# An Efficient Modular Exponentiation Proof Scheme

Darren Li

December 2, 2022

# Overview

- We present an efficient proof scheme for left-to-right modular exponentiation
- When one calculates $a^n \equiv r \pmod{m}$ using left-to-right modular exponentiation, *we can prove and verify the correctness of this calculation much faster than the calculation itself*
- Among other uses, we have doubled the speed of the current search for Generalized Fermat primes at PrimeGrid using our work

# PrimeGrid

- Large primes hold significant scientific value to number theorists; PrimeGrid is the largest distributed systematic search for primes, with 350K volunteers worldwide
- One of the types of primes PrimeGrid searches for are Generalized Fermat primes, i.e. primes of the form $b^{(2^k)} + 1$ where $k \leq 22$ for integer $b$; *Generalized Fermat search may produce a prime of record-breaking size*

# Technology of PrimeGrid

- To search for Generalized Fermat primes, PrimeGrid uses *Genefer*, developed by my advisor Yves Gallot, for primality tests of Generalized Fermat prime candidates

# Fast verification - why?

- In distributed computing, all results must be verified
- Slow verification does the same work twice, and this is what PrimeGrid previously did
- Fast verification involves creating a proof that the calculation is right, and requires that both making the proof and verifying the proof is efficient
- We show that this is possible for all modular exponentiation calculations, which is the bulk of primality tests
- This reduces the work necessary for a primality test on PrimeGrid from $2$ times to $\sim 1.01$ times

# Gerbicz-Pietrzak proof scheme

- The most general proof scheme preceding ours is the Gerbicz-Pietrzak proof scheme, using a combination of the ideas presented by Robert Gerbicz, and the Pietrzak VDF construction by Krzysztof Pietrzak
- The Gerbicz-Pietrzak proof scheme only applies to calculations of the form $a^{(2^L)} \bmod m$, i.e. a process of squaring a value $L$ times

# Gerbicz double check

- Gerbicz presented a *double check scheme*:

## Gerbicz double check scheme

Let $x_i = a^{2^i}$ represent the steps of the squaring process. To verify that $x_B, x_{2B}, x_{3B}, \ldots$ satisfies $x_i = x_{i-B}^{2^B}$, we may instead check:

$$\prod_{i=1}^{j} x_{iB} = \prod_{i=1}^{j} x_{(i-1)B}^{2^B} = \left(\prod_{i=0}^{j-1} x_{iB}\right)^{2^B}$$

- Can catch computation errors on friendly hosts, but is too large to use and too easy to fake to use as proof

# Gerbicz-Pietrzak Example

- As an example, consider the computation of $2^{256}$:

$$\underbrace{2^1}_{\underline{x_0}} \to \underbrace{2^2}_{x_1} \to \underbrace{2^4}_{\underline{x_2}} \to \underbrace{2^8}_{x_3} \to \underbrace{2^{16}}_{\underline{x_4}} \to \underbrace{2^{32}}_{x_5} \to \underbrace{2^{64}}_{\underline{x_6}} \to \underbrace{2^{128}}_{x_7} \to \underbrace{2^{256}}_{\underline{x_8}}$$

- Gerbicz double check: $(x_2x_4x_6x_8) \stackrel{?}{=} (x_0x_2x_4x_6)^4$
- Gerbicz-Pietrzak proof: initially, $x_8 \stackrel{?}{=} x_0^{256}$; prover sends $x_4$, verifier selects $Q = 3$, becomes $x_4x_8^3 \stackrel{?}{=} (x_0x_4^3)^{16}$
- Prover sends $(x_0x_4^3)^4 = x_2x_6^3$, verifier selects $Q = 2$, becomes $x_2x_4^2x_6^3x_8^6 \stackrel{?}{=} (x_0x_2^2x_4^3x_6^6)^4$ which the verifier can compute
- Gerbicz-Pietrzak proof is then $[x_8, x_4, x_2x_6^3]$

# Our contribution

- We generalize the Gerbicz-Pietrzak construction to a sound proof scheme for all instances of left-to-right modular exponentiation
- Our methods have been implemented in *Genefer22* (the successor to Genefer) and **have already been** deployed on PrimeGrid, doubling PrimeGrid's search efficiency and creating a fair opportunity for everyone

# Proof scheme

- We begin by describing a simple extension of the Gerbicz double check to general exponents
- An analog of the Pietrzak proof scheme is impossible to directly accomplish
- We begin by approaching the problem informally, introducing *claims* that can be easily split
- We then formalize this interaction as a succinct argument for a formal language

# Soundness of proof scheme

- To prove soundness even in adversarial conditions, we use the *generalized forking lemma* to show soundness assuming the low order assumption
- **In other words, if the low order assumption cannot be efficiently broken, one cannot efficiently forge a proof of a false result**

# Double check scheme

- For left-to-right modular exponentiation of $a^n$, the sequence of calculations is instead $u_i = a^{\lfloor n/2^i \rfloor}$
- The relation between $u_i$ and $u_{i+j}$ is $u_i = u_{i+j}^{2^j} a^{\lfloor n/2^i \rfloor \bmod 2^j}$
- Taking products across $i = 0, B, 2B, \ldots$ and $j = B$ provides an analogy for the Gerbicz double check:

$$\prod_{i=0} u_{iB} = \left( \prod_{i=0} u_{iB+B} \right)^{2^B} a^{\sum_{i=0} \lfloor n/2^{iB} \rfloor \bmod 2^B}$$

# Interactive proof

Consider the "product" of many different claims.
Let $S(iB, B) = a^{\lfloor n/2^{iB} \rfloor} \bmod 2^B$ so $u_{iB} = u_{iB+B}^{2^B} S(iB, B)$, then:

$$P(i, B) = \left\{ u_{iB} \stackrel{?}{=} u_{(i+1)B}^{2^B} S(iB, B) \right\}$$

# Interactive proof, part 2

The product $P_1 = \left\{a \stackrel{?}{=} b\right\}$ and $P_2 = \left\{c \stackrel{?}{=} d\right\}$ is $P_1 P_2 = \left\{ac \stackrel{?}{=} bd\right\}$

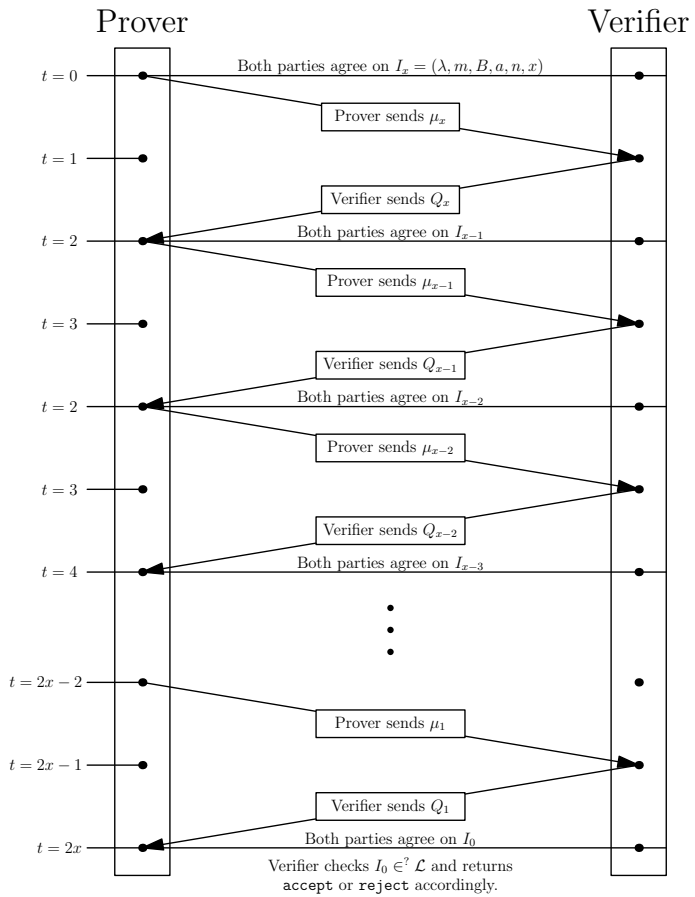
$$P(i, 2B) \iff P(2i, B) \wedge P(2i+1, B) \overset{1-\text{Negl}}{\iff} P(2i, B)P(2i+1, B)^Q$$

This allows us to recursively decompose the original claim:

$$\underbrace{\prod_{i=0} P(i, 2B)^{w_i}}_{\mathbf{A}} \rightarrow \underbrace{\left(\prod_{i=0} P(2i, B)^{w_i}\right)}_{\mathbf{B}} \underbrace{\left(\prod_{i=0} P(2i+1, B)^{w_i}\right)^Q}_{\mathbf{C}}$$

# Interactive proof, part 3

- Suppose $\mathbf{A} = \left\{ \mathbf{A}_1 \stackrel{?}{=} \mathbf{A}_2^{2^B} a^{\mathbf{A}_\Sigma} \right\}$ and likewise for $\mathbf{B}$ and $\mathbf{C}$; $\mathbf{A}/\mathbf{B}/\mathbf{C}_\Sigma$ is easy to calculate, and $\mathbf{A}_1 = \mathbf{B}_1$, $\mathbf{A}_2 = \mathbf{C}_2$, $\mathbf{B}_2 = \mathbf{C}_1$; so all the prover needs to send in one round is $\mathbf{B}_2$
- After enough rounds, the verifier can expand the claim and compute whether or not $\mathbf{A}_1 = \mathbf{A}_2^{2^B} a^{\mathbf{A}_\Sigma}$ actually holds




Interactive proof process

# Example for our proof scheme

- Consider the computation of $2^{137}$:

$$\underbrace{2^0}_{\underline{u_8}} \to \underbrace{2^1}_{u_7} \to \underbrace{2^2}_{\underline{u_6}} \to \underbrace{2^4}_{u_5} \to \underbrace{2^8}_{\underline{u_4}} \to \underbrace{2^{17}}_{u_3} \to \underbrace{2^{34}}_{\underline{u_2}} \to \underbrace{2^{68}}_{u_1} \to \underbrace{2^{137}}_{\underline{u_0}}$$

- Initial claim: $u_0 \stackrel{?}{=} u_8^{256} 2^{137}$; prover sends $u_4$, becomes $u_4 \stackrel{?}{=} u_8^{16} 2^8$ and $u_0 \stackrel{?}{=} u_4^{16} 2^{137-8\times 16}$, or together, $(u_0 u_4^Q) \stackrel{?}{=} (u_4 u_8^Q)^{16} 2^{8Q+9}$
- Prover sends $u_2 u_6^Q$, claim becomes

$$(u_0 u_2^{Q_2} u_4^Q u_6^{Q_2 Q}) \stackrel{?}{=} (u_2 u_4^{Q_2} u_6^Q u_8^{Q_2 Q})^4 2^{2Q_2 Q+2Q_2+1}$$

# Proof of soundness

## Generalized Forking Lemma (Bellare, Neven 2005)

For **any** randomized Turing machine $\mathcal{A}'$ run on inputs $(I; Q_x, Q_{x-1}, \ldots, Q_1; R)$ for random tape $R$, randomly sampled $I$ and $Q \leftarrow [1, g]$, suppose it produces $(J, \sigma)$, $0 \leq J \leq x$, $1 \leq \sigma \leq h$. Let $P = \Pr[J \neq 0]$. Sample $Q' \leftarrow [1, g]$ and run *again* $\mathcal{A}'(I; Q_x, Q_{x-1}, \ldots, Q_{J+1}, Q'_J, Q'_{J-1}, \ldots, Q'_1; R) \rightarrow (J', \sigma')$. Then $J = J'$ but $0 < J$ and $\sigma \neq \sigma'$ with probability at least $P^2/x - P/h$.

Along with Bellare and Neven's formal proof, intuitively, there are only $x$ possible values for $J$ but many more for $\sigma$, so it averages out - the number of pairs of different $J$ is bounded

# Proof of soundness, part 2

- Suppose there is an adversary, that when given an initially false claim can interact and fabricate sent values $\mu_x, \mu_{x-1}, \ldots, \mu_1$ to cause the verifier to finish with a true claim with non-negligible probability; the adversary deceives the verifier
- Consider a machine, that interacts with this adversary, giving $Q_x, Q_{x-1}, \ldots, Q_1$, and returns $(y, Q_y)$ where $y$ is the last round such that the claim was false, as well as $\mu_y$ and the false claim

# Proof of soundness, part 3

- Then by the Generalized Forking Lemma, we can find some input such that the resulting $y$ is the same and but different $Q_y, Q'_y$ with non-negligible probability, using the adversary twice
- Both $\mu_y$ and the last false claim will be same for both the $Q_y$ run and the $Q'_y$ run

# Proof of soundness, part 4

- Suppose the last false claim has form $R_1 \stackrel{?}{=} B_1^{2^{2s}} a^C$: then the value the prover, by sending the value $\mu$, is claiming that both $R_1 \stackrel{?}{=} \mu^{2^s} a^{c_1}$, or $\mu \stackrel{?}{=} B_1^{2^s} a^{c_2}$ - one of which must be false
- Then $\left(B_1^{2^s} a^{c_2}/\mu\right)^{|Q-Q'|} = 1$ while $1 \leq |Q - Q'| < 2^\lambda$ and $B_1^{2^s} a^{c_2}/\mu \neq 1$; the low order assumption is broken

# Finale

We have proven that if the low order assumption cannot be quickly broken, neither can our proof scheme, attaining the same safety as the general Pietrzak VDF.
It remains an open problem to prove unconditional soundness beyond the constraints presented by Pietrzak.

Thanks!

# Pietrzak VDF construction

- We can apply the Pietrzak VDF construction to interactively halve the claim that $u_n = u_0^{2^L}$ until we reach the limit of stored checkpoints

## One round of the Pietrzak VDF

Let $Q$ be randomly sampled from $[1, 2^\lambda]$ by the verifier, for a security parameter $\lambda$, then:

$$a^{2^{2s}} = r \iff a^{2^s} = \beta \text{ and } \beta^{2^s} = r \overset{P=1-\mathsf{Negl}(\lambda)}{\iff} (a\beta^Q)^{2^s} = \beta r^Q$$

- Fiat-Shamir heuristic can turn the exchanged $\beta$ values into a proof

# Of claim multiplication

We have:

$$P(2i, B) \cdot P(2i+1, B)^Q$$

$$= \left\{ u_{2iB} u_{(2i+1)B}^Q \stackrel{?}{=} \left( u_{(2i+1)B} u_{(2i+2)B}^Q \right)^{2^B} S(2iB, B) S(2iB + B, B)^Q \right\}$$

# Why is there no such analogy for Pietrzak?

Define $E_1 = \lfloor n/2^i \rfloor \bmod 2^{2s}$, $E_2 = \lfloor n/2^i \rfloor \bmod 2^s$, $E_3 = \lfloor n/2^{i+s} \rfloor \bmod 2^s$, $E_4 = E_2 + Q\,E_3$; if we mirror Pietrzak we get

$$
\begin{aligned}
& u_i = u_{i+2s}^{2^{2s}} a^{E_1} \\
\iff & u_i = u_{i+s}^{2^s} a^{E_2} \wedge u_{i+s} = u_{i+2s}^{2^s} a^{E_3} \\
\overset{\text{Negl}}{\iff} & u_i u_{i+s}^{Q} = \left( u_{i+s} u_{i+2s}^{Q} \right)^{2^s} a^{E_4}
\end{aligned}
$$

This obtained form does not have the initial form; we cannot divide it again.

# For interactive proof

The verifier can group together and verify the final claim as follows:

$$\prod_{i=0}^{j} P(i,B)^{w_i} = \left\{ \prod_{i=0}^{j} u_{iB}^{w_i} \stackrel{?}{=} \left( \prod_{i=0}^{j} u_{(i+1)B}^{w_i} \right)^{2^B} a^{\sum_{i=0}^{j} w_i \left( \lfloor n/2^{iB} \rfloor \bmod 2^B \right)} \right\}$$

# Also for interactive proof

- Formally, a claim consists of the result $r$, the beginning $b$, the size $2s$, and the current weights $w_i$, forming

$$r = b^{2^{2s}} a^{\sum w_i \left( \lfloor n/2^{2is} \rfloor \bmod 2^{2s} \right)}$$

- When the prover sends $\mu$ and the verifier selects $Q$, the claim becomes

$$\mu^Q r = \left( b^Q \mu \right)^{2^s} a^{\sum Q^{i \bmod 2} w_{\lfloor i/2 \rfloor} \left( \lfloor n/2^{is} \rfloor \bmod 2^s \right)}$$

# Proof of soundness: why?

- After being reduced y both $(\mu, Q)$ and $(\mu, Q')$ the resulting claims are true, i.e.

$$\mu^{Q} R_1 = \left(B_1 \mu^{Q}\right)^{2^s} a^{c_1 + Q c_2} \qquad \mu^{Q'} R_1 = \left(B_1 \mu^{Q'}\right)^{2^s} a^{c_1 + Q' c_2}$$

- so...

$$R_1 / \left(\mu^{2^s} a^{c_1}\right) = \left(B_1^{2^s} a^{c_2} / \mu\right)^{Q} = \left(B_1^{2^s} a^{c_2} / \mu\right)^{Q'}$$