Reducing the Number of Qubits in Quantum Factoring

Clémence Chevignard, Pierre-Alain Fouque, André Schrottenloher

Reducing the Number of Qubits in Quantum Factoring

Clémence Chevignard, Pierre-Alain Fouque, and André Schrottenloher

Univ Rennes, Inria, CNRS, IRISA
firstname.lastname@inria.fr

Context. Shor's breakthrough algorithm [13] showed that the problems of factoring and computing Discrete Logarithms could be solved in polynomial time on a quantum computer. Since then, many authors have introduced variants of the algorithm and refined its cost estimates, in order to minimize the requirements in qubits, gate count or circuit depth [2,15,14,8,5,12]. Since Shor's algorithm is considered to be the most relevant application of quantum computers to cryptanalysis, these works also aim at settling the point at which quantum computing architectures could become "cryptographically relevant".

In this paper, we focus on the optimization of space. Consider the Discrete Logarithm (DL) problem in the group $\mathbb{Z}_N^*$, where N is prime. Let us note $n = \log_2 N$. We take a multiplicative generator G. The DL of A is the quantity D such that $A = G^D \bmod N$. It is found by calling Shor's quantum period-finding subroutine on the function $f(x, y) = G^x A^{-y} \bmod N$ defined over $\mathbb{Z}^2$. This subroutine simply calls f in superposition over all $(x, y) \in [0; 2^{m_1} - 1] \times [0; 2^{m_2} - 1]$, performs a QFT, and a measurement (Figure 1). The period $(D, 1)$ is found after some efficient post-processing. Therefore, the number of logical qubits depends on two parameters: the input size $m_1 + m_2$, and the workspace size. Factoring an RSA semiprime N can be reduced to solving a DL instance in $\mathbb{Z}_N^*$ where the DL is of expected size $\frac{1}{2}n$.

The semi-classical Fourier transform [7] intertwines the operations of $f(x, y)$ and the last QFT step, in order to reduce the input register to a single "flying" qubit which is reset and measured multiple times. Due to the space required by modular multiplication circuits, most previous works settled for $2n + \mathcal{O}(1)$ qubits in the workspace register. Zalka proposed a method with $1.5n$ qubits [15].

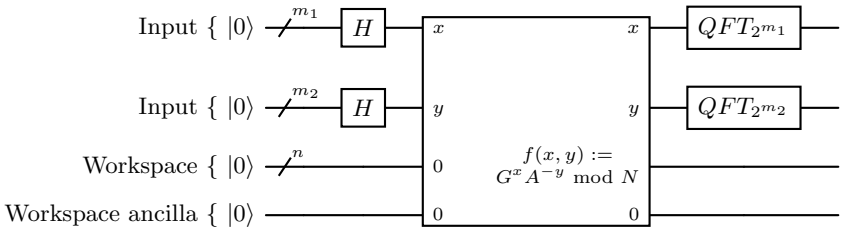

Fig. 1. Shor's quantum DL subroutine.

Gidney and Ekerå [6] used around $3n$ qubits. Typically, an RSA-2048 factoring instance will then require around 6000 qubits.

In this paper, we follow an orthogonal approach: we start by reducing the workspace register, while the input register will become dominating (the semi-classical Fourier transform is not applicable anymore). Our result combines three techniques: 1. the *compression* method, which was first studied by May and Schlieper [10]; 2. a new arithmetic circuit based on Residue Number Systems (RNS); 3. the Ekerå-Håstad algorithm that optimizes the input register size [4,3]. Perhaps surprisingly, we conclude that factoring an RSA modulus of n bits can be done with less than n logical qubits.

Output Compression in Shor's Algorithm. Let r be a small constant. Let h be a good hash function mapping $\mathbb{Z}_N^*$ to $\{0,1\}^r$. In the circuit of Figure 1, let us replace the call to f by a call to $h \circ f$. May and Schlieper showed that under the assumption that h is picked from a universal hash function family, the algorithm outputs almost the same distribution of measurements [10].

In this paper, we use a variant of this compression method, where h is the truncation of f to a small ($r = 22$) number of bits, followed by a random linear mask on $\mathbb{F}_2^r$: $h \circ f(x,y) = \beta \cdot (G^x A^{-y} \bmod N) \bmod 2^r$. Contrary to May and Schlieper's setting, we do not have a universal hashing family. However, we are able to prove a similar result, under the following heuristic.

Heuristic 1. When N is selected at random, the function f behaves like a random (periodic) function $\{0,1\}^{m_1+m_2} \to \mathbb{Z}_N^*$.

This heuristic is quite strong, but also quite similar to those used in cryptanalytic algorithms such as Pollard's rho method [11]. Equipped with this tool, the next step is to actually implement this compression.

Computation of the Truncated Output. The main ingredient of our work is a computation of $(G^x A^{-y} \bmod N) \bmod 2^r$ using only $\mathcal{O}(\log \log N)$ bits of space. Our method relies purely on classical arithmetic, namely Residue Number Systems (RNS), which are commonly used to reduce the depth arithmetic circuits. An RNS represents a large integer by a collection of residues modulo small primes, of $\mathcal{O}(\log \log N)$ bits. Our algorithm computes the reduction modulo 2^r (truncation) by operating on these residues.

Similarly to the usual implementation of Shor's algorithm, we reduce the exponentiation to a *multi-product* of precomputed modular powers: a product of the form $(\prod_i A_i^{z_i}) \bmod N$ where A_i are $\log N$-bit integers (either $G^{2^j} \bmod N$ or $A^{2^j} \bmod N$) and the z_i are the bits of the input. Before reduction, the product $Z := (\prod_i A_i^{z_i})$ has $(m_1 + m_2) \log N = \mathcal{O}\big((\log N)^2\big)$ bits. We use the RNS to represent Z as: $Z = (\sum_i (Z \bmod p_i) M_i) \bmod M$ where the p_i are small primes, M_i and M are large constant integers, and $(Z \bmod p_i)$ can be computed with $\mathcal{O}(\log \log N)$ space. The next steps are *explicit Chinese remaindering*, which allows to compute exactly the quotient in the Euclidean division by M, and *Barrett*

reduction, which allows to re-express the quotient in the Euclidean division by N, as it replaces an operation $\left\lfloor \frac{\cdot}{N} \right\rfloor$ by a multiplication followed by a binary shift.

After these two steps, the result is contained in the high-order bits of a sum of large integers. The final step is to truncate these integers. This introduces some probability of error, due to carry propagation, which can be bounded if these integers behave as random. This is our second heuristic, which we validated experimentally.

We obtain a gate count of $\mathcal{O}\big((\log N)^3\big)$ for factoring, which is comparable to many implementations of Shor's algorithm (although more recent works have reduced this using different period-finding subroutines [12] or improved integer multiplication circuits [9]). The gate count is largely dominated by the computation of RNS residues.

Ekerå-Håstad Method for Optimizing the Input Size The last ingredient of our space optimization is the algorithm of Ekerå and Håstad [4]. This is a variant of Shor's algorithm where the input register is minimized. The algorithm uses multiple runs and recovers the hidden period by a lattice-based post-processing of the measurement results. As an example, when looking for a short discrete logarithm of d bits, the input register can be reduced to $d + o(d)$ bits, even if d is much smaller than $\log N$. This method is also used to factor RSA moduli, since this case is reduced to a DL instance of size $\frac{1}{2}\log_2 N$ in $\mathbb{Z}_N^*$.

Previously the Ekerå-Håstad algorithm was used as an optimization of the gate count and depth, since the input register size could be neglected anyway. However, our method prevents the use of the semiclassical Fourier transform, and the input register dominates the space complexity.

We reuse estimates from Ekerå [3], combined with our compression result, to estimate the input register size and corresponding number of runs. We propose an implementation of the truncated exponentiation circuit, and estimate precisely its gate counts, depth, and number of qubits. This leads to new trade-offs for cryptographic parameters, requiring more computations than the current best optimizations of Shor's algorithm [6], but much less qubits. For RSA-2048, we estimate to require 1730 qubits for the full circuit (incl. 1146 for the input register), $2^{35.54}$ Toffoli gates for a single run, and 40 runs. For comparison, Gidney and Ekerå's logical circuit [6] needs about 6144 qubits and $2^{31.26}$ Toffoli gates, but requires a single run.

The reduction in the number of qubits is stronger for DL instances in safe-prime groups. Indeed, it is possible to use a large group $\mathbb{Z}_N^*$, but short exponents (in a smaller subgroup) for the DL. The large ambient group thwarts attacks based on the General Number Field Sieve, while the short exponent suffices to resist cycle-finding attacks. For example, a 2048-bit group and a DL of size 224 bits provide 112 bits of classical security [1]. Here we estimate that our algorithm would use 684 qubits only, $2^{32.15}$ Toffoli gates, and need 20 measurement results. We stress that these are only preliminary estimates, which could benefit from further circuit optimizations.

References

1. Barker, E., Chen, L., Roginsky, A., Vassilev, A., Davis, R.: Recommendation for pair-wise key-establishment schemes using discrete logarithm cryptography. Tech. rep., NIST (April 2018). https://doi.org/10.6028/NIST.SP.800-56Ar3
2. Beauregard, S.: Circuit for Shor's algorithm using $2n + 3$ qubits. arXiv preprint quant-ph/0205095 (2002)
3. Ekerå, M.: On post-processing in the quantum algorithm for computing short discrete logarithms. Des. Codes Cryptogr. **88**(11), 2313–2335 (2020). https://doi.org/10.1007/S10623-020-00783-2
4. Ekerå, M., Håstad, J.: Quantum algorithms for computing short discrete logarithms and factoring RSA integers. In: PQCrypto. Lecture Notes in Computer Science, vol. 10346, pp. 347–363. Springer (2017). https://doi.org/10.1007/978-3-319-59879-6_20
5. Gidney, C.: Factoring with $n+2$ clean qubits and $n-1$ dirty qubits. arXiv preprint arXiv:1706.07884 (2017)
6. Gidney, C., Ekerå, M.: How to factor 2048 bit rsa integers in 8 hours using 20 million noisy qubits. Quantum **5**, 433 (2021). https://doi.org/10.22331/q-2021-04-15-433
7. Griffiths, R.B., Niu, C.S.: Semiclassical fourier transform for quantum computation. Physical Review Letters **76**(17), 3228 (1996). https://doi.org/10.1103/PhysRevLett.76.3228
8. Häner, T., Roetteler, M., Svore, K.M.: Factoring using $2n + 2$ qubits with toffoli based modular multiplication. arXiv preprint arXiv:1611.07995 (2016)
9. Kahanamoku-Meyer, G.D., Yao, N.Y.: Fast quantum integer multiplication with zero ancillas (2024)
10. May, A., Schlieper, L.: Quantum period finding is compression robust. IACR Trans. Symmetric Cryptol. **2022**(1), 183–211 (2022). https://doi.org/10.46586/TOSC.V2022.I1.183-211
11. Pollard, J.M.: A monte carlo method for factorization. BIT Numerical Mathematics **15**, 331–334 (1975). https://doi.org/10.1007/bf01933667
12. Regev, O.: An efficient quantum factoring algorithm. CoRR **abs/2308.06572** (2023)
13. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM J. Comput. **26**(5), 1484–1509 (1997). https://doi.org/10.1137/S0097539795293172
14. Takahashi, Y., Kunihiro, N.: A quantum circuit for Shor's factoring algorithm using $2n + 2$ qubits. Quantum Information & Computation **6**(2), 184–192 (2006)
15. Zalka, C.: Shor's algorithm with fewer (pure) qubits. arXiv preprint quant-ph/0601097 (2006)