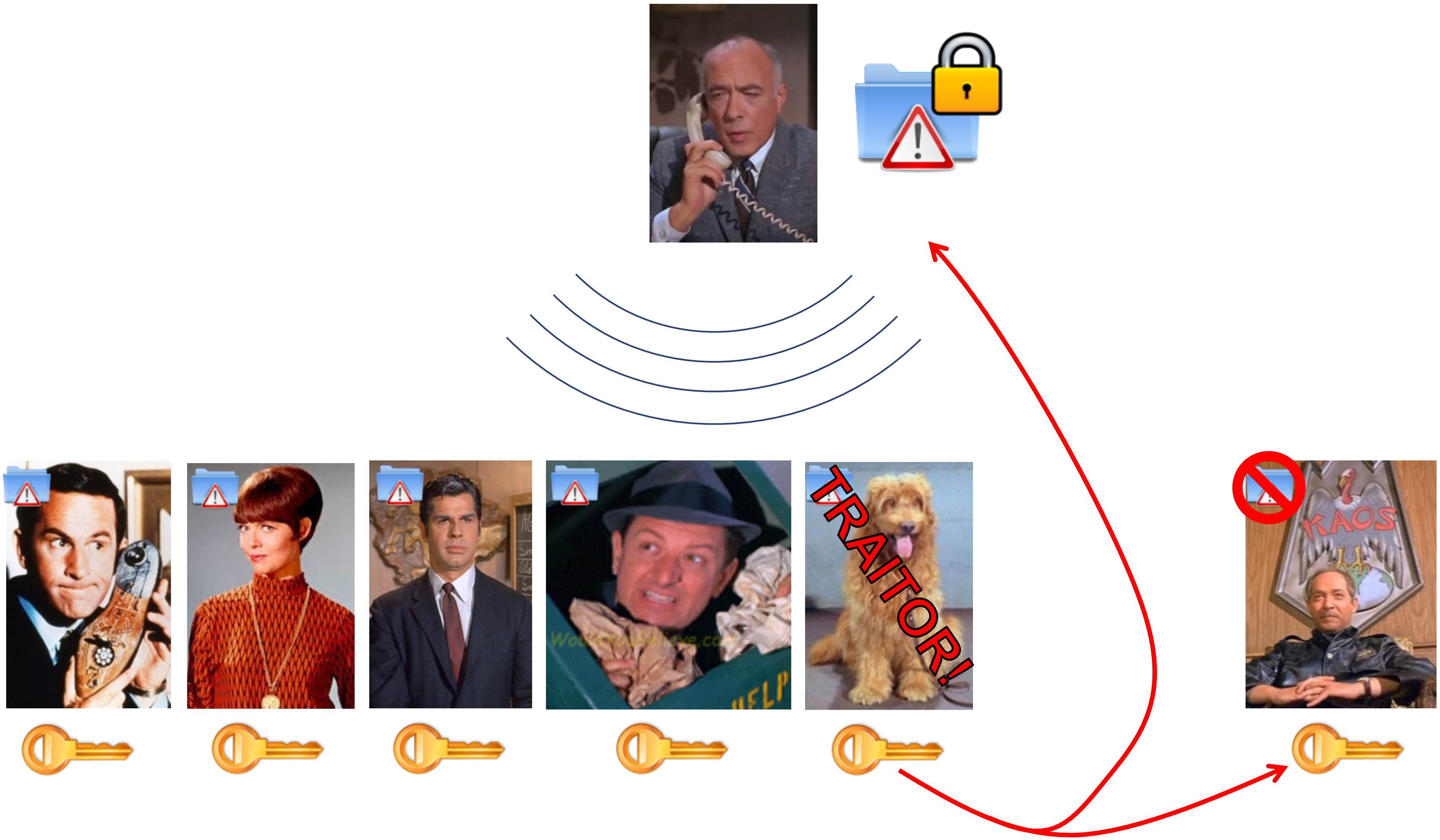
Anonymous Traitor Tracing

Mark Zhandry

Join work with Ryo Nishimaki, Daniel Wichs

Traitor Tracing



Goal: Using leaked key, identify traitor to revoke key, punish, disincentivize

Considerations

- What's wrong with u🔑 = (🔑, u) ?
- What if adversary obfuscates **Dec(u🔑, ·)** ?
- What if broken key that only recovers half the message?

➡ Assume traitor produces pirate decoder: ☠:C → {0,1}

➡ Only given oracle access to ☠

- What if 2 spys? **k** spys?

➡ Allow adversary to get arbitrarily many secret keys
(Bounded collusion also interesting)

Syntax

Setup(): Outputs **(msk,pk)**

Enc(pk, m): Outputs a ciphertext **c**

KeyGen(msk, u∈[N]): Outputs user u’s secret key u🔑

Dec(u🔑, c): Outputs **m**

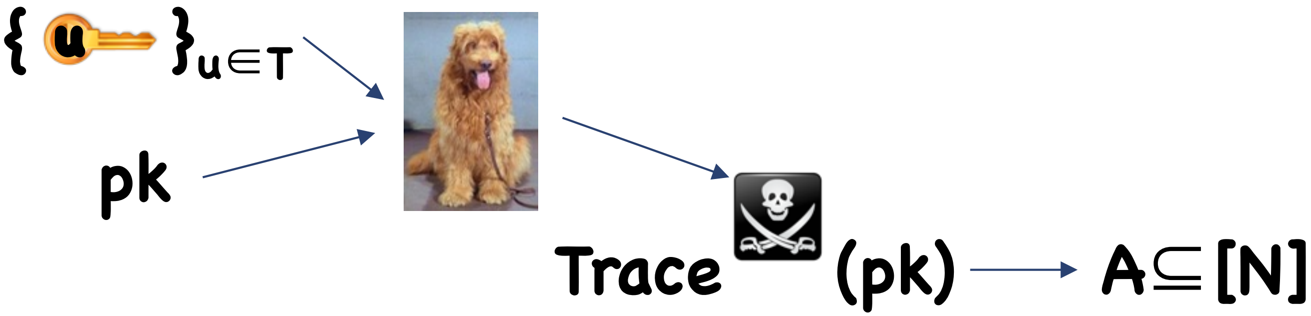
Trace☠(pk): Outputs an “accused set” **A⊆[N]**

Properties

Correctness: **Dec(**u🔑**,** 🔒📁 **) =** 📁 for all **u**

Semantic Security: w/o any u🔑, 🔒📁 hides 📁

Traceability:



{ u🔑 }$_{u \in T}$

pk

Trace☠**(pk)** → $A \subseteq [N]$

- $A \setminus T = \varnothing$
- If ☠ "usefull" (breaks 🔒📁), then $A \neq \varnothing$

A Trivial System

Each user gets own public key/secret key for PKE scheme

Ciphertext = encryption under each public key

Tracing: encrypt **m** under several public key, junk for others

- Successful decryption → Traitor

Limitation: parameter sizes, running times grow with **N**

Goal: minimize **|c|, |pk|, |🔑u|, |msk|**
(Also, handle exponential **N**)

Prior Work

Combinatorial (CFN’94, …)

- Bounded collusion **k**
- Very weak generic assumptions (OWF, PKE)
- State of the art: **|c|, |pk|, |🔑u| = poly(k, log N)**

Algebraic (BF’99, BSW’06, …)

- Bounded or unbounded collusion
- Specific assumptions (DDH, Subgroup Decision)
- State of the art for unbounded: **|c|, |pk|, |🔑u| =** $O(N^{\frac{1}{2}})$

Obfuscation-Based (GGHRSW’13,BZ’14)

- Generally always unbounded collision
- Extremely strong assumptions (iO, FE)
- State of the art: **|c|, |pk|, |🔑u| = polylog(N)**

Who Keeps Track of User Info?

After tracing, get index **u** of user (integer from **1** to **N**)

- Sufficient for revocation
- How to prosecute? Maintain database:

u=1 → Address 1, Credit card number 1
u=2 → Address 2, Credit card number 2
…

This approach: ability to punish implies <u>lack</u> of anonymity

Q: Are tracing an anonymity at odds?

Embedding Arbitrary Info in Key

Why not set **u** = “Address, Credit card number”?

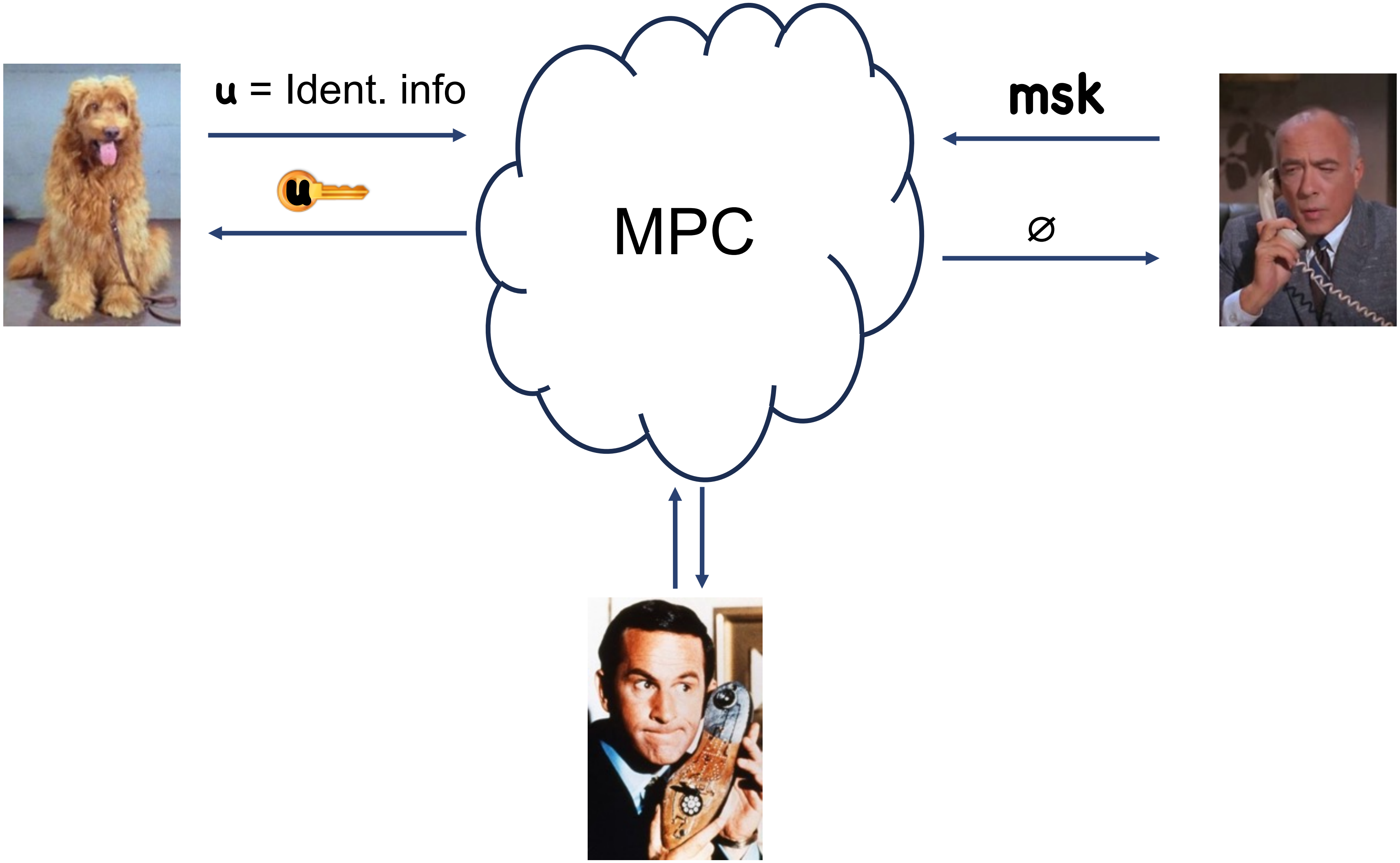
- Length of identifying info $\mathbf{L}$ → $\mathbf{N = 2^L}$

- Current systems: **N** polynomial

  → **L** is logarithmic

- To embed arbitrary info, need exponential number of identities

Anonymity



u = Ident. info

u

MPC

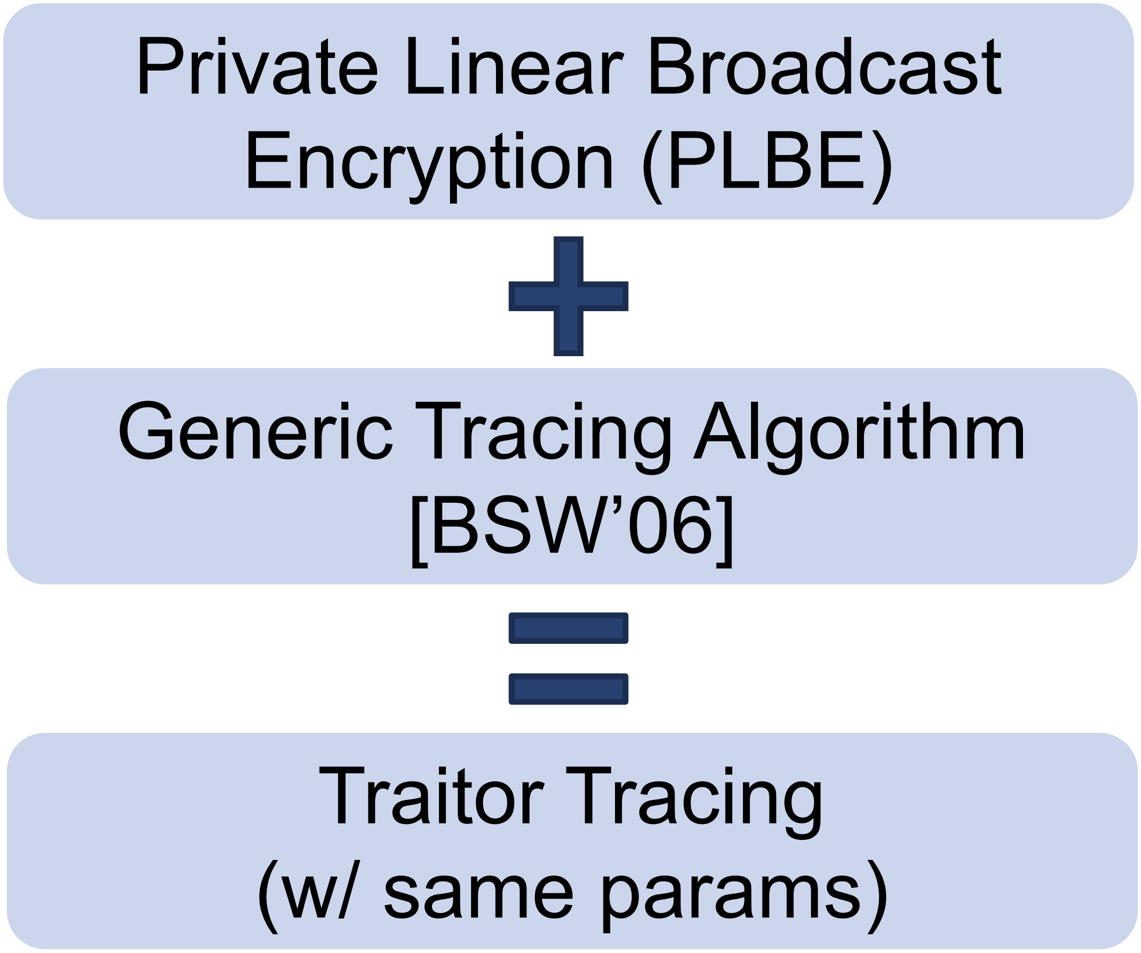
msk

∅

Verification authority

Previous Traitor Tracing

Formula for essentially all schemes with unbounded collusions:



Private Linear Broadcast Encryption (PLBE)

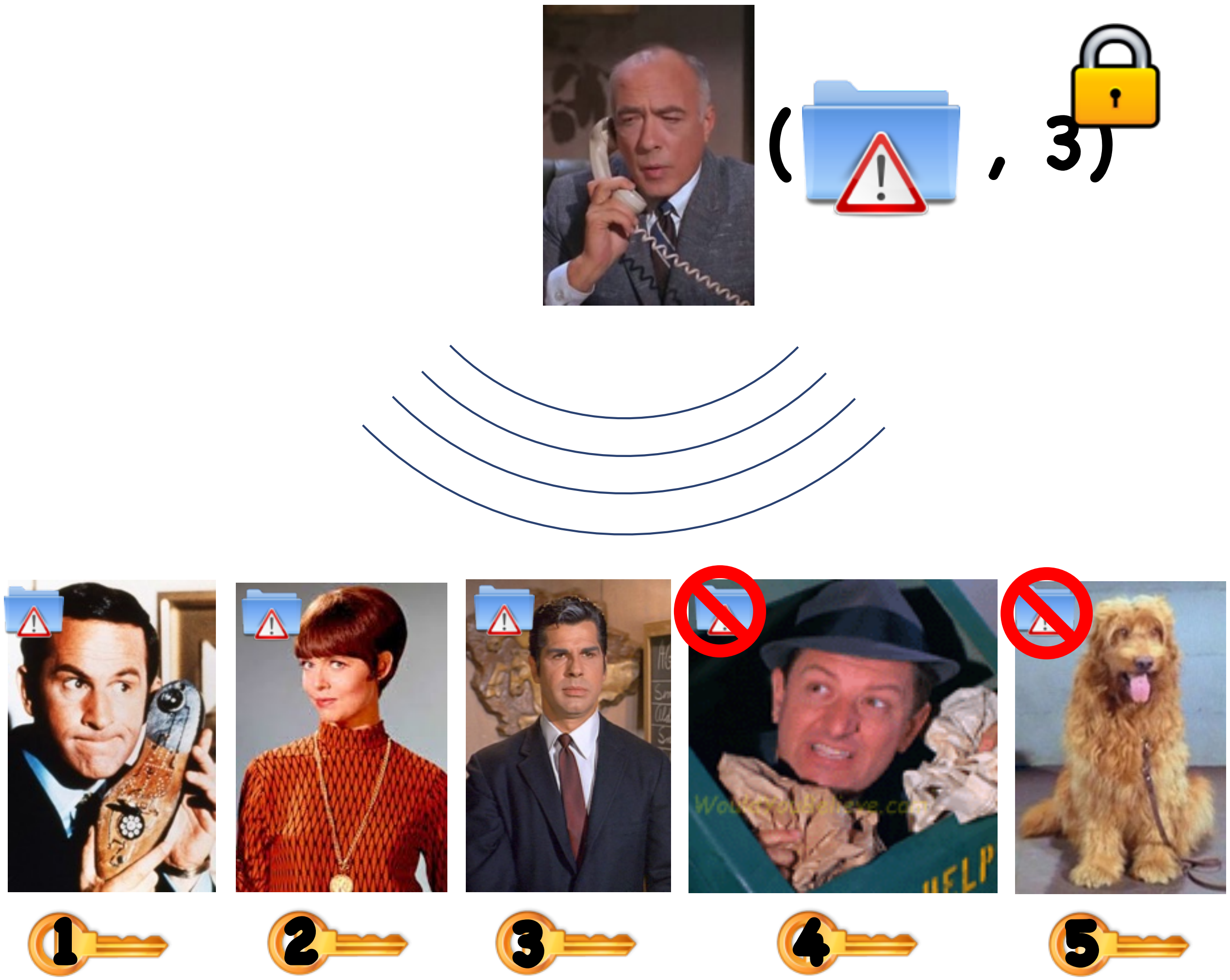
+

Generic Tracing Algorithm [BSW’06]

=

Traitor Tracing (w/ same params)

Private Linear Broadcast Encryption



Functionality: encrypt to intervals

Security: as little info about interval leaked as possible

Private Linear Broadcast Encryption

ID = {1, …, N}

Setup(): Outputs **(msk,pk)**

Enc(pk, m, v∈[0,N]): Outputs a ciphertext **c**

KeyGen(msk, u∈[N]): Outputs user **u**’s secret key 🔑u

Dec(🔑u, c): Outputs **m**

Properties of PLBE

Correctness: **Dec(u🔑, (📁 , v)🔒) = 📁** if $u \le v$

Semantic Security:

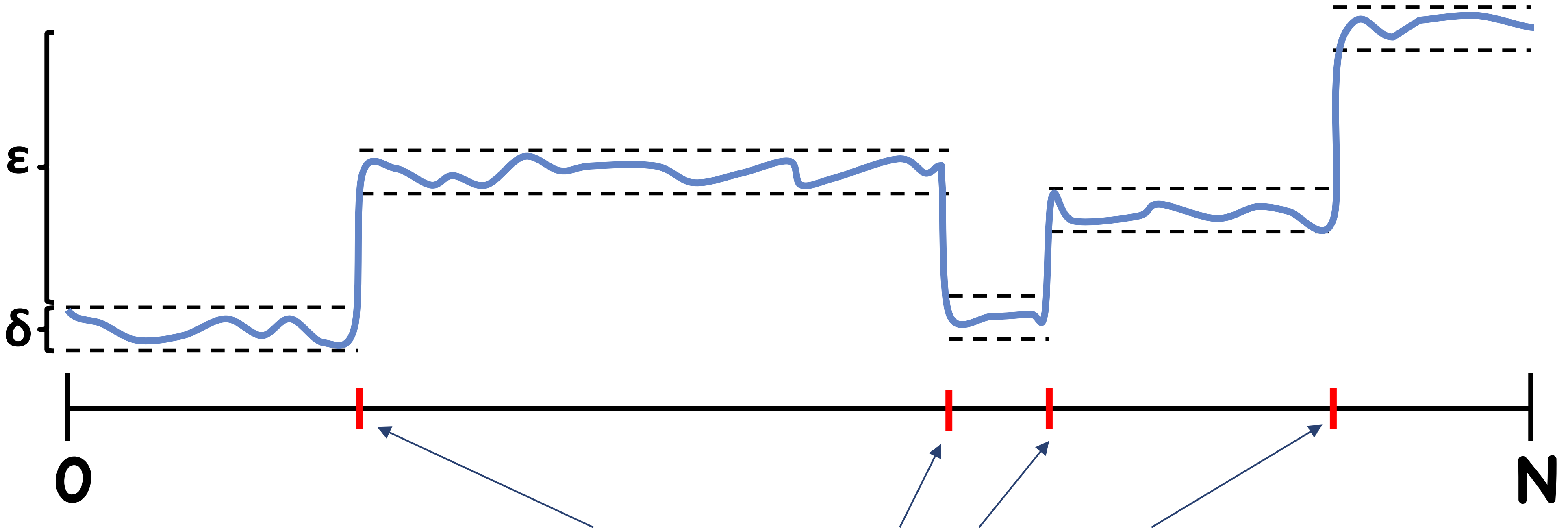
Enc(pk, (📁 , 0)🔒) reveals no info about 📁
even given many u🔑

Recipient privacy:

Cannot distinguish **Enc(📁 , u)🔒** from **Enc(📁 , u-1)🔒**
unless you know u🔑

TT from PLBE

f(v) = Pr[☠ decrypts (⚠ , v)🔒]

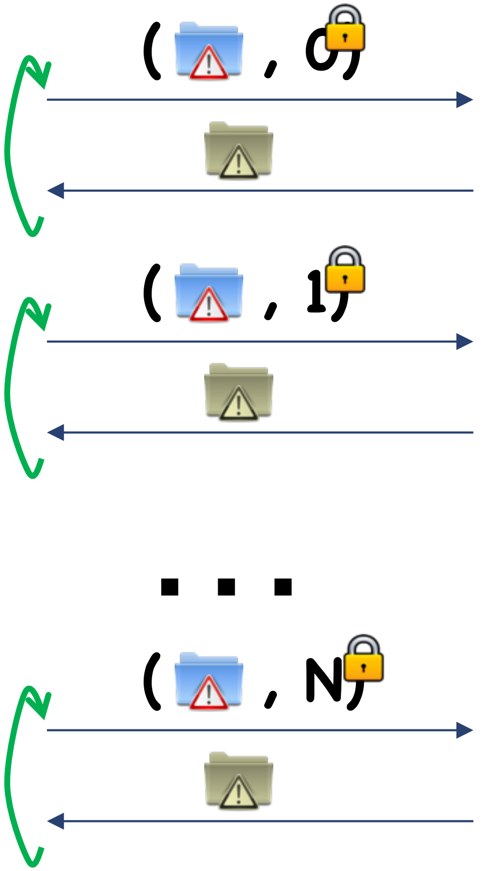


🔑 owned by adversary

PLBE security → δ negligible

Decoder functionality → ε “large”

Tracing PLBE [BSW'06]



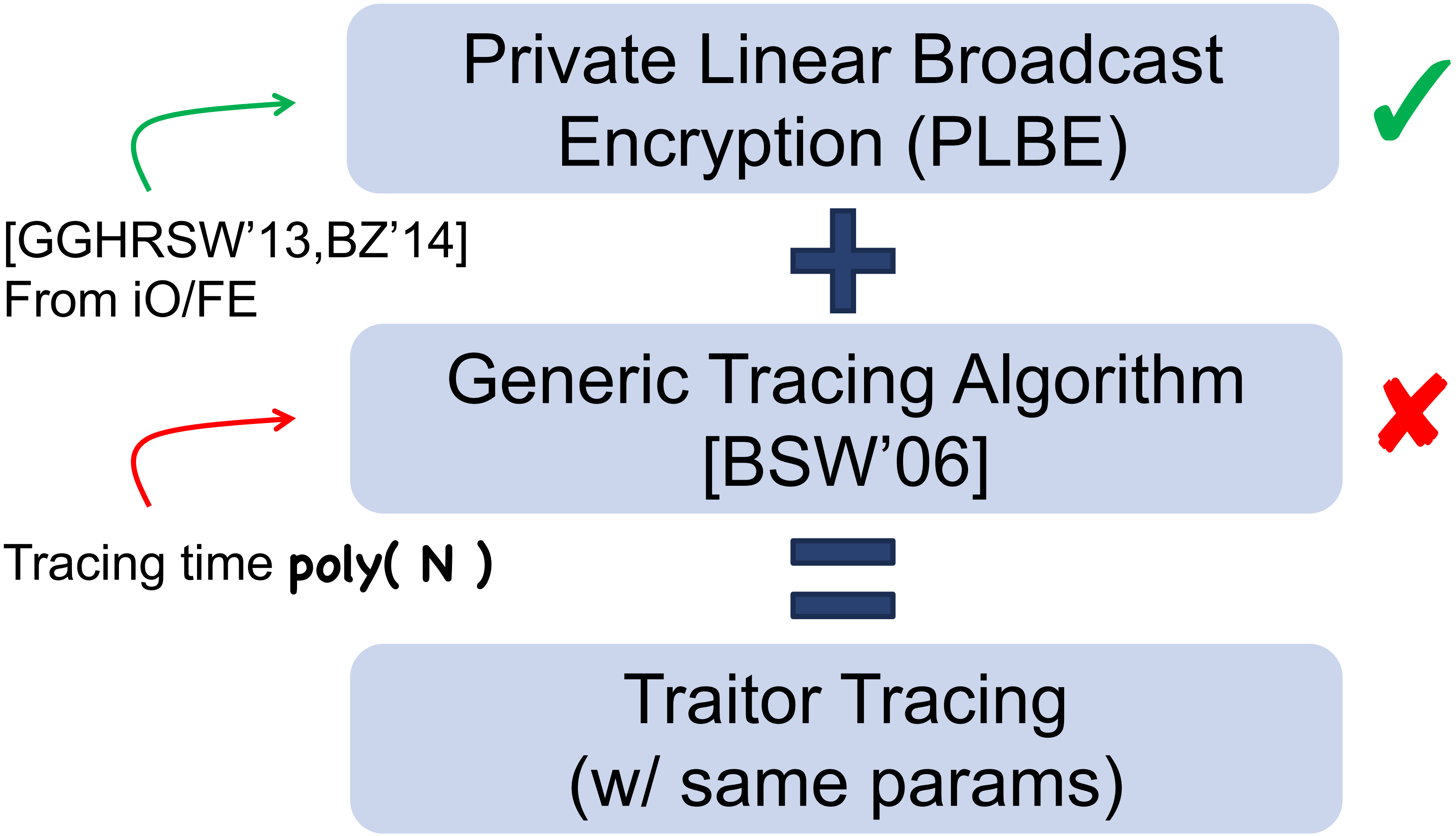
p_0 = Pr[📁 = 📁]

p_1 = Pr[📁 = 📁]

p_N = Pr[📁 = 📁]

Output any **u** for which $|p_{u-1} - p_u|$ is large

Large-Identity Traitor Tracing from PLBE



Private Linear Broadcast Encryption (PLBE) ✓

[GGHRSW’13,BZ’14]
From iO/FE

+

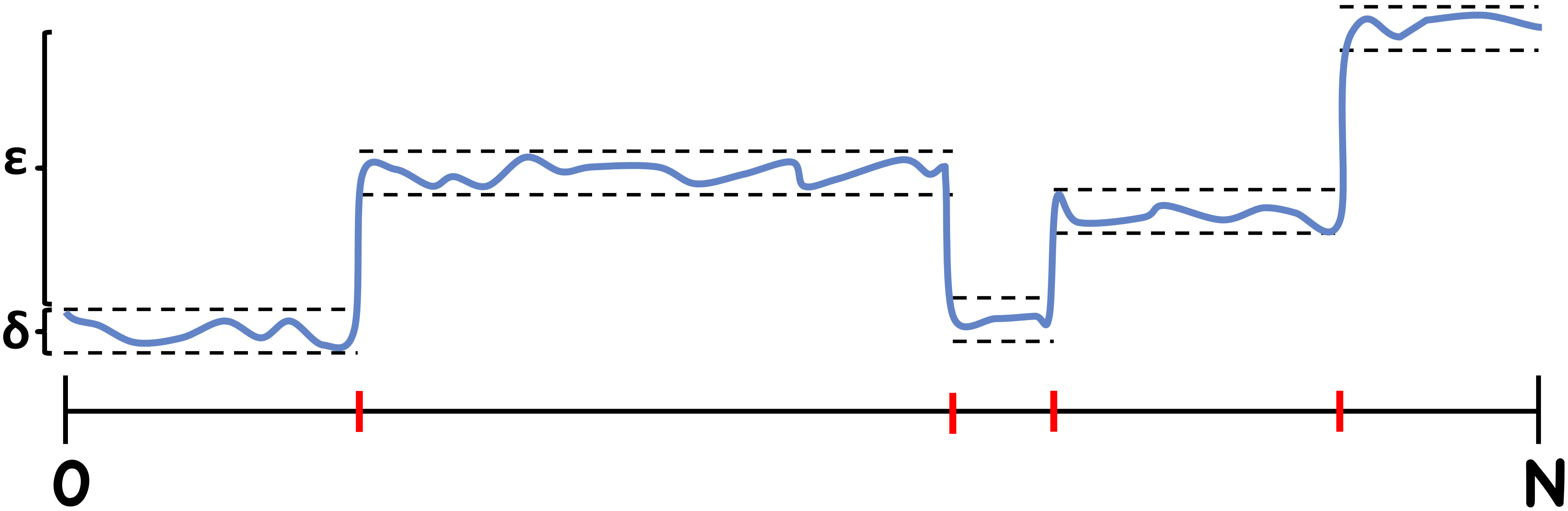
Generic Tracing Algorithm [BSW’06] ✗

Tracing time **poly(N)**

=

Traitor Tracing (w/ same params)

Algorithmic Problem: Oracle Jump Finding



Given oracle access to **f: [0,N] → [0,1]**

- Several “jumps”
- Between jumps, f varies minimally
- At jump, arbitrary change
- **f(0)** small, **f(N)** large (implies noticeable change at some jump)

Goal: Find location of one of the jumps

Oracle Jump Finding

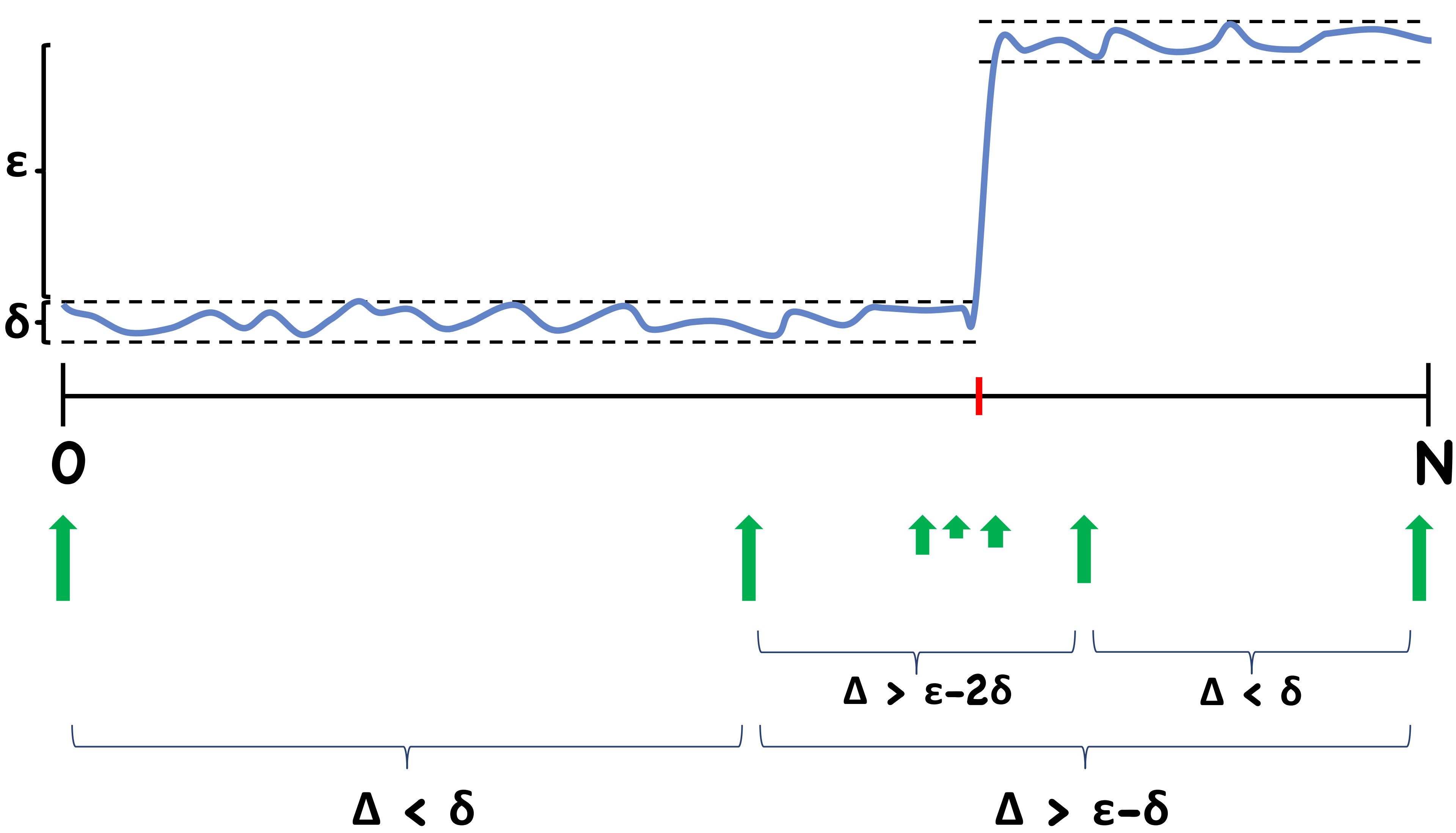
BSW’06 alg → Linear search to find jump

- Visits every point, so running time **O(N)**

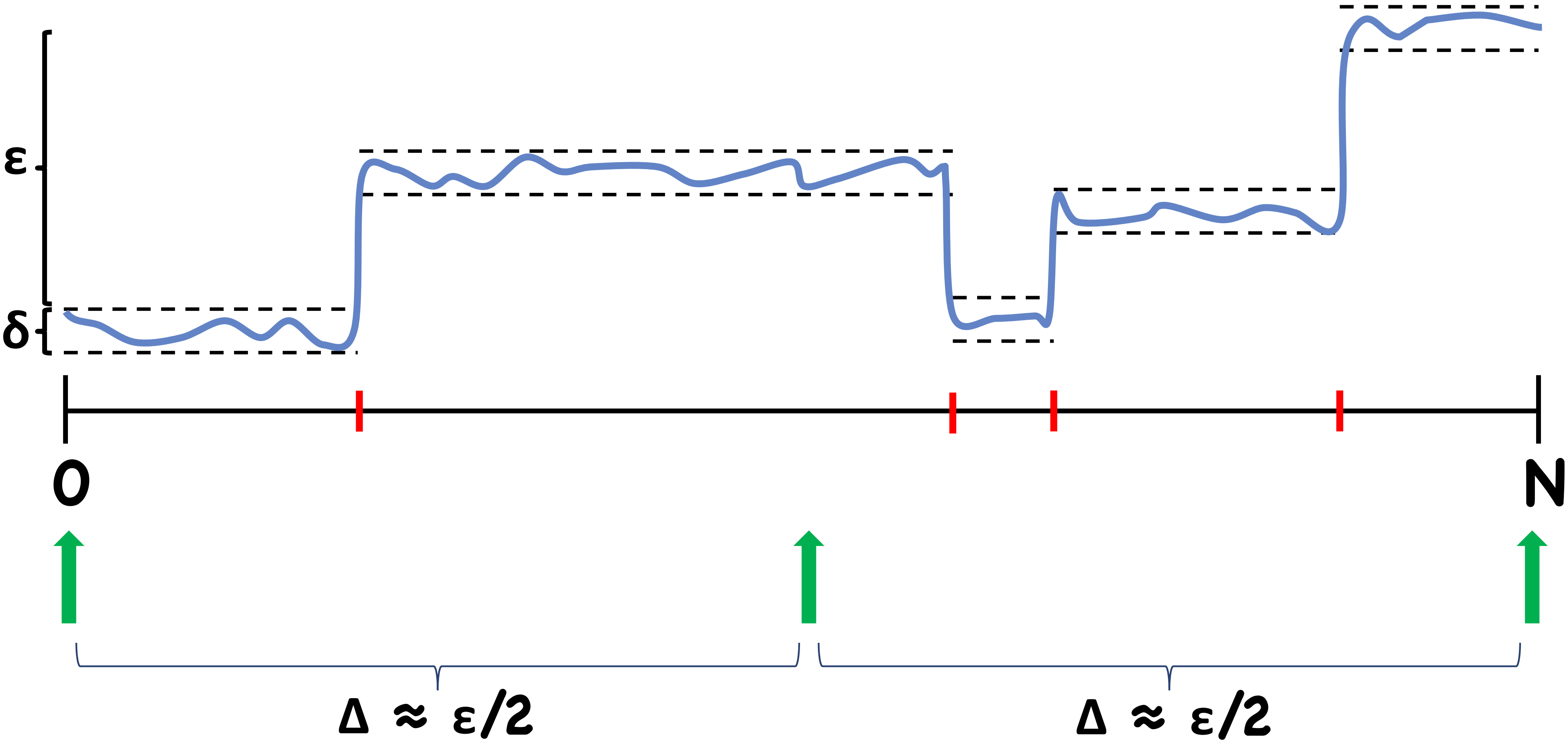
For efficient tracing of large **N**, need running time **polylog(N)**

- Can’t visit every point in domain

Binary Search?



Binary Search?



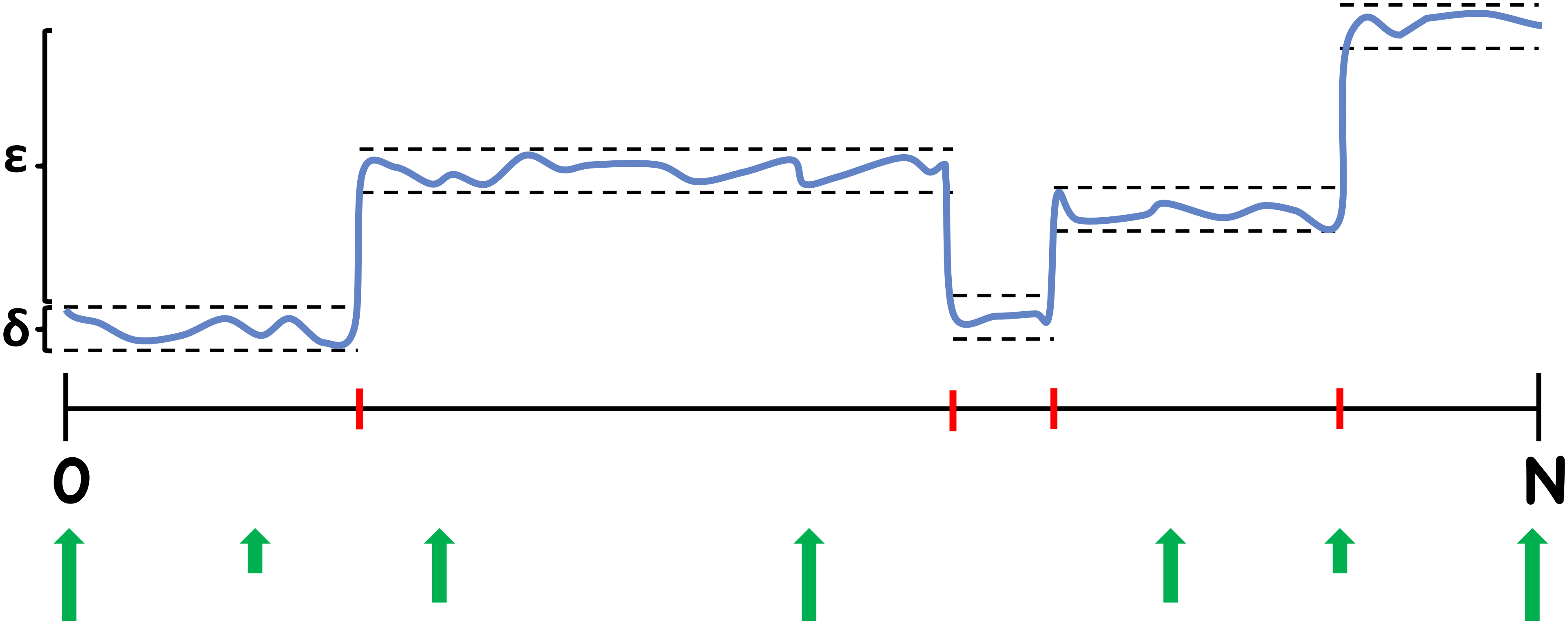
Which side do I recurse on?

- Larger gap? Gap decreases by ½ each time

  Gap doesn't tell us how many jumps
- Both? Still polynomial time in **log(N)**?

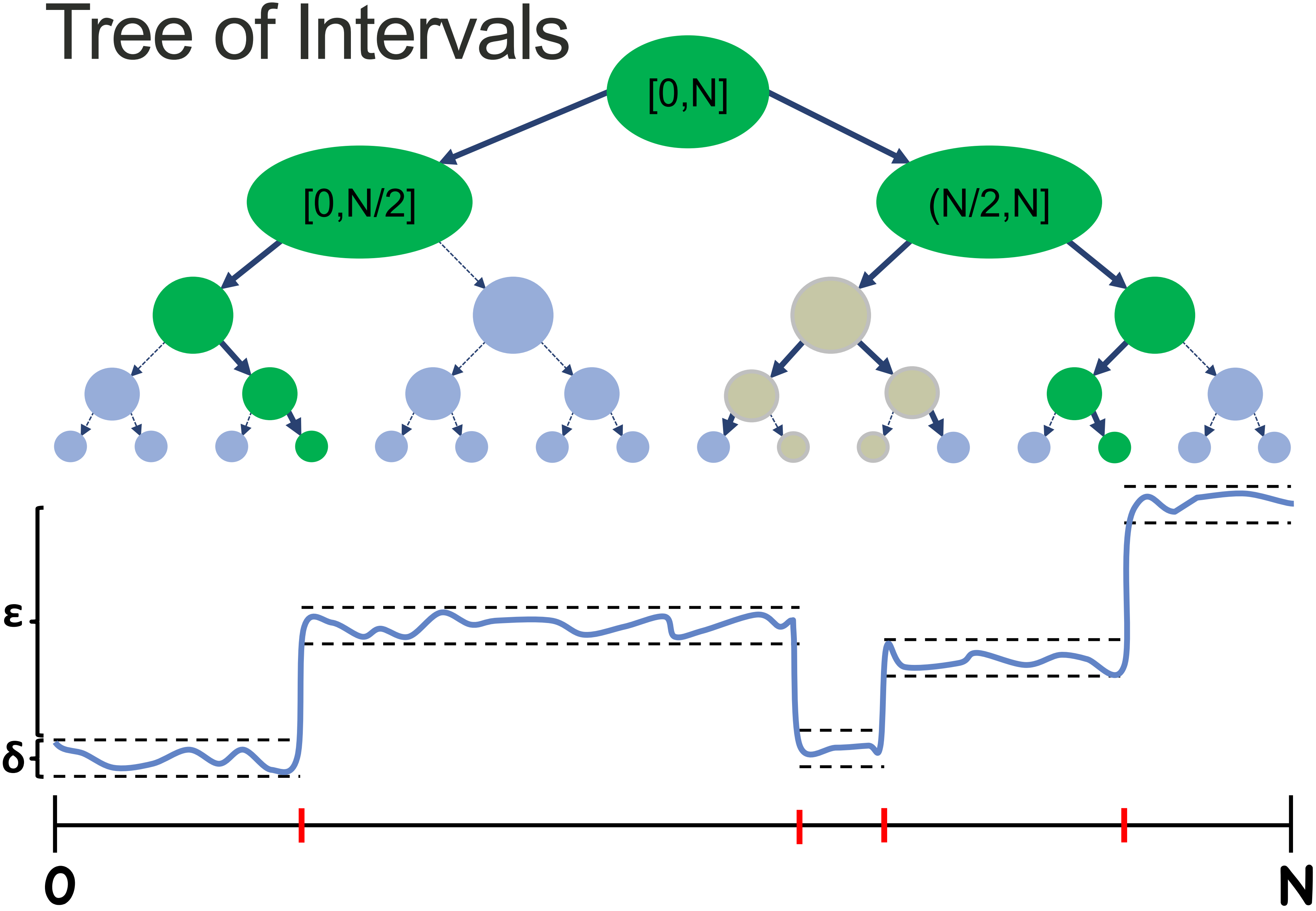
Always recurse on gap

Alg from [BCP'14], entirely different context



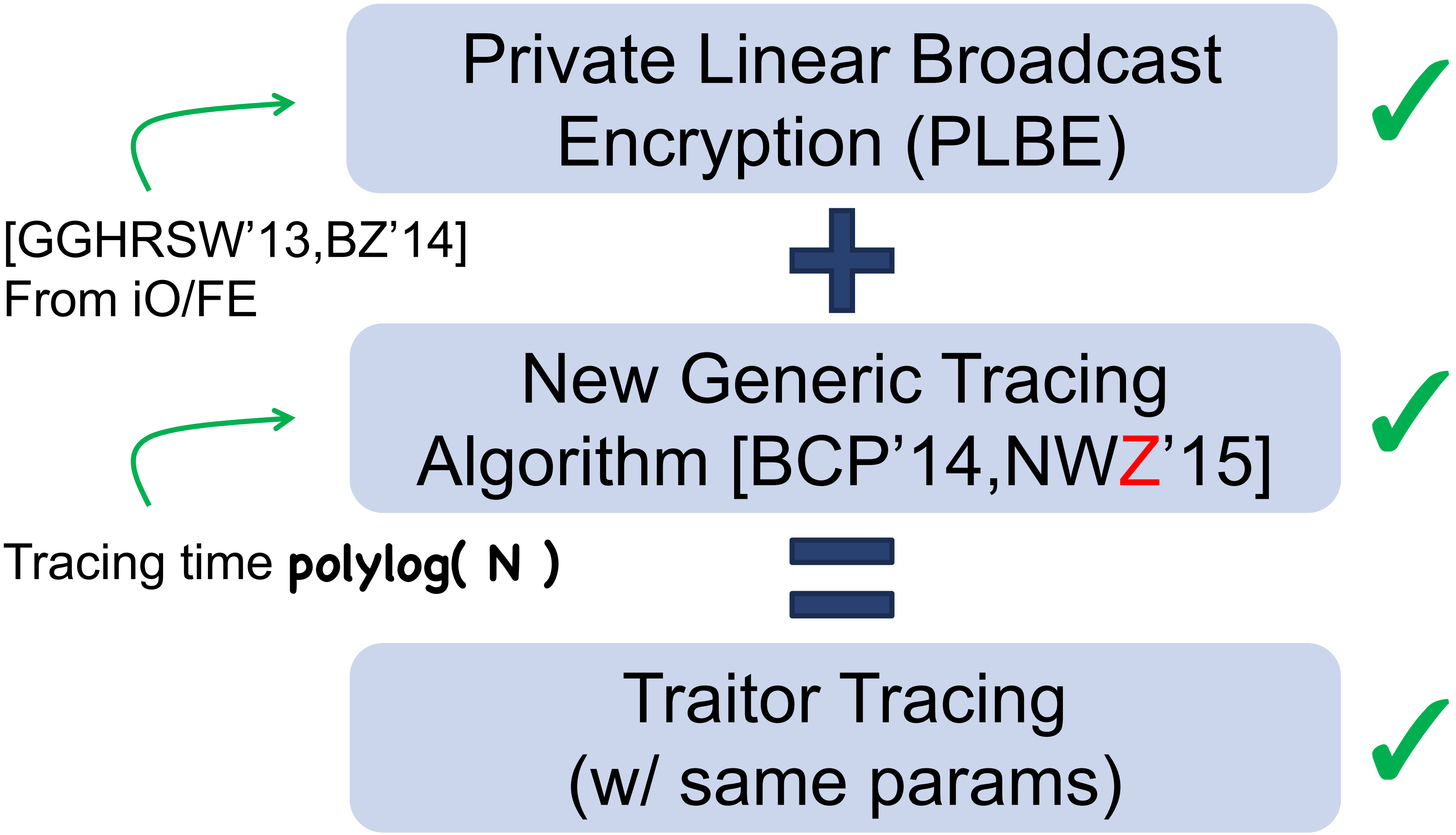
Question: why guaranteed to be polynomial time?

Tree of Intervals



Recurse on interval → jump in that interval

Large-Identity Traitor Tracing from PLBE



Private Linear Broadcast Encryption (PLBE) ✓

+

New Generic Tracing Algorithm [BCP'14,NWZ'15] ✓

=

Traitor Tracing (w/ same params) ✓

[GGHRSW'13,BZ'14] From iO/FE

Tracing time **polylog(N)**

Limitations of PLBE Approach

Suppose I want to embed much more info into key

- User ID = Name + Address + Map + Picture/Video + …

Given **msk**, can recover **v** from **(📁 , v🔒)**

- Find **v'** s.t. **v'**🔑 decrypts ctxt, **v'+1**🔑 but does not

Given **pk**, can recover **u** from **u**🔑

- Find **u'** s.t. **u**🔑 decrypts **(📁 , u'🔒)**, but not **(📁 , u'−1🔒)**

PLBE: **|ctxt|, |u🔑| ≥ log N = | identifying info |**

Q: Is this inherent to Traitor Tracing?

Limitations of Traitor Tracing

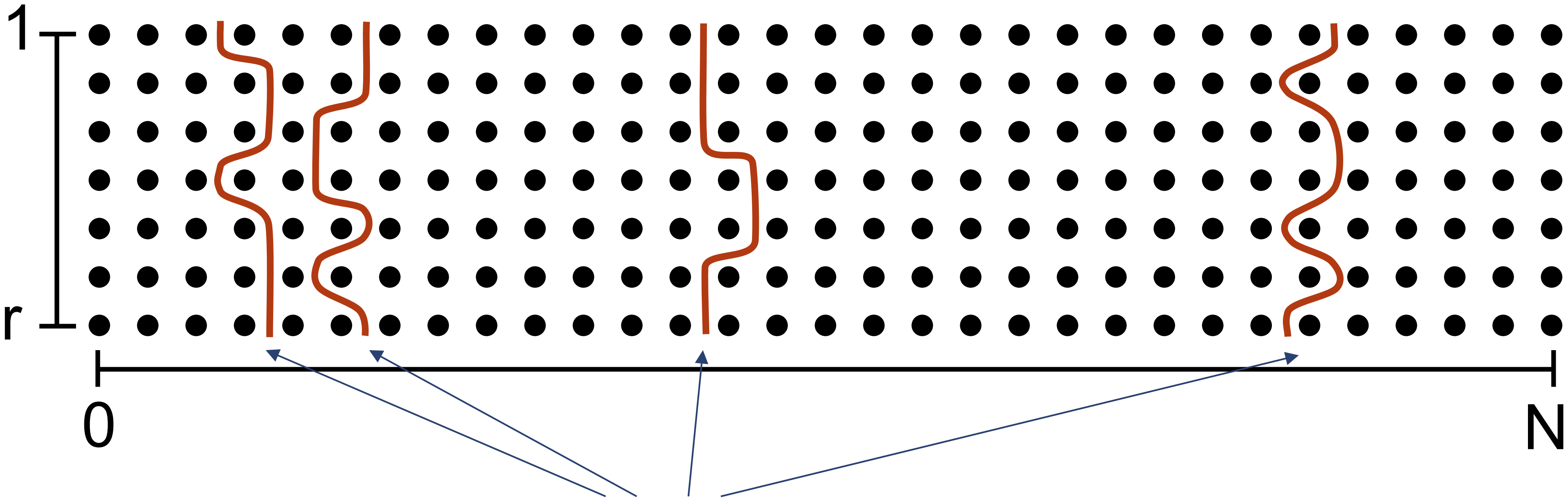
Given **pk**, u🔑, recover **u**: trace 🏴‍☠️ = **Dec(**u🔑**, ·)**

TT: **|**u🔑**| ≥ | identifying info |**

For ctxt size, apparently no such restriction

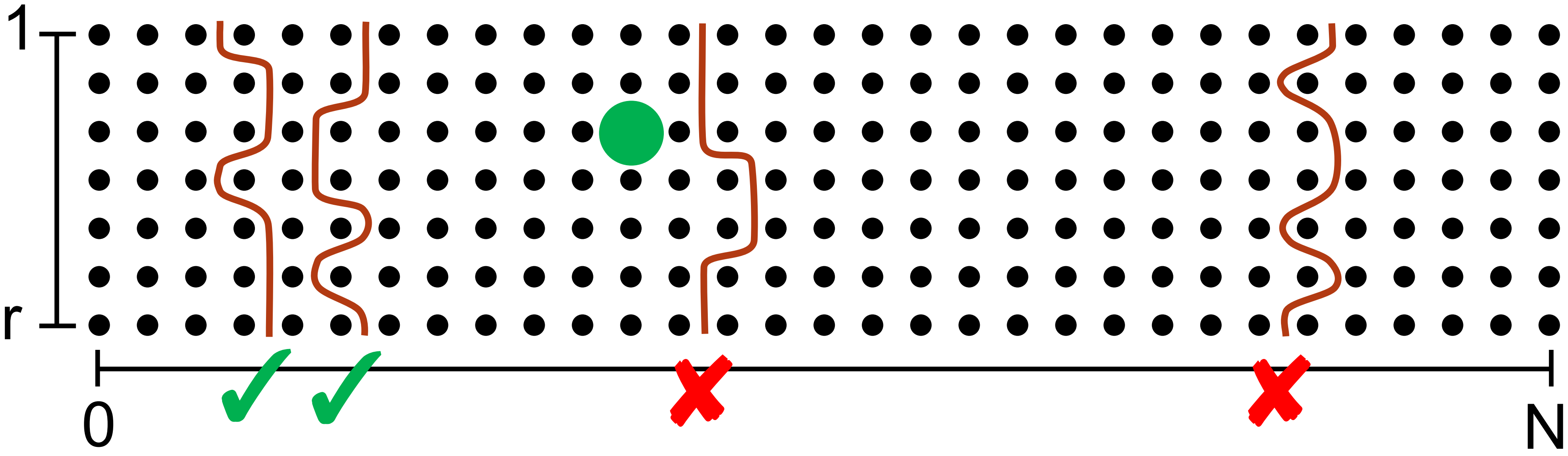
To get small ciphertexts, need alternative to PLBE

Private Block Linear Broadcast



Identifying info encoded as curve

Private Block Linear Broadcast



Ctxt: (📁, ●)🔒

Functionality: can decrypt if point “to the right” of curve

Security:

- Can’t decrypt if point “to the left” of curve
- Can’t learn anything about ● except “left” or “right”

Private Block Linear Broadcast

Theorem: Can trace as long as

- Curves do not intersect
- Curves confined to oscillate about a single column

Size of info encoded by curve: **≥r**

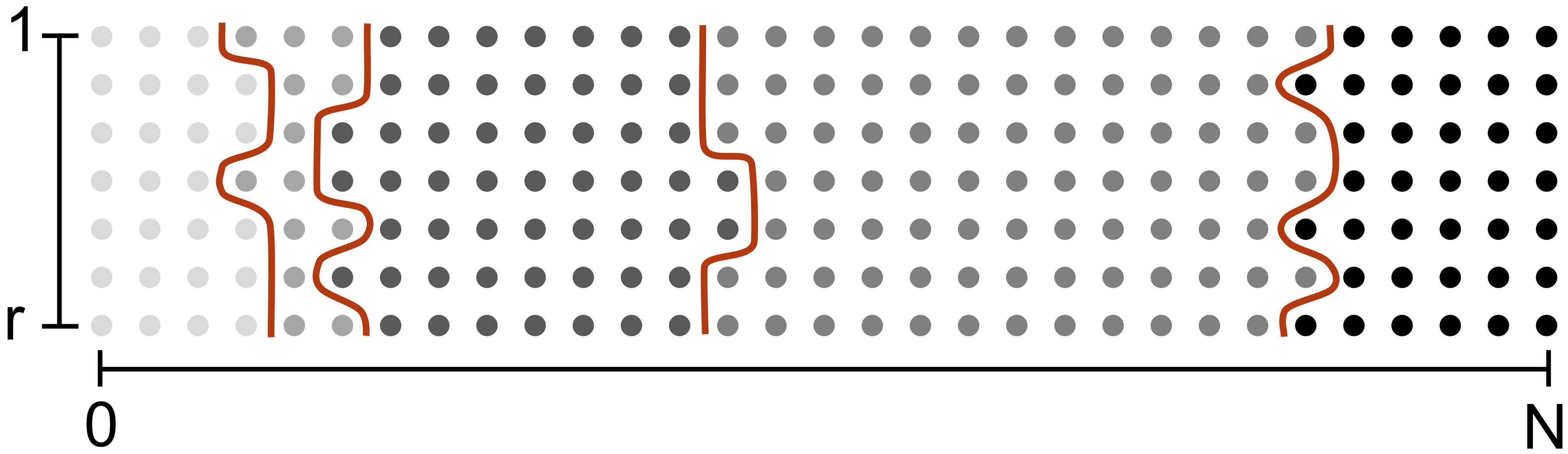
Info encoded in ctxt: 📁+🟢

- **| message | + log r + log N**

Ctxts only need to grow logarithmically with embedded info

- Can achieve from obfuscation using [AS’15]

Tracing PBLBE



● ● ● ● ● represents **Pr[** ☠ decrypts **(** 📁 **,** 🔒 **)]**

Small variation **δ** between curves

Large variation **ε** across domain

→ Large jump at some curve

→ Gives rise to generalization of Jump Finding Problem

Conclusion

First traitor tracing system to handle exponential number of user identities

- Allows for “identity based” traitor tracing
- Allows for anonymity + tracing to coexist
- Can embed arbitrarily large info into key w/o affecting ctxt size
- Also show how to revoke

Main open question:

TT from weaker assumptions (MMaps, lattices, etc)

Thanks!