VDFs and novel RSA assumptions

A Note on Low Order Assumptions in RSA groups

István András Seres and Péter Burcsi

Eötvös Loránd University

2020 April 24

Table of Contents

Table of Contents

Low Order assumption à la Boneh et al. [BBF18]

Definition

The *Low Order assumption [BBF18]*. For any probabilistic polynomial time adversary $\mathcal{A}$ finding any element of low order is hard:

$$Pr\left[u^l = 1,\ u \notin \{1, -1\} : \begin{array}{c} \mathbb{G} \xleftarrow{\$} GGen(\lambda) \\ (u, l) \leftarrow \mathcal{A}(\mathbb{G}) \\ \text{and } l < 2^{poly(\lambda)} \end{array} \right] \leq \mathrm{negl}(\lambda)$$

Low Order assumption à la Boneh et al. [BBF18]

Definition

The *Low Order assumption [BBF18]*. For any probabilistic polynomial time adversary $\mathcal{A}$ finding any element of low order is hard:

$$Pr\left[u^l = 1,\ u \notin \{1,-1\} : \begin{array}{c} \mathbb{G} \xleftarrow{\$} GGen(\lambda) \\ (u,l) \leftarrow \mathcal{A}(\mathbb{G}) \\ \text{and } l < 2^{poly(\lambda)} \end{array}\right] \leq \mathrm{negl}(\lambda)$$

- Seems like quite a strong assumption since $l < 2^{poly(\lambda)}$ and $p, q, N, \phi(N) \in \mathcal{O}(2^{poly(\lambda)})$

Low Order assumption à la Boneh et al. [BBF18]

Definition

The *Low Order assumption [BBF18]*. For any probabilistic polynomial time adversary $\mathcal{A}$ finding any element of low order is hard:

$$Pr\left[u^l = 1,\ u \notin \{1, -1\} : \begin{array}{c} \mathbb{G} \xleftarrow{\$} GGen(\lambda) \\ (u, l) \leftarrow \mathcal{A}(\mathbb{G}) \\ \text{and } l < 2^{poly(\lambda)} \end{array}\right] \leq \mathrm{negl}(\lambda)$$

- Seems like quite a strong assumption since $l < 2^{poly(\lambda)}$ and $p, q, N, \phi(N) \in \mathcal{O}(2^{poly(\lambda)})$
- But also weak somewhat as it is not obvious how such an adversary would help in factoring the modulus if $l \neq 2$.

Low Order assumption à la Boneh et al. [BBF18]

Definition

The *Low Order assumption [BBF18]*. For any probabilistic polynomial time adversary $\mathcal{A}$ finding any element of low order is hard:

$$Pr\left[u^l = 1,\ u \notin \{1,-1\} : \begin{array}{c} \mathbb{G} \xleftarrow{\$} GGen(\lambda) \\ (u,l) \leftarrow \mathcal{A}(\mathbb{G}) \\ \text{and } l < 2^{poly(\lambda)} \end{array} \right] \leq \mathrm{negl}(\lambda)$$

- Seems like quite a strong assumption since $l < 2^{poly(\lambda)}$ and $p, q, N, \phi(N) \in \mathcal{O}(2^{poly(\lambda)})$
- But also weak somewhat as it is not obvious how such an adversary would help in factoring the modulus if $l \neq 2$.
- So, how does this relate to other RSA assumptions?

Adaptive Root assumption [Wes19]

Definition

The *Adaptive Root Assumption* holds for *GGen* if there is no efficient adversary $(\mathcal{A}_0, \mathcal{A}_1)$ that succeeds in the following task. First, $\mathcal{A}_0$ outputs an element $w \in \mathbb{G}$ and some state st. Then, a random prime in $\mathrm{Primes}(\lambda)$ is chosen and $\mathcal{A}_1(w, l, st)$ outputs $w^{1/l} \in \mathbb{G}$. For all efficient $(\mathcal{A}_0, \mathcal{A}_1)$:

$$Pr\left[u^l = w \neq 1 : \begin{array}{c} \mathbb{G} \xleftarrow{\$} GGen(\lambda) \\ (w, st) \leftarrow \mathcal{A}_0(\mathbb{G}) \\ l \xleftarrow{\$} \Pi_\lambda = \mathrm{Primes}(\lambda) \\ u \leftarrow \mathcal{A}_1(w, l, st) \end{array}\right] \leq \mathrm{negl}(\lambda)$$

Adaptive Root assumption [Wes19]

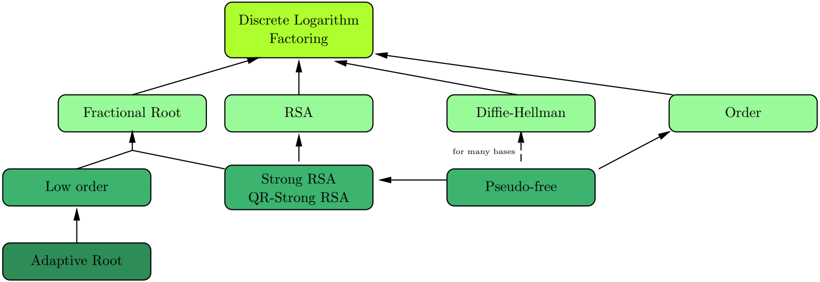
Definition

The *Adaptive Root Assumption* holds for *GGen* if there is no efficient adversary $(\mathcal{A}_0, \mathcal{A}_1)$ that succeeds in the following task. First, $\mathcal{A}_0$ outputs an element $w \in \mathbb{G}$ and some state st. Then, a random prime in $\mathrm{Primes}(\lambda)$ is chosen and $\mathcal{A}_1(w, l, st)$ outputs $w^{1/l} \in \mathbb{G}$. For all efficient $(\mathcal{A}_0, \mathcal{A}_1)$:

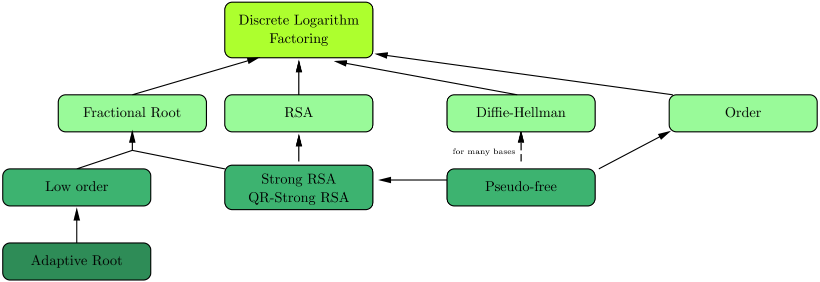
$$Pr\left[u^l = w \neq 1 : \begin{array}{c} \mathbb{G} \xleftarrow{\$} GGen(\lambda) \\ (w, st) \leftarrow \mathcal{A}_0(\mathbb{G}) \\ l \xleftarrow{\$} \Pi_\lambda = \mathrm{Primes}(\lambda) \\ u \leftarrow \mathcal{A}_1(w, l, st) \end{array} \right] \leq \mathrm{negl}(\lambda)$$

The number of primes in Π_λ should be exponential in λ: it is possible to precompute w using $2^{|\Pi_\lambda|}$ exponentiations.

The RSA Assumption Zoo: a glimpse

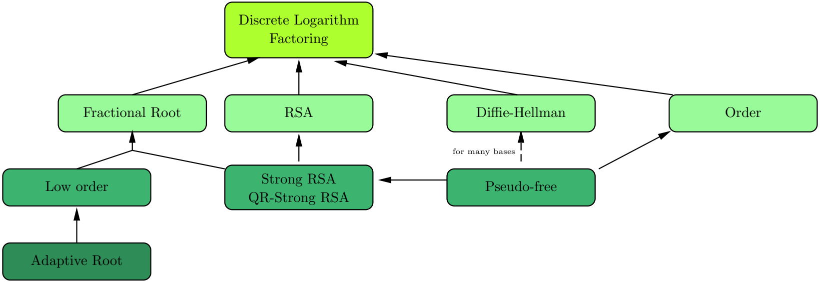


The RSA Assumption Zoo: a glimpse



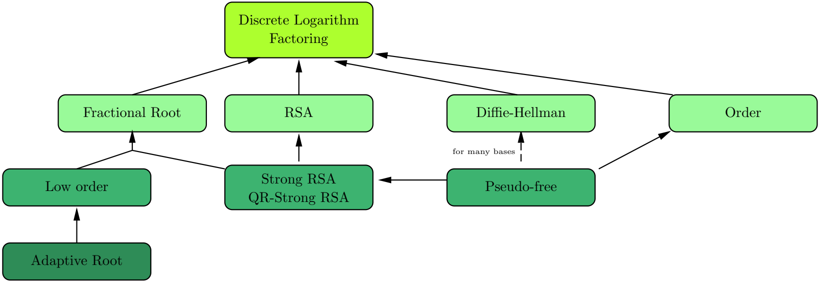
- RSA assumption and Factoring?

The RSA Assumption Zoo: a glimpse



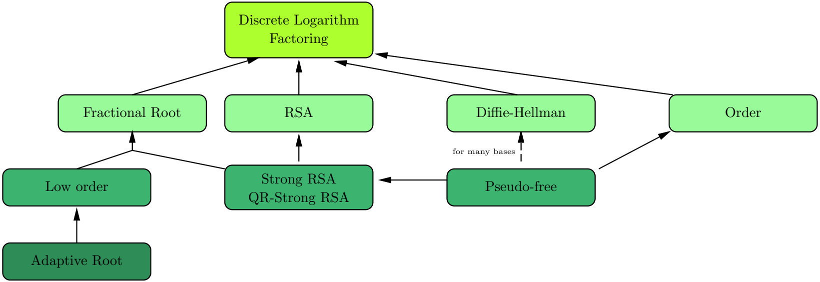
- RSA assumption and Factoring?
- Strong RSA: there are exponential witnesses! [GK16]

The RSA Assumption Zoo: a glimpse



- RSA assumption and Factoring?
- Strong RSA: there are exponential witnesses! [GK16]
- Low Order: is it equivalent to Factoring?

The RSA Assumption Zoo: a glimpse



- RSA assumption and Factoring?
- Strong RSA: there are exponential witnesses! [GK16]
- Low Order: is it equivalent to Factoring?
- Adaptive Root: seems like a hard problem but seems like a weak assumption: Interactive and Exponential witnesses!

Remarks on the AR assumption

- Exponentially stronger than the RSA assumption.

Remarks on the AR assumption

- Exponentially stronger than the RSA assumption.
- AR is a t-search problem with exponentially many witnesses.

Remarks on the AR assumption

- Exponentially stronger than the RSA assumption.
- AR is a t-search problem with exponentially many witnesses.
- Interactive assumption

Remarks on the AR assumption

- Exponentially stronger than the RSA assumption.
- AR is a t-search problem with exponentially many witnesses.
- Interactive assumption
- Seems like a hard problem but still we know very little about its complexity! Pls help! :)

Remarks on the AR assumption

- Exponentially stronger than the RSA assumption.
- AR is a t-search problem with exponentially many witnesses.
- Interactive assumption
- Seems like a hard problem but still we know very little about its complexity! Pls help! :)
- Maybe we'll have a similar result like the one by Oded Regev [Reg09], namely a reduction from LWE to SIS. Reduction from a 1-search problem to an exponential t-search problem!

Remarks on the AR assumption

- Exponentially stronger than the RSA assumption.
- AR is a t-search problem with exponentially many witnesses.
- Interactive assumption
- Seems like a hard problem but still we know very little about its complexity! PIs help! :)
- Maybe we'll have a similar result like the one by Oded Regev [Reg09], namely a reduction from LWE to SIS. Reduction from a 1-search problem to an exponential t-search problem!
- Highly recommended reading the position paper on cryptographic assumptions by Goldwasser and Kalai [GK16]!

Table of Contents

Defining VDFs (Verifiable Delay Functions)

A VDF $V = (\mathsf{Setup}, \mathsf{Eval}, \mathsf{Verify})$ is a triple of algorithms:

- $\mathsf{Setup}(\lambda, t) \to \mathbf{pp} = (\mathsf{ek}, \mathsf{vk})$

Defining VDFs (Verifiable Delay Functions)

A VDF $V = (\mathsf{Setup}, \mathsf{Eval}, \mathsf{Verify})$ is a triple of algorithms:

- $\mathsf{Setup}(\lambda, t) \to \mathbf{pp} = (\mathsf{ek}, \mathsf{vk})$
- $\mathsf{Eval}(\mathsf{ek}, x) \to (y, \pi)$

Defining VDFs (Verifiable Delay Functions)

A VDF $V = (\mathsf{Setup}, \mathsf{Eval}, \mathsf{Verify})$ is a triple of algorithms:

- $\mathsf{Setup}(\lambda, t) \to \mathbf{pp} = (\mathsf{ek}, \mathsf{vk})$
- $\mathsf{Eval}(\mathsf{ek}, x) \to (y, \pi)$
- $\mathsf{Verify}(\mathsf{vk}, x, y, \pi) \to \{Yes, No\}$

Defining VDFs (Verifiable Delay Functions)

A VDF $V = (\mathsf{Setup}, \mathsf{Eval}, \mathsf{Verify})$ is a triple of algorithms:

- $\mathsf{Setup}(\lambda, t) \to \mathbf{pp} = (\mathsf{ek}, \mathsf{vk})$
- $\mathsf{Eval}(\mathsf{ek}, x) \to (y, \pi)$
- $\mathsf{Verify}(\mathsf{vk}, x, y, \pi) \to \{Yes, No\}$

Security requirements

- Correctness and SOUNDNESS

Defining VDFs (Verifiable Delay Functions)

A VDF $V = (\mathsf{Setup}, \mathsf{Eval}, \mathsf{Verify})$ is a triple of algorithms:

- $\mathsf{Setup}(\lambda, t) \to \mathbf{pp} = (\mathsf{ek}, \mathsf{vk})$
- $\mathsf{Eval}(\mathsf{ek}, x) \to (y, \pi)$
- $\mathsf{Verify}(\mathsf{vk}, x, y, \pi) \to \{Yes, No\}$

Security requirements

- Correctness and SOUNDNESS
- Eval cannot be paralellized! Class groups???

Defining VDFs (Verifiable Delay Functions)

A VDF $V = (\mathsf{Setup}, \mathsf{Eval}, \mathsf{Verify})$ is a triple of algorithms:

- $\mathsf{Setup}(\lambda, t) \to \mathbf{pp} = (\mathsf{ek}, \mathsf{vk})$
- $\mathsf{Eval}(\mathsf{ek}, x) \to (y, \pi)$
- $\mathsf{Verify}(\mathsf{vk}, x, y, \pi) \to \{Yes, No\}$

Security requirements

- Correctness and SOUNDNESS
- Eval cannot be paralellized! Class groups???
- There should be an exponential gap between the time complexity of Eval and Verify

Example

Iterating a cryptographic hash function is a VDF? How to make it a VDF?

Why people are excited about VDFs?

- Non-interactive timestamping [LSS19]

Why people are excited about VDFs?

- Non-interactive timestamping [LSS19]
- Randomness beacons [BGB17]

Why people are excited about VDFs?

- Non-interactive timestamping [LSS19]
- Randomness beacons [BGB17]
- Proof of replication [FBGB19]

Why people are excited about VDFs?

- Non-interactive timestamping [LSS19]
- Randomness beacons [BGB17]
- Proof of replication [FBGB19]
- Resource-efficient blockchains [CP19]

Necessity of using a group of unknown order [RSS]

$$\mathcal{L}_{\mathsf{EXP}} = \left\{ \left(\mathbb{G}, g, h, T \right) : h = g^{(2^T)} \in \mathbb{G} \right\}$$

Necessity of using a group of unknown order [RSS]

$$\mathcal{L}_{\mathsf{EXP}} = \left\{ \left(\mathbb{G}, g, h, T \right) : h = g^{(2^T)} \in \mathbb{G} \right\}$$

Suppose we would know $ord(\mathbb{G})$.

Necessity of using a group of unknown order [RSS]

$$\mathcal{L}_{\mathsf{EXP}} = \left\{ \left(\mathbb{G}, g, h, T \right) : h = g^{(2^T)} \in \mathbb{G} \right\}$$

Suppose we would know $ord(\mathbb{G})$.
Then evaluation of the function would be super efficient? Why?

Necessity of using a group of unknown order [RSS]

$$\mathcal{L}_{\mathsf{EXP}} = \left\{ \left(\mathbb{G}, g, h, T\right) : h = g^{(2^T)} \in \mathbb{G} \right\}$$

Suppose we would know $ord(\mathbb{G})$.
Then evaluation of the function would be super efficient? Why?
One could reduce the exponent $2^T \mod ord(\mathbb{G})$.

Necessity of using a group of unknown order [RSS]

$$\mathcal{L}_{\mathsf{EXP}} = \left\{ \left(\mathbb{G}, g, h, T \right) : h = g^{(2^T)} \in \mathbb{G} \right\}$$

Suppose we would know $ord(\mathbb{G})$.
Then evaluation of the function would be super efficient? Why?
One could reduce the exponent $2^T \mod ord(\mathbb{G})$.
Currently known groups of unknown order: RSA and class groups!

An algebraic construction

Let $\mathbb{G}$ be a finite cyclic group with generator $g \in \mathbb{G}$.

$$\mathbb{G} := \{1, g, g^2, g^3, \dots\}$$

An algebraic construction

Let $\mathbb{G}$ be a finite cyclic group with generator $g \in \mathbb{G}$.

$$\mathbb{G} := \{1, g, g^2, g^3, \dots\}$$

Assumption: $\mathbb{G}$ has an unknown order.

$$\mathbf{pp} = (\mathbb{G}, H : X \to \mathbb{G})$$

An algebraic construction

Let $\mathbb{G}$ be a finite cyclic group with generator $g \in \mathbb{G}$.

$$\mathbb{G} := \{1, g, g^2, g^3, \dots\}$$

Assumption: $\mathbb{G}$ has an unknown order.

$$\mathbf{pp} = (\mathbb{G}, H : X \to \mathbb{G})$$

$$\boxed{\mathsf{Eval}(\mathbf{pp}, \mathsf{x}) : \quad \textit{output} \quad y = H(x)^{(2^T)} \in \mathbb{G}}$$

An algebraic construction

Let $\mathbb{G}$ be a finite cyclic group with generator $g \in \mathbb{G}$.

$$\mathbb{G} := \{1, g, g^2, g^3, \dots\}$$

Assumption: $\mathbb{G}$ has an unknown order.

$$\mathbf{pp} = (\mathbb{G}, H : X \to \mathbb{G})$$

$$\boxed{\mathsf{Eval}(\mathbf{pp}, \mathsf{x}) : \quad \textit{output} \quad y = H(x)^{(2^T)} \in \mathbb{G}}$$

Great lecture by Dan Boneh! Watch it here:
https://www.youtube.com/watch?v=dN-1q8c50q0

Pietrzak's protocol, The base case of the recursion

$$\mathcal{L}_{\mathsf{EXP}} = \left\{ \left(\mathbb{G}, g, h, T \right) : h = g^{(2^T)} \in \mathbb{G} \right\}$$

- The verifier checks that $g, h \in \mathbb{G}$ and outputs *reject* if not,

Pietrzak's protocol, The base case of the recursion

$$\mathcal{L}_{\mathsf{EXP}} = \left\{ \left(\mathbb{G}, g, h, T \right) : h = g^{(2^T)} \in \mathbb{G} \right\}$$

- The verifier checks that $g, h \in \mathbb{G}$ and outputs *reject* if not,
- If $T = 1$ the verifier checks that $h = g^2$ in $\mathbb{G}$, outputs *accept* or *reject*, and stops.

Pietrzak's protocol: The recursion step, ie. if $T > 1$ the prover and verifier do:

- The prover computes $v \leftarrow g^{(2^{T/2})} \in \mathbb{G}$ and sends v to the verifier. The verifier checks that $v \in \mathbb{G}$ and outputs *reject* and stops, if not.

Pietrzak's protocol: The recursion step, ie. if $T > 1$ the prover and verifier do:

- The prover computes $v \leftarrow g^{(2^{T/2})} \in \mathbb{G}$ and sends v to the verifier. The verifier checks that $v \in \mathbb{G}$ and outputs *reject* and stops, if not.Next, the prover needs to convince the verifier that $h = v^{(2^{T/2})}$ and $v = g^{(2^{T/2})}$, which proves that $h = g^{(2^{T})}$.

Pietrzak's protocol: The recursion step, ie. if $T > 1$ the prover and verifier do:

- The prover computes $v \leftarrow g^{(2^{T/2})} \in \mathbb{G}$ and sends v to the verifier. The verifier checks that $v \in \mathbb{G}$ and outputs *reject* and stops, if not.Next, the prover needs to convince the verifier that $h = v^{(2^{T/2})}$ and $v = g^{(2^{T/2})}$, which proves that $h = g^{(2^T)}$.They can be verified simultaneously by checking a random linear combination:

$$v^r h = (g^r v)^{(2^{T/2})}, \quad \textit{where} \quad r \xleftarrow{\$} \{1, \ldots, 2^\lambda\}.$$

- The verifier sends the prover a random $r \xleftarrow{\$} \{1, \ldots, 2^\lambda\}$.

Pietrzak's protocol: The recursion step, ie. if $T > 1$ the prover and verifier do:

- The prover computes $v \leftarrow g^{(2^{T/2})} \in \mathbb{G}$ and sends v to the verifier. The verifier checks that $v \in \mathbb{G}$ and outputs *reject* and stops, if not.Next, the prover needs to convince the verifier that $h = v^{(2^{T/2})}$ and $v = g^{(2^{T/2})}$, which proves that $h = g^{(2^T)}$.They can be verified simultaneously by checking a random linear combination:

$$v^r h = (g^r v)^{(2^{T/2})}, \quad \textit{where} \quad r \xleftarrow{\$} \{1, \ldots, 2^\lambda\}.$$

- The verifier sends the prover a random $r \xleftarrow{\$} \{1, \ldots, 2^\lambda\}$.
- Both the prover and verifier compute $g_1 \leftarrow g^r v$ and $h_1 \leftarrow v^r h \in \mathbb{G}$.

Pietrzak's protocol: The recursion step, ie. if $T > 1$ the prover and verifier do:

- The prover computes $v \leftarrow g^{(2^{T/2})} \in \mathbb{G}$ and sends v to the verifier. The verifier checks that $v \in \mathbb{G}$ and outputs *reject* and stops, if not.Next, the prover needs to convince the verifier that $h = v^{(2^{T/2})}$ and $v = g^{(2^{T/2})}$, which proves that $h = g^{(2^T)}$.They can be verified simultaneously by checking a random linear combination:

$$v^r h = (g^r v)^{(2^{T/2})}, \quad \textit{where} \quad r \xleftarrow{\$} \{1, \ldots, 2^\lambda\}.$$

- The verifier sends the prover a random $r \xleftarrow{\$} \{1, \ldots, 2^\lambda\}$.
- Both the prover and verifier compute $g_1 \leftarrow g^r v$ and $h_1 \leftarrow v^r h \in \mathbb{G}$.
- The prover and verifier recursively engage in an interactive proof that $(\mathbb{G}, g_1, h_1, T/2) \in \mathcal{L}_{\mathrm{EXP}}$, namely that $h_1 = g_1^{(2^{T/2})} \in \mathbb{G}$.

Table of Contents

Breaking soundness of Pietrzak's VDF and non-necessity of the exponential LO assumption

- Given (u, l), a low order element u, with order $l < 2^{poly(\lambda)}$ can potentially break the soundness of the argument system.

Breaking soundness of Pietrzak's VDF and non-necessity of the exponential LO assumption

- Given (u, l), a low order element u, with order $l < 2^{poly(\lambda)}$ can potentially break the soundness of the argument system.
- If $(\mathbb{G}, g, h, T) \in \mathcal{L}_{\mathsf{EXP}}$, then $(\mathbb{G}, g, hu, T) \notin \mathcal{L}_{\mathsf{EXP}}$ and will be incorrectly accepted by the verifier with probability $1/l$.

Breaking soundness of Pietrzak's VDF and non-necessity of the exponential LO assumption

- Given (u, l), a low order element u, with order $l < 2^{poly(\lambda)}$ can potentially break the soundness of the argument system.
- If $(\mathbb{G}, g, h, T) \in \mathcal{L}_{\mathsf{EXP}}$, then $(\mathbb{G}, g, hu, T) \notin \mathcal{L}_{\mathsf{EXP}}$ and will be incorrectly accepted by the verifier with probability $1/l$.
- Malicious prover sends $v \leftarrow g^{(2^{T/2})} u \in \mathbb{G}$.

Breaking soundness of Pietrzak's VDF and non-necessity of the exponential LO assumption

- Given (u, l), a low order element u, with order $l < 2^{poly(\lambda)}$ can potentially break the soundness of the argument system.
- If $(\mathbb{G}, g, h, T) \in \mathcal{L}_{\mathsf{EXP}}$, then $(\mathbb{G}, g, hu, T) \notin \mathcal{L}_{\mathsf{EXP}}$ and will be incorrectly accepted by the verifier with probability $1/l$.
- Malicious prover sends $v \leftarrow g^{(2^{T/2})}u \in \mathbb{G}$.
- Soundness of the argument system does not hold whenever $r + 1 \equiv 2^{T/2} \mod l$, since $(\mathbb{G}, g^r v, v^r(hu), T/2) \in \mathcal{L}_{\mathsf{EXP}}$.

Breaking soundness of Pietrzak's VDF and non-necessity of the exponential LO assumption

- Given (u, l), a low order element u, with order $l < 2^{poly(\lambda)}$ can potentially break the soundness of the argument system.
- If $(\mathbb{G}, g, h, T) \in \mathcal{L}_{\text{EXP}}$, then $(\mathbb{G}, g, hu, T) \notin \mathcal{L}_{\text{EXP}}$ and will be incorrectly accepted by the verifier with probability $1/l$.
- Malicious prover sends $v \leftarrow g^{(2^{T/2})} u \in \mathbb{G}$.
- Soundness of the argument system does not hold whenever $r + 1 \equiv 2^{T/2} \mod l$, since $(\mathbb{G}, g^r v, v^r(hu), T/2) \in \mathcal{L}_{\text{EXP}}$.

$$(g^r v)^{2^{(T/2)}} = v^r(hu)$$

Breaking soundness of Pietrzak's VDF and non-necessity of the exponential LO assumption

- Given (u, l), a low order element u, with order $l < 2^{poly(\lambda)}$ can potentially break the soundness of the argument system.
- If $(\mathbb{G}, g, h, T) \in \mathcal{L}_{\mathsf{EXP}}$, then $(\mathbb{G}, g, hu, T) \notin \mathcal{L}_{\mathsf{EXP}}$ and will be incorrectly accepted by the verifier with probability $1/l$.
- Malicious prover sends $v \leftarrow g^{(2^{T/2})} u \in \mathbb{G}$.
- Soundness of the argument system does not hold whenever $r + 1 \equiv 2^{T/2} \mod l$, since $(\mathbb{G}, g^r v, v^r(hu), T/2) \in \mathcal{L}_{\mathsf{EXP}}$.

$$(g^r v)^{2^{(T/2)}} = v^r(hu)$$

$$(g^r g^{(2^{T/2})} u)^{2^{(T/2)}} = (g^{(2^{T/2})} u)^r (hu)$$

Breaking soundness of Pietrzak's VDF and non-necessity of the exponential LO assumption

- Given (u, l), a low order element u, with order $l < 2^{poly(\lambda)}$ can potentially break the soundness of the argument system.
- If $(\mathbb{G}, g, h, T) \in \mathcal{L}_{\text{EXP}}$, then $(\mathbb{G}, g, hu, T) \notin \mathcal{L}_{\text{EXP}}$ and will be incorrectly accepted by the verifier with probability $1/l$.
- Malicious prover sends $v \leftarrow g^{(2^{T/2})} u \in \mathbb{G}$.
- Soundness of the argument system does not hold whenever $r + 1 \equiv 2^{T/2} \mod l$, since $(\mathbb{G}, g^r v, v^r(hu), T/2) \in \mathcal{L}_{\text{EXP}}$.

$$(g^r v)^{2^{(T/2)}} = v^r(hu)$$

$$(g^r g^{(2^{T/2})} u)^{2^{(T/2)}} = (g^{(2^{T/2})} u)^r (hu)$$

$$u^{(2^{T/2})} = u^{r+1} \iff r + 1 \equiv 2^{T/2} \mod l$$

Imprecision in [BBF18]

Soundness of the argument system does not hold whenever $r + 1 \equiv 2^{T/2} \mod l$, since $(\mathbb{G}, g^r v, v^r(hu), T/2) \in \mathcal{L}_{\mathsf{EXP}}$.

Imprecision in [BBF18]

Soundness of the argument system does not hold whenever $r+1 \equiv 2^{T/2} \mod l$, since $(\mathbb{G}, g^r v, v^r(hu), T/2) \in \mathcal{L}_{\mathsf{EXP}}$.
[BBF18]: "This happens with non-negligible probability."

Imprecision in [BBF18]

Soundness of the argument system does not hold whenever $r+1 \equiv 2^{T/2} \mod l$, since $(\mathbb{G}, g^r v, v^r(hu), T/2) \in \mathcal{L}_{\mathsf{EXP}}$.
[BBF18]: "This happens with non-negligible probability."
This is not true since $1 \leq l \leq 2^{poly(\lambda)}$

Imprecision in [BBF18]

Soundness of the argument system does not hold whenever $r+1 \equiv 2^{T/2} \mod l$, since $(\mathbb{G}, g^r v, v^r(hu), T/2) \in \mathcal{L}_{\mathsf{EXP}}$.
[BBF18]: "This happens with non-negligible probability."
This is not true since $1 \leq l \leq 2^{poly(\lambda)}$
There might be weird adversaries who can only return low order elements with their order being in $2^{\Theta(poly(\lambda))}$, even though they would break soundness of Pietrzak's protocol with negligible probability, ie. $1/2^{\Theta(poly(\lambda))}$.

Imprecision in [BBF18]

Soundness of the argument system does not hold whenever $r+1 \equiv 2^{T/2} \mod l$, since $(\mathbb{G}, g^r v, v^r(hu), T/2) \in \mathcal{L}_{\mathsf{EXP}}$.
[BBF18]: "This happens with non-negligible probability."
This is not true since $1 \leq l \leq 2^{poly(\lambda)}$
There might be weird adversaries who can only return low order elements with their order being in $2^{\Theta(poly(\lambda))}$, even though they would break soundness of Pietrzak's protocol with negligible probability, ie. $1/2^{\Theta(poly(\lambda))}$.
Remark: there are non-negligible $\phi(N)$ having factors in $2^{\Theta(poly(\lambda))}$. [BS13, Wei01]

Superpolynomial LO assumptions are sufficient

Definition

The *Subexponential Low Order assumption*. For any probabilistic polynomial time adversary $\mathcal{A}$, and for any $0 < \epsilon$, finding any element of subexponentially low order is hard:

$$\Pr\left[u^l = 1,\ u \notin \{1,-1\} : \begin{array}{c} \mathbb{G} \xleftarrow{\$} GGen(\lambda) \\ (u,l) \leftarrow \mathcal{A}(\mathbb{G}) \\ \text{and } l < 2^{\log^{1+\epsilon}(\lambda)} \end{array} \right] \leq \mathrm{negl}(\lambda) \quad (1)$$

Sufficiency of the superpolynomial LO assumption

It is a simple but technical proof which applies the general forking lemma [BN06].

Sufficiency of the superpolynomial LO assumption

It is a simple but technical proof which applies the general forking lemma [BN06].
Why can't we assume even weaker assumptions, like polynomial LO?

Sufficiency of the superpolynomial LO assumption

It is a simple but technical proof which applies the general forking lemma [BN06].

Why can't we assume even weaker assumptions, like polynomial LO?

In the Forking-lemma we have that breaking Pietrzak's soundness yields an adversary against $f(\lambda)$-LO with success probability $(\epsilon^2/t) - (\epsilon/f(\lambda))$.

Sufficiency of the superpolynomial LO assumption

It is a simple but technical proof which applies the general forking lemma [BN06].
Why can't we assume even weaker assumptions, like polynomial LO?
In the Forking-lemma we have that breaking Pietrzak's soundness yields an adversary against $f(\lambda)$-LO with success probability $(\epsilon^2/t) - (\epsilon/f(\lambda))$.
It seems that $f(\lambda)$ needs to be superpolynomial.

Sufficiency of the superpolynomial LO assumption

It is a simple but technical proof which applies the general forking lemma [BN06].
Why can't we assume even weaker assumptions, like polynomial LO?
In the Forking-lemma we have that breaking Pietrzak's soundness yields an adversary against $f(\lambda)$-LO with success probability $(\epsilon^2/t) - (\epsilon/f(\lambda))$.
It seems that $f(\lambda)$ needs to be superpolynomial.
Maybe with other techniques we can prove even the sufficiency of polynomial LO assumptions?

Factoring$\equiv$ superpolynomial LOs???

Theorem

Let $\mathfrak{B}$ be a fixed integer. The Factoring assumption is reducible in polynomial time to the Low Order assumption for RSA-moduli when $\phi(N)$ has no prime factor between $\mathfrak{B}$ and $2^{poly(\lambda)}$ and $gcd(p-1, q-1) = 2$.

Factoring$\equiv$ superpolynomial LOs???

Theorem

Let $\mathfrak{B}$ be a fixed integer. The Factoring assumption is reducible in polynomial time to the Low Order assumption for RSA-moduli when $\phi(N)$ has no prime factor between $\mathfrak{B}$ and $2^{poly(\lambda)}$ and $gcd(p-1, q-1) = 2$.

Let's assume there exists an efficient adversary $\mathcal{A}$, who can break the LO assumption with non-negligible probability.

$$\Pr[\mathcal{A} \quad breaks \quad LO] \geq \frac{1}{q(\lambda)}.$$

Factoring$\equiv$ superpolynomial LOs???

Theorem

Let $\mathfrak{B}$ be a fixed integer. The Factoring assumption is reducible in polynomial time to the Low Order assumption for RSA-moduli when $\phi(N)$ has no prime factor between $\mathfrak{B}$ and $2^{poly(\lambda)}$ and $gcd(p-1, q-1) = 2$.

Let's assume there exists an efficient adversary $\mathcal{A}$, who can break the LO assumption with non-negligible probability.

$$\Pr[\mathcal{A} \quad breaks \quad LO] \geq \frac{1}{q(\lambda)}.$$

We devise an efficient adversary $\mathcal{B}$ who can factor non-negligible fraction of random RSA moduli by using $\mathcal{A}$ as a subroutine.

Main idea of the reduction

Given (u, l) pair such that $u^l \equiv 1 \mod N$ and $2 \leq l \leq 2^{poly(\lambda)} \wedge u \neq -1$.

Main idea of the reduction

Given (u, l) pair such that $u^l \equiv 1 \mod N$ and $2 \leq l \leq 2^{poly(\lambda)} \wedge u \neq -1$.Note that, the order l of $u \in (\mathbb{Z}/pq\mathbb{Z})^\times \cong (\mathbb{Z}/p\mathbb{Z})^\times \times (\mathbb{Z}/q\mathbb{Z})^\times$ is the least common multiple of its (multiplicative) orders modulo p and modulo q, ie. $l = lcm(ord_p(u), ord_q(u))$.

Main idea of the reduction

Given (u, l) pair such that $u^l \equiv 1 \mod N$ and $2 \leq l \leq 2^{poly(\lambda)} \wedge u \neq -1$.Note that, the order l of $u \in (\mathbb{Z}/pq\mathbb{Z})^\times \cong (\mathbb{Z}/p\mathbb{Z})^\times \times (\mathbb{Z}/q\mathbb{Z})^\times$ is the least common multiple of its (multiplicative) orders modulo p and modulo q, ie. $l = lcm(ord_p(u), ord_q(u))$.

Note that, whenever $ord_p(u) \neq ord_q(u)$, adversary $\mathcal{B}$ could factor $N = pq$ if l was smooth enough.

Main idea of the reduction

Given (u, l) pair such that $u^l \equiv 1 \mod N$ and $2 \leq l \leq 2^{poly(\lambda)} \wedge u \neq -1$.Note that, the order l of $u \in (\mathbb{Z}/pq\mathbb{Z})^\times \cong (\mathbb{Z}/p\mathbb{Z})^\times \times (\mathbb{Z}/q\mathbb{Z})^\times$ is the least common multiple of its (multiplicative) orders modulo p and modulo q, ie. $l = lcm(ord_p(u), ord_q(u))$.

Note that, whenever $ord_p(u) \neq ord_q(u)$, adversary $\mathcal{B}$ could factor $N = pq$ if l was smooth enough.The reason being that, adversary $\mathcal{B}$ raises u to the power of $\frac{l}{r}$ for all prime factors r of l, until modulo one prime factor of N, but not the other the order of u divides $\frac{l}{r}$, which can be detected by $0 < \gcd(u^{\frac{l}{r}} - 1 \bmod N, N) < N$, hence factoring the modulus N.

Main idea of the reduction

Given (u, l) pair such that $u^l \equiv 1 \mod N$ and $2 \leq l \leq 2^{poly(\lambda)} \wedge u \neq -1$.Note that, the order l of $u \in (\mathbb{Z}/pq\mathbb{Z})^\times \cong (\mathbb{Z}/p\mathbb{Z})^\times \times (\mathbb{Z}/q\mathbb{Z})^\times$ is the least common multiple of its (multiplicative) orders modulo p and modulo q, ie. $l = lcm(ord_p(u), ord_q(u))$.
Note that, whenever $ord_p(u) \neq ord_q(u)$, adversary $\mathcal{B}$ could factor $N = pq$ if l was smooth enough.The reason being that, adversary $\mathcal{B}$ raises u to the power of $\frac{l}{r}$ for all prime factors r of l, until modulo one prime factor of N, but not the other the order of u divides $\frac{l}{r}$, which can be detected by $0 < \gcd(u^{\frac{l}{r}} - 1 \mod N, N) < N$, hence factoring the modulus N.
Hence, towards our goal one thing that we need to show is that $ord_p(u) \neq ord_q(u)$ with non-negligible probability.

Partial reduction II. $\Pr[ord_p(u) \neq ord_q(u)] =?$

$$\Pr[gcd(\frac{p-1}{2}, \frac{q-1}{2}) = 1 | p, q \in_R \mathcal{O}(2^{poly(\lambda)})] \approx \frac{1}{\zeta(2)} = \frac{6}{\pi^2}$$

Partial reduction II. $\Pr[ord_p(u) \neq ord_q(u)] =?$

$\Pr[gcd(\frac{p-1}{2}, \frac{q-1}{2}) = 1 | p, q \in_R \mathcal{O}(2^{poly(\lambda)})] \approx \frac{1}{\zeta(2)} = \frac{6}{\pi^2}$
So with constant probability $gcd(p-1, q-1) = 2$. Let's examine these two cases.

- $ord_p(u) = ord_q(u) = 1$. It can't be by the definition of LO!
- $ord_p(u) = ord_q(u) = 2$ Then we can factor trivially!

Partial reduction II. $\Pr[ord_p(u) \neq ord_q(u)] =?$

$\Pr[gcd(\frac{p-1}{2}, \frac{q-1}{2}) = 1 | p, q \in_R \mathcal{O}(2^{poly(\lambda)})] \approx \frac{1}{\zeta(2)} = \frac{6}{\pi^2}$
So with constant probability $gcd(p-1, q-1) = 2$. Let's examine these two cases.

- $ord_p(u) = ord_q(u) = 1$. It can't be by the definition of LO!
- $ord_p(u) = ord_q(u) = 2$ Then we can factor trivially!

Hence we can assume that $ord_p(u) \neq ord_q(u)$!

Partial reduction II. $\Pr[ord_p(u) \neq ord_q(u)] = ?$

$\Pr[gcd(\frac{p-1}{2}, \frac{q-1}{2}) = 1 | p, q \in_R \mathcal{O}(2^{poly(\lambda)})] \approx \frac{1}{\zeta(2)} = \frac{6}{\pi^2}$
So with constant probability $gcd(p-1, q-1) = 2$. Let's examine these two cases.

- $ord_p(u) = ord_q(u) = 1$. It can't be by the definition of LO!
- $ord_p(u) = ord_q(u) = 2$ Then we can factor trivially!

Hence we can assume that $ord_p(u) \neq ord_q(u)$!
Given order l ($1 \leq l \leq 2^{poly(\lambda)}$) of u, then $\mathcal{B}$ wants to find all of its prime factors in a brute force-manner, but still in polynomial-time in λ.

Partial reduction II. $\Pr[ord_p(u) \neq ord_q(u)] = ?$

$\Pr[gcd(\frac{p-1}{2}, \frac{q-1}{2}) = 1 | p, q \in_R \mathcal{O}(2^{poly(\lambda)})] \approx \frac{1}{\zeta(2)} = \frac{6}{\pi^2}$
So with constant probability $gcd(p-1, q-1) = 2$. Let's examine these two cases.

- $ord_p(u) = ord_q(u) = 1$. It can't be by the definition of LO!
- $ord_p(u) = ord_q(u) = 2$ Then we can factor trivially!

Hence we can assume that $ord_p(u) \neq ord_q(u)$!
Given order l $(1 \leq l \leq 2^{poly(\lambda)})$ of u, then $\mathcal{B}$ wants to find all of its prime factors in a brute force-manner, but still in polynomial-time in λ.Namely, $\mathcal{B}$ wants to find l's smallest prime factor l_1, which is smaller than $\mathfrak{B}$.

Partial reduction II. $\Pr[ord_p(u) \neq ord_q(u)] =?$

$\Pr[gcd(\frac{p-1}{2}, \frac{q-1}{2}) = 1 | p, q \in_R \mathcal{O}(2^{poly(\lambda)})] \approx \frac{1}{\zeta(2)} = \frac{6}{\pi^2}$
So with constant probability $gcd(p-1, q-1) = 2$. Let's examine these two cases.

- $ord_p(u) = ord_q(u) = 1$. It can't be by the definition of LO!
- $ord_p(u) = ord_q(u) = 2$ Then we can factor trivially!

Hence we can assume that $ord_p(u) \neq ord_q(u)$!
Given order l $(1 \leq l \leq 2^{poly(\lambda)})$ of u, then $\mathcal{B}$ wants to find all of its prime factors in a brute force-manner, but still in polynomial-time in λ.Namely, $\mathcal{B}$ wants to find l's smallest prime factor l_1, which is smaller than $\mathfrak{B}$.Suppose a_1 is the largest integer such that $l_1^{a_1} | l$.

Partial reduction II. $\Pr[ord_p(u) \neq ord_q(u)] =?$

$\Pr[gcd(\frac{p-1}{2}, \frac{q-1}{2}) = 1 | p, q \in_R \mathcal{O}(2^{poly(\lambda)})] \approx \frac{1}{\zeta(2)} = \frac{6}{\pi^2}$
So with constant probability $gcd(p-1, q-1) = 2$. Let's examine these two cases.

- $ord_p(u) = ord_q(u) = 1$. It can't be by the definition of LO!
- $ord_p(u) = ord_q(u) = 2$ Then we can factor trivially!

Hence we can assume that $ord_p(u) \neq ord_q(u)$!
Given order l $(1 \leq l \leq 2^{poly(\lambda)})$ of u, then $\mathcal{B}$ wants to find all of its prime factors in a brute force-manner, but still in polynomial-time in λ.Namely, $\mathcal{B}$ wants to find l's smallest prime factor l_1, which is smaller than $\mathfrak{B}$.Suppose a_1 is the largest integer such that $l_1^{a_1}|l$.Then, recursively we would like to find the smallest prime factor of $\frac{l}{l_1^{a_1}}$, denoted l_2 which is smaller than $\mathfrak{B}$ and so on.

Partial reduction II. $\Pr[ord_p(u) \neq ord_q(u)] =?$

$\Pr[gcd(\frac{p-1}{2}, \frac{q-1}{2}) = 1 | p, q \in_R \mathcal{O}(2^{poly(\lambda)})] \approx \frac{1}{\zeta(2)} = \frac{6}{\pi^2}$
So with constant probability $gcd(p-1, q-1) = 2$. Let's examine these two cases.

- $ord_p(u) = ord_q(u) = 1$. It can't be by the definition of LO!
- $ord_p(u) = ord_q(u) = 2$ Then we can factor trivially!

Hence we can assume that $ord_p(u) \neq ord_q(u)$!
Given order l $(1 \leq l \leq 2^{poly(\lambda)})$ of u, then $\mathcal{B}$ wants to find all of its prime factors in a brute force-manner, but still in polynomial-time in λ.Namely, $\mathcal{B}$ wants to find l's smallest prime factor l_1, which is smaller than $\mathfrak{B}$.Suppose a_1 is the largest integer such that $l_1^{a_1} | l$.Then, recursively we would like to find the smallest prime factor of $\frac{l}{l_1^{a_1}}$, denoted l_2 which is smaller than $\mathfrak{B}$ and so on.Hence adversary $\mathcal{B}$ hopes that all of the prime factors of l are smaller than $\mathfrak{B}$.

Partial Reduction III. LO-smooth Integers [Wei01]

We need to establish the fraction of primes up to $N = \mathcal{O}(2^{s(\lambda)})$, that do not have prime factors in $(\mathfrak{B}, 2^{poly(\lambda)}]$.

Partial Reduction III. LO-smooth Integers [Wei01]

We need to establish the fraction of primes up to $N = \mathcal{O}(2^{s(\lambda)})$, that do not have prime factors in $(\mathfrak{B}, 2^{poly(\lambda)}]$.

$$\mathcal{P}_{los}(\lambda) = \frac{\Gamma(2^{s(\lambda)}, 2^{poly(\lambda)}, \mathfrak{B})}{2^{s(\lambda)}} \approx \frac{2^{s(\lambda)}\eta(s(\lambda)/poly(\lambda), s(\lambda)/\mathfrak{B})}{2^{s(\lambda)}} \geq$$

$$\geq \frac{s(\lambda)/poly(\lambda)}{2s(\lambda)/\mathfrak{B}} = \frac{\mathfrak{B}}{2poly(\lambda)}.$$

Partial Reduction III. LO-smooth Integers [Wei01]

We need to establish the fraction of primes up to $N = \mathcal{O}(2^{s(\lambda)})$, that do not have prime factors in $(\mathfrak{B}, 2^{poly(\lambda)}]$.

$$\mathcal{P}_{los}(\lambda) = \frac{\Gamma(2^{s(\lambda)}, 2^{poly(\lambda)}, \mathfrak{B})}{2^{s(\lambda)}} \approx \frac{2^{s(\lambda)}\eta(s(\lambda)/poly(\lambda), s(\lambda)/\mathfrak{B})}{2^{s(\lambda)}} \geq$$

$$\geq \frac{s(\lambda)/poly(\lambda)}{2s(\lambda)/\mathfrak{B}} = \frac{\mathfrak{B}}{2poly(\lambda)}.$$

$$\Pr[\mathcal{B} \quad breaks \quad Factoring] \geq \frac{6}{\pi^2} q(\lambda)\mathcal{P}_{los}^2(\lambda)$$

A cool application of a HVZK by Goldberg et al [GRSB19]

Lemma

The map $x \to x^e \mod N$ is a permutation of Z_N^* if and only if $gcd(e, \phi(N)) = 1$.

A cool application of a HVZK by Goldberg et al [GRSB19]

Lemma

The map $x \to x^e \mod N$ is a permutation of Z_N^* if and only if $gcd(e, \phi(N)) = 1$.

We have an efficient ZKP for this language by Goldberg et al.:

$$\mathcal{L}_{\mathrm{perm}\mathbb{Z}_N^*} = \{(N, e) | N, e > 0 \wedge gcd(e, \phi(N)) = 1\}$$

A cool application of a HVZK by Goldberg et al [GRSB19]

Lemma

The map $x \to x^e \mod N$ is a permutation of Z_N^* if and only if $gcd(e, \phi(N)) = 1$.

We have an efficient ZKP for this language by Goldberg et al.:

$$\mathcal{L}_{\text{perm}\mathbb{Z}_N^*} = \{(N, e) | N, e > 0 \wedge gcd(e, \phi(N)) = 1\}$$

Let p_n denote the largest prime smaller than $\mathfrak{B}$. Then let $e = \prod_{i=1}^{n} p_i$, where p_i is the ith odd prime.

A cool application of a HVZK by Goldberg et al [GRSB19]

Lemma

The map $x \to x^e \mod N$ is a permutation of Z_N^* if and only if $gcd(e, \phi(N)) = 1$.

We have an efficient ZKP for this language by Goldberg et al.:

$$\mathcal{L}_{\text{perm}\mathbb{Z}_N^*} = \{(N, e) | N, e > 0 \wedge gcd(e, \phi(N)) = 1\}$$

Let p_n denote the largest prime smaller than $\mathfrak{B}$. Then let $e = \prod_{i=1}^{n} p_i$, where p_i is the ith odd prime.
In a typical parameter setting $\lambda = 80, \mathcal{B} = 2^{10}$ the proof is $6.4KB$.
Generally the proof contains $\approx \lambda / \log 3$ group elements and requires the same amount of modular exponentiations.

Open Problems

- Subexponential LO assumptions are equivalent to Factoring?

Open Problems

- Subexponential LO assumptions are equivalent to Factoring?
- What would be a **necessary and sufficient** assumption for proving the soundness of Pietrzak's protocol?

Open Problems

- Subexponential LO assumptions are equivalent to Factoring?
- What would be a **necessary and sufficient** assumption for proving the soundness of Pietrzak's protocol?
- Is the Adaptive Root assumption a secure assumption?

Open Problems

- Subexponential LO assumptions are equivalent to Factoring?
- What would be a **necessary and sufficient** assumption for proving the soundness of Pietrzak's protocol?
- Is the Adaptive Root assumption a secure assumption?
- How does AR relate to Factoring, Strong RSA, RSA etc?

Open Problems

- Subexponential LO assumptions are equivalent to Factoring?
- What would be a **necessary and sufficient** assumption for proving the soundness of Pietrzak's protocol?
- Is the Adaptive Root assumption a secure assumption?
- How does AR relate to Factoring, Strong RSA, RSA etc?
- Is there any special-purpose algorithm which breaks either LO or AR assumption and is asymptotically faster than the fastest known factoring algorithm?

Open Problems

- Subexponential LO assumptions are equivalent to Factoring?
- What would be a **necessary and sufficient** assumption for proving the soundness of Pietrzak's protocol?
- Is the Adaptive Root assumption a secure assumption?
- How does AR relate to Factoring, Strong RSA, RSA etc?
- Is there any special-purpose algorithm which breaks either LO or AR assumption and is asymptotically faster than the fastest known factoring algorithm?
- If you could answer any of these questions then you can claim some cash! For more info, see https://rsa.cash

References I

Dan Boneh, Benedikt Bünz, and Ben Fisch, *A survey of two verifiable delay functions.*, IACR Cryptology ePrint Archive **2018** (2018), 712.

Benedikt Bünz, Steven Goldfeder, and Joseph Bonneau, *Proofs-of-delay and randomness beacons in ethereum*, IEEE Security and Privacy on the blockchain (IEEE S&B) (2017).

Mihir Bellare and Gregory Neven, *Multi-signatures in the plain public-key model and a general forking lemma*, Proceedings of the 13th ACM conference on Computer and communications security, 2006, pp. 390–399.

References II

- Eric Bach and Jonathan P Sorenson, *Approximately counting semismooth integers*, Proceedings of the 38th International Symposium on Symbolic and Algebraic Computation, 2013, pp. 23–30.
- Bram Cohen and Krzysztof Pietrzak, *The chia network blockchain*, 2019.
- Ben Fisch, Joseph Bonneau, Nicola Greco, and Juan Benet, *Scaling proof-of-replication for filecoin mining*, Tech. report, Technical Report. Stanford University. Accessed May, 2019.
- Shafi Goldwasser and Yael Tauman Kalai, *Cryptographic assumptions: A position paper*, Theory of Cryptography Conference, Springer, 2016, pp. 505–522.

References III

Sharon Goldberg, Leonid Reyzin, Omar Sagga, and Foteini Baldimtsi, *Efficient noninteractive certification of rsa moduli and beyond*, International Conference on the Theory and Application of Cryptology and Information Security, Springer, 2019, pp. 700–727.

Esteban Landerreche, Marc Stevens, and Christian Schaffner, *Non-interactive cryptographic timestamping based on verifiable delay functions.*, IACR Cryptology ePrint Archive **2019** (2019), 197.

Oded Regev, *On lattices, learning with errors, random linear codes, and cryptography*, Journal of the ACM (JACM) **56** (2009), no. 6, 1–40.

References IV

- Lior Rotem, Gil Segev, and Ido Shahaf, *Generic-group delay functions require hidden-order groups*.
- Andreas Weingartner, *Integers free of prime divisors from an interval, i*, Acta Arithmetica **98** (2001), 117–131.
- Benjamin Wesolowski, *Efficient verifiable delay functions*, Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 2019, pp. 379–407.

Acknowledgements

We thank **Benjamin Wesolowski, Dankrad Feist, Dmitry Khovratovich** and **Justin Drake** for insightful discussions.

Thanks!

Questions?