HIMMO: A Lightweight Collusion-Resistant Key Predistribution Scheme

Oscar García-Morchón[1], Domingo Gómez-Pérez[2], Jaime Gutiérrez[2], Ronald Rietman[2], Berry Schoenmakers[3], and Ludo Tolhuizen[1]

[1] Philips Group Innovation, Research, Eindhoven, The Netherlands
oscar.garcia,ronald.rietman,ludo.tolhuizen@philips.com
[2] University of Cantabria, Santander, Spain
domingo.gomez,jaime.gutierrez@unican.es
[3] Eindhoven Universtiy of Technology, Eindhoven, The Netherlands
berry@win.tue.nl

Abstract. In this paper we introduce HIMMO as a truly practical and lightweight collusion-resistant key predistribution scheme. The scheme is reminiscent of Blundo et al's elegant key predistribution scheme, in which the master key is a symmetric bivariate polynomial over a finite field, and a unique common key is defined for every pair of nodes as the evaluation of the polynomial at the finite field elements associated with the nodes. Unlike Blundo et al's scheme, however, which completely breaks down once the number of colluding nodes exceeds the degree of the polynomial, the new scheme is designed to tolerate any number of colluding nodes.

Key establishment in HIMMO amounts to the evaluation of a single low-degree univariate polynomial involving reasonably sized numbers, thus exhibiting excellent performance even for constrained devices such as 8-bit CPUs, as we demonstrate. On top of this, the scheme is very versatile, as it not only supports implicit authentication of the nodes like any key predistribution scheme, but also supports identity-based key predistribution in a natural and efficient way. The latter property derives from the fact that HIMMO supports long node identifiers at a reasonable cost, allowing outputs of a collision-resistant hash function to be used as node identifiers. Moreover, HIMMO allows for a transparent way to split the master key between multiple parties.

The new scheme is superior to any of the existing alternatives due to the intricate way it combines the use of multiple symmetric bivariate polynomials evaluated over "different" finite rings. We have extensively analyzed the security of HIMMO against two attacks. For these attacks, we have identified the Hiding Information (HI) problem and the Mixing Modular Operations (MMO) problem as the underlying problems. These problems are closely related to some well-defined lattice problems, and therefore the best attacks on HIMMO are dependent on lattice-basis reduction. Based on these connections, we propose concrete values for all relevant parameters, for which we conjecture that the scheme is secure.

Keywords: Key predistribution scheme, collusion attack, identity, lattice analysis

1 Introduction

Background Efficient and practical pairwise key establishment is of extreme importance for industrial deployment of large networks of resource-constrained devices, such as wireless sensors. The nodes in these networks are severely limited with respect to computational power, energy, and bandwidth. In this paper we propose an innovative method for pairwise key establishment. We do so by addressing a longstanding open problem regarding the existence of key predistribution schemes, as introduced by Matsumoto and Imai [17], which are both highly secure and highly efficient. Here, highly secure means that large collusions of corrupted nodes are tolerated, and highly efficient means that the running time for key establishment takes a fraction of a second only, even for very constrained devices such as 8-bit CPUs, and that the memory footprint is low.

The starting point for our work is Blundo et al.'s elegant key predistribution scheme [5]. Key predistribution schemes allow for the establishment of pairwise keys in a large network of nodes, where each node uses its so-called keying material as secret input [17]. In Blundo et al.'s scheme, a master key consisting of a symmetric bivariate polynomial over a finite field is generated by a trusted party, and each node's keying material consists of the univariate polynomial obtained as the evaluation of the master key at a field element associated with the node.

Key establishment in Blundo et al.'s scheme is very efficient, as it amounts to the evaluation of a polynomial over a finite field. The scheme is secure against collusions whose size does not exceed the degree of the polynomial. For larger collusions, however, the security of Blundo et al.'s scheme breaks down completely.

The challenge is to find a key predistribution scheme that achieves efficiency comparable to Blundo et al.'s scheme but with much better security, basically tolerating collusions of practically any size. The potential of such an efficient and collusion-resistant scheme is huge, as it enables secure communication in large networks of wireless sensors, for example. Moreover, key predistribution schemes naturally have benefits such as implicit authentication (ruling out man-in-the-middle attacks) and identity-based modes (avoiding the need for public keys).

Related work We motivate our approach by discussing related work covering three alternative approaches to key establishment.

The first approach is to use any Diffie-Hellman based scheme, involving a one-way function defined for a discrete log setting (including elliptic curve cryptography). The main drawback of these schemes is simply that evaluation of the one-way function is too costly, and this drawback extends to schemes involving pairings. Some relevant examples are the schemes and related constructions in [6,9,10,11,13,20,22].

The second approach is to use a simpler one-way function. For instance, [16] proposes a key exchange protocol based on the NTRU one-way function. Basically, two nodes establish a common key by exchanging NTRU encryptions of their private keys, which is computationally efficient. However, several serious drawbacks remain. In the first place, there is no protection against man-in-the-middle attacks as the corresponding public keys of the nodes are not certified. Secondly, the communication complexity of such a key establishment protocol is high, as it involves an exchange of public key encryptions; hence the protocol is too costly for resource-constrained devices. And, finally such a scheme lacks identity-based modes.

The third approach is to actually build a key predistribution scheme. All pairwise keys are determined by a master key, and one gets all the benefits mentioned above. Clearly, the main challenge is to achieve collusion resistance. In the literature there has only been one attempt at constructing an efficient scheme with collusion-resistance against arbitrary collusions [24]. However, the collusion resistance claims turned out to be flawed [2].

Contributions The HIMMO scheme introduced in this paper is the first efficient key predistribution scheme tolerating large collusions of corrupted nodes. Key establishment between two nodes amounts to the evaluation of a single low-degree univariate polynomial for each node. The numbers involved are reasonably sized, allowing for excellent performance even on constrained devices such as 8-bit CPUs. The performance of the scheme is thus comparable to the performance of an NTRU-based scheme, except that HIMMO requires a minimal amount of information exchange only (taking full advantage of the fact that pairwise keys are predetermined). The only information that needs to be exchanged serves to reconcile the keys computed by both nodes, which in general will differ slightly. Allowing a small discrepancy between the keys computed by the respective nodes turns out be an important degree of freedom, giving us just enough leeway to find a successful solution.

Being a key predistribution scheme, HIMMO automatically provides implicit authentication of the nodes, hence protection against man-in-the-middle attacks. Furthermore, the parameters can be set such that key establishment becomes identity-based. Identities may be bit strings of arbitrary length. The identities are simply hashed to a relatively small range of identifiers, for which the trusted party holding the master key may generate key material.

We have extensively analyzed the security of our scheme. In particular, we study two attacks against HIMMO. We have identified two hard problems, namely the Hiding Information (HI) problem and the Mixing Modular Operations (MMO) problem that are relevant for those attacks. These problems are closely related to some well-defined lattice problems (see [18,12], respectively), and therefore our best attacks on HIMMO are dependent on lattice-basis reduction. We have formulated the relevant lattices, and we have performed numerous experiments to estimate the complexity of solving the (approximate) closest vector problem for these lattices. Based on our analysis we propose concrete values for all relevant parameters, for which we conjecture that the scheme is secure.

Finally, as a bonus we note that HIMMO is resistant to quantum computing as the cryptanalysis is entirely lattice-based.

Roadmap The paper is organized as follows. In Section 2, we describe key predistribution schemes and provide some basic examples. Section 3 introduces the HIMMO scheme, followed by a brief performance analysis in Section 4. In Section 5, we discuss the security model, describe two attacks, review the HI and MMO problems and security assumptions, also showing connection between the HI and MMO problems and the structure of the keying material and keys. In Section 6, we describe experimental results supporting our security analysis of the HI problem, and present parameters for which we consider HIMMO to be secure. Section 7 details further HIMMO-based schemes. In Section 8, we draw conclusions and indicate directions for further research. In the appendix, we validate the HIMMO scheme in that we show that the generated keys indeed are pairwise the same.

2 Key Predistribution Schemes

Key predistribution schemes have been introduced by Matsumoto and Imai [17], generalizing earlier work of Blom [4]. In a key predistribution scheme, a trusted party provides nodes in a system with information enabling any pair of nodes to establish a common key. A key predistribution scheme comprises three components:

– A **setup algorithm**, executed by the trusted party, that on input of a security parameter κ generates system parameters σ and secret root keying material R.
– A **keying material extraction algorithm**, executed by the trusted party, which on input R, σ and a node identifier ξ generates secret keying material G_ξ.
– A **key establishment protocol** applied by two nodes ξ and η for generating a pre-determined key $K(\xi, \eta)$, using ξ, η and σ as common input, in which ξ and η use their secret keying material G_ξ and G_η, respectively.

The pre-determined key $K(\xi, \eta)$ is the same for all executions of the key establishment protocol, and may depend on which node initiates the key establishment protocol, that is, $K(\xi, \eta)$ and $K(\eta, \xi)$ need not be equal.

The key predistribution schemes from [17] and [4] are non-interactive: in the key establishment protocol, each node ξ can compute the common key with any other node η without any communication. Such key predistribution schemes have recently been studied under the name of ID-based non-interactive key establishment (ID-NIKE), usually employing variations of the Diffie-Hellman key exchange, pairings, bilinear or multilinear functions that are costly to implement [6,9,10,11,13,20,22]. Note that for the HIMMO scheme from this paper, the key establishment protocol will be one-pass: the initiator of the protocol sends a message to the other node involved in the protocol, but no reply is required.

In a very simple key predistribution scheme [17], the secret root keying material R is a random symmetric function, so $R(\xi, \eta) = R(\eta, \xi)$ for all nodes ξ and η, the keying material G_ξ is a table of pairs $(\eta, R(\xi, \eta))$, and as its common key with node η, node ξ uses $R(\xi, \eta)$ that it obtains from its look-up table. As R is symmetric, node ξ and η obtain a common key without interaction. Because of the random choice of R, no information on the key between nodes ξ and η can be obtained from the keys between all other pairs of nodes. For systems involving many nodes, however, the tables get large and it is preferable that G_ξ specifies a function to be applied in the key establishment protocol.

A straightforward and efficient key predistribution scheme was described by Blundo et al. [5] in 1992. In this scheme, which we will call *Blundo's scheme*, the trusted party first randomly generates a symmetric bivariate polynomial $R(x, y)$ of degree α in each of the variables with coefficients from $\mathbb{Z}_p$, the ring of integers modulo p. Next, the trusted party provides, in a secure manner, to any node ξ in the network the keying material $R(\xi, y) \in \mathbb{Z}_p[y]$. The key $K(\xi, \eta)$ that node ξ uses in order to communicate with node η equals $R(\xi, \eta)$ (computed modulo p). As R is symmetric, $K(\xi, \eta) = K(\eta, \xi)$. Blundo's scheme is fast and requires little storage. Other advantages are that it allows for any network of size at most p, so that it scales well, e.g., to the Internet, and that nodes can be added to a running network without the need to update already deployed nodes. Blundo's scheme offers information-theoretic security if at most α nodes are compromised. However, simple interpolation using the keying material of any $\alpha+1$ nodes allows to retrieve the root keying material [5], thereby compromising the complete system. Also, the keying material of a single node ξ can be obtained by simple interpolation of the keys of any $\alpha + 1$ colluding nodes with ξ.

3 Description of HIMMO for Key Establishment

In this section, we describe the HIMMO scheme for key establishment. HIMMO has been designed to achieve fast key computation, low bandwidth needs, small memory footprint and low energy consumption. This is the reason why our scheme relies on simple polynomials. In order to achieve collusion resistance, we apply two novel design principles. First, polynomials in different finite rings are mixed to obtain the secret keying material of a device, that is again a simple polynomial. Second, in the key establishment protocol, part of the polynomial evaluation is hidden. Our analysis shows that these two design principles enable an operating and secure scheme.

We use the following notation: for each integer x and positive integer M, we denote by $\langle x \rangle_M$ the unique integer $y \in \{0, 1, \ldots, M-1\}$ such that $x \equiv y \bmod M$.

Following the general description of key predistribution schemes from Section 2, HIMMO comprises three components.

The **setup algorithm** which, on input of a security parameter κ, results in the following system parameters:

- B, the bit length of the identifiers to be used in the system
- b, the bit length of the generated keys
- α, the degree of polynomials to be used in the system
- $m \geq 2$
- the public modulus N, an odd integer of length exactly $(\alpha+1)B+b$ bits

and the following secret randomly generated root keying material:

- m distinct random moduli $q_1, q_2, \ldots, q_m$ of the form $q_i = N - 2^b\beta_i$, where where $0 \leq \beta_i < 2^B$ and at least one of $\beta_1, \ldots, \beta_m$ is odd.
- for $1 \leq i \leq m$ and $0 \leq j \leq k \leq \alpha$, a random integer $R^{(i)}_{j,k}$ with $0 \leq R^{(i)}_{j,k} \leq q_i - 1$, and for $k < j \leq \alpha$, $R^{(i)}_{j,k} = R^{(i)}_{k,j}$

The **keying material extraction algorithm**, which computes for each node ξ in the system, with $0 \leq \xi < 2^B$, the coefficients of the key generating polynomial G_ξ:

$$G_\xi(y) = \sum_{k=0}^{\alpha} G_{\xi,k} y^k \text{ where } G_{\xi,k} = \Big\langle \sum_{i=1}^{m} \langle \sum_{j=0}^{\alpha} R^{(i)}_{j,k} \xi^j \rangle_{q_i} \Big\rangle_N. \tag{1}$$

The **key establishment protocol**, in which a node ξ wishing to communicate with node η with $0 \leq \eta < 2^B$, computes

$$K_{\xi,\eta} = \Big\langle \langle G_\xi(\eta) \rangle_N \Big\rangle_{2^b} \tag{2}$$

and provides η with the helper data $h(\xi, \eta)$ defined as

$$h(\xi, \eta) = \langle K_{\xi,\eta} \rangle_{2^s}, \text{ where } s = \lceil \log_2(4m+1) \rceil. \tag{3}$$

Node η obtains $K_{\xi,\eta}$ as

$$K_{\xi,\eta} = \langle K_{\eta,\xi} + jN \rangle_{2^b}, \tag{4}$$

where j is the unique integer such that

$$|j| \leq 2m \text{ and } jN \equiv h(\xi, \eta) - K_{\eta,\xi} \bmod 2^s. \tag{5}$$

The common key $K(\xi, \eta)$ for nodes ξ and η is

$$K(\xi, \eta) = \lfloor 2^{-s} K_{\xi,\eta} \rfloor. \tag{6}$$

Due to the mixing of modular operations, $\langle G_\xi(\eta) \rangle_N$ and $\langle G_\eta(\xi) \rangle_N$ can differ much from each other. The judicious combination of the system parameters, however, implies that the b last bits of these evaluations, that is, $K_{\xi,\eta}$ and $K_{\eta,\xi}$, although not necessarily equal, are close to each other. As shown in Theorem 1 in the appendix, if $0 \leq \xi, \eta \leq 2^B$, then

$$K_{\xi,\eta} \in \{ \langle K_{\eta,\xi} + jN \rangle_{2^b} \mid 0 \leq |j| \leq 2m \}. \tag{7}$$

Node η can compute $K_{\eta,\xi}$ and use Equation (7) to determine a candidate set C of $4m + 1$ keys that contains $K_{\xi,\eta}$. In some use cases, η can obtain $K_{\xi,\eta}$ by decrypting messages encrypted with $K_{\xi,\eta}$ with all keys from C, and discarding keys that yield invalid messages. In these cases, no helper data needs to be sent and the b-bits key $K_{\xi,\eta}$ can be used as common key between ξ and η.
If discarding keys is infeasible, the helper data $h(\xi, \eta)$ assists η to determine $K_{\xi,\eta}$. As $h(\xi, \eta)$ reveals the s least significant bits of $K_{\xi,\eta}$, only the $b - s$ most significant bits $K_{\xi,\eta}$, that is, the number $\lfloor 2^{-s} K_{\xi,\eta} \rfloor$, are used as common key. In order to not reduce the key length too much, s, and thus m, should not be too large; in particular, b should be larger than s.
We now show that application of Equations (4) and (5) indeed results in $K_{\xi,\eta}$. Let $0 \leq \xi, \eta \leq 2^B$. According to Equation (7), there is an integer j such that $|j| \leq 2m$ and $K_{\xi,\eta} \equiv K_{\eta,\xi} + jN \bmod 2^b$. As $s \leq b$ and $K_{\xi,\eta} \equiv h(\xi, \eta) \bmod 2^s$, it follows that $jN \equiv h(\xi, \eta) - K_{\eta,\xi} \bmod 2^s$, so node η can compute $\langle jN \rangle_{2^s}$. As N and 2^s are relatively prime, node η thus can compute $\langle j \rangle_{2^s}$. As j is in the set $\{-2m, -2m+1, \ldots, 2m\}$ which contains $4m+1 \leq 2^s$ consecutive integers, node η can obtain j from $\langle j \rangle_{2^s}$.

4 HIMMO Performance

HIMMO has been designed keeping in mind that it has to enable very efficient performance. From Equation (2) we observe that obtaining a symmetric key just requires the evaluation of a polynomial of degree α modulo N and taking the b least significant bits. This means that only $\alpha + 1$ modular multiplications are required to compute the key. In each multiplication, the B bit identifier multiplies the $(\alpha + 1)B + b$ bit coefficient and the result is reduced modulo N. These modular operations can be implemented in a very efficient manner for appropriate choices for N, e.g. for $N = 2^{(\alpha+1)B+b} - 1$.

In order to evaluate the performance of the HIMMO scheme, we have implemented it on a very resource-constrained 8-bit CPU ATMEGA128L running at 8 MHz, on the 32-bit NXP LPC1769 LPCXpresso Board running at 120 MHz, and on an Intel i3 3120M (64-bit) running at 2.50 GHz running Xubuntu 14.04. The implementations for the NXP LPC1769 and Intel i3 3120M are based on a C library including the big integer arithmetic for addition and multiplication. Other operations are not required. Our implementation for the ATMEGA128L is optimized in assembler and fits in just $428B$ of Flash memory. This shows that HIMMO can fit even in very resource constrained devices. We also note that the RAM consumption is linear with α since we have to keep in memory a term that is $(\alpha+2)B+b$ bits. Tables 1 and 2 provide a brief summary of the performance of the HIMMO scheme implemented in the above CPUs. For instance, for security parameter $\alpha = 26$ and $B = b = 128$, the execution of the HIMMO algorithm in the very resource-constrained ATMEGA128L only takes 223 milliseconds. This time is around 3000 times slower than on the much more powerful Intel i3 3120M (64-bit) due to the different in clock speed, CPU word size, and fact than the algorithm for the ATMEGA128L is optimized in assembler and the algorithm in the Intel is in plain C. Finally, note that the tables include a row specifying the lattice dimension required in the identify attack to the HI problem that is further explained in Sections 5.3 and 6.2.

Table 1. HIMMO performance for $B = b = 128$ as a function of α.

		α			
		26	34	40	50
Keying material size (KB)		6.90	11.18	15.07	22.83
Lattice dimension		405	665	902	1377
CPU time (msec)	ATMEGA128L (8-bit @ 8 MHz)	223	367	497	743
	NXP LPC1769 (32-bit @ 120 MHz)	7.46	11.82	15.74	23.48
	Intel i3 3120M (64-bit @ 2.5 GHz)	0.034	0.053	0.069	0.103

Table 2. HIMMO performance for $\alpha = 26$ as a function of $b = B$.

		$b = B$			
		64	128	192	256
Keying material size (KB)		3.45	6.90	10.34	13.79
CPU time (msec)	ATMEGA128L (8-bit @ 8 MHz)	63	223	393	632
	NXP LPC1769 (32-bit @ 120 MHz)	2.55	7.46	14.93	25.16
	Intel i3 3120M (64-bit @ 2.5 GHz)	0.015	0.034	0.062	0.100

5 Security Model, Assumptions, and Analysis

This section is outlined as follows: in Section 5.1 we present a computational security model for a generic key predistribution scheme. In Section 5.2 we present the two interpolation problems that form the basis upon which HIMMO is built, and present evidence why these problems are difficult, for suitable parameter choices in HIMMO. In Section 5.3 we consider two possible strategies that the adversary has for winning the game that constitutes the security model in the case that the key predistribution scheme is HIMMO. We show how winning the game using these strategies depends on being able to solve either the HI or the MMO problem. We have not been able to identify promising other strategies.

5.1 Security Model

We formalize the notion of collusion resistance, namely that an attacker who has obtained the keying materials of any number of different identifiers should not be able to calculate the key of a pair of uncompromised identifiers.

We consider a security model that is a game between a challenger and an adversary. The challenger has full knowledge of the key predistribution scheme and all secret parameters that the trusted party used in setting it up; the adversary only knows the public parameters of the system. The adversary can present queries to the challenger. In a query, the adversary randomly picks a valid identifier ζ and the challenger responds with the keying material G_ζ.
After presenting c queries and receiving the corresponding responses, the adversary chooses a pair of identifiers (ξ, η), guesses the key $K_{\xi,\eta}$, and presents these results to the challenger, who checks if the key guess for the pair (ξ, η) was correct. The adversary wins if and only if the following holds:

1. neither ξ nor η was used as the input to any query;
2. the adversary guessed the key $K_{\xi,\eta}$ correctly.

These conditions are similar to the winning condition in the computational COMP-SK security model which is used in the security analysis of ID-NIKE in [20].

5.2 The HI and MMO Problems

We present the two mathematical interpolation problems upon which the HIMMO system is built. The first of these problems is the Hiding Information problem.

Problem 1 (Hiding Information (HI) problem).
Let $f \in \mathbb{Z}[x]$ be of degree at most α, and let $x_i \in \mathbb{Z}$ and $y_i = \langle\langle f(x_i)\rangle_N\rangle_r$ for $0 \leq i \leq c$.
HI problem: given α, N, r, $(x_1, y_1), \ldots, (x_c, y_c)$, and x_0, find y_0.

This problem was studied in [18], where it was shown to be equivalent to a lattice problem in dimension $\alpha + 1 + c$, and that c must be large enough for the solution y_0 to be unique. For the instances of the HI problem that pertain to the HIMMO system, the resulting lattice dimension and structure are such that the known techniques for finding y_0 fail to give the correct answer for $\alpha \gtrsim 20$. A more detailed exposition is given in Section 6.

The second interpolation problem deals with interpolation of a function that is the sum of multiple polynomials, each evaluated modulo a different number. We distinguish two versions of this problem, depending on whether the moduli are given or unknown.

Problem 2 (Mixing Modular Operations (MMO) Problems).
Let $m \geq 2$, $g_1, \ldots, g_m \in \mathbb{Z}[x]$, all of degree at most α, and let $x_i \in \mathbb{Z}$ and $y_i = \sum_{j=1}^{m} \langle g_j(x_i)\rangle_{q_j}$, for $0 \leq i \leq c$.
MMO problem with known moduli: given $\alpha, m, q_1, \ldots, q_m, (x_1, y_1), \ldots, (x_c, y_c)$, and x_0, find y_0.
MMO problem: given α, m, $(x_1, y_1), \ldots, (x_c, y_c)$, and x_0, find y_0.

In [12], it was shown that the MMO problem with known moduli and c colluding nodes can be reduced to finding a vector in a lattice with dimension $m(\alpha+1+c)$, and that c must be at least $m(\alpha+1)$ to find a unique solution. Thus the adversary has to solve a lattice problem in a lattice of dimension at least $m(m+1)(\alpha+1)$, which quickly becomes infeasible if m grows.

Setting up the lattice requires knowledge of the secret moduli $q_1, \ldots, q_m$. When the moduli are unknown, there appears to be no efficient way to reconstruct them from any c observations. For these reasons, we consider solving the MMO problem to be infeasible.

5.3 Security Analysis

In this section, we analyse two strategies that an adversary playing the game described in Section 5.1 can use to find $K_{\xi,\eta}$ from the keying materials $G_{\zeta_i}(y)$, $1 \le i \le c$. These two types of attack are in line with related security models used in other key predistribution schemes such as Matsumoto and Imai [17] and attacks on Blundo's scheme and Zhang's scheme.

The first strategy to calculate $K_{\xi,\eta}$ is to calculate G_ξ from the keying materials G_{ζ_i}, and then to use Equation (2). Turning to the definition of the $G_{\zeta_i}(y)$ in terms of the root keying material, see Equation (1), we see that finding the coefficient $G_{\xi,k}$ of the polynomial $G_\xi(y)$ from the coefficients $G_{\zeta_i,k}$ of the polynomials $G_{\zeta_i}(y)$ amounts to solving an instance of the MMO problem with unknown moduli, which we consider infeasible. In fact, the adversary has to solve a somewhat more complex problem, because the definition of the keying material coefficients in Equation (1) has an additional mod N operation.

In the second strategy, which avoids determining the moduli, the adversary evaluates $\langle G_{\zeta_i}(\xi)\rangle_N$, and takes the b least significant bits thereof, the key $K_{\zeta_i,\xi}$. This key is used as an approximation to K_{ξ,ζ_i}. Finding $K_{\xi,\eta}$ from the set K_{ξ,ζ_i} amounts to solving the HI problem with $r = 2^b$. For such a low value of r, compared to N, solving the HI problem is infeasible if α is large enough.

Using $r > 2^b$, e.g., the whole output of $\langle G_{\zeta_i}(\xi)\rangle_N$, in the HI problem to find $K_{\xi,\eta}$ is not feasible since $G_\xi(\zeta_i)$and $G_{\zeta_i}(\xi)$ only show symmetry in the b least significant bits. The $(\alpha+1)B$ most significant bits are related through the moduli q_i.

To show this, from Lemma 1 in the appendix, $\langle G_\xi(\eta)\rangle_N$ is the sum of three terms. The first term is invariant under the exchange of ξ and η, the second term is a small multiple of N and the third a multiple of 2^b. We consider the effect of the last term in the difference between $\langle G_\xi(\eta)\rangle_N$ and $\langle G_\eta(\xi)\rangle_N$, i.e., $(\mu_\eta(\xi) - \mu_\xi(\eta))2^b$, where according to Lemma 1,

$$\mu_\xi(\eta) = \sum_{i=1}^{m} \beta_i \left\lfloor \frac{A_i(\xi,\eta)}{q_i} \right\rfloor,$$

$$\text{with } A_i(\xi,\eta) = \sum_{k=0}^{\alpha} \left\langle R_k^{(i)}(\xi) \right\rangle_{q_i} \eta^k, \text{ and } R_k^{(i)}(\xi) = \sum_{j=0}^{\alpha} R_{j,k}^{(i)} \xi^j.$$

Although each $R^{(i)}$ is symmetric, the function $A_i(\xi, \eta)$ is not, as $R_k^{(i)}$ is evaluated modulo q_i, while the evaluation of A_i in η is performed over the integers (as is the summation and multiplication with the β_i's). If η is large, then $A_i(\xi, \eta)$ influences all bits, including the highest order bits; if the $\beta_i's$ are large, $\mu_\xi(\eta)$ affects all bits, including the highest order bits. Indeed, assume that the coefficients of $A_i(\xi, \eta)$, i.e., the integers $\left\langle R_k^{(i)}(\xi)\right\rangle_{q_i}$ are uniformly distributed in $\{0, 1, \dots, q_i - 1\}$ then the expected value of $A_i(\xi, \eta)$ equals $\frac{1}{2}q_i \sum_{k=0}^{\alpha} \eta^k \approx \frac{1}{2}q_i\eta^\alpha$. We further assume that each β_i is uniformly chosen from the integers in $[0, 2^B)$. Then the expected value of $\mu_\xi(\eta)$ is $m2^{B-2}\eta^\alpha$. Hence, if we take

$$\eta > 2^B(2/m)^{1/\alpha}, \tag{8}$$

then we expect that $2^b\mu_\xi(\eta)$ is larger than $2^b m 2^{B-2}\eta^\alpha > 2^{(\alpha+1)B+b-1}$, so that $2^b\mu_\xi(\eta)$ affects all bits of $\langle G_\xi(\eta)\rangle_N$. Since the $\mu_\xi(\eta)$ and $\mu_\eta(\xi)$ are affected by the mixing of modular operations, we conjecture that with identifiers η satisfying Equation (8), no information on the $(\alpha+1)B$ most significant bits of $\langle G_\xi(\eta)\rangle_N$ can be obtained from $\langle G_\eta(\xi)\rangle_N$.

The requirement on η expressed in Equation (8) reduces the number of identifiers that we can use from 2^B to $2^B\big(1-(2/m)^{1/\alpha}\big)$. In other words, the "effective bit length" of the identifiers is reduced by $\left\lceil -\log_2\big(1-(2/m)^{1/\alpha}\big)\right\rceil$ bits. For reasonable parameters, this is not a very big number, e.g., for $m = 10$ and $\alpha = 26$, the loss is just over four bits in the identifier space.

6 Experimental Results and HIMMO Parameters

This section describes experiments and provides further background supporting the results in Section 5.3. We also propose specific configuration parameters for which we believe HIMMO to be secure based on these experimental results.

6.1 Experimental Results in the Structure of $\langle G_\xi(\eta)\rangle_N$

In Figure 1 we show experimental evidence for the claim that the $(\alpha+1)B$ most significant bits of $\langle G_\xi(\eta)\rangle_N$ and $\langle G_\eta(\xi)\rangle_N$ are uncorrelated if the identifier interval is restricted according to Equation (8).

Note that this property does not appear to hold for a few most significant bits, which are equal with probability significantly above 0.5. This is because in our simulations some coefficients or identifiers might be slightly smaller so that the effect of the mixing of modular operations does not propagate to the very most significant bits. These few bits may thus be used in an attack as well if a way to find correlations between them is figured out. Currently, this remains an open problem. Effectively, the attacker would then solve a HI problem with a somewhat larger value of r, say $r = 2^{b+\delta b}$, which is still much smaller than N, and hence does not lead to much improvement in the second attack strategy discussed in Section 5.3.

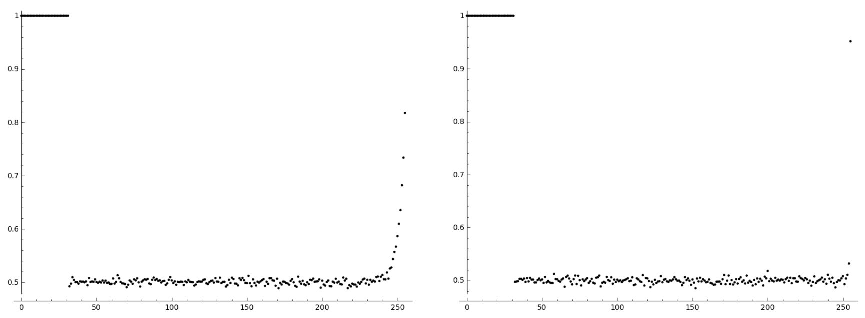

Fig. 1. The probability that the i-th bit of $\langle G_\xi(\eta)\rangle_N$ equals the i-th bit of $\langle G_\eta(\xi)\rangle_N$ when $K_{\xi,\eta} = K_{\eta,\xi}$ as a function of i in a HIMMO system with $\alpha = 6$, $b = B = 32$ and $m = 5$. In the plot on the left, ξ and η are averaged over the interval $[0, 2^B)$, in the plot on the right ξ and η are averaged over the interval $[2^B(2/m)^{(1/\alpha)}, 2^B)$.

6.2 HIMMO and the HI Problem

Minimum Value of c: The HI problem is analyzed in [18]. It is shown that the HI problem is equivalent to a *noisy polynomial interpolation problem*, which in turn is shown to be related to an approximation problem in a certain lattice of dimension $\alpha + 1 + c$. The approximation problem is to find a lattice vector that lies close enough, in infinity norm, to a target vector, i.e., it is a relaxed version of the well-known closest vector problem. It is shown that there are many lattice vectors that solve the approximation problem, each of these vectors gives an estimate for y_0.
It is also shown that because of the structure of this lattice, there is a cross-over value $c_{\min}$, which depends on the distribution of the x_i, such that when $c < c_{\min}$ the lattice vectors that solve the close vector problem give rise to many different estimates for y_0, whereas for $c > c_{\min}$, all solutions to the close vector problem give rise to one, or at most a few different estimates for y_0.
The value $c_{\min}$ is found as the zero of the function S, defined as

$$S(c) = \log\left(\frac{2^{b+1}}{\sqrt{c}}\right) + \frac{1}{2(c-\alpha-1)}\Bigg(\sum_{i=1}^{\alpha}\big(\log((\alpha+1+i)!) - \log(i!)\big) - \log\binom{c}{\alpha+1} - \alpha(\alpha+1)\log(L)\Bigg), \quad (9)$$

where we correct an error in the result from [18] and adapt their notation to the one used in this paper.
In the derivation of this formula, it is assumed that the c points x_i are uniformly chosen from an interval of length L. Numerical experiments in [18] confirm the validity of using this indicator. When $B = b$ and $L = 2^B$, an approximation to $c_{\min}$ is $c_{\min} \approx (\alpha+1)(\alpha+2)/2$. In Table 3 we give $c_{\min}(\alpha, b, B)$ for several values of α and $b = B$, and compare its value to $(\alpha+1)(\alpha+2)/2$.

For a moderately large value of $\alpha = 40$ the attacker must solve a lattice problem in about 900 dimensions, which lies above the upper limit for practical lattice reduction algorithms. We point out that the record in the *Ideal Lattice Challenge* is in dimension 825, see [8]. Increasing α makes this lattice attack even more infeasible. In fact, our experiments described further below show that for $b = B = 32$ and $\alpha > 27$ the approximate methods for finding a close lattice vector fail.

The above analysis holds when the identifiers are uniformly distributed in $[0, 2^B)$. When the attacker can choose the identifiers $x_1, \ldots, x_c$, then he can pick them from a smaller interval containing the identifier x_0 of the node under attack, improving his chances to attack the system. For instance, in the previous case with $\alpha = 40$, $b = B = 32$ and $L = 256$, the indicator function is positive for $c \geq 120$. Simulations confirm this lower bound for a successful attack.
In a practical deployment of HIMMO such a small L attack can be prevented by making it infeasible for the adversary to choose the x_i freely, e.g., by letting the HIMMO identifier x_i be a secure B-bit hash of a node's identity.

Table 3. The value $c_{\min}$ for $B = b$ as a function of α and b and compared with $f(\alpha) := (\alpha + 1)(\alpha + 2)/2$.

		b				
α	$f(\alpha)$	8	16	32	64	128
16	153	122	142	148	151	152
20	231	178	212	223	228	230
24	325	243	297	313	320	323
28	435	317	396	419	428	432
32	561	398	508	539	551	556
36	703	487	635	675	690	697
40	861	582	775	825	845	853

Performance of Lattice Attack on HI Problem The close vector problem in infinity norm can be solved as if it were the closest vector problem for the same target vector. There exist exact algorithms for the closest vector problem, these have running times and memory requirements that grow exponentially in the lattice dimension and turn out to become infeasible if the lattice dimension is larger than about 100. For example, the algorithm from [3] is reported to require 3TB of memory and 2080 hours of computation time for a lattice of dimension 90. These algorithms are thus not suitable for solving the HI problem for $\alpha > 12$. There exist approximate algorithms with more modest memory requirements and smaller running time, polynomial in the lattice dimension. These are based on lattice reduction and rounding or the embedding technique [15] explained further below. The downside is that for these algorithms the upper bound for the error grows exponentially (or slightly slower) in the lattice dimension, so they can break down when the lattice dimension becomes too large. This is investigated experimentally for the lattices we encounter in solving the HI problem.

We first choose a value for b, the number of key bits, B, the number of ID bits, and α, the polynomial degree. We then choose a random odd integer N in the interval $(2^{(\alpha+1)B+b-1}, 2^{(\alpha+1)B+b})$ and $\alpha+1$ random integer polynomial coefficients $f_0, \ldots, f_\alpha$ from $[0, N)$. With these coefficients we construct a polynomial $f(x) = \sum_{j=0}^{\alpha} f_j x^j$. We choose a number c and pick $c+1$ different numbers $x_0, x_1, \ldots, x_c$ from the interval $[0, 2^B)$ and calculate the numbers $y_i = \langle\langle f(x_i)\rangle_N\rangle_{2^b}$, $0 \le i \le c$. The numbers α, b, B, N, x_0 and the c pairs (x_i, y_i) for $1 \le i \le c$ are input to the reconstruction algorithm. The algorithm outputs an estimate $\hat{y}_0$ for y_0. This is the same set-up that we used for the HIMMO challenges [14].

In the remainder of this section we report on our findings in solving our challenges. We choose $b = B = 32$ for all challenges.

The main part of the algorithm is the generation of a set of integer coefficients $g_0, \ldots, g_\alpha$ in $[0, N)$ and the corresponding polynomial $g(x) = \sum_{j=0}^{\alpha} g_j x^j$. The aim is to find $g(x)$ such that $\langle\langle g(x_i)\rangle_N\rangle_{2^b} = y_i$ for $1 \le i \le c$. The estimate $\hat{y}_0$ is then output as $\hat{y}_0 = \langle\langle g(x_0)\rangle_N\rangle_{2^b}$. The algorithm for obtaining the coefficients g_j, makes use of the equivalence of this reconstruction problem to a lattice problem, as described in [18].

It is rare to find a perfect fit in all points x_i, $1 \le i \le c$. Define $e_i = \langle\langle N^{-1}\rangle_{2^b}(\langle\langle g(x_i)\rangle_N\rangle_{2^b} - y_i)\rangle_{2^b}$ and

$$\varepsilon_i = \begin{cases} e_i & \text{if } 0 \le e_i < 2^{b-1} \\ e_i - 2^b & \text{if } 2^{b-1} \le e_i < 2^b \end{cases}.$$

The absolute value $|\varepsilon_i|$ is a measure for the quality of the fit in the point x_i. Since small errors in the fit can have a huge influence on the interpolation error in x_0, we define

$$\ell_i = \tfrac{1}{2}\log_2(1 + \varepsilon_i^2) \tag{10}$$

for use in plots.

In the challenges, $c = (\alpha+1)((\alpha+1)B/b+1)$, which is somewhat larger than $2c_{\min}$. The points x_i are sorted in ascending order, and $x_{c/2} < x_0 < x_{c/2+1}$. For the larger challenges, using all c points becomes infeasible. Therefore we use a subset of $\lfloor c/2 \rfloor$ points (or slightly more), i.e., $c_{\text{used}} \gtrsim \lfloor c/2 \rfloor$. A first strategy (strategy A) is to use the points $\{x_{u(j)} \mid u(j) = \lfloor (c - c_{\text{used}})/2 \rfloor + j, 1 \le j \le \lfloor c_{\text{used}} \rfloor\}$ surrounding x_0. These points lie in an interval of expected length 2^{B-1} and a shorter interval is expected to result in a better interpolation, according to Eq. (9), which is decreasing in the interval length. The unused points can be used to measure the extrapolation errors. A second strategy (strategy B) is to interleave the unused and used points, e.g., with $c_{\text{used}} = \lfloor c/2 \rfloor$ and $u(j) = 2j-1$. Then we lose the advantage of effectively halving the size of the interval, but we can measure the quality of the interpolation in the unused points, which are also spread all over the interval.

The lattice is spanned by the rows of the block matrix

$$L = \begin{pmatrix} 2^b N \mathbb{I}_{c_{\text{used}}} & 0 \\ 2^b \mathbb{V} & \mathbb{I}_{\alpha+1} \end{pmatrix},$$

where $\mathbb{I}_{c_{\text{used}}}$ and $\mathbb{I}_{\alpha+1}$ denote unit matrices of size $c_{\text{used}} \times c_{\text{used}}$ and $(\alpha+1)\times(\alpha+1)$ respectively, and $\mathbb{V}$ denotes the $(\alpha+1) \times c_{\text{used}}$ Vandermonde matrix with elements $V_{i,j} = x^i_{u(j)}$, $0 \le i \le \alpha$, $1 \le j \le c_{\text{used}}$. The problem is to find a lattice vector that lies inside a hypercube of edge length N around a target vector t that is constructed with the values $y_{u(j)}$, $1 \le j \le c_{\text{used}}$.
This is a relaxed version of the Closest Vector Problem, and we use the embedding technique [15] for finding such a close vector. The embedding technique works as follows. First, the matrix L is extended by a column of zeros on the right. Then the matrix is extended by a row on the bottom. The first $\alpha+1+c_{\text{used}}$ elements of this row (the ones below L) form the vector $-t$, for the last element (below the column of zeros added in the first step) we take $(N-1)/2$, which is a reasonable guess for the distance between the lattice defined by L and the target vector. The resulting matrix is interpreted as the basis of a $\alpha+2+c_{\text{used}}$-dimensional lattice. The next step is to find a reduced basis for this lattice, and, in this basis, find the basis vector of which the last component is equal to $(N-1)/2$. The first $\alpha+1+c_{\text{used}}$ components of this basis vector are then added to t, which results in the lattice approximation to t.

The coefficients g_j are obtained from the corresponding components of the resulting lattice vector. We refer to [18] for details.

Results using LLL The first algorithm we use for finding a reduced lattice basis is the LLL [19] wrapper, as implemented in the `fpLLL` library [1] that is included in Sage [21], version 6.6. We always use the default parameters $\delta = 0.99$, $\eta = 0.501$.

The results for an instance of the HI problem for $\alpha = 16$ are shown in Fig. 2. The left-hand graph shows the errors when the used points are chosen according to strategy A. The fit in the used points is almost perfect, and the correct value of y_0 is found. The extrapolation error increases for the unused points further away from the central zone. The right-hand graph shows the errors when the used points are chosen according to strategy B. Here both the fit and extrapolation are almost perfect for all points, and the correct value of y_0 is recovered.

Fig. 3 shows the results for an instance of the HI problem with $\alpha = 18$. The interpolation with used points chosen according to strategy A is again good, but for strategy B, LLL doesn't even give a correct fit in the used points and then fails to interpolate correctly in the unused points. This is consistent with the expectation that strategy A should give a better interpolation.

Fig. 4 shows the results for an instance of the HI problem with $\alpha = 19$. Now $c = 420$, and when 210 of these points are used according to either strategy A or B, both the fit and the interpolation fail. Using 230 points according to strategy A instead of 210, the correct value of y_0 is recovered with LLL.

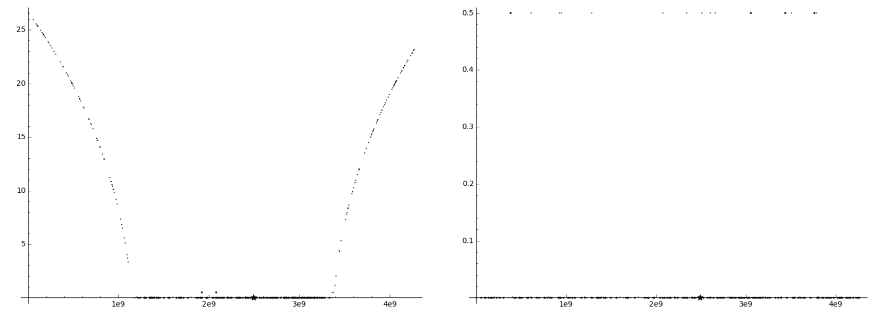

Fig. 2. Plots of the points (x_i, ℓ_i) where ℓ_i is defined in Eq.(10). The points that are used in the reconstruction algorithm are denotes by big dots, the ones that are not used by small dots and (x_0, ℓ_0) is denoted by a star. For these plots, $\alpha = 16$, $c = 306$, and we used 153 points for the reconstruction. In the graph on the left, the used points were chosen according to strategy A, in the graph on the right according to strategy B.

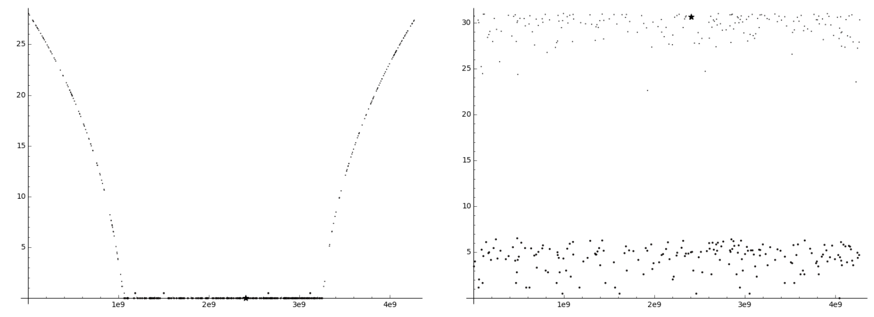

Fig. 3. Plots of the points (x_i, ℓ_i) where ℓ_i is defined in Eq.(10). The points that are used in the reconstruction algorithm are denotes by big dots, the ones that are not used by small dots and (x_0, ℓ_0) is denoted by a star. For these plots, $\alpha = 18$, $c = 380$, and we used 190 points for the reconstruction. In the graph on the left, the used points were chosen according to strategy A, in the graph on the right according to strategy B.

The largest value of α for which we were able to recover the correct y_0 using $c/2 + 20$ points and LLL as the lattice reduction algorithm, is $\alpha = 20$. Taking even more points doesn't appear to help. An example for $\alpha = 25$ is shown in Fig. 5.

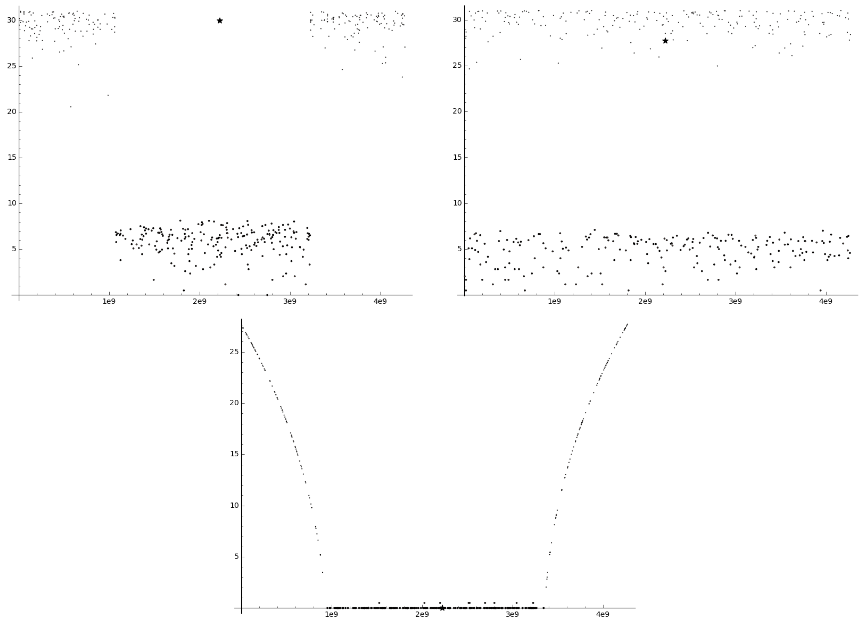

Fig. 4. Plots of the points (x_i, ℓ_i) where ℓ_i is defined in Eq.(10). The points that are used in the reconstruction algorithm are denotes by big dots, the ones that are not used by small dots and (x_0, ℓ_0) is denoted by a star. For these plots, $\alpha = 19$, $c = 420$. In the graphs in the top row we used 210 points for the reconstruction. In the graph on the left, the used points were chosen according to strategy A, in the graph on the right according to strategy B. In the graph on the bottom row we used 230 points according to strategy A.

Fig 6 shows the times needed for solving a couple of instances of the HI problem using strategy A with $c/2$ points, the embedding method and LLL, running in Sage on a 3.0 GHz Intel Xeon.

Results using BKZ The `fpLLL` library that is included in Sage also includes a routine to perform a blockwise Korkine–Zolotarev (BKZ) lattice reduction. This routine has two important parameters, the block size and the precision. In contrast to the LLL implementation, there is no wrapper routine that starts off with a fairly low precision and increases it until the LLL reduction succeeds. Instead, the precision with which the floating point numbers in the BKZ algorithm are calculated, must be given explicitly. If the precision parameter is set

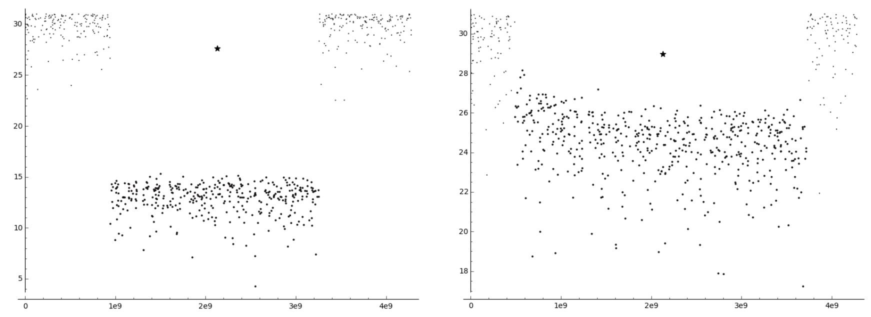
Fig. 5. Plots of the points (x_i, ℓ_i) where ℓ_i is defined in Eq.(10). The points that are used in the reconstruction algorithm are denotes by big dots, the ones that are not used by small dots and (x_0, ℓ_0) is denoted by a star. The plots show the results of applying LLL and strategy A on an instance of the HI problem with $\alpha = 25$, $c = 704$. In the graph on the left, $c_{\text{used}} = 372$, in the graph on the right $c_{\text{used}} = 528$.

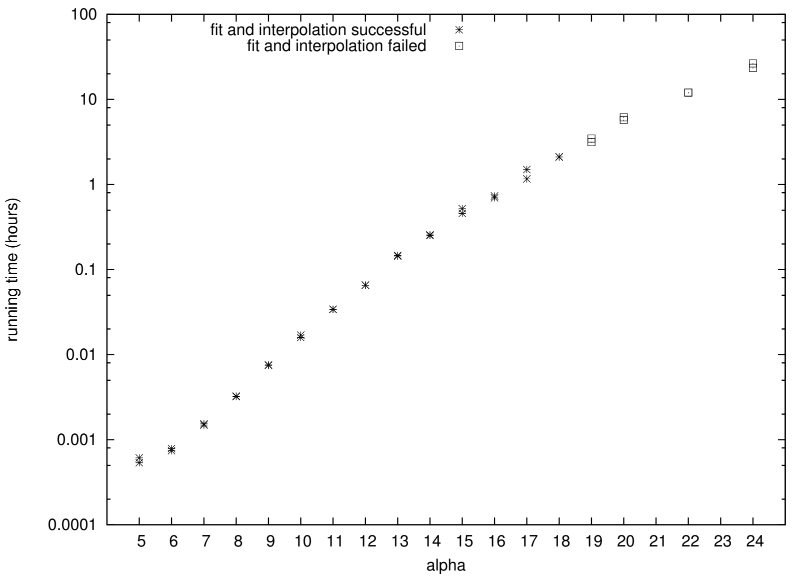

Fig. 6. Running times for solving the HI problem using $(\alpha+1)(\alpha+2)/2$ points according to strategy A, embedding and the fpLLL-wrapper. For $\alpha \geq 19$, this failed to give a good approximation to y_0.

to a value that is too small, BKZ fails with the error message `'infinite loop in babai'`. To avoid this error in the basis reduction of our lattice, the precision must be set to a value not much smaller than $(\alpha + 2)b$ bits, the number of bits of N. This is not surprising, as the largest numbers that appear in the basis matrix are about N times as large as the smallest number in the same column. Typically, the LLL wrapper, which is called as a preconditioning step from the BKZ routine, uses a similarly sized precision parameter in its final, successful, run.

The block size determines the quality of the lattice reduction; a block size value of 2 is equivalent to LLL, larger block sizes lead to a better reduced basis, but it is a priori unclear what blocksize is needed to solve our interpolation problem, as the worst-case error bounds for these algorithms are quite conservative and the algorithms typically behave much better than the worst-case error bounds would suggest.

In Fig. 7 we show the results for an instance of a HI problem with $\alpha = 24$, using 325 points according to strategy A. For this instance, the block size must be increased to 16 in order to obtain a good interpolation.

The timing results for a number of BKZ runs are given in Fig. 8.

The largest instance we have been able to solve so far using this method has $\alpha = 27$, with $c = 812$, using $c/2 + 20$ points according to strategy A, and the BKZ algorithm with 928 bits precision and block size 18. It ran for 303 hours. The reason that we did not result in breaking even larger instances is not lack of patience, but most probably a problem with the BKZ implementation: for larger instances when α grows, it appears that a block size value of at least 20 is needed, but then the BKZ routine appears to end up in an infinite loop, without outputting any log messages, probably because of a subtle bug triggered by the block size and the large precision parameter. The above also seems to indicate that BKZ performance decreases when α increases due to the higher block size that is required.

6.3 HIMMO Security Parameters

For security reasons, the HIMMO parameters advantageously have the following characteristics:

- a large value of b so that keys cannot be guessed by brute-force.
- a large value α so that attacking a keying material $G_\xi(y)$ requires solving a lattice of big dimension.
- keeping the q_i's secret and optionally taking a relatively high value of m to ensure that attacking the root keying material $R^{(i)}(x, y)$ involves solving a lattice of big dimension.

A set of parameters that the authors consider to lead to a complexity-theoretic secure HIMMO instance is $b = B = 80$, $\alpha = 50$, and $m = 10$. With these parameters, attacking the 80-bit keys generated by a specific device would require solving a lattice of dimension 1377 for the HI problem once enough nodes have

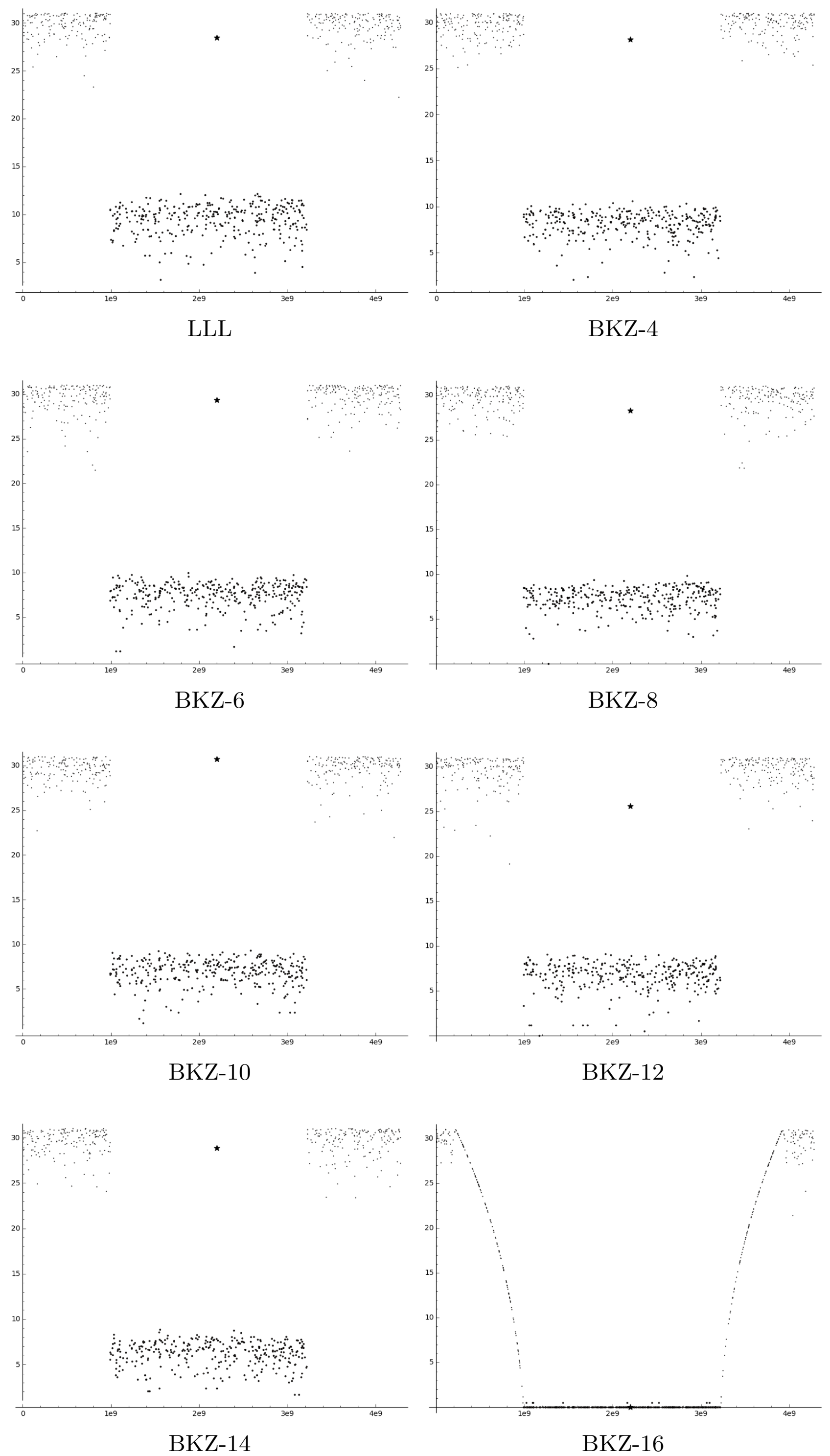

Fig. 7. Plots of the points (x_i, ℓ_i) where ℓ_i is defined in Eq.(10). The points that are used in the reconstruction algorithm are denotes by big dots, the ones that are not used by small dots and (x_0, ℓ_0) is denoted by a star. The plots show the results of applying LLL and strategy A on an instance of the HI problem with $\alpha = 24$, $c = 650$, $c_{\text{used}} = 325$. In the top left graph, the LLL wrapper algorithm was used, in the other graphs the BKZ algorithm with precision 832 bits and block sizes 4, 6, 8, 10, 12, 14, and 16.

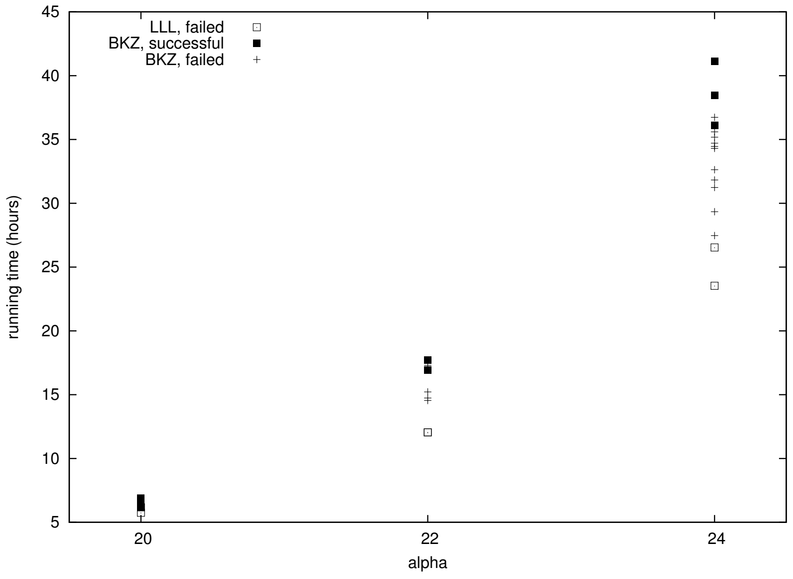

Fig. 8. Running times for solving two instances of a HI problem using $(\alpha+1)(\alpha+2)/2$ points according to strategy A, embedding and either fpLLL-wrapper or BKZ with various block sizes $(4, 6, 8, \ldots)$. For the instances with $\alpha = 20$, block size 4 was sufficient to retrieve the correct answer, for $\alpha = 22$ both instances required block size 8 and for $\alpha = 24$ one instance required block size 14 and the other 16.

been compromised. As the highest dimensional lattice successfully attacked so far was on a lattice of dimension 455 (for $\alpha = 27$), we feel this parameter choice is quite conservative. Attacking HIMMO through the MMO problem is hopeless since the q_i's are secret, and even if they were known, an attacker would have to deal with a lattice of dimension 5610. Finding a key by means of a brute force attack is not feasible either due to the chosen key length.

As described in the next section, HIMMO enables practical applications that require mapping a bit-string of arbitrary length to a B bit identifier. In this case, B should be equal the output size of a collision-free hash function. In order that birthday attacks on the hash function and brute force attacks on the key have approximately equal complexity, we choose $B = 2b$. For such applications, the authors thus consider the following set of parameters to lead to a complexity -theoretic secure HIMMO instance: $b = 80, B = 160, \alpha = 50$ and $m = 10$.

7 Practical Protocols and Schemes Enabled by HIMMO

HIMMO's collusion resistance and excellent performance provides us with a new primitive to enable very practical security protocols. Building on HIMMO's pairwise key agreement, any pair of devices in a network of any size can securely communicate with each other. With HIMMO, the system remains flexible since nodes can be added to a running network without the need to update already deployed nodes.

We now describe a simple protocol that allows a node ξ to directly send a message M to node η without incurring any round trip delays. Node ξ computes its key $K_{\xi,\eta}$, the helper data $h(K_{\xi,\eta})$ and the common key $K(\xi, \eta)$ as explained Section 3. It protects M by using $K(\xi, \eta)$ and some authenticated encryption algorithm e, and sends to node η the helper data $h(K_{\xi,\eta})$ and the encrypted message $E = e(M, K(\xi, \eta))$. Upon reception, node η computes $K_{\eta,\xi}$ and combines it with the helper data $h(K_{\xi,\eta})$ to obtain $K(\xi, \eta)$, as explained in Section 3. Node η subsequently obtains M by by decrypting and verifying the authenticity of the received message E.

The fact that the HIMMO scheme can efficiently use long B-bit identifiers allows us to design further identity-based protocols providing more functionality. These protocols are built by mapping an input bit string of arbitrary length to a B-bit HIMMO identifier by means of a collision resistant hash function H. For instance, we can enable **implicit certification and verification of credentials** between any pair of entities, as follows.
In a registration phase, a node Ξ that wants to register with the system provides the trusted party with its set of identifiers, e.g., in the case of a device: type, manufacturing date, etc. The trusted party can add further parameters for better node identification, such as the issue date of the keying material and its expiration date. The concatenation of all these identifiers constitutes $\mathit{Credentials}(\Xi)$, the credentials of Ξ. The trusted party obtains the node's HIMMO identity as $\xi = H(\mathit{Credentials}(\Xi))$. This HIMMO identity is used in the keying material extraction algorithm to compute the secret keying material G_ξ of Ξ. We observe

that *Credentials*(Ξ) are linked to the secret keying material by means of $H(\cdot)$ and the keying material extraction algorithm.
In the operational phase, two devices can execute a protocol that allows not only for direct secure communication of a message M but also for implicit certification and verification of the credentials of the sender Ξ because the key generating polynomial assigned to a node is linked to its credentials by means of H. The protocol builds on the protocol for direct secure sending of a message as described above. In fact, node Ξ with HIMMO identity ξ uses the above protocol to send to the node with HIMMO identity η the message M' defined as the concatenation of ξ, M and *Credentials*(Ξ). After η has obtained M', it verifies the credentials of Ξ by checking whether $\xi = H(\mathit{Credentials}(\Xi))$.

If the output size of $H(\cdot)$ is long enough, e.g., 256 bits, and equal to B, then it is infeasible for an attacker to find any other set of credentials leading to the same output ξ. The fact that credential verification might be prone to birthday attacks motivates the choice $B = 2b$ for the relation between identifier and key sizes in the HIMMO scheme. In this way, the scheme provides an equivalent security level for credential verification and key generation. The capability for credential verification enables applications such as the verification of the expiration date of the credentials (and the keying material) of a node, the verification of the access roles of the sender node ξ encoded in its credentials, or the capability of using any bit-string as the identity of the nodes.

The previous protocols have the **key escrow capability** since the trusted party keeps the secret root keying material that allows for the generation of any key in the system. In some settings, we would like to have this capability **shared between several trusted parties** to enhance the security of the system. HIMMO supports such an extension supporting ℓ different trusted parties in the following way. The setup algorithm consists of two steps: in a first step, parameters (b, B, m, α, N) are centrally determined and published; in a second step, for $1 \leq j \leq \ell$, trusted party j independently generates m secret $q_{j,i}$ and the corresponding m secret symmetric bivariate polynomials $R^{(j,i)}(x, y)$ over $\mathbb{Z}_{q_{j,i}}$. In the keying material extraction phase, each node ξ securely receives from each of the ℓ trusted parties a key generating polynomial $G_\xi^{(j)}(y) \in \mathbb{Z}_N[y]$. Node η computes the coefficients of its final key generating polynomial G_ξ by adding the corresponding coefficients of $G_\xi^{(1)}, \ldots, G_\xi^{(\ell)}$, so

$$G_\xi(y) = \left\langle \sum_{j=1}^{\ell} G_\xi^{(j)}(y) \right\rangle_N. \tag{11}$$

Key generation in the key establishment protocol is done as in HIMMO. Note that the scheme operates exactly as a scheme with a single trusted party which generates the $m \cdot \ell$ root keying material polynomials $R^{(j,i)}(x, y) \in \mathbb{Z}_{q_{j,i}}[x, y]$ for $1 \leq i \leq m, 1 \leq j \leq \ell$. Clearly, if $\ell > 1$, a single trusted party cannot determine the key generating polynomial of individual nodes.

Another functionality enabled by HIMMO refers to **secure broadcast in the sense of source authentication**. In this case, we assume a set of devices has been configured with their HIMMO identities and their secret key generating polynomials. The TTP wishes then to securely (in the sense of source authentication) send a broadcast message M to all devices. The TTP then computes the hash of M as $\mu = H(M)$ and obtains a key generating polynomial G_μ associated to μ. Next the TTP broadcasts $\{M, G_\mu\}$ where G_μ is playing the role of a signature. When device η receives $\{M', G_{\mu'}\}$, it obtains $\mu' = H(M')$ and verifies the validity of the message by checking whether $K_{\eta,\mu'}$ and $K_{\mu',\eta}$ satisfy Equation (7). A similar protocol is described in [23] although the underlying security primitive is now broken [2].

HIMMO can also be extended to allow for **t-key agreement**, i.e., key agreement in a group of t devices if the TTP uses polynomials in t variables and distributes a key generating polynomial in $t-1$ variables to the nodes. Note that in this case the HIMMO parameters need to be adapted to introduce the HI and MMO problem in the correct parts of the keys and coefficients of the key generating polynomial.

8 Conclusions

We have put forth a completely new approach to key predistribution schemes, avoiding the use of any costly one-way functions (and pairings) in a discrete log setting. Rather, we have used an approach remotely akin to NTRU, involving an intricate combination of polynomials evaluated over different “finite rings.” We believe that this approach is of high potential and may spark further research into related primitives.

The performance of the HIMMO key agreement protocol is very competitive, allowing for lightweight implementations needed for applications such as wireless sensor networks and the Internet of Things. We have also shown that the best (collusion) attacks currently known are based on lattice-basis reduction, and that these attacks are bound to fail for the proposed parameter selection using state-of-the-art algorithms, viz. the BKZ algorithm as implemented in the `fpLLL` library. Future work may address the use of more accurate algorithms in the attack, e.g., BKZ-2.0 [7] with large block sizes and aggressive pruning.

References

1. M. Albrecht, D. Cadé, X. Pujol, and D. Stehlé. fplll-4.0, a floating-point LLL implementation. Available at `http://perso.ens-lyon.fr/damien.stehle`.
2. Martin Albrecht, Craig Gentry, Shai Halevi, and Jonathan Katz. Attacking Cryptographic Schemes Based on ”Perturbation Polynomials”. In *CCS09, Proc. 16th ACM Conference on computer and communications security*, pages 1–10. ACM, 2009.
3. Anja Becker, Nicolas Gama, and Antoine Joux. Solving shortest and closest vector problems: The decomposition approach. Cryptology ePrint Archive, Report 2013/685, 2013. `http://eprint.iacr.org/`.

4. R. Blom. An optimal class of symmetric key generation systems. In T. Beth, N. Cot, and I. Ingemarsson, editors, *EUROCRYPT '84*, LNCS 209, pages 335–338. Springer, 1985.
5. C. Blundo, A. de Santis, A.Herzberg, S. Kutten, U. Vaccaro, and M. Yung. Perfectly secure key distribution for dynamic conferences. *Information and Computation*, 146:1–23, 1998.
6. Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In *Advances in Cryptology-ASIACRYPT 2013*, pages 280–300. Springer, 2013.
7. Yuanmi Chen and Phong Q. Nguyen. BKZ 2.0: Better Lattice Security Estimates. In *Advances in Cryptology ASIACRYPT 2011*, volume Volume 7073 of *Lecture Notes in Computer Science*, pages 1–20, 2011. `http://www.iacr.org/archive/asiacrypt2011/70730001/70730001.pdf`.
8. TU Darmstadt. Welcome to the ideal lattice challenge. Web repository, 2014. `http://www.latticechallenge.org`.
9. Régis Dupont and Andreas Enge. Provably secure non-interactive key distribution based on pairings. *Discrete Applied Mathematics*, 154(2):270–276, 2006.
10. Eduarda S.V. Freire, Dennis Hofheinz, Eike Kiltz, and Kenneth G. Paterson. Non-interactive key exchange. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *Public-Key Cryptography PKC 2013*, volume 7778 of *Lecture Notes in Computer Science*, pages 254–271. Springer Berlin Heidelberg, 2013.
11. Eduarda SV Freire, Dennis Hofheinz, Kenneth G Paterson, and Christoph Striecks. Programmable hash functions in the multilinear setting. In *Advances in Cryptology–CRYPTO 2013*, pages 513–530. Springer, 2013.
12. Oscar García-Morchón, Domingo Gómez-Pérez, Jaime Gutiérrez, Ronald Rietman, and Ludo Tolhuizen. The MMO problem. In *Proc. ISSAC'14*, pages 186–193. ACM, 2014.
13. Rosario Gennaro, Shai Halvei, Hugo Krawczyk, Tal Rabin, Steffen Reidt, and Stephen D. Wolthusen. Strongly-resilient and non-interactive hierarchical key-agreement in manets. In *ESORICS 2008*, volume 5283 of *Lecture Notes in Computer Science*, pages 49–65. Springer, 2008.
14. The HIMMO Challenge. `http://www.himmo-scheme.com`.
15. R. Kannan. Minkowski's convex body theorem an integer ptogramming. *Math. Oper. Res.*, 12(3):415–440, 1987.
16. Xinyu Lei and Xiaofeng Liao. NTRU-KE: A lattice-based public key exchange protocol. Cryptology ePrint Archive, Report 2013/718, 2013.
17. T. Matsumoto and H. Imai. On the key predistribution system: a practical solution to the key distribution problem. In C. Pomerance, editor, *Advances in Cryptology – CRYPTO'87*, LNCS 293, pages 185–193. Springer, 1988.
18. Oscar García Morchon, Ronald Rietman, Igor E. Shparlinski, and Ludo Tolhuizen. Interpolation and approximation of polynomials in finite fields over a short interval from noisy values. *Experimental mathematics*, 23:241–260, 2014.
19. Phong Q. Nguyen and Brigitte Vallée, editors. *The LLL Algorithm - Survey and Applications*. Information Security and Cryptography. Springer, 2010.
20. Kenneth G Paterson and Sriramkrishnan Srinivasan. On the relations between non-interactive key distribution, identity-based encryption and trapdoor discrete log groups. *Designs, Codes and Cryptography*, 52(2):219–241, 2009.
21. Sage. `http://www.sagemath.org`.
22. Ryuichi Sakai, Kiyoshi Ohgishi, and Masao Kasahara. Cryptosystems based on pairing. In *The 2000 Symposium on Cryptography and Information Security, Okinawa, Japan*, pages 135–148, 2000.

23. W. Zhang, N. Subramanian, and G. Wang. Lightweight and Compromise-Resilient Message Authentication in Sensor Networks. In *Proceedings IEEE INFOCOM 2008*, 2008.
24. W. Zhang, M. Tran, S. Zhu, and G. Cao. A Random Perturbation-based Scheme for Pairwise Key Establishment in Sensor Networks. In *8th ACM Int. Symp. on Mobile Ad Hoc Networking and Computing (MobiHoc) 2007*, pages 90–99, 2007.

Appendix: validation of HIMMO

As stated in Section 3, the key $K_{\xi,\eta}$ generated by node ξ for communicating with node η need not equal $K_{\eta,\xi}$. In this appendix, we validate HIMMO by showing a relationship between those keys.

Lemma 1 *For all integers ξ and η we have that*

$$\langle G_\xi(\eta)\rangle_N = \sum_{i=1}^{m} \langle R^{(i)}(\xi,\eta)\rangle_{q_i} + \lambda_\xi(\eta)N - \mu_\xi(\eta)2^b, \text{ with}$$

$$\lambda_\xi(\eta) = \sum_{i=1}^{m} \left\lfloor \frac{A_i(\xi,\eta)}{q_i} \right\rfloor - \left\lfloor \frac{1}{N}\sum_{i=1}^{m} A_i(\xi,\eta) \right\rfloor \text{ and } \mu_\xi(\eta) = \sum_{i=1}^{m} \beta_i \left\lfloor \frac{A_i(\xi,\eta)}{q_i} \right\rfloor, \text{ where}$$

$$A_i(\xi,\eta) = \sum_{k=0}^{\alpha} \langle R_k^{(i)}(\xi)\rangle_{q_i} \eta^k \text{ and } R_k^{(i)}(\xi) = \sum_{j=0}^{\alpha} R_{j,k}^{(i)} \xi^j.$$

Proof. We clearly have that

$$\langle G_\xi(\eta)\rangle_N = \langle H_\xi(\eta)\rangle_N \text{ where } H_\xi(\eta) = \sum_{k=0}^{\alpha}\sum_{i=1}^{m} \langle R_k^{(i)}(\xi)\rangle_{q_i} \eta^k.$$

As a consequence,

$$H_\xi(\eta) = \sum_{i=1}^{m} \left(\left\langle \sum_{k=0}^{\alpha} \langle R_k^{(i)}(\xi)\rangle_{q_i} \eta^k \right\rangle_{q_i} + q_i \left\lfloor \frac{1}{q_i} \sum_{k=0}^{\alpha} \langle R_k^{(i)}(\xi)\rangle_{q_i} \eta^k \right\rfloor \right).$$

Using the definition of $A_i(\xi,\eta)$, we find that

$$H_\xi(\eta) = \sum_{i=1}^{m} \langle R^{(i)}(\xi,\eta)\rangle_{q_i} + N\sum_{i=1}^{m} \left\lfloor \frac{A_i(\xi,\eta)}{q_i} \right\rfloor - \sum_{i=1}^{m} (N-q_i) \left\lfloor \frac{A_i(\xi,\eta)}{q_i} \right\rfloor.$$

As $\langle H_\xi(\eta)\rangle_N = H_\xi(\eta) - N\lfloor H_\xi(\eta)/N \rfloor$, and $H_\xi(\eta) = \sum_{i=1}^{m} A_i(\xi,\eta)$, we infer that

$$\langle H_\xi(\eta)\rangle_N = \sum_{i=1}^{m} \langle R^{(i)}(\xi,\eta)\rangle_{q_i} + N\left(\sum_{i=1}^{m} \left\lfloor \frac{A_i(\xi,\eta)}{q_i} \right\rfloor - \left\lfloor \frac{1}{N} \sum_{i=1}^{m} A_i(\xi,\eta) \right\rfloor \right) - \sum_{i=1}^{m} (N-q_i) \left\lfloor \frac{A_i(\xi,\eta)}{q_i} \right\rfloor. \quad \square$$

Theorem 1 *Let $0 \leq \xi, \eta \leq 2^B - 1$. We have that*

$$K_{\eta,\xi} \in \Big\{ \big\langle K_{\xi,\eta} + jN \big\rangle_{2^b} \;\Big|\; j \in \mathbb{Z}, |j| \leq 2m \}.$$

Proof. Using the notation from Lemma 1, we have

$$K_{\xi,\eta} = \Big\langle \big\langle G_\xi(\eta) \big\rangle_N \Big\rangle_{2^b} = \Big\langle \sum_{i=1}^{m} \big\langle R^{(i)}(\xi,\eta) \big\rangle_{q_i} + N\lambda_\xi(\eta) \Big\rangle_{2^b}, \text{ and}$$

$$K_{\eta,\xi} = \Big\langle \sum_{i=1}^{m} \big\langle R^{(i)}(\eta,\xi) \big\rangle_{q_i} + N\lambda_\eta(\xi) \Big\rangle_{2^b}.$$

As each root keying polynomial $R^{(i)}$ is symmetric,

$$K_{\xi,\eta} = \big\langle K_{\eta,\xi} + N(\lambda_\xi(\eta) - \lambda_\eta(\xi)) \big\rangle_{2^b}.$$

We now give an upper bound to the absolute value of $\lambda_\xi(\eta) - \lambda_\eta(\xi)$.
By definition, $\langle A_i(\xi,\eta) \rangle_{q_i} = A_i(\xi,\eta) - q_i \lfloor A_i(\xi,\eta)/q_i \rfloor$ for each i, whence

$$\begin{aligned} \lambda_\xi(\eta) &= \sum_{i=1}^{m} \frac{A_i(\xi,\eta)}{q_i} - \sum_{i=1}^{m} \frac{\langle A_i(\xi,\eta) \rangle_{q_i}}{q_i} - \Big\lfloor \frac{1}{N} \sum_{i=1}^{m} A_i(\xi,\eta) \Big\rfloor \\ &= \tilde{\lambda}_\xi(\eta) - \sum_{i=1}^{m} \frac{\big\langle R^{(i)}(\xi,\eta) \big\rangle_{q_i}}{q_i}, \text{ where } \tilde{\lambda}_\xi(\eta) = \sum_{i=1}^{m} \frac{A_i(\xi,\eta)}{q_i} - \Big\lfloor \frac{1}{N} \sum_{i=1}^{m} A_i(\xi,\eta) \Big\rfloor. \end{aligned}$$

The symmetry of the root keying polynomials implies that

$$\lambda_\xi(\eta) - \lambda_\eta(\xi) = \tilde{\lambda}_\xi(\eta) - \tilde{\lambda}_\eta(\xi). \tag{12}$$

We continue with providing upper and lower bounds on $\tilde{\lambda}_\xi(\eta)$.
As $\lfloor x \rfloor \leq x$ for all x, and for all i, $A_i(\xi,\eta) \geq 0$ and $q_i \leq N$, it follows that $\tilde{\lambda}_\xi(\eta) \geq 0$.
We clearly have that

$$\tilde{\lambda}_\xi(\eta) \leq \sum_{i=1}^{m} \frac{A_i(\xi,\eta)}{q_i} + \Big(1 - \frac{1}{N} \sum_{i=1}^{m} A_i(\xi,\eta)\Big) = 1 + \sum_{i=1}^{m} \frac{N - q_i}{Nq_i} A_i(\xi,\eta).$$

Moreover, for each i we have that

$$\begin{aligned} A_i(\xi,\eta) = \sum_{k=0}^{\alpha} \big\langle R_k^{(i)}(\xi) \big\rangle_{q_i} \eta^k &\leq \sum_{k=0}^{\alpha} (q_i - 1)\eta^k \leq (q_i - 1) \sum_{k=0}^{\alpha} (2^B - 1)^k \\ &< q_i \sum_{k=0}^{\alpha} \binom{\alpha}{k} (2^B - 1)^k = q_i 2^{\alpha B}. \end{aligned}$$

We conclude that $0 \leq \lambda'_\xi(\eta) < 1 + \sum_{i=1}^{m} (N - q_i) 2^{\alpha B}/N$. As $0 \leq N - q_i = \beta_i 2^b \leq 2^{B+b}$, and $N > 2^{(\alpha+1)B+b-1}$, we have that

$$0 \leq \lambda'_\xi(\eta) < 1 + 2m.$$

Of course, the same bounds are valid for $\tilde{\lambda}_\xi(\eta)$. Combining these bounds with (12), and the fact $\lambda_\xi(\eta) - \lambda_\eta(\xi)$ is an integer number, the theorem follows. $\square$

It can be shown that under reasonable conditions, the bound from Theorem 1 cannot be significantly improved.