J. Sens. Actuator Netw. **2014**, *3*, 181-206; doi:10.3390/jsan3030181

OPEN ACCESS

Journal of Sensor and Actuator Networks
ISSN 2224-2708
www.mdpi.com/journal/jsan

Article

An Authenticated Key Agreement Scheme for Wireless Sensor Networks

Mee Loong Yang [1,*], Adnan Al-Anbuky [2] and William Liu [1]

[1] School of Computer and Mathematical Sciences, Auckland University of Technology, Auckland 1142, New Zealand; E-Mail: bobby.yang@aut.ac.nz
[2] School of Engineering, Auckland University of Technology, Auckland 1142, New Zealand; E-Mail: aalanbuk@aut.ac.nz

* Author to whom correspondence should be addressed; E-Mail: bobby.yang@aut.ac.nz; Tel.: +64-9-921-9999; Fax: +64-9-921-9944.

Received: 6 May 2014; in revised form: 17 June 2014 / Accepted: 17 June 2014 / Published: 1 July 2014

Abstract: We propose a new authenticated key agreement scheme based on Blom's scheme, but using multiple master keys and public keys in permutations to compute the private keys in each node. The computations are over a small prime field, and by storing them in a random order in the node, the private-public-master-key associations (PPMka) of the private keys are lost. If a node is captured, the PPMka of the private keys cannot be determined with certainty, making it difficult to begin to attack the scheme. We obtained analytical results to show that, using suitable keying parameters, the probability of discovering the correct PPMka can be made so small, that a very powerful adversary needs to capture the entire network of tens of thousands of nodes or expend an infeasible amount of effort to try all of the possible solutions. We verified our results using computer-simulated attacks on the scheme. The unknown PPMka enables our scheme to break free from the capture threshold of the original Blom's scheme, so that it can be used in large networks of low-resource devices, such as sensor nodes.

Keywords: key agreement; Blom; security; authentication; *ad hoc*; mobile; sensors; wireless sensor networks

1. Introduction

Wireless sensor devices are physically small electronic devices equipped with the appropriate sensors, a micro-controller, a limited amount of memory and a radio transceiver for communicating with other devices. They are designed to be inexpensive, so that they can be deployed in large numbers. A small battery provides the necessary power. They communicate using radio and messages may be relayed over several nodes to the final destination. They can be deployed for monitoring in all kinds of applications, such as building structures, seismic activities, soil condition, *etc*. Their wireless communication also makes them useful for mobile applications, such as for wild-life monitoring, vehicular networks, bodily health monitoring and in difficult to access areas. They may be installed in fixed, mobile or *ad hoc* applications.

One consequence of their open wireless communications is that an adversary can easily eavesdrop on messages and also transmit malicious messages into the network. This vulnerability may be a setback to their widespread acceptance, especially in sensitive applications. It is therefore necessary to be able to protect the communications using proven cryptographic techniques. To do this requires the communicating nodes to share secret keys.

The physical deployment environment allows the adversary to physically take control of nodes and extract secret keys from the node's memory. Due to cost, sensor nodes do not have tamper proof mechanisms. To minimise the impact of compromised nodes, the keys should be shared with as few nodes as possible, preferably between pairs only. In large *ad hoc* mobile networks, there are a large number of pairwise keys, and nodes would need a large amount of memory to store them. A better solution is to use a key agreement scheme where pairs of nodes would compute their pairwise keys after exchanging some information over the insecure channel. Such schemes, such as those by Diffie-Hellman (DH), by Rivest, Shamir and Adleman (RSA) and by El-Gamal, are already widely used in computer networks. These use public key cryptographic (PKC) algorithms involving complex mathematical operations on large integers and require substantial computational, memory and energy resources that are not readily available in sensor nodes.

Symmetric cryptographic key agreement schemes are more efficient, but they generally have limitations, such as large memory requirements, limited key sizes and scalability. This paper, an extension of our previous works in [1–3], presents a symmetric key scheme, which retains the advantages of the symmetric key scheme and also is able to overcome these limitations.

1.1. This Contribution

Blom's key agreement scheme [4,5] is fast, efficient and has mutual authentication features, making it attractive for low-resource sensor devices in *ad hoc* mobile networks. Unfortunately, as nodes can be captured and have their keys stolen, Blom's scheme can be completely broken once a certain number of nodes are compromised. Our scheme is able to break free of this limitation. The main idea is to use multiple master keys and public keys in permutations to obtain multiple private keys for each node. The computations are over a small prime field, and the private keys are stored in a random order. As a result, the private-public-master-key association (PPMka) information is lost. Without the PPMka, captured private keys are unusable for breaking the scheme. We obtained analytical results to compute the probabilities of retrieving the PPMka and showed that, with suitable keying parameters, the adversary will need to capture a very large number of nodes or expend an infeasible amount of resources to obtain the PPMka. This makes our scheme useful as the cryptographic primitive for large sensor networks.

1.2. Structure of Paper

The paper is structured as follows: In Section 2, we describe some related works using Blom's key agreement scheme. In Section 3, we describe the basic concepts and features of our scheme. In Section 4, we define our security and adversary models and analyse possible attacks on the scheme. We show that without the PPMka information, the scheme cannot be attacked. In Section 5, we analyse how the PPMka information may be discovered and compute the probabilities of successful attacks. These are compared to those obtained using computer simulated attacks on the scheme. In Section 6, we discuss the performance of the scheme in terms of memory requirements, computation times and scalability. Some keying and performance parameters are given for practical implementations. In Section 7, we discuss the strengths and limitations of the scheme, and we give our conclusion in Section 8.

Notations and Terms Used

ID	the public key ID, an integer
$\mathbf{K}$	private key, a secret $(1{\times}m)$ row vector unique to the node
$\mathbf{M}$	master key, an $(m{\times}m)$ secret symmetric matrix belonging to the trusted authority (TA)
N	the number of master keys
R	pairwise key set, the set of integers used to form the pairwise key
$\mathbf{S}$	private key set, the set of $N\eta$ private keys
$\mathbf{V}$	public key, an $(m{\times}1)$ column vector unique to the node and available to everyone
m	the size of the master key matrix
n_c	the number of captured or compromised nodes
η	the number of public keys assigned to each node
p	the prime modulus for all operations, except public keys
q	the prime modulus for public key operations only
s	the public key seed, an integer $\in [0, q-1]$

2. Related Works

Blom's scheme [5] is unconditionally secure in that, if not more than a certain number of nodes are compromised, the scheme cannot be broken, as there is simply insufficient information [6]. On the other hand, if enough nodes are compromised, the attacker would be able derive the master key and completely break the scheme. Blundo's polynomial conference key distribution scheme [7] with bivariate symmetric polynomials is equivalent to Blom's scheme. For sufficiently large pairwise keys and application in a large networks, each node would require a substantial amount of memory to store its private key.

A number of attempts have been made on either Blom's or Blundo's scheme to enhance node capture resilience by using multiple key spaces, so that the attacker has less chance of obtaining all of the nodes in the same key space. For example, the scheme in [8] used multiple key spaces and incorporated a probabilistic method similar to Eschenauer and Gilgor's [9], such that pairs of nodes must discover their shared key space to compute their pairwise key. To achieve full connectivity, if a pair of nodes do not share a key space, secured intermediary nodes are used to establish their pairwise key. An equivalent scheme in [10] was independently discovered at the same time. The pairwise key sizes were 64 bits. In these schemes, resilience against node capture is enhanced since the probability of capturing enough nodes in the same key space is reduced. A similar idea using multiple key spaces was proposed in [11], but in this case, the nodes are connected in a complete bipartite graph. In [12], only the cluster heads implemented Blom's scheme, thus allowing the overall network size to be larger than the number of cluster heads, which must be within the capture threshold to be secure.

A different idea in [13] based on the bivariate polynomial with multiple-key spaces added random perturbations to the polynomials, so that captured nodes cannot be used to break the scheme. They were able to compute 80-bit pairwise keys in about 0.13 s, requiring about 15 KB ROM and 0.33 KB RAM. In a similar approach, the work in [14] used random perturbations, which are hashed with the pairwise key obtained using Blom's scheme. After establishing the pairwise key, the private keys are erased to prevent the adversary from obtaining them. A newly deployed node would not be able to implement Blom's scheme to connect to an already secured node. Instead, it is deployed with an ID and a secret key shared with the base station. To authenticate a new node, the secured node would contact the BSto obtain the secret key shared with the node. Another implementation in [15] also uses random perturbations. Here, small random perturbations are added to the private keys to break the direct connection to the master key, making it more difficult to break. The pairwise keys computed are identical after the effect of the small random perturbations are removed.

A scheme in which the private vectors of the nodes can be updated was proposed in [16]. In this scheme, the modified Blom's scheme used hashed values of the prime seeds, and similarly, nodes have private vectors, which are hashes of the original private vectors. Their scheme limits the node capture to less than the capture threshold.

3. The BYka Scheme

3.1. Blom's Scheme

Blom's scheme [5], on which our scheme is based, is briefly described as follows. An entity, called the trusted authority (TA) generates for itself a master key $\mathbf{M}$, which is a random $(m \times m)$ symmetric matrix over the prime field $\mathbb{F}_p$. It assigns a node a public key $\mathbf{V}$, which is an $(m \times 1)$ column vector in $\mathbb{F}_p$. The TA computes and stores in the node its private key $\mathbf{K} = \mathbf{V}^T \cdot \mathbf{M}$ (mod p). To obtain their pairwise key, a pair of nodes, e.g., nodes A and B exchange their public keys and compute (mod p),

$$\begin{aligned} \text{Node A:} \quad K_{AB} &= \mathbf{K}_A \cdot \mathbf{V}_B = (\mathbf{V}_A^T \cdot \mathbf{M}) \cdot \mathbf{V}_B \\ \text{Node B:} \quad K_{BA} &= \mathbf{K}_B \cdot \mathbf{V}_A = (\mathbf{V}_B^T \cdot \mathbf{M}) \cdot \mathbf{V}_A \end{aligned}$$

The quantity $(\mathbf{V}_x \cdot \mathbf{M} \cdot \mathbf{V}_y)$ is a (1×1) scalar, and transposing $K_{BA} = \mathbf{V}_A^T \mathbf{M}^T \cdot \mathbf{V}_B$. Since $\mathbf{M}$ is symmetric, the two keys K_{AB} and K_{BA} are identical.

3.2. The BYka Scheme

Our multiple-key Blom's scheme [1,2], now called the Blom–Yang key agreement (BYka) scheme, uses the Blom's scheme as the cryptographic primitive, but with multiple master keys and public keys used in permutations in a single key space.

3.3. Setup

The TA selects the keying parameters: the number of secret master keys N, the size m, the number of public keys in each node η, the prime modulus for key computations p and the prime modulus for public key computations q. For example, $N = 7, m = 16, \eta = 6, p = 31$ and $q = 65521$, to obtain pairwise keys of 128 bits for a network of about 10,000 nodes.

The TA generates N master keys $\mathbf{M}_1, \mathbf{M}_2, \cdots, \mathbf{M}_N$, over the prime field $\mathbb{F}_p$. These are $(m \times m)$ symmetric matrices.

3.3.1. Public Key Set and IDs

The TA assigns to each node η unique public keys, called the public key set, each one an $(m \times 1)$ column vector of the Vandermonde matrix over the field $\mathbb{F}_q$. As the elements of a column in the Vandermonde matrix are s^{i-1} for $i = 1, \cdots, m$, where s is called the "seed", the node needs only be assigned η seeds $\{s, \cdots, s + \eta - 1\}$. The seeds are consecutive, and the smallest seed s is a multiple of η. In this way, no two nodes share a common seed. The node's public key set can be succinctly represented by the smallest seed s, which also serves as its public key *ID*, e.g., using $\eta = 6$, a node A with public key $ID_A = 240$ has public key seeds $\{240, 241, \cdots, 245\}$. Given a node's public key *ID*, anyone knowing q can generate its public key set as follows,

$$\begin{aligned} \mathbf{V}_i^T &= [1 \quad s_i \quad s_i^2 \cdots s_i^{m-1}] \pmod q \\ \text{where} \quad s_i &= ID + i - 1, \quad \text{for} \quad i = 1, \cdots, \eta \end{aligned} \tag{1}$$

When pairs of nodes exchange their public keys, they only need to transmit their IDs consisting of a few bits, e.g., 16 bits. This is an important feature, saving time and energy for radio transmissions.

3.3.2. Private Key Set, S

The TA computes the private keys for each node using all the permutations of their η public keys with its N master keys to obtain the node's "private key set" $\mathbf{S} = \{\mathbf{K}_{11}, \cdots, \mathbf{K}_{\eta N}\}$, where $\mathbf{K}_{ij}$, called the private key, is a $(1 \times m)$ row vector, computed as follows,

$$\begin{aligned} \mathbf{K}_{ij} &= \mathbf{V}_i^T \mathbf{M}_j \pmod p \\ \text{for } i &= 1, \cdots, \eta \text{ and } j = 1, \cdots, N \end{aligned} \tag{2}$$

PPMka

The private key $\mathbf{K}_{ij}$ is computed from the $i-th$ public key $\mathbf{V}_i$ and the $j-th$ master key $\mathbf{M}_j$. We call the relationship of a private key with the public key and master key used to compute it the "private-public-master-key association" (PPMka). The TA transfers the private key set to the node using a secure connection and stores them in random order. Alternatively, the private key set can be first shuffled before transferring to the node. If a node is compromised and the private keys obtained, the adversary cannot tell from the storage location which public key and master key was used to compute it.

3.3.3. Key Aliasing

The number of public key seeds must be large enough to accommodate the network size. To do this, the public key operations are over a large field $\mathbb{F}q$, for example, $q = 65521$ catering to about 10,000 nodes, but it can be much larger. As the private key operations are over a small field $\mathbb{F}p$, it is possible for multiple public keys to map to the same private key, a phenomenon we call "key aliasing", described as follows. Consider the private key $\mathbf{K}_k = \mathbf{V}_{s_n}^T\mathbf{M}_y$, where s_n is the seed for $\mathbf{V}_n$. Denoting the elements of $\mathbf{M}_y$ as $M_{y_{ij}}$ and using Equation (1), the $u-th$ element of $\mathbf{K}_k$ is,

$$\begin{aligned} \mathbf{K}_{k_u} &= \sum_{i=1}^{m} s_n^{i-1} M_{y_{iu}} \pmod{p} \\ &= M_{y_{1u}} + s_n^1 M_{y_{2u}} + \cdots + s_n^{m-1} M_{y_{mu}} \end{aligned} \tag{3}$$

For two nodes, say A and B, if any of their public key seeds are congruent, e.g., $s_A \equiv s_B \pmod{p}$, and for all $i = 0, \cdots, m-1$, the elements s_A^{i-1} and s_B^{i-1} are smaller than q (the elements in the public key vectors do not "wrap round" q), then we have $s_A^{i-1} \equiv s_B^{i-1} \pmod{p}$ for all i. As a result, their private keys associated with the same master key are identical since,

$$\begin{aligned} \mathbf{K}_{A_u} &= M_{y_{1u}} + s_A^1 M_{y_{2u}} + \cdots + s_A^{m-1} M_{y_{mu}} \pmod{p} \\ \text{and } \mathbf{K}_{B_u} &= M_{y_{1u}} + s_B^1 M_{y_{2u}} + \cdots + s_B^{m-1} M_{y_{mu}} \pmod{p} = \mathbf{K}_{A_u} \end{aligned}$$

To prevent key aliasing, a seed s_n is chosen, such that at least one vector element exceeds q, and the residue $r \pmod{q}$ is different from $s_n \pmod{p}$ and is not zero. The requirements of a seed s_n are then,

$$\left.\begin{aligned} \text{for some } w &\leqslant m, \ s_n^{w-1} > q \\ \textit{i.e.}, \ s_n^{w-1} &\equiv r_n \pmod{q} \\ \text{and } \ r_n &\not\equiv \begin{cases} 0 \pmod{p}, & \text{and} \\ s_n \pmod{p} \end{cases} \end{aligned}\right\} \tag{4}$$

The TA installs into each node their "keying material" comprising the global keying parameters $\{m, N, \eta, p, q\}$, the node's individual public key ID and private key set **S**. All of these are static and can be stored in the ROM or flash memory.

3.4. Pairwise Key Computation

After deployment, any pair of nodes can compute their pairwise key after exchanging their IDs. For example, nodes A and B have obtained each other's IDs. Each node generates their counterpart's public keys using Equation (1) and, then, using all of the permutations with its own private key set, computes (mod p) the set R, called the "pairwise key set", as follows,

$$\left.\begin{array}{l} \text{Node A:}\quad R_A = \left\{\mathbf{K}_{A_{ij}}\mathbf{V}_{B_k}\right\} = \left\{(\mathbf{V}_{A_i}^T\mathbf{M}_j)\mathbf{V}_{B_k}\right\} \pmod p \\ \text{Node B:}\quad R_B = \left\{\mathbf{K}_{B_{ij}}\mathbf{V}_{A_k}\right\} = \left\{(\mathbf{V}_{B_i}^T\mathbf{M}_j)\mathbf{V}_{A_k}\right\} \pmod p \\ \quad\text{for } i,k = 1,\cdots,\eta, \text{ and } j = 1,\cdots,N \end{array}\right\} \tag{5}$$

Transposing each element in R_B, we have,

$$R_B = \left\{((\mathbf{V}_{B_i}^T\mathbf{M}_j)\mathbf{V}_{A_k})^T\right\} = \left\{(\mathbf{V}_{A_k}^T\mathbf{M}_j^T)\mathbf{V}_{B_i}\right\}$$

Since $\mathbf{M}_j$ is symmetric and i, j, k are merely independent counters, the sets R_A and R_B each contain $N\eta^2$ identical numbers $\in [0, p-1]$, though not in the same order. These numbers are used by both nodes to form their pairwise key K_{pair}.

Pairwise Key

The pairwise key can be constructed from the pairwise key set R using several methods. In one method, the number of occurrences of the integers in R are counted and used as the input to a hash function to output the pairwise key. In another method, the numbers in R are sorted and concatenated into a large key. It is also possible to increment all elements in R by one to make them all non-zero and then multiply them together (mod S_k) to obtain the pairwise key, where S_k is a large prime number of the desired key size. Once the nodes have obtained their identical pairwise key, they can use it for encrypting messages or to transport a randomly generated session key for subsequent communications.

4. Security of the BYka Scheme

4.1. Security Model

This section defines the components of the system, the adversary and its capabilities and the meaning of system breakdown.

4.1.1. System

The system comprises nodes belonging to one administrative unit under the same TA. It is assumed that TA has access to a cryptographically secure random number generator. The master keys are assumed secure and cannot be stolen. If need be, they can be deleted after generating all of the possible public and private key sets. The nodes have access to secure cryptographic algorithms, such as AESencryption and hash algorithms.

4.1.2. Adversary

The adversary is a very powerful agent with powerful computing resources. It is able to move about freely in the deployment space to monitor transmissions, replay messages and insert its own fabricated messages. It is also able to physically capture nodes and extract all the keying material, including the public key IDs, the private key sets **S** and the keying parameters from ROM and RAM memory.

4.1.3. System Breakdown

The scheme is considered broken if the adversary is able to, by monitoring transmissions or using the keys from captured nodes,

(1) obtain the pairwise keys of any other pairs of uncompromised nodes, or
(2) fabricate new valid public and private keys, or
(3) compute the master keys of the TA.

Identity theft attacks, where the adversary clones a node by fabricating a new node with the identical keys from the captured node, though a very serious threat, is beyond the scope of this paper.

4.1.4. Vulnerabilities

The vulnerabilities of the BYka scheme are broken down and analysed in the three main parts:

(1) Strength of the keys against brute force attacks
(2) Security of the underlying Blom's scheme, as it applies to the BYka scheme
(3) Resilience against node capture

4.2. Strength of Keys against Brute Force Attacks

The master keys and private keys are random and large. For example, with values of $N = 7$, $m = 16$, $\eta = 6$ and $p = 31$, there are 2^{634} possible master keys and 2^{208} private keys. A brute force attack is not feasible.

Pairwise Key

One limitation in the original Blom's scheme is that the pairwise key is only the same size as the data size of the master key elements. In our BYka scheme, the pairwise key size can be up to $p^{N\eta^2}$ integers $\in [0, p-1]$.

The BYka scheme can be viewed as a mechanism for two nodes to derive a common secret pairwise key set R consisting of $N\eta^2$ integers from which to construct their pairwise key. The number of possible keys, the "key space", is limited by the number of possible combinations of the $N\eta^2$ integers. To determine the key space size, we consider the following partitioning problem.

Given a row of $N\eta^2$ items, we wish to partition them into p groups. This is illustrated in Figure 1 for the case of partitioning eight items into four groups. To create the partitions, we first insert $(p-1)$ items into the row, so that there are now $(N\eta^2 + p - 1)$ items. If any $(p-1)$ items are now removed, $(p-1)$

gaps would be created, separating the remaining items into p groups as desired. Let group g_0 contain the integer zero, g_1 contain one, g_2 contain two, *etc*. The total number of integers is always $N\eta^2$. The number of ways to remove $(p-1)$ items from $(N\eta^2+p-1)$ gives the key space size as follows,

$$\begin{aligned} K_{sp} &= \binom{N\eta^2+p-1}{p-1} \\ &= \log_2\left[\binom{N\eta^2+p-1}{p-1}\right] \quad \text{bits} \end{aligned} \tag{6}$$

Table 1 shows the key space sizes for various keying parameters in bits. It can be seen that the key spaces of 64 bits and larger are possible.

Figure 1. Partitioning eight items into four groups.

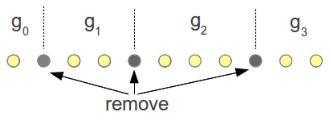


Table 1. Key space in bits.

η	N	Values of p				
		13	17	19	23	31
6	6	🟨 64	🟥 80	🟥 88	⬛ 102	⬛ 127
	7	🟨 67	🟥 84	🟥 92	⬛ 106	🟦 134
	8	🟨 69	🟥 87	🟥 95	⬛ 111	🟦 139
7	6	🟨 69	🟥 87	🟥 95	⬛ 111	🟦 140
	7	🟨 72	🟥 91	⬛ 99	⬛ 116	🟦 146
	8	🟨 74	🟥 94	⬛ 103	⬛ 120	🟦 152
8	6	🟨 74	🟥 93	⬛ 102	⬛ 119	🟦 151
	7	🟨 77	⬛ 97	⬛ 106	⬛ 124	🟦 157
	8	🟨 79	⬛ 100	⬛ 109	🟦 128	🟦 163

Legend: Key space 🟨 64 bits, 🟥 80 bits, ⬛ 96 bits, 🟦 128 bits

4.3. Security of the Underlying Blom's Scheme

Blom's scheme is vulnerable to the Sybil attack, and the master key can be derived if enough nodes are captured. We now examine how this can be done and then analyse how our BYka scheme would fare.

4.3.1. Sybil Attacks

In this attack, the attacker would fabricate new public and private keys by combining captured keys and use them to masquerade legitimate nodes. Consider that n nodes and their public and private keys have been obtained. The attacker can fabricate a new public key $\mathbf{V}_X$ by linear combination of captured public keys as follow:

$$\mathbf{V}_X = \alpha_1\mathbf{V}_1 + \cdots + \alpha_n\mathbf{V}_n \pmod p \tag{7}$$

The corresponding private key $\mathbf{K}_X$ would also be a similar linear combination of the captured private keys,

$$\begin{aligned}\mathbf{K}_X &= \mathbf{V}_X^T\mathbf{M} = (\alpha_1\mathbf{V}_1^T + \cdots + \alpha_n\mathbf{V}_n^T)\mathbf{M} \\ &= \alpha_1\mathbf{V}_1^T\mathbf{M} + \cdots + \alpha_n\mathbf{V}_n^T\mathbf{M} \\ &= \alpha_1\mathbf{K}_1 + \cdots + \alpha_n\mathbf{K}_n \pmod p \end{aligned} \tag{8}$$

By choosing various combinations of $\alpha_1, \cdots, \alpha_n$, the attacker is able to fabricate any public key and the corresponding private key at will.

Mitigation

To defeat this attack, three conditions must be met:

(1) the public keys must conform to a prescribed structure,
(2) the public keys are linearly independent, and
(3) no more than $(m-1)$ nodes are captured, *i.e.*, $n < m$.

The first condition ensures that a key formed from arbitrary linear combinations of captured keys would not be accepted. If all of the public keys are of a prescribed structure, such as those of the column of the Vandermonde matrix, arbitrary public keys would simply be discarded.

If all of the public key vectors are linearly independent and $n < m$, then by definition, the solution of Equation (7) is trivial, *i.e.*, $\alpha_1, \cdots, \alpha_n = 0$. On the other hand, if $n \geqslant m$, then, as there are at most m linearly independent $(m \times 1)$ vectors, any m public keys can be combined to obtain a non-trivial solution in Equation (7) and obtain the corresponding private key using Equation (8).

4.3.2. Attacking the Master Key

Consider that m nodes have been captured and all of the public keys are linearly independent. The attacker would be able to construct a system of m linear equations from each private key using the relationship, $\mathbf{K}_i = \mathbf{V}_i^T\mathbf{M}$, which, after transposing, can be written as $\mathbf{M}^T\mathbf{V}_i = \mathbf{K}_i^T$ where $\mathbf{M}^T = \mathbf{M}$. Combining these from the m captured nodes, we have,

$$\begin{aligned}\mathbf{M}\begin{bmatrix}\mathbf{V}_1 & \mathbf{V}_2 & \cdots & \mathbf{V}_m\end{bmatrix} &= \begin{bmatrix}\mathbf{K}_1^T & \mathbf{K}_2^T & \cdots & \mathbf{K}_m^T\end{bmatrix} \\ \textit{i.e.},\ \mathbf{MV} &= \mathbf{K} \\ \text{If } \mathbf{V} \text{ is invertible, then } \mathbf{M} &= \mathbf{KV}^{-1}\end{aligned} \tag{9}$$

From linear algebra, the matrix $(m \times m)$ $\mathbf{V}$ is invertible if, and only if, the determinant $|\mathbf{V}| \neq 0$. Since the column vectors in $\mathbf{V}$ are linearly independent (for example, the Vandermonde matrix), then

V is non-singular with a non-zero determinant. The elements of the master key can be obtained, for example using the Gaussian elimination method.

Capture Threshold λ

The above shows the main limitation of Blom's scheme. If the number of captured nodes reaches m, called the "capture threshold", the entire scheme can be broken. Bloms's scheme is said to be $(m-1)$ secure if the number of nodes deployed is <m. Then, even if all of the nodes are captured, there is no determinate solution for **M**, and it is unconditionally secure.

To implement a Blom's scheme that is $(m-1)$ secure, a large m would be required and, together with the requirement for large pairwise key sizes, the nodes would require a large memory to store the private keys. This places a limit on Blom's scheme.

4.3.3. Immunity to MITM Attacks and Mutual Authentication

In the man-in-the-middle (MITM) attack, an adversary node E interposes itself between two nodes A and B. It posses as A to B and, similarly, as B to A. If this is successful, it acts as an intermediary between A and B, reading and modifying messages before forwarding them. In Blom's scheme, if the attacker E forwards its own ID_E to node A to impersonate node B, node A would compute the pairwise key K_{AE}. Node E cannot compute K_{EA}, as it does not have the private key for ID_E. If E forwards ID_B to node A and ID_A to node B, both nodes A and B can compute their pairwise key K_{AB}, which cannot be obtained by node E. Messages encrypted between nodes A and B cannot be read by E. Blom's scheme is immune to MITM attacks, as both nodes must use keying material from the TA to compute their pairwise key. In this way, the scheme is mutually authenticating.

4.3.4. Implications for the BYka Scheme

The BYka scheme inherits the mutual authentication and immunity to the MITM attacks as in Blom's scheme. In addition, it would also appear to inherit the capture threshold limitation. In fact, the BYka scheme's capture threshold is lower at $\lambda = \lceil \frac{m}{\eta} \rceil$, since each node carries ηN private keys. However, the capture threshold is not applicable, since, to use the captured private keys, the attacker needs to associate each private key with the public key and master key used to compute it, *i.e.*, discover the PPMka. In the original Blom's scheme with only one key, the PPMka is obvious.

4.4. Resilience against Sybil Attacks

The Sybil attack cannot be mounted as in Blom's scheme. Consider that m private keys $\mathbf{K}_{C_1 M_1}, \cdots, \mathbf{K}_{C_m M_1}$ and the corresponding public keys $\mathbf{V}_{C_1}, \cdots, \mathbf{V}_{C_m}$ associated with one of the master keys $\mathbf{M}_1$ have been obtained. The attacker chooses a public key ID_X seed s_{X_1} and constructs the public key $\mathbf{V}_{X_1}$ as a Vandermonde column vector, such that,

$$\mathbf{V}_{X_1} \;=\; \alpha_{1_1}\mathbf{V}_{C_1} + \cdots + \alpha_{1_m}\mathbf{V}_{C_m} \pmod{q} \tag{10}$$

The coefficients $\alpha_{1_1}, \cdots, \alpha_{1_m}$ can be obtained and used to construct the private key associated with $\mathbf{M}_1$ and $\mathbf{V}_{X_1}$,

$$\begin{aligned} \mathbf{K}_{X_{1_{M1}}} &= \mathbf{V}_{X_1}^T \mathbf{M}_1 = (\alpha_{1_1} \mathbf{V}_{C_1}^T + \cdots + \alpha_{1_m} \mathbf{V}_{C_m}^T) \mathbf{M}_1 \\ &= \alpha_{1_1} \mathbf{K}_{C_{1M_1}} + \cdots + \alpha_{1_m} \mathbf{K}_{C_{mM_1}} \pmod p \end{aligned} \tag{11}$$

Here, $\mathbf{K}_{C_{1M_1}}$ is the private key associated with the master key $\mathbf{M}_1$ and public key $\mathbf{V}_{C_1}$. The difficulty is identifying which of the $N\eta$ private keys in the node is this particular one, and similarly for $\mathbf{K}_{C_{2M1}}$, *etc.* Each private key is a row vector with elements, which are sums and products of random numbers, and is indistinguishable from the others. The order of storage in memory is also random and unrelated to the order in which they were computed. An adversary cannot derive any information about the private-public-master-key associations (PPMka) from examining the keys or its storage location.

If the PPMka information is not available, the adversary will need to try all of the possible PPMka as follows. From each node, there are $\frac{(N\eta)!}{(N\eta-\eta)!}$ ways to select the η private keys associated with $\mathbf{M}_1$ and the public keys $\mathbf{V}_{C_1}, \cdots, \mathbf{V}_{C_\eta}$. To select all of the private keys in the $\frac{m}{\eta}$ captured nodes associated with $\mathbf{M}_1$ and the corresponding public keys for use in Equation (11), we have Φ_1 possible ways, given by,

$$\Phi_1 = \left[\frac{(N\eta)!}{(N\eta - \eta)!} \right]^{\frac{m}{\eta}}$$

To complete the Sybil attack, all of the public and private keys are similarly constructed for each of the master keys and used together. The total number of possible solutions for all of the PPMka's is,

$$\Phi = \sum_{i=0}^{N-1} \left[\frac{(N\eta - i\eta)!}{(N\eta - i\eta - \eta)!} \right]^{\frac{m}{\eta}} \tag{12}$$

Table 2. Number of Solutions Φ.

η	N	Master Key Size m			
		12	16	24	32
6	6	2.16×10^{18}	2.84×10^{27}	3.90×10^{36}	7.61×10^{54}
	7	1.64×10^{19}	5.67×10^{28}	2.07×10^{38}	2.91×10^{57}
	8	9.45×10^{19}	7.46×10^{28}	6.30×10^{39}	4.79×10^{59}
7	6	1.97×10^{22}	2.55×10^{33}	3.43×10^{44}	4.65×10^{55}
	7	2.07×10^{23}	8.37×10^{34}	3.55×10^{46}	1.53×10^{58}
	8	1.57×10^{24}	1.68×10^{36}	1.90×10^{48}	2.20×10^{60}
8	6	2.41×10^{26}	2.41×10^{26}	3.55×10^{39}	5.37×10^{52}
	7	3.52×10^{27}	3.52×10^{27}	1.91×10^{41}	1.08×10^{55}
	8	3.54×10^{28}	3.54×10^{28}	5.88×10^{42}	1.03×10^{57}

As an example, with $N = 7, \eta = 6$ and $m = 16$, $\Phi = 5.67 \times 10^{28}$ possible solutions. Hence, without knowing the PPMka, the Sybil attack requires an unfeasibly large number of trials. Table 2 gives the possible number of solutions for various keying parameters.

4.5. Brute Force Attack on the Master Keys

Similarly, to solve for all of the master keys using the captured private keys without knowing the PPMka information, the number of possible sets of $m \times m$ linear equations is also given in Equation (12). Each attempt involves constructing the $(m \times m)$ system of linear equations, solving them using, say, the Gaussian elimination method and testing each solution to see if it can successfully compute a captured node's private key using one of its public keys. The possible number of solutions is also given in Table 2.

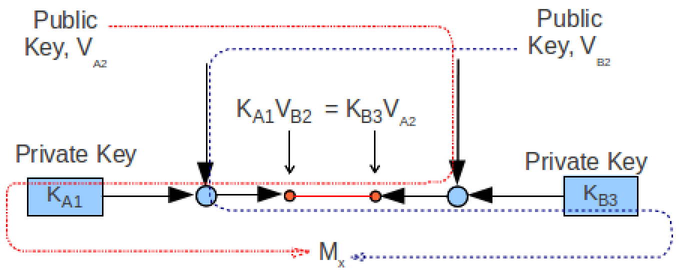
Hence, due to the unknown PPMka in the BYka scheme, there is only a probabilistic chance of breaking the scheme, even if sufficient captured keys are available. With suitable keying parameters, the chance can be made so small, that the scheme cannot be feasibly broken. However, the scheme can be broken if the PPMka can be discovered. We show next how discovering the PPMka can be made very difficult by using key operations over a small prime field $\mathbb{F}_p$.

5. Attacks to Discover the PPMka

5.1. Pairing Attack

If the keys from a pair of captured nodes are used to compute their pairwise key set, the identical numbers in the key set can expose the related public and master keys. This is called the "pairing attack". For example, using nodes A and B, their pairwise key sets R_A and R_B will contain $N\eta^2$ identical numbers. This is illustrated in Figure 2 showing only one of the matching numbers in R_A and R_B. The identical numbers $\mathbf{K}_{A1}\mathbf{V}_{B2} = \mathbf{K}_{B3}\mathbf{V}_{A2}$ reveal that private keys $\mathbf{K}_{A1}$ and $\mathbf{K}_{B3}$ are both associated with the same master key, say $\mathbf{M}_x$, and also reveal the PPMka: $\mathbf{K}_{A1} = \mathbf{V}_{A2}^T\mathbf{M}_x$ and $\mathbf{K}_{B3} = \mathbf{V}_{B2}^T\mathbf{M}_x$. If all of the $N\eta^2$ numbers are unique, then it is easy to discover all of the PPMka. However, since there are $N\eta^2$ numbers in $R \in [0, p-1]$ and p is a small prime, there will be ambiguities. For example, with $p = 31$, $N = 7$ and $\eta = 6$, there are 252 numbers, each one $\in [0, 30]$.

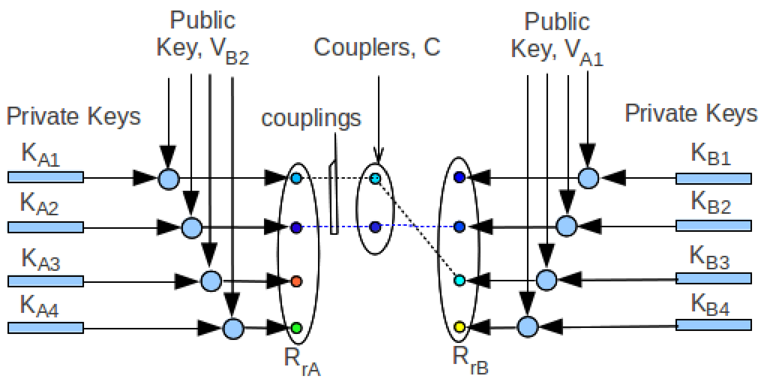
Figure 2. Pairing attack showing one of the matching numbers.



A more efficient pairing attack is to use only one of the public keys to compute the partial key sets R_r. The number of elements in the partial key set is now reduced to $N\eta$. This is illustrated in Figure 3 for the simple case where $N = 2$, $\eta = 2$. Here, as $\mathbf{K}_{A_1}\mathbf{V}_{B_2} = \mathbf{K}_{B_3}\mathbf{V}_{A_1}$, both must be associated with the same master key say, $\mathbf{M}_1$. Hence, $\mathbf{K}_{A_1} = \mathbf{V}_{A_1}^T\mathbf{M}_1$ and $\mathbf{K}_{B_2} = \mathbf{V}_{B_2}^T\mathbf{M}_1$.

If all of the numbers in the partial key sets R_r are unique, the above attack would be successful. However, if they are not all unique, we say that there are "collisions" that give rise to ambiguities, since more than one PPMka is possible for the affected private key.

Figure 3. Pairing attack having unique couplers, for the case $N = 2, \eta = 2$.



5.1.1. Couplers and Couplings

Each pairing attack, e.g., Figure 3, should produce exactly N identical numbers in sets R_{rA} and R_{rB} if all of the numbers are unique. The set C contains the distinct identical numbers called "couplers". The links connecting the couplers to the numbers in R_{rA} and R_{rB} are called "couplings". The number of couplings, denoted as N_c, is $\geq$ number of couplers.

In the ideal case where there is no collision, there would be exactly $N_c = N$ couplings on each side of C, each one linking the private key to the associated master key and public key, exposing the PPMka. In this way, by successively pairing an exposed node with other nodes, all of the PPMka can be obtained. However, if the couplers are not unique, then the associated master key is ambiguous for the affected private key.

The probability of having only unique numbers in R_r, hence exactly N couplers in a set of $N\eta$ numbers, is $P_u = \frac{p}{p}\frac{p-1}{p}\cdots\frac{p-N\eta-1}{p}$. To make this attack more difficult, P_u can be made very small by choosing a small value of p and somewhat larger values of N and η. For example, with $p = 31, N = 5$, $\eta = 6$, $P_u = 1.49 \times 10^{-11}$. For $N = 7, \eta = 6$, then $P_u = 0$, since $N\eta > p$.

5.1.2. Pairing Attack Strategies

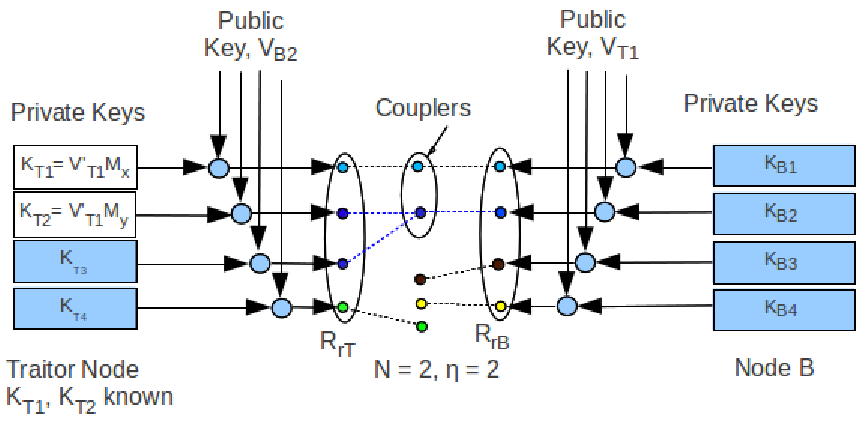
We consider two extreme approaches to discovering the PPMka information to show the difficulty and effort required. First, we consider the "unlimited capture" case where the attacker is able to pick and choose any of the nodes for pairing, and second, the "limited capture" case, where the attacker has obtained only a sufficient number of captured nodes.

5.2. Unlimited Capture

5.2.1. Traitor Node

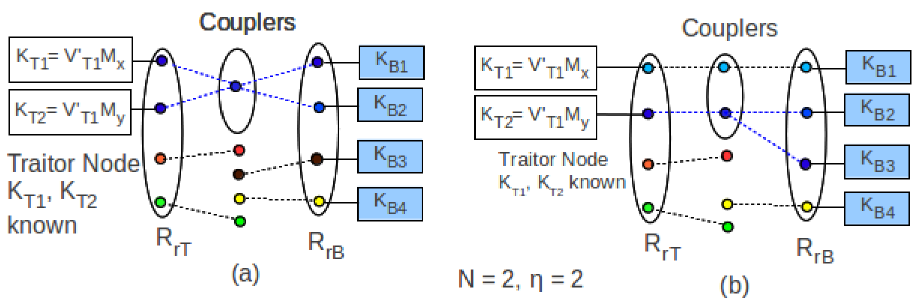
The attack would be easier if it is possible to find one node in which all of the N private keys associated with one public key, say $\mathbf{V}_1$, is known. This set of private keys can be used to reveal the PPMka of other private keys. We call this the "traitor node". For example, in Figure 4, the traitor node T is available, whose keys $\mathbf{K}_{T1}$ and $\mathbf{K}_{T2}$ are known to be associated with $\mathbf{M}_x$ and $\mathbf{M}_y$, respectively. If the node B is paired with it and if the number of couplings in R_{rB} is N, they distinctly link the connected private keys in B to the exposed private keys in T revealing the PPMka, *i.e.*, $\mathbf{K}_{B1}$ and $\mathbf{K}_{B2}$ must be associated with $\mathbf{M}_y$ and $\mathbf{M}_x$, respectively, and both associated with $\mathbf{V}_{B2}$.

Figure 4. The traitor node can be used to attack private-public-master-key associations (PPMka).



This is not so straightforward if the number of couplers in R_{rB} is $N_c \neq N$, as in Figure 5. In Figure 5a, the partial key set R_{rB} obtained using public key $\mathbf{V}_{B2}$ has less couplers than N, *i.e.*, only one coupler instead of two. While the private keys $\mathbf{K}_{B1}$ and $\mathbf{K}_{B2}$ can be associated with $\mathbf{V}_{B2}$, their associations with the master keys are ambiguous. Furthermore, in Figure 5b, R_{rB} has more than N couplers, *i.e.*, three instead of two. Now, it is not clear whether $\mathbf{K}_{B2}$ or $\mathbf{K}_{B3}$ is associated with $\mathbf{V}_{B2}$ and $\mathbf{M}_y$. Hence, when a node is paired with the traitor and has exactly N couplers, the PPMka of the connected private keys will be revealed. Finding a traitor node is thus the first step to discovering the PPMka information.

Figure 5. The traitor node cannot be used to attack the PPMka.

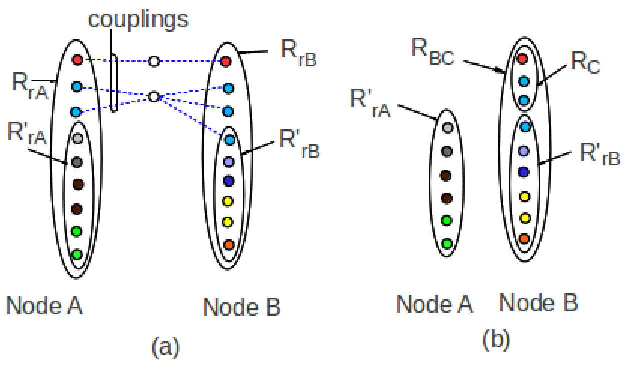


5.2.2. Probability of Finding a Traitor Node

A traitor node T is found if, in a pairing, the number of couplings it has is $N_c \leqslant N$; for example, in Figure 3, both nodes can be used as the traitor node. If $N_c > N$, there are ambiguities, since there are >1 possible associations between the N_c private keys and the N master keys.

To calculate the probability of finding a traitor node, we consider the following problem. In Figure 6a, the pairing attack produces partial key sets R_{rA} and R_{rB}. We remove the couplers from R_{rA}, to form the set, R_c, leaving the reduced partial key set R'_{rA}; see Figure 6b. A traitor node is found if the reduced set R'_{rA} is disjoint with $(R'_{rB} \cup R_c)$ or R'_{rB} is disjoint with $(R'_{rA} \cup R_c)$. Additionally, sets R'_{rA}, R'_{rB} and R_c can all be disjoint. The probability of these occurrences can be found by counting the number of arrangements for the above cases. Let N_a, N_b and N_c be the number of elements in sets R'_{rA}, R'_{rB} and R_c, respectively. Here, $N_c = N$, $N_a = N_b = N\eta - N$.

Figure 6. Finding the traitor node.



Two Disjoint Sets

Consider the case where the two sets R'_{rA} and $(R'_{rB} \cup R_c)$ are disjoint. The set R'_{rA} can have one number repeated N_a times, e.g., $\{1,1,1,\cdots,1\}$,$\{2,2,2,\cdots,2\}$, *etc.*, or two different numbers in various arrangements, e.g., $\{1,1,\cdots,1,2\}$, $\{1,1,\cdots,2,2\}$, *etc.*, or three different numbers, e.g., $\{1,1,\cdots,2,3\}$, $\{1,2,\cdots,2,3\}$, and so on. For each case, the remaining numbers can be used in the set $(R'_{rB} \cup R_c)$.

Before proceeding, first consider the number of ways $Q_{N_a r}$ of arranging N_a numbers, such that each arrangement uses all of the given r numbers. For example, in arranging four numbers using all three numbers $\{6,7,8\}$, arrangements like $\{6,6,7,8\}$ and $\{6,7,7,8\}$ would be included, but excluded those arrangements using only one or two of the numbers, such as $\{6,6,6,6\}$ and $\{6,6,7,6\}$, *etc.* Let the number of arrangements be $Q_{N_a r}$. It can be shown that,

$$Q_{N_a r} = r^{N_a} - \sum_{i=1}^{r-1} \binom{r}{i} Q_{N_a i} \text{ and } Q_{N_a 1} = 1 \tag{13}$$

The total number of arrangements where R'_{rA} is disjoint with $(R'_{rB} \cup R_c)$ is then,

$$\theta_u = \sum_{r=1}^{N_a} \binom{p}{r} Q_{N_a r} (p-r)^{N\eta} \tag{14}$$

All Disjoints Sets

It is also possible that the sets R'_{rA}, R'_{rB} and R_c are all disjoint. The number of possible arrangements θ_d can be similarly shown to be given by;

$$\theta_d = \sum_{r=1}^{N_c} \left[\binom{p}{r} Q_{N_c r} \sum_{k=1}^{N_a} \binom{p-r}{k} Q_{N_a k} (p-r-k)^{N_b} \right] \tag{15}$$

where $Q_{N_c r}$ and $Q_{N_a k}$ are obtained as in Equation (13). The set $(R'_{rB} \cup R_c)$ also includes the cases where R'_{rB} and R_c are disjoint. Overall, the total number of arrangements of either R'_{rA} being disjoint with $(R'_{rB} \cup R_c)$, or R'_{rB} being disjoint with $(R'_{rA} \cup R_c)$, or all three sets R'_{rA}, R'_{rB} and R_c disjoint is,

$$\theta_t = 2\theta_u - \theta_d \tag{16}$$

The probability of finding a traitor node is then,

$$P_t = \frac{\theta_t}{p^{2N\eta - N}} = \frac{2\theta_u - \theta_d}{p^{2N\eta - N}} \tag{17}$$

With suitable keying parameters, the probability of finding a traitor node can be made very small. For example, with $N = 7, \eta = 6$ and $p = 31$, the probability is only 5.04×10^{-15}.

5.2.3. Expected Node Capture n_c to Find a Traitor Node

We assume the attacker is able to capture any number of nodes, and as each new node is captured, it is paired with each of the previous ones to find a traitor. Since the probability of finding a traitor node is P_t, the expected number of attempts to find one is $\frac{1}{P_t}$.

Each node has η public keys to try, so each pair of nodes allows η^2 attempts. If the number of nodes captured is n_c, the number of pairs that can be formed is $\binom{n_c}{2}$, giving a total of $\eta^2 \binom{n_c}{2}$ pairing attempts. To find a traitor node, we have,

$$\begin{aligned} \eta^2 \frac{n_c!}{2!(n_c-2)!} &\geqslant \frac{1}{P_t} \\ \textit{i.e.,} \quad n_c &\geqslant \frac{1}{2}\left(1 + \sqrt{1 + \frac{8}{\eta^2 P_t}}\right) \end{aligned} \tag{18}$$

The expected number of captured nodes n_c required to find a traitor node is shown in Table 3 for some keying parameters. It can be seen that for these cases, thousands of nodes need to be captured, just to find one traitor node.

Table 3. Capture sizes n_c to find a traitor node.

η	N	Prime Modulus, p				
		13	17	19	23	31
6	6	2.28×10^{7}	5.74×10^{6}	2.98×10^{6}	8.66×10^{5}	9.96×10^{4}
	7	1.03×10^{9}	2.52×10^{8}	1.28×10^{8}	3.48×10^{7}	3.32×10^{6}
	8	4.68×10^{10}	1.13×10^{10}	5.63×10^{9}	1.47×10^{9}	1.22×10^{8}
7	6	1.29×10^{9}	3.17×10^{8}	1.60×10^{8}	4.34×10^{7}	4.07×10^{6}
	7	1.19×10^{11}	2.85×10^{10}	1.42×10^{10}	3.66×10^{9}	2.95×10^{8}
	8	1.09×10^{13}	2.58×10^{12}	1.27×10^{12}	3.20×10^{11}	2.33×10^{10}
8	6	7.56×10^{10}	1.82×10^{10}	9.04×10^{9}	2.33×10^{9}	1.90×10^{8}
	7	1.39×10^{13}	3.30×10^{12}	1.63×10^{12}	4.07×10^{11}	2.94×10^{10}
	8	2.55×10^{15}	6.03×10^{14}	2.96×10^{14}	7.26×10^{13}	4.86×10^{12}

Legend: Key sizes 🟨 64 bits, 🟥 80 bits, ⬛ 96 bits, 🟦 128 bits

Finding a traitor node does not break the scheme, but only slightly improves the chances of finding the PPMka in subsequent pairings.

5.3. Limited Capture Pairing Attack

In this case, the attacker, having obtained $\frac{m}{n}$ (sufficient) nodes, would try to obtain the master keys by solving the system of equations formed from the captured keys. By pairing the nodes using only one of each other's public keys, the set of reduced key sets of $N\eta$ numbers are obtained.

In the ideal case, the pairing would produce exactly N couplings in each node, one for each master key and all related to the same public key. However, if the number of couplings is $N_c > N$, then there are N_c possible ways to associate the related private keys to the public key and one of the master keys, say $\mathbf{M}_1$. Using all of the η public keys one at a time, the number of possible associations, hence the number of sets of equations, obtained from one node is $[N_c]^\eta$ related to the public keys and the master key $\mathbf{M}_1$. Using all of the $\frac{m}{\eta}$ captured nodes, the $m\times m$ equations required are obtained and solved for the master $\mathbf{M}_1$. The number of sets of equations possible to solve for $\mathbf{M}_1$ is $[N_c]^{\eta\frac{m}{\eta}} = [N_c]^m$.

After obtaining the first master key, the exposed private key is removed, leaving $N_c - 1$ keys to choose from to solve for the next master key. In total, to solve for all of the master keys, the possible number of sets of equations, *i.e.*, the number of iterations required, is:

$$\Phi = \sum_{i=0}^{N-1} [N_c - i]^m$$

Binomial Distribution Approximation

Figure 7 shows the distribution of the number of couplings in the pairing attacks for the case $p = 31$, $N = 6$, $\eta = 6$. Other cases exhibit the same distribution, and they suggest that the distribution of the number of couplings x can be approximated by the binomial distribution,

$$\begin{aligned} P(X = x) &= \binom{N\eta}{x} p_r^x (1 - p_r)^{(N\eta - x)} \\ \text{where the mean is} \quad \mu &= N\eta p_r \end{aligned} \tag{19}$$

Figure 7. Distribution of the number of couplings for $p = 31, N = 6, \eta = 6$.

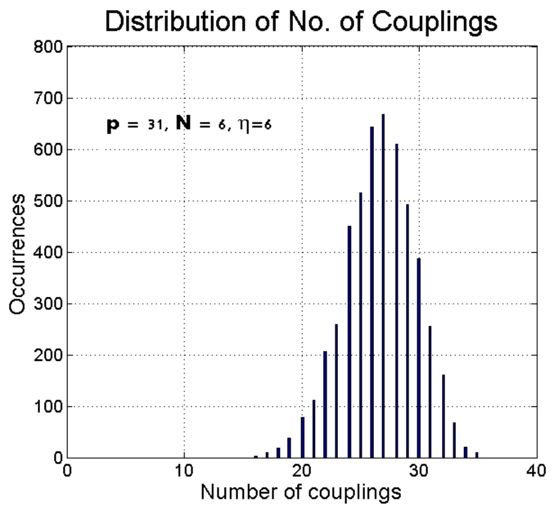


Table 4. Values of $log(\Phi)$. Probable number of master key solutions, Φ.

η	N	$m = 12$			$m = 16$			$m = 24$		
		13	17	31	13	17	31	13	17	31
6	6	17.55	17.37	16.98	23.40	23.15	22.64	35.10	34.73	33.96
	7	18.38	18.22	17.90	24.50	24.30	23.86	36.76	36.44	35.79
	8	19.09	19.09	18.68	25.46	25.46	24.90	38.19	38.19	37.35
7	6	18.22	18.22	17.73	24.30	24.30	23.63	36.44	36.44	35.45
	7	19.09	19.09	18.68	25.46	25.46	24.90	38.19	38.19	37.35
	8	19.95	19.84	19.48	26.60	26.45	25.97	39.91	39.68	38.96
8	6	18.96	18.82	18.53	25.28	25.09	24.71	37.91	37.64	37.06
	7	19.84	19.82	19.48	26.45	26.30	25.97	39.68	39.44	38.96
	8	20.59	20.49	20.28	27.46	27.32	27.04	41.18	40.98	40.56

Legend: Key sizes 🟨 64 bits, 🟥 80 bits, ⬛ 96 bits, 🟦 128 bits

From Equation (17), we can compute the probability of N couplings, *i.e.*, $P(X = N)$. After solving for p_r, we obtain the mean $\mu = (N\eta)p_r$. Then, using the expected number of couplings in a pairing as $N_c = \mu$, the number of iterations required is,

$$\Phi = \sum_{i=0}^{N-1} [N_c - i]^m = \sum_{i=0}^{N-1} [\mu - i]^m \tag{20}$$

Table 4 gives the probable number of master keys solutions 10^{Φ} for various keying parameters.

5.4. Experimental Results of Pairing Attacks

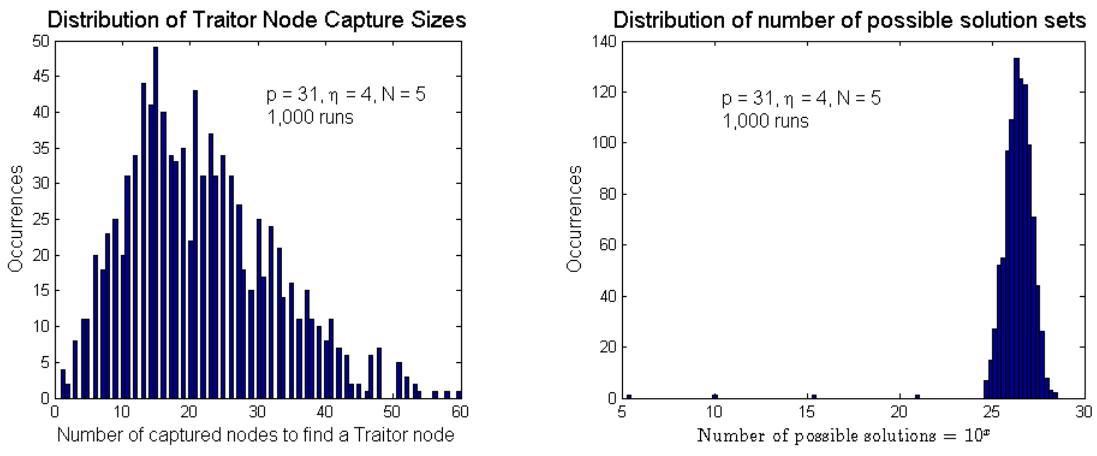
A computer programme was used to implement the pairing attacks to determine the traitor capture sizes n_c and the number of possible master key solutions Φ. The programme first generates the master keys. It then randomly creates new nodes with unique IDs to simulate captured nodes. As each node is created, it is paired with each of the previously "captured" nodes until a traitor node is found. At the same time, the number of couplings is accumulated for the first $\frac{m}{\eta}$ nodes. This is the probable number of couplings in the limited captured case. When a traitor node is found, a new implementation is made using a new set of master keys and this is repeated for 1000 runs.

These are real attacks on real systems as the public and private keys can be implemented in real sensor nodes. They are "simulated" attacks in the sense that capturing the nodes and extracting the keys are done in the computer programme. This greatly accelerates the attacks. Real-life attacks would require much more effort and time.

Due to the large traitor capture sizes, only cases that give results within a reasonable time is given in Table 5. These results are the mean values for 1000 runs for each case, except for the case $\eta, N = 5$, where the results were for 600 runs, due to the long execution times for each run.

Figure 8 show the typical distribution of the results of pairing attacks over 1000 runs for the simple case $m = 24,\ p = 31,\ \eta = 4,\ N = 5$. The experimental results were quite closely comparable with our analytical results (see Table 5), even though the capture sizes are slightly smaller. This may be due to the random number generator used in the computer programme.

Figure 8. Result of pairing attacks on the scheme using $m = 24, p = 31, \eta = 4, N = 5$.



(a) Traitor capture sizes (b) Number of solution sets

Table 5. Comparison: analytical and experimental results for 1000 runs using $p = 31$.

η	N	Traitor Capture n_c		Number of Solutions, Φ	
		Equation (18)	Expt.	Equation (20)	Expt.
4	4	5.59	5.23	7.97×10^{22}	1.06×10^{24}
	5	23.23	21.48	5.43×10^{26}	8.46×10^{26}
	6	128.05	113.53	7.92×10^{28}	1.10×10^{29}
5	4	24.45	21.37	7.95×10^{26}	8.16×10^{26}
	5	237.99	*209.22	7.93×10^{28}	*9.96×10^{29}
6	3	10.76	9.62	1.00×10^{24}	1.17×10^{25}
	4	155.91	135.88	1.68×10^{28}	1.42×10^{29}
7	3	37.57	33.04	7.95×10^{25}	7.17×10^{26}

* 600 runs only, due to long execution times.

6. Performance and Implementation

6.1. Performance

6.1.1. Implicit Authentication

The BYka scheme implicitly authenticates itself, since success in obtaining the common pairwise key is only possible if both nodes obtained their private key sets from the TA or its subsidiary. There is no need to authenticate the ID, since an illegitimate node providing a false ID cannot compute a common pairwise key with a legitimate node.

6.1.2. Communication Overheads

The initial public key exchange requires the public ID to be transmitted. These are integers $\in [0, q-1]$. Using $q = 65521$, the number of bits is 16 bits. This saves time and, more importantly, energy for transmission.

6.1.3. Compact Code

The pairwise key computation code is very simple and requires only a few steps. The pseudo code is given in Listing 1.

6.1.4. Memory Requirements

During execution, RAM is required for some counters, the pairwise key, some temporary data, the $N\eta^2$ numbers in the pairwise key set and the counterpart's public keys. While the $m\eta$ elements of the public keys need to be computed, it is possible to write the code such that only one element is used at a time, requiring only one memory space in RAM. Overall, the largest amount of RAM required is for the pairwise key, $Q_R = N\eta^2 \times b$ bits, where b is the data size in bits. Since our typical prime modulus is $p \leqslant 31$, *i.e.*, $b \leqslant 5$ bits, we can simplify coding if we use one byte for the data size. The private key set requires the largest storage, $Q_o = \eta N m \times b$ bits, or $Q_o = \eta N m$ bytes if one byte is used to store each b bit integer. As it is static, it can be stored in ROM.

```
Input: Neighbour node's public ID
Output: The pairwise key $K_{pair}$
Generate all the public key seeds
for each public key seed do
    generate public key vector (mod $q$)
    for each private key do
        multiply with the public key vector (mod $p$)
        save result in key set $R$
    end
end
for each $R_i$ do
    $K_{pair} = K_{pair} \cdot (R_i + 1)$ (mod $S_k$)
end
```

Listing 1: BYka pairwise key computation pseudo code.

6.1.5. Computation Time

The main parts of the computation include generating the public key vectors involving $(m-2)\eta$ modulo multiplications and computing the numbers in the pairwise key set involving $mN\eta^2$ modulo multiplications and $(m-1)N\eta^2$ modulo additions. The modulo operations are on small integers, except for the final pairwise key computation. The experimental results to obtain the computation times for the BYka scheme in the MICAzmote [17], which has an eight-bit ATmega128 processor running at 8 MHz with 4 KB RAM, 4 KB EEPROMand 128 KB flash memory, implemented using TinyOS [18], gave the following linearised result,

$$T_{comp} = 0.0428[mN\eta^2 + (m-2)\eta] + 23.72 \quad \textit{ms}$$

6.1.6. Scalability

The scalability of the BYka scheme is limited by the key space sizes of the pairwise keys, private keys and the public keys. Except for the public keys, these key spaces are very large. The public key is limited by the number of the number of sets of public key seeds, $\approx \frac{q}{\eta}$. Using $q = 65521$, there are about 10,000 possible nodes, while using a 32-bit prime for q, it is possible to have about 600×10^6 nodes.

6.2. Implementation

The parameters need to be selected for system performance and the desired level of resilience. In general, larger values of m, N and η increase the resilience, but also increase the memory requirements and the computation times. Smaller values of p reduce the chance of discovering the PPMka information, but also reduce the pairwise key space. A good choice is $p = 31$, and being a Mersenne prime, the modulo operation can be done very efficiently. Table 6 can be used as a guide to select the keying parameters for the case using master key matrix size $m = 16$.

Table 6. Security and performance features using $m = 16$. S_k is the pairwise key size, traitor node capture size n_c, number of possible master key solutions Φ. *Computation times are for the MICAz mote with an eight-bit CPU at 8 MHz with 4 KB ROM 4 KB RAM 128 KB flash.

η	N	$p = 13$		$p = 17$		$p = 31$		Q_o	T^*_{comp}
		$S_k \geqslant 64$ bits		$S_k \geqslant 80$ bits		$S_k \geqslant 128$ bits			
		$log(n_c)$	$log(\Phi)$	$log(n_c)$	$log(\Phi)$	$log(n_c)$	$log(\Phi)$	(bytes)	(ms)
6	6	🟨 7.36	🟨 23.40	🟥 6.76	🟥 23.16	⬛ 5.00	⬛ 22.64	576	175
	7	🟨 9.01	🟨 24.50	🟥 8.40	🟥 24.30	🟦 6.52	🟦 23.86	672	200
	8	🟨 10.67	🟨 25.46	🟥 10.05	🟥 25.46	🟦 8.09	🟦 24.90	768	225
7	6	🟨 9.11	🟨 24.30	🟥 8.50	🟥 24.30	🟦 6.61	🟦 23.63	672	229
	7	🟨 11.08	🟨 25.46	🟥 10.46	🟥 25.46	🟦 8.47	🟦 24.90	784	263
	8	🟨 13.04	🟨 26.60	🟥 12.41	🟥 26.45	🟦 10.37	🟦 25.97	896	296
8	6	🟨 10.88	🟨 25.28	🟥 10.26	🟥 25.09	🟦 8.28	🟦 24.71	768	292
	7	🟨 13.14	🟨 26.45	⬛ 12.52	⬛ 26.30	🟦 10.47	🟦 25.97	896	335
	8	🟨 15.41	🟨 27.46	⬛ 14.78	⬛ 27.32	🟦 12.69	🟦 27.04	1024	379

Legend: Key sizes 🟨 64 bits, 🟥 80 bits, ⬛ 96 bits, 🟦 128 bits

7. Discussions

7.1. Exclusive Communications

Our scheme only enable pairs of nodes belonging to the same TA to establish pairwise keys with each other. There is no possibility for pairwise key establishment with non-member nodes, which can be a desirable feature for sensor networks.

7.2. Key Escrow

The trusted authority is the key escrow entity and must be well protected. The TA is able to obtain all of the keys and decipher all previously recorded messages. This may be a desirable feature within some organisations. In the BYka scheme, the master keys generation and storage can be dispersed among a committee of TA's. In this way, protection against some rouge TA's is possible, since they must all work together to generate the full set of keys.

7.3. Compromised Key

If the private keys of a node are obtained, the adversary is able to obtain all previous keys and decrypt all previously recorded messages. There is no perfect forward secrecy. In addition, the BYka scheme is vulnerable to the compromised-key impersonation attack where, if a node C is compromised, an adversary E cannot only impersonate node C, it can also use the stolen keys to impersonate any other nodes to communicate with C. For example, node E has obtained node C's keys. It impersonates node B and sends ID_B to node C, which uses it to compute the pairwise key K_{CB}. Unknown to C, node E also uses ID_B with C's private keys to compute the same pairwise key K_{CB}.

8. Conclusion

We proposed a new authenticated key agreement scheme where pairs of nodes, having obtained each other's public key IDs, can compute large common pairwise keys using their private keys obtained from the same trusted authority. The initial public key exchange is only a few bits, the size of the public key ID, a 16 bit integer, saving on time and energy. The computations use simple modulo arithmetic operations on small integers, making it fast, efficient and requiring few resources. These features make it very attractive for use as the cryptographic primitive for secure communications in low-resource devices, such as wireless sensor nodes, especially in *ad hoc* and mobile network applications.

We analysed the security of the scheme against a powerful attacker who is able to capture any number of nodes and extract all of the keying material. Our analysis showed that the captured keys cannot be used directly to break the scheme. The attacker must first discover for each private key the public key and master key used to compute it, *i.e.*, the private-public-master-key associations (PPMka).

We showed how an attacker may use captured nodes to discover the PPMka information. We obtained analytical results to calculate the probabilities of successfully breaking the scheme using these compromised nodes. These results were verified using computer simulated attacks. We showed that using suitable keying parameters, the attacker would need to capture tens of thousands of nodes or, alternatively, try an unfeasibly large number of solutions. The probability of breaking the scheme would be so small, that it is virtually unconditionally secure.

Finally, we presented some implementation parameters to achieve the desired performance in terms of computation time, key size and memory requirements for the MICAz mote.

Author Contributions

This paper is part of Mee Loong's PhD research and both Adnan and William contributed to the supervision.

Conflict of Interest

The authors declare no conflict of interest.

References

1. Yang, M.L.; Al-Anbuky, A.; Liu, W. A Fast and Efficient Key Agreement Scheme for Wireless Sensor Networks. In Proceedings of International Conference on Wireless and Mobile Communications, Venice, Italy, 24–29 June 2012; pp. 231–237.

2. Yang, M.L.; Al-Anbuky, A.; Liu, W. The Multiple-Key Blom's Scheme for Key Establishment in Mobile Ad Hoc Sensor Networks. In Proceedings of the 19th Asia-Pacific Conference on Communications, Bali, Indonesia, 29–31 August 2013; pp. 422–427.
3. Yang, M.L.; Al-Anbuky, A.; Liu, W. Security of the Multiple-Key Blom's Key Agreement Scheme for Sensor Networks. In *ICT Systems Security and Privacy Protection*; Cuppens-Boulahia, N.; Jajodia, S.; Cuppens, F., Eds.; Springer: Berlin/Heideberg, Germany, 2014; pp. 66–79.
4. Blom, R. Non-Public Key Distribution. *Advances in Cryptology*; Springer: Berlin/Heideberg, Germany, 1983; pp. 231–236.
5. Blom, R. *An Optimal Class of Symmetric Key Generation Systems*; Technical Report; Linkopping University, Linkopping, Sweden, 1984.
6. Menezes, A.J.; Oorschot, P.C.; Vanstone, S.A. *Handbook of Applied Cryptography*; CRC: Boca Raton, FL, USA, 2001.
7. Blundo, C.; De Santis, A.; Herzberg, A.; Kutten, S.; Vaccaro, U.; Yung, M. *Perfectly-Secure Key Distribution for Dynamic Conferences*; Technical Report; Universita di Salerno, Baronissi, Italy, 1995.
8. Liu, D.; Ning, P. Establishing Pairwise Keys in Distributed Sensor Networks. In Proceedings of the 10th ACM Conference on Computer and Communications Security, Washington, DC, USA, 27–30 October 2003.
9. Eschenauer, L.; Gligor, V.D. A key-management scheme for distributed sensor networks. In Proceedings of the 9th ACM Conference on Computer and Communications Security, Washington, DC, USA, 18–22 November 2002; pp. 41–47.
10. Du, W.; Han, S.Y.; Deng, J.; Varshney, P.K. A Pairwise Key Pre-distribution Scheme for Wireless Sensor Networks. In Proceedings of the Conference on Computer and Communications Security, Washington, DC, USA, 27–30 October 2003.
11. Lee, J.; Stinson, D.R. Deterministic Key Predistribution Schemes for Distributed Sensor Networks. In *Selected Areas in Cryptography*; Springer-Verlag: Berlin/Heidelberg, Germany, 2005; Volume 3357, pp. 294–307.
12. Chen, N.; Yao, J.B.; Wen, G.J. An Improved Matrix Key Pre-distribution Scheme for Wireless Sensor Networks. In Proceedings of International Conference on Embedded Software Systems, Chengdu, China, 29–31 July 2008; pp. 40–45.
13. Zhang, W.; Zhu, S.; Cao, G. A Random Perturbation-Based Scheme for Pairwise Key Establishment in Sensor Networks. In Proceedings of MobiHoc'07, Montrï£¡ï£¡al, QC, Canada, 9–14 September 2007.
14. Chien, H.Y.; Chen, R.C.; Shen, A. Efficient Key Pre-distribution for Sensor Nodes with Strong Connectivity and Low Storage Space. In Poceedings of the 22nd International Conference on Advanced Information Networking and Applications (AINA'08), Okinawa, Japan, 25–28 March 2008; pp. 327–333.
15. Yu, C.M.; Lu, C.S.; Kuo, S.Y. Noninteractive Pairwise Key Establishment for Sensor Networks. *IEEE Trans. Inf. Forensics Secur.* **2010**, *5*, 556–569.
16. Zhou, J.; He, M. An Improved Distributed Key Mangement Scheme in Wireless Sensor Networks. In *Information Security Applications*; Springer: Berlin/Heideberg, Germany, 2009; pp. 305–319.

17. Memsic Corp. MICAz Datasheet. Available online: http://www.docstoc.com/docs/20049970/MICAz-Datasheet (accessed on 17 June 2014).
18. Levis, P.; Gay, D. *TinyOS Programming*; Cambridge University Press: Cambridge, UK, 2006. Available online: http://csl.stanford.edu/~pal/pubs/tinyos-programming.pdf (accessed on 17 June 2014)