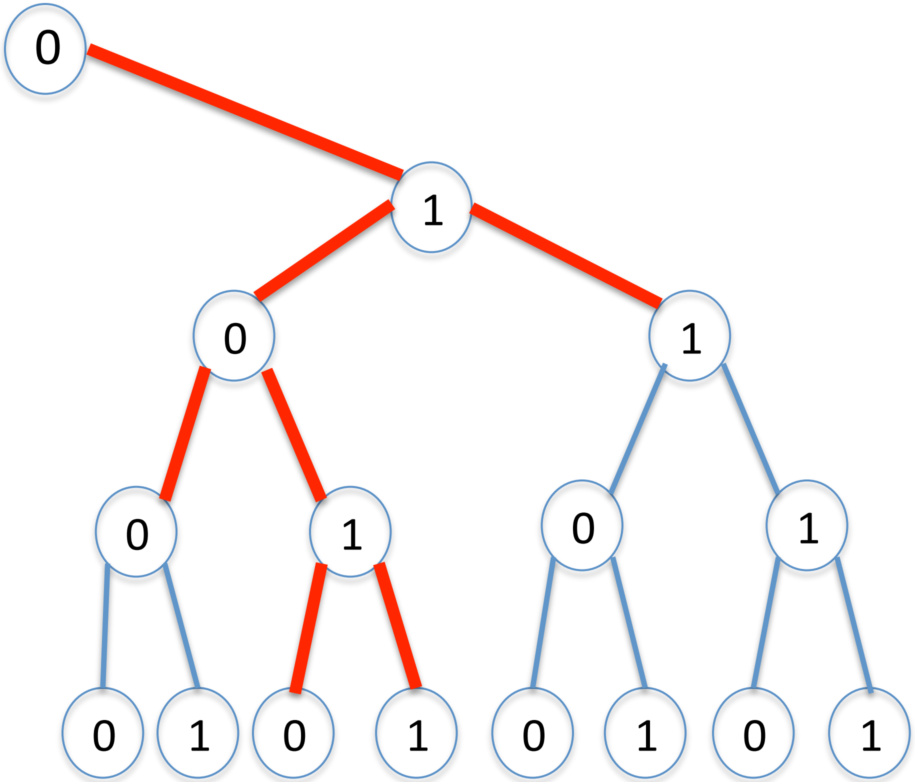


$$Sign(sk, m) = g^{1/e} \bmod n$$

$$e = \prod_{i=1}^{L} F(k, m[1..i]) \prod_{i=1}^{u} F(k, m || i)$$

#Recursions

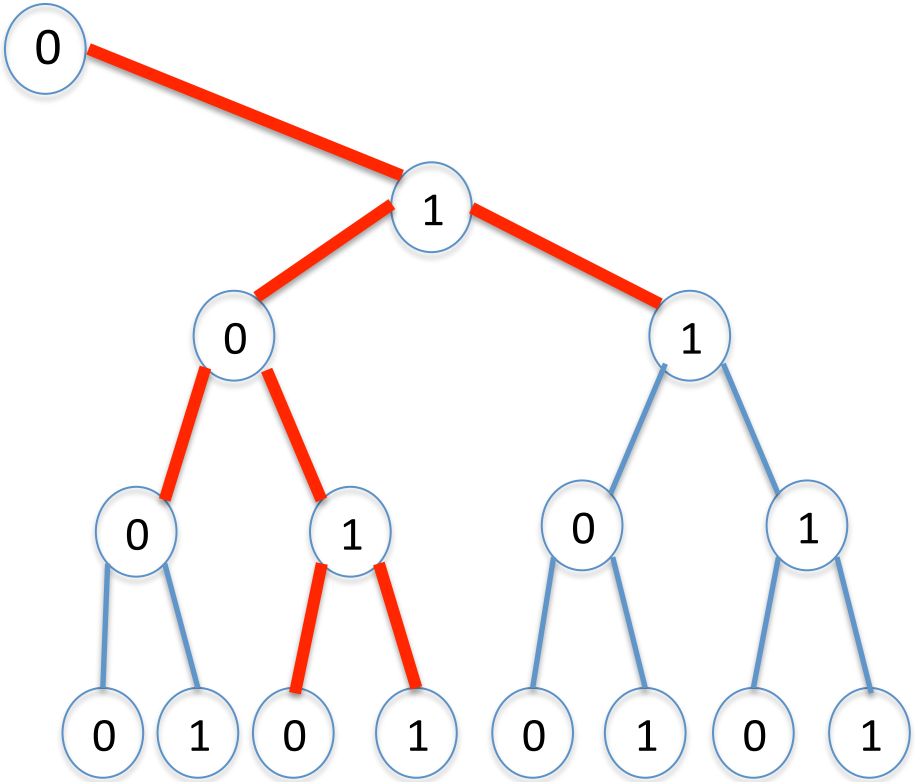
For each exit branch of the tree with overwhelming probability there are at most $2L^2$ recursive calls.



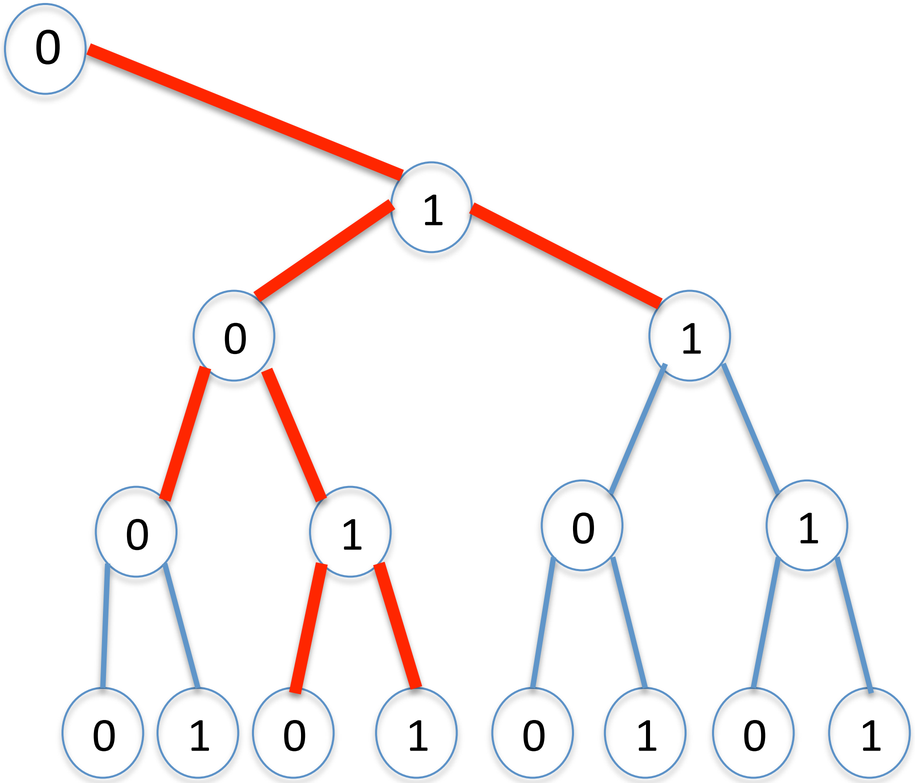
#Recursions

For each exit branch of the tree with overwhelming probability there are at most $2L^2$ recursive calls.

Upper bound at each level: 2L.



#Recursions

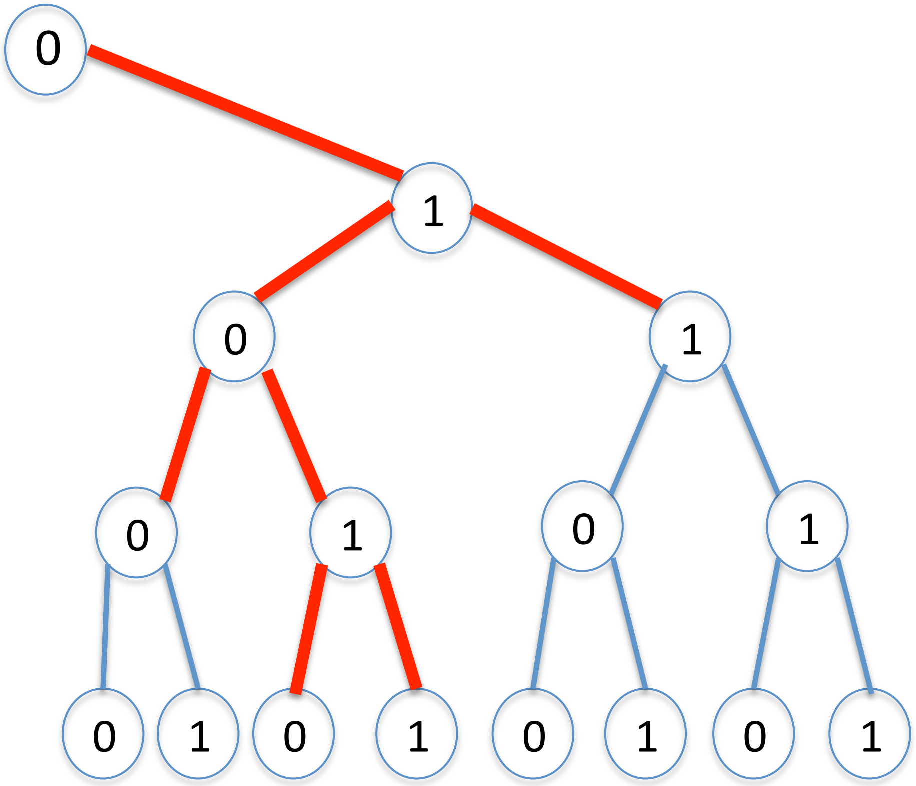


For each exit branch of the tree with overwhelming probability there are at most $2L^2$ recursive calls.

Upper bound at each level: 2L.

Each level can have at most double the calls as the previous one.

#Recursions



For each exit branch of the tree with overwhelming probability there are at most $2L^2$ recursive calls.

Upper bound at each level: 2L.

Each level can have at most double the calls as the previous one.

If the previous level had at most L calls, then the current one has at most 2L. Otherwise the previous level had between L and 2L calls and we apply the Chernoff bound to these Bernoulli random variables.

Smoothness Analysis

In the context of elliptic-curve signatures Coron, Handschuh and Naccache [CHN99] avoided point counting by working over larger underlying fields.

Smoothness Analysis

In the context of elliptic-curve signatures Coron, Handschuh and Naccache [CHN99] avoided point counting by working over larger underlying fields.

As in our case a naïve analysis of the smoothness property would result in very bad parameters, so they only increased the size of the underlying field enough to get smooth curves with low probability.

Smoothness Analysis

In the context of elliptic-curve signatures Coron, Handschuh and Naccache [CHN99] avoided point counting by working over larger underlying fields.

As in our case a naïve analysis of the smoothness property would result in very bad parameters, so they only increased the size of the underlying field enough to get smooth curves with low probability.

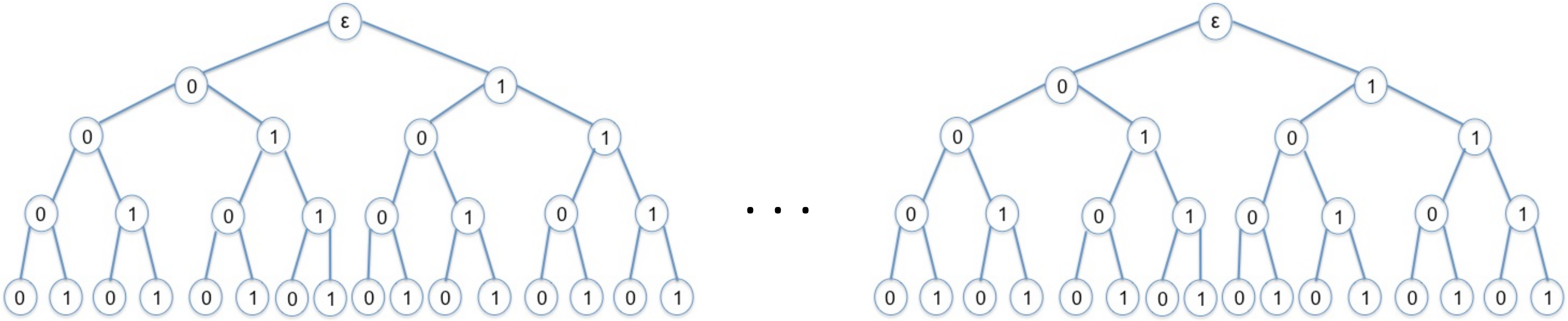
Iterate the protocol over many independent curves to guarantee that at least one of them is not smooth.

Smoothness Analysis

In the context of elliptic-curve signatures Coron, Handschuh and Naccache [CHN99] avoided point counting by working over larger underlying fields.

As in our case a naïve analysis of the smoothness property would result in very bad parameters, so they only increased the size of the underlying field enough to get smooth curves with low probability.

Iterate the protocol over many independent curves to guarantee that at least one of them is not smooth.



Summary

- Our scheme does not need to generate primes while signing.

Summary

- Our scheme does not need to generate primes while signing.

- Public-key and signature size competitive with other schemes.

Summary

- Our scheme does not need to generate primes while signing.
- Public-key and signature size competitive with other schemes.
- Weakness: verification performance.

Summary

- ✧ Our scheme does not need to generate primes while signing.
- ✧ Public-key and signature size competitive with other schemes.
- ✧ Weakness: verification performance.
- ✧ Could be a stepping-stone to more practical signatures in the standard model.

Thank You!