ChatGPT

Algorithmes de test de primalité les plus rapides (étude approfondie)

1. Algorithme de primalité le plus rapide pour $n <$ 65 535 bits

Pour des entiers de taille jusqu'à ~65 535 bits (environ 20 000 chiffres décimaux), le test probabiliste **Miller-Rabin** est généralement considéré comme le plus rapide en pratique. Ce test monte-carlo détecte un nombre composé avec une probabilité d'au moins 3/4 à chaque itération, et en répétant le test avec plusieurs bases aléatoires on peut rendre la probabilité d'erreur négligeablement petite (typiquement $< 2^{-128}$) [1] [2]. En effet, le seul type d'erreur possible est de qualifier *à tort* un nombre composé de "probablement premier" (faux positif), mais on peut rendre ce risque plus faible que celui d'une défaillance matérielle en répétant le test. Le coût d'une itération de Miller-Rabin est très modéré: avec une exponentiation modulaire optimisée par FFT, le temps d'exécution est de l'ordre de $O(k \cdot (\log n)^2 \log\log n)$ pour k itérations [3]. Grâce à sa simplicité d'implémentation et sa rapidité, Miller-Rabin est omniprésent dans les logiciels (ex: génération de clés RSA) [4]. En pratique, pour des entiers jusqu'à 65k bits, quelques itérations de Miller-Rabin (éventuellement précédées de divisions d'essai par de petits primes) suffisent à obtenir un test de primalité ultra-rapide et fiable.

Si l'on exige une **certitude absolue** (preuve déterministe), l'algorithme le plus efficace reste l'**ECPP** (*Elliptic Curve Primality Proving*). ECPP est actuellement *« l'algorithme le plus rapide connu pour tester la primalité de nombres généraux »* [5]. En pratique, ECPP peut prouver la primalité d'entiers de 20 000 chiffres en quelques heures sur un PC moderne. Par exemple, en 2025 un nombre primo de 109 297 chiffres (≈**363 000 bits**) a été certifié premier par ECPP [6]. ECPP est donc tout à fait applicable en-dessous de 65 535 bits. Son implémentation la plus répandue, **Primo** de Marcel Martin, a longtemps détenu les records de preuve de primalité, rivalisant avec des versions optimisées par J. Franke et al. (dite *fastECPP*) [7]. En résumé, pour des entiers "petits" (jusqu'à ~65 535 bits), **Miller-Rabin** offre la vitesse maximale (test probabiliste), tandis que **ECPP** est le plus rapide pour une preuve déterministe.

2. Algorithme le plus rapide pour des entiers de taille arbitraire

Lorsque la taille des entiers devient arbitrairement grande, les algorithmes qui dominent en pratique restent sensiblement les mêmes. Le test de Miller-Rabin conserve une complexité *polynomiale average-case* en la taille de n et peut s'appliquer à des nombres gigantesques (plusieurs centaines de milliers de bits) tant qu'on accepte son caractère probabiliste. En revanche, si l'on cherche l'algorithme *asymptotiquement* le plus performant avec certitude, les méthodes comme **ECPP** et **APR-CL** sont incontournables. ECPP demeure le champion empirique pour les entiers généraux de très grande taille [5]. Son temps d'exécution n'est pas prouvé polynomial dans le pire cas, mais heuristiquement il tourne en $O((\log n)^{5+\varepsilon})$ (avec multiplication rapide) [8], ce qui est quasi-polynomial. L'algorithme **APR-CL** (Adleman-Pomerance-Rumely amélioré par Cohen & Lenstra) offre une complexité garantie subexponentielle ~$(\log n)^{O(\log\log\log n)}$ [9]. En pratique toutefois, APR-CL est supplanté par ECPP, un peu plus rapide et qui fournit en prime un **certificat** de primalité [10].

Notons qu'il existe aussi un algorithme déterministe **polynomial en théorie**, l'algorithme **AKS** (Agrawal–Kayal–Saxena, 2002). AKS a une complexité prouvée initialement en $O((\log n)^{12})$, réduite plus tard à environ $O((\log n)^6)$ avec des optimisations [11]. Cependant, AKS est nettement plus lent

en pratique que ECPP ou APR-CL pour les tailles d'entiers d'intérêt (des dizaines de milliers de bits) [12]. En somme, pour des entiers arbitrairement grands, la méthode la plus rapide connue est **ECPP** (sous conjectures raisonnables) et, à défaut de certitude, le test **Miller-Rabin** répété reste utilisable quelle que soit la taille (son coût croît polynomiquement en $\log n$). On peut également combiner les approches: par exemple, filtrer les candidats par Miller-Rabin, puis effectuer une preuve finale par ECPP pour bénéficier à la fois de la vitesse et de la certitude [13] [14].

3. Analyse approfondie des complexités

Miller-Rabin est un algorithme probabiliste de classe *Las Vegas* (toujours correct quand il renvoie "composé", et une petite chance d'erreur sur "probablement premier"). Sa complexité par test est dominée par une exponentiation modulaire $a^d \bmod n$. Avec l'exponentiation rapide, cela nécessite $O(\log n)$ multiplications modulaires. En arithmétique classique (multiplication en $O((\log n)^2)$), une itération coûte donc $O((\log n)^3)$. Des améliorations via FFT ramènent ce coût à ~$O((\log n)^2 \log\log n)$ [3]. Comme au plus 25% des bases peuvent *mentir* pour un composé [15], il suffit de quelques bases aléatoires pour obtenir une confiance extrêmement élevée. Ainsi, la complexité attendue pour distinguer un nombre composé est $O((\log n)^3)$, et pour certifier un nombre **probablement** premier $O(k\cdot (\log n)^3)$ avec k petit (par ex. $k=64$). L'espace mémoire requis est minime (quelques entiers de la taille de n).

ECPP (Elliptic Curve Primality Proving) a une complexité heuristique quasi-polynomiale. D'après Atkin et Morain, sa version originale a une complexité attendue en $O((\log n)^{5+\varepsilon})$ [8], améliorée à $O((\log n)^{4+\varepsilon})$ dans une variante due à Shallit [16]. En pratique, ECPP se révèle extrêmement efficace : sa complexité empirique est souvent proche de linéaire dans le nombre de chiffres de n pour des tailles utilisées en cryptographie (centaines de chiffres), et elle reste gérable pour des nombres de plusieurs dizaines de milliers de chiffres. Il s'agit d'un algorithme **randomisé quasi-déterministe** : il utilise des choix aléatoires pour trouver des courbes elliptiques adaptées, mais produit en sortie une preuve certifiable. Notons que ECPP a été démontré *polynomial* pour *presque tous* les entiers (sous certaines distributions) [16], mais sans garantie uniforme dans le pire cas.

APR-CL est un algorithme déterministe de 1983, première méthode générale subexponentielle. Sa complexité exacte est technique, mais peut s'écrire ~$\exp(O(\log\log n \cdot \log\log\log n))$, ce qui se réécrit $(\log n)^{O(\log\log\log n)}$ [9]. Cette croissance est plus lente que n'importe quelle puissance $(\log n)^c$, ce qui rend APR-CL praticable pour des tailles assez grandes – toutefois il est moins performant que ECPP dans la plupart des cas pratiques [10].

AKS a une complexité prouvée polynomiale $O((\log n)^c)$ (avec c descendant d'environ 12 à ~6 dans les améliorations ultérieures) [11]. Malgré cela, les constantes cachées et la nature combinatoire de l'algorithme le pénalisent lourdement. Pour des entrées de taille réelle (jusqu'à quelques milliers de bits), AKS est largement dépassé par Miller-Rabin ou ECPP. Son intérêt est surtout théorique (prouver que PRIMES ∈ P sans conjectures).

Autres tests probabilistes: Le test de **Solovay-Strassen** (1977) est basé sur le symbole de Jacobi: sa complexité est semblable à Miller-Rabin (~$O((\log n)^3)$) mais il est un peu moins puissant (taux d'erreur de 50% par test contre 25% pour Miller-Rabin). Il est aujourd'hui peu utilisé au profit de Miller-Rabin. Le test **Fermat** (qui vérifie $a^{n-1}\equiv1 \pmod n$) est encore plus simple mais souffre de faux positifs pour les nombres de Carmichael; il reste employé pour un pré-dépistage rapide des composés [17] [18]. Une variante améliorée est le test **Baillie-PSW**, combinant un Miller-Rabin (base 2) et un test de Lucas: *aucun* contre-exemple n'est connu à ce jour. Baillie-PSW est traitée comme déterministe pour $n<2^{64}$ et largement au-delà [12]. De nombreux logiciels (OpenSSL, Java

BigInteger, GMP) utilisent BPSW ou un petit ensemble fixe de bases de Miller-Rabin pour un test **déterministe** jusqu'à certaines bornes (par exemple, tester bases ${2,7,61}$ suffit à rendre Miller-Rabin exact pour $n<2^{32}$ [19]).

En résumé, **Miller-Rabin** et consorts offrent une complexité quasi-linéaire en pratique sur la taille de l'entrée, au prix d'une probabilité d'erreur arbitrairement petite. Les algorithmes **ECPP/APR-CL** apportent une certitude avec une complexité subexponentielle (heuristiquement quasi-polynomiale pour ECPP). **AKS** assure la théorie avec une complexité polynomiale élevée, mais reste non-compétitif en pratique.

4. Comparaison avec le test de Miller-Rabin

Le test de Miller-Rabin étant la référence courante, il est instructif de comparer les autres approches à celui-ci:

- **Vitesse** : Miller-Rabin est extrêmement rapide pour détecter la composité, grâce à de simples opérations modulaires. En un clin d'œil, il élimine la plupart des non-premiers. Les algorithmes déterministes comme AKS sont bien plus lents (même pour des entiers de 1024–4096 bits, Miller-Rabin l'emporte haut la main [12]). ECPP, malgré sa complexité plus élevée, parvient souvent à prouver la primalité en quelques secondes pour des nombres de taille courante (2048 bits), ce qui reste acceptable. Cependant, Miller-Rabin excelle dès qu'il s'agit de tester *rapidement* un grand nombre de candidats, par exemple dans la génération de primes aléatoires pour RSA. Son overhead par test est minimal comparé aux calculs lourds d'ECPP.

- **Certitude vs probabilisme** : Miller-Rabin ne donne qu'un verdict "probablement premier". En pratique on choisit suffisamment de bases pour que le risque d'erreur soit négligeable (par ex. 64 itérations $\Rightarrow$ risque $<2^{-128}$). De plus, comme mentionné, une erreur signifierait *étiqueter un composé comme premier* (un faux positif) – événement jugé plus improbable qu'un bit aléatoire de RAM en panne. Dans la plupart des applications (cryptographie, où un faux positif induirait une clé invalide), c'est un compromis acceptable. À l'inverse, ECPP fournit une **preuve** : il établit mathématiquement que le nombre est premier, souvent sous la forme d'un certificat vérifiable en $O((\log n)^3)$. Pour des usages critiques (records de primalité, validations indépendantes), cette certitude est indispensable. APR-CL également donne un verdict certain, mais sans certificat facilement vérifiable par un tiers.

- **Complexité asymptotique** : Miller-Rabin appartient aux algorithmes probabilistes en temps polynomial (en moyenne). Son avantage est qu'il n'a pas de composant superpolynomial caché – hormis la nécessité éventuelle de plusieurs répétitions, qui reste un facteur constant dans la plupart des cas. Les tests déterministes ont des complexités plus élevées (voir section précédente). Par exemple, AKS est polynomial *dans le pire cas* mais pour des tailles où $(\log n)^6$ dépasse la complexité pratique de ECPP, il perd son intérêt. ECPP a une complexité empirique meilleure que Miller-Rabin pour prouver la primalité d'un **unique** grand nombre – mais si l'on doit tester des millions de nombres, Miller-Rabin sera plus rentable.

- **Implémentation** : Miller-Rabin est simple à coder (~20 lignes peuvent suffire). ECPP est beaucoup plus complexe à implémenter correctement, nécessitant des connaissances en courbes elliptiques et théorie des nombres. C'est pourquoi l'usage courant est d'appeler des bibliothèques éprouvées (par ex. `Primo`, `FLINT` ECPP, etc.) [5]. AKS est conceptuellement plus simple que ECPP mais son implémentation efficace reste pointue (optimisations polynômes, choix de paramètres). En pratique, on utilise Miller-Rabin pour la majorité des besoins, et on ne

recourt à un algorithme de preuve que dans les cas spécifiques où la *certitude absolue* ou un certificat sont requis [14].

En bref, **Miller-Rabin** demeure l'étalon or du test de primalité rapide, au point qu'il est souvent inutile de chercher plus rapide pour des entiers "courants". Les alternatives comme **ECPP** s'imposent surtout lorsqu'on vise la preuve rigoureuse de primalité, ou qu'on manipule des nombres hors de portée des tests probabilistes habituels (par ex. milliards de bits, ce qui dépasse largement les besoins actuels). Miller-Rabin sert même de composant à d'autres tests hybrides (ex: Baillie-PSW combine Miller-Rabin et Lucas). Son seul inconvénient théorique – le petit risque de faux positif – peut être atténué à volonté, rendant la différence avec un test déterministe souvent académique.

5. Perspectives récentes et approches innovantes

La recherche sur les tests de primalité continue d'apporter des optimisations et des variantes. En **2023**, des travaux ont proposé le test *Gauss-Euler* comme une amélioration possible des tests probabilistes classiques [20]. Cette méthode combine le critère d'Euler (symboles de Legendre) et des tests de congruences pour réduire encore la probabilité qu'un nombre composé passe entre les mailles. Par exemple, Gauss-Euler vérifie des conditions mod n pour des petites bases spécifiques (comme des primes $p\equiv5 \pmod 8$ et $p\equiv1 \pmod 8$) en plus de Miller-Rabin [21] [22]. Les auteurs annoncent un temps comparable à Miller-Rabin (essentiellement $O((\log n)^2)$ d'après leur analyse) et aucun faux positif détecté sur de larges intervalles [20]. Cependant, comme pour d'autres tests non prouvés, la possibilité d'un contre-exemple ne peut être totalement exclue – les inventeurs de Gauss-Euler offrent même une récompense pour qui trouverait un composé passant tous leurs critères [23]. Cela rappelle la situation du test Baillie-PSW, empirique mais sans contre-exemple connu depuis 40 ans.

Parallèlement, l'essor de l'**intelligence artificielle** a suscité des expérimentations sur l'application du *machine learning* à la primalité. Des réseaux de neurones ont été entraînés à prédire si un nombre est premier, mais ces tentatives restent confinées à des tailles modestes. Par exemple, dès 2006 Egri et Shultz ont présenté un réseau de neurones capable de reconnaître les nombres premiers jusqu'à un certain nombre de bits [24]. Ils ont utilisé une architecture *KBCC (knowledge-based cascade-correlation)* et obtenu des résultats sur de petits entiers, mais ont reconnu que *« la praticité de cette approche est éclipsée par d'autres algorithmes de détection de primarité »* classiques [24]. En effet, rien ne surpasse l'efficacité d'un test mathématique comme Miller-Rabin qui exploite la structure modulaire des nombres, alors qu'un réseau de neurones devrait en quelque sorte *réapprendre* ces critères. Les réseaux de neurones ont tendance à **sur-apprendre** la suite des premiers connue au lieu de découvrir une nouvelle "règle" générale [25]. Jusqu'ici, aucune méthode IA ne rivalise donc avec les algorithmes arithmétiques en termes de fiabilité et de passage à l'échelle sur de très grands entiers. Néanmoins, des recherches se poursuivent, par exemple en *apprentissage de nouveautés* (novelty detection) pour repérer des motifs caractéristiques des nombres composés vs premiers [26]. Ces travaux restent exploratoires et n'ont pas encore débouché sur un outil concret surpassant les méthodes existantes.

En somme, les avancées récentes confirment la robustesse des algorithmes de primalité établis. Des variantes hybrides comme Baillie-PSW ou Gauss-Euler renforcent la confiance dans les tests probabilistes en éliminant davantage de faux positifs connus [20]. Du côté déterministe, les améliorations se situent surtout dans l'implémentation (optimisations multiprécision, parallélisation d'ECPP, etc.) plutôt que dans de nouveaux algorithmes révolutionnaires. L'intérêt pour le machine learning dans ce domaine est pour l'instant plus théorique que pratique – une curiosité intellectuelle qui n'a pas encore transformé le paysage des tests de primalité.

Sources: Les informations ci-dessus proviennent de la littérature récente et des références classiques sur le sujet. Notamment, le **Journal of Student Research (2022)** offre une synthèse comparative des tests (Miller-Rabin, ECPP, APR-CL, AKS) [9] [3]. Les **pages du projet PrimePages** documentent les records obtenus via ECPP et détaillent sa complexité heuristique [8]. Enfin, les discussions sur **Stack Exchange/Overflow** confirment le statut de ECPP comme algorithme le plus rapide connu pour un test de primalité général [5] et soulignent l'omniprésence de Miller-Rabin en pratique [4]. Toutes ces sources s'accordent pour dire que le **test de Miller-Rabin** (probabiliste) et l'**ECPP** (déterministe) représentent l'état de l'art en matière de rapidité pour la vérification de primalité, chacun dans leur catégorie de garantie. Les autres méthodes viennent en complément ou pour des cas particuliers, mais ne détrônent pas ce duo de tête en 2025. [5] [8]

---

[1] [3] [4] [9] [10] [11] [12] 3860-Article Text-21356-1-4-20221002.docx
https://www.jsr.org/hs/index.php/path/article/download/3860/1589

[2] [13] [14] algorithm - Fastest primality test - Stack Overflow
https://stackoverflow.com/questions/4493645/fastest-primality-test

[5] c# - What is the best, most performant algorithm to find all primes up to a given number? - Stack Overflow
https://stackoverflow.com/questions/7991463/what-is-the-best-most-performant-algorithm-to-find-all-primes-up-to-a-given-num

[6] [7] [8] [16] PrimePage Primes: Elliptic Curve Primality Proof
https://t5k.org/top20/page.php?id=27

[15] [17] [18] Primality tests - Algorithms for Competitive Programming
https://cp-algorithms.com/algebra/primality_tests.html

[19] Rabin-Miller Primality Test - Elaboration needed
https://crypto.stackexchange.com/questions/106799/rabin-miller-primality-test-elaboration-needed

[20] [21] [22] [23] arxiv.org
https://arxiv.org/pdf/2311.07048

[24] [25] [26] prediction - Could a neural network detect primes? - Artificial Intelligence Stack Exchange
https://ai.stackexchange.com/questions/3389/could-a-neural-network-detect-primes