Non-Interactive Hierarchical Pairwise Key Predistribution Scheme with Multi-Level Key Establishment

Qiyan Wang, Himanshu Khurana, Klara Nahrstedt

University of Illinois at Urbana-Champaign,
IL 61801,USA

Abstract. Networking environments with connectivity, bandwidth and computational constraints such as critical infrastructure networks or MANETs benefit from non-interactive key predistribution capabilities. In these networks, nodes can compute shared keys using public identities without the need for interactions once basic key materials are distributed to them. Motivated by the electric power grid, in this paper we propose a novel key predistribution solution for hierarchical networks, namely, Non-interactive Hierarchical pairwise Key Predistribution (NHKP) that supports (1) hierarchical key predistribution and (2) non-interactive multi-level key establishment. In NHKP each node of the hierarchy gets its key materials from its parent node directly and any pair of nodes, even at different levels, can establish a shared key without interactions. To the best of our knowledge, this is the first scheme to date that supports direct multi-level key establishment. NHKP is constructed using multivariate symmetric polynomials and specially designed multivariate perturbation polynomials (PPs). It achieves perfect resistance to collusion attacks at the lowest Q levels (Q is a tunable parameter) and partial resistance at the upper levels. Our scheme provides a partial answer to an open question posed by Gennaro et al. [1] who developed a hierarchical key predistribution scheme with such resistance only at the leaf level. Furthermore, NHKP is efficient in terms of computation and storage overheads. Our prototype implementation shows that NHKP is practical.
Note that although our scheme uses a similar idea of PP, which was originally proposed in [2] and recently broken in [15], our construction for PPs is in nature different from previous constructions. We use multiple variables in PPs (instead of a single variable as in [2]) to introduce more randomness and each PP is constructed on the fly using random Langrange interpolation. Consequently, the space of PPs in our scheme has T dimensions, where T is a large parameter, rather than two dimensions as used in [2]. These make it very hard for the adversary to break any of PPs or the master polynomial. This paper only provides a security analysis of our scheme, and a formal security proof is the focus of our future research.

1 Introduction

Networking environments with connectivity, bandwidth and computational constraints such as critical infrastructure networks or MANETs benefit from non-interactive key predistribution capabilities. In these networks, nodes can compute shared keys using

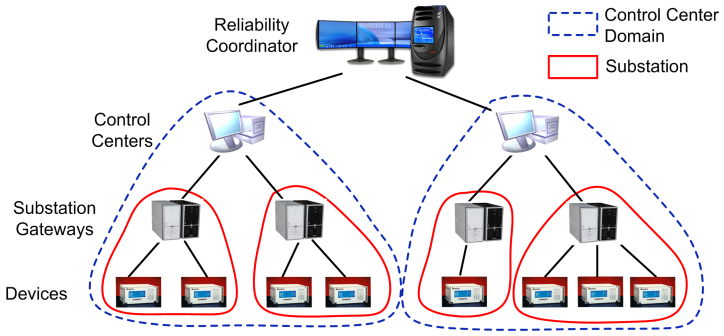


Fig. 1. An example of four-level power grid

public identities without the need for interactions once basic key materials are distributed to them.

One of the largest critical infrastructure networks is the electric power grid, where networks are constrained due to limited connectivity, low-bandwidth communication links and a large number of computation-constraint devices. As shown in Fig. 1, the power grid features a hierarchical structure with upper-level nodes (parent nodes) governing a number of lower-level nodes (child nodes). We believe that heavy-weight and resource-intensive key management solutions like X.509 PKI are not suitable for such environments. Instead, we propose a Non-interactive Hierarchical pairwise Key Predistribution (NHKP) scheme that has two functional features. First, it enables each node of the hierarchy to get its key materials from its parent node directly. Second, any pair of nodes in the hierarchy, even at different levels, can establish a shared key without interactions. This serves as a basis for secured multi-level communications, which are crucial in the power grid to accommodate the coordination of power-load balancing and close estimation of system states. For example, a control center may communicate with some sensing devices in its domain or in other control center domains to collect sensed current/voltage information so as to estimate the system state.

There are three key challenges of designing an effective NHKP scheme: 1) resistance to collusion attacks[1] at all levels of the hierarchy, 2) high computational efficiency, and 3) small storage overheads. The collusion problem arises because the key materials of any nodes are derived from the master key materials (held by the root node) due to the hierarchical nature, and combining individual key materials may infer some knowledge of the master key materials. Recently, Gennaro et al. [1] proposed a scheme that partially addresses the problem in that their scheme provides hierarchical key predistribution, but does **not** support multi-level key establishment. Moreover, this scheme incurs very large computational overheads and is perfectly resistant to collusion attacks only at the leaf level of the hierarchy. How to achieve perfect resistance at all levels is posed by them as an open question.

[1] *Collusion attack* refers to the case where the attacker compromises multiple nodes and combines the secrets learnt from these victim nodes to infer other secrets of non-compromised nodes.

Our Contributions. In this work, we develop a novel NHKP scheme, which supports (1) hierarchical key predistribution and (2) non-interactive multi-level key establishment. NHKP has strong resistance to collusion attacks and very low computation and storage overheads.

We first develop a basic NHKP scheme that supports these two features but is partially resistant to a threshold of node corruptions at each level. This scheme relies on multivariate symmetric polynomial that is an extension of the scheme of Blundo et al. [6]. Based on the basic NHKP scheme, we develop an advanced NHKP scheme that retains all properties of the basic scheme and provides much stronger resistance to collusion attacks. Our approach is inspired by the scheme proposed by Zhang et al. in [2], where they use univariate PPs to design a non-interactive pairwise key predistribution scheme for sensor networks. However, our analysis shows that the approach to constructing univariate PPs in [2] cannot be directly applied to NHKP, because their construction cannot be extended to build multivariate PPs that are necessary for the NHKP scheme. To solve this problem, we design a novel construction algorithm for multivariate PPs by making use of Lagrange Interpolation [4] and some structural features of NHKP. The resultant scheme has perfect resistance to collusion attacks at the lowest Q levels, where Q is a tunable security parameter, and retains partial resistance at upper levels (one can view the scheme of Gennaro et al. [1] as a special case of NHKP with $Q = 1$). The advanced NHKP scheme considerably raises the bar to collusion attacks in that it allows one to choose a suitable Q to cover all those levels that contain easy-to-compromise nodes, and forces attackers to attempt those nodes that are hard to bring down.

We have a prototype implementation of the advanced NHKP scheme. Our performance experiments indicate that NHKP is practical and substantially improves upon prior work by Gennaro et al. [1] (hereafter referred to as Gennaro08). Gennaro08 involves a large number of expensive exponential computations and thus suffers substantial computational overheads. In contrast, our schemes are constructed only using simple modular additions/multiplications and hence are significantly more efficient. Experiments show that the key generation in NHKP is at least thousands of times faster than that of Gennaro08 for a moderately large hierarchy. This renders a significant advantage to NHKP for real-time applications and computationally constrained devices. In addition, the storage overhead of NHKP is smaller than Gennaro08 as well, especially for large hierarchies with strong collusion resistance. To secure a four-level large-scale hierarchy with *perfect* collusion resistance to *all* levels, NHKP only incurs 28.7 KB storage overhead at a leaf node, and less than 2.8 MB storage at any upper-level nodes.

To the best of our knowledge, this is the first hierarchical key predistribution scheme that supports multi-level key establishment. Our approach of using multivariate perturbation polynomials to achieve perfect collusion resistance at the lowest Q levels provides a partial solution to the open question raised by Gennaro et al. [1] Although our work is mainly motivated by the power grid, our solution is general in nature and applicable to any hierarchical networks. For example, consider a hierarchical military MANET where the root node of the hierarchy is the central commander and the second level comprises a number of tanks, each of which has multiple affiliated foot soldiers belonging to the leaf level. In this case, entities at a higher level predistribute key mate-

rials to lower-level entities and any two of them should be able to securely communicate with each other to share battlefield information.

Note that although our scheme uses a similar idea of PP, which was originally proposed in [2] and recently broken in [15], our construction for PPs is in nature different from previous constructions. We use multiple variables in PPs (instead of a single variable as in [2]) to introduce more randomness and each PP is constructed on the fly using random Langrange interpolation. Consequently, the space of PPs in our scheme has T dimensions, where T is a large parameter, rather than two dimensions as used in [2]. These make it very hard for the adversary to break any of PPs or the master polynomial. This paper only provides a security analysis of our scheme (Appendix 2), and a formal security proof is the focus of our future research.

The rest paper is organized as follows. Section 2 gives some background knowledge and describes our basic NHKP scheme. Section 3 elaborates on the construction of multivariate PPs and the advanced NHKP scheme. Section 4 evaluates the NHKP and compares it against Gennaro08. Section 5 presents the related work. Section 6 concludes the paper.

2 Basic NHKP Scheme

Before going into our design details, we first present the threat model, and briefly review the non-interactive polynomial-based key predistribution scheme of Blundo et al. [6] that is the basis of our scheme.

2.1 Threat Model

We assume the attacker can selectively compromise an arbitrary node at his choice, but we assume the root node of the hierarchy is secured and free from compromise. We believe this assumption is realistic since the root node, just like the top-level CA in PKI and the key server in other key management schemes [1][2][3][6], generates all secrets for the entire network, and compromising the root node is tantamount to breaking the whole system. Hence, such key servers are usually fully safeguarded in practice.

Once a node gets compromised, the secrets it knows are revealed to the attacker. Any compromised nodes can be used to launch collusion attacks. If the attacker compromises a (non-leaf) node that is responsible for distributing keys, we assume the attacker can also learn the secrets that this node has distributed. In addition, we assume that the attacker has considerably powerful but bounded computational capability. This means that the attacker could access a very powerful machine or a cluster of machines for parallel computing, but he cannot invert a secure hash function (such as SHA-256).

2.2 Background: A Non-Interactive Polynomial-Based Key Predistribution Scheme

The polynomial-based key predistribution scheme was originally designed by Blundo et al. [6] for conference key agreement. To achieve a n-party, $(t+1)$-secure group key agreement scheme, one needs a n-variable, t-degree symmetric polynomial (as defined

Table 1. Notations

$L+1$	The number of levels of the hierarchy (The root is at level 0, and the leaves are at level L)
Q	The number of levels with perfect collusion resistance
q, c	q is a large prime, $c = \lceil \log_2 q \rceil$
t_l	The degree of variables x_l, y_l, $1 \leq l \leq L$
$\mathcal{I}_l$	The set of identifiers at level l, $1 \leq l \leq L$
$id_{\langle l,u \rangle}$	The identifier of the u-th node at level l, $1 \leq l \leq L$
$null$	The reserved identifier with value equal to 1
$F(x_1, ..., x_L; y_1, ..., y_L)$	The master polynomial over F_q
$G_{\langle l,i \rangle}(x_{l+1}, ..., x_L; y_1, ..., y_L)$	The polynomial of i-th node at level l, $1 \leq l \leq L-1$
$G_{\langle L,i \rangle}(y_1, ..., y_L)$	The polynomial of i-th node at the leaf level
$\phi_{\langle l,i \rangle}(x_{l+1}, ..., x_L; y_1, ..., y_L)$	The PP added to $G_{\langle l,i \rangle}(x_{l+1}, ..., x_L; y_1, ..., y_L)$
$\phi_{\langle L,i \rangle}(y_1, ..., y_L)$	The PP added to $G_{\langle L,i \rangle}(y_1, ..., y_L)$

below). Since our goal is to establish pairwise keys, for simplicity, we only discuss the special case of [6] ($n = 2$), that is, pairwise key establishment using a bivariate symmetric polynomial.

First, the key distributor randomly picks a t-degree bivariate symmetric polynomial $f(x, y) = \sum_{i,j=0}^{t} A_{i,j} x^i y^j$, s.t. $f(x, y) = f(y, x)$, where F_q is a finite field and q is a prime number that is large enough to accommodate cryptographic keys. For each node u, the key distributor assigns it a unique identifier id_u, and then computes and distributes a polynomial share of $f(x, y)$, namely, $f(id_u, y)$, for this node. Consequently, any pair of nodes u and v can compute a shared key using each other's identifiers: node u evaluates its polynomial $f(id_u, y)$ at point $y = id_v$ obtaining $f(id_u, id_v)$, and node v computes the key by evaluating $f(id_v, y)$ at $y = id_u$. Since $f(id_u, id_v) = f(id_v, id_u)$, the keys generated by u and v are identical and can be used to encrypt/decrypt correspondences.

The security proof in [6] ensures that this scheme is unconditionally secure and $(t+1)$-collusion resistant. That is, the coalition of no more than t compromised nodes knows nothing about the pairwise key between any two non-compromised nodes.

2.3 Description of the Basic NHKP Scheme

We consider a $(L+1)$-level hierarchy, where the root is at level 0 and the leaves reside at level L. At the initialization, the root node constructs a random $2L$-variable symmetric polynomial over F_q as the master polynomial.

$$F(x_1, ..., x_L; y_1, ..., y_L) = \sum_{i_1, j_1 = 0}^{t_1} \cdots \sum_{i_L, j_L = 0}^{t_L} A_{i_1, ..., i_L; j_1, ..., j_L} x_1^{i_1} \cdots x_L^{i_L} y_1^{j_1} \cdots y_L^{j_L}$$

$$s.t. \quad F(..., x_i, ...; ..., y_i, ...) = F(..., y_i, ...; ..., x_i, ...), \quad \forall i \in [1, L]$$

This master polynomial (i.e., coefficients) is secretly kept by the root node.

Key predistribution. From the second level down to the leaf level, each node obtains from its parent a unique polynomial share that is derived from its parent's poly-

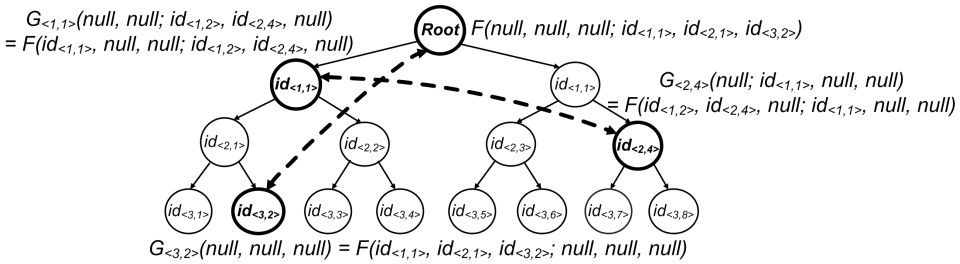


Fig. 2. Examples of key establishment in a four-level hierarchy

nomial according to its identifer. We let $id_{\langle l,u\rangle}$ denote the identifier assigned to the u-th node at level l. For expression simplicity, we will use $id_{\langle l,u\rangle}$ to refer to nodes in the remaining paper. For example, "$id_{\langle 2,1\rangle}$ establishes a key with $id_{\langle 3,4\rangle}$" means "the 1-st node at level 2 establishes a key with the 4-th node at level 3". The detailed key predistribution process is described as follows.

- For each node $id_{\langle 1,i\rangle}$ at level 1, the root node computes and distributes a $(2L-1)$-variable polynomial share $G_{\langle 1,i\rangle}(x_2,...,x_L;y_1,...,y_L) = F(id_{\langle 1,i\rangle},x_2,...,x_L;y_1,...,y_L)$.
- For each node $id_{\langle l,u\rangle}$ at level l $(2 \leq l \leq L-1)$, its parent $id_{\langle l-1,v\rangle}$ calculates and distributes a $(2L-l)$-variable polynomial $G_{\langle l,u\rangle}(x_{l+1},...,x_L;y_1,...,y_L)$ = $G_{\langle l-1,v\rangle}(id_{\langle l,u\rangle}, x_{l+1},...,x_L;y_1,...,y_L)$.
- For each leaf node $id_{\langle L,u\rangle}$, its parent $id_{\langle L-1,v\rangle}$ calculates and distributes a L-variable polynomial share $G_{\langle L,u\rangle}(y_1,...,y_L) = G_{\langle L-1,v\rangle}(id_{\langle L,u\rangle};y_1,...,y_L)$.

Multi-level key establishment. Using the predistributed polynomials, any pair of nodes can establish a unique pairwise key. We first introduce the notion of *Identifier Vector* (IV) that is necessary for the key establishment. Each node in the hierarchy has a unique IV that comprises a sequence of L identifiers. Consider an arbitrary node $id_{\langle l,u\rangle}$ at level l, $1 \leq l \leq L$. Suppose the path from the root to $id_{\langle l,u\rangle}$ is "root $\rightarrow id_{\langle 1,v_1\rangle} \rightarrow \cdots \rightarrow id_{\langle l-1,v_{l-1}\rangle} \rightarrow id_{\langle l,u\rangle}$". Then the IV of $id_{\langle l,u\rangle}$ is defined as $(id_{\langle 1,v_1\rangle},...,id_{\langle l-1,v_{l-1}\rangle},id_{\langle l,u\rangle},null,...,null)$ with the first l elements identical to the identifiers of the nodes along the path and the last $L-l$ elements equal to $null$, where $null$ is a reserved identifier with value equal to 1[2]. For example, in Fig. 2, the IV of node $id_{\langle 1,1\rangle}$ is $(id_{\langle 1,1\rangle},null,null)$, and the IV of node $id_{\langle 3,2\rangle}$ is $(id_{\langle 1,1\rangle},id_{\langle 2,1\rangle},id_{\langle 3,2\rangle})$. Specially, the IV of the root node has all elements equal to $null$.

To compute a pairwise key, each participant evaluates its polynomial by fixing all x's (if any) to be $null$ and setting all y's as the elements of the peer's IV. In particular, consider a pair of nodes $id_{\langle l_1,u\rangle}$, $id_{\langle l_2,v\rangle}$, $0 \leq l_1,l_2 \leq L$, whose IVs are $(\delta_1,...,\delta_L)$ and $(\eta_1,...,\eta_L)$ respectively. To setup a shared key, node $id_{\langle l_1,u\rangle}$ calculates $G_{\langle l_1,u\rangle}(null,...,null;\eta_1,...,\eta_L)$, while node $id_{\langle l_2,v\rangle}$ computes $G_{\langle l_2,v\rangle}(null,...,null;$

[2] The value of $null$ can be set to be any constant positive integer. To minimize the computational cost of key generation, we choose $null = 1$.

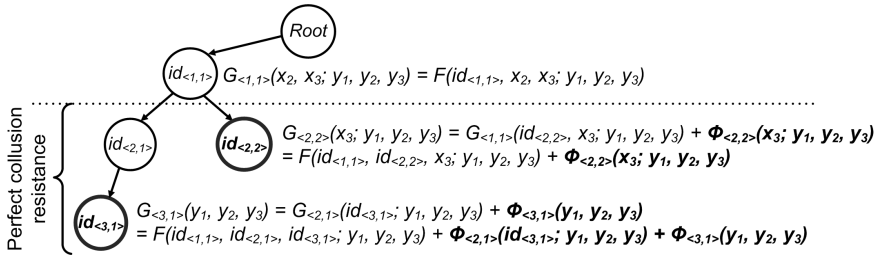


Fig. 3. An example of key predistribution in the advanced NHKP scheme with four levels and $Q = 2$

$\delta_1, ..., \delta_L)$. Because of the symmetry of the master polynomial, the two separately computed results are identical and can be used as the shared key. Fig. 2 gives some examples of the key establishment.

This basic NHKP scheme is resistant to a threshold number of node corruptions. If the attacker compromises more than $\frac{t_L+2}{2}$ nodes at level l ($1 \leq l \leq L$), then he may learn the secrets of a node at level $l-1$; if there are more than $\prod_{i=1}^{l} \frac{t_i+2}{2}$ compromised nodes at level l, then the attacker could break the master polynomial and learn all the secrets of the system. A straightforward method to improve collusion resistance is to use high-degree polynomials (i.e., increasing t_i), but this may incur huge storage overheads. Sec. 3 will present an efficient way to strengthen the resistance to collusion attacks.

3 Advanced NHKP Scheme

In this section, we first introduce the key idea of the advanced NHKP scheme, and then present the challenges and solutions to constructing multivariate perturbation polynomials. Finally, we give a full description the advanced NHKP.

3.1 Basic Idea

The advanced NHKP is designed based on the basic scheme presented in Sec. 2. To achieve strong resistance to collusion attacks, we introduce random Perturbation Polynomials (PPs) in the key predistribution to randomize the key materials held by the nodes. In particular, instead of giving each node the original polynomial share (e.g., $G_{\langle 1,1\rangle}(id_{\langle 2,2\rangle}, \cdots)$ for node $id_{\langle 2,2\rangle}$ in Fig. 3), the key distributor provides a perturbed polynomial ($G_{\langle 2,2\rangle}(\cdots)$) that is obtained by adding the original share ($G_{\langle 1,1\rangle}(id_{\langle 2,2\rangle}, \cdots)$) with a random PP ($\phi_{\langle 2,2\rangle}(\cdots)$). As a result, even if a node is compromised, the attacker is unable to extract from the exposed perturbed polynomial ($G_{\langle 2,2\rangle}(\cdots)$) the original share ($G_{\langle 1,1\rangle}(id_{\langle 2,2\rangle}, \cdots)$), which is necessary to infer the secrets of non-compromised nodes. In order to hide the original polynomial share behind the perturbed polynomial, the PP should contain the same number of variables as the original polynomial. In particular, the PPs constructed for level l, $1 \leq l \leq L$, should contain $2L - l$ variables.

For the correctness of key establishment, the construction of PPs must ensure that any pair of perturbed polynomials (e.g., $G_{\langle 2,2\rangle}(\cdots)$ and $G_{\langle 3,1\rangle}(\cdots)$) are still able to

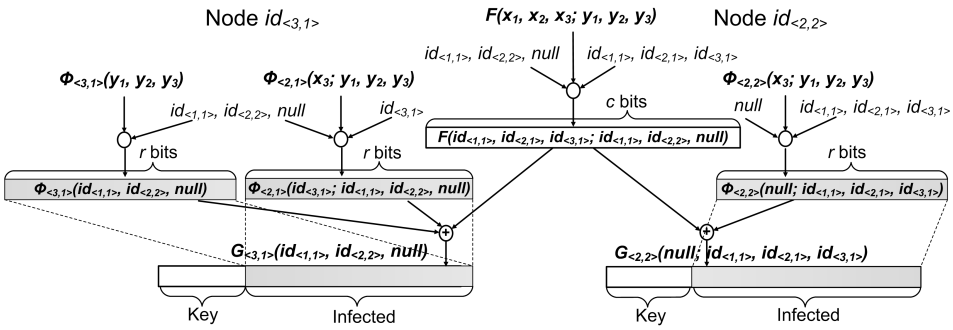


Fig. 4. An example of key establishment in the advanced NHKP scheme (between $id_{\langle 3,1\rangle}$ and $id_{\langle 2,2\rangle}$ according to Fig. 3)

derive an identical key even if they comprise totally different PPs ($\phi_{\langle 2,2\rangle}(\cdots)$ and $\phi_{\langle 3,1\rangle}(\cdots)$). One way to achieve this is to make the output length r (w.r.t. bits) of each PP shorter than that of the original polynomials ($c = \lceil \log_2 q \rceil$, $c > r$), so that the most significant $c-r$ bits of the results of perturbed polynomials are not changed by the PPs and can be used as the shared key[3]. This property of PPs is called *partial infection*, and r is referred to as *infection length*. Fig. 4 gives an example of key establishment with the presence of PPs.

Due to the existence of PPs, the length of the truncated key segment (that is the most significant $c - r$ bits of the results of perturbed polynomials) may be not long enough to be directly used as a cryptographic key. To solve this problem, multiple master polynomials can be used simultaneously and each of them generates a $(c - r)$-bit key segments; consequently, applying a hash computation on the concatenation of these key segments can produce a sufficiently strong key.

3.2 Challenges of Constructing Multivariate Perturbation Polynomials

A desired construction for multivariate PPs should not only ensure the correctness of key establishment, but also incur small storage and computation overheads and scale to large hierarchies. This is a quite challenging problem. One way to construct k-variable PPs is to multiply k univariate PPs together. That is, $\phi(z_1, ..., z_k) = \prod_{i=1}^{k} \alpha_i(z_i)$, where $\alpha_i(z_i)$ $(1 \leq i \leq k)$ is a r_i-bit univariate PP; $\phi(z_1, ..., z_k)$ is a $(\Sigma_{i=1}^{k} r_i)$-bit k-variable PP as long as $\Sigma_{i=1}^{k} r_i < c$. Zhang et al. [2] designed an algorithm to construct univariate PPs. However, the infection length of each PP generated by this algorithm is at least $\frac{c}{2}$, and thus the multiplication result of these PPs will not have the property of partial infection (i.e., $\Sigma_{i=1}^{k} r_i \geq c$).

An alternative way to construct univariate PP is based on *Lagrange interpolation*, which is defined as follows [4].

[3] In some cases, the most significant $c - r$ bits are changed due to the carry generated in the addition of the least significant r bits, but the result is predictable and hence can be rectified. We refer to Appendix 1 for more details.

Definition 1. *Given a set of* $n+1$ *data points* (a_0, b_0), ..., (a_n, b_n), *where no two* a_j *are the same, the Lagrange interpolation polynomial is a linear combination* $L(x) = \sum_{j=0}^{n} b_j l_j(x)$ *of Lagrange basis polynomials* $l_j(x) = \prod_{i=0, i\neq j}^{n} \frac{x-a_i}{a_j-a_i}$.

The idea of using Lagrange interpolation to construct univariate PPs (with domain $\mathcal{I}$ and infection length r) is quite simple. For each element $a_i \in \mathcal{I}$, we pick a random b_i from $\{0, ..., 2^r - 1\}$, and use (a_i, b_i), $x_i \in \mathcal{I}$, as the data points to construct the Lagrange interpolation polynomial that passes through all these points. Obviously, the obtained polynomial is a satisfactory r-bit PP with domain $\mathcal{I}$. Unlike the algorithm of Zhang et al. [2] that requires $r \geq \frac{c}{2}$, Lagrange interpolation allows choosing an arbitrary injection length r as long as the probability that the attacker can correctly guess all the values of b_i's is negligibly small. At the downside, Lagrange interpolation is hard to scale to a very large domain $\mathcal{I}$, since the degree of the generated Lagrange interpolation polynomial is determined by the size of $\mathcal{I}$.

3.3 Novel Construction for Multivariate Perturbation Polynomials

As presented in Sec. 3.1, NHKP requires $(2L - l)$-variable PPs $\phi_{\langle l,-\rangle}(x_{l+1}, ..., x_L; y_1, ..., y_L)$ for level l, $L - Q + 1 \leq l \leq L$ (the leaf-level PPs $\phi_{\langle L,-\rangle}(y_1, ..., y_L)$ can be viewed as a special case where no x_-'s variables are contained). Sec. 3.2 shows that constructing such multivariate PPs in a general case is quite hard. However, we can make use of some structural features of NHKP to design specific multivariate PPs that only serve NHKP.

Firstly, recall that in the key establishment the value of x_j $(l+1 \leq j \leq L)$ is always $null$; as for key predistribution, the inputs taken by x_j $(l+1 \leq j \leq L)$ in $\phi_{\langle l,u\rangle}(x_{l+1}, ..., x_L; y_1, ..., y_L)$ are the identifers of the descendent nodes of $id_{\langle l,u\rangle}$ that reside at level j, which we denote as $\mathcal{I}_{j,\langle l,u\rangle}$ (e.g., in Fig. 2, $\mathcal{I}_{3,\langle 2,4\rangle} = \{id_{\langle 3,7\rangle}, id_{\langle 3,8\rangle}\}$). Therefore, the range of values of x_j in any evaluation of $\phi_{\langle l,u\rangle}(x_{l+1}, ..., x_L; y_1, ..., y_L)$ is $\mathcal{I}_{j,\langle l,u\rangle} \cup \{null\}$, which contains much fewer elements than $\mathcal{I}_j$. Another observation is that the number of nodes at one of few top levels is much smaller than that of nodes at bottom levels. For example, in the power-grid hierarchy of a reliability coordinator, there will be only tens of control centers at the second level but hundreds of thousands of devices at the leaf level. In the following construction, we introduce a tunable parameter J to select those top-level domains $(\mathcal{I}_1, ..., \mathcal{I}_J)$ that contain a limited number of elements. Consequently, based on these two observations, we construct the multivariate PPs as follows.

Construction 1.

$$\phi_{\langle l,u\rangle}(x_{l+1}, ..., x_L; y_1, ..., y_L) = \sum_{\omega=1}^{\lambda} \prod_{\tau=l+1}^{L} \alpha^{(\tau)}_{\omega,\langle l,u_\eta\rangle}(x_\tau) \prod_{\sigma=1}^{J} \beta^{(\sigma)}_{\omega,\langle l,u_\eta\rangle}(y_\sigma)\gamma_\omega(y_{J+1}, ..., y_L)$$

where

- $\alpha^{(\tau)}_{-,\langle l,u\rangle}(x_\tau)$ is a r_1-bit univariate PP over domain $\mathcal{I}_{\tau,\langle l,u\rangle} \cup \{null\}$, and is independently computed using Lagrange interpolation by choosing random data points (a, b), $b \in_R \{0, ..., 2^{r_1} - 1\}$, $a \in \mathcal{I}_{\tau,\langle l,u\rangle}$.
- $\beta^{(\sigma)}_{-,\langle l,u\rangle}(y_\sigma)$ is a r_2-bit univariate PP over domain $\mathcal{I}_\sigma \cup \{null\}$, and is independently computed using Lagrange interpolation with random data points (a, b), $b \in_R \{0, ..., 2^{r_2} - 1\}$, $a \in \mathcal{I}_\sigma$.

- $\gamma_{-}(y_{J+1}, ..., y_L)$ is a r_3-bit multivariate PP (where $r_3 < c - (Q-1)r_1 - J \cdot r_2$) with domains $\mathcal{I}_k$, $J+1 \leq k \leq L$. Its construction will be described later.
- λ, J, r_1, r_2 and r_3 are tunable parameters. The selections of these parameters will be shown in Sec. 4.

In Construction 1, we use Lagrange interpolation to cover those variables with relatively small domains (i.e., $x_{l+1}, ..., x_L$, $y_1, ..., y_J$), and consequently the generated univariate PPs will have fairly small degrees and short injection lengths. The rest variables are covered by $\gamma_{-}(y_{J+1}, ..., y_L)$ that must be able to scale to very large domains (i.e., $\mathcal{I}_k$, $J+1 \leq k \leq L$).

Now we show how to build $\gamma_{-}(y_{J+1}, ..., y_L)$. Our construction is based on another characteristic of NHKP. That is, the input of $(y_{J+1}, ..., y_L)$ is always a part of some node's IV. Hence, instead of ensuring for any combination $(\delta_{J+1}, ..., \delta_L) \in \mathcal{I}_{J+1} \times \cdots \times \mathcal{I}_L$, $\gamma_{-}(\delta_{J+1}, ..., \delta_L) \leq 2^{r_3} - 1$, we **only** need to consider those combinations that are parts of valid IVs. For instance, according to Fig. 2, it is not necessary to ensure $\gamma_{-}(id_{\langle 2,4 \rangle}, id_{\langle 3,3 \rangle}) \leq 2^{r_3} - 1$ (where $J = 1$), since $(id_{\langle 2,4 \rangle}, id_{\langle 3,3 \rangle})$ is not a part of any node's IV.

Based on this observation, we design an algorithm to select suitable identifiers for a set Υ of pre-selected random polynomials $\{\gamma_i(y_{J+1}, ..., y_L)\}$ such that each of them is a valid r_3-bit PP with domains that are made up of the selected identifiers. This algorithm selects the identifiers starting from level $J+1$ down to level L. At each level $k \in [J+1, L]$, it outputs the suitable $\mathcal{I}_k$ together with the "child-and-parent" relationship between each identifier in $\mathcal{I}_k$ and some identifier in $\mathcal{I}_{k-1}$, e.g., $id_{\langle k,u \rangle} \in \mathcal{I}_k$ is bound as a child to $id_{\langle k-1,v \rangle} \in \mathcal{I}_{k-1}$. Linking these identifiers enables the identifier selection process to only focus on valid IVs, thereby considerably reducing the number of testing cases. A concrete description of this algorithm is as below.

Construction 2: We first select a set Υ of λ random polynomials $\gamma_i(y_{J+1}, ..., y_L)$ over F_q, s.t., $\gamma_i(null, ..., null) \leq 2^{r_3} - 1$, $1 \leq i \leq \lambda$. For level $J+1$, we form $\mathcal{I}_{J+1}$ by collecting identifiers $\delta_{J+1} \in \{2, ..., q-1\}$ (recall that value 1 is reserved as $null$), s.t., $\forall i \in [1, \lambda]$, $\gamma_i(\delta_{J+1}, null, ..., null) \leq 2^{r_3} - 1$. For level l ($J+2 \leq l \leq L$), for each $\delta_l \in \{2, ..., q-1\}$ we choose a random "parent" identifier δ^*_{l-1} from $\mathcal{I}_{l-1}$ for δ_l, and test if $\forall i \in [1, \lambda]$, $\gamma_i(\delta^*_{J+1}, ..., \delta^*_{l-1}, \delta_l, null, ..., null) \leq 2^{r_3} - 1$ (if $l = L$, no $null$ is contained), where δ^*_j is the parent of δ^*_{j+1} ($J+1 \leq j \leq l-1$). If δ_l meets the conditions, we add δ_l into $\mathcal{I}_l$ and bind it to δ^*_{l-1} as a child.

If the values of $\gamma_i(y_{J+1}, .., y_L)$, $1 \leq i \leq \lambda$, are uniformly distributed over $\{0, ..., q-1\}$, the expected size of each domain $\mathcal{I}_k$, $J+1 \leq k \leq L$, is $2^c \cdot (\frac{2^{r_3}}{2^c})^{\lambda} = 2^{c-\lambda(c-r_3)}$, which can support very large hierarchies with a large c and a medium λ. For those upper levels (e.g., $k = J+1$) that may contain a relatively small number of nodes, the identifier selection process can terminate after collecting enough identifiers, instead of exploring the entire identifer space $\{2, ..., q-1\}$. The complexity of this algorithm is $O((L-J)q)$ (note that the algorithm of Zhang et al. [2] for constructing univariate PPs has the complexity of $O(q)$, but their algorithm only constructs a single domain).

Table 2. Advanced NHKP scheme

Initialization
Inputs: $F_q, c, r, r_3, \lambda, \chi$ // *χ is the length of keys to be generated.* The root node picks π random master polynomials $F^{(s)}(x_1,...,x_L;y_1,...,y_L)$ over F_q, $1 \leq s \leq \pi$, where $\pi = \frac{\chi}{c-r}$. Then the root follows Construction 2 to generate a set of r_3-bit ($r_3 < r$) PPs $\Upsilon = \{\gamma_i(x_{J+1},...,x_L)\}_{1\leq i\leq\lambda}$ with domains $\mathcal{I}_{J+1},...,\mathcal{I}_L$, and selects $\mathcal{I}_1,...,\mathcal{I}_J$ at random.
Key predistribution
At level $l = 1$: The root assigns each node u at level 1 an identifier $id_{\langle 1,u\rangle} \in \mathcal{I}_1$, and then computes $G^{(s)}_{\langle 1,u\rangle}(x_2,...,x_L;y_1,...,y_L) = F^{(s)}(id_{\langle 1,u\rangle},x_2,...,x_L;y_1,...,y_L)$, $1 \leq s \leq \pi$, which are distributed to $id_{\langle 1,u\rangle}$ together with Υ and $\mathcal{I}_{k,\langle 1,u\rangle} \subset \mathcal{I}_k$, $2 \leq k \leq L$.
At level $l \in [2, L-Q]$: Each node $id_{\langle l-1,u\rangle}$ at level $l-1$ assigns each of its children an identifier $id_{\langle l,v\rangle} \in \mathcal{I}_{l,\langle l-1,u\rangle}$, and then computes $G^{(s)}_{\langle l,v\rangle}(x_{l+1},...,x_L;y_1,...,y_L) = G^{(s)}_{\langle l-1,u\rangle}(id_{\langle l,v\rangle},x_{l+1},...,x_L;y_1,...,y_L)$, $1 \leq s \leq \pi$, which are distributed to $id_{\langle l,v\rangle}$ together with Υ and $\mathcal{I}_{k,\langle l,v\rangle} \subset \mathcal{I}_{k,\langle l-1,u\rangle}$, $l+1 \leq k \leq L$.
At level $l \in [L-Q+1, L]$: Each node $id_{\langle l-1,u\rangle}$ at level $l-1$ assigns each of its children an identifier $id_{\langle l,v\rangle} \in \mathcal{I}_{l,\langle l-1,u\rangle}$, and then computes $G^{(s)}_{\langle l,v\rangle}(x_{l+1},...,x_L;y_1,...,y_L) = G^{(s)}_{\langle l-1,u\rangle}(id_{\langle l,v\rangle},x_{l+1},...,x_L;y_1,...,y_L) + \phi^{(s)}_{\langle l,v\rangle}(x_{l+1},...,x_L;y_1,...,y_L)$, $1 \leq s \leq \pi$, where $\phi^{(s)}_{\langle l,v\rangle}(x_{l+1},...,x_L;y_1,...,y_L)$ is constructed using Υ according to Construction 1. The key materials distributed to $id_{\langle l,v\rangle}$ include $\{G^{(s)}_{\langle l,v\rangle}(x_{l+1},...,x_L;y_1,...,y_L)\}_{1\leq s\leq\pi}$, $\{\mathcal{I}_{k,\langle l,v\rangle}\}_{l+1\leq k\leq L}$, and Υ.
Multi-level key establishment
Inputs: $(\delta_1,...,\delta_L)$ // *$(\delta_1,...,\delta_L)$ is the peer's IV.* If any node $id_{\langle l,u\rangle}$ ($0 \leq l \leq L$) wants to establish a shared key with the node having IV $(\delta_1,...,\delta_L)$, it first evaluates $KS^{(s)} = trunc_r(G^{(s)}_{\langle l,u\rangle}(null,...,null;\delta_1,...,\delta_L))$, $1 \leq s \leq \pi$, where $trunc_r(\cdot)$ is a function that truncates a binary string into the first r bits. Then $id_{\langle l,u\rangle}$ computes $Hash(KS^{(1)}||\cdots||KS^{(\pi)})$, which will form the shared key.

3.4 Full Description of the Advanced NHKP

Similar to the basic scheme, the advanced NHKP is also made up of three parts: initialization, key predistribution, and multi-level key establishment (as shown in Table 2). As we mentioned before, to generate sufficiently strong cryptographic keys, we need multiple master polynomials, each of which produces a $(c-r)$-bit key segment. At the initialization, the root node constructs a set Υ of PPs according to construction 2, which are to be given to all nodes in the hierarchy. The key predistribution process at the top $L-Q$ levels is the same as that of the basic scheme. For the nodes at the lowest Q levels, each of their polynomials contains a random PP that is constructed using Υ according to Construction 1. Using these predistributed polynomials, any pair of nodes can establish multiple shared key segments, and concatenating these key segments generates the shared key.

Like the basic NHKP, the advanced scheme provides each node residing at the top $L - Q$ levels with the original shares of the master polynomials and thus is partially resistant to a threshold of node corruptions at these $L - Q$ levels. The threshold is the same as that of the basic scheme. Whereas, because of the existence of PPs at the lowest Q levels, the advanced can tolerate any number of node compromises at these Q levels as long as appropriate parameters are selected. Formally, we have the following theorem.

Theorem 1. *In the advanced NHKP scheme with parameters λ, J and $\{t_k\}_{1\leq k\leq L}$, the attacker cannot learn any secrets of non-compromised nodes no matter how many nodes at the lowest Q levels he can compromise, as long as $\lambda \geq \prod_{k=J+1}^{L}(t_k + 1)$.*

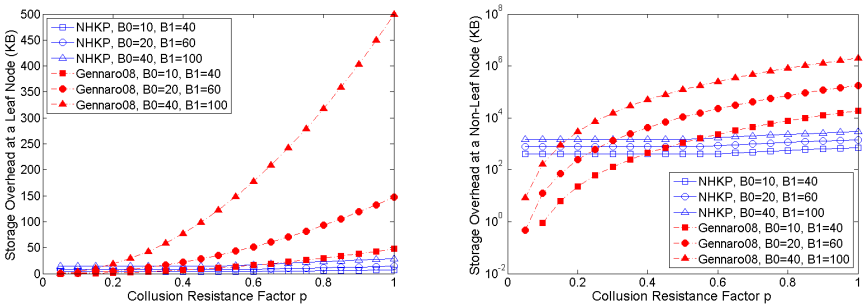
Proof. See Appendix 2. □

4 Evaluation

In this section, we evaluate the performance of the (advanced) NHKP scheme and compare it against Gennaro08 [1]. In particular, we measure how much storage and computation costs are incurred in both schemes to achieve a certain level of resistance to collusion attacks. To evaluate the collusion resistance, we adopt the metric of *(collusion) resistance factor*, which represents what faction of children of a node u that the attacker needs to compromise in order to learn the secrets of u. For the top $L-Q$ levels in NHKP and the non-leaf levels in Gennaro08, the resistance factor of level l is calculated as $\rho_l = \frac{t_l+2}{2\cdot B_{l-1}}$, where $\frac{t_l+2}{2}$ is the threshold for level l and B_{l-1} is the maximum number of children of any node at level $l-1$. For those levels with perfect resistance to collusion attacks, $\rho_l = 1$. Using resistance factor, we can also evaluate the hardness to break the master polynomials. In particular, the attacker has to compromise at least a fraction of $\prod_{j=1}^{l}\rho_j$ of nodes at level l, $1 \leq l \leq L-Q$, to break any master polynomial. We let $\rho = min\{\rho_i : 1 \leq i \leq L\}$.

We implement a prototype of NHKP for four-level hierarchies of different sizes. We choose $Q = 2$ and $J = 1$ to fully protect the lowest two levels. Since the resistance of these two levels is guaranteed by the PPs instead of the thresholds (that are determined by the degrees of master polynomials), we can choose small degrees for these two levels (i.e., t_2 and t_3) to minimize storage overheads. Hence, we choose $t_2 = t_3 = 1$. To obtain 128-bit keys, we use 4 master polynomials (i.e., $\pi = 4$) with output length 188 bits each (i.e., $c = 188$). We construct a set Υ of 4 bivariate PPs $\gamma_i(y_2, y_3)$ with infection length 146 bits (i.e., $\lambda = 4$, $r_3 = 146$), and the infection length of each univariate PP (i.e., $\alpha^{(\tau)}_{\omega,\langle l,u\rangle}$ and $\beta^{(\sigma)}_{\omega,\langle l,u\rangle}$ in Construction 1) is 5 bits (i.e., $r_1 = r_2 = 5$). This configuration can support up to 2^{20} nodes at each level. Tuning these parameters to scale to even larger hierarchies is feasible.

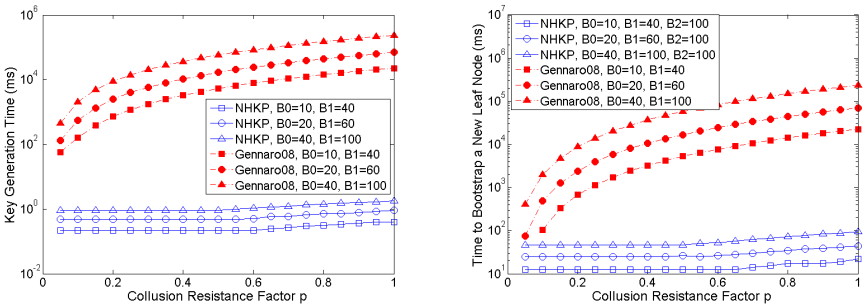
We evaluate the storage overheads by measuring the size of polynomials stored at a leaf node and the maximum storage cost at any non-leaf nodes. Fig. 5 shows that the NHKP incurs small storage overheads at each node, even for very large hierarchies with high requirement on collusion resistance. To achieve $\rho = 1$ in a hierarchy with 40 second-level nodes, 4,000 third-level nodes and 400,000 leaf nodes, the storage overhead at each leaf node is only 28.7 KB and the storage at any other upper-level node



(a) Storage overhead at a leaf node

(b) Maximum storage overhead at any non-leaf node (including the root)

Fig. 5. Comparison between NHKP and Gennaro08 [1] on storage overheads



(a) Time to generate a shared key

(b) Time to bootstrap a new leaf node

Fig. 6. Comparison between NHKP and Gennaro08 [1] on computational overheads

(including the root node) is less than 2.8 MB. Whereas, the storage overheads of Gennaro08 increase rapidly with the resistance factor ρ and the size of the hierarchies. In the same case described above, the storage costs at a leaf node and the root node in Gennaro08 are 498 KB and 1.9 GB respectively.

As for the computational overheads, we measures the computing times to generate a shared key and to derive key materials to bootstrap a new leaf node. We implement the prototype using Miracl [16] on a machine with dual 2.26 GHz CPUs and 2GB RAM. The experimental results (Fig. 6) show that the NHKP is significantly more efficient than Gennaro08. In NHKP, the computing time to generate a shared key or to bootstrap a new leaf node increases slightly with collision resistance ρ and the hierarchy size. For $\rho = 1$ and large-scale hierarchies, it only takes the NHKP less than 2 ms to generate a shared key, which is 10,000 to 100,000 times faster than Gennaro08. Table 3 summarizes the comparisons between NHKP and Gennaro08.

Table 3. Comparison between NHKP and Gennaro08 [1]

Schemes	Multi-level key estab.	Computational efficiency	Collusion resistance		Storage overhead
			Leaf level	Non-leaf level	
Gennaro08	–	Low	Perfect	Threshold	Medium
NHKP	✓	Very high	Perfect	Perfect to $Q-1$ levels Threshold to others	Small

5 Related Work

Non-interactive key predistribution enables communication parties to compute the shared key using public identities without need for interactions. Blom [5] proposed a matrix-based non-interactive pairwise key predistribution scheme that is resistant to a threshold of λ node corruptions. Blundo et al. [6] put forward a non-interactive group key predistribution scheme that relies on multivariate symmetric polynomials and also achieves λ-collusion resistance. Hanaoka et al. [7] show that if many pairs of nodes in the network never communicate, one can improve the collusion resistance of these two schemes [5][6] by a significant factor via removing unused communication links. Zhang et al. [2] gave an alternative approach to strengthen the scheme of Blundo et al. [6] using univariate perturbation polynomials. Sakai et al. [3] relied on pairing-based cryptosystem to design a non-interactive pairwise key establishment scheme that achieves perfect collusion resistance against any number of node corruptions. All the above schemes [5][6][7][2][3] are flat-structured and do not support hierarchical key predistribution. Consequently, each node has to get the key material from a central key distributor, which makes them unsuitable for hierarchical networks.

Ramkumar et al. [8] constructed a hierarchical pairwise key predistribution scheme by extending the probabilistic key predistribution scheme designed for sensor networks [9]. In their scheme, each node recursively inherits a subset of keys (called *key ring*) from its parent, and the common keys of two nodes serves the shared key. Strictly, this scheme is not a non-interactive key predistribution scheme in that when two nodes do not share any common key, they may need to contact an (or multiple) intermediate node(s) to setup a shared key indirectly. Gennaro et al. [1] designed a non-interactive hierarchical key predistribution scheme by combining the scheme of Blundo et al. [6] with the identity-based key establishment scheme [3]. The resultant scheme is perfectly resistant to collusion attacks of leaf nodes, and has partial collusion resistance for any upper levels. This scheme incurs very large computational overheads, and does not allow any non-leaf node in the hierarchy to securely communicate with other nodes. Huang and Medhi [10] proposed a three-level hierarchical group key predistribution scheme for MANETs, aiming for flexible group key establishment when a node moves from one group to another group. However, this is not a non-interactive scheme since communications between two groups are needed to agree on a roaming key. Matt [11] described an identity-based pariwise key predistribution scheme by combining Sakai et al.'s scheme [3] with Blundo et al.'s scheme [6]. Nevertheless, this is not a hierarchical key predistribution scheme in that it requires each node to contact the central key authority directly in the key predistribution. Following the identity-based encryp-

tion scheme of Boneh and Franklin [12], Horwitz and Lynn [13] initiated a study of hierarchical identity-based encryption schemes. They designed a three-level scheme that is perfectly resistant against collusion attacks at the second level and has partial resistance for the leaf nodes. Gentry and Silverberg [14] proposed a L-level hierarchical identity-based encryption scheme with perfect collusion resistance. However, their scheme incurs larger communication and computational overheads. These encryption functionalities [13][14] can be utilized to establish pairwise keys but require interactions.

6 Conclusion

In this paper, we propose a novel NHKP scheme that provides hierarchical key predistribution and multi-level key establishment. Our scheme has strong resistance to collusion attacks. We achieve this by adding a random multivariate perturbation polynomial to each polynomial share in key predistribution. The resultant scheme is perfectly resistant to collusion attacks at the lowest Q levels and partially resistant to a threshold of node corruptions at upper levels. We design an effective algorithm to construct such multivariate PPs by making use of Lagrange interpolation and some structural features of NHKP. The prototype implementation shows that the NHKP is efficient in terms of storage and computation overheads and can scale to very large hierarchies.

References

1. R. Gennaro, S. Halevi, H. Krawczyk, T. Rabin, S. Reidt, and S. D. Wolthusen, "Strongly-resistant and non-interactive hierarchical key-agreement in MANETs", *The 13th European Symposium on Research in Computer Security (ESORICS'08)*, 2008.
2. W. Zhang, M. Tran, S. Zhu, and G. Cao, "A random perturbation-based scheme for pairwise key establishment in sensor networks", *The 8th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'07)*, 2007.
3. R. Sakai, K. Ohgishi, and M. Kasahara, "Cryptosystems based on pairings", *In Proceedings of Symposium on Cryptography and Information Security (SCIS)*, 2000.
4. R. Séroul, "Lagrange interpolation", §10.9 in *Programming for Mathematicians. Berlin: Springer-Verlag*, pp. 269-273, 2000.
5. R. Blom, "An optimal class of symmetric key generation systems", *Annual International Conference on the Theory and Applications of Cryptographic Techniques (Eurocrypt'84)*, LNCS, vol. 209, pp. 335-338, 1985.
6. C. Blundo, A. D. Santis, A. Herzberg, S. Kutten, U. Vaccard and M. Yung, "Perfectly-secure key distribution for dynamic conference", *In Advances in Cryptology (CRYPTO'92)*, Springer-Verlag, Berlin, pp. 471–486. 1993.
7. G. Hanaoka, T. Nishioka, Y. Zheng, and H. Imai, "A hierarchical non-interactive key-sharing scheme with low memory size and high resistance against collusion attacks", *The Computer Journal*, 45(3):293-303, 2002.
8. M. Ramkumar, N. Memon, and R. Simha, "A hierarchical key predistribution scheme", *Electro/Information Technology Conference (EIT'05)*, 2005.
9. L. Eschenauer and V. D. Gligor, "A key-management scheme for distributed sensor networks", *The 9th ACM Conference on Computer and Communications Security (CCS'02)*, pp. 41-47, Nov. 2002.

10. D. Huang, and D. Medhi, "A secure group key management scheme for hierarchical mobile ad hoc networks", *Ad Hoc Networks, Elsevier*, 6:560-577, 2007.
11. B. Matt, "Toward hierarchical identity-based cryptography for tactical networks", *Military Communications Conference (MILCOM'04)*, pp. 727-735, 2004.
12. D. Boneh, and M. Franklin, "Identity-based encryption from the Weil Pairing", *In Advances in Cryptology (CRYPTO'01)*, LNCS 2139, pp. 213-229, 2001.
13. T. Horwitz, and Ben Lynn, "Towards hierarchical identity-based encryption", *Annual International Conference on the Theory and Applications of Cryptographic Techniques (Eurocrypt'02)*, LNCS 2332, pp. 466-481, 2002.
14. C. Gentry, and A. Silverberg, "Hierarchical ID-based cryptography", *Annual International Conference on the Theory and Application of Cryptology & Information Security (Asiacrypt'02)*, pp. 548-566, LNCS 2501, 2002.
15. M. Albrecht, C. Gentry, S. Halevi and J. Katz, "Attacking Cryptographic Schemes Based on "Perturbation Polynomials"", ACM CCS'09, Chicago, USA, April, 2009.
16. `http://www.shamus.ie/`

Appendix 1: Key Establishment in the Presence of Addition Carries

Whether the most significant $c-r$ bits are changed depends on if a carry is generated in the addition of the least significant r bits. For example, if the result of the PP is $(\boxed{0101})_2$ ($r=4$) and the output of the original polynomial is $(\boxed{10}\boxed{1000})_2$ ($c=6$), then the most significant 2 bits are not "infected"; whereas, adding $(\boxed{10}\boxed{1000})_2$ by $(\boxed{1010})_2$ will change the most significant 2 bits. Consider a pair of nodes u and v, whose perturbed polynomials output R_u and R_v respectively. If there is a carry generated in either R_u or R_v, then the most significant $c-r$ bits of R_u and R_v will be different. However, the difference is predicable. In particular, one of R_u, R_u+2^r and R_u-2^r will have the most significant $c-r$ bits identical with those of R_v. For example, suppose the result obtained by node v is $R_v=(\boxed{10}\boxed{1000})_2+(\boxed{0101})_2=(\boxed{10}\boxed{1101})_2$ and $R_u=(\boxed{10}\boxed{1000})_2+(\boxed{1010})_2=\boxed{11}\boxed{0010})_2$; then the most significant 2 bits of $R_u-2^r=\boxed{10}\boxed{0010})_2$ are the same as those of R_v.

Zhang et al. [2] suggests a method to remove the jitters (if any) from the most significant $c-r$ bits and ensure the correctness of key establishment. In particular, they let the first message that v sends to u piggyback a hash value that is computed using all the $(c-r)$-bit key segments (recall that the final shared key is formed by concatenating these key segments), based on which node u can determine which key v is using. By using a secure hash function (such as SHA-256) and adding some random numbers into the hash computation, this hash value does not reveal any information of the shared key. For more details, we refer to [2].

Appendix 2: Proof of Theorem 1

Theorem 1. *In the advanced NHKP scheme with parameters λ, J and $\{t_k\}_{1\le k\le L}$, the attacker cannot learn any secrets of non-compromised nodes no matter how many nodes at the lowest Q levels he can compromise, as long as $\lambda \ge \prod_{k=J+1}^{L}(t_k+1)$.*

Proof. In this proof, we only consider the case where all compromised nodes are at the same level. For the scenario where compromised nodes are scattered at multiple levels, we consider a stronger case where any compromised node at level $k<l$ is replaced with its ancestor node at level l, where l is the highest level at which any compromised nodes are located (note that an upper-level node holds more secrets than a lower-level node). We assume the compromised nodes are $id_{\langle l,u_1\rangle}, ..., id_{\langle l,u_n\rangle}$ residing at level $l\in[L-Q+1,L]$, where n can be arbitrarily large.

Case 1: Breaking polynomials at higher levels. From level $L-Q+1$ down to level l, each level introduces random PPs, and the higher level the target polynomials are located the more random PPs are accumulated at the compromised nodes. Hence, it is easier for the attacker to break the polynomials (of some node) at level $l-1$ than those at higher levels. For simplicity, we only consider this worst case of breaking polynomials at level $l-1$. On the other hand, to break the polynomials of any node $id_{\langle l-1,v\rangle}$, the best strategy for the attacker is to compromise $id_{l-1,v}$'s child nodes, whose polynomial

shares are derived directly from $id_{\langle l-1,v\rangle}$'s polynomials and thus carry more information about the target polynomials. Hence, we assume that all the compromised nodes are $id_{\langle l-1,v\rangle}$'s children. We express the target polynomial of $id_{\langle l-1,v\rangle}$ as follows.

$$G_{\langle l-1,v\rangle}(x_l,...,x_L;y_1,...,y_L) = \sum_{i_l=0}^{|\mathcal{I}_{l,\langle l-1,v\rangle}|} \cdots \sum_{j_1=0}^{|\mathcal{I}_1|} \cdots \sum_{j_J=0}^{|\mathcal{I}_j|} \sum_{j_{J+1}=0}^{t_{J+1}} \cdots$$

$$\cdots \sum_{j_L=0}^{t_L} B_{\langle l-1,v\rangle,i_l,...,i_L;j_1,...,j_L} x_l^{i_l} \cdots x_L^{i_L} y_1^{j_1} \cdots y_L^{j_L}$$

From each compromised node $id_{\langle l,u_\eta\rangle}$, $1 \le \eta \le n$, the attacker learns $G_{\langle l,u_\eta\rangle}(x_{l+1}, ..., x_L; y_1, ..., y_L)$ which is formed by

$$G_{\langle l,u_\eta\rangle}(x_{l+1},...,x_L;y_1,...,y_L) = G_{\langle l-1,v\rangle}(id_{\langle l,u_\eta\rangle},x_{l+1},...,x_L;y_1,...,y_L)$$
$$+\sum_{\omega=1}^{\lambda}\prod_{\tau=l+1}^{L}\alpha^{(\tau)}_{\omega,\langle l,u_\eta\rangle}(x_\tau)\prod_{\sigma=1}^{J}\beta^{(\sigma)}_{\omega,\langle l,u_\eta\rangle}(y_\sigma)\gamma_\omega(y_{J+1},...,y_L)$$

Combining the obtained polynomials, the attacker can form the following linear equation system.

$$B_{\langle l,u_\eta\rangle,i_{l+1},...,i_L;j_1,...,j_L} = \sum_{i_l=0}^{|\mathcal{I}_{l,\langle l-1,v\rangle}|} B_{\langle l-1,v\rangle,i_l,...,i_L;j_1,...,j_L} id^{i_l}_{\langle l,u_\eta\rangle}$$
$$+\sum_{\omega=1}^{\lambda}\prod_{\tau=l+1}^{L}C^{(\tau)}_{\omega,\langle l,u_\eta\rangle,i_\tau}\prod_{\sigma=1}^{J}D^{(\sigma)}_{\omega,\langle l,u_\eta\rangle,j_\sigma}E_{\omega,j_{J+1},...,j_L}$$
$$0 \le i_\tau \le |\mathcal{I}_{\tau,\langle l,u_\eta\rangle}|, \quad \tau \in [l+1, L]$$
$$0 \le j_\sigma \le |\mathcal{I}_\sigma|, \quad \sigma \in [1, J]$$
$$0 \le j_\sigma \le t_\sigma, \quad \sigma \in [J+1, L]$$

where,

- each $B_{\langle l,u_\eta\rangle,i_{l+1},...,i_L;j_1,...,j_L}$ is known.
- each $B_{\langle l-1,v\rangle,i_l,...,i_L;j_1,...,j_L}$ is unknown.
- $\alpha^{(\tau)}_{\omega,\langle l,u_\eta\rangle}(x_\tau) = \sum_{i_\tau=0}^{|\mathcal{I}_{\tau,\langle l,u_\eta\rangle}|} C^{(\tau)}_{\omega,\langle l,u_\eta\rangle,i_\tau} x_\tau^{i_\tau}$, and each $C^{(\tau)}_{\omega,\langle l,u_\eta\rangle,i_\tau}$ is unknown.
- $\beta^{(\sigma)}_{\omega,\langle l,u_\eta\rangle}(y_\sigma) = \sum_{j_\sigma}^{|\mathcal{I}_\sigma|} D^{(\sigma)}_{\omega,\langle l,u_\eta\rangle,j_\sigma} y_\sigma^{j_\sigma}$, and each $D^{(\sigma)}_{\omega,\langle l,u_\eta\rangle,j_\sigma}$ is unknown.
- $\gamma_\omega(y_{J+1},...,y_L) = \sum_{j_{J+1}=0}^{t_{J+1}} \cdots \sum_{j_L=0}^{t_L} E_{\omega,j_{J+1},...,j_L} y_{J+1}^{j_{J+1}} \cdots y_L^{j_L}$, and each $E_{\omega,j_{J+1},...,j_L}$ is known.

This linear system comprises $n \cdot \prod_{\tau=l+1}^{L}(|\mathcal{I}_{\tau,\langle l,u_\eta\rangle}|+1)\prod_{\sigma=1}^{J}(|\mathcal{I}_\sigma|+1)\prod_{k=J+1}^{L}(t_k+1)$ equations, while it involves $\prod_{\tau=l}^{L}(|\mathcal{I}_{\tau,\langle l-1,v\rangle}|+1)\prod_{\sigma=1}^{J}(|\mathcal{I}_\sigma|+1)\prod_{k=J+1}^{L}(t_k+1)+\lambda\cdot n\cdot\prod_{\tau=l+1}^{L}(|\mathcal{I}_{\tau,\langle l,u_\eta\rangle}|+1)\prod_{\sigma=1}^{J}(|\mathcal{I}_\sigma|+1)$ unknowns. When $\lambda \ge \prod_{k=J+1}^{L}(t_k+1)$, this linear system is unsolvable since there are more variables than equations. Hence, the attacker cannot obtain any information about the target polynomial.

Case 2: Breaking polynomials at the same level. We consider the attacker attempts to break the polynomials of some node $id_{\langle l,v\rangle}$ at level l. Similarly, the best strategy for the attacker is to compromise those nodes that share the same parent node with $id_{\langle l,v\rangle}$.

We start with a special case of $l = L - Q + 1$, in which the parent node (say $id_{\langle L-Q,p\rangle}$ of these compromised nodes has the original shares of the master polynomials. The target polynomial in this case is

$$G_{\langle L-Q,p\rangle}(id_{\langle L-Q+1,v\rangle}, x_{L-Q+2}, ..., x_L; y_1, ..., y_L)$$
$$= \sum_{i_{L-Q+2}=0}^{t_{L-Q+2}} \cdots \sum_{i_L=0}^{t_L} \sum_{j_1=0}^{t_1} \cdots \sum_{j_L=0}^{t_L} B'_{\langle L-Q+1,v\rangle, i_{L-Q+2},...,i_L;j_1,...,j_L} x_{L-Q+2}^{i_{L-Q+2}} \cdots x_L^{i_L} y_1^{j_1} \cdots y_L^{j_L}$$

For each exposed polynomial $G_{\langle L-Q+1,u_\eta\rangle}(x_{L-Q+2}, ..., x_L; y_1, ..., y_L)$, $1 \le \eta \le n$, the attacker evaluates it at $y_{L-Q+1} = id_{\langle L-Q+1,v\rangle}$, and combines the results to form a linear system as below.

$$\sum_{j_{L-Q+1}=0}^{t_{L-Q+1}} B_{\langle L-Q+1,u_\eta\rangle, i_{L-Q+2},...,i_L;j_1,...,j_L} id_{\langle L-Q+1,v\rangle}^{j_{L-Q+1}}$$
$$= \sum_{j_{L-Q+1}=0}^{t_{L-Q+1}} B'_{\langle L-Q+1,v\rangle, i_{L-Q+2},...,i_L;j_1,...,j_L} id_{\langle L-Q+1,u_\eta\rangle}^{j_{L-Q+1}}$$
$$+ \sum_{\omega=1}^{\lambda} \prod_{\tau=L-Q+2}^{L} C^{(\tau)}_{\omega,\langle L-Q+1,u_\eta\rangle,i_\tau} \prod_{\sigma=1}^{J} D^{(\sigma)}_{\omega,\langle L-Q+1,u_\eta\rangle,j_\sigma} \sum_{j_{L-Q+1}=0}^{t_{L-Q+1}} E_{\omega,j_{J+1},...,j_L} id_{\langle L-Q+1,v\rangle}^{j_{L-Q+1}}$$
$$0 \le i_\tau \le |\mathcal{I}_{\tau,\langle l,u_\eta\rangle}|, \quad \tau \in [L-Q+2, L]$$
$$0 \le j_\sigma \le |\mathcal{I}_\sigma|, \quad \sigma \in [1, J]$$
$$0 \le j_\sigma \le t_\sigma, \quad \sigma \in [J+1, L-Q] \cup [L-Q+2, L]$$

where,

- $B_{\langle L-Q+1,u_\eta\rangle, i_{L-Q+2},...,i_L;j_1,...,j_L}$ and $E_{\omega,j_{J+1},...,j_L}$ are known.
- $B'_{\langle L-Q+1,v\rangle, i_{L-Q+2},...,i_L;j_1,...,j_L}$, $C^{(\tau)}_{\omega,\langle l,u_\eta\rangle,i_\tau}$, and $D^{(\sigma)}_{\omega,\langle l,u_\eta\rangle,j_\sigma}$ are unknown.

There are $n \cdot \prod_{\tau=L-Q+2}^{L}(|\mathcal{I}_{\tau,\langle L-Q+1,u_\eta\rangle}|+1) \prod_{\sigma=1}^{J}(|\mathcal{I}_\sigma|+1) \prod_{k=J+1,k\neq L-Q+1}^{L}(t_k+1)$ equations, and $\prod_{\tau=L-Q+2}^{L}(t_\tau+1) \prod_{\sigma=1}^{L}(t_\sigma+1) + \lambda \cdot n \cdot \prod_{\tau=L-Q+2}^{L}(|\mathcal{I}_{\tau,\langle L-Q+1,u_\eta\rangle}|+1) \prod_{\sigma=1}^{J}(|\mathcal{I}_\sigma|+1)$ unknowns. The number of unknowns is larger than that of equations when $\lambda \ge \prod_{k=J+1}^{L}(t_k+1)$.

As for the case where $l < L-Q+1$, the target polynomial is no longer a symmetric polynomial since the parent of these compromised nodes holds perturbed polynomials instead of original polynomial shares. Consequently, more unknown variables are introduced into the linear systems, which makes it even harder for the attacker to break the target polynomial.

Case 3: Breaking polynomials at lower levels. We consider the worst case where the compromised nodes reside at level $L-Q+1$ and the attacker attempts to break the polynomials of some node at level $d \in [L-Q+2, L]$ (say $id_{\langle d,v\rangle}$), whose ancestor

node $id_{\langle L-Q+1,b\rangle}$ at level $L-Q+1$ shares the same parent $id_{\langle L-Q,p\rangle}$ with these compromised nodes. The target polynomial in this case is

$$
\begin{aligned}
&G_{\langle L-Q,p\rangle}(id_{\langle L-Q+1,b\rangle},...,id_{\langle d,v\rangle},x_{d+1},...,x_L;y_1,...,y_L)\\
&=\sum_{i_{d+1}=0}^{t_{d+1}}\cdots\sum_{i_L=0}^{t_L}\sum_{j_1=0}^{t_1}\cdots\sum_{j_L=0}^{t_L}B'_{\langle d,v\rangle,i_{d+1},...,i_L;j_1,...,j_L}x_{d+1}^{i_{d+1}}\cdots x_L^{i_L}y_1^{j_1}\cdots y_L^{j_L}
\end{aligned}
$$

The attacker evaluates each learnt polynomial $G_{\langle L-Q+1,u_\eta\rangle}(x_{L-Q+2},...,x_L;y_1,...,y_L)$ at $(y_{L-Q+1},...,y_d)=(id_{\langle L-Q+1,b\rangle},...,id_{\langle d,v\rangle})$, obtaining the following linear system.

$$
\begin{aligned}
&\sum_{j_{L-Q+1}=0}^{t_{L-Q+1}}\cdots\sum_{j_d=0}^{t_d}B_{\langle L-Q+1,u_\eta\rangle,i_{L-Q+2},...,i_L;j_1,...,j_L}id_{\langle L-Q+1,b\rangle}^{j_{L-Q+1}}\cdots id_{\langle d,v\rangle}^{j_d}\\
&=\sum_{j_{L-Q+1}=0}^{t_{L-Q+1}}B'_{\langle d,v\rangle,i_{d+1},...,i_L;j_1,...,j_L}id_{\langle L-Q+1,u_\eta\rangle}^{j_{L-Q+1}}\\
&+\sum_{\omega=1}^{\lambda}\prod_{\tau=L-Q+2}^{L}C^{(\tau)}_{\omega,\langle L-Q+1,u_\eta\rangle,i_\tau}\prod_{\sigma=1}^{J}D^{(\sigma)}_{\omega,\langle L-Q+1,u_\eta\rangle,j_\sigma}\sum_{j_{L-Q+1}=0}^{t_{L-Q+1}}\cdots\\
&\cdots\sum_{j_d=0}^{t_d}E_{\omega,j_{J+1},...,j_L}id_{\langle L-Q+1,b\rangle}^{j_{L-Q+1}}\cdots id_{\langle d,v\rangle}^{j_d}\\
&\quad 0\le i_\tau\le|\mathcal{I}_{\tau,\langle l,u_\eta\rangle}|,\quad \tau\in[L-Q+2,L]\\
&\quad 0\le j_\sigma\le|\mathcal{I}_\sigma|,\quad \sigma\in[1,J]\\
&\quad 0\le j_\sigma\le t_\sigma,\quad \sigma\in[J+1,L-Q]\cup[d+1,L]
\end{aligned}
$$

where,

- $B_{\langle L-Q+1,u_\eta\rangle,i_{L-Q+2},...,i_L;j_1,...,j_L}$ and $E_{\omega,j_{J+1},...,j_L}$ are known.
- $B'_{\langle d,v\rangle,i_{d+1},...,i_L;j_1,...,j_L}$, $C^{(\tau)}_{\omega,\langle l,u_\eta\rangle,i_\tau}$, and $D^{(\sigma)}_{\omega,\langle l,u_\eta\rangle,j_\sigma}$ are unknown.

This linear system is made up of $n\cdot\prod_{\tau=L-Q+2}^{L}(|\mathcal{I}_{\tau,\langle L-Q+1,u_\eta\rangle}|+1)\prod_{\sigma=1}^{J}(|\mathcal{I}_\sigma|+1)\prod_{k\in[J+1,L-Q]\cup[d+1,L]}(t_k+1)$ equations, and contains $\prod_{\tau=d+1}^{L}(t_\tau+1)\prod_{\sigma=1}^{L}(t_\sigma+1)+\lambda\cdot n\cdot\prod_{\tau=L-Q+2}^{L}(|\mathcal{I}_{\tau,\langle L-Q+1,u_\eta\rangle}|+1)\prod_{\sigma=1}^{J}(|\mathcal{I}_\sigma|+1)$ unknowns. Again, the equations are fewer than the unknowns and thus the attacker cannot infer any extra secrets. □