Key generation errors in the HIMMO scheme

Cloostermans, B.

Award date:
2014

Disclaimer
This document contains a student thesis (bachelor's or master's), as authored by a student at Eindhoven University of Technology. Student theses are made available in the TU/e repository upon obtaining the required degree. The grade received is not published on the document as presented in the repository. The required complexity or quality of research of student theses may vary by program, and the required minimum study period may vary in duration.

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
Key Generation Errors in the HIMMO Scheme

B. Cloostermans

Supervised by:
Dr.ir. L.A.M. Schoenmakers (TU/e)
Dr.ir. L.M.G.M Tolhuizen (Phillips)

A thesis submitted in partial fulfilment of the requirements
for the degree of

Master of Science

August 22, 2014
Abstract

HIMMO is a light-weight symmetric key establishment scheme aiming at fully collision-resistant identity-based key agreement. In an identity-based pairwise key agreement scheme, a Trusted Third Party provides each node in the system with private keying material. Two nodes can then generate a pairwise key by using their own keying material and the identity number of the other node. The full collusion resistance property implies that the scheme remains secure even if arbitrarily many nodes are compromised. Finally, the light-weight property means that HIMMO can efficiently run on devices with limited storage capacity and computation power. HIMMO mix is a variation on HIMMO aiming at the same properties.

Generated pairwise keys in HIMMO are nearly equal, but not exactly so. In compensating for this discrepancy, nodes waste valuable storage space and computation time. We provide new insights into the behavior of the key generation errors. This allows a node to very efficiently compensate for the error it makes while generating keys, resulting in lower storage requirements and faster key computation time. Our modifications reduce storage requirements for common HIMMO setups by 10% to 75% for common HIMMO setups. Compensating for the errors is even more important in HIMMO mix, as errors prevented HIMMO mix from being set up with anything but trivial parameters. Using our modifications, HIMMO mix errors can efficiently be dealt with for almost any choice of parameters.

Our new insights also revealed security risks. Key generation errors provide an attacker with additional information about the keying material of a target node. This makes retrieving the keying material of a target node easier than was previously assumed. Using information gathered from key generation errors, we show how an attacker can retrieve a node’s keying material in a HIMMO system and several HIMMO mix systems which were assumed to have the full collusion resistance property. However, for sufficiently large system parameters, retrieving keying material remains computationally infeasible.
Acknowledgements

This thesis is the result of my graduation project for the master program Industrial and Applied Mathematica at Eindhoven University of Technology. The project was carried out during an internship at Philips Research.

I would like to thank my supervisors at Philips, Ludo Tolhuizen, Ronald Rietman and Oscar Garcia, not only for the opportunity to do the internship, but also for their guidance on my research and their invaluable suggestions on how to improve my thesis.

Of course, I would like to express my deep gratitude to my supervisor from TU/e, Berry Schoenmakers for his supervision and insightful comments. I would also like to thank Benne de Weger and Aart Blokhuis for being on my graduation committee.

Thanks also go to my parents, for their continued support. Finally, I wish to thank my girlfriend Christine and all my friends at university for making my studies at Eindhoven a very enjoyable time. Without you guys I would never have made it this far!
## Contents

5.3 Experiments .............................................. 39

6 The HIMMO Mix Scheme ................................. 43
   6.1 System Operations .................................. 43
   6.2 Analysis ............................................ 44
   6.3 Key Reconciliation ................................. 47
   6.4 Keying Material Share Attack ...................... 47

7 HIMMO Modifications in HIMMO Mix .................. 51
   7.1 Improving Key Equality ............................ 51
   7.2 Comparisons ...................................... 53

8 The Error-based Attack for HIMMO Mix ............... 55
   8.1 Obtainable Bits ................................... 55
   8.2 Required Bits ..................................... 56
   8.3 Attacking the First $n-1$ Systems ................. 57
   8.4 Attacking the Last System ......................... 58
   8.5 Experiments ...................................... 59

9 Improved Key Equality and the Error-based Attack .... 63

10 Conclusions ............................................... 67
   10.1 Key Equality ...................................... 67
   10.2 Keying Material Share Attack ..................... 68

Bibliography ................................................... 69
List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Blundo Operation</td>
<td>2</td>
</tr>
<tr>
<td>2.1</td>
<td>Example of relevant bits in HIMMO keying material</td>
<td>10</td>
</tr>
<tr>
<td>3.1</td>
<td>Plot of $\lambda_\xi(\eta)$</td>
<td>15</td>
</tr>
<tr>
<td>3.2</td>
<td>Plot of $\delta_{\xi,\eta}$</td>
<td>15</td>
</tr>
<tr>
<td>3.3</td>
<td>Plot of $\lambda_\xi(\eta) - S(\xi, \eta)$</td>
<td>17</td>
</tr>
<tr>
<td>3.4</td>
<td>Plot of $\Lambda_\xi(\eta)$</td>
<td>17</td>
</tr>
<tr>
<td>5.1</td>
<td>Bounds found for $\Lambda_\xi(y)$</td>
<td>34</td>
</tr>
<tr>
<td>5.2</td>
<td>Number of most significant bits an attacker finds by examining key generation errors</td>
<td>40</td>
</tr>
<tr>
<td>8.1</td>
<td>Keying material bits which colluding nodes can retrieve using only most significant key bits</td>
<td>56</td>
</tr>
<tr>
<td>8.2</td>
<td>Example of relevant bits in HIMMO mix keying material</td>
<td>57</td>
</tr>
<tr>
<td>8.3</td>
<td>Number of most significant bits an attacker finds with $\alpha = 2$</td>
<td>60</td>
</tr>
<tr>
<td>8.4</td>
<td>Number of most significant bits an attacker finds with $\alpha = 2$</td>
<td>60</td>
</tr>
<tr>
<td>8.5</td>
<td>Number of most significant bits an attacker finds with $\alpha = 4$</td>
<td>61</td>
</tr>
<tr>
<td>9.1</td>
<td>An example of bounds an attacker finds for $\Lambda'_\xi(\eta)$</td>
<td>63</td>
</tr>
<tr>
<td>9.2</td>
<td>Another example of bounds an attacker finds for $\Lambda'_\xi(\eta)$</td>
<td>64</td>
</tr>
</tbody>
</table>
### List of Tables

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>HIMMO storage requirements for high degree polynomials</td>
<td>30</td>
</tr>
<tr>
<td>4.2</td>
<td>Storage requirements for HIMMO with short keys</td>
<td>31</td>
</tr>
<tr>
<td>5.1</td>
<td>Number of colluding nodes required to retrieve $G_{\xi}(y)$ with $s$ most significant bits known</td>
<td>40</td>
</tr>
<tr>
<td>7.1</td>
<td>HIMMO mix storage requirements for $n = 2$</td>
<td>54</td>
</tr>
<tr>
<td>7.2</td>
<td>HIMMO mix storage requirements for $\alpha = 5$</td>
<td>54</td>
</tr>
<tr>
<td>8.1</td>
<td>Number of colluding nodes required to retrieve HIMMO mix keying material with $\alpha = 2$</td>
<td>59</td>
</tr>
<tr>
<td>8.2</td>
<td>Number of colluding nodes required to retrieve HIMMO mix keying material with $\alpha = 3$</td>
<td>60</td>
</tr>
<tr>
<td>8.3</td>
<td>Number of colluding nodes required to retrieve HIMMO mix keying material with $\alpha = 4$</td>
<td>61</td>
</tr>
</tbody>
</table>
Notations

\( \langle x \rangle_M \ (x \in \mathbb{Z}, M \in \mathbb{N}_+) \) the integer \( y \in \{0, \ldots, M - 1\} \) such that \( x \equiv y \mod M \)

key size in bits \( b \)

identifier size in bits \( B \)

degree of polynomial \( \alpha \)

number of HIMMO instances in HIMMO mix \( n \)

public moduli \( \text{odd numbers } N_j \text{ of exactly } (\alpha + 1)B + b \text{ bits for } 1 \leq j \leq n \)

number of secret moduli in a HIMMO system \( m \)

secret moduli \( q_{11}, \ldots, q_{mn} \), where \( q_{ij} = N_j - 2^k \beta_{ij} \), and \( 0 \leq \beta_{ij} < 2^B \)

secret root keying material coefficients of \( n \cdot m \) symmetric polynomials \( R^{(i,j)}(x, y) \),
\( R^{(i,j)}(x, y) = \sum_{k=0}^{\alpha} \sum_{l=0}^{\alpha} R^{(i,j)}_{k,l} x^k y^l = \sum_{k=0}^{\alpha} R^{(i,j)}_{k} y^k \),
with \( 0 \leq R^{(i,j)}_{k,l} = R^{(i,j)}_{l,k} \leq q_{ij} - 1 \)

identifiers \( \xi, \eta \)

noise \( E^{(j)}(y) = \sum_{k=0}^{\alpha} \sum_{l=1}^{n} \epsilon_{\xi,k} y^l \), \( \sum_{j=1}^{n} \epsilon_{\xi,k} = 0 \) for \( 1 \leq j \leq n \)

keying material of node \( \xi \) coefficients of polynomials \( G^{(j)}(y) = \sum_{k=0}^{\alpha} G^{(j)}_{k} y^k \),
where \( G^{(j)}_{k} = (\sum_{i=1}^{m}(R^{(i,j)}(\xi))_{q_{ij}} + \epsilon^{(j)}_{k} N_j)^{2^B} \)

key generated by \( \xi \) for link with \( \eta \) \( K_{\xi,\eta} = (\sum_{j=1}^{n}(G^{(j)}(\eta))_{N_j})^{2^B} \)

key difference \( \delta_{\xi,\eta} \), where \( K_{\xi,\eta} = (K_{\eta,\xi} + \delta_{\xi,\eta})^{2^B} \)

bound on key difference \( \Delta \)

In the case of standard HIMMO \( (n = 1) \) the subscripts and superscripts corresponding to the HIMMO instance are omitted.
Auxiliary functions

\[ A^{(i)}_\xi (y) = \sum_{k=0}^{\alpha} (R^{(i)}_k (\xi)) q_i y^k \]

\[ \Lambda_\xi (y) = \sum_{i=1}^{m} \frac{N - q_i}{N q_i} A^{(i)}_\xi (y) \]

\[ S(x, y) = \sum_{i=1}^{m} \frac{(R^{(i)}(x, y)) q_i}{q_i} \]

\[ \lambda_\xi (y) = \Lambda_\xi (y) + \frac{1}{N} \langle G_\xi (y) \rangle_N + S(\xi, y) \]
Chapter 1

Introduction

1.1 Symmetric Key Generation

The method of key distribution discussed in this thesis was first discussed by Matsumoto and Imai [1]. Their key establishment scheme considers several nodes in a network which wish to be able to generate a shared secure key. To achieve this, they propose that a Trusted Third Party (TTP) chooses a secret symmetric function \( R(x, y) \), thus \( R(x, y) = R(y, x) \). Each node \( \xi \) receives its keying material, a function \( G_{\xi}(y) = R(\xi, y) \). Nodes \( \xi \) and \( \eta \) can then generate a shared key as

\[
K_{\xi,\eta} = G_{\xi}(\eta) = R(\xi, \eta) = G_{\eta}(\xi) = K_{\eta,\xi}. \tag{1.1}
\]

In [2], Blundo et al. choose \( R(x, y) \) to be a symmetric bivariate polynomial of degree \( \alpha \). Their complete protocol then works as follows.

1. **System initialization**
   The TTP chooses a public integer \( \alpha \) and generates a symmetric bivariate polynomial \( R(x, y) \) of degree \( \alpha \) in each variable.

2. **Node registration**
   For each node \( \xi \) that wants to register the TTP provides node \( \xi \) with the secret keying material consisting of the polynomial \( G_{\xi}(y) \), defined as

\[
G_{\xi}(y) = R(\xi, y). \tag{1.2}
\]

3. **Key agreement**
   Node \( \xi \) generates its shared key with node \( \eta \) from its keying material share as
\[ K_{\xi,\eta} = G_{\xi}(\eta) = R(\xi, \eta) = K_{\eta,\xi}. \]  

(1.3)

Node \( \xi \) can then send an encrypted message to node \( \xi \) using \( K_{\xi,\eta} \) as key. A graphic representation of the three phases is shown in Figure 1.1.

The main concern for system security here is that an attacker could compromise multiple nodes and try to reconstruct keying material using the information found in these nodes. This is known as a collusion attack.

Blundo et al. showed that their scheme provides information-theoretic security if an attacker knows the keying material of \( c \) colluding nodes as long as \( c \leq \alpha \). If \( c > \alpha + 1 \), an attacker can easily recover \( R(x, y) \) using Lagrange interpolation [2].

1.2 Outline of Thesis

As all our work is related to the HIMMO key generation scheme, we start with an explanation of the scheme in Chapter 2. This chapter contains all previous knowledge on HIMMO relevant for our contributions, including the fact that nodes in a HIMMO system make small errors when generating a key. Our contributions start in Chapter 3, where we give new insight into the behavior of key generation errors. In Chapter 4 we build on the previous chapter to drastically reduce key generation errors. Chapter 5 contains the bad news. Our insight into the behavior of key generation errors also reveals a new type of attack on HIMMO.

Chapter 6 explains HIMMO mix, which is a variation on the HIMMO scheme aimed at better protecting the keying material share of nodes. Chapters 7 and 8 explain how our contributions to HIMMO can be applied to HIMMO mix.

Finally, in Chapter 9 we explain a method that could potentially protect HIMMO from the attack described in Chapter 5. This only leaves the wrapping up in Chapter 10.
Chapter 2

The HIMMO Key Generation Scheme

In 2012, Garcia et al. [3] proposed a variation on Blundo’s classic scheme: HIMMO, a key establishment scheme which aims at full collusion resistance, that is, an attacker cannot reconstruct keying material regardless of how many nodes are compromised. To achieve this collusion resistance property, HIMMO relies on two mathematical problems: Hiding Information (HI) and Mixing Modular Operations (MMO), hence the name HIMMO.

**Hiding Information (HI) Problem:** Let \( G_\xi(y) \in \mathbb{Z}_N[y] \) be a polynomial of degree at most \( \alpha \), and let \( K_{\xi,\eta} = \langle \langle G_\xi(\eta) \rangle \rangle_2 \). Reconstruct \( G_\xi(y) \) based on \( c \) pairs \( (\eta, K_{\xi,\eta}) \).

**Mixing Modular Operations (MMO) Problem:** Let \( m \geq 2 \) and \( q_1, \ldots, q_m \) and \( N \) be different positive numbers. Let \( R_1, \ldots, R_m \) be polynomials of degree at most \( \alpha \) with \( \alpha \geq 2 \) and integer coefficients. Defining \( G(\xi) = \langle \sum_{i=1}^{m} \langle R_i(\xi) \rangle \rangle_2 \), for \( \xi \in \mathbb{Z} \), the MMO problem is to recover \( R_1, \ldots, R_m \), given \( m, \alpha, N \), and \( c \) pairs \( (\xi, G(\xi)) \).

The HI problem is related to the Noisy Polynomial Interpolation problem [4] and for this problem attacks are known for some parameters. The MMO problem is a new problem which seems to be mathematically difficult [5].

HIMMO is designed to run on resource-constrained devices, meaning both storage requirements for keying material, and the computation time of keys should be minimal.

In this chapter we will describe the basic HIMMO system.

### 2.1 System Operations

Like Blundo, HIMMO operation comprises three phases:
1. System initialization
The TTP selects five public positive integers \( m \geq 2, \alpha \geq 2, B, b \) and \( N \), an odd positive integer of \((\alpha + 1)B + b\) bits (so \(2^{(\alpha+1)B+b-1} < N < 2^{(\alpha+1)B+b}\)). Then the TTP generates the following private material: \( m \) distinct positive integers \( q_1, \ldots, q_m \) of the form \( q_i = N - 2^b \beta_i \) where \( 0 < \beta_i < 2^B \), \( m \) symmetric bivariate polynomials \( R^{(1)}(x, y), \ldots, R^{(m)}(x, y) \), all of degree at most \( \alpha \) in each variable, such that for \( 1 \leq i \leq m \) the polynomial \( R^{(i)} \) is in \( \mathbb{Z}_{q_i}[x, y] \). For \( 1 \leq i \leq m \), we write

\[
R^{(i)}(x, y) = \sum_{k=0}^{\alpha} R^{(i)}_k(y)x^k \text{ with } R^{(i)}_k(y) \in \mathbb{Z}_{q_i}[y].
\] (2.1)

2. Node registration
For each node \( \xi \in \{1, \ldots, 2^B - 1\} \) that wants to register, the TTP provides node \( \xi \) with the secret keying material consisting of the coefficients \( G_{\xi,0}, \ldots, G_{\xi,\alpha} \), defined as

\[
G_{\xi,k} = \left\langle \sum_{i=1}^{m} R^{(i)}_{k}(\xi) \right\rangle_{q_i} N.
\] (2.2)

3. Key agreement
Node \( \xi \) generates its key with node \( \eta \) from its keying material share as

\[
K_{\xi,\eta} = \left\langle \left\langle G_{\xi}(\eta) \right\rangle_{N} \right\rangle_{2^b} = \left\langle \left\langle \sum_{k=0}^{\alpha} G_{\xi,k} \eta^k \right\rangle_{N} \right\rangle_{2^b}.
\] (2.3)

The keys \( K_{\xi,\eta} \) and \( K_{\eta,\xi} \) are approximately the same. The difference between them will be bounded in the following section. Since keys are not guaranteed to be equal, a key reconciliation phase is required. Methods for key reconciliation will be described in Section 2.3.

Multiple HIMMO systems can be combined to create longer HIMMO keys. The process of combining these HIMMO keys is very simple. Suppose we want a HIMMO system with \( b = 8 \), which still generates 128-bit keys. We then use 16 HIMMO systems and concatenate the generated keys.

In general we have several HIMMO systems which generate \( b \)-bit keys. If we want concatenate these keys into a single \( l \)-bit key we need \( \lceil l/b \rceil \) HIMMO systems.
2.2 Analysis

The keys that node $\xi$ and $\eta$ generate resemble a base key defined as follows.

**Definition 2.1.** Let $0 < \xi, \eta < 2^B$. Then the base key $\overline{K}_{\xi,\eta}$ that node $\xi$ and $\eta$ aim to generate is defined as

$$\overline{K}_{\xi,\eta} = \left( \sum_{i=1}^{m} \langle R^{(i)}(\xi, \eta) \rangle_{q_i} \right)_{2^B} = \overline{K}_{\eta,\xi}. \quad (2.4)$$

We say that node $\xi$ attempts to generate $\overline{K}_{\xi,\eta}$, but makes a small error in doing so. An expression for this error is given in the following theorem.

**Theorem 2.2 (HIMMO key error).** Let

$$A^{(i)}_\xi(y) = \sum_{k=0}^{\alpha} \langle R^{(i)}_k(\xi) \rangle_{q_i} y^k. \quad (2.5)$$

Then for $0 < \xi, \eta < 2^B$ we have that

$$K_{\xi,\eta} = (\overline{K}_{\xi,\eta} + \lambda(\eta)N)_{2^B}, \quad (2.6)$$

where $\lambda(\eta)$ is the integer function

$$\lambda(\eta) = \sum_{i=1}^{m} \left[ \frac{A^{(i)}_\xi(y)}{q_i} \right] - \left[ \frac{1}{N} \sum_{i=1}^{m} A^{(i)}_\xi(y) \right]. \quad (2.7)$$

**Proof.** We have that

$$G_\xi(\eta) = \sum_{k=0}^{\alpha} \sum_{i=1}^{m} \langle R^{(i)}_k(\xi) \rangle_{q_i} \eta^k. \quad (2.8)$$

We thus have that

$$G_\xi(\eta) = \sum_{i=1}^{m} \left( \left( \sum_{k=0}^{\alpha} \langle R^{(i)}_k(\xi) \eta^k \rangle_{q_i} + q_i \left[ \frac{1}{q_i} \sum_{k=0}^{\alpha} \langle R^{(i)}_k(\xi) \rangle_{q_i} \eta^k \right] \right) \right). \quad (2.9)$$
Combining (2.1), (2.5) and (2.9), we obtain

\[ G_{\xi}(\eta) = \sum_{i=1}^{m} (R^{(i)}(\xi, \eta))_{q_i} + N \sum_{i=1}^{m} \left\lfloor \frac{A_{\xi}^{(i)}(\eta)}{q_i} \right\rfloor \right) - \sum_{i=1}^{m} (N - q_i) \left\lfloor \frac{A_{\xi}^{(i)}(\eta)}{q_i} \right\rfloor \right]. \quad (2.10) \]

Since \( \langle x \rangle_{N} = x - N \lfloor \frac{x}{N} \rfloor \) and \( G_{\xi}(\eta) = \sum_{i=1}^{m} A_{\xi}^{(i)}(\eta) \), we have that

\[ \langle G_{\xi}(\eta) \rangle_{N} = \sum_{i=1}^{m} \langle R^{(i)}(\xi, \eta) \rangle_{q_i} - \sum_{i=1}^{m} (N - q_i) \left\lfloor \frac{A_{\xi}^{(i)}(\eta)}{q_i} \right\rfloor + \lambda_{\xi}(\eta)N. \quad (2.11) \]

As \( (N - q_i)^2 = 0 \), for \( 1 \leq i \leq m \), the definition of \( K_{\xi,\eta} \) and (2.11) complete the proof.

Theorem 2.2 immediately implies an equation for the difference between \( K_{\xi,\eta} \) and \( K_{\eta,\xi} \).

**Corollary 2.3.** For \( 0 < \xi, \eta < 2^{B} \) we have that

\[ K_{\xi,\eta} = \langle K_{\eta,\xi} + \delta_{\xi,\eta}N \rangle_{2^{B}}, \quad (2.12) \]

where \( \delta_{\xi,\eta} \) is the integer

\[ \delta_{\xi,\eta} = \lambda_{\xi}(\eta) - \lambda_{\eta}(\xi). \quad (2.13) \]

The expression for \( \delta_{\xi,\eta} \) in (2.13) can be used to bound \( \delta_{\xi,\eta} \). This bound is not tight and in Chapter 3 we will provide a tighter bound.

**Theorem 2.4 (HIMMO Key Equality).** Let \( 0 < \xi, \eta < 2^{B} \) and \( \Delta = 3m - 1 \). Then

\[ \delta_{\xi,\eta} \in \{-\Delta, \ldots, \Delta\}. \quad (2.14) \]

**Proof.** As \( x - 1 < [x] \leq x \), equation (2.7) implies that

\[ -m + \sum_{i=1}^{m} \frac{N - q_i}{Nq_i} A_{\xi}^{(i)}(\eta) \leq \sum_{i=1}^{m} \frac{N - q_i}{Nq_i} A_{\xi}^{(i)}(\eta) + 1. \quad (2.15) \]
For $1 \leq i \leq m$, we have that $\frac{N - q_i}{N \eta_i} A_{\xi}^{(i)}(\eta)$ is bounded by

$$\frac{N - q_i}{N \eta_i} A_{\xi}^{(i)}(\eta) = \frac{N - q_i}{N \eta_i} \sum_{k=0}^{\alpha} (R_k^{(i)}(\xi)) q_i \eta^k \leq \frac{2^b \beta_i}{N \eta_i} \sum_{k=0}^{\alpha} (q_i - 1) \eta^k < \frac{2^{B+b}}{N} \frac{2^{(\alpha+1)B} - 1}{2^B - 1}$$

$$< \frac{2^{B+b}}{N} 2^{\alpha B} \leq 2, \quad (2.16)$$

and thus

$$\sum_{i=1}^{m} \frac{N - q_i}{N \eta_i} A_{\xi}^{(i)}(\eta) < 2m. \quad (2.17)$$

Since $N - q_i > 0$, we also have that $0 < \frac{N - q_i}{N \eta_i} A_{\xi}^{(i)}(\eta)$ for all $1 \leq i \leq m$. Substituting these bounds into (2.15) gives

$$- m + 1 \leq \lambda_{\xi}(\eta) \leq 2m. \quad (2.18)$$

Substituting (2.18) into (2.13) proves the theorem.

2.3 Key Reconciliation

We cannot guarantee nodes $\xi$ and $\eta$ generate the same key. However, since $\delta_{\xi,\eta} \in \{-\Delta, \ldots, \Delta\}$, we can guarantee that node $\xi$ finds at most $2\Delta + 1$ candidate keys for $K_{\xi,\eta}$ based on its own key $K_{\eta,\xi}$. So to ensure key agreement between nodes a key reconciliation phase is required. When node $\xi$ sends a message to node $\eta$ it needs to include enough information about $K_{\xi,\eta}$ so that node $\eta$ can find $K_{\xi,\eta}$ amongst its $2\Delta + 1$ candidate keys. In this section we discuss two known methods nodes can use for this.

2.3.1 Providing a Hash Function

In [3] the suggested method to compute key differences is a hash function. Node $\xi$ sends the value $H(K_{\xi,\eta})$ to node $\eta$ such that $H(i) \neq H(K_{\xi,\eta})$ for all potential keys $i$ different from $K_{\xi,\eta}$. An example of a possible hash function is given in [6]. When dealing with concatenated HIMMO keys there are two ways to build the hash function. Either the key from each system is hashed individually or the concatenated key is hashed.
If each key is hashed individually, the system is weak to brute force attacks. If \( b \) is small enough, an attacker can simply hash all \( 2^b \) possible keys. If the concatenated key is hashed, another problem arises. Suppose we have concatenated \( k \) HIMMO keys. In the computation of a each HIMMO key, node \( \eta \) finds \( 2\Delta + 1 \) candidate keys for \( K_{\xi,\eta} \). Thus, for the \( k \) concatenated HIMMO keys, there are \((2\Delta + 1)^k\) candidate keys. This means node \( \eta \) will have to compute up to \((2\Delta + 1)^k\) hashes to ensure key agreement.

Application of this method of key reconciliation is actually so inefficient that we will not be considering it for the rest of our thesis.

### 2.3.2 Providing Least Significant Bits

Another method node \( \xi \) can use to allow node \( \eta \) to find \( K_{\xi,\eta} \) is simply telling \( \eta \) some of the least significant bits of \( K_{\xi,\eta} \). Since node \( \eta \) needs to find \( K_{\xi,\eta} \) amongst his \( 2\Delta + 1 \) candidate keys, node \( \xi \) will need to provide at least \( t = \lceil \log_2(2\Delta + 1) \rceil \) bits of \( K_{\xi,\eta} \). In other words: \( (K_{\xi,\eta})_{2^t} \).

Node \( \eta \) solves

\[
(K_{\xi,\eta})_{2^t} = (K_{\eta,\xi} + \delta N)_{2^t}, \quad |\delta| \leq \Delta
\]

(2.19)

to find \( \delta \). It can then find \( K_{\xi,\eta} \) by computing

\[
K_{\xi,\eta} = (K_{\eta,\xi} + \delta N)_{2^t}.
\]

(2.20)

Since \( 2^t \) divides \( 2^b \), we have that

\[
K_{\xi,\eta} = K_{\eta,\xi} + \delta N \mod 2^b \Rightarrow K_{\xi,\eta} = K_{\eta,\xi} + \delta N \mod 2^t
\]

(2.21)

Because \( N \) and \( 2^t \) are coprime, (2.19) has a unique solution.

Bits which node \( \xi \) simply gives away should not be used in the final key. Node \( \xi \) provides node \( \eta \) with the \( t \) least significant bits of \( K_{\xi,\eta} \), so the key length is reduced by \( t \) bits.

This is a very expensive way of key reconciliation if HIMMO is set up with small \( b \). For example, if \( b = 32 \), after losing 4 bits in the key reconciliation phase node \( \xi \) and \( \eta \) still have a 28-bit key left. But if \( b = 8 \) and 4 bits are lost in the key reconciliation phase, half of their key is gone. This means \( \xi \) and \( \eta \) essentially wasted half of their computations.
2.4 Irrelevant bits

It turns out roughly half of the bits of keying material of a node $\xi$ hardly influence the keys $\xi$ generates. This means these bits need not be stored by $\xi$. Since HIMMO is designed to run on constrained devices, this is a nice improvement to incorporate and [3] describes how to do this. However, the irrelevance of these bits is a double edged sword. If these bits are not required by node $\xi$ to generate its keys, an attacker also does not need to retrieve these bits to reconstruct the keying material of node $\xi$.

In this section we only give some intuition as to why these bits hardly influence generated keys. For the purpose of this thesis we are only interested in the fact that this simplifies retrieving the keying material of nodes.

Let $E_\xi(y) = \sum_{k=0}^{\alpha} \epsilon_{\xi,k} y^k$ be a polynomial such that

$$0 \leq \epsilon_{\xi,k} \leq 2^{(\alpha+1-k)B+b}, \quad (2.22)$$

and

$$\langle \epsilon_{\xi,k} \rangle_{2^b} = 0 \quad (2.23)$$

for all $0 \leq k \leq \alpha$. Then for all $0 < \eta < 2^B$ we have that

$$0 \leq \epsilon_{\xi,k} \eta^k < 2^{(\alpha+1-k)B+b-1} \eta^{kB} = 2^{(\alpha+1)B+b-1} < N, \quad (2.24)$$

so

$$\langle \epsilon_{\xi,k} \eta^k \rangle_N = \epsilon_{\xi,k} \eta^k. \quad (2.25)$$

Furthermore, as $\langle \epsilon_{\xi,k} \rangle_{2^b} = 0$, we also have that

$$\langle \epsilon_{\xi,k} \eta^k \rangle_{2^b} = 0. \quad (2.26)$$

Combining (2.25) and (2.26) gives

$$\langle (\epsilon_{\xi,k} \eta^k) \rangle_{2^b} = 0. \quad (2.27)$$
Chapter 2. The HIMMO Key Generation Scheme

If we ignore carry for a moment, we find

$$\langle \langle E_\xi(y) \rangle_N \rangle_{2^b} \equiv 0, \quad (2.28)$$

and thus

$$\langle \langle G_\xi(y) + E_\xi(y) \rangle_N \rangle_{2^b} = \langle \langle G_\xi(y) \rangle_N \rangle_{2^b}. \quad (2.29)$$

Carry complicates things, but only a little. Instead of (2.29) we find

$$\langle \langle G_\xi(y) + E_\xi(y) \rangle_N \rangle_{2^b} = \langle \langle G_\xi(y) \rangle_N + \delta_N \rangle_{2^b} \quad (2.30)$$

for some small integer $\delta$; an error very similar to the error which is already made in the standard HIMMO system. For an example with $B = b = 3$ and $\alpha = 2$, we look at Figure 2.1. Blue bits in this figure are the only bits required to generate keys.

![Figure 2.1](image.png)

**Figure 2.1:** Example of relevant bits in HIMMO keying material

Thus, only $b$ least significant bits and the $kB$ most significant bits of coefficient $G_{\xi,k}$ require storage. This totals to

$$\sum_{k=0}^{\alpha} (b + kB) = (\alpha + 1)b + \alpha(\alpha + 1)B/2 \quad (2.31)$$

bits.

### 2.5 Keying Material Share Attack

For the HIMMO scheme, [7] describes an attack which allows colluding nodes to retrieve the keying material of a target node. An attacker can collect keys $K_{\eta,\xi} \approx K_{\xi,\eta_1}, \ldots, K_{\eta_c,\xi} \approx K_{\xi,\eta_c}$ and try to reconstruct $G_\xi(y)$ based on these keys. For $1 \leq i \leq c$, these keys $K_{\xi,\eta_i}$ satisfy
\[ K_{\xi,\eta_i} = \langle\langle G_{\xi}(\eta_i) \rangle\rangle_{2^b}. \quad (2.32) \]

We define the Vandermonde matrix \( V \) of size \( c \times (\alpha + 1) \) as
\[
V = \begin{pmatrix}
1 & \eta_1 & \eta_1^2 & \cdots & \eta_1^{\alpha} \\
1 & \eta_2 & \eta_2^2 & \cdots & \eta_2^{\alpha} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & \eta_c & \eta_c^2 & \cdots & \eta_c^{\alpha}
\end{pmatrix}. \quad (2.33)
\]

Let \( h \) be a column vector of length \( c \) containing the values \( K_{\xi,\eta_1}, \ldots, K_{\xi,\eta_c} \). Finding \( G_{\xi}(y) \) then becomes equivalent to finding \( r \) such that
\[
h = \langle\langle Vr \rangle\rangle_{2^b}. \quad (2.34)
\]

Using that \( \langle x \rangle_N = x - \lfloor x/N \rfloor N \) twice, we can rewrite (2.34) to
\[
h = Vr - \left[ N \left( Vr - N \left[ \frac{Vr}{N} \right] \right) \right] 2^b. \quad (2.35)
\]

For any integers \( a \) and \( b \), \( \lfloor a/b \rfloor \) is equal to the unique integer \( \lambda \) such that \( |(a/b) - \lambda - (b - 1)/(2b)| < 1/2 \). Thus we can write the problem as follows.

Given \( h \), find the integer column vectors \( r \) of length \( \alpha + 1 \), and \( \lambda_1 \) and \( \lambda_2 \) of length \( c \) such that
\[
h = Vr - \lambda_1 N - \lambda_2 2^b \quad (2.36)
\]
and
\[
\left\| \frac{Vr}{N} - \lambda_1 N - \frac{(N-1)e_c}{2N} \right\|_\infty < \frac{1}{2}, \quad (2.37)
\]
\[
\left\| 2^{-b}(Vr - N\lambda_1) - \lambda_2 - \frac{(2^b - 1)e_c}{2^{b+1}} \right\|_\infty < \frac{1}{2}, \quad (2.38)
\]
and
\[
\left\| \frac{r}{N} - \frac{(N-1)e_{\alpha+1}}{2N} \right\|_\infty < \frac{1}{2}. \quad (2.39)
\]
Here \( \mathbf{e}_i \) represents the all ones column vector of length \( i \). Now let \( \mathbf{x} \) be the integer column vector of length \( 2c + \alpha + 1 \) such that

\[
\mathbf{x} = \begin{pmatrix} \mathbf{r} \\ -\lambda_1 \\ -\lambda_2 \end{pmatrix},
\]

and let \( \mathbf{A} \) be a matrix of size \( c \times (2c + \alpha + 1) \) defined by

\[
\mathbf{A} = \begin{pmatrix} \mathbf{V} & N\mathbf{I}_c & 2^{b}\mathbf{I}_c \end{pmatrix}.
\]

Then equation (2.36) can be written as

\[
\mathbf{h} = \mathbf{A}\mathbf{x}.
\]

We define the matrix \( \mathbf{B} \) of size \( (2c + \alpha + 1)^2 \) as

\[
\mathbf{B} = \begin{pmatrix} \frac{1}{N} \mathbf{V} & \mathbf{I}_c & 0 \\ 2^{-b} \mathbf{V} & \frac{N}{2^{b}} \mathbf{I}_c & \mathbf{I}_c \\ \frac{1}{N} \mathbf{I}_{c+1} & 0 & 0 \end{pmatrix}
\]

and the column vector \( \mathbf{u} \) of length \( 2c + \alpha + 1 \) as

\[
\mathbf{u} = \begin{pmatrix} \frac{N-1}{2N} \mathbf{e}_c \\ 2^{b-1} \frac{N}{2^{b+1}} \mathbf{e}_c \\ \frac{N-1}{2N} \mathbf{e}_{\alpha+1} \end{pmatrix}.
\]

Now inequalities (2.37), (2.38) and (2.39) are equal to the inequality

\[
\| \mathbf{B}\mathbf{x} - \mathbf{u} \|_{\infty} < \frac{1}{2},
\]

Let \( \mathbf{x}_0 \) be an arbitrary solution of equation (2.42). Then every integer solution \( \mathbf{x} \) of equation (2.42) can be written as \( \mathbf{x} = \mathbf{x}_0 + \mathbf{w} \), where \( \mathbf{A}\mathbf{w} = 0 \). This means we can write equation (2.45) as

\[
\| \mathbf{y} - (\mathbf{u} - \mathbf{B}\mathbf{x}_0) \|_{\infty} < \frac{1}{2},
\]

where \( \mathbf{y} \) is the column vector of length \( 2^{b} (2c + \alpha + 1) \).
with $y \in \mathcal{L}$, where $\mathcal{L}$ equals

$$
\mathcal{L} = \{Bw \mid w \in \mathbb{Z}^{2c+\alpha+1}, \mathbf{Aw} = 0\}.
$$

All that remains is finding an integer vector that can play the role of $x_0$. Since $\gcd(N, 2^b) = 1$, we can use Euclid’s algorithm to find $\mu_0$ and $\mu_1$ such that $\mu_0 N + \mu_1 2^b = 1$. Then

$$
x_0 = \begin{pmatrix}
0_{\alpha+1} \\
\mu_0 \mathbf{h} \\
\mu_1 \mathbf{h}
\end{pmatrix}
$$

satisfies (2.42).

The keying material of node $\xi$ consists of $\alpha + 1$ coefficients of bit length $(\alpha + 1)B + b$. In Section 2.4 we have seen that only $(\alpha + 1)b + \alpha(\alpha + 1)B/2$ bits actually influence the key. As every colluding node obtains $b$ bits of information, an attacker will need to compromise at least

$$
\frac{(\alpha + 1)b + \alpha(\alpha + 1)B/2}{b}
$$

nodes to reconstruct $G_\xi(y)$.

As $c \geq \frac{(\alpha + 1)b + \alpha(\alpha + 1)B/2}{b}$, the dimension of the lattice colluding nodes need to solve is at least $\alpha + 1 + \frac{(\alpha + 1)(\alpha + 1)B + b}{2b}$. This means we can increase $\alpha$ to increase the dimension of the lattice problem an attacker has to deal with. Finding a close lattice vector is an NP-hard problem, so we can choose $\alpha$ large enough so that the lattice problem becomes infeasible. Current lattice reduction algorithms seem to be unable to handle lattice of dimension larger than approximately 500.

### 2.6 Parameter Choices

In this section we describe two common parameter choices for HIMMO and the reasoning behind them. Parameters are chosen for HIMMO based on the HI problem described in the previous section. The aim of HIMMO is full collusion resistance and that means the HI problem should not be solvable for the choice of parameters.

**High degree polynomials**

Common choices for $b$ and $B$ are $B = b = 32$, or $B = b = 64$. For these parameters, we
can let $\alpha = 30$ so that an attacker must solve a lattice problem of minimum dimension 527.

The number of polynomials mixed, $m$, only influences the difficulty of retrieving the root keying material. Since the $q_i$’s are secret there is no known attack on the root keying material and thus $m = 2$ suffices. However, it is not unthinkable $q_i$’s can be retrieved. Therefore it is sensible to choose $m$ large enough so that an attack on the root keying material is infeasible, even if the $q_i$’s are known. The lattice dimension of an attack on the root keying material described in [3] is at least $m(m + 1)(\alpha + 1)$. Choosing $m = 15$ ensures that any attack on the root keying material is infeasible.

**Small keys**

In [7] it is suggested that it is not possible to reconstruct keying material for $b = 8$ and $\alpha = 10$. Based on this observation a common way to set up HIMMO is obviously with $b = 8$, $\alpha = 10$. For the identifier size we stick with $B = 32$ or $B = 64$. If HIMMO is set up with these parameters choosing $m = 2$ or $m = 3$ is required, to allow for reasonably efficient key reconciliation, as seen in Section 2.3.
Chapter 3

Error Symmetry in HIMMO

In this chapter we take a closer look at the behavior of errors that node $\xi$ and $\eta$ make when attempting to generate $K_{\xi,\eta}$. We will see that part of the error that node $\xi$ makes in attempting to generate $K_{\xi,\eta}$ is also made by node $\eta$. This allows us to give a tighter bound for the key difference than given in Section 2.2.

We start with some intuition on how the errors behave. For this reason we show an example of key differences with $B = b = 32$, $\alpha = 6$ and $m = 4$. We fix $\xi$ and plot $\lambda_\xi(\eta)$ for 1000 random identifiers $\eta$ in Figure 3.1. We also plot $\delta_{\xi,\eta}$ for the same identifiers in Figure 3.2.

![Figure 3.1: Plot of $\lambda_\xi(\eta)$](image1)

![Figure 3.2: Plot of $\delta_{\xi,\eta}$](image2)

In Figure 3.1 we see that for this choice of $\xi$, $\lambda_\xi(\eta)$ is an integer in the range $[-3, 3]$. In Figure 3.2 we see that for this choice of $\xi$, all values $\delta_{\xi,\eta}$ lie in $[0, 2]$. One would expect this range to be a lot larger, at least as large as the range of $\lambda_\xi(\eta)$.

This small experiment suggests there is a symmetric part in the function $\lambda_\xi(y)$; part of the error that node $\xi$ makes, is also made by node $\eta$. This is indeed the case and we formalize this in Theorem 3.1.
Theorem 3.1 (HIMMO Error Symmetry). Let $\Lambda_\xi(y) \in \mathbb{Q}[y]$ be the polynomial with positive coefficients defined as

$$\Lambda_\xi(y) = \sum_{i=1}^{m} \frac{N - q_i}{Nq_i} \sum_{k=0}^{\alpha} (R_{\xi}^{(i)}(\xi)) q_i y^k,$$  \hspace{1cm} (3.1)

and let $S$ be the symmetric function

$$S(x, y) = \sum_{i=1}^{m} \frac{(R_{\xi}^{(i)}(x, y)) q_i}{q_i}.$$ \hspace{1cm} (3.2)

Then for all $0 < \xi, \eta < 2^b$ we have that

$$\lambda_\xi(\eta) = \Lambda_\xi(\eta) - S(\xi, \eta) + \frac{(G_\xi(\eta))_N}{N}. \hspace{1cm} (3.3)$$

Proof. Using that $\lfloor a/N \rfloor = a/N - \langle a \rangle N/N$, we rewrite (2.7):

$$\lambda_\xi(\eta) = \sum_{i=1}^{m} \frac{N - q_i}{Nq_i} A_{\xi}^{(i)}(\eta) - \sum_{i=1}^{m} \frac{(A_{\xi}^{(i)}(\eta)) q_i}{q_i} + \frac{1}{N} \sum_{i=1}^{m} A_{\xi}^{(i)}(\eta)_N.$$ \hspace{1cm} (3.4)

Equation (2.5) then implies

$$\sum_{i=1}^{m} \frac{(A_{\xi}^{(i)}(\eta)) q_i}{q_i} = S(\xi, \eta) \hspace{1cm} (3.5)$$

which completes the proof.  \hspace{1cm} \Box

Theorem 3.1 the following corollary.

Corollary 3.2. For all $0 < \xi, \eta < 2^B$ we have that

$$\delta_{\xi, \eta} = \Lambda_\xi(\eta) + \frac{(G_\xi(\eta))_N}{N} - \left( \Lambda_\eta(\xi) + \frac{(G_\eta(\xi))_N}{N} \right). \hspace{1cm} (3.6)$$

To see how the function $S(\xi, y)$ influences $\lambda_\xi(y)$, we plot $\lambda_\xi(\eta) - S(\xi, \eta)$ in Figure 3.3. We also plot the function $\Lambda_\xi(\eta)$ as defined in Theorem 3.1 in Figure 3.4.
We see that $\lambda_\xi(y) - S(\xi, \eta)$ is a lot more predictable than $\lambda_\xi(y)$. It is no longer integer, but it is influenced by only two terms: $(G_\xi(\eta))_N$ (the untruncated key of node $\xi$) and $\Lambda_\xi(\eta)$ (a polynomial with positive coefficients). The observation that $\lambda_\xi(y)$ contains a symmetric part will play a crucial role throughout this thesis. For now we only use it to improve the bound on $\delta_{\xi,\eta}$.

**Theorem 3.3** (HIMMO Key Equality 2). Let $0 < \xi, \eta < 2^B$ and $\Delta = 2m$. Then

$$\delta_{\xi,\eta} \in \{-\Delta, \ldots, \Delta\}. \quad (3.7)$$

**Proof.** Using Theorem 3.1 and that $0 \leq \frac{(G_\xi(\eta))_N}{N} < 1$, we find that

$$\Lambda_\xi(\eta) - \Lambda_\eta(\xi) - 1 < \delta_{\xi,\eta} < \Lambda_\xi(\eta) - \Lambda_\eta(\xi) + 1. \quad (3.8)$$

Substituting (2.17) into (3.8) completes the proof.
Chapter 4

Improving Key Equality

Keys generated by the original HIMMO scheme are not necessarily equal, and this means that a key reconciliation phase is required. If somehow HIMMO could be modified so that keys are always equal ($\Delta = 0$), the reconciliation phase could be skipped. In this section we explain how to achieve exactly this. This sounds like a major improvement, but as we will see, this modification is not without its drawbacks.

Modifying HIMMO in this manner not only significantly increases storage at nodes, but also opens up an attack on the root keying material. Therefore we also describe how nodes can achieve approximate key equality ($\Delta = 1$). This is still a considerable improvement on the original scheme and does not come with any of the drawbacks of perfect key.

4.1 Exact Key Equality

The intuition on how to achieve HIMMO with matching keys is quite simple. Node $\xi$ makes an error while computing its shared key with $\eta$. Equation (3.6) is a function which describes this error. Thus the TTP provides each node $\xi$ with this function. Now every time node $\xi$ generates its key it also computes the error it makes while generating its key. It then modifies its key to a key which contains no error. Since nodes no longer make an error while generating keys, keys are the same.

We start with a theorem showing that there exist modified HIMMO keys that are always equal. Then we show how to adapt HIMMO so that nodes can compute these modified keys.

**Theorem 4.1** (HIMMO with matching keys). Let
\[ K'_{\xi,\eta} = \langle K_{\xi,\eta} - [\Lambda_\xi(\eta) + \frac{1}{N}\langle G_\xi(\eta)\rangle_N]N\rangle_2^b. \] (4.1)

Then

\[ K'_{\xi,\eta} = K'_{\eta,\xi}. \] (4.2)

**Proof.** Since \( \lfloor x \rfloor \leq x < \lfloor x \rfloor + 1 \), Corollary (3.2) implies

\[ \lfloor \Lambda_\xi(\eta) + \frac{1}{N}\langle G_\xi(\eta)\rangle_N \rfloor - \lfloor \Lambda_\eta(\xi) + \frac{1}{N}\langle G_\eta(\xi)\rangle_N \rfloor - 1 < \delta_{\xi,\eta} \]

\[ < \lfloor \Lambda_\xi(\eta) + \frac{1}{N}\langle G_\xi(\eta)\rangle_N \rfloor - \lfloor \Lambda_\eta(\xi) + \frac{1}{N}\langle G_\eta(\xi)\rangle_N \rfloor + 1. \] (4.3)

Using that \( \delta_{\xi,\eta} \) is integer, we thus find

\[ \delta_{\xi,\eta} = \lfloor \Lambda_\xi(\eta) + \frac{1}{N}\langle G_\xi(\eta)\rangle_N \rfloor - \lfloor \Lambda_\eta(\xi) + \frac{1}{N}\langle G_\eta(\xi)\rangle_N \rfloor. \] (4.4)

Substituting (4.4) into (2.12) completes the proof.

To be able to compute matching keys, node \( \xi \) needs to be able to compute \( K'_{\xi,\eta} \) as described in Theorem 4.1 for all \( 0 < \eta < 2^B \). Node \( \xi \) can already compute \( K_{\xi,\eta} \) and \( \langle G_\xi(\eta)\rangle_N \) with the information provided in normal HIMMO. Thus, the only extra information required is the function \( \Lambda_\xi(y) \).

As \( \Lambda_\xi(y) \in \mathbb{Q}[y] \) is a polynomial of degree at most \( \alpha \), node \( \xi \) can simply store the coefficients of \( \Lambda_\xi(y) \). Let

\[ \Lambda_\xi(y) = \sum_{k=0}^{\alpha} \Lambda_{\xi,k}y^k. \] (4.5)

Then for \( 0 \leq k \leq \alpha \) we can write

\[ \Lambda_{\xi,k} = \sum_{i=1}^{m} 2^h \beta_i \text{lcm}(q_1, \ldots, q_{i-1}, q_{i+1}, \ldots, q_m) \frac{A^{(i)}_{\xi,k}}{N\text{lcm}(q_1, \ldots, q_m)}. \] (4.6)
Consequences for system security

To generate matching keys in this manner, the cost in security we have to pay is substantial.

By providing each node with $\text{lcm}(q_1, \ldots, q_m)$, we provide an attacker with a way to retrieve the secret moduli $q_i$. Even though factorization is a hard problem in general, we have quite a strong constraint on the possible values for $q_i$. There are $2^B$ candidates for $q_i$, so an attacker could use trial division to find the $q_i$’s using $2^B$ tries.

In HIMMO keeping the moduli $q_i$ secret is important. In [3] a method is described which allows colluding nodes to retrieve the root keying material if $q_1, \ldots, q_m$ are known.

Extra storage requirements

Node $\xi$ needs to store $\alpha + 2$ extra values; a single denominator and $\alpha + 1$ numerators. For $0 \leq k \leq \alpha$ the denominator of $\Lambda_{\xi,k}$ is equal to $\text{lcm}(q_1, \ldots, q_m)$. For $1 \leq i \leq m$ we have that $q_i$ is an $((\alpha + 1)B + b)$-bit number so storage for $\text{lcm}(q_1, \ldots, q_m)$ requires at most $m((\alpha + 1)B + b)$ bits.

For $1 \leq i \leq m$ we have that $2^b\beta_i$ is a $(B + b)$-bit number, $A_{\xi,k}^{(i)}$ is an $((\alpha + 1)B + b)$ bit number and $\text{lcm}(q_1, \ldots, q_{i-1}, q_{i+1}, \ldots, q_m)$ is an $((\alpha + 1)B + b)(m - 1)$-bit number. Thus, for $0 \leq k \leq \alpha$ we have that the numerator of $\Lambda_{\xi,k}$ is at most

$$B + b + (\alpha + 1)B + b + ((\alpha + 1)B + b)(m - 1) = m((\alpha + 1)B + b) + B + b \quad (4.7)$$

bits. Total storage requirements for the auxiliary function $\Lambda_\xi(y)$ thus sum up to

$$(\alpha + 1)(m((\alpha + 1)B + b) + B + b) + (m - 1)((\alpha + 1)B + b) \quad (4.8)$$

bits. We note that storage of the complete function $G_\xi(y)$ only takes $(\alpha + 1)((\alpha + 1)B + b)$ bits so storage of $\Lambda_\xi(y)$ increases keying material storage by more than a factor $m$.

Comparison to normal HIMMO

It seems that modifying HIMMO in this manner does more harm than good. Generated keys are now equal, but storage requirements at nodes are multiplied by approximately $m$. More importantly, an attacker now has all the tools he needs to retrieve the root keying material.
4.2 Approximate Key Equality

Realistically node $\xi$ cannot store the exact function $\Lambda_\xi(y)$. However, it turns out node $\xi$ can very efficiently store an approximation of $\Lambda_\xi(y)$. It can use this approximation to try and compensate for its error as much as possible. However, as node $\xi$ does not have the exact function $\Lambda_\xi(y)$ at its disposal, sometimes a small error will remain.

We start with a theorem showing that keys modified using an approximation of $\Lambda_\xi(y)$ can guarantee that $\Delta = 1$. We then provide two functions that can be used as an approximation for $\Lambda_\xi(y)$.

**Theorem 4.2 (Improved key equality).** Let $\Gamma_\xi(y)$ be any real-valued function such that

$$\Lambda_\xi(\eta) - 1 \leq \Gamma_\xi(\eta) \leq \Lambda_\xi(\eta)$$  \hspace{1cm} (4.9)

for all $0 < \eta < 2^B$. Let

$$K_{\xi,\eta}' = \langle K_{\xi,\eta} - \lfloor \Gamma_\xi(\eta) + \frac{1}{N}(G_\xi(\eta))_N \rfloor N \rangle_{2^B}.$$  \hspace{1cm} (4.10)

Then there exists a $\delta_{\xi,\eta}' \in \{-1, 0, 1\}$ such that

$$K_{\xi,\eta}' = \langle K_{\eta,\xi} + \delta_{\xi,\eta}' N \rangle_{2^B}.$$  \hspace{1cm} (4.11)

**Proof.** Combining (2.12), (4.10) and (4.11) gives

$$\delta_{\xi,\eta}' = \delta_{\xi,\eta} - \lfloor \Gamma_\xi(\eta) + \frac{1}{N}(G_\xi(\eta))_N \rfloor + \lfloor \Gamma_\eta(\xi) + \frac{1}{N}(G_\eta(\xi))_N \rfloor.$$  \hspace{1cm} (4.12)

Then (4.9) implies

$$\delta_{\xi,\eta} - \lfloor \Lambda_\xi(\eta) + \frac{1}{N}(G_\xi(\eta))_N \rfloor + \lfloor \Lambda_\eta(\xi) + \frac{1}{N}(G_\eta(\xi))_N \rfloor - 1 < \delta_{\xi,\eta}',$$  \hspace{1cm} (4.13)

and

$$\delta_{\xi,\eta}' < \delta_{\xi,\eta} - \lfloor \Lambda_\xi(\eta) + \frac{1}{N}(G_\xi(\eta))_N \rfloor + \lfloor \Lambda_\eta(\xi) + \frac{1}{N}(G_\eta(\xi))_N \rfloor + 1.$$  \hspace{1cm} (4.14)
Chapter 4. Improving Key Equality

Since \(0 \leq x - \lfloor x \rfloor < 1\) we find

\[-1 < \delta_{\xi,\eta} - \lfloor \Lambda_\xi(\eta) + \frac{1}{N} \langle G_\xi(\eta) \rangle_N \rfloor + \lfloor \Lambda_\eta(\xi) + \frac{1}{N} \langle G_\eta(\xi) \rangle_N \rfloor < 1,\]  

and thus

\[-2 < \delta'_{\xi,\eta} < 2.\]  

(4.16)

Recalling that \(\delta'_{\xi,\eta}\) is integer completes the proof.

Note that using any approximation of \(\Lambda_\xi(y)\) will at most guarantee that (4.16) changes to

\[-(1 + \epsilon) < \delta'_{\xi,\eta} < 1 + \epsilon,\]  

(4.17)

which yields the same result as our constraint on the approximation. This suggests it is not possible to guarantee \(\Delta = 0\) using only an approximation of \(\Lambda_\xi(y)\).

Like in the previous section, to allow nodes to compute their modified key \(K'_{\xi,\eta}\), the TTP provides each node \(\xi\) with a function \(\Gamma_\xi(y)\).

So what functions \(\Gamma_\xi(y)\) can the TTP choose to provide node \(\xi\) with? We show two functions that satisfy the constraints imposed by Theorem 4.5.

**Lemma 4.3.** Let \(\Gamma_\xi(y) = \lfloor \Lambda_\xi(y) \rfloor\). We then have that

\[\Lambda_\xi(\eta) - 1 < \Gamma_\xi(\eta) \leq \Lambda_\xi(\eta)\]  

(4.18)

for all \(0 < \eta < 2^B\).

*Proof.* The proof for this lemma follows immediately from the definition of the floor function.

**Storage requirements**

As \(\Lambda_\xi(\eta)\) is a polynomial with positive coefficients, it is a monotonically increasing function. Equation (2.17) shows that \(0 \leq \Lambda_\xi(\eta) < 2m\) for \(0 < \eta < 2^B\). This means node \(\xi\) can store \(\lfloor \Lambda_\xi(y) \rfloor\) using only the values \(I_k = \min\{\eta|\Lambda_\xi(\eta) > k\}\) for \(k \in \{0, 1, \ldots, 2m - 1\}\).
Each value $I_k$ is a $B$-bit identifier so $(2^m - 1)B$ bits in total are required for storage of the auxiliary function $\lfloor \Lambda_\xi(y) \rfloor$.

**Computation requirements for TTP**

Computation of the values $I_k$ is done by the TTP using bisection. Since the values $I_k$ lie in $\{1, \ldots, 2^B - 1\}$ each value $I_k$ is found in at most $B$ iterations. This means the TTP needs to do an extra $(2^m - 1)B$ computations in total.

**Final remark**

If we let $\Gamma_\xi(y) = \lfloor \Lambda_\xi(y) \rfloor$, $\Gamma_\xi(y)$ is integer, and as $0 \leq \langle G_\xi(\eta) \rangle < N$, we have

$$\left\lfloor \frac{\Gamma_\xi(\eta)}{N} \right\rfloor + \frac{1}{N} \langle G_\xi(\eta) \rangle = \Gamma_\xi(\eta).$$

This slightly simplifies the computation in (4.10).

Storage of $\Gamma_\xi(y)$ as described above actually requires a significant number of bits and there is a more efficient choice for $\Gamma_\xi(y)$. The idea behind this choice for $\Gamma_\xi(y)$ is that $\Lambda_\xi(y)$ is closely approximated solely by its coefficient for $y^\alpha$.

**Lemma 4.4.** Let $m < 2^{B-3}$, $0 \leq k \leq \alpha$. Let $a_k$ be such that

$$\sum_{k=0}^{\alpha} a_k y^k = \Lambda_\xi(y),$$

and let $r = B + b - 2$. Then for the function $\Gamma_\xi(y) = \frac{1}{N} \lfloor 2^{-r} a_\alpha \rfloor 2^r y^\alpha$, we have that

$$\Lambda_\xi(\eta) - 1 < \Gamma_\xi(\eta) < \Lambda_\xi(\eta).$$

for all $0 < \eta < 2^B$.

**Proof.** We have that

$$\Lambda_\xi(\eta) - \Gamma_\xi(\eta) = \sum_{k=0}^{\alpha} \frac{a_k}{N} \eta^k - \frac{1}{N} \lfloor 2^{-r} a_\alpha \rfloor 2^r \eta^\alpha = \sum_{k=0}^{\alpha-1} \frac{a_k}{N} \eta^k + \langle a_\alpha \rangle 2^r \eta^\alpha.$$ (4.22)

Using that $a_k < m \cdot 2^{B+b}$, we bound $\sum_{k=0}^{\alpha-1} \frac{a_k}{N} \eta^k$:

$$0 < \sum_{k=0}^{\alpha-1} \frac{a_k}{N} \eta^k < \frac{m \cdot 2^{B+b} 2^{\alpha B-1}}{N 2^{B-1}} = \frac{m \cdot 2^{B+b}}{2^{(\alpha+1)B+b-1}} 2^{(\alpha-1)B} = m \cdot 2^{1-B}.\quad (4.23)$$
Using the bound on $m$ we find

$$m \cdot 2^{1-B} < 2^{B-3} \cdot 2^{1-B} = 2^{-2}. \quad (4.24)$$

The term $\frac{1}{N} \langle a_\alpha \rangle^{2^r} \eta^\alpha$ satisfies

$$0 \leq \frac{1}{N} \langle a_\alpha \rangle^{2^r} \eta^\alpha < \frac{2^{2\alpha B}}{N} < \frac{2^{B+b-2\alpha B}}{2^{(\alpha+1)B+b-1}} = 2^{-1}. \quad (4.25)$$

Then (4.24) and (4.25) together imply

$$0 \leq \Lambda_\xi(\eta) - \Gamma_\xi(\eta) < 2^{-2} + 2^{-1} < 1. \quad (4.26)$$

Clearly, any reasonable set of parameters for HIMMO satisfies the constraint imposed on $m$ in Lemma 4.4. The most common choices for $B$ in HIMMO are $B = 32$ or $B = 64$. In the case of $B = 32$ we require $m < 2^{29}$ while choosing $m = 2^{10}$ is already absurdly high.

**Storage requirements**

The function $\Gamma_\xi(y)$ as described in Lemma 4.4 can be stored very efficiently by node $\xi$ as it only requires storage of $[2^{-r}a_\alpha]$. Since $\beta_i < 2^B$,

$$a_\alpha < m2^{B+b} = 2^{\log_2(m)+B+b}, \quad (4.27)$$

and thus

$$[2^{-r}a_\alpha] < \frac{2^{\log_2(m)+B+b}}{2^{B+b-2}} = 2^{\log_2(m)+2}. \quad (4.28)$$

This means only $[2 + \log_2(m)]$ bits are required for storage of $\Gamma_\xi(y)$. This is a negligible number of bits compared to the storage required for the keying material $\langle G_\xi(y) \rangle_N$.

### 4.3 An Alternative Method of Key Reconciliation

As explained in Section 2.3 both known methods of key reconciliation have significant disadvantages. In this chapter we propose an alternative method of key reconciliation...
using (3.6). This expression allows nodes to reconcile their key without providing any bits of the actual key. That means no key bits are lost in the key reconciliation phase.

Note that this form of key reconciliation is not in conflict with Section 4.2, where we claim generating matching keys is not possible if node $\xi$ only gets an approximation for $\Lambda_\xi(y)$. In Section 4.2 node $\xi$ and $\eta$ are required to reconstruct an integer representing their individual error without communicating. In this section node $\xi$ communicates with node $\eta$ so that $\eta$ can compute the joint error $\delta_{\xi,\eta}$.

We start with a theorem showing that some approximation of $\Lambda_\xi(\eta)$ can allow for computation of $\delta_{\xi,\eta}$. Then we provide a function which can play the role of this approximation.

**Theorem 4.5** (Computing $\delta_{\xi,\eta}$ explicitly). Let $\Gamma_\xi(y)$ be any real-valued function such that

\[
\Lambda_\xi(\eta) - \frac{1}{3} < \Gamma_\xi(\eta) \leq \Lambda_\xi(\eta)
\]  

(4.29)

for all $\eta \in \{1, \ldots, 2^B - 1\}$. Now let

\[
\rho_\xi(\eta) = 3\Gamma_\xi(\eta) + \frac{2}{N}\langle G_\xi(\eta) \rangle_N.
\]  

(4.30)

Then

\[
\delta_{\xi,\eta} = \left\lceil \frac{1}{3}([\rho_\xi(\eta)] - \rho_\eta(\xi)) - 1 \right\rceil.
\]  

(4.31)

**Proof.** Using (4.29) we bound $\rho_\xi(\eta)$:

\[
3\Lambda_\xi(\eta) + \frac{2}{N}\langle G_\xi(\eta) \rangle_N - 1 < \rho_\xi(\eta) \leq 3\Lambda_\xi(\eta) + \frac{2}{N}\langle G_\xi(\eta) \rangle_N.
\]  

(4.32)

Since $x - 1 < |x| \leq x$, we have that

\[
\rho_\xi(\eta) - \rho_\eta(\xi) - 1 < [\rho_\xi(\eta)] - \rho_\eta(\xi) \leq \rho_\xi(\eta) - \rho_\eta(\xi).
\]  

(4.33)

Combining (4.32) and (4.33) gives

\[
3\delta_{\xi,\eta} - 3 < [\rho_\xi(\eta)] - \rho_\eta(\xi) - 1 < 3\delta_{\xi,\eta}.
\]  

(4.34)
Dividing (4.34) by three and noting that $\delta_{\xi,\eta}$ is integer completes the proof.

Thus to allow explicit computation of $\delta_{\xi,\eta}$ by nodes $\xi$ and $\eta$, the TTP provides each node $\xi$ with $\Gamma_\xi(y)$. Node $\xi$ computes $\lfloor \rho_\xi(\eta) \rfloor$ and when communicates with $\eta$ it also provides $\lfloor \rho_\xi(\eta) \rfloor$. Node $\eta$ then computes $\delta_{\xi,\eta}$ as described in Theorem 4.5.

We provide a function that satisfies the constraints imposed by Theorem 4.5. The idea behind this function is the same as the function in Lemma 4.4, so these lemmas and their proofs are nearly identical.

**Lemma 4.6.** Let $m < 2^{B-5}$. Now for $0 \leq k \leq \alpha$ let $a_k$ be such that

$$\sum_{k=0}^{\alpha} a_k y^k = \Lambda_\xi(y),$$

(4.35)

and let $r = B + b - 3$. Then for the function $\Gamma_\xi(y) = \frac{1}{N} \lfloor 2^{-r} a_\alpha \rfloor 2^{r} y^\alpha$ we have that

$$\Lambda_\xi(\eta) - \frac{1}{3} < \Gamma_\xi(\eta) < \Lambda_\xi(\eta)$$

(4.36)

for all $0 < \eta < 2^B$.

**Proof.** Analogously to (4.23) we find

$$0 < \sum_{k=0}^{\alpha-1} \frac{a_k}{N} \eta^k < m \cdot 2^{1-B}.$$  

(4.37)

Using the bound on $m$ we find

$$m \cdot 2^{1-B} < 2^{B-5} \cdot 2^{1-B} = 2^{-4}.$$  

(4.38)

The term $\frac{1}{N} \langle a_\alpha \rangle 2^r \eta^\alpha$ satisfies

$$0 \leq \frac{1}{N} \langle a_\alpha \rangle 2^r \eta^\alpha < \frac{2^r 2^\alpha B}{N} < \frac{2^{B+b-3} 2^\alpha B}{2(\alpha+1) B+b-1} = 2^{-2}.$$  

(4.39)

Then (4.38) and (4.39) together imply

$$0 \leq \Lambda_\xi(\eta) - \Gamma_\xi(\eta) < 2^{-4} + 2^{-2} < \frac{1}{3},$$  

(4.40)
and thus

\[ \Lambda_\xi(\eta) - \frac{1}{3} < \Gamma_\xi(\eta) < \Lambda_\xi(y). \]  

(4.41)

**Storage requirements**

Storage of \( \frac{1}{N} \lfloor 2^{-r} a_\alpha \rfloor 2^r y^k \) is done exactly the same way as in Section 4.2. Since now \( r = B + b - 3 \), storage requirements at nodes is increased by \( \lceil \log_2 (m + 3) \rceil \) bits.

**Consequences for security**

Node \( \xi \) now provides node \( \eta \) with \( \lfloor \rho_\xi(\eta) \rfloor \). This value represents a very crude approximation of \( \Lambda_\xi(\eta) + \frac{1}{N} \langle G_\xi(\eta) \rangle_N \) and is thus difficult to use in an attack on the keying material of node \( \xi \). Nevertheless, in Chapter 5 we will see how an attacker can make use of this value to slightly improve an attack on the keying material share of node \( \xi \).

### 4.4 Combining the Methods

Our method of key reconciliation can be efficiently combined with the improved key equality in Section 4.2. As \( \lceil \log_2 (2\Delta + 1) \rceil \) bits of communication are required for key reconciliation, it is nice if we can also reduce \( \Delta \) to 1 when using our method of key reconciliation.

The intuition on how to combine these methods is very simple. Let \( \xi \) and \( \eta \) be two nodes in the HIMMO system such that \( \frac{1}{3} \lfloor \rho_{\xi,\eta} \rfloor = 10/3 \), \( \frac{1}{3} \rho_{\xi,\eta} = 1 \), and \( \delta_{\xi,\eta} = 2 \). When node \( \xi \) sends \( \lfloor \rho_{\xi,\eta} \rfloor = 10 \) to \( \eta \), it essentially says “The error I make is approximately \((10/3)N\).” Node \( \eta \) computes \( \rho_{\xi,\eta} \) and then concludes: “The error I make is approximately \( N \), so our joint error is approximately \((3 + \frac{1}{3})N - N\). Since our joint error is an integer multiple of \( N \), it must be \( 2N \).”

When using the improved key equality, node \( \xi \) would conclude: “The error I make is approximately \( 3N \), so I will subtract \( 3N \) from my key.” When combining our methods, node \( \xi \) would still subtract \( 3N \) from its key and now tell \( \eta \): “The error I make is approximately \( \frac{1}{3} \).”

We give a slight modification of Theorem 4.5.

**Theorem 4.7** (Computing \( \delta'_{\xi,\eta} \) explicitly). Let \( \Gamma_\xi(y) \) be any real-valued function such that...
Chapter 4. Improving Key Equality

\[ \Lambda_\xi(\eta) - \frac{1}{3} < \Gamma_\xi(\eta) \leq \Lambda_\xi(\eta) \]  
(4.42)

for all \( \eta \in \{1, \ldots, 2^B - 1\} \). Now let

\[ \rho'_\xi(\eta) = \rho_\xi(\eta) - 3\lfloor \rho_\xi(\eta) \rfloor. \]  
(4.43)

Then

\[ \delta'_{\xi,\eta} = \lceil \frac{1}{3}(\lfloor \rho'_\xi(\eta) \rfloor - \rho'_\eta(\xi) - 1) \rceil. \]  
(4.44)

**Proof.** Clearly, any function \( \Gamma_\xi(y) \) that satisfies (4.42) also satisfies (4.9). Therefore substituting (4.43) into (4.12) completes the proof. \qed

To combine our methods, node \( \xi \) first computes its modified key as described in Theorem 4.2. Then node \( \xi \) provides node \( \eta \) with \( \lfloor \rho'_\xi(\eta) \rfloor \). The benefit of combining the two methods is that \( \rho'_\xi(\eta) \in \{0, 1, 2\} \). Thus, nodes can find matching keys with just 2 bits of communication.

### 4.5 Comparisons

In this section we compare how our modifications to HIMMO described in this chapter fare against the original setup of HIMMO. For these comparisons we will be computing storage requirements for the keying material at nodes. We do not use the method of zeroing out bits to reduce storage, which was briefly discussed in Section 2.4. However, we note that a reduction in storage requirements also comes with a similar reduction in computation time. Reducing the number of subkeys a node has to compute by a factor \( x \) not only decreases storage requirements by (roughly) a factor \( x \), but decreases the computation time for the final key by the same factor.

Four methods will be compared.

1. \( \Delta = 2^m \): HIMMO with the error bound \( \Delta = 2^m \), using least significant bits for key reconciliation.
2. $\Delta = 1(1)$: HIMMO with improved key equality, using the error compensation function from Lemma 4.3. Least significant bits are used for key reconciliation.

3. $\Delta = 1(2)$: HIMMO with improved key equality, using the error compensation function from Lemma 4.4. Least significant bits are used for key reconciliation.

4. Key rec: HIMMO with improved key reconciliation, using the error compensation function from Lemma 4.6.

As described in Section 2.6, two strategies can be employed when choosing HIMMO parameters. Either we make the HI problem computationally infeasible by choosing a high polynomial degree $\alpha$, or we choose $b$ and $\alpha$ so that reconstruction of the keying material is impossible, regardless of the number of colluding devices.

For both strategies we choose several parameters to generate concatenated HIMMO keys and compute storage requirements for the keying material, that is: the public modulus $N$, the coefficients of $G_\xi(y)$, and the auxiliary function $\Gamma_\xi(y)$ if it is used.

**High degree polynomials**

We let $B = b = 32$, and vary the other parameters. To avoid skewed results, we generate 1024-bit keys. This key is a lot larger than HIMMO keys generally are, but it limits the influence of rounding. For example, if 2 bits are used for key reconciliation we have a 30-bit subkey and if 6 bits are used we have a 26-bit key. However, when generating a 128-bit HIMMO key, both methods perform equally, since $\lceil 128/30 \rceil = 5 = \lceil 128/26 \rceil$.

Results for this experiment are shown in Table 4.1.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>$\alpha$</th>
<th>30</th>
<th>40</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>15</td>
</tr>
<tr>
<td>$\Delta = 2m$</td>
<td></td>
<td>152</td>
<td>164</td>
</tr>
<tr>
<td>$\Delta = 1(1)$</td>
<td></td>
<td>144</td>
<td>147</td>
</tr>
<tr>
<td>$\Delta = 1(2)$</td>
<td></td>
<td>143</td>
<td>143</td>
</tr>
<tr>
<td>New key rec.</td>
<td></td>
<td>131</td>
<td>131</td>
</tr>
</tbody>
</table>

Table 4.1: HIMMO storage requirements for high degree polynomials

For these parameter choices, it doesn’t matter much whether or not improved key equality or our method of key reconciliation is used. Storage is reduced, but not by a very impressive factor. At best, if $m = 30$, our method of key reconciliation reduces storage requirements by about 14%.
Short HIMMO keys

We compute storage requirements for HIMMO with $b = 8$ and $\alpha = 10$. Results are shown in Table 4.2.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>$B$</th>
<th>32</th>
<th>64</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m$</td>
<td>2</td>
<td>15</td>
<td>32</td>
</tr>
<tr>
<td>Memory [B]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\Delta = 2m$</td>
<td>17280</td>
<td>34560</td>
<td>-</td>
</tr>
<tr>
<td>$\Delta = 1 \ (1)$</td>
<td>12144</td>
<td>14432</td>
<td>17424</td>
</tr>
<tr>
<td>$\Delta = 1 \ (2)$</td>
<td>11889</td>
<td>11897</td>
<td>11900</td>
</tr>
<tr>
<td>New key rec.</td>
<td>8648</td>
<td>8654</td>
<td>8656</td>
</tr>
</tbody>
</table>

Table 4.2: Storage requirements for HIMMO with short keys

For these parameters, any reduction to $\Delta$ greatly reduces storage requirements. Using improved key reconciliation, storage is reduced by approximately a factor 2 if $m = 2$. If $m$ is chosen larger, the reduction in storage is even more extreme. Letting $m = 32$ is actually quite desirable, as this guarantees an attack on the root keying material is infeasible, even if an attacker manages to retrieve the secret moduli $q_i$. Without our modifications it is not possible to set up HIMMO this way. If $m = 32$, then $\lceil \log(4m + 1) \rceil \geq 8$ out of 8 bits would be required for key reconciliation, leaving no actual key.
Chapter 5

An Error-based HIMMO Attack

Equation (3.6) is a dangerous equation for node $\xi$. Although it might be somewhat troublesome to retrieve, the joint error $\delta_{\xi,\eta}$ is public, and an evaluation of $\xi$’s keying material $\langle G_{\xi}(\eta) \rangle_N$ is one of the four other terms in this equation. If an attacking node $\eta$ knows or approximates the three remaining terms it will find (an approximation for) $\langle G_{\xi}(\eta) \rangle_N$. In this chapter we show how $c$ colluding nodes $\eta_1, \ldots, \eta_c$ can approximate $\langle G_{\xi}(\eta_i) \rangle_N$ for $1 \leq i \leq c$. This means an attacker can solve an easier problem than the HI problem to retrieve keying material of nodes.

5.1 Gathering information

We start with a general outline of the process. Then we will explain each step individually.

1. Each pair of attacking nodes $(\eta_i, \eta_j)$ computes $\Lambda_{\eta_i}(\eta_j) - \Lambda_{\eta_j}(\eta_i)$

2. Each attacking node $\eta_i$ finds upper and lower bounds for $\Lambda_{\xi}(\eta_i) - \Lambda_{\eta_i}(\xi)$

3. Attacking nodes collude, using information found in step 1 and 2, to find improved bounds on $\Lambda_{\xi}(\eta_i) - \Lambda_{\eta_i}(\xi)$

4. Each attacking node $\eta_i$ uses the improved upper and lower bound from step 3 to find an upper and lower bound for $\langle G_{\xi}(\eta_i) \rangle_N$

This is very straightforward. Interchanging the terms in (3.6) gives
\[ \Lambda_{\eta_i}(\eta_j) - \Lambda_{\eta_j}(\eta_i) = \delta_{\eta_i,\eta_j} - \frac{1}{N} \langle G_{\eta_i}(\eta_j) \rangle_N + \frac{1}{N} \langle G_{\eta_j}(\eta_i) \rangle_N. \quad (5.1) \]

All variables on the right side of (5.1) can be computed by nodes \( \eta_i \) and \( \eta_j \).

2. Each attacking node \( \eta_i \) finds upper and lower bounds for \( \Lambda_\xi(\eta_i) - \Lambda_\eta(\xi) \)

As \( 0 \leq \frac{1}{N} \langle G_\xi(\eta) \rangle_N < 1 \), (5.1) implies

\[ \delta_{\xi,\eta_i} + \frac{1}{N} \langle G_{\eta_i}(\xi) \rangle_N < \Lambda_\xi(\eta_i) - \Lambda_\eta(\xi) < \delta_{\xi,\eta_i} + \frac{1}{N} \langle G_{\eta_i}(\xi) \rangle_N + 1. \quad (5.2) \]

If our method of key reconciliation is used, each node \( \eta_i \) is provided with \( \lfloor \rho_\xi(\eta_i) \rfloor \). Each node \( \eta_i \) can then find a slightly tighter bound for \( \Gamma_\xi(y) \) since (4.29) and (4.30) together imply

\[ \rho_\xi(\eta_i) - \frac{1}{3} < \Gamma_\xi(\eta_i) < \rho_\xi(\eta_i) + \frac{1}{3}. \quad (5.3) \]

3. Attacking nodes collude, using the information from step 1 and 2, to find improved bounds on \( \Lambda_\xi(\eta_i) - \Lambda_\eta(\xi) \)

Before describing how to improve the bounds, we give some intuition as to why this improvement is possible. Let us assume each colluding node \( \eta_i \) knows \( \Lambda_{\eta_i}(\xi) \) exactly, which is actually not far from the truth as colluding nodes can use (5.1) to find very tight bounds for \( \Lambda_{\eta_i}(\xi) \). Then from (5.2) it finds a range of size 1 for \( \Lambda_\xi(\eta_i) \). An example of bounds found is then shown in Figure 5.1.

![Figure 5.1: Bounds found for \( \Lambda_\xi(y) \)](image-url)

Figure 5.1 shows \( \Lambda_\xi(y) \) and bounds for \( \Lambda_\xi(y) \) found by 250 colluding nodes. An attacker knows that \( \Lambda_\xi(y) \) is a polynomial with rational, positive coefficients that follows all bounds found by the colluding nodes. We can see that some bounds lie closer to \( \Lambda_\xi(y) \) than others, giving tighter constraints on coefficients of \( \Lambda_\xi(y) \). This suggests node \( \eta_i \).
could improve its bound for $\Lambda_{\xi}(\eta_i)$ by using the bounds all other nodes $\eta_j$ have gathered for $\Lambda_{\xi}(\eta_j)$.

To do this, each node $\eta_i$ creates two LPs: a maximization program and a minimization program. Let $\eta_0 = \xi$. Then for $0 \leq i \leq c$ we write

$$\Lambda_{\eta_i}(y) = \sum_{k=0}^{\alpha} f_{i,k} y^k.$$  \hspace{1cm} (5.4)

For all $0 \leq i \leq c$ and $0 \leq k \leq \alpha$, \((4.20)\) implies that $0 \leq f_{i,k} \leq m \cdot 2^{2b}$. Let $a_{ij}$ and $b_{ij}$ be the lower and upper bounds found for $\Lambda_{\eta_i}(\eta_j) - \Lambda_{\eta_j}(\eta_i)$. Each colluding node $\eta_\ell$ can now find an improved lower bound for $\Lambda_{\xi}(\eta_\ell) - \Lambda_{\eta_\ell}(\xi)$ using the following LP:

**minimize** \[ \sum_{k=0}^{\alpha} \left( f_{0,\ell} \eta_j^k - f_{\ell,0} \eta_i^k \right) \] \hspace{1cm} (5.5)

**subject to** \[ a_{ij} \leq \sum_{k=0}^{\alpha} \left( f_{i,k} \eta_j^k - f_{j,k} \eta_i^k \right) \leq b_{ij}, \text{ for all } 0 \leq i, j \leq c, i \neq j \] \hspace{1cm} (5.6)

$0 \leq f_{i,k} \leq m \cdot 2^{2b}$, for $0 \leq i \leq c, 0 \leq k \leq \alpha$ \hspace{1cm} (5.7)

In this LP, \((5.6)\) is the embodiment of the bounds found for $\Lambda_{\eta_i}(\eta_j) - \Lambda_{\eta_j}(\eta_i)$. Equation \((5.7)\) represents the bounds on the coefficients of all polynomials $\Lambda_{\xi}(y)$. Completely analogously to this LP, an improved upper bound for $\Lambda_{\xi}(\eta_\ell) - \Lambda_{\eta_\ell}(\xi)$ is found by

**maximize** \[ \sum_{k=0}^{\alpha} \left( f_{0,\ell} \eta_j^k - f_{\ell,0} \eta_i^k \right) \] \hspace{1cm} (5.8)

**subject to** \[ a_{ij} \leq \sum_{k=0}^{\alpha} \left( f_{i,k} \eta_j^k - f_{j,k} \eta_i^k \right) \leq b_{ij}, \text{ for all } 0 \leq i, j \leq c, i \neq j \] \hspace{1cm} (5.9)

$0 \leq f_{i,k} \leq m \cdot 2^{2b}$, for $0 \leq i \leq c, 0 \leq k \leq \alpha$ \hspace{1cm} (5.10)

Note that these LPs need to be computed for each colluding node, resulting in $2c$ linear programs an attacker needs to solve.

Lemma \([4.4]\) showed that $\Lambda_{\xi}(y)$ is very closely approximated solely by its coefficient for $y^\alpha$. An attacker can also use this insight to simplify the LP’s he solves by ignoring the lower order coefficients. Furthermore, both LPs can be split up into two smaller LP’s if an attacker first computes bounds for $\Lambda_{\eta_i}(\xi)$ for all $1 \leq i \leq c$ and then inserts these
bounds into an LP to find bounds for \( \Lambda_\xi(\eta_i) \). However, the LP’s described above are already easy to solve, so an attacker does not need to simplify them.

4. **Each attacking node** \( \eta_i \) **uses the improved upper and lower bound from step 3 to find an upper and lower bound for** \( \langle G_\xi(\eta_i) \rangle_N \)

This step is again really straightforward. Node \( \eta_i \) simply substitutes the bounds found for \( \Lambda_\xi(\eta_i) - \Lambda_\eta_i(\xi) \) into (3.6).

We obviously want to know how well each node \( \eta_i \) can approximate \( \langle G_\xi(\eta_i) \rangle_N \). We give a lower bound on the expected size of the range colluding nodes will find the using the following lemma. For this we assume all \( \frac{1}{N}\langle G_\xi(\eta) \rangle_N \) are uniformly distributed in the range \([0, 1)\).

**Lemma 5.1.** Let \( x_1, \ldots, x_n \) be i.i.d. distributed from uniform distribution \( U[0, 1) \), and let

\[
\begin{align*}
x_{\text{min}} &= \min_{1 \leq i \leq n} x_i \\
x_{\text{max}} &= \max_{1 \leq i \leq n} x_i.
\end{align*}
\]

Then

\[
E[x_{\text{min}}] = \frac{1}{n + 1} \quad \text{and} \quad E[x_{\text{max}}] = \frac{n}{n + 1}.
\]

For a proof we refer to [8].

Lemma 5.1 implies that in expectation, the best upper bound and the best lower bound both lie at a distance of \( 1/(c + 1) \) from \( \langle G_\xi(y) \rangle_N \). This means the expected best bound that \( c \) colluding nodes find for any \( \Lambda_\xi(\eta_i) \) will not be lower than \( 2/(c + 1) \).

5.2 **Keying Material Share Attack, Revisited**

After having found some extra information about \( \langle G_\xi(\eta_i) \rangle_N \) an attacker will want to use this information to more efficiently retrieve the keying material of node \( \xi \). To get a clean formulation of the problem at hand, we assume an attacker has \( s \) most significant bits of each key at his disposal.

This is slightly difference from the information an attacker finds in the previous section, as a range of size \( 2^{-s}x \) for \( x \) does not guarantee access to even one most significant bit of \( x \). However in most cases a range of size \( 2^{-s}x \) will provide almost \( s \) most significant bits.
The information each colluding node $\eta_i$ now has at its disposal is given by

$$h_i = \langle G_\xi(\eta_i) \rangle_N^b,$$

and

$$g_i = \left\lfloor \frac{\langle G_\xi(\eta_i) \rangle_N}{2^{s'}} \right\rfloor,$$

where $s' = (\alpha + 1)B + b - s$, so that $g_i$ contains the $s$ most significant bits of $\langle G_\xi(\eta_i) \rangle_N$.

We start with explaining what extra constraints are imposed by the information each node $\eta_i$ finds about $\langle G_\xi(\eta_i) \rangle_N$. Let

$$g = \begin{pmatrix} g_1 & \cdots & g_c \end{pmatrix}^T.$$

Then $g$ must satisfy

$$g = \left\lfloor \frac{\langle Vr \rangle_N}{2^{s'}} \right\rfloor.$$

As $x = [x/N]N$, (5.14) can be rewritten to

$$g = \left\lfloor \frac{Vr - \left\lfloor \frac{Vr}{N} \right\rfloor N}{2^{s'}} \right\rfloor.$$

We formulate this problem similarly to the way (2.35) was reformulated. The problem becomes to find $\lambda_2$ such that

$$g = \lambda_2,$$

where

$$\left\| \frac{Vr}{2^{s'}} - \frac{\lambda_1}{2^{s'}} - \lambda_2 - \frac{2^{s'} - 1}{2^{s'+1}} \right\|_\infty < \frac{1}{2}.$$

Note that (5.16) is quite subtle. We could also write down the constraints for $g$ without introducing $\lambda_2$, but formulating (5.17) in this manner will keep the dimension of $\mathcal{L}$ in (5.19) minimal.
We can now set up a lattice similar to the lattice described in Section 2.5. The resulting problem is as follows. Find $y \in \mathcal{L}$ satisfying

$$
\|y - (u - Bx_0)\|_{\infty} < \frac{1}{2},
$$

where

$$
\mathcal{L} = \{ Bw | w \in \mathbb{Z}^{3c+\alpha+1}, Aw = 0 \}.
$$

The vectors $h$ of length $2c$, $x$ of length $3c + \alpha + 1$, and $u$ of length $3c + \alpha + 1$ are given by

$$
h = \begin{pmatrix} h_1 & \ldots & h_c & g_1 & \ldots & g_c \end{pmatrix}^T,
$$

$$
x = \begin{pmatrix} r \\ \lambda_0 \\ \lambda_1 \\ \lambda_2 \end{pmatrix},
$$

and

$$
u = \begin{pmatrix} \frac{N-1}{2N} e_{\alpha+1} \\ \frac{2^\alpha-1}{2^{\alpha+1}} e_c \\ \frac{N-1}{2N} e_c \\ \frac{2^\alpha-1}{2^{\alpha+1}} e_c \end{pmatrix}.
$$

$e_i$ represents the all ones column vector of length $i$. The matrices $A$ of size $2c \times (3c+\alpha+1)$ and $B$ of size $(3c + \alpha + 1) \times (3c + \alpha + 1)$ are given by

$$
A = \begin{pmatrix} V & NI_c & 2^\beta I_c & 0 \\ 0 & 0 & 0 & I_c \end{pmatrix},
$$

and
Chapter 5. An Error-Based HIMMO Attack

\[ \mathbf{B} = \begin{pmatrix} \frac{1}{N} \mathbf{V} & \mathbf{I}_c & 0 & 0 \\ 2^{-b} \mathbf{V} & \frac{N}{2^b} \mathbf{I}_c & \mathbf{I}_c & 0 \\ \frac{1}{N} \mathbf{I}_{\alpha+1} & 0 & 0 & 0 \\ 2^{-b} \mathbf{V} & \frac{N}{2^b} \mathbf{I}_c & 0 & -\mathbf{I}_c \end{pmatrix}. \] (5.24)

A valid target vector \( \mathbf{x}_0 \) is given by

\[ \mathbf{x}_0 = \begin{pmatrix} 0_{\alpha+1} \\ \mu_0 \mathbf{h} \\ \mu_1 \mathbf{h} \\ \mathbf{g} \end{pmatrix}, \] (5.25)

with \( \mu_0 \) and \( \mu_1 \) as in (2.48). As \( \dim \ker \mathbf{A} = 2c \), the dimension of \( \mathcal{L} \) is equal to \( (3c + \alpha + 1) - 2c = (c + \alpha + 1) \).

5.3 Experiments

As noted in Section 2.6 there are two strategies that can be employed when setting up HIMMO. If a high polynomial degree \( \alpha \) is chosen, lattice dimensions will be massive ensuring it is infeasible to retrieve keying material of nodes. Having access to a few more bits of information will barely influence the number of colluding nodes required, so lattice dimensions will remain massive. So for these systems, our attack does (almost) nothing.

However, when HIMMO is set up with \( b = 8 \) and \( \alpha = 10 \), lattice dimensions are a lot smaller. Therefore we will focus our experiments on trying to break HIMMO with this choice of parameters. We let \( B = 16 \) to keep computations manageable.

First, we see how many most significant bits can be retrieved using the attack from Section 5.1. Since the number of bits an attacker finds varies we do 100 attacks for all \( c = 2^k, k = 2, \ldots, 10 \). For each \( c \) we then plot the average number of bits found in the best and worst attack in Figure 5.2. We also plot the average number of bits found over all attacks.

Like the bound based on Lemma 5.1 suggests, the number of bits an attacker expects to find increases logarithmically in \( c \). Furthermore, although on average an attacker finds roughly \( \log_2(c) - 1 \) most significant bits, he can get lucky and find a lot more. Further experiments indicate that the number of bits an attacker expects to find is influenced
almost exclusively by the number of colluding nodes $c$. This is also in line with the bound based on Lemma 5.1.

Next we solve the lattice problem from Section 5.2 using Babai’s rounding technique [9]. The experiment is set up as follows. We fix $s$ and find the minimum number of colluding nodes $c_{\text{min}}$ for which we observe valid polynomial reconstruction for $G_\xi(y)$. Results are shown in Table 5.1.

<table>
<thead>
<tr>
<th>$s$</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_{\text{min}}$</td>
<td>-</td>
<td>84</td>
<td>78</td>
<td>69</td>
<td>66</td>
<td>62</td>
</tr>
</tbody>
</table>

Table 5.1: Number of colluding nodes required to retrieve $G_\xi(y)$ with $s$ most significant bits known.

For $s = 5$ we did not manage to reconstruct $G_\xi(y)$. This does not mean reconstruction of $G_\xi(y)$ is impossible in this case as there exist more powerful algorithms than Babai’s rounding technique to solve the close vector problem. If $s = 6$ an attacker can retrieve $G_\xi(y)$ with 84 colluding nodes. Figure 5.2 shows each $\eta_i$ can also expect to find these 6 bits if $c \approx 2^7 = 128$. Thus, if an attacker colludes with $c = 128$ nodes, we expect he has enough information to retrieve $G_\xi(y)$.

If $B$ is chosen larger than 16, the dimension of the lattice problem increases. For $B = 32$, based on theoretical lower bounds, we expect an attacker has to deal with lattice problems of dimension approximately 150 to 200, depending on the number of bits $s$ he manages to find. Even though solving lattice problems of this dimension is out of range for us, it lies well below dimension 500, the minimum dimension which is deemed safe. Furthermore, since an attacker does not have to deal with the exact
HI problem using this approach, we can assume the exact same attack also works on HIMMO with $B = 32$. 
Chapter 6

The HIMMO Mix Scheme

The aim of HIMMO is full collusion resistance for both the root keying material and the keying material of individual nodes. Section 2.5 describes how colluding nodes can retrieve the keying material of individual nodes, using that retrieving the keying material of a target node is similar to the HI problem.

HIMMO mix was designed so that to retrieve the keying material of a target node, an attacker does not only need to deal with the HI problem, but also with the (seemingly much harder) MMO problem.

6.1 System Operations

Since the description for HIMMO mix contains a lot of subscripts and superscripts, we start with an intuitive description of how HIMMO mix is supposed to work as the intuition behind HIMMO mix is quite simple.

Essentially, HIMMO mix is just $n$ HIMMO systems combined. To compute a HIMMO mix key, a node computes all $n$ HIMMO keys and adds these together. To ensure an attacker cannot attack each instance individually some noise is added to the coefficients of the keying material share. This noise disappears only if all $n$ HIMMO keys are added.

The formal description of HIMMO mix is as follows. To simplify notation slightly we formulate HIMMO mix with $B = b$.

System initialization
The TTP selects four public positive integers $m \geq 2$, $n \geq 2$, $\alpha \geq 2$, and $b$. Then the TTP selects $n$ public odd positive integers of $(\alpha + 2)b$ bits: $N_1, \ldots, N_n$. Next the TTP generates the following private material: $n \cdot m$ distinct positive integers $q_{11}, \ldots, q_{mn}$ of
the form \( q_{ij} = N_j - 2^b \beta_{ij} \) where \( 1 \leq \beta_{ij} \leq 2^b - 1, \) \( n \cdot m \) symmetric bivariate polynomials \( R^{(1,1)}(x, y), \ldots, R^{(m,n)}(x, y), \) all of degree at most \( \alpha \) in each variable, such that for \( 1 \leq i \leq m, 1 \leq j \leq n \) the polynomial \( R^{(i,j)} \) is in \( \mathbb{Z}_{q_{ij}}[x, y]. \) For \( 1 \leq i \leq m \) and \( 1 \leq j \leq n, \) we write

\[
R^{(i,j)}(x, y) = \sum_{k=0}^{\alpha} R_k^{(i,j)}(y)x^k \quad \text{with} \quad R_k^{(i,j)}(y) \in \mathbb{Z}_{q_{ij}}[y].
\]

**Node registration**

For each node \( \xi \in \{1, \ldots, 2^b - 1\} \) that wants to register, the TTP selects \( n(\alpha+1) \) integers \( \epsilon^{(j)}_{\xi,k} \) (henceforth called noise) such that \( |\epsilon^{(j)}_{\xi,k}| < 2^b \) for \( 0 \leq k \leq \alpha, \) and \( 1 \leq j \leq n. \) Also we require that \( \langle \sum_{j=1}^{n} \epsilon^{(j)}_{\xi,k} \rangle_{2^b} = 0 \) for all \( 0 \leq k \leq \alpha. \) The TTP provides node \( \xi \) with the secret keying material consisting of the coefficients \( G^{(1)}_{\xi,0}, \ldots, G^{(n)}_{\xi,\alpha}, \) defined as

\[
G^{(j)}_{\xi,k} = \left\langle \sum_{i=1}^{m} \langle R_k^{(i,j)}(\xi) \rangle_{q_{ij}} + \epsilon^{(j)}_{\xi,k} \rangle \right\rangle_{N_j}.\]

**Key agreement**

Node \( \xi \) generates its key with node \( \eta \) from its keying material share as

\[
K_{\xi,\eta} = \left\langle \sum_{j=1}^{n} \langle G^{(j)}_{\xi}(\eta) \rangle_{N_j} \right\rangle = \left\langle \sum_{j=1}^{n} \left\langle \sum_{k=0}^{\alpha} G^{(j)}_{\xi,k} \eta^k \right\rangle \right\rangle_{N_j}.
\]

Then for all \( 0 \leq \xi, \eta \leq 2^b - 1 \) we have that

\[
K_{\xi,\eta} \in \{ (K_{\eta,\xi} + \sum_{j=1}^{n} \delta_j N_j)_{2^b} | |\delta_j| \leq \Delta \},
\]

where \( \Delta = 3m. \)

**6.2 Analysis**

Similar to the original HIMMO scheme, keys that nodes \( \xi \) and \( \eta \) generate resemble a base key defined as follows.
Chapter 6. The HIMMO Mix Scheme

**Definition 6.1.** Let $0 < \xi, \eta < 2^b$. Then the base key $K_{\xi,\eta}$ that node $\xi$ and $\eta$ aim to generate is defined as

$$K_{\xi,\eta} = \left( \sum_{j=1}^{n} \sum_{i=1}^{m} (R_{i,j}^{(i,j)}(\xi,\eta))_{q_{ij}} \right)_{2^b} = K_{\eta,\xi}. \quad (6.1)$$

**Theorem 6.2 (HIMMO mix key error).** Let

$$A^{(i,j)}_{\xi}(y) = \sum_{k=0}^{\alpha} (R_{k}^{(i,j)}(\xi))_{q_{ij}} y^k, \quad (6.2)$$

and

$$\mathcal{E}^{(j)}_{\xi}(y) = \sum_{k=0}^{\alpha} \epsilon^{(j)}_{\xi,k} y^k. \quad (6.3)$$

Then for $0 < \xi, \eta < 2^b$ we have that

$$K_{\xi,\eta} = (K_{\xi,\eta} + \sum_{j=1}^{n} \lambda^{(j)}_{\xi}(\eta) N_j)_{2^b}, \quad (6.4)$$

where $\lambda^{(j)}_{\xi}(y)$ is the integer function

$$\lambda^{(j)}_{\xi}(y) = \sum_{i=1}^{m} \left[ \frac{A^{(i,j)}_{\xi}(y)}{q_{ij}} \right] - \left[ \frac{1}{N} \left( \sum_{i=1}^{m} A^{(i,j)}_{\xi}(y) + \mathcal{E}^{(j)}_{\xi}(\eta) \right) \right]. \quad (6.5)$$

**Proof.** We have

$$G^{(j)}_{\xi}(\eta) = \sum_{k=0}^{\alpha} \sum_{i=1}^{m} \left( (R_{k}^{(i,j)}(\xi))_{q_{ij}} + \epsilon^{(j)}_{\xi,k} \right) \eta^k. \quad (6.6)$$

Repeating the steps in (2.9), (2.10), and (2.11) we find

$$\langle (G^{(j)}_{\xi}(\eta))_{N_j} \rangle_{2^b} = \langle \sum_{i=1}^{m} (R^{(i,j)}(\xi,\eta))_{q_{ij}} + \lambda^{(j)}_{\xi}(\eta) N_j + \mathcal{E}^{(j)}_{\xi}(\eta) \rangle_{2^b}. \quad (6.7)$$

Since $\langle \sum_{j=1}^{n} \mathcal{E}^{(j)}_{\xi}(\eta) \rangle_{2^b} = 0$, applying Definition 6.1 completes the proof.
Like in HIMMO, we find an equation for the difference between $K_{\xi,\eta}$ and $K_{\eta,\xi}$.

**Corollary 6.3.** For $0 < \xi, \eta < 2^b$ we have that

$$K_{\xi,\eta} = \langle K_{\eta,\xi} + \sum_{j=1}^{n} \delta_{x,\eta}^{(j)} N \rangle_{2^b},$$

(6.8)

where $\delta_{x,\eta}^{(j)}$ is the integer

$$\delta_{x,\eta}^{(j)} = \lambda_{x}^{(j)}(\eta) - \lambda_{y}^{(j)}(\xi).$$

(6.9)

**Theorem 6.4 (HIMMO mix Key Equality).** Let $0 < \xi, \eta < 2^b$ and let $\Delta = 3m$. Then for all $1 \leq j \leq n$ holds that

$$\delta_{x,\eta}^{(j)} \in \{-\Delta, \ldots, \Delta\}.$$  

(6.10)

**Proof.** We have that

$$0 < \frac{1}{N} e_{x}(\eta) = \frac{1}{N} \sum_{k=0}^{\alpha} \xi_{x,y}^{(j)} k < \frac{2^b}{N} \sum_{k=0}^{\alpha} (2^b)^k < \frac{2^b 2^{(\alpha+1)b}}{N 2^b - 1} < \frac{2^b}{N 2^{ab}} \ll 1.$$  

(6.11)

Combining (6.5) and (6.11) implies that

$$\lambda_{x}^{(j)}(y) > \sum_{i=1}^{m} \left[ \frac{A_{x}^{(i,j)}(y)}{q_{ij}} \right] - \left[ \frac{1}{N} \sum_{i=1}^{m} A_{x}^{(i,j)}(y) \right] - 1,$$

(6.12)

and

$$\lambda_{x}^{(j)}(y) < \sum_{i=1}^{m} \left[ \frac{A_{x}^{(i,j)}(y)}{q_{ij}} \right] - \left[ \frac{1}{N} \sum_{i=1}^{m} A_{x}^{(i,j)}(y) \right].$$

(6.13)

Application of (2.18) gives

\[ \square \]
and substituting \(6.14\) into \(6.9\) completes the proof.

\[ -m \leq \lambda^{(j)}_{\xi}(\eta) \leq 2m, \quad (6.14) \]

6.3 Key Reconciliation

Key reconciliation is a major issue in HIMMO mix. We can only guarantee that

\[ K_{\xi,\eta} \in \{ \langle K_{\eta,\xi} + \sum_{j=1}^{n} \delta_j N_j \rangle_{2^b} \mid |\delta_j| \leq \Delta \quad \forall 1 \leq j \leq n \}. \quad (6.15) \]

This means node \(\eta\) finds up to \((2\Delta + 1)^n\) candidate keys for \(K_{\xi,\eta}\) based on its own key \(K_{\eta,\xi}\). Known key reconciliation methods for HIMMO mix are the methods described in Section 2.3, but their disadvantages are even more significant in HIMMO mix.

**Providing a hash function**

If node \(\xi\) provides node \(\eta\) with a hash function of \(K_{\xi,\eta}\), node \(\eta\) will need to do up to \((2\Delta + 1)^n\) computations before finding \(K_{\xi,\eta}\), even if the unconcatenated key is hashed. As HIMMO mix is designed to run on constrained devices this is not an option.

**Providing least significant bits**

As node \(\eta\) finds \((2\Delta + 1)^n\) candidate keys for \(K_{\xi,\eta}\) based on its own key, \(\xi\) will need to provide \(\eta\) with at least \(n(2\Delta + 1)\) bits of \(K_{\xi,\eta}\). We will not go in detail how \(\eta\) can actually compute \(K_{\xi,\eta}\) based on the least significant bits of \(K_{\xi,\eta}\), as this is already quite a cumbersome task. However it obviously beats computing an exponential number of hash functions.

6.4 Keying Material Share Attack

The addition of the noise polynomial \(\mathcal{E}^{(j)}_{\xi}(y)\) ensures that

\[ \langle (\langle G^{(j)}_{\eta_i}(\xi) \rangle_{N_j})_{2^b} - (\langle G^{(j)}_{\xi}(\eta_i) \rangle_{N_j})_{2^b} = (\langle \mathcal{E}^{(j)}_{\xi}(\xi) \rangle_{2^b}. \quad (6.16) \]
Since the coefficients of the noise polynomials \( c^{(j)}(y) \) are chosen uniformly at random in the range \([0, 2^b)\), node \( \eta_i \) finds no information about \( \langle G_\xi(\eta_i) \rangle_N \). That means the only information \( \eta_i \) finds about the keying material of node \( \xi \) is

\[
\left\langle \sum_{j=1}^{n} (G^{(j)}(\xi))_{N_j} \right\rangle_{2^b} \approx \left\langle \sum_{j=1}^{n} (G^{(j)}(\eta_i))_{N_j} \right\rangle_{2^b}.
\]  

(6.17)

We describe how retrieving the keying material of node \( \xi \) in HIMMO mix can be described by a lattice problem. Like the attack from Section 2.5, this attack is based on the attack described in [7]. Again, we set up the lattice similar to Section 2.5.

Let \( h \) contain the observed keys from identifiers \( \eta_1, \ldots, \eta_c \). The problem can be formulated as follows. Given \( h = \begin{pmatrix} K_{\xi,\eta_1} & \cdots & K_{\xi,\eta_c} \end{pmatrix}^T \), find integer vectors \( r_1, \ldots, r_n \) of length \( \alpha + 1 \) such that

\[
0 \leq r_j \leq N_j - 1,
\]

(6.18)

for all \( 1 \leq j \leq n \), and

\[
h = \left\langle \sum_{j=1}^{n} (V r_j)_{N_j} \right\rangle_{2^b},
\]

(6.19)

\[
h = \sum_{j=1}^{n} \left(V r_j - N_j \left\lfloor \frac{V r_j}{N_j} \right\rfloor \right) - 2^b \left[ \sum_{j=1}^{n} \left(V r_j - N_j \left\lfloor \frac{V r_j}{N_j} \right\rfloor \right) \right].
\]

(6.20)

Similarly to the way 2.35 was reformulated, we can formulate this problem as follows.

Given \( h \), find integer column vectors \( r_1, \ldots, r_n \) of length \( \alpha + 1 \) and \( \lambda_0, \ldots, \lambda_n \) of length \( c \) such that

\[
h = -2^b \lambda_0 + \sum_{j=1}^{n} (-N_j \lambda_j + V r_j),
\]

(6.21)

\[
\left\| \sum_{j=1}^{n} \frac{V r_j - N_j \lambda_j}{2^b} - \lambda_0 - \frac{2^{b-1} E_2}{2^{b+1}} \right\|_\infty < \frac{1}{2},
\]

(6.22)

and for \( 1 \leq j \leq n \)
Chapter 6. The HIMMO Mix Scheme

\[ \left\| \frac{Vr_j - \lambda_j}{N_j} - \frac{(N_j - 1)e_c}{2N_j} \right\|_\infty < \frac{1}{2}, \quad (6.23) \]

and

\[ \left\| \frac{r_j}{N_j} - \frac{(N_j - 1)e_{n+1}}{2N_j} \right\|_\infty < \frac{1}{2}, \quad (6.24) \]

\( e_i \) represents the all ones column vector of length \( i \). We concatenate \(-\lambda_0, \ldots, -\lambda_n\) and \( r_1, \ldots, r_{n+1} \) into an integer column vector \( x \) of length \((n + 1)c + n(\alpha + 1)\) as

\[ x^T = \left( -\lambda_n^T \ldots -\lambda_1^T \ r_1^T \ldots r_{n+1}^T \right). \quad (6.25) \]

We define \( A \) of size \( c \times ((n + 1)c + n(\alpha + 1)) \) based on (6.17) as

\[ A = \begin{pmatrix} 2^b I_c & N_1 I_c & \ldots & N_n I_c & V \ldots & V \end{pmatrix}. \quad (6.26) \]

Next, we define \( B \) of size \(((n + 1)c + n(\alpha + 1))^2\) as

\[ B = \begin{pmatrix} I_c & \frac{N_1}{2^b} I_c & \ldots & \frac{N_n}{2^b} I_c & 2^{-b} I_c & \ldots & 2^{-c} I_c \\ 0 & I_c & \ldots & 0 & \frac{1}{N_1} V & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & I_c & 0 & \ldots & \frac{1}{N_n} V \\ 0 & 0 & \ldots & 0 & \frac{1}{N_1} I_{n+1} & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 0 & 0 & \ldots & \frac{1}{N_n} I_{n+1} \end{pmatrix}, \quad (6.27) \]

and the column vector \( u \) of length \((n + 1)c + n(\alpha + 1)\) as

\[ u^T = \left( \frac{2^{b+1} - 1}{2^{2^b}} e_c^T \ N_1^{b+1} e_c^T \ldots \ N_n^{b+1} e_c^T \ \frac{N_1 - 1}{2^{N_1}} e^{T}_{n+1} \ldots \ \frac{N_n - 1}{2^{N_n}} e^{T}_{n+1} \right). \quad (6.28) \]

The problem can then be formulated as follows. Find \( y \in L \) satisfying

\[ \left\| y - (u - Bx_0) \right\|_\infty < \frac{1}{2}, \quad (6.29) \]

where
\[ \mathcal{L} = \{ \mathbf{Bw} | \mathbf{w} \in \mathbb{Z}^{((n+1)c+n(\alpha+1))}, \mathbf{Aw} = 0 \}. \] (6.30)

A valid choice of target vector \( \mathbf{x}_0 \) is

\[ \mathbf{x}_0 = \left( \mu_0 \mathbf{h}^T \mu_1 \mathbf{h}^T \mathbf{0}_{(n-1)c+n(\alpha+1)}^T \right)^T, \] (6.31)

where \( \mu_0 2^h + \mu_1 N_1 = 1 \). We can again use Euclid’s algorithm to compute \( \mu_0 \) and \( \mu_1 \).

As \( \dim \ker \mathbf{A} = c \), the dimension of the resulting lattice is equal to \( (n+1)c+n(\alpha+1) - c = n(c+\alpha+1) \). Since the number of colluding nodes required depends linearly on \( n \), the lattice dimension grows quadratically in \( n \). Storage requirements of keying material at nodes only increases linearly in \( n \). We also recall that both storage requirements and lattice dimension grow quadratically in \( \alpha \). This means that for HIMMO mix we want to choose \( \alpha \) as small as possible, and increase \( n \) to increase the dimension of the lattice that colluding nodes use to retrieve keying material.
Chapter 7

HIMMO Modifications in HIMMO Mix

As seen in the previous chapter, key generation errors are a large problem in HIMMO mix. In this section we show how our methods of dealing with HIMMO errors can be applied in HIMMO mix.

Since HIMMO mix is essentially \( n \) HIMMO systems, the error symmetry from Chapter 3 will also hold in each of these systems. This implies that in each of these systems we can apply improved key equality and our method of key reconciliation.

However, we do not exactly have \( n \) HIMMO systems, but systems with a slight modification in the least significant bits. As we will see this does not influence the feasibility of our methods, but it does make for some annoying rewriting.

7.1 Improving Key Equality

We start by rewriting Corollary 3.2 for HIMMO mix.

**Theorem 7.1 (HIMMO Mix Error Symmetry).** Let \( 0 < \xi, \eta < 2^b \) and let

\[
\Lambda_{\xi}^{(j)}(y) = \sum_{k=0}^{\alpha} \left( \sum_{i=1}^{m} \left( \frac{N - q_{ij}}{Nq_{ij}} R_{k}^{(i,j)}(\xi)q_{ij} \right) + \epsilon_{\xi,k}^{(j)} \right) y^{k}
\]

(7.1)

for all \( 1 \leq j \leq n \). Then for each of these \( j \), we have that

\[
\delta_{\xi,\eta}^{(j)} = \Lambda_{\xi}^{(j)}(\eta) + \frac{1}{N}(G_{\xi}^{(j)}(\eta))_{N} - \left( \Lambda_{\eta}^{(j)}(\xi) + \frac{1}{N}(G_{\eta}^{(j)}(\xi))_{N} \right).
\]

(7.2)
Proof. Similarly to (3.4), rewriting (6.5) gives

$$\lambda_{\xi}^{(j)}(y) = \frac{N - q_i}{N q_i} A_{\xi}^{(i,j)}(y) - \frac{1}{N} E_{\xi}^{(j)}(y) + \sum_{i=1}^{m} \frac{\langle R^{(i,j)}(\xi, \eta) \rangle_{q_{ij}}}{q_{ij}} + \frac{1}{N} \langle G_{\xi}^{(j)}(\eta) \rangle_{N}. \quad (7.3)$$

Substituting (7.3) into (6.9) completes the proof.

The expression for a function $\Lambda_{\xi}^{(j)}(y)$ is almost completely identical to the expression for $\Lambda_{\xi}(y)$ in normal HIMMO. The difference is an almost negligible term $\frac{1}{N} E_{\xi}^{(j)}(y)$, which is bounded by the following lemma.

**Lemma 7.2.** Let $0 < \xi < 2^b$. Then for all $1 \leq j \leq n$, $0 < \eta < 2^b$ we have that

$$\frac{1}{N} E_{\xi}^{(j)}(\eta) < 2^{1-ab}. \quad (7.4)$$

**Proof.** We have that

$$E_{\xi}^{(j)}(\eta) = \sum_{k=0}^{\alpha} e_{\xi,k}^{(j)} \eta^k < \frac{1}{N} \sum_{k=0}^{\alpha} 2^{2b} < 2^{1-ab} \quad (7.5)$$

We use Lemma 7.2 to reduce the bound of $\Delta = 3m$ like we did for HIMMO in Chapter 3.

**Theorem 7.3 (HIMMO Mix Key Equality).** Let $0 < \xi, \eta < 2^b$ and $\Delta = 2m + 1$. Then for all $1 \leq j \leq n$ we have that

$$\delta_{\xi,\eta}^{(j)} \in \{-\Delta, \ldots, \Delta\}. \quad (7.6)$$

**Proof.** Theorem 3.3 shows that

$$|\delta_{\xi}^{(j)}(\eta) + E_{\xi}^{(j)}(\eta) - E_{\eta}^{(j)}(\xi)| \leq 2m. \quad (7.7)$$

Inserting the bound from Lemma 7.2 for $E_{\xi}^{(j)}(\eta)$ and $E_{\eta}^{(j)}(\xi)$ completes the proof.
Lemma 7.2 shows that the introduction of noise only influences $\Lambda^{(j)}_j(y)$ by a negligible amount. Thus, we can use the improved key equality and our method of reconciliation exactly on each HIMMO system as described in Chapter 4.

Especially our method of key reconciliation from Section 4.3 deserves attention here. The computation of $\rho^{(j)}_j(\eta)$ is not influenced by values of the other HIMMO systems, meaning each $\delta^{(j)}_{j,\eta}$ can be computed independently of the others. This is a major improvement over the other methods of key reconciliation. When computing $K_{\xi,\eta}$, instead of having to choose between $(2\Delta + 1)^n$ possible keys, node $\eta$ only needs to do $n$ very basic computations. On top of that no key bits are lost.

### 7.2 Comparisons

To see just how much our modifications influence storage requirements at nodes, we again show some examples.

Like in Section 4.5, the four methods we compare are:

1. $\Delta = 2m + 1$: HIMMO with the error bound $\Delta = 2m + 1$, using least significant bits for key reconciliation.
2. $\Delta = 1(1)$: HIMMO with improved key equality, using the error compensation function from Lemma 4.3 Least significant bits are used for key reconciliation.
3. $\Delta = 1(2)$: HIMMO with improved key equality, using the error compensation function from Lemma 4.4 Least significant bits are used for key reconciliation.
4. **Key rec**: HIMMO with improved key reconciliation, using the error compensation function from Lemma 4.6

For each of these methods we set up HIMMO mix systems which generate 128-bit keys. We let $b = 32$ and vary the other parameters.

**Varying the polynomial degree**

We fix $n = 2$ and vary the polynomial degree. We do this for $m = 2$ and $m = 15$. Results are shown in Table 7.1.
For these parameter choices, our modifications reduce storage by a considerable amount. For $m = 2$, a storage reduction of about 33% is achieved. Again, if $m$ is chosen 15, which is very desirable, we have reduced storage by about 43%.

### Varying the number of HIMMO systems

We now fix $\alpha = 5$ and vary the number of HIMMO systems. We do this for $m = 2$ and $m = 15$. Results are shown in Table 7.2.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>2</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m = 2m + 1$</td>
<td>6272</td>
<td>-</td>
</tr>
<tr>
<td>$\Delta = 1$ (1)</td>
<td>5088</td>
<td>13568</td>
</tr>
<tr>
<td>$\Delta = 1$ (2)</td>
<td>4716</td>
<td>12576</td>
</tr>
<tr>
<td>New key rec.</td>
<td>3146</td>
<td>6292</td>
</tr>
</tbody>
</table>

**Table 7.2:** HIMMO mix storage requirements for $\alpha = 5$

We could analyze the exact storage reduction for some of these systems, but there is no need to as the results are obvious. Without our modifications, choosing $n > 2$ for HIMMO mix is a bad idea. Applying improved key equality reduces storage requirements by a decent amount, but setting up HIMMO mix with large $n$ is still very expensive. We can conclude here that our method of key reconciliation is *required* to make HIMMO mix a realistic system to set up with $n > 2$. 

---

<table>
<thead>
<tr>
<th>Parameters</th>
<th>$\alpha$</th>
<th>5</th>
<th>10</th>
<th>20</th>
<th>5</th>
<th>10</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory [B]</td>
<td>$\Delta = 2m + 1$</td>
<td>2352</td>
<td>6912</td>
<td>23232</td>
<td>2744</td>
<td>8064</td>
<td>27104</td>
</tr>
<tr>
<td></td>
<td>$\Delta = 1$ (1)</td>
<td>2120</td>
<td>5920</td>
<td>19520</td>
<td>3160</td>
<td>6960</td>
<td>20560</td>
</tr>
<tr>
<td></td>
<td>$\Delta = 1$ (2)</td>
<td>1965</td>
<td>5765</td>
<td>19365</td>
<td>1968</td>
<td>5768</td>
<td>19368</td>
</tr>
<tr>
<td></td>
<td>New key rec.</td>
<td>1573</td>
<td>4613</td>
<td>15493</td>
<td>1575</td>
<td>4615</td>
<td>15495</td>
</tr>
</tbody>
</table>

**Table 7.1:** HIMMO mix storage requirements for $n = 2$
Chapter 8

The Error-based Attack for HIMMO Mix

HIMMO mix was designed so that an attacker cannot target individual HIMMO systems. Thus, assuming no errors are made, the only information an attacking node $\eta$ should find about the keying material of node $\xi$ is

$$\left\langle \sum_{j=1}^{n} (G_{\eta}(\xi))_{N_{j}} \right\rangle_{2^{b}} = \left\langle \sum_{j=1}^{n} (G_{\eta}(\xi))_{N_{j}} \right\rangle_{2^{b}}. \tag{8.1}$$

That means node $\eta$ sees only the $b$ least significant bits of the sum of all HIMMO keys generated by $\xi$. However, in Chapter 5 we described an attack where colluding nodes could derive most significant bits of keys generated by another node in the HIMMO system.

The only modification made to HIMMO systems in HIMMO mix is adding noise to hide least significant bits of the key. Since the most significant bits are not modified, the exact same attack on a single HIMMO system in HIMMO mix retrieves the exact same number of most significant bits.

8.1 Obtainable Bits

Suppose an attacker has managed to retrieve the $s$ most significant bits of an untruncated key $\langle G_{\xi}(\eta) \rangle_{N}$. We now see what information about the keying material is actually in these bits. To make our life a little easier here, we ignore carry.

By definition,
\[(G_{\xi}(\eta))_N = \sum_{k=0}^{\alpha} G_{\xi,k} \eta^k. \quad (8.2)\]

As \(\eta\) is a \(b\)-bit identifier and each coefficient \(G_{\xi,k}\) is \((\alpha + 2)b\) bits, \(G_{\xi,k} \eta^k\) is a \((\alpha + 2 + k)b\) bit number. An attacker can only see bits of \(G_{\xi,k}\) with index larger than \(2^{(\alpha + 2)b - s}\), implying (if we ignore carry) he can only find information on bits of \(G_{\xi,k}\) with index at least

\[(\alpha + 2 + k)b - ((\alpha + 2)b - s) = kb + s. \quad (8.3)\]

Figure 8.1 shows keying material bits of a single HIMMO system which influence 2 most significant bits of \(\langle G_{\xi}(\eta) \rangle_N\) in blue.

![Figure 8.1: Keying material bits which colluding nodes can retrieve using only most significant key bits](image)

If we now take another look at Figure 2.1, we see that it is not possible to completely retrieve keying material of a node in HIMMO using only most significant bits of several untruncated keys \(\langle G_{\xi}(\eta) \rangle_N\). This is because no information about the \(b\) least significant bits of coefficients \(G_{\xi}(y)\) can be seen in the most significant bits (ignoring carry). However, in HIMMO mix there are fewer bits that need to be retrieved.

### 8.2 Required Bits

We take another look at the way the keying material of node \(\xi\) is generated, specifically at the way the noise is chosen. For each coefficient \(G_{\xi,k}^{(j)}\), the TTP adds noise \(0 \leq \epsilon_{\xi,k}^{(j)} < 2^b\), and the only constraint on the noise is that \(\langle \sum_{j=1}^{n} \epsilon_{\xi,k}^{(j)} \rangle_2^b = 0\). This means for the first \(n - 1\) systems, it does not matter which noise is chosen. There is always some way to choose noise in the \(n\)-th system so that the noise cancels out. Since noise is added to the \(b\) least significant bits of each coefficient, an attacker can choose these bits arbitrary for the first \(n - 1\) systems and only worry about computing them for the \(n\)-th system.

In Section 2.4 we described which bits are irrelevant in a normal HIMMO system. These bits are still irrelevant in HIMMO mix for the same reason. Thus, for HIMMO system
Chapter 8. The Error-based Attack for HIMMO Mix

$j, j = 1, \ldots, n - 1$, only the $j_b$ most significant bits are required to generate keys. Figure 8.2 shows these bits in blue for a HIMMO system with $b = 3$ and $\alpha = 2$.

$$G(\xi) \theta, 0 \quad G(\xi) \theta, 1 \quad G(\xi) \theta, 2$$

\[ j = 1, \ldots, n - 1 \]

\[ j = n \]

\[ G(\xi) \eta, 0 \quad G(\xi) \eta, 1 \quad G(\xi) \eta, 2 \]

Figure 8.2: Example of relevant bits in HIMMO mix keying material

This suggests that for the first $n - 1$ systems, an attacker can actually retrieve the keying material of node $\xi$ using only most significant bits of several $\langle G(\xi) \eta, i \rangle_{N_j}$.

### 8.3 Attacking the First $n - 1$ Systems

Attacking the first $n - 1$ systems can be done by solving yet another lattice problem. Now, for $j = 1, \ldots, n - 1$, we want to retrieve $G(\xi) \eta, j$ using only $s$ most significant bits of several evaluations $G(\xi) \eta, j$. We have actually done all the work for this in Section 5.2 where we use both most significant bits and least significant bits in the lattice. We only need to remove the parts where least significant bits are used.

Let $g = \left( g_1 \ldots g_c \right)$, where for $1 \leq i \leq c$,

\[ g_i = \left\lfloor \frac{\langle G(\xi) \eta, i \rangle_{N_j}}{2^{s'}} \right\rfloor, \quad (8.4) \]

where $s' = (\alpha + 1)B + b - s$. The problem can then be formulated as follows. Find $y \in \mathcal{L}$ satisfying

\[ \| y - (u - Bx_0) \|_\infty < \frac{1}{2}, \quad (8.5) \]

where
\[ \mathcal{L} = \{ Bw | w \in \mathbb{Z}^{2c+\alpha+1}, Aw = 0 \}. \] (8.6)

The column vector \( x \) of length \( 2c + \alpha + 1 \), \( A \) of dimension \( c \times (2c + \alpha + 1) \), \( B \) of size \( (2c + \alpha + 1) \times (2c + \alpha + 1) \), and \( u \) of length \( (2c + \alpha + 1) \) are as given by

\[
x = \begin{pmatrix} r \\ \lambda_1 \\ \lambda_2 \end{pmatrix}, \quad (8.7)
\]

\[
A = \begin{pmatrix} 0_{c+\alpha+1} & I_c \end{pmatrix}, \quad (8.8)
\]

\[
B = \begin{pmatrix} \frac{1}{N} V & I_c & 0 \\ \frac{1}{N} I_{\alpha+1} & 0 & 0 \\ 2^{-s} V & \frac{N}{2^s} & -I_c \end{pmatrix}, \quad (8.9)
\]

and

\[
u = \begin{pmatrix} \frac{N-1}{2N} e_{\alpha+1} \\ \frac{N-1}{2N} e_{c} \\ 2^{s-1} e_c \end{pmatrix}. \quad (8.10)
\]

e_i represents the all ones column vector of length i. A valid target vector \( x_0 \) is given by

\[
x_0 = \begin{pmatrix} 0_{c+\alpha+1} \\ g \end{pmatrix}. \quad (8.11)
\]

Since \( \text{dim ker A} = c \), we again end up with a lattice of dimension \( c + \alpha + 1 \).

### 8.4 Attacking the Last System

Let \( \overline{G}_\xi^{(j)}(y) \) be the solution an attacker found for \( G_\xi^{(j)}(y) \) for \( n = 1, \ldots, n - 1 \). Then to find node \( \xi \)'s HIMMO mix keying material, he needs to find \( \overline{G}_\xi^{(n)}(y) \) such that

\[
\left\langle \sum_{j=1}^{n} \overline{G}_\xi^{(j)}(\eta) \right\rangle_{N_j} = K_{\xi,\eta}, \quad (8.12)
\]
for all $0 < \eta < 2^b$. We rewrite (8.12) to

$$\langle \langle \bar{G}_\xi^{(n)}(\eta) \rangle_N \rangle_2^b = \langle K_{\xi,\eta} - \sum_{j=1}^{n-1} \langle G_j^{(j)}(\eta) \rangle_{N_j} \rangle_2^b.$$  (8.13)

Since an attacker colluded with $c$ nodes $\eta_1, \ldots, \eta_c$ to find most significant bits of $\langle G_\xi(\eta_i) \rangle_{N_j}$, he can also compute $K_{\xi,\eta_i}$ for all these nodes. This implies that he essentially has $c$ evaluations of $\langle \langle \bar{G}_\xi^{(n)}(\eta) \rangle_N \rangle_2^b$. In other words: $c$ keys out of a standard HIMMO system. This means that for the last HIMMO system, an attacker can solve the HI problem using the attack from Section 2.3.

### 8.5 Experiments

We focus our experiments on HIMMO mix with $b = 16$. Like in Section 8.5, we see how many most significant bits we can find using the attack from Section 5.1. Next we show the minimum number of colluding nodes $c_{\text{min}}$ for which successful polynomial reconstruction of $G_j^{(j)}(y)$ was observed using Babai’s rounding technique [9]. From there we conclude how many nodes need to collude to make the full attack work. We start with trying to retrieve keying material for HIMMO mix set up with $\alpha = 2$ and work our way up from there. When retrieving most significant bits, for each system we do 100 attacks like in Section 8.5.

#### $\alpha = 2$

The number of most significant bits an attacker finds with $c$ colluding nodes is given in Figure 8.3 and the minimum number of colluding nodes $c_{\text{min}}$ required to retrieve keying material is shown in Table 8.1.

<table>
<thead>
<tr>
<th>$s$</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_{\text{min}}$</td>
<td>-</td>
<td>21</td>
<td>14</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
</tr>
</tbody>
</table>

**Table 8.1:** Number of colluding nodes required to retrieve HIMMO mix keying material with $\alpha = 2$

We see that retrieving HIMMO mix keying material can be done with very few nodes if $\alpha = 2$. 16 nodes can expect to find roughly 3 most significant bits. If 3 most significant bits are known, 21 nodes can retrieve $G_j^{(j)}(y)$ for all $1 \leq j \leq n$, regardless of $n$. Thus, 21 nodes is expected to be enough to break HIMMO mix with these parameters.

#### $\alpha = 3$

The number of most significant bits an attacker finds with $c$ colluding nodes is given in
Retrieving HIMMO mix keying material is still fairly easy if $\alpha = 3$. 32 nodes can expect to find 4 most significant bits. If 4 most significant bits are known, 37 nodes can retrieve $G(j)(y)$ for all $1 \leq j \leq n$, regardless of $n$. Thus, 37 nodes is expected to be enough to break HIMMO mix with these parameters.
\( \alpha = 4 \)

The number of most significant bits an attacker finds with \( c \) colluding nodes is given in Figure 8.5 and the minimum number of colluding nodes \( c_{\text{min}} \) required to retrieve keying material is shown in Table 8.3.

![Figure 8.5: Number of most significant bits an attacker finds with \( \alpha = 4 \)](image)

<table>
<thead>
<tr>
<th>( s )</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>( c_{\text{min}} )</td>
<td>-</td>
<td>34</td>
<td>26</td>
<td>22</td>
</tr>
</tbody>
</table>

Table 8.3: Number of colluding nodes required to retrieve HIMMO mix keying material with \( \alpha = 4 \)

The attack works a lot less efficient if \( \alpha = 4 \). We did not manage to retrieve keying material with less than \( s = 8 \) most significant bits known. Even with \( s = 8 \), reconstruction of \( G^{(j)}(y) \) often failed. However, 512 nodes can still expect to find approximately 8 most significant bits, so these 512 colluding nodes could still manage retrieve \( G^{(j)}(y) \) for all \( 1 \leq j \leq n \), regardless of \( n \). Note that number of colluding nodes required can vary a lot here, as it depends almost solely on the number of most significant bits an attacker can find.

\( \alpha \geq 5 \)

For \( \alpha \geq 5 \) we were unable to reconstruct any keying material using our attack. This does not imply HIMMO mix with these parameters is secure, as colluding nodes \( \eta_1, \ldots, \eta_c \) can still retrieve most significant bits of \( G^{(j)}(\eta_h) \). However, as reconstructing \( G^{(j)}(y) \) using only the \( s \) most significant bits greatly resembles the HI problem, it seems reasonable to assume our attack no longer works if \( \alpha \) is chosen large enough. What “large enough” means remains unclear.
Chapter 9

Improved Key Equality and the Error-based Attack

In this section we analyze how the attack from Chapter 5 performs when nodes apply the improved key equality proposed in Section 4.2. The attack from Chapter 5 retrieves information about the keying material of node $\xi$ based on the information $\xi$ gives away in the key reconciliation phase. In the original HIMMO scheme we have that $\Delta = 2^m$. Thus, node $\xi$ needs to give away at least $\lceil \log_2(4m + 1) \rceil$ bits of information during the key reconciliation phase. If nodes apply improved key equality, node $\xi$ only gives away 2 bits of information. This obviously means an attacker has less information to work with and this suggests the attack should become harder. Unfortunately this is not really the case and this can easily be seen from a plot of bounds an attacker finds, shown in Figure 9.1.

![Figure 9.1: An example of bounds an attacker finds for $\Lambda_\xi(\eta)$](image)

The left side Figure 9.1 contains an example of bounds an attacker can find for $\Lambda_\xi(\eta)$ for a HIMMO system with $B = b = 32$. Since the error that node $\xi$ makes is quite small, $\Lambda_\xi(y)$ only has a few discontinuities. Once enough bounds are found, it is obvious where these discontinuities occur and how bounds for $\Lambda_\xi(\eta)$ should be retrieved.
We note that as $y$ increases, discontinuities in $\Lambda'_\xi(y)$ start occurring more often. For this reason we also see what happens when we choose identifiers slightly larger than $2^{32}$, up to $2^{34}$.

**Figure 9.2:** Another example of bounds an attacker finds for $\Lambda'_\xi(\eta)$

Figure 9.2 shows that errors quickly grow massive for identifiers larger than $2^{32}$. Errors are now bounded by $5 \times 10^6$, meaning $\Lambda'_\xi(y)$ has roughly $5 \times 10^6$ discontinuities. If somehow node $\xi$ can still compensate for this error efficiently, this could make retrieval of $\Lambda'_\xi(\eta)$ a lot harder.

That the improved key equality described in Section 4.2 can actually be used to achieve this.

We give a variation on Lemma 4.4.

**Lemma 9.1.** Let $0 \leq k \leq \alpha$ and let $a_k$ be such that

$$
\sum_{k=0}^{\alpha} a_k y^k = \sum_{i=1}^{m} \frac{N - q_i}{q_i} \Lambda^{(i)}_\xi(y),
$$

and let $r = b + \lceil \log(m) \rceil$. Then for the function $\Gamma_\xi(y) = \frac{1}{N} [2^{-r} a_\alpha] 2^r \eta^\alpha$ we have that

$$
\Lambda_\xi(\eta) - 1 < \Gamma_\xi(\eta) < \Lambda_\xi(\eta),
$$

for all $0 < \eta < 2^{B(1+\frac{1}{\alpha}) - \frac{\log(m)}{\alpha} - \frac{2}{\alpha}}$.

**Proof.** We have that

$$
\Lambda_\xi(\eta) - \Gamma_\xi(\eta) = \sum_{k=0}^{\alpha} \frac{a_k}{N} \eta^k - \frac{1}{N} [2^{-r} a_\alpha] 2^r \eta^\alpha = \sum_{k=0}^{\alpha-1} \frac{a_k}{N} \eta^k + \langle a_\alpha \rangle 2^r \eta^\alpha.
$$

We bound $\sum_{k=0}^{\alpha-1} \frac{a_k}{N} \eta^k$ as follows.
\[0 < \sum_{k=0}^{\alpha-1} \frac{a_k}{N} \eta^k < \sum_{k=0}^{\alpha-1} \frac{m2^{B+b}}{N} \left(2^{B(1+\frac{1}{\alpha})} - \log_2(m) - 2\right) = \sum_{k=0}^{\alpha-1} \frac{2^\alpha + 2B + b - 2}{N} < 2^{-1}. \quad (9.4)\]

The term \(\frac{1}{N}(a_\alpha)_2\eta^0\) then satisfies

\[0 \leq \frac{1}{N}(a_\alpha)_2\eta^\alpha < \frac{2^r}{N} \left(2^{B(1+\frac{1}{\alpha})} - \log_2(m) - \frac{2}{\alpha}\right) = \frac{1}{N} 2^{r+\alpha B + B - \log_2(m) - 2} = \frac{2^{(\alpha + 1)B + b - 2}}{N} < \frac{1}{2} \quad (9.5)\]

Then (4.24) and (9.5) together imply

\[0 \leq \Lambda_\xi(\eta) - \Gamma_\xi(\eta) < 2^{-1} + 2^{-1} = 1. \quad (9.6)\]

Lemma 9.1 shows that our method of improved key equality can be applied for identifiers larger than \(2^B\). For example, if we let \(B = 32\), \(m = 15\) and \(\alpha = 10\), we can efficiently compensate errors for any identifier \(\eta\) in (approximately) the range \((0, 2^{34.7})\). This means we could set up HIMMO with identifiers only in the range \((2^{34.7} - 2^{32}, 2^{34.7})\). For this setup of HIMMO our exact attack described in Section 5.1 no longer works. It is not known if a variation on the attack from Section 5.1 can still be used to retrieve some information about the most significant bits of \(\langle G_\xi(\eta)\rangle_N\) for these large choices of \(\eta\).
Chapter 10

Conclusions

10.1 Key Equality

One of the main problems with HIMMO was the fact that nodes make an error when generating their shared key. We have proposed a solution for this problem letting node $\xi$ store an auxiliary function $\Lambda_\xi(y)$ which represents the error it makes. However, the extra storage requirements and loss of security do not make this a practical solution.

Letting node $\xi$ store an approximation of $\Lambda_\xi(y)$ as proposed in Lemma 4.4 is a very efficient solution for reducing the key difference. At a negligible cost of $2 + \log_2 m$ bits in storage, this reduces the error bound to $\Delta = 1$.

At the cost of another extra single bit of storage, we have also proposed a new method of key reconciliation. Before the introduction of our method, the only reasonably usable method was providing least significant bits of generated keys as a method of key reconciliation. However, this method wastes computation power at nodes, which is not desirable since HIMMO is meant to run on constrained devices.

If HIMMO is set up to generate large keys ($b = 32$), applying our method of key reconciliation yields a reduction in storage requirements between 10% and 15%, depending on other parameter choices. If HIMMO is set up to generate small keys ($b = 8$), applying our method of key reconciliation reduces storage by at least 50%.

Our modifications are even more important in HIMMO mix. Until our method of key reconciliation was introduced, setting up HIMMO mix with anything but very small parameters was simply not possible. Using our method of key reconciliation the key generation errors can efficiently be dealt with for any choice of parameters.
10.2 Keying Material Share Attack

Our attack in Chapter 5 has shown that retrieving the keying material of node $\xi$ with colluding nodes is not completely identical to the HI problem, as was previously assumed. Colluding nodes can retrieve bits of $\xi$’s untruncated keys, other than the $b$ least significant bits. This means the keying material of node $\xi$ can be retrieved with fewer colluding nodes than previously thought.

It was thought to be impossible to retrieve the keying material of a node in HIMMO set up with parameters $b = 8$, $\alpha = 10$. However, as an attacker has access to more information than previously thought, this is not the case. We show how approximately 128 nodes can retrieve keying material of a HIMMO system with $B = 16$, $b = 8$ and $\alpha = 10$.

We have also shown how almost the exact same attack also retrieves keying material of individual HIMMO systems in HIMMO mix if $\alpha \leq 4$. We must thus take great care in choosing any kind of parameters for HIMMO mix. Choosing $\alpha \leq 4$ makes it very easy for colluding nodes to retrieve the keying material of another node, regardless of the number of combined HIMMO systems. If $\alpha$ is chosen larger than 4 our attack did not manage to retrieve keying material. This however does not mean HIMMO mix systems with these parameters are secure. To make any kind of statement on this, more research is required.

This leaves only HIMMO with high degree polynomials as a secure system. The extra information an attacker can find only barely reduces the dimension of the lattice problem he has to solve. Since high degree polynomials means high dimension lattices, this problem is still too large to solve.

However, all hope is not lost for HIMMO with short keys and HIMMO mix. We have shown in Chapter 9 how our method of improved key equality can efficiently hide key generation errors which at least stops our attack from retrieving more information. This does not mean no attack can work, so this method needs further research before being applied.
Bibliography


