HIMMO Security

Oscar García-Morchón, Ronald Rietman, Ludo Tolhuizen, Philips
Group Innovation, Research, Eindhoven, The Netherlands;
oscar.garcia,ronald.rietman,ludo.tolhuizen@philips.com
Domingo Gómez-Pérez and Jaime Gutiérrez, University of Cantabria,
Santander, Spain; domingo.gomez,jaime.gutierrez@unican.es

1 Introduction

This paper describes HIMMO, an identity-based pairwise symmetric key establishment method. The acronym "HIMMO" is derived from two interpolation problems that are essential for the security of the scheme: the HI problem [6], which is related to the well-known noisy interpolation problem, and the apparently novel MMO problem [5].

HIMMO is non-interactive: nodes in a network can directly generate a common key without exchanging messages, saving valuable round-trip time. Each node in the network has an identifier, and a trusted third pay (TTP) provides it with secret keying material—linked to the node identifier—in a secure way. A node that wishes to communicate with another node uses its own secret keying material and the identity of the other node to generate a common pairwise key. HIMMO allows for efficient operation with respect to both the amount of stored keying material and the key computation time, which is especially relevant for resource-constrained devices. It has similar operational characteristics as previous ID-based symmetric key establishment methods, but has superior resistance against attacks in which multiple colluding or compromised nodes co-operate to obtain information on keys between other non-colluding or non-compromised nodes.

The paper is organized as follows. In Section 2, we describe an ID-based pairwise symmetric key establishment method in full generality, give a known example, and define the types of attack against an ID-based symmetric key establishment scheme that we will consider in this paper. Next, in Section 3, we present the HI problem and the MMO problem. Then, in Section 4 we present the HIMMO scheme for pairwise symmetric key establishment. In Section 5, we validate the HIMMO scheme in that we show that the generated keys indeed are pairwise the same. In Section 6, we investigate the role of the HI and MMO problems in attacks on HIMMO. In Section 7, we describe the experimental results of an implementation of one such attack. On the one hand, the number of colluding nodes must be sufficiently large to allow to construct keys between a pair of non-colluding nodes. On the other hand, the accuracy of the approximation algorithm we use in this implementation is diminishing in the number of colluding nodes. If the number of colluding nodes required for key construction is so large that the accuracy of the approximation algorithm is too small, then this attack cannot be successful. We give a recommendation for parameter choices in HIMMO that make at least this attack infeasible. In Section 8, we draw conclusions and indicate directions for further research.

2 Definitions

In this section, we describe an ID-based pairwise symmetric key establishment scheme, provide a well-known example, and discuss the types of attack against such a scheme considered in this paper.

An Identity-based symmetric key establishment scheme $\mathcal{E}$ is specified by three algorithms: Setup, Keying Material Extraction, and Key Generation.

- **Setup:** takes a security parameter k and returns public system parameters and secret root keying only known to the Trusted Third Party (TTP) of the system.
- **Keying Material Extraction:** executed by the TTP, this algorithm takes as input the system parameters, root-keying material, and an arbitrary identifier η in the identifier space $\mathcal{M}$ and returns a secret keying material share linked to η.
- **Key Generation:** Key generation executed by node ξ with node η takes ξ's secret keying material and public identifier η as input to Key Generation algorithm to obtain $K_{\xi,\eta}$ in a key space $\mathcal{K}$.

The scheme must be such that for all $\xi, \eta \in \mathcal{M}$, $K_{\eta,\xi}$ and $K_{\xi,\eta}$ are exactly or approximately equal.

A straightforward ID-based scheme has been described by Blundo et al. [3]. The secret root keying material $R(x, y)$ is a symmetric polynomial of degree α with coefficients from $\mathbb{Z}_p$, say $R(x, y) = \sum_{i=0}^{\alpha} \sum_{j=0}^{\alpha} R_{j,k} x^j y^k$ with $R_{j,k} = R_{k,j}$. The Keying Material Extraction algorithm takes as input $R(x, y)$ and an identifier $\xi \in \mathbb{Z}_p$, and generates, for $0 \leq k \leq \alpha$,

$$G_{\xi,k} = \langle \sum_{j=0}^{\alpha} R_{j,k} \xi^j \rangle_p,$$

where $\langle x \rangle_p$ denotes the integer in $\{0, \ldots, p-1\}$ such that $x \equiv \langle x \rangle_p \bmod p$. The coefficients $G_{\xi,0}, \ldots, G_{\xi,\alpha}$ constitute the secret keying material of node ξ. In the key generation phase, node ξ computes its key $K_{\xi,\eta}$ with η as

$$K_{\xi,\eta} = \langle \sum_{k=0}^{\alpha} G_{\xi}(\eta) \rangle_p.$$

As $K_{\xi,\eta} = \langle G(\xi, \eta) \rangle_p$, and G is symmetric, it follows that $K_{\xi,\eta} = K_{\eta,\xi}$ for all identifiers ξ and η.

Blundo's scheme is fast and requires little storage. Unfortunately, the entire network can be compromised by simple interpolation using the keying material of any $\alpha + 1$ colluding nodes [3]. Also, the keying material of a singe node ξ can be obtained by simple interpolation of the keys of any $\alpha+1$ colluding nodes with ξ. HIMMO aims to keep the favorable operational characteristics of Blundo's scheme while improving collusion resistance by making interpolation or other standard algebraic techniques infeasible.

In the security analysis of this paper, we will consider the situation that an adversary has obtained the secret keying material shares linked to c distinct nodes $\eta_1, \ldots, \eta_c$. The adversary aims to find the key $K_{\xi,\eta}$ between two nodes ξ and η, neither of which is in the set $\{\eta_1, \ldots, \eta_c\}$. The adversary has two options for doing so.

1) The adversary finds a root keying material that is consistent with the keying materials of nodes $\eta_1, \ldots, \eta_c$. From this root keying material, he calculates the keying material of node ξ, and from that he calculates $K_{\xi,\eta}$.
   Note that the adversary does not need to recover the actual root keying material, but just some information that is consistent with, the operation of the actual root keying material.
2) The adversary finds a keying material for node ξ that is consistent with the keys $K_{\eta_i,\xi}$ that he can calculate from the keying materials of nodes $\eta_1, \ldots, \eta_c$. From the keying material for node ξ he calculates $K_{\xi,\eta}$.

The adversary hopes that the consistency requirements in either of these scenarios limit the possible outcomes of his guess for $K_{\xi,\eta}$, as this increases his odds for finding the correct key. On the other hand, the designer wants to create a system with the property that for each set $\{\eta_1, \ldots, \eta_c\}$, it is computationally infeasible for an adversary to guess $K_{\xi,\eta}$ with significantly better odds than a random guess. As stated before, with the Blundo scheme, both attacks work if $c \geq \alpha + 1$, where α is the x- degree of $R(x, y)$.

3 HI AND MMO PROBLEMS

In this section, we formally describe two mathematical problems that are relevant in attacks on HIMMO, namely the Hiding Information (HI) problem and the Mixing Modular Operations (MMO) problem.

Hiding Information Problem Let N, s and α be positive integers, let I be a sub-interval of $\{0, 1, \ldots, N-1\}$. The HI problem in (N, s, α, I) is as follows:

Problem 1. *Given $x \in I$ and the set $J = \left\{ \left(x_i, \left\langle \langle f(x_i) \rangle_N \right\rangle_s \right) \mid 1 \leq i \leq c \right\}$, where $\{x_1, \ldots, x_c\} \subset I$, and $f \in \mathbb{Z}_N[y]$ has degree at most α, compute $\left\langle \langle f(x) \rangle_N \right\rangle_s$.*

Note that Problem 1 does *not* require reconstruction of the coefficients of f, but for a *given* x, reconstruction of $\left\langle \langle f(x) \rangle_N \right\rangle_s$.

As shown in [6], the HI problem is equivalent to a **noisy polynomial interpolation problem** on the interval $\mathcal{I}$. Contrary to the conventional settings of the noisy polynomial interpolation problem, in our case, the interval I over which we interpolate can be a (small) subset of $\{0, 1, \ldots, N-1\}$. In [6], this problem was studied and shown to be equivalent to an approximation problem in a certain lattice.

Mixing Modular Operations problem One of the new components of HIMMO is mixing modular operations (MMO). We present the corresponding problem using the same notation as in the HI problem, i. e. N, m and α are positive integers and I is a sub-interval of $\{0, 1, \ldots, N-1\}$. Also, let $q_1 < \ldots < q_m$ be distinct positive integers. The MMO problem in $(N, m, q_1, \ldots, q_m, \alpha, I)$ is as follows:

Problem 2. *Given $x \in I$ and the set*

$$J = \big\{(x_1, h(x_1)), \ldots, (x_c, h(x_c))\big\}$$

where $\{x_1, \ldots, x_c\} \subset I$, and for all y,

$$h(y) = \sum_{i=1}^{m} \big\langle g^{(i)}(y) \big\rangle_{q_i}$$

for some unknown polynomials $g^{(1)}, \ldots, g^{(m)} \in \mathbb{Z}[X]$ of degree at most α, compute $h(x)$.

Additionally, we define the MMO problem with unknown moduli as the same interpolation problem, with the difference that the values of the moduli q_i are not given.

The MMO problem has been introduced in [5]. There it was shown that the MMO problem with known moduli is equivalent to a lattice problem of the same type as for the HI problem, whereas no solution is known for MMO with unknown moduli.

4 Description of the HIMMO system

In this section, we describe the HIMMO system. We use the following notation: for each integer x and positive integer M we define $\langle x \rangle_M$ as follows:

$$\langle x \rangle_M \text{ is the integer } y \in \{0, 1, \ldots, M-1\} \text{ such that } x \equiv y \bmod M. \qquad (1)$$

As any ID-based symmetric key establishment scheme, HIMMO system requires a trusted third party (TTP) and has three phases.

In the set-up phase, the TTP obtains from a system designer positive integers B, b, m and α, where $m \geq 2$. The number B is the bit length of the identifiers that will be used in the system, while b denotes the bit length of the keys that will be generated. The TTP randomly generates the public modulus N, an odd number of length exactly $(\alpha+1)B + b$ bits (so $2^{(\alpha+1)B+b-1} < N < 2^{(\alpha+1)B+b}$). It also randomly generates m distinct secret integers $\beta_1, \ldots, \beta_m$ with $0 \leq \beta_i < 2^B$ and at least one odd β_i. For $1 \leq i \leq m$, the secret modulus q_i is defined as $q_i = N - 2^b \beta_i$. Finally, the TTP generates

Table 1
Notation *Berry suggested to remove it, because it is repeated in the text*

$\langle x\rangle_M$ ($x \in \mathbb{Z}$, $M \in \mathbb{N}_{>0}$)	the integer $y \in \{0, 1, \ldots, M-1\}$ such that $x \equiv y \pmod{M}$
ID size in bits	B
key size in bits	b
polynomial degree in all variables	α
public modulus	odd number N of exactly $(\alpha+1)B+b$ bits
number of secret moduli	m
secret moduli	$q_1, \ldots, q_m$, where $q_i = N - 2^b\beta_i$ and $0 \leq \beta_i < 2^B$
secret root keying material	coefficients of m symmetric polynomials $R^{(i)}(x,y)$ $R^{(i)}(x,y) = \sum_{j=0}^{\alpha}\sum_{k=0}^{\alpha} R_{j,k}^{(i)}x^jy^k = \sum_{k=0}^{\alpha} R_k^{(i)}(x)y^k$, with $0 \leq R_{j,k}^{(i)} = R_{k,j}^{(i)} \leq q_i - 1$
identifiers	ξ, η
keying material of node ξ	coefficients of polynomial $G_\xi(y) = \sum_{k=0}^{\alpha} G_{\xi,k}y^k$, where $G_{\xi,k} = \left\langle \sum_{i=1}^{m} \langle R_k^{(i)}(\xi)\rangle_{q_i} \right\rangle_N$
key generated by ξ for link with η	$K_{\xi,\eta} = \left\langle \langle G_\xi(\eta)\rangle_N \right\rangle_{2^b}$

the secret root keying material, that consists of the coefficients of m bi-variate symmetric polynomials of degree at most α in each variable. For $1 \leq i \leq m$, the i-th root keying polynomial $R^{(i)}(x,y)$ is written as

$$R^{(i)}(x,y) = \sum_{j=0}^{\alpha}\sum_{k=0}^{\alpha} R_{j,k}^{(i)}x^jy^k \text{ with } 0 \leq R_{j,k}^{(i)} = R_{k,j}^{(i)} \leq q_i - 1.$$

In the keying material extraction phase, the TTP provides to each node ξ in the system, with $0 \leq \xi < 2^B$, the coefficients of the key generating polynomial G_ξ:

$$G_\xi(y) = \sum_{k=0}^{\alpha} G_{\xi,k}y^k \text{ where } G_{\xi,k} = \langle \sum_{i=1}^{m} \langle \sum_{j=0}^{\alpha} R_{j,k}^{(i)}\xi^j\rangle_{q_i}\rangle_N. \tag{2}$$

In the key generation phase, a node ξ wishing to communicate with node η with $0 \leq \eta < 2^B$, computes

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

With examples, it can be shown that $K_{\xi,\eta}$ and $K_{\eta,\xi}$ are not always equal. However, as will be shown in Section 5, for all identifiers ξ and η with $0 \leq \xi, \eta \leq 2^B$,

$$K_{\xi,\eta} \in \{\langle K_{\eta,\xi} + jN\rangle_{2^b} \mid 0 \leq\mid j \mid\leq \Delta\} \text{ with } \Delta = 2m.$$

In order to perform key reconciliation , i.e. to make sure that ξ and η use the same key to protect their future communications, the initiator of the key generation (say node ξ) sends to the other node, simultaneously with an encrypted message, information on $K_{\xi,\eta}$ that enables node η to select $K_{\xi,\eta}$ from the candidate set $C = \{\langle K_{\eta,\xi} + jN\rangle_{2^b} \mid 0 \leq |j| \leq \Delta\}$. No extra

communication thus is required for key reconciliation. The key $K_{\xi,\eta}$ will be used for securing future communication between ξ and η.
As an example of information used for key reconciliation, node ξ sends a hash-value H of $K_{\xi,\eta}$. Node η computes the hash value for all candidates in C and selects as common key the element of C with hash value H. This method incurs additional delay as node η needs to compute hashes for all candidates, and compare them with H. Of course, b should be so large that brute-force attacks on the has function are infeasible.
Alternatively, node ξ sends to node η the number $r = \langle K_{\xi,\eta}\rangle_{2^s}$, where $s = \lceil \log_2(2\Delta+1)\rceil$. Node η can efficiently obtain the integer j such that $|j| \leq \Delta$ and $K_{\xi,\eta} \equiv K_{\eta,\xi} + jN \bmod 2^b$ by using that $jN \equiv K_{\xi,\eta} - K_{\eta,\xi} \equiv r - K_{\eta,\xi} \bmod 2^s$. As r reveals the s least significant bits of $K_{\xi,\eta}$, only the $b-s$ most significant bits $K_{\xi,\eta}$ should be employed as the key, that is, the number $\lfloor 2^{-s}K_{\xi,\eta}\rfloor$.

5 Validation of HIMMO

As stated before, $K_{\xi,\eta}$ and $K_{\eta,\xi}$ need not be equal. In this section, we will describe a set of candidates values for $K_{\eta,\xi}$ given $K_{\xi,\eta}$. In this way, we show that with HIMMO, as described in Section 4, any pair of nodes ξ and η arrive at a common key.

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

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

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

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

Proof We clearly have that

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

As a consequence,

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

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

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

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

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

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

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

Proof Using the notation from Lemma 1, we have

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

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

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

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

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

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

The symmetry of the root keying polynomials implies that

$$\lambda_\xi(\eta) - \lambda_\eta(\xi) = \tilde\lambda_\xi(\eta) - \tilde\lambda_\eta(\xi). \tag{3}$$

We continue with providing upper and lower bounds on $\tilde\lambda_\xi(\eta)$.
As $\lfloor x\rfloor \le x$ for all x, and for all i, $A_i(\xi,\eta)\ge 0$ and $q_i \le N$, it follows that $\tilde\lambda_\xi(\eta) \ge 0$.
We clearly have that

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

Moreover, for each i we have that

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

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

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

Of course, the same bounds are valid for $\tilde{\lambda}_\xi(\eta)$. Combining these bounds with (3), and the fact $\lambda_\xi(\eta) - \lambda_\eta(\xi)$ is an integer number, the theorem follows. □

Corollary 1. *The key generation phase in HIMMO system works. In other words, any two nodes η, ξ following the HIMMO protocol generate the same key.*

Proof This is a direct consequence of Theorem 1 and the discussion on the sending of additional information at the end of Section 4. □

In the following example, we show that under reasonable conditions, the bound from Theorem 1 cannot be significantly improved.

Example For $1 \leq i \leq m$, we choose $R^{(i)}(x,y) = (q_i - 1)x^\alpha y^\alpha$. We assume that $2^{\alpha B} < q_i$. Under this assumption, for each ξ with $0 \leq \xi \leq 2^B - 1$, we have that $\xi^\alpha < q_i$, and so

$$G_\xi(y) = \Big\langle \sum_{i=1}^{m} \big\langle (q_i - 1)\xi^\alpha \big\rangle_{q_i} \Big\rangle_N y^\alpha = \Big\langle \sum_{i=1}^{m} (q_i - \xi^\alpha) \Big\rangle_N y^\alpha = \Big\langle \sum_{i=1}^{m} (q_i - N - \xi^\alpha) \Big\rangle_N y^\alpha$$
$$= \Big[\rho(\xi)N - \sum_{i=1}^{m} \beta_i 2^b - m\xi^\alpha \Big] y^\alpha, \text{ where } \rho(\xi) = \Big\lceil \Big(\sum_{i=1}^{m} \beta_i 2^b + m\xi^\alpha \Big)/N \Big\rceil.$$

In particular,

$$\langle G_\xi(1) \rangle_N = \rho(\xi)N - \sum_{i=1}^{m} \beta_i 2^b - m\xi^\alpha.$$

Moreover, we have that

$$\langle G_1(\xi) \rangle_N = \sigma(\xi)N - (m + \sum_{i=1}^{m} \beta_i 2^b)\xi^\alpha, \text{ where } \sigma(\xi) = \Big\lceil \Big(m + \sum_{i=1}^{m} \beta_i 2^b \Big) \xi^\alpha / N \Big\rceil.$$

We thus have $\big\langle G_1(\xi) \big\rangle_N - \big\langle G_\xi(1) \big\rangle_N \equiv (\sigma(\xi) - \rho(\xi))N \pmod{2^b}$, and so

$$K_{1,\xi} \equiv K_{\xi,1} + [\sigma(\xi) - \rho(\xi)]N.$$

It is clear that

$$\sigma(\xi) - \rho(\xi) \approx \frac{1}{N}(\xi^\alpha - 1) \sum_{i=1}^{m} \beta_i 2^b.$$

So if $\xi = 2^B - 1$, each β_i is approximately 2^B, and $N \approx 2^{(\alpha+1)B+b-1}$, then we have near equality in the upper bound from Theorem 1.

6 Security analysis

In this section, we consider the two attacks described for general ID-based symmetric key establishment schemes in Section 2 when applied to HIMMO.

6.1 First attack scenario: finding equivalent root keying material

The first attack scenario from Section 2 involves solving $\alpha+1$ instances of an MMO problem (viz. obtaining the coefficients of the root keying polynomials from (2) for $0 \leq k \leq \alpha$), coupled by the requirement that the coefficients of the bivariate root keying material polynomials are symmetric. In [5], we have shown that the MMO problem with known moduli $q_1, \ldots, q_m$ and c colluding nodes can be reduced to finding a vector in a lattice of dimension $m(\alpha+c+1)$ which is close (in infinity norm) to a well-defined target vector. Setting up the lattice requires knowledge of the secret moduli $q_1, \ldots, q_m$. In HIMMO, these moduli are kept secret by the TTP, and we see no way to reconstruct them from any c observations. For this reason, we consider recovering the root keying material infeasible. Moreover, even if the moduli were known to the attacker, by increasing m, the lattice dimension can be made too big (tens of thousands) to be handled with current lattice-based algorithms.

6.2 A closer look at the relation between $\langle G_\xi(\eta)\rangle_N$ and $\langle G_\eta(\xi)\rangle_N$

In the second attack scenario from Section 2, the adversary tries to emulate the key generation process of node ξ using the keying materials from the colluding nodes $\eta_1, \ldots, \eta_c$. From these keying materials, the adversary can obtain $\left\langle G_{\eta_i}(\xi)\right\rangle_N$ for $1 \leq i \leq c$ and, by entering the the key generation process with node ξ, he can obtain $\left\langle G_\xi(\eta_i)\right\rangle_N$ for $1 \leq i \leq c$.

In the HI problem, we aim to emulate the key generation process of a node ξ from its keys K_{ξ,η_i} with colluding nodes $\eta_1, \ldots, \eta_c$. In this subsection, we wish to argue that knowledge of all bits of $\left\langle G_\xi(\eta_i)\right\rangle_N$ for $1 \leq i \leq c$ instead of just the b least significant bits does not help much in trying to emulate the key generating process of node ξ.

In Lemma 1, we observed that $\langle G_\xi(\eta)\rangle_N$ is the sum of three terms: a term that is symmetric in ξ and η, a small multiple of N, $\lambda_\xi(\eta)N$,and a multiple of 2^b, $-\mu_\xi(\eta)2^b$. We used this to show that given $K_{\eta,\xi}$, there are at most $4m+1$ choices for $K_{\xi,\eta}$, so that node ξ can ensure that node η can determine ξ, η by sending about $\lceil \log_2 m \rceil + 2$ bits of additional information. We now consider the effect of the last term in the difference between $\langle G_\xi(\eta)\rangle_N$ and $\langle G_\eta(\xi)\rangle_N$, i.e., $(\mu_\eta(\xi) - \mu_\xi(\eta))2^b$, where we recall that

$$\mu_\xi(\eta) = \sum_{i=1}^{m} \beta_i \left\lfloor \frac{A_i(\xi,\eta)}{q_i} \right\rfloor, \text{ with } A_i(\xi,\eta) = \sum_{k=0}^{\alpha} \left\langle R_k^{(i)}(\xi)\right\rangle_{q_i} \eta^k.$$

Although each $R^{(i)}$ is symmetric, the function $A_i(\xi,\eta)$ is not, as $R_k^{(i)}$ is evaluated modulo q_i, while the evaluation of A_i in η is performed over the integers (as is the the summation and multiplication with the β_i's). If η is large, then $A_i(\xi,\eta)$ influences all bits, including the highest order bits; if the $\beta_i's$ are large, $\mu_\xi(\eta)$ affects all bits, including the highest order bits.

Indeed, assume that the coefficients of $A_i(\xi,\eta)$, i.e., the integers $\langle R_k^{(i)}(\xi)\rangle_{q_i}$ are uniformly distributed in $\{0,1,\ldots,q_i-1\}$ then the expected value of $A_i(\xi,\eta)$ equals $\frac{1}{2}q_i\sum_{k=0}^{\alpha}\eta^k \approx \frac{1}{2}q_i\eta^\alpha$. We further assume that each β_i is uniformly chosen from the integers in $[0,2^B)$. Then the expected value of $\mu_\xi(\eta)$ is $m2^{B-2}\eta^\alpha$. Hence, if we take

$$\eta > 2^B(2/m)^{1/\alpha}, \tag{4}$$

then we expect that $2^b\mu_\xi(\eta)$ is larger than $2^b m 2^{B-2}\eta^\alpha > 2^{(\alpha+1)B+b-1}$, so that $2^b\mu_\xi(\eta)$ affects all bits of $\langle G_\xi(\eta)\rangle_N$. As we see no way to relate $\mu_\xi(\eta)$ and $\mu_\eta(\xi)$, we think that with identifiers η satisfying (4), no information on the MSB of $\langle G_\xi(\eta)\rangle_N$ can be obtained from $\langle G_\eta(\xi)\rangle_N$. In other words, the $(\alpha+1)B$ most significant bits of the generated keys and the coefficients of $\langle G_\xi(x)\rangle_N$ are affected by mixing of modular operations (the MMO problem) while the b least significant bits show the evaluation of a polynomial.

The requirement on η expressed in (4) reduces the number of identifiers that we can use from 2^B to $2^B\big(1-(2/m)^{1/\alpha}\big)$. In other words, the "effective bit length" of the identifiers is reduced by $\big\lceil -\log_2\big(1-(2/m)^{1/\alpha}\big)\big\rceil$ bits.

6.3 Second attack scenario: finding equivalent keying material

The second attack scenario from Section 2 involves finding a polynomial $G_\xi \in \mathbb{Z}_N[x]$ for which

$$\big\langle G_\xi(\eta_i)\big\rangle_N = \big\langle K_{\eta_i,\xi}+\lambda_i N\big\rangle_{2^b}+\mu_i 2^b \text{ with } |\lambda_i|\le\Delta \text{ and } 0\le\mu_i\le\big\lfloor N/2^b\big\rfloor, 1\le i\le c.$$

If $\lambda_1,\ldots,\lambda_c$ are known, finding a G_ξ and guessing the key $K_{\xi,\eta}$ as $\langle G_\xi(\eta)\rangle_{2^b}$ amounts to solving a HI problem.

As argued in Subsection 6.2, it seems that the adversary cannot learn much more about $G_\xi(\eta_i)$ from $G_{\eta_i}(\xi)$ than from $\langle G_{\eta_i}(\xi)\rangle_{2^b}$ because of the mixing in the most significant bits. The analysis in [6] shows that the HI problem with c colluding nodes can be reduced to finding a vector in a lattice of dimension $\alpha+c+1$ which is close (in infinity norm) to a well-defined target vector. Even if the lattice vector so obtained corresponds to candidate keying material that generates the correct keys for all colluding nodes, it may generate incorrect keys between the node under attack and other nodes. In [6, Section 6.1] a function is derived, depending on the system parameters α, b, B, the size $|\mathcal{I}|$ of the interval and the number c of colluding nodes with identifiers randomly distributed over $\mathcal{I}$, the sign of which is an indicator of the "predictive power" of a lattice vector matching the observed keys. Numerical experiments in [6] confirm the validity of using this indicator.

When $B = b$ and $|\mathcal{I}| = 2^b$, the indicator function is negative if $c < (\alpha + 1)(\alpha + 2)/2$. A simpler way to understand why this must be so it to count the number of bits of each polynomial coefficient that can influence the last b bits of the evaluation of the polynomial in a identifier $\eta < 2^b$: only the b least significant bits and the kb most significant bits of the coefficient of η^k have any significant impact, the other bits of that coefficient can affect the final result only through a carry in the addition of the polynomial terms. The attacker must thus estimate $\sum_{k=0}^{\alpha}(b+kb) = b(\alpha+1)(\alpha+2)/2$ bits, while each observation gives him b bits. So with fewer than $(\alpha+1)(\alpha+2)/2$ observations, there will be many fits to the observed points and the predictive power is negligible.

If we assume that the attacker cannot choose the identifiers $\eta_1, \ldots, \eta_c$, but these are given to him randomly from the set of all possible identifiers, then c must be greater than $(\alpha + 1)(\alpha + 2)/2$. The lower bound on c, and thus the minimum lattice dimension, grows quadratically in α. For $\alpha = 26$ the attacker must solve a lattice problem in more than 405 dimensions, which is about the upper limit for practical lattice reduction algorithms. Increasing α makes this lattice attack even more infeasible.

When the attacker can choose the identifiers $\eta_1, \ldots, \eta_c$, he can pick them from a smaller interval containing the identifier of the node under attack. For $\alpha = 26$, $b = B = 32$ and $|\mathcal{I}| = 256$, the indicator function is positive for $c \geq 73$. Simulations in [6] confirm this lower bound for a successful attack.

7 Experimental results

Both attack scenarios from Section 6 involve solving a lattice problem. As explained in Section 6.1, we consider retrieving the root keying material infeasible. For the second attack scenario, finding equivalent keying material, the adversary needs to find a vector of dimension $\alpha + c + 1$ that is close to a well-defined target vector, where c is the number of nodes for which the adversary has the keying material. In this section, we describe the experimental results for solving this lattice problem. We restricted ourselves to the case that the identifies of the colluding nodes are uniformly distributed over the complete identifier space.

7.1 Description of our experiments

We first choose a value for b, the number of key bits, B, the number of ID bits, and α, the polynomial degree. We then choose a random odd integer N in the interval $(2^{(\alpha+1)B+b-1}, 2^{(\alpha+1)B+b})$ and $\alpha + 1$ random integer polynomial coefficients $g_0, \ldots, g_\alpha$ from $[0, N)$. With these coefficients we construct a polynomial $G(x) = \sum_{j=0}^{\alpha} g_j x^j$.

We choose a number c and pick c different numbers $\eta_1, \ldots, \eta_c$ from the interval $[0, 2^b)$ and calculate the numbers $h_k = \langle\langle G(\eta_k)\rangle_N\rangle_{2^b}$, $1 \leq k \leq c$.

The numbers α, b, B, N and the c pairs (η_k, h_k) are input to the reconstruction algorithm. This algorithm outputs a set of integer coefficients $\hat{g}_0, \ldots, \hat{g}_\alpha$ in

$[0, N)$.

We say that the algorithm has produced a perfect fit to the observed values if

$$h_k = \langle\langle\sum_{j=0}^{\alpha} \hat{g}_j \eta_k^j\rangle_N\rangle_{2^b} \text{ for } 1 \leq k \leq c.$$

For an integer $\eta \notin \{\eta_1, \ldots, \eta_c\}$, we say that the algorithm has produced a correct interpolation in η if $\langle\langle\sum_{j=0}^{\alpha} \hat{g}_j \eta^j\rangle_N\rangle_{2^b} = \langle\langle G(\eta)\rangle_N\rangle_{2^b}$.

7.2 Description of the reconstruction algorithm

The algorithm for obtaining the coefficients $\hat{g}_j$, $0 \leq j \leq \alpha$ makes use of the equivalence of this reconstriction problem to a lattice problem, as described in [6]. The lattice is spanned by the rows of the block matrix

$$\begin{pmatrix} N\mathbb{I}_c & 0 \\ \mathbb{V} & 2^{-b}\mathbb{I}_{\alpha+1} \end{pmatrix},$$

where $\mathbb{I}_c$ and $\mathbb{I}_{\alpha+1}$ denote unit matrices of size $c \times c$ and $(\alpha+1) \times (\alpha+1)$ respectively, and $\mathbb{V}$ denotes the $(\alpha+1)\times c$ Vandermonde matrix with elements $V_{i,j} = \eta_j^i$, $0 \leq i \leq \alpha$, $1 \leq j \leq c$. The problem is to find a lattice vector that lies inside a hypercube of iedge length $N/2^b$ around a target vector that is constructed with the values h_j.

This is a relaxed version of the Closest Vector Problem, and we use a standard technique for finding a lattice vector that is expeted to be close to a target vector. The procedure uses two steps:

1) We perform a basis reduction, in order to make the lattice basis more orthogonal. We use LLL [7] with default parameters, as implemented in Sage [8].
2) With the LLL-reduced basis, we use Babai's nearest plane algorithm [2] to find a lattice vector close to the target vector.

The coefficients $\hat{g}_j$ are obtained from the corresponding components of the resulting lattice vector. We refer to [6] for details.

7.3 Choosing c

In our experiments, c, the number of observations, is an important parameter. The main reason is that c must be large enough for a fit to the observations to also be a good interpolator for the last b bits of G.

In [6] the authors give an estimate of the length of the shortest lattice vector that corresponds to a solution that fits the last b bits of G in the observed values, but does not agree in other points. If the length of this vector is large enough, all lattice vectors that are close to the target vector must correspond to solutions that agree with the last b bits of G in most points.

Specifically, the authors consider the lattice as a direct sum of two lattices, one of which being the lattice spanned by the short vectors that correspond to the polynomials that evaluate, modulo N, to numbers that are zero modulo 2^b in the entire interval $[0, 2^B)$. The short vectors in the other lattice correspond to polynomials that evaluate modulo N to numbers that are zero modulo 2^b in the points $\eta_1, \dots, \eta_c$, but non-zero in other points. The expectation value of the inverse squared volume of this lattice can be calculated explicitly, when $\eta_1, \dots, \eta_c$ are uniformly distributed over $[0, 2^B)$, and taking this result to the power $-1/(2d)$, where $d = c - \alpha - 1$ is the dimension of this lattice, gives an estimate of the average length of a short vector in this lattice. They define the function S as the logarithm of this length divided by half the length of the diagonal of the hypercube, which gives

$$S(\alpha, b, B, c) = \log\left(\frac{2^{b+1}}{\sqrt{c}}\right) + \frac{1}{2(c-\alpha-1)}\Bigg(\sum_{i=1}^{\alpha}\big(\log((\alpha+1+i)!) - \log(i!)\big) \\ - \log\begin{pmatrix} c \\ \alpha+1 \end{pmatrix} - \alpha(\alpha+1)\log(2^B)\Bigg),$$

where we correct an error in the result from [6] and adapt their notation to the one used in this paper.
When $S > 0$, a good fit is thus expected to be a good interpolation, while for $S < 0$ this is not guaranteed. The probability of finding a good interpolation in this case will depend on the number of short vectors in this lattice and their lengths, and this cannot be accurately derived from simple volume considerations alone. With this caveat, the volumetric estimate of this probability is

$$\lambda^{\alpha+1}\left(\frac{\sqrt{c}}{2}\right)^{c-\alpha-1} \exp\left((c-\alpha-1)S(\alpha, b, B, c)\right) \text{ for some } \lambda \in [1, \sqrt{c}).$$

As S is increasing in c, we can define $c_{\text{min}}(\alpha, b, B)$ as the smallest value of c such that $S(\alpha, b, B, c) > 0$. In Table 2 we give $c_{\text{min}}(\alpha, b, B)$ for several values of α and $b = B$, and compare its value to $(\alpha+1)(\alpha+2)/2$.

We thus want to choose $c \geq c_{\text{min}}$ in order to obtain a good interpolation. For smaller c, the probability for obtaining a good interpolation is expected to decrease very rapidly to zero.
On the other hand, rounding algorithms do not necessarily give a perfect fit. The quality of the fit is expected to decrease as c, and thus the lattice dimension, grows. We can only obtain a good interpolation if the lattice algorithm still gives a good fit for $c = c_{\text{min}}$. Table 3 summarizes our results. We did 10 runs for each case, counting the number of good fits and interpolations. So, for example, for $B = b = 16$ and $\alpha = 8$, and $c = 39$, the notation 10,0 means that we obtained a good fit 10 times, and a good interpolation 0 times. Perfect fits and interpolations turned out to be very rare, which is why we relax the definition a bit: we call a fit good, if for all k, $1 \leq k \leq c$ it holds that

$$\Big\langle \langle \sum_{j=0}^{\alpha} \hat{g}_j \eta_k^j \rangle_N \Big\rangle_{2^b} = \big\langle h_k + \lambda_k N \big\rangle_{2^b} \text{ with } \lambda_k \in \{-1, 0, 1\},$$

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

				b		
α	$f(\alpha)$	8	16	32	64	128
4	15	14	15	15	15	15
8	45	39	43	44	45	45
12	91	75	85	89	90	91
16	153	122	142	148	151	152
20	231	178	212	223	228	230
24	325	243	297	313	320	323
28	435	317	396	419	428	432
32	561	398	508	539	551	556
36	703	487	635	675	690	697
40	861	582	775	825	845	853

Table 3
Number of good fits and interpolations out of 10 runs for $c = \lfloor 0.9c_{\text{min}} \rceil$ and $c = c_{\text{min}}$.

	$b = B = 16$		$b = B = 32$	
$\alpha = 8$	$(c = 39)$ 10, 0	$(c = 43)$ 10, 10	$(c = 40)$ 10, 0	$(c = 44)$ 10, 10
$\alpha = 12$	$(c = 77)$ 2, 0	$(c = 85)$ 7, 7	$(c = 80)$ 10, 0	$(c = 89)$ 10, 10
$\alpha = 16$	$(c = 128)$ 0, 0	$(c = 142)$ 0, 0	$(c = 133)$ 10, 0	$(c = 148)$ 0, 0

and we call an interpolation good if for many of 1000 randomly chosen η the interpolation at η is correct. With this definition, our experiments show that, if c is large enough, a good fit leads to a good interpolation. They also show that no good interpolation is obtained from a fit that is not good. Finally, they show that the number of good fits goes down as c grows. For $\alpha = 16$ the lattice algorithm we use cannot produce a good fit for values of c that we would expect would make a good fit a good interpolation.

8 Conclusions and future work

We described HIMMO, a novel identity-based pairwise symmetric key establishment method. The method is non-interactive: nodes can directly generate a common key without exchanging messages, saving valuable round-trip time. From an implementation point of view, HIMMO is very attractive as well.

We have placed the HI and MMO problems [5], [6] in the context of the HIMMO scheme and analyzed its security. The experimental results from Section 7 strongly suggest that application of the LLL algorithm followed by Babai's nearest plane algorithm does not yield the correct result if $\alpha > 20$. The explanation for this is that the number of colluding nodes required for successful reconstruction is so large that the applied approximation algorithm

(LLL , followed by Babai's nearest plane algorithm) is no longer sufficiently accurate.

Future research can include using more accurate algorithms than LLL in the attack, for example, the BKZ algorithm. The comprehensive summary of the experiments of Gama and Nguyen [4] helps analyzing the practical behaviour of LLL, BKZ and the "deep insertion" variant of LLL. However, the specific form of the lattices derived from HIMMO may make drawing conclusions from this reference without sufficient experiments a little risky. For exact algorithms for finding the closest vector, both the running time and the memory requirements are exponential in the lattice dimension and thus quickly become impractical. For example, the algorithm from [1] is reported to require 3TB of memory and 2080 hours of computation time for a lattice of dimension 90.

Acknowledgements

We thank Bouke Cloostermans (TU Eindhoven) for reducing the upper bound in Section 5 from $3m-1$ in a previous version to the current value of $2m$, and Igor Shparlinski and Berry Schoenmakers for their comments on earlier drafts of this paper.

References

[1] Nicolas Gama Anja Becker and Antoine Joux. Solving shortest and closest vector problems: The decomposition approach. Cryptology ePrint Archive, Report 2013/685, 2013. http://eprint.iacr.org/.

[2] L. Babai. On Lovász lattice reduction and the nearest lattice point problem. *Combinatorica*, 6:1–13, 1986.

[3] C. Blundo, A. de Santis, A.Herzberg, S. Kutten, U. Vaccaro, and M. Yung. Perfectly secure key distribution for dynamic conferences. *Information and Computation*, 146:1–23, 1998.

[4] Nicolas Gama and Phong Q. Nguyen. Predicting lattice reduction. In Nigel P. Smart, editor, *EUROCRYPT*, volume 4965 of *Lecture Notes in Computer Science*, pages 31–51. Springer, 2008.

[5] Oscar García-Morchón, Domingo Gómez-Pérez, Jaime Gutiérrez, Ronald Rietman, and Ludo Tolhuizen. The MMO problem. In *Proceedings ISSAC'14*, pages 186–193. ACM, 2014.

[6] Oscar García Morchon, Ronald Rietman, Igor E. Shparlinski, and Ludo Tolhuizen. Interpolation and approximation of polynomials in finite fields over a short interval from noisy values. *Experimental mathematics*, Accepted, 2014.

[7] Phong Q. Nguyen and Brigitte Vallée, editors. *The LLL Algorithm - Survey and Applications*. Information Security and Cryptography. Springer, 2010.

[8] Sage.