Conférence de lancement de l'ANR Ciao, Février 2020, Bordeaux, France

VERIFIABLE DELAY FUNCTIONS

cnrs Inria université de BORDEAUX

Benjamin Wesolowski

VERIFIABLE DELAY FUNCTIONS

How to slow things down

VERIFIABLE DELAY FUNCTIONS

[Boneh, Bonneau, Bünz, Fisch 2018] A VDF is a function that

- Requires **time** to evaluate (sequential evaluation, and parallelism does not allow to go faster)
- The output can easily be verified

Syntactically:

➡ **setup**$(T) \rightarrow$ public parameters pp

➡ **eval**$(pp, x) \rightarrow$ output y, proof π (takes time T)

➡ **verify**$(pp, x, y, \pi) \rightarrow$ {true, false}

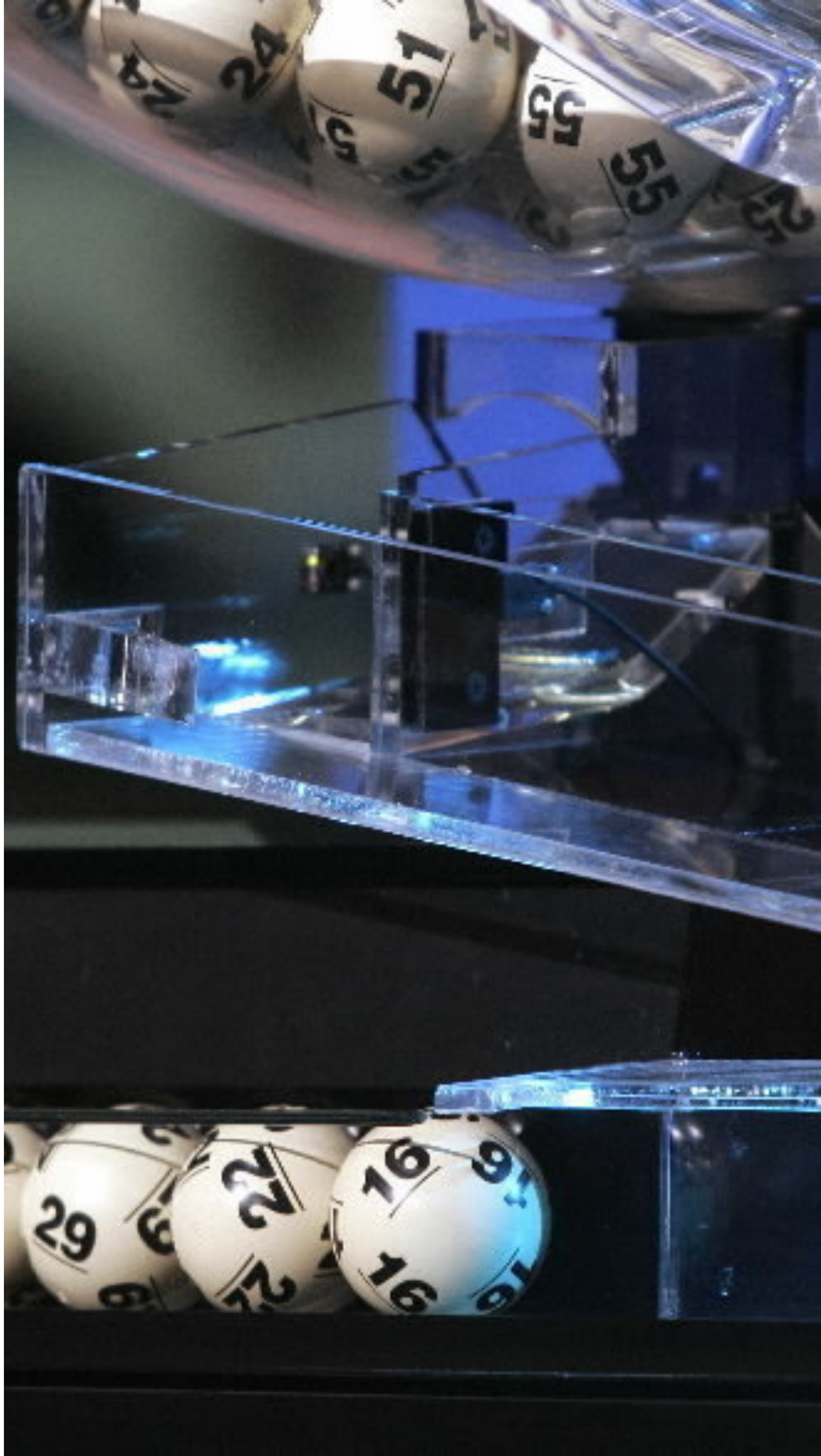
REQUIREMENTS

We need the following properties:

- **Sequentiality:** if A is a parallel algorithm such that time(A, x) $< T$, then A cannot distinguish **eval**(pp, x) from random
- **Uniqueness:** if **verify**(pp, x, y, π) $=$ **verify**(pp, x, y', π') $=$ true, then $y = y'$

PUBLIC RANDOMNESS

A motivation



AD HOC "METHODS"

A CRYPTOGRAPHIC ATTEMPT

A group G of people want to generate some randomness:

- Each person $A \in G$ generates privately a random bit-string r_A
- They all broadcast a commitment $c(r_A)$ (hiding, binding)
- Once all the commitments are distributed, everyone opens
- Random value is $r = \bigoplus_{A \in G} r_A$

'Commit-then-reveal' protocol

A CRYPTOGRAPHIC ATTEMPT

- Two rounds
- Does not scale!
- If someone does not open the commitment, need to restart

'Commit-then-reveal' protocol

SLOTH AND UNICORN

Solution proposed in [Lenstra, W. 2017]:

- Instead of commitments, each party A directly reveals r_A

No commitment, so no 'opening' phase

Trouble: last person to reveal has full control of $r = \bigoplus_{A \in G} r_A$...

- Instead, let $r = f(r_{A_1} \,||\, r_{A_2} \,||\, \ldots \,||\, r_{A_n})$, where f takes **time** to evaluate (in [Lenstra, W. 2017] the *Sloth* function)

If f takes 10 minutes, nobody knows r
until 10 minutes after the last reveal:
impossible to manipulate r!

VERIFIABLE DELAY FUNCTION

We want

- $f(x)$ **slow** to evaluate, even for parties with a lot of parallel power or specialised hardware
- $f(x) = y$ easy to **verify** by anyone

Use a **verifiable delay function**

A VERIFIABLE DELAY FUNCTION

Slow yet efficient

ITERATED HASHING

What is slow to compute, and cannot be sped up by parallelism? Maybe iterated hashing...

$$x \longrightarrow H(x) \longrightarrow H(H(x)) \longrightarrow \ldots \longrightarrow H(\ldots H(H(x))\ldots) = y$$

- Slow, sequential computation... but how to check $f(x) = y$?
- No simple and efficient way...

TIME LOCK PUZZLE

Drawing inspiration from time-lock puzzles [Rivest, Shamir, Wagner 1996]

- Let G be a group of unknown order
- Given $x \in G$, computing x^{2^T} requires T sequential squarings

$$x \longrightarrow x^2 \longrightarrow x^{2^2} \longrightarrow x^{2^3} \longrightarrow \ldots \longrightarrow x^{2^T}$$

- The VDF could be $f(x) = x^{2^T}$, but can this be verified?

Approach of [W. 2019], also taken in [Pietrzak 2019]

PROOF OF CORRECT EXPONENTIATION

- Given $(x, y) \in G$, Alice wants to prove that $y = x^{2^T}$
  - ➡ Together with $y = x^{2^T}$, Alice computes a 'proof' π
  - ➡ Given (x, y, π), anyone can efficiently verify that $y = x^{2^T}$
- We present the method as an interactive protocol: Alice wants to prove **to Bob (the verifier)** that $y = x^{2^T}$
- The protocols is then be made non-interactive (Fiat-Shamir...)

INTERACTIVE ARGUMENT

- Given $(x, y) \in G$, Alice wants to prove to Bob that $y = x^{2^T}$

Alice		Bob
	$\xleftarrow{\ell}$	Choose a random (large) prime ℓ
Find q and r such that $2^T = q\ell + r,\ 0 \leq r < \ell$	$\xrightarrow{\pi = x^q}$	
		Compute $r = 2^T \bmod \ell$
		Accept if $\pi^\ell x^r = y$

NON-INTERACTIVE VDF

The VDF on input $x \in G$ is the following:

- ➡ Compute $y = x^{2^T}$ (slow, sequential part)
- ➡ Let ℓ = hash_to_prime(x, y, T)
- ➡ Find q and r such that $2^T = q\ell + r$, and $0 \leq r < \ell$
- ➡ Compute $\pi = x^q$ How long does the computation of π take?
- ➡ Output: (y, π), only 2 group elements

- **verify**(pp, x, y, π): $\pi^\ell x^r = y$, only 2 small exponentiations

PROPERTIES

	Size of proof	Evaluation	Verifier
	number of group elements	number of group operations	
Sloth [Lenstra, W. 2017]	1	T	$O(T)$
[Pietrzak 2019]	$\log(T)$	$T(1 + 2/T^{1/2})$	$O(\log(T))$
This work [W. 2019]	1	$T(1 + 2/\log(T))$	$O(1)$

SECURITY

- Given $(x, y) \in G$, Alice wants to prove to Bob that $y = x^{2^T}$

Alice

Bob

Bob: Choose a random (large) prime ℓ

Bob → Alice: ℓ

Alice: Find q and r such that $2^T = q\ell + r,\ 0 \leq r < \ell$

Alice → Bob: $\pi = x^q$

Bob: Compute $r = 2^T \bmod \ell$

Accept if $\pi^\ell x^r = y$

SECURITY

- Suppose $y \neq x^{2^T}$ (i.e., Alice is dishonest)
- Let $w = y/x^{2^T} \neq 1_G$
- **Claim:** for Alice to convince Bob, she must be able to extract ℓ-th roots of w with good probability (unpredictable ℓ)
- **Proof:** when Bob generates a random ℓ, Alice computes π such that $\pi^\ell x^r = y$ (acceptance condition), where $2^T = q\ell + r$. Let $\rho = \pi/x^q$. Then,

$$\rho^\ell = \pi^\ell/x^{q\ell} = (y/x^r)/x^{q\ell} = y/x^{q\ell + r} = w$$

i.e., ρ is an ℓ-th root of w

ADAPTIVE ROOT ASSUMPTION

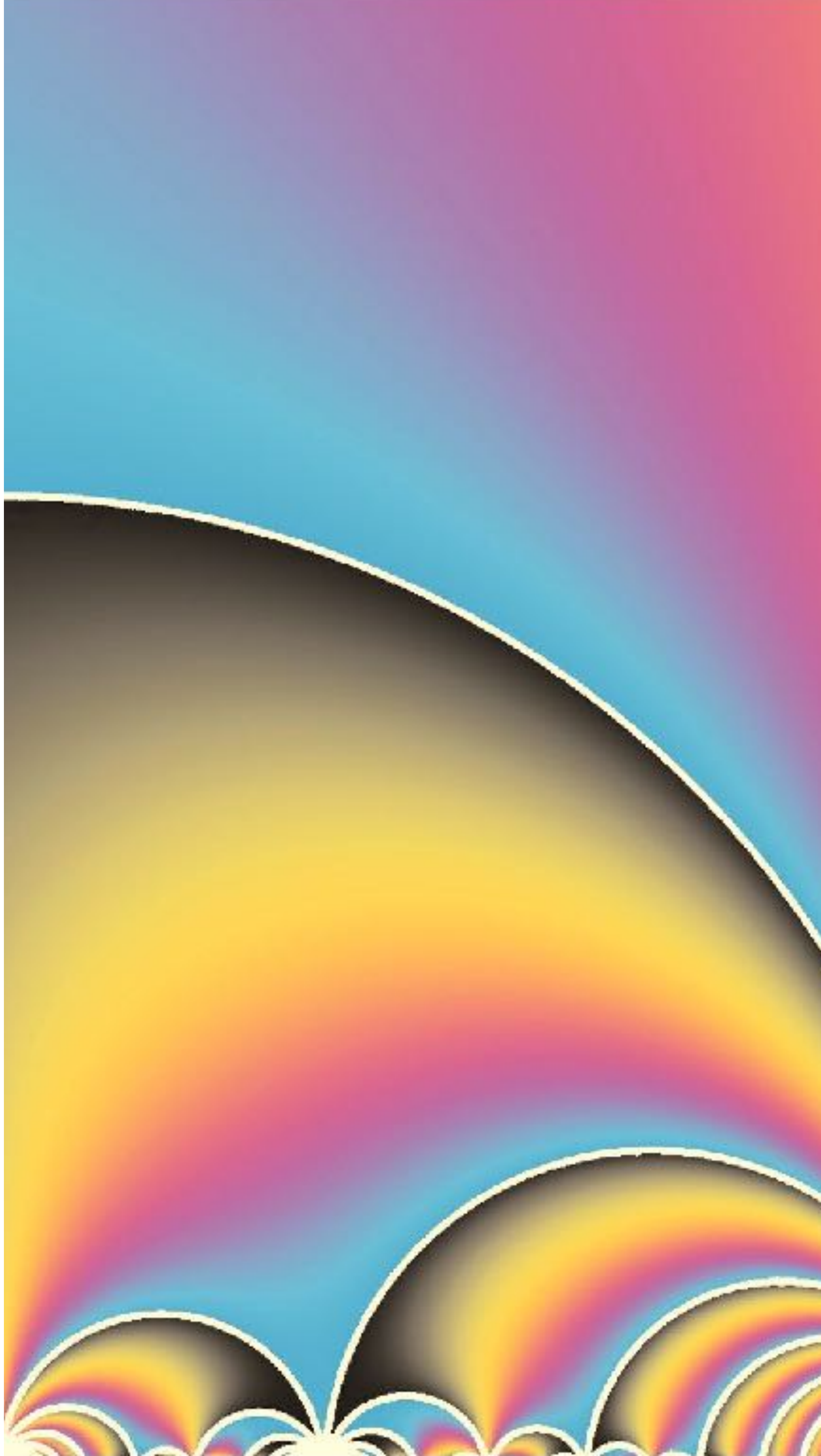
We assume the following game is hard in the group G:

- The player outputs an element $w \in G$, other than the neutral element 1_G
- The challenger generates a random (large) prime ℓ
- The player has to find an ℓ-th root of w (i.e., $w^{1/\ell}$)

In which groups does this assumption hold?

GROUPS OF UNKNOWN ORDER

From number theory



THE PROBLEM WITH KNOWN ORDER

- Suppose $w \in G$ has known order n
- The challenger generates a random (large) prime ℓ
- Computing $k = \ell^{-1} \bmod n$ is easy (invertible with overwhelming probability)
- w^k is an ℓ-th root of w

RSA GROUPS

Let $N = pq$ an RSA modulus

- Without the factorisation of N, order of $(\mathbb{Z}/N\mathbb{Z})^\times$ is unknown
- We still know the small subgroup $\{\pm 1\}$... trouble
- Use $G = (\mathbb{Z}/N\mathbb{Z})^\times/\{\pm 1\}$
- **Problem:** need to generate N so that nobody knows the factorisation (trusted setup? large random N? MPC?)

RSA MPC

Goal of the Ethereum Foundation and Protocol labs, working with Ligero:

- A 2048 bits modulus N, secret factorisation
- Result of an $(n-1)$-maliciously secure MPC
- 1024 participants

CLASS GROUPS

Let p be a random large prime, K the imaginary quadratic field of discriminant $-p$, and G its class group

- Computing the order of G is hard (complexity $L_p(1/2)$)
- Easy setup! Can even change p at every new evaluation... becomes 'quantum resistant'
- Careful: the 2-torsion is easy to compute

ADAPTIVE ROOT ASSUMPTION

- Open question:« adaptive root assumption » is not known to be equivalent to finding an element of known order
- It is hard in the generic group model [Boneh, Bünz, Fisch 2018]
- Is it as hard as it looks in RSA groups and class groups? At least, root extraction (non-adaptive) is believed to be hard

SLOWNESS IN THE REAL WORLD

Practical considerations

TIME LOCK ASSUMPTION

Assumption: computing

$$x \longrightarrow x^2 \longrightarrow x^{2^2} \longrightarrow x^{2^3} \longrightarrow \dots \longrightarrow x^{2^T}$$

takes time $\approx T \times$ (latency of one squaring in the group)

- What is that latency?
- Can a rich adversary get a much better latency than easily available hardware?

Solution: massively invest in building the fastest hardware, and make it widely available

$100,000 COMPETITION

Chia Network organises a VDF competition (second round finished Jul 18 with $100,000 in total prize money)

- Fastest possible implementation of **class group arithmetic**
- https://www.chia.net

$1,000,000 COMPETITION

Funded 50/50 by the Ethereum Foundation and Protocol Labs

- Fastest possible implementation of **modular arithmetic**, modulo a 2048-bit RSA modulus
- Latency of 1ns per squaring?
- https://vdfresearch.org

LOWER BOUNDS?

Let

$$\text{MODSQ-MOD2}_{b,N} : \{0, 1\}^b \longrightarrow \{0, 1\}$$

the function that sends x to the least significant bit of (x^2 mod N)

Theorem [W., Williams 2020]: For all odd $0 \leq N \leq 2^b - 1$, every fan-in two circuit of depth less than $\log_2(b - O(1))$ fails to compute $\text{MODSQ-MOD2}_{b,N}$ on at least 24% of all b-bit inputs

In simpler words: A circuit that performs « squaring modulo N » in binary representation reliably has depth at least $\approx \log_2(b)$

Conférence de lancement de l'ANR Ciao, Février 2020, Bordeaux, France

VERIFIABLE DELAY FUNCTIONS

cnrs
Inria
université de BORDEAUX

Benjamin Wesolowski