CSCI 599 - Term paper
The hidden subgroup problem

Sumeet Shirgure
USC - 9330099198

Fall 2021

1 Introduction

In this paper, we will study the framework of the hidden subgroup problem, while also looking at some generalizations that are relevant to cryptography. A good reason to study it is to understand something that unifies some of the fastest algorithms for problems solved by quantum computers alone. The field of cryptography experienced a tremendous change when Shor introduced algorithms [20] for integer factorization and discrete logarithms in **BQP**. If a quantum computer with enough qubits could operate without failure due to noise or decoherence, it could be used to break public-key cryptosystems such as the RSA scheme, and Diffie-Hellman key exchanges that rely on the hardness of the discrete logarithm problem over finite fields. Both of these problems are special cases of the (abelian) hidden subgroup problem. While search problems like the ones solved by Grover's algorithm may have no structure, the exponential speedups achieved by quantum computers are known to be instances of *promise* problems like the HSP, where some structure in the input is promised.

There is another motivation to study HSPs (especially non-abelian ones). In cryptography the computational hardness of some problem is assumed / believed, and sometimes proved. It is essential to prove the equivalence of a cryptosystem and a known hard mathematical problem. Such proofs are called "security reductions". Often the hardness is not proved completely, and only reduced to some widely believed conjecture such as $\mathbf{P} \neq \mathbf{NP}$ or the existence of a one-way function (which incidentally implies the former.) For this reason it is natural to ask which HSPs are hard, and why. Any hardness results or lower bounds on such problems might give us a better picture of the complexity classes they reside in. Moreover they raise our confidence of the security of cryptosystems that could be based on them.

Outline Section 2 introduces the hidden subgroup problem. Section 3 gives some ideas on solving abelian HSPs in general. Specific instances of abelian HSPs relevant to cryptography are also discussed. Section 4 discusses some

non-abelian HSPs, some results on hardness and also introduces lattice based cryptosystems. Finally we conclude with some discussions in section 5. Required concepts, theorems and proofs are presented in the appendix.

2 The hidden subgroup problem

Given a group G, a subroup $H < G$, and a set X, we say a function $f : G \to X$ hides the subgroup H, if f is constant on cosets of H, while it is different between different cosets of H. Such a function f is also called *H-periodic*. The hidden subgroup problem is a *promise* problem where we are given a function f that hides some $H < G$. f is available to us through an oracle that uses $O(log|G| + log|X|)$ bits. Using only the information gained from evaluations of f, we must determine a generating set of H. Note that a small ($O(log|G|)$ sized) generating set always exists. And since each element of G can be labeled using $log|G|$ bits, the output complexity is polynomial.

If the group G is abelian, there exists a **BQP** algorithm to solve the corresponding HSP. On the other hand, the existence of effecient quantum algorithms for HSPs of certain non-abelian groups would solve problems like graph isomorphism and some shortest vector problems on lattices in probabilistic polynomial time on a quantum computer.

3 Solving abelian HSPs

Here we will focus on finite abelian groups. Since any finite abelian group G is isomorphic to a direct sum of prime-power order cyclic groups, and each element is in a conjugacy class of its own - complex-valued functions f over G have a simple form of Fourier transform. And, more importantly, that transform can be implemented effeciently on a quantum computer.

The fundamental reason why quantum computers gain exponential speedup over classical computers on abelian HSPs seems to be this characterization of abelian groups, and the effeciency of Fourier transforms of functions on them. That, coupled with the fact that Fourier transforms have a desirable shift-invariance property, allows us to compute the hidden subgroups effeciently.

The standard (also called the *"Fourier sampling"* [3]) method of solving abelian HSPs starts by creating a uniform superposition state. (All normalization factors are omitted):

$$|\psi\rangle = \sum_{g \in G} |g\rangle \oplus |0\rangle \tag{1}$$

Next we evaluate the function reversibly via the oracle and store the result in the second register.

$$|\psi\rangle = \sum_{g \in G} |g\rangle \oplus |f(g)\rangle \tag{2}$$

The above operation entangles the two registers. We now measure the second register, which disentangles it from the first and puts it in a uniform superpo-

sition of one of f's level sets. E.g if we measure the value $f(cH)$, it puts the system in a superposition of all elements in a random coset $|cH\rangle$

$$|\psi\rangle = \sum_{h \in H} |ch\rangle \oplus |f(ch)\rangle = |cH\rangle \oplus |f(cH)\rangle \tag{3}$$

where c is some coset representative. For an abelian group (or in general, for a normal subgroup $H \triangleleft G$,) cH is a member of the factor group G/H.

Finally we perform an inverse Fourier transform on the resulting state of the first register and then measure in computational basis. For abelian groups, this gives us a member of the factor group G/H. Repeated experiments give us random generators of G/H, thus giving us H in polynomial time.

For exact details refer to [14]. Of course, implementing Fourier transforms over arbitrary groups is interesting in itself. The first algorithm approximating QFT over arbitrary cyclic groups was given by Kitaev [11].

The order finding subroutine of Shor's algorithm can be cast as a hidden subgroup problem of the group $\mathbb{Z}_p^*$ for some $p = \Theta(n)$, where n is the number we need to factor, and the hidden subgroup is the one generated multiplicatively by a random $a \in \mathbb{Z}_n$ of the algorithm.

Discrete logarithms for finite groups can also be cast as an HSP. The DLP is defined as follows - given a group $\mathbb{Z}_p^*$ and a generator g, for some element $x = g^r$, determine its order r. The HSP is then constructed by letting $\mathbb{Z}_p \times \mathbb{Z}_p$ be the base group, and the (computable) homomorphism $f : \mathbb{Z}_p \times \mathbb{Z}_p \to G; f(a,b) = g^a x^{-b}$ acts as the hiding oracle. By the homomorphism theorem, f's kernel $kerf = \{(a,b) : f(a,b) = 1\}$ forms a normal subgroup of $\mathbb{Z}_p \times \mathbb{Z}_p$. f hides that kernel, which is generated by $(r, 1)$.

A similar treatment of some specific elliptic curve groups results in an effecient solution to their elliptic curve discrete logarithm problem (ECDLP) [18]. The hardness of which is relied upon by corresponding elliptic curve cryptography schemes.

4 Non abelian HSPs

For non-abelian groups, since the group law need not commute, the left cosets $gH = \{gh : h \in H\}$ are not necessarily the same as the right cosets $Hg = \{hg : h \in H\}$ for some subgroup $H < G$. This difference seems to be the key in solubility by Fourier transforms, as the subgroups for which this is the case - i.e normal subgroups, can be effeciently computed even for non-abelian groups [8]. There exists a way to generalize Fourier transforms to non-abelian groups as well, but is not as simple. (See [8] and [9].)

However, the general case of the non-abelian HSP doesn't yield to this process. The difficulty doesn't appear to lie in the inability to perform a Fourier transform, but rather in the interpretation of the resulting state. Specifically, starting from $m = \Theta(log|G|)$ random coset states $|g_1 H\rangle, ..., |g_m H\rangle$ (like the one in equation 3), there exists a quantum observable that gives enough information to determine all possible subgroups H hidden by f. However, it is not known

how to *effeciently* implement such an observable. The algorithm given in [5] uses only m oracle queries, but needs $O(|G|)$ measurements. Note that the algorithm mentioned doesn't even need to be able to perform a QFT over G.

There exist interesting and important problems reducible to finding non-abelian hidden subgroups. One of them being the graph isomorphism (GI) problem, defined as follows : given two connected graphs (with same number of vertices in each), decide if they're isomorphic. This problem has many important applications in a lot of areas ranging from chemistry to electronics to computer science. In many ways, GI is similar to factoring - both, having succint certificates, are in **NP**. Neither is known to be **NP**-hard, nor to be in **P**, and are categorized as **NPI** (NP-intermediate), under the assumption that $\mathbf{P} \neq \mathbf{NP}$ [13]. However, factoring is in **BQP**, while GI isn't known to be. GI can be reduced to a non-abelian HSP, as finding the subgroup of a permutation group on the vertices of a graph. [16], [15] and [7] give a negative result about the effecient solutions of the general HSP over the symmetric group, and also the special cases relevant to solving GI, by Fourier sampling methods like the one described in section 3. They do this by giving an information theoretic lower bound on the number of coset states needed to determine a hidden subgroup.

Even though GI hardness is not the best candidate for making cryptosystems, it does makes a fun little appearance in an example of a zero knowledge proof in [4]. (See [6] for zero knowledge proofs.) To that end, some prime candidates for post-quantum cryptography happen to be lattice - based cryptosystems. A lattice $L \subset \mathbb{R}^n$ is the set of all linear integer combinations of some basis vectors $\mathbf{b_1}, \mathbf{b_2}, ..., \mathbf{b_n} \in \mathbb{R}^n$. For lattices with non-unique basis sets, optimization problems like the shortest vector problem (SVP) or its approximations are considered hard ([1], [2]) even for quantum computers. SVP asks the shortest vector in L (i.e the one closest to the origin) given a basis set. The first connection between lattice problems and quantum computation was given in [17], by reducing a promise problem - the $f(n)$-unique SVP, with the promise that the shortest vector is shorter by a factor of at least $f(n)$ from all other non-parallel vectors in the lattice; to a hidden subgroup problem over the dihedral group D_n. Even though D_n is much simpler than the symmetric group S_n, there is no known effecient algorithm to solve the corresponding HSP. As noted in the paper, the best algorithm known is quasipolynomial in n [12].

5 Conclusion

In summary, the Fourier sampling method seems to be a powerful tool, at least in theory, to solve problems reducible to the abelian hidden subgroup problem. The promise of the oracle function having the subgroup hiding property is also key in obtaining a significant speedup over classical algorithms. However, that power also seems to diminish when the underlying object under consideration becomes more and more unstructured. Naturally, the lesser the amount of prior knowledge we have, the greater is the time we will need to solve the problem at hand. So it shouldn't come as a surprise that we don't know how to solve for

the general non-abelian case as effeciently as we would like to.

On the other end, this difficulty in determining non-normal hidden subgroups is good news for cryptographers. The discussed attempt of trying to break lattice-based cryptosystems relies on introducing a promise problem, and even the weakened version of the problem is not effeciently solvable by quantum computers. Moreover, the impossibility results discussed in section 4 only increase our confidence in the post-quantum security of lattice based cryptosystems. Any attempt at solving non-abelian HSPs must necessarily involve some means other than Fourier sampling. This raises the question, do such methods exist? Perhaps the most exciting use of information theory would be to provide bounds on the computational power of machines. One great example is [7]. What other ways could information theory be used to bound computational effeciency in solving certain problems?

For potential **NPI** problems like GI, there still hasn't been any progress towards solving it faster than classical computers. The impossibility results mentioned earlier are also applicable to solving GI. An interesting direction in this area would be to consider special cases of graphs as the promise problems, and research if quantum computers provide any advantage. GI is already classically solved effeciently for a number of special graphs (e.g graphs with bounded parameters like treewidth or genus.) Is there some class of graphs where quantum algorithms outperform classical ones for deciding the corresponding graph isomorphism? Any progress in this direction will shed more light on the class **NPI**. Because from the state of things, there is good evidence that **BQP** separates problems within it. Hardness results for GI, or even other problems suspected to be in **NPI**, only strengthen this belief.

References

[1] Ajtai, M. Generating hard instances of lattice problems (extended abstract). In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing* (New York, NY, USA, 1996), STOC '96, Association for Computing Machinery, p. 99–108.

[2] Ajtai, M. The shortest vector problem in l2 is np-hard for randomized reductions (extended abstract). In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing* (New York, NY, USA, 1998), STOC '98, Association for Computing Machinery, p. 10–19.

[3] Bernstein, E., and Vazirani, U. Quantum complexity theory. *SIAM J. Comput. 26*, 5 (oct 1997), 1411–1473.

[4] Blum, M. How to prove a theorem so no one else can claim it. In *In: Proceedings of the International Congress of Mathematicians* (1987), pp. 1444–1451.

[5] Ettinger, M., Høyer, P., and Knill, E. The quantum query complexity of the hidden subgroup problem is polynomial. *Information Processing Letters 91*, 1 (Jul 2004), 43–48.

[6] Goldwasser, S., Micali, S., and Rackoff, C. The knowledge complexity of interactive proof systems, 1989.

[7] Hallgren, S., Moore, C., Rötteler, M., Russell, A., and Sen, P. Limitations of quantum coset states for graph isomorphism. *J. ACM 57*, 6 (nov 2010).

[8] Hallgren, S., Russell, A., and Ta-shma, A. The hidden subgroup problem and quantum computation using group representations. *SIAM Journal on Computing 32* (01 2002).

[9] Jozsa, R. Quantum factoring, discrete logarithms, and the hidden subgroup problem. *IEEE MultiMedia 3*, 2 (mar 1996), 34–43.

[10] Judson, T. W. *Abstract Algebra: Theory and Applications*. online, 2021.

[11] Kitaev, A. Y. Quantum measurements and the abelian stabilizer problem. *Electron. Colloquium Comput. Complex.*, 3 (1996).

[12] Kuperberg, G. A subexponential-time quantum algorithm for the dihedral hidden subgroup problem. *SIAM J. Comput. 35*, 1 (jul 2005), 170–188.

[13] Ladner, R. E. On the structure of polynomial time reducibility. *J. ACM 22*, 1 (jan 1975), 155–171.

[14] Lomont, C. The hidden subgroup problem - review and open problems. *arXiv preprint quant-ph/0411037* (2004).

[15] Moore, C., and Russell, A. The symmetric group defies strong fourier sampling: Part ii. *ArXiv abs/quant-ph/0501066* (2005).

[16] Moore, C., Russell, A., and Schulman, L. The symmetric group defies strong fourier sampling. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)* (2005), pp. 479–488.

[17] Regev, O. Quantum computation and lattice problems. *SIAM J. Comput. 33*, 3 (mar 2004), 738–760.

[18] Roetteler, M., Naehrig, M., Svore, K. M., and Lauter, K. Quantum resource estimates for computing elliptic curve discrete logarithms, 2017.

[19] Serre, J.-P. *Linear representations of finite groups*. Springer-Verlag, New York, 1977.

[20] Shor, P. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science* (1994), pp. 124–134.

[21] Woit, P. *Quantum theory, groups and representations : an introduction.* Springer, Cham, Switzerland, 2017.

Appendices

A Group theory

A good introduction to abstract algebra and group theory is [10].

A group $(G, \circ)$ is a set G and a binary operation $\circ : G \times G \to G$ on G with the following properties :

- *closure* : $\forall a, b \in G, a \circ b \in G$
- *associativity* : $\forall a, b, c \in G, (a \circ b) \circ c = a \circ (b \circ c)$
- *identity* : there exists a unique $e \in G : \forall a, a \circ e = e \circ a = e$
- *inverses* : $\forall a \exists a^{-1} : a \circ a^{-1} = a^{-1} \circ a = e$

We usually omit the $\circ$ and write $a \circ b$ as ab. We also omit the group operation, and refer to the group by the its ground set G, when it's understood. The size of the set G is called the order of the group, also denoted $|G|$.

If the group operation commutes, i.e $\forall a, b \in G, ab = ba$, G is called an abelian group. A subgroup H of G is a subset of G which forms a group under the same operation. This is denoted by $H < G$.

For a finite group G, if we pick any element $g \in G$, the set $\langle g \rangle = \{g^k : k \in \mathbb{N}\}$ must be finite. The smallest $r \in \mathbb{N}$ such that $g^r = e$ is called the order of g. $r = |\langle g \rangle|$; $\langle g \rangle = \{e, g, g^2, ..., g^{r-1}\}$ is called the group generated by g. In general, a group G is said to be generated by elements, called "generators" $g_1, g_2, ..., g_m$ if every element $g \in G$ can be expressed as a product of (possibly repeated) generators. For every group G, there exists a set of $m = O(log(|G|))$ sized generators. A cyclic group is one that can be generated by a single (not necessarily unique) generator. An important example of cyclic groups are $\mathbb{Z}_n = \{0, 1, ..., n-1\}$ under addition modulo n.

For $H < G$, the left coset of H in G with the coset representative g is defined to be the set $gH := \{gh : h \in H\}$. Analgously, the right coset is defined $Hg := \{hg : h \in G\}$. Any element in gH can act as its coset representative. For additive groups like $\mathbb{Z}_n$, it's conventional to write cosets as $g + H, g \in \mathbb{Z}_n$. An alternative construction of cosets is from the equivalence classes of the relation on G given by $(a, b) \iff ab^{-1} \in H$; i.e $a, b \in gH \iff ab^{-1} \in H$. And since the map $h \to gh$ is a bijection from $H \to gH$, $|H| = |gH|$. And since the equivalence classes partition G, there are exactly $|G|/|H|$ cosets of H in G. This implies $|H|$ divides $|G|$, and the order of every element also divides $|G|$. (Known as Lagrange's theorem.) E.g, the cosets of $\{0, 4, 8\} = H < G = \mathbb{Z}_{12}$ are $H, 1 + H = \{1, 5, 9\}, 2 + H$ and $3 + H$.

For $g_1, g_2 \in G$, the conjugate of g_2 w.r.t g_1 is the element $g_1^{-1} g_2 g_1$. The conjugacy relation also forms equivalence classes in G - define the conjugacy class G_x of $x \in G$ as $G_x := \{g^{-1}xg : g \in G\}$.

For $N < G$, N is a normal subgroup, denoted $N \triangleleft G$, if $\forall g : g^{-1}Ng = N$. Equivalently, $\forall g \in G : gN = Ng$; i.e the left and right cosets coincide. If $N \triangleleft G$, the cosets of N form a group called the factor group, denoted G/N, given by $(aN)(bN) = (ab)N$. E.g, consider the dihedral group D_n, generated by two elements r and f, satisfying the relations $r^n = e$, $f^2 = e$, $frf = r^{-1}$. The group of rotations $R_n = \langle r \rangle$ is a normal subgroup $R_n \triangleleft D_n$. For abelian groups, all subgroups are normal, and each conjugacy class has only one element.

B Morphisms and matrix groups

For any two groups $(G, \circ_G)$ and $(H, \circ_H)$, their cartesian product $K := G \times H = \{(g, h) : g \in G, h \in H\}$ forms a group under the operation $\circ_K : K \times K \to K$ given by $(g_1, h_1) \circ_K (g_2, h_2) = (g_1 \circ_G g_2, h_1 \circ_H h_2)$. K is called the external direct product, or the direct sum of G and H.

The most effective way of studying relations between two groups is by maps that preserve structure (morphisms). Formally, a group homomorphism is a function $\phi : G \to H$ such that $\forall a, b \in G$, $\phi(a \circ_G b) = \phi(a) \circ_H \phi(b)$. When the domain of symbols is understood, we omit $\circ$, and write $\phi(ab) = \phi(a)\phi(b)$. Homomorphisms preserve identities and inverses : $\phi(e_G) = e_H$, $\phi(a^{-1}) = \phi(a)^{-1}$.

If the group homomorphism is a bijection (i.e a one-one and onto map), it's called an isomorphism. If an isomorphism $\phi : G \to H$ exists, G is said to be isomorphic to H, denoted $G \cong H$. Group isomorphism is the notion used to distinguish groups, as isomorphic groups differ only in labels, and are the same for all practical purposes. An important idea is that all cyclic groups of order n are isomorphic to $\mathbb{Z}_n$. Furthermore, if a group has prime order p, by Lagrange's theorem, it must be $\mathbb{Z}_p$. We can also show that $\mathbb{Z}_{mn} \cong \mathbb{Z}_m \times \mathbb{Z}_n$.

A fundamental theorem for abelian groups states that any such group G is isomorphic to a direct sum of cyclic groups with prime power order. I.e : $G \cong \mathbb{Z}_{p_1} \times \mathbb{Z}_{p_2} \times ... \mathbb{Z}_{p_k}$ where p_is are (not necessarily distinct) prime powers. E.g $\mathbb{Z}_2 \times \mathbb{Z}_2 \times \mathbb{Z}_3$ and $\mathbb{Z}_4 \times \mathbb{Z}_3$ are the only two abelian groups of order 12.

A homomorphism from an object onto itself is called an endomorphism. Analgously, an isomorphism from an object to itself is an automorphism. The most familiar examples of morphisms in linear algebra are matrix groups : an endomorphism on an n dimensional vector space V is a linear transform $L : V \to V$. The set of all such endomorphisms is denoted $End(V)$. Note that $End(V)$ forms a monoid (group without inverses) under functional composition; and under a choice of basis, $End(V)$ is the set of all n by n matrices with elements from the field of V (usually $\mathbb{R}$ or $\mathbb{C}$). Similarly, $Aut(V)$ is the group of all automorphisms on an n dimensional vector space V under functional composition. (Since bijections guarantee inverses.) Under a choice of basis, $Aut(V)$ is isomorphic to the set of all n by n invertible matrices. This group is called the general linear group, denoted $GL(V)$. For complex valued vector

spaces with fixed basis, the matrix group is denoted $GL_n(\mathbb{C})$

Isomorphisms and automorphisms can be defined intuitively for other objects like graphs as well - two graphs G and H are isomorphic if there is a bijection (graph isomorphism) $\phi : V[G] \to V[H]$ such that $(u, v) \in E[G] \Leftrightarrow (\phi(u), \phi(v)) \in E[H]$. The group of all automorphisms of a graph G is denoted $Aut(G)$. It naturally forms a subgroup of all bijections on G, called the symmetric group, denoted $Sym(G)$.

The kernel of a group homomorphism $\phi : G \to H$ is defined as the set mapped to identity : $\ker \phi := \{g \in G : \phi(g) = e_H\}$. One of the fundamental theorems of group homomorphisms is that $\ker \phi \triangleleft G$.

C Representation theory

Representation theory aims to study abstract algebraic structures by representing their elements as linear transformations of vector spaces. A good introduction to the topic is [19]. Representation theory is also deeply related to quantum mechanics itself [21].

The following discussion borrows content and notation from section 2 of [8].

A fundamental theorem from representation theory for finite groups is that every group G has exactly r inequivalent irreducible representations, where r is the number of conjugacy classes of G. For abelian groups, all irreducible representations are 1 dimensional, and there exist exactly $|G|$ such representations.

For finite abelian groups $G \cong \Pi_{j=1}^{n} \mathbb{Z}_{p_j}$, these representations take a simple form, where each group element $x \in G$ corresponds to a tuple $(x_1, x_2, ..., x_n)$, and the representation parametrized by $h \in G$ can be defined as $\rho_h(x) = exp\left\{2\pi i \sum_{j=1}^{n} \frac{x_j h_j}{p_j}\right\}$

It is instructive to note that the Fourier transform of a subgroup hiding function $f : G \to \mathbb{C}$ over an abelian group has the following property:

$$\hat{f}(\rho_h) = \frac{1}{\sqrt{|G|}} \sum_{s \in G} \rho_h(s) f(s) \tag{4}$$

$$\hat{f}(\rho_h) = \frac{1}{\sqrt{|G|}} \sum_{s_1 \in \mathbb{Z}_{p_1}} ... \sum_{s_n \in \mathbb{Z}_{p_n}} \rho_h(s) f(s_1, s_2, ..., s_n) \tag{5}$$

Without losing generality, let's assume the subgroup H hidden by f is generated by $s_1, s_2, ..., s_k; k \leq n$. This means that f is independent of $s_i \forall i \leq k$.

$$\hat{f}(\rho_h) = \frac{1}{\sqrt{|G|}} \sum_{s_{k+1} \in \mathbb{Z}_{p_{k+1}}} ... \sum_{s_n \in \mathbb{Z}_{pn}} f(s_{k+1}, ..., s_n) \sum_{s_1 \in \mathbb{Z}_{p_1}} ... \sum_{s_k \in \mathbb{Z}_{p_k}} \rho_h(s) \tag{6}$$

The "coefficient" of $f(s)$ for some $s \in G$ in the above equation is hence

$$\frac{1}{\sqrt{|G|}} \sum_{s_1 \in \mathbb{Z}_{p_1}} ... \sum_{s_k \in \mathbb{Z}_{p_k}} \rho_h(s) \tag{7}$$

$$= \frac{e^{2\pi i\theta}}{\sqrt{|G|}} \sum_{s_1 \in \mathbb{Z}_{p_1}} ... \sum_{s_k \in \mathbb{Z}_{p_k}} exp\left\{2\pi i \sum_{j=1}^{k} \frac{s_j h_j}{p_j}\right\} \tag{8}$$

where $\theta = \sum_{j=k+1}^{n} \frac{s_j h_j}{p_j}$.

$$... = \frac{e^{2\pi i\theta}}{\sqrt{|G|}} \sum_{s_1 \in \mathbb{Z}_{p_1}} ... \sum_{s_k \in \mathbb{Z}_{p_k}} \prod_{j=1}^{k} exp\left\{2\pi i \frac{s_j h_j}{p_j}\right\} \tag{9}$$

$$= \frac{e^{2\pi i\theta}}{\sqrt{|G|}} \prod_{j=1}^{k} \sum_{s_j \in \mathbb{Z}_{p_j}} exp\left\{2\pi i \frac{s_j h_j}{p_j}\right\} \tag{10}$$

$$= \frac{e^{2\pi i\theta}}{\sqrt{|G|}} \prod_{j=1}^{k} (p_j \delta_{h_j 0}) \tag{11}$$

$$= \rho_h(s)|H|\delta_{h\in G/H}/\sqrt{|G|} \tag{12}$$

$$\Rightarrow \hat{f}(\rho_h) = \frac{1}{\sqrt{|G|}} \sum_{s\in G} |H|\delta_{h\in G/H}\rho_h(s)f(s) \tag{13}$$

The inverse transform now becomes :

$$\Rightarrow f(s) = \frac{1}{\sqrt{|G|}} \sum_{h\in G/H} \hat{f}(\rho_h)\rho_h(s^{-1}) \tag{14}$$

(i.e sums over just the elements in the factor group.)