HIMMO: A Lightweight, Fully Collusion Resistant Key Pre-Distribution Scheme

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

Abstract

Public-key cryptography addresses key distribution and agreement in a very elegant way by allowing any pair of nodes to generate a common secret without sharing any information beforehand. In the alternative approach of key pre-distribution schemes (KPS), a trusted-third party (TTP) securely provides each node with a (node-dependent) secret function allowing pairs of nodes to agree on a common key in a non-interactive way, which is a big advantage in delay-critical applications. However, no known KPS is simultaneously secure and efficient.

This paper proposes HIMMO, a KPS which relies on the recently introduced Hiding Information (HI) and Mixing Modular Operations (MMO) problems. Our security analysis shows that HIMMO is fully collusion resistant for appropriate parameter choices. HIMMO is also lightweight for these parameters and thus makes non-interactive key establishment feasible even in very large networks. Additionally, the identity-based nature of HIMMO enables implicit certification and verification of credentials, as well as secure broadcast by the TTP. HIMMO can also accommodate multiple TTPs so that no single TTP knows the keys shared between nodes. All these features make HIMMO a very promising candidate to enable more efficient security protocols.

1 Introduction

Public-key cryptography addresses the security cornerstone of key distribution and agreement in a very elegant way by allowing any pair of parties to generate a common secret without sharing any information beforehand.

Matsumoto and Imai proposed an alternative “key pre-distribution scheme” (KPS) approach in 1987 [12], by generalizing work of Blom from 1984 [4]. In this approach, a trusted third party (TTP) provides each node with a function which it extracts from some secret root keying material and the node’s identity. During operation, a node uses this function to directly compute a common symmetric key with each other node from the latter’s identity. So no interaction between communicating parties is required to establish a common key, which is a big advantage in delay-critical applications. KPS also support adding new nodes to a running network without the need to update already deployed nodes since the secret functions owned by any of the nodes is generated by the same root keying material.

As Matsumoto and Imai wrote in [12], *identities play very important roles in KPS, and in this point, KPS are like the ID-based cryptosystem; the independent work of A. Shamir’s* [16]. Apart from the non-interactive key agreement using identities as explained above, KPS supporting the use of long identities can also enable, other **identity-based schemes**. A first identity-based scheme enabled by KPS provides implicit certification and verification of credentials in a one-to-one communication setting. Since a KPS is identity-based, we can link the credentials of a node to its identifier, and therefore also its secret function. A node sending a message to another node can attach its credentials. From the received credentials, the receiving node can then obtain the identity of the sender and subsequently generate the common key. Authentication of the sent message, using the common key, allows verification of the credentials of the server. This construction allows embedding information in the credentials that can be verified during operation such as the node features, the roles of the node, or dates specifying the expiration date of the keying material. Another identity-based scheme relates to secure broadcast [12], [18] by the TTP, in which the broadcast message is linked to a function obtained from the root keying material and a fingerprint of the message. In such a configuration, nodes can directly communicate with each other in a secure way and also perform source authentication of any broadcast message generated by the TTP. Other identity-based schemes enable direct key agreement between t parties or differentiating between groups of nodes by using root keying material with specific features [5]. Finally, as already identified by Matsumoto and Imai [12], KPS can also operate with multiple TTPs removing a single point of failure and eliminating privacy issues caused by a single TTP being able to decrypt all the messages exchanged between any pair of nodes in the network.

All the features enabled by the above identity-based schemes make KPS useful. However, existing KPS fail in an important aspect: there is **no known KPS scheme** that is both **efficient and secure** – in the sense of fully collusion resistant – at the same time. This is the reason why the above identity-based schemes are also not feasible today.

In the **literature** we find several KPS. A straightforward and efficient KPS scheme was described by Blundo *et al.* [5] in 1992. In this scheme, which we will call *Blundo's scheme*, the TTP first randomly generates a symmetric bivariate polynomial $R(x, y)$ of degree α in each of the variables with coefficients from $\mathbb{Z}_p$, the ring of integers modulo p. Next, the TTP provides, in a secure manner, to any node ξ in the network the keying material l $R(\xi, y) \in \mathbb{Z}_p[y]$. The key that node ξ uses in order to communicate with node η equals $R(\xi, \eta)$ (computed modulo p). As node η uses key $R(\eta, \xi)$ to communicate with ξ, and R is symmetric, nodes ξ and η use the same keys for communicating with each other. Blundo's scheme is fast and requires little storage. Other advantages are that it allows for any network of size at most p, so that it scales well, e.g., to the Internet, and that nodes can be added to a running network without the need to update already deployed nodes. Unfortunately, this scheme offers information-theoretic security only if at most α nodes are compromised. However, simple interpolation using the keying material of any $\alpha + 1$ nodes allows the retrieve the root keying material [5], thereby compromising the complete system. Also, the keying material of a single node ξ can be obtained by simple interpolation of the keys of any $\alpha + 1$ colluding nodes with ξ. More recently, Zhang *et al.* [19] presented a scheme that aimed to overcome the limited resistance of Blundo's scheme against attacks by colluding nodes by offering complexity-theoretic security. In their scheme, the TTP generates, apart from the root keying material $R(x, y)$, two univariate polynomials g and h. The set of the nodes is determined such that $g(\eta)$ and $h(\eta)$ both are small for each node η. Next, the TTP randomly provides, in a secure manner, to any node ξ in the network as keying material

either $R(\xi, y)+g(y)$ or $R(\xi, y)+h(y)$. Key generation is performed as in Blundo's scheme, with a small additional key reconciliation step to ensure that nodes ξ and η will use the same key for communicating with each other.
Zhang's scheme was broken by Albrecht *et al.* [1] who showed that the coefficients of g and h can be obtained by finding short vectors in a well-defined lattice of which the dimension does not depend on the degree of R, and is very small for the parameters suggested in [19].
KPS schemes are related —to some extent—to identity-based encryption [6]. The main different is that identities in identity-based encryption are public keys. In identity-based encryption, a symmetric key is not directly generated as in a KPS. Instead, direct key sharing is done by encrypting a symmetric key with the public key (identifier) of the other party that can decrypt with its private key. This leads to higher resource requirements.
There is also some relation with non-interactive key exchange schemes (NIKE) [8]. In NIKE, nodes are pre-configured with public and private keys and any pair of nodes can obtain a common secret by combining the public-key of the other node and its own private key in a non-interactive way. However, NIKE schemes have two limitations. First, any node needs to know the public keys of any node with whom it wishes to communicate, potentially requiring some information exchange. Second, public-keys need certification to link them to the identities of the involved parties.

The **contributions** of this paper are HIMMO and identity-based schemes enabled by HIMMO. HIMMO is a KPS with similar operational characteristics as previous key pre-distribution schemes, [5], [12], [19] allowing for efficient pairwise key generation from identifiers even in very large networks, but it is also fully collusion resistant for suitable configuration parameters. The acronym "HIMMO" is derived from two interpolation problems that are essential for the security of the scheme: the HI problem [13], which is related to the well-known noisy interpolation problem, and the MMO problem [10]. HIMMO, as any KPS, 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 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. Beyond non-interactive key agreement, HIMMO enables all the above multiple identity-based schemes. For instance, HIMMO achieves combined key agreement and implicit credential verification at least around one order of magnitude faster than when using ECDH and ECDSA and without involving any communication interaction.

The paper is organized as follows. In Section 2, we review the HI problem and the MMO problem. Section 3 describes the HIMMO scheme. In Section 4, we discuss the security model, the impact of the HI and MMO problems on the security of HIMMO, and finally, we discuss the parameters that make HIMMO secure. In Section 5, we describe the experimental results supporting our security analysis. In Section 6, we detail how HIMMO enables all above identity-based schemes. Section 7 briefly discusses HIMMO's performance advantages compared with other other solutions. In Section 8, we draw conclusions and indicate directions for further research. In the appendix, we validate the HIMMO scheme in that we show that the generated keys indeed are pairwise the same.

2 HI and MMO Problems

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

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

As shown in [13], the HI problem is equivalent to a **noisy polynomial interpolation problem**, which in turn is shown to be related to an approximation problem in a certain lattice.

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

The MMO problem has been introduced in [10]. There it was shown that the MMO problem with known moduli is equivalent to a lattice problem of the same kind as for the HI problem, whereas no solution is known for the MMO problem with unknown moduli.

3 Description of the HIMMO scheme for key establishment

In this section, we describe the HIMMO scheme for key establishment. The other applications of HIMMO outlined in the introduction are detailed in Section 6. In this description, we use the following notation: for each integer x and positive integer M, we denote by $\langle x\rangle_M$ the unique integer $y \in \{0, 1, \ldots, M-1\}$ such that $x \equiv y \bmod M$.

Like for any KPS, a trusted third party (TTP) is required, and three phases can be distinguished [12].

In the **setup phase**, the TTP obtains or selects 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 moduli $q_1, \ldots, q_m$ of the $q_i = N - 2^b\beta_i$, where $0 \leq \beta_i < 2^B$ and at least one of $\beta_1, \ldots, \beta_m$ is odd. Finally, the TTP generates 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

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

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:

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

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

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

It can be shown that $K_{\xi,\eta}$ and $K_{\eta,\xi}$ need not be equal. However, as shown in Theorem A.1 in the appendix, for all identifiers ξ and η with $0 \leq \xi, \eta \leq 2^B$,

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

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 $C = \{\langle K_{\eta,\xi} + jN \rangle_{2^b} \mid 0 \leq |j| \leq 2m\}$. No additional 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 $K_{\xi,\eta}$, cf. [18]. 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. Alternatively, node ξ sends to node η the number $r = \langle K_{\xi,\eta} \rangle_{2^s}$, where $s = \lceil \log_2(4m+1) \rceil$. Node η can efficiently obtain the integer j such that $|j| \leq 2m$ 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 $K_{\xi,\eta}$, that is, the number $\lfloor 2^{-s} K_{\xi,\eta} \rfloor$, should be used as key.

4 Security analysis

We consider a **security model** in which an adversary has obtained the secret keying material shares of 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 $\{\eta_1, \ldots, \eta_c\}$. Inspired by the security model of Matsumoto and Imai [12] and related attacks on Blundo's scheme and Zhang's scheme, we consider that the adversary has two options for doing so.

1) Attacking node ξ's keying material: The adversary finds keying material for node ξ that is consistent with the keys $K_{\eta_i,\xi}$, which he can calculate from the keying materials of nodes $\eta_1, \ldots, \eta_c$. From the keying material that the adversary obtained for node ξ, he calculates $K_{\xi,\eta}$.
2) Attacking the root keying material: The adversary finds 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 attacks need not retrieve the atual polynomial $G_\xi(y)$ of node ξ or the actual root keying material, it is sufficient 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.

Matsumoto and Imai identified two KPS types: schemes relying on information-theoretic security (such as Blundo's scheme) and schemes relying on complexity-theoretic security (as Zhang's scheme pretended to be). We shall argue that HIMMO belongs to both categories: for small c, the colluding nodes do not have enough information to obtain a good estimate of $K_{\xi,\eta}$. Only when c is greater than a certain minimum value c_{min}, the consistency requirements limit the possible outcomes of the keying material or root keying material such that the odds for finding the correct key are increased. However, the system parameters m and α can be chosen such that $c_{\min}$ becomes so large that finding a keying material or root keying material that is consistent with all requirements becomes computationally infeasible.

Our analysis shows that HIMMO can be the first KPS that make interpolation or other standard algebraic techniques infeasible, if suitable parameters are chosen.

4.1 Symmetry of $\langle G_\eta(\xi)\rangle_N$ and $\langle G_\xi(\eta)\rangle_N$

This section argues the following conjecture that is critical for the security analysis of HIMMO.

Conjecture 4.1. *It is infeasible to relate the $(\alpha+1)B$ most significant bits of $\langle G_\xi(\eta)\rangle_N$ to those of $\langle G_\eta(\xi)\rangle_N$.*

To support this conjecture, recall that from Lemma A.1, $\langle G_\xi(\eta)\rangle_N$ is the sum of three terms. The first term is invariant under the exchange of ξ and η, the second and the third terms are small multiples of N and 2^b, respectively.

We consider the effect of the last term in the difference between $\langle G_\xi(\eta)\rangle_N$ and $\langle G_\eta(\xi)\rangle_N$, i.e., $(\mu_\eta(\xi)-\mu_\xi(\eta))2^b$, where according to Lemma A.1 in the appendix,

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

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

Indeed, assume that the coefficients of $A_i(\xi,\eta)$, i.e., the integers $\big\langle R_k^{(i)}(\xi)\big\rangle_{q_i}$ 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

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

then we expect that $2^b\mu_\xi(\eta)$ is larger than $2^b m2^{B-2}\eta^\alpha>2^{(\alpha+1)B+b-1}$, so that $2^b\mu_\xi(\eta)$ affects all bits of $\langle G_\xi(\eta)\rangle_N$. Since the $\mu_\xi(\eta)$ and $\mu_\eta(\xi)$ are affected by the mixing of modular operations, we conjecture that with identifiers η satisfying (2), no information on the $(\alpha+1)B$ most significant bits of $\langle G_\xi(\eta)\rangle_N$ can be obtained from $\langle G_\eta(\xi)\rangle_N$.

The requirement on η expressed in (2) 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. For reasonable parameters, this is not a very big number, e.g., for $m=10$ and $\alpha=25$, the loss is three bits in the identifier space.

In Figure 1 we show experimental evidence for the claim that the $(\alpha+1)B$ most significant bits of $\langle G_\xi(\eta)\rangle_N$ and $\langle G_\eta(\xi)\rangle_N$ are uncorrelated if the identifier interval is restricted according to Equation 2. Note that this property does not appear to hold for a few most significant bits, which are equal with probability significantly above 0.5. This is because in our simulations some coefficients or identifiers might be slightly smaller so that the mixing of modular operations effect does not propagate to the very most significant bits. These bits may thus be used in an attack as well.

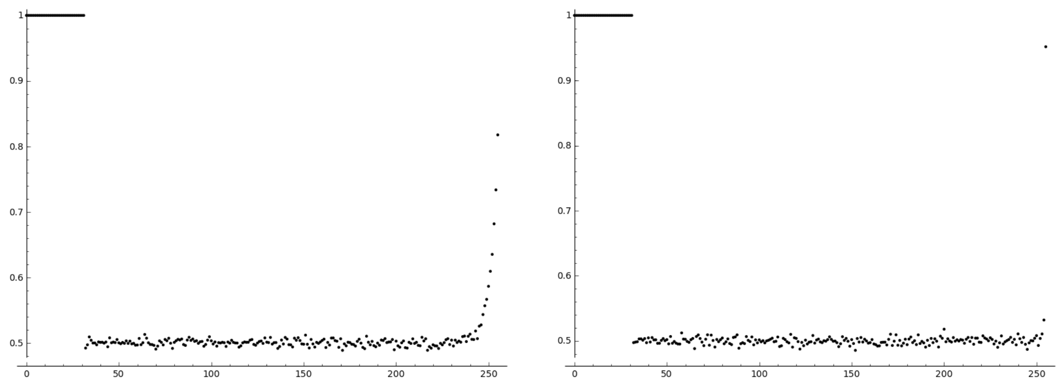

Figure 1. The probability that the i-th bit of $\langle G_\xi(\eta)\rangle_N$ equals the i-th bit of $\langle G_\eta(\xi)\rangle_N$ when $K_{\xi,\eta} = K_{\eta,\xi}$ as a function of i in a HIMMO system with $\alpha = 6$, $b = B = 32$ and $m = 5$. In the plot on the left, ξ and η are averaged over the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

4.2 Attacking node ξ's keying material

The first attack scenario involves finding a polynomial $G \in \mathbb{Z}_N[y]$ as in TheoremA.1, in other words,

$$\left\langle G(\eta_i)\right\rangle_N = \left\langle K_{\eta_i,\xi} + \lambda_i N\right\rangle_{2^b} + \mu_i 2^b \text{ with } |\lambda_i| \leq 2m \text{ and } 0 \leq \mu_i \leq \left\lfloor N/2^b \right\rfloor, 1 \leq i \leq c.$$

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

As argued in Section 4.1, 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 [13] 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. 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 is also given. 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, [13] considers 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 $\eta_1, \ldots, \eta_c$, but non-zero in other points. The expectation value of the inverse squared volume of this lattice can be calculated explicitly, when $\eta_1, \ldots, \eta_c$ 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

$$S(\alpha, b, B, c) = \log\left(\frac{2^{b+1}}{\sqrt{c}}\right) + \frac{1}{2(c-\alpha-1)}\left(\sum_{i=1}^{\alpha}\big(\log((\alpha+1+i)!) - \log(i!)\big) - \log\binom{c}{\alpha+1} - \alpha(\alpha+1)\log(2^B)\right), \quad (3)$$

where we correct an error in the result from [13] 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.

Numerical experiments in [13] confirm the validity of using this indicator. When $B = b$ and $|I| = 2^b$, the indicator function is negative if $c \approx (\alpha+1)(\alpha+2)/2$.

A simpler way to understand this is 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 $\alpha = 40$ the attacker must solve a lattice problem in about 900 dimensions, which lies above the upper limit for practical lattice reduction algorithms. We must point out that the record in the *Ideal Lattice Challenge* is in dimension 825, see [7]. Increasing α makes this lattice attack even more infeasible.

The above analysis holds when the identifiers are uniformly distributed in $[0, 2^B)$ When the attacker can choose the identifiers $\eta_1, \ldots, \eta_c$, then he can pick them from a smaller interval containing the identifier of the node under attack, improving his chances to attack the system. For instance, in the previous case with $\alpha = 40$, $b = B = 32$ and $|I| = 256$, the indicator function is positive for $c \geq 120$. Simulations confirm this lower bound for a successful attack.

4.3 Attacking the root keying material

In Section 4.1 we have argued that the generated keys and coefficients of the keying materials contain the contributions of the evaluation of multiple polynomials in different rings. Therefore, the second attack scenario can be reduced to a slightly modified MMO problem in which the values of function h are reduced modulo N, see Problem 2. Thus, finding an equivalent root keying material involves solving $\alpha + 1$ instances of this problem, (viz. obtaining the coefficients of the root keying polynomials from (1) for $0 \leq k \leq \alpha$). In [10], it was 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 there is no efficient 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.

4.4 HIMMO Security Parameters

The selection of the parameters of the system to be secure involves the following:

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

A configuration that the authors consider to be complexity-theoretic secure is to take $b = B = 128$ bits, $\alpha = 26$, and $m = 10$. We remark that the identifiers of the nodes should be uniformily distributed.

5 Experimental results

Both attack scenarios from Section 4 involve solving a lattice problem. As explained in Section 4.3, we consider retrieving the root keying material infeasible. For 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 identifiers of the colluding nodes are uniformly distributed over the complete identifier space.

5.1 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 $\langle\langle \sum_{j=0}^{\alpha} \hat{g}_j \eta^j \rangle_N\rangle_{2^b} = \langle\langle G(\eta)\rangle_N\rangle_{2^b}$. We also use the terms 'good fit' and 'good interpolation', they are defined more precisely in the context of our experiments in subsection 5.3.

5.2 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 reconstruction problem to a lattice problem, as described in [13]. 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 edge length $N/2^b$ around a target vector that is constructed with the values h_k.
This is a relaxed version of the Closest Vector Problem, and we use a standard technique for finding a lattice vector that is expected 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, for that we use LLL, see [14], with default parameters, as implemented in Sage [15].
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 [13] for details.

5.3 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.
According to Subsection 4.2, the probability of finding a good interpolation if $S(\alpha, b, B, c)$ is negative, where S is defined in Equation (3) depends 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 1 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, approximate 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 2 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

Table 1
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
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 2
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

is why we relax the definition somewhat bit: we call a fit good, if for all k, $1 \leq k \leq c$ it holds that

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

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 expect to be such that a good fit yields a good interpolation as well. We thus conclude that our attack breaks down for lattice dimension larger than 150. This result is in line with literature in which it is reported that the LLL algorithm breaks at some point of time when the lattice dimension grows, e.g., in average in dimension ≈ 180 according to [17].

6 Some identity-based schemes enabled by HIMMO

If we take the following HIMMO configuration parameters: $B = 2b = 160$ bits, $m = 10$, and $\alpha = 26$, then attacking the 80-bit keys generated by a specific device would require solving a lattice of dimension 406 for the HI problem once enough nodes have been compromised. Attacking HIMMO through the MMO problem is hopeless since the q_i's are secret, and even if they are known, an attacker would have to deal with a lattice of dimension 40600. Attacking a key by means of a brute force attack is also not feasible due to the chosen key length. Thus, we believe that HIMMO can achieve the full collusion property enabling the schemes discussed in the introduction.

The first application of HIMMO is in **non-interactive key agreement**, in large networks, where nodes can be added to a running network without the need to update already deployed nodes. With the above parameters, any pair of nodes in a network, each identified by a 160-bit long identity, can directly agree on a common key based on their credentials, e.g., MAC addresses.

Another scheme enabled by HIMMO and its identifiers is about **implicit certification and verification of credentials**. A node that wants to register with the system provides the TTP with its credentials, e.g., device type, manufacturing date, etc. The TTP, which can also add further information to the node's credentials such as the issue date of the keying material and its expiration date, obtains the node's identity as $\xi = H(credentials)$, where H is a public hash function. When a first node with identity ξ wants to securely send a message M to a second node with identity η, the following steps are taken.

- Step 1: Node ξ computes a common key $K_{\xi,\eta}$ with node η, and uses $K_{\xi,\eta}$ to encrypt and authenticates its credentials and message M, say $e = E_{K_{\xi,\eta}}(\textit{credentials}|M)$.
- Step 2: Node ξ sends (ξ, e) to node η.
- Step 3: Node η receives (ξ', e'). It computes its common key $K_{\eta,\xi'}$ with ξ' to decrypt e' obtaining the message M and verifying the authenticity of the received message. Furthermore, it checks whether the *credentials'* in e correspond with ξ', that is, it validates if $\xi' = H(\textit{credentials}')$.

This method allows not only for direct secure communication of message M but also for implicit certification and verification of ξ's credentials because the key generating polynomial assigned to a node is linked to its credentials by means of the one-way hash function H. If the output size of H is long enough, e.g., 256 bits, and H is a secure one-way hash function, then it is infeasible for an attacker to find any other set of credentials leading to the same identity ξ. The fact that credential verification might be prone to Birthday paradox attacks motivates the choice for the relation between identifier and key sizes, namely, $B = 2b$. In this way, the scheme provides an equivalent security level for credential verification and key generation. The capability for credential verification enables applications such as the verification of the expiration date of the credentials (and the keying material) of a node or the verification of the access roles of the sender node ξ.

An extension of a KPS uses **multiple TTPs** to remove privacy issues caused by a single TTP being able to decrypt all the messages exchanged between any pair of nodes in the network [1]. The method already outlined in [12] can be used HIMMO as follows. The setup phase is divided into two sub-phases: in a first step, parameters (b, B, m, α, N) are centrally determined and published; in a second step, for $1 \leq l \leq s$, TTP l independently generates m secret $q_{l,i}$ and the corresponding m secret symmetric bivariate polynomials $R^{(l,i)}(x, y)$ over $\mathbb{Z}_{q_{l,i}}$. In the keying material extraction phase, each node ξ securely receives from each of the s TTPs a key generating polynomial $G_\xi^{(l)}(y) \in \mathbb{Z}_N[y]$. Node η computes the coefficients of its final key generating polynomial G_ξ by adding the corresponding coefficients of $G_\xi^{(1)}, \ldots, G_\xi^{(s)}$, so

$$G_\xi(y) = \langle \sum_{l=1}^{s} G_\xi^{(l)}(y) \rangle_N. \tag{4}$$

Key generation is done as in HIMMO. Note that the scheme operates exactly as a

1. Note that this might be desirable in some applications such as key escrow.

scheme with a single TTP which generates the ms root keying material polynomials $R^{(l,i)}(x,y) \in \mathbb{Z}_{q_{l,i}}[x,y]$ for $1 \leq i \leq m, 1 \leq l \leq s$. Clearly, if $s > 1$, a single TTP cannot determine the key generating polynomial of individual nodes.

There are other schemes enabled by HIMMO such as **secure broadcast** in which the TTP wishes to broadcast a message of which the origin can be verified by the receiving nodes. This protocol is described in [18]. HIMMO can also be extended to allow for key agreement with groups of $t-1$ devices if the TTP uses polynomials in t variables and distributes a key generating polynomial in $t-1$ variables to the nodes. Note that in this case the HIMMO parameters need to be adapted to introduce the HI and MMO problem in the correct parts of the keys and coefficients of the key generating polynomial.

7 HIMMO Performance

Figure 2 provides a brief summary of the performance of the HIMMO scheme. The first graph shows the key generation time for $\alpha = 26$ as a function of $b = B$. In the next three figures, we see – as a function of α and for $b = B = 128$ – the key generation time, the size of the key generating function, and the lattice dimension associated to the HI problem.

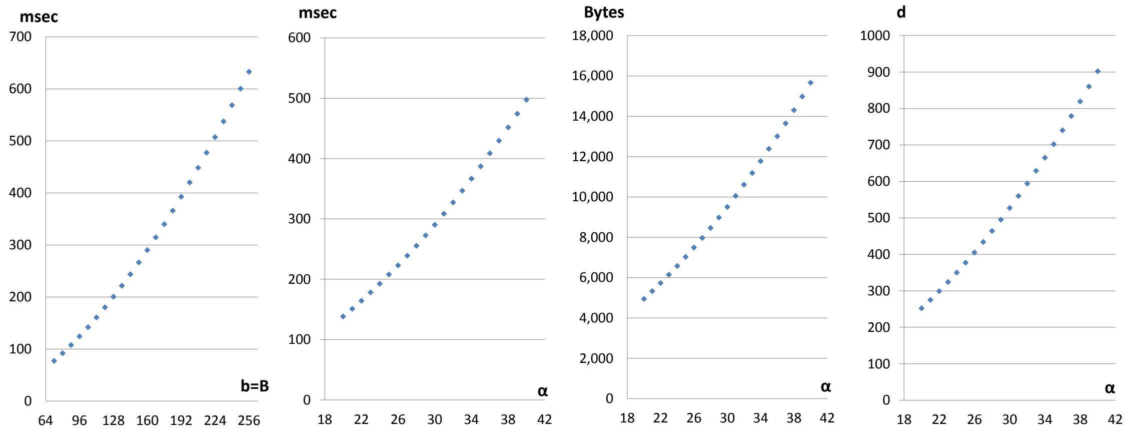

Figure 2. HIMMO Performance.

A detailed description of an implementation of HIMMO and its performance is not feasible due to space reasons. However, we include a comparison table (Table 3) to illustrate the performance advantages of HIMMO compared with ECDH and ECDSA when implementing a simple interaction between two nodes: a first node ξ wants to send in a secure way some information to η and η wants to securely receive the message from ξ and verify its credentials. The first two protocols involve communicating before node ξ can send an encrypted message, whereas HIMMO allows node ξ to directly compute the key with η based on its identifier and send the encrypted message. Also, notice that ECDH only provides key agreement, to get key agreement and verification of credentials is needed to use also ECDSA increasing the resource requirements. The results are based on an implementation on the ATMEGA128L running at 8 MHz and illustrate the performance when this protocol is implemented with ECDH only, ECDH and ECDSA for a security level of 80 bits and HIMMO using security parameters $\alpha = 26$ and $2b = B = 160$. In Table 3, CPU refers to the overall computing needs, the memory refers to the amount of information that needs to be stored in flash, RAM is the RAM memory needs, exchanged data refers to the amount of data exchanged between ξ and η, round trips are the number of interactions between both nodes, and finally, the security properties illustrate the features of the security protocols.

Table 3
HIMMO performance and comparison with ECDH and ECDH+ECDSA.

	CPU time	Key material + code	RAM	Exchanged data	Security properties
ECDH [11]	3.97 s	16018 B	1774 B	480 B	Key agreement
ECDH +ECDSA [11]	11.9 s	35326 B	3284 B	704 B	Key agreement and credential verification
HIMMO	0.290 s	7560 B	1220 B	448 B	Key agreement and credential verification

8 Conclusions and future work

The identity-based KPS introduced so far failed because they cannot be both efficient and secure at the same time.

We describe HIMMO, a KPS with similar operational characteristics as previous schemes but offering full collusion resistance for adequate parameters. HIMMO is non-interactive: nodes can directly generate a common key without exchanging messages, which is particularly attractive in applications requiring short response times. From an implementation point of view, HIMMO is very attractive as well. HIMMO can also be used as a building block for enabling implicit certification and verification of credentials and for enabling secure broadcast. To remove privacy concerns resulting from the TTP knowing the keying material from all nodes and thus being able to construct all keys used in the network, a version of HIMMO with multiple TTPs can be applied.

We have shown the relevance of the HI and MMO problems [10], [13] in the context of the HIMMO scheme and analyzed its security. In particular, we show that HIMMO's design combines both problems preventing an attacker from either attacking the keying material of a node or the root keying material of the TTP. The experimental results from Section 5 strongly suggest that application of the LLL algorithm followed by Babai's nearest plane algorithm used to attack the HI problem does not yield the correct result if $\alpha > 20$. For parameters considered secure according to our analysis, HIMMO performs better than alternative schemes such as ECDH and ECDH combined with ECDSA. The operational features of HIMMO together with its identity-based nature, its fully collusion resistant property and its very low resource needs, make HIMMO a promising candidate to enable secure communication links in many applications areas such as the Internet of Things.

Future work 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 [9] helps analyzing the practical behavior 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 [3] 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 the appendix from $3m - 1$ in a previous version of the paper to the current value of $2m$, and Igor Shparlinski and Berry Schoenmakers for their comments on earlier drafts of this paper.

References

[1] Martin Albrecht, Craig Gentry, Shai Halevi, and Jonathan Katz. Attacking Cryptographic Schemes Based on "Perturbation Polynomials". In *CCS09, Proc. 16th ACM Conference on computer and communications security*, pages 1–10. ACM, 2009.

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

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

[4] R. Blom. An optimal class of symmetric key generation systems. In T. Beth, N. Cot, and I. Ingemarsson, editors, *EUROCRYPT '84*, LNCS 209, pages 335–338. Springer, 1985.

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

[6] D. Boneh and M. Franklin. Identity-Based Encryption from the Weil Pairing. *SIAM Journal of Computing*, 32(3):586–615, 2003.

[7] TU Darmstadt. Welcome to the ideal lattice challenge. Web repository, 2014. http://www.latticechallenge.org.

[8] Eduarda S.V. Freire, Dennis Hofheinz, Eike Kiltz, and Kenneth G. Paterson. Non-interactive key exchange. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *Public-Key Cryptography PKC 2013*, volume 7778 of *Lecture Notes in Computer Science*, pages 254–271. Springer Berlin Heidelberg, 2013.

[9] Nicolas Gama and Phong Q. Nguyen. Predicting lattice reduction. In Nigel P. Smart, editor, *EUROCRYPT 2008*, LNCS 4965, pages 31–51. Springer, 2008.

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

[11] An Liu and Peng Ning. Tinyecc: A configurable library for elliptic curve cryptography in wireless sensor networks. In *Proceedings of the 7th International Conference on Information Processing in Sensor Networks*, IPSN '08, pages 245–256, Washington, DC, USA, 2008. IEEE Computer Society.

[12] T. Matsumoto and H. Imai. On the key predistribution system: a practical solution to the key distribution problem. In C. Pomerance, editor, *Advances in Cryptology – CRYPTO'87*, LNCS 293, pages 185–193. Springer, 1988.

[13] Oscar García Morchon, Ronald Rietman, Igor E. Shparlinski, and Ludo Tolhuizen. Interpolation and approximation of polynomials in finite fields over a short interval from noisy values. *Experimental mathematics*, 23:241–260, 2014.

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

[15] Sage. http://www.sagemath.org.

[16] Adi Shamir. Identity-based cryptosystems and signature schemes. In *Proceedings of CRYPTO 84 on Advances in Cryptology*, pages 47–53, New York, NY, USA, 1985. Springer-Verlag New York, Inc.

[17] Daniel Stehle. Floating-point lll: Theoretical and practical aspects. In *The LLL Algorithm, Survey and Applications*. Springer-Verlag, 2010.

[18] W. Zhang, N. Subramanian, and G. Wang. Lightweight and Compromise-Resilient Message Authentication in Sensor Networks. In *Proceedings IEEE INFOCOM 2008*, 2008.

[19] W. Zhang, M. Tran, S. Zhu, and G. Cao. A Random Perturbation-based Scheme for Pairwise Key Establishment in Sensor Networks. In *8th ACM Int. Symp. on Mobile Ad Hoc Networking and Computing (MobiHoc) 2007*, pages 90–99, 2007.

APPENDIX
VALIDATION OF HIMMO

As stated before, $K_{\xi,\eta}$ and $K_{\eta,\xi}$ need not be equal. In this appendix, we 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 3, any pair of nodes ξ and η arrive at a common key.

Lemma A.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

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

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

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

Proof Using the notation from Lemma A.1, we have

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

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

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

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

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

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

The symmetry of the root keying polynomials implies that

$$\lambda_\xi(\eta) - \lambda_\eta(\xi) = \tilde{\lambda}_\xi(\eta) - \tilde{\lambda}_\eta(\xi). \tag{5}$$

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

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

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

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

Of course, the same bounds are valid for $\tilde{\lambda}_\xi(\eta)$. Combining these bounds with (5), and the fact $\lambda_\xi(\eta) - \lambda_\eta(\xi)$ is an integer number, the theorem follows. □

Corollary A.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 A.1 and the discussion on the sending of additional information at the end of Section 3. □

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