Information Processing Letters 116 (2016) 367–372

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

2D Hash Chain robust Random Key Distribution scheme

Mohammad Ehdaie [a,*], Nikos Alexiou [b], Mahmoud Ahmadian [a], Mohammad Reza Aref [c], Panos Papadimitratos [b]

[a] CCL, K.N. Toosi University of Technology, Iran
[b] The School of Electrical Engineering, KTH, Sweden
[c] ISSL, Sharif University of Technology, Iran

ARTICLE INFO

Article history:
Received 15 July 2013
Received in revised form 29 October 2015
Accepted 15 December 2015
Available online 18 December 2015
Communicated by L. Viganò

Keywords:
Wireless Sensor Network
Random Key Distribution
Node capture
Hash Chain
Distributed systems

ABSTRACT

Many Random Key Distribution (RKD) schemes have been proposed in the literature to enable security applications in Wireless Sensor Networks (WSNs). A main security aspect of RKD schemes is their resistance against node capture attacks, since compromising the sensors and capturing their keys is a common risk in such networks. We propose a new method, based on a 2-Dimensional Hash Chain (2DHC), that can be applied on any RKD scheme to improve their resilience. Our method maintains the flexibility and low cost features of RKD schemes and it doesn't require any special-purpose hardware or extra memory to store keys in the sensors. We demonstrate that our approach significantly increases the resilience of RKD schemes against node capture at the cost of a few additional computations, while maintaining network connectivity at the same level.

1. Introduction

The first step for securing *Wireless Sensor Network (WSN)* applications is to distribute keys amongst sensors. A *Random Key Distribution (RKD)* scheme assigns key to sensors and specifies how they can find common keys to establish a secure channel. A physical node compromise is always a risk: an adversary who captures some nodes in a network obtains their keys and then, is able to compromise communication links between benign nodes (e.g., see [1]).

We aim at increasing resilience of RKD schemes for WSNs to node capture, while maintaining the flexibility and low cost features of RKD. We propose the *2-Dimensional Hash Chain (2DHC)* scheme that leverages cryptographic hash-chains to create a resilient and adaptive key discovery mechanism. We demonstrate that our scheme can significantly improve resilience against node capture attacks and thus, the overall security of RKD schemes. Moreover, this scheme does not increase per-node storage, or any significant additional computation and communication overhead.

The rest of this paper is organized as follows: Section 2 presents succinctly related work on RKD schemes. Section 3 defines the adversary models and the problem at hand. 2D Hash Chain technique is presented in Section 4. Finally, in Section 5 we conclude the results.

2. Related work

In a RKD scheme, each sensor is equipped with a set of symmetric keys, called the key ring, randomly chosen from a large pool of keys. To discover their common keys, a *Key Discovery Protocol (KDP)* has to be run between the sensors that want to establish a secure link. Two main approaches to establish common symmetric keys between the sensors are the basic scheme [2] and the q-composite [3] scheme, where the sensors have to discover one and at least q common keys, respectively, to establish a secure link.

There are several works in the literature that engage hash functions to increase the resilience of RKD schemes.

* Corresponding author.
E-mail address: mohammad@ehdaie.com (M. Ehdaie).

http://dx.doi.org/10.1016/j.ipl.2015.12.006
0020-0190/

They usually develop one or several chains of keys or key materials and take advantage of the one-way property of hash functions.

One interesting improvement is the so-called *Key-Chain improvement* [4]: consider a key pool as in the basic scheme. Extend the pool by creating some hash chains with the primary keys as roots of the chains. For each node, select m keys randomly from the extended key pool, such that no more than one key is selected from each chain.

If the primary key pool size and memory size are the same as in the basic scheme, connectivity will not affected: each pair of nodes can establish a secure channel, if and only if they have keys from the same chain. The node that owns the key which is closer to the chain root can traverse the chain downwards to find the shared key carried by the second node. This idea can decrease the chances of an adversary to compromise secure links in the network: consider a link that is secured with a key, say K. Even if the adversary captures some nodes and obtains a key in the chain corresponding to K, she succeeds if and only if the adversary's key is closer to the root than K. The probability the adversary fails to compromise the link given that she owns a key in the same chain as of K, shows the percentage of improvement in the resilience of RKD scheme. This probability depends on the chain length. For the best case, when the chain length tends to infinity, the improvement over the basic scheme would be 33% [4].

In the following section, we present our scheme using a family of one-way functions and show that our scheme can lead to a 55% improvement over the basic scheme.

3. System model & problem statement

Assume a WSN that runs the basic RKD scheme to distribute the keys between the sensors. Also, consider an adversary who randomly captures s nodes and obtains m stored keys per captured sensor. With the stolen keys in his/her possession, the adversary is then able to compromise communication links between benign nodes if the key (or the keys) used to secure the link(s) is/are included in the key rings of the captured nodes. The adversarial gain is evaluated in terms of broken communication links given s captured nodes, using a fail function [2] per key distribution scheme.

$$Fail(s) = \frac{\text{Num. of comp. links given s comp. nodes}}{\text{Number of all network links}} \quad (1)$$

Our objective is to reduce the fail function value, i.e. the chance of the adversary to break secure communication links given captured sensors.

Notation: We summarize the notations we use throughout the paper here:

- P: key pool.
- $|P|$: key pool size.
- m: Key ring size.
- p_c: Probability that two nodes can establish a secure connection.
- s: Number of captured nodes in the network.
- l: Key-chain length.
- $p(i)$: Probability of two nodes sharing exactly i keys.
- q: The minimum number of shared keys for two nodes to have a secure link (q-composite scheme).

4. 2D Hash Chain technique

4.1. Description of the technique

4.1.1. Preliminaries

Consider two one-way functions, say h and h', with the commutative property, i.e. for every input x, we have $h(h'(x)) = h'(h(x))$. Modular exponentiation is one example of such functions. If we define $h(x) = x^a \bmod p$ and $h'(x) = x^b \bmod p$ for some a, b and p, we have:

$$\begin{aligned} h(h'(x)) &= h(x^b \bmod p) = x^{a\times b} \mod p \\ &= h'(x^a \bmod p) = h'(h(x)) \end{aligned} \quad (2)$$

Besides, if the parameters are properly chosen, computing x from p, a and $h(x)$ is hard (also from p, b and $h'(x)$). These imply that modular exponentiation is a family of one-way functions with the commutative property. For more information about such functions, see [5].

Now, if we apply h, i times on an input like x, and apply h' on the result j times, we get $h'^j(h^i(x))$ which is equal to $h^i(h'^j(x))$. In other words, the order of applying these functions does not affect the result.

The above mentioned property implies that knowing the value of $h^{i_a}(h'^{j_a}(x))$ for a given i_a and j_a, computing $h^{i_b}(h'^{j_b}(x))$ is easy for any $i_b \geq i_a$ and $j_b \geq j_a$. But, if we have $i_b < i_a$ or $j_b < j_a$, then computing $h^{i_b}(h'^{j_b}(x))$ from $h^{i_a}(h'^{j_a}(x))$ is impractical.

4.1.2. Pool generation

Assume that we have a pool of keys like the basic scheme, of size $|P|$. We extend the pool by creating a 2-dimensional chain from each key in the key pool. The chains are created according to two one-way functions, say h and h', with commutative property, as mentioned in Sec. 4.1.1.

Fig. 1 shows an illustration of a chain. The root of the chain, k_1, could be any of the elements of the pool. The element in ith row and jth column of the chain is obtained by applying the functions h and h' on the root, $i-1$ times and $j-1$ times, respectively. The chain is assumed to be of size l.

4.1.3. Keys assignment

For each node in the network, choose m chains randomly, and then pick one element from each chain with equal probability. Next, store the selected elements in the sensor. In this way, each sensor has m random keys from the extended pool and there is no more than one element from each chain in a sensor.

4.1.4. Link establishment

Similar to the basic scheme, each pair of nodes that owns keys from the same chain can find a shared key and establish a secure channel. Assume that two nodes, say node A and node B, both own a key from a chain. Let $h^{i_a}(h'^{j_a}(k_0))$ and $h^{i_b}(h'^{j_b}(k_0))$ denote the keys belonging to A and B, respectively. There are four potential cases:

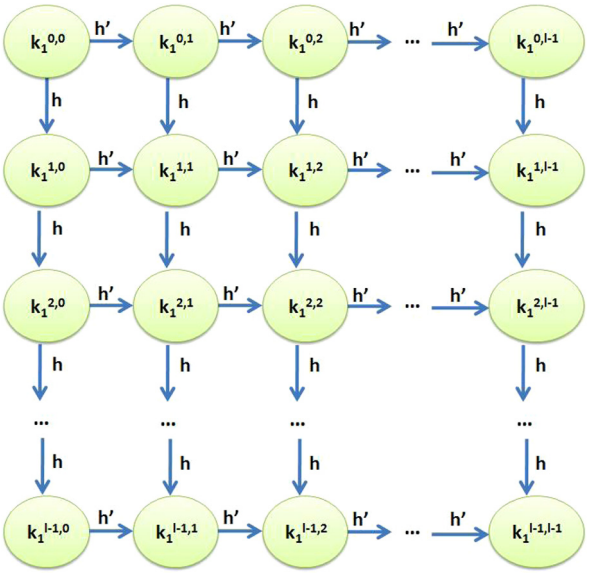

Fig. 1. A 2-Dimensional Hash Chain.

- $i_a \leq i_b$ and $j_a \leq j_b$: In this case, A can proceed in the chain, probably both vertically and horizontally, to create the same key held by B. Therefore, $h^{i_b}(h'^{j_b}(k_0))$ can be used as a shared key between A and B to secure their communications.
- $i_a \geq i_b$ and $j_a \geq j_b$: Now, B can proceed in the chain to reach the point by A.
- $i_a \leq i_b$ and $j_a \geq j_b$: In this case, both A and B should proceed in the chain, A, vertically and B, horizontally, to reach a new point. So, $h^{i_b}(h'^{j_a}(k_0))$ can be used as a shared key between A and B to secure their communications.
- $i_a \geq i_b$ and $j_a \leq j_b$: This is like the previous case, but, in a reverse process. $h^{i_a}(h'^{j_b}(k_0))$ can be used as a shared key between A and B to secure their communications.

This shows that if the parameters of the system are the same as in the basic scheme, the connectivity will not be violated: every two nodes that share a key in the primary pool can establish a secure channel. Their shared key will be $h^{max(i_a,i_b)}(h'^{max(j_a,j_b)}(k_0))$.

4.2. Security analysis

Using the fail function [2], we study the effect of this technique on the resilience of the scheme against node capture.

In the basic scheme, when s nodes are captured in the network, the fraction of compromised links between benign nodes would be [2]:

$$Fail_{basic}(s) = 1 - (1 - \frac{m}{|P|})^s \tag{3}$$

In the Key-Chain improvement scheme [4], the fraction of compromised links between benign nodes is decreased to:

$$Fail_{chain\text{-}improvement}(s) = 1 - (1 - \alpha \times \frac{m}{|P|})^s \tag{4}$$

provided α is a parameter calculated according to the key chain length, l, and is given by:

$$\alpha = 1 - \frac{\sum_{i=1}^{l-1} i^2}{l^3} \tag{5}$$

Actually, this is a simplified formula to show the effect of the Key-Chain improvement technique. The exact formula for the fraction of compromised links is given in [4]. Numerical illustration shows that the result does not change significantly. A sketch of the proof for the above formula is appeared in Appendix A.

α is a parameter that shows the effect of the key chain improvement on the resilience of the basic scheme. The closer α to 0 is, the more improvement in the resilience of the scheme is achieved. $\alpha = 1$ is the same as the basic scheme. Indeed, $1 - \alpha$ could be seen as the approximate percentage of improvement in the resilience of the scheme. This is because:

$$1 - (1 - \alpha \times \frac{m}{|P|})^s \simeq \alpha \times (1 - (1 - \frac{m}{|P|})^s) \tag{6}$$

while s has a reasonably low value and $m \ll |P|$.

As a numerical illustration, when l tends to infinity, the value of α tends to $\frac{2}{3}$, which is the best case for the resilience improvement (33% improvement: $1 - \alpha = \frac{1}{3}$). It is somehow trivial: when the chain length is large enough, the probability that adversary's key is the farthest from the root among three keys (adversary's key, A's key and B's key) will be $\frac{1}{3}$. It implies that if the adversary captures a key from the same chain as that of A and B shared key, the probability that he/she fails to break the link between A and B is:

$$Prob\{i_c > max(i_a, i_b)\} = \frac{1}{3} \tag{7}$$

where i_a, i_b and i_c denote the index of A's, B's and the captured key, respectively, in the chain.

Now, in the 2DHC, consider the link between A and B and assume that $h^{i_a}(h'^{j_a}(k_0))$ and $h^{i_b}(h'^{j_b}(k_0))$ denote the keys belonging to A and B, respectively. Thus, they can use $h^{max(i_a,i_b)}(h'^{max(j_a,j_b)}(k_0))$ as the shared key to secure their link. Given that the adversary obtains a key in the same chain with k_0, the link will be compromised if and only if:

$$i_c \leq max(i_a, i_b) \text{ and } j_c \leq max(j_a, j_b) \tag{8}$$

Since the two above events are independent and identical, the probability that the adversary be likely is:

$$\begin{aligned} &prob\{i_c \leq max(i_a, i_b) \text{ and } j_c \leq max(j_a, j_b)\} \\ &= prob\{i_c \leq max(i_a, i_b)\} \times prob\{j_c \leq max(j_a, j_b)\} \\ &= prob^2\{i_c \leq max(i_a, i_b)\} = \alpha^2 = (1 - \frac{\sum_{i=1}^{l-1} i^2}{l^3})^2 \end{aligned} \tag{9}$$

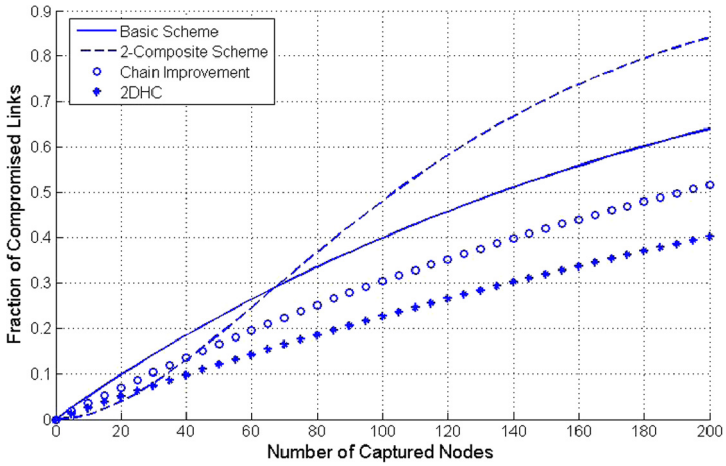

Fig. 2. Fail function for the basic scheme, the 2-composite scheme, the chain improvement scheme and the 2DHC scheme with $\alpha = 0.71 (l = 10)$.

and for the Fail function, we have:

$$Fail_{basic,2DHC}(s) = 1 - (1 - \alpha^2 \times \frac{m}{|P|})^s \simeq \alpha^2 \times Fail_{basic}(s) \tag{10}$$

As we observed in the previous section, $\alpha = prob\{i_c \leq max(i_a, i_b)\}$ tends to $\frac{2}{3}$ for sufficiently large values of l. Hence, in 2DHC, the "adversary chance" could reach $(\frac{2}{3})^2 = \frac{4}{9}$ for proper chain length. It is a significant reduction in the adversary chance for compromising the links (more than 55% resilience improvement over the basic scheme).

As a numerical illustration, we plotted the Fail function for the 2DHC scheme with $\alpha = 0.71$ (i.e. $l = 10$) in comparison with the basic scheme as well as the Chain Improvement scheme, in Fig. 2.

4.3. More discussions

4.3.1. The overhead

2DHC scheme needs no more storage for keys rather than the basic scheme. Also, by using the same number of keys in a key ring, it keeps the same level of connectivity in the network. Thus, there is no communicational overhead.

The significant improvement in the resilience costs only a few extra computations, i.e. at most $2 \times l$ hash computations. Therefore, it is much better than the chain improvement technique (as a 1-Dimensional Hash Chain technique), since the chain improvement technique needs at most l computations and decreases the Fail function by a factor of α, while we need $2 \times l$ computations and the Fail function is decreased by α^2.

As a numerical illustration, we let $l = 5$ in our scheme (corresponding to at most 10 times hash) and get $\alpha = 1 - 0.24 = 0.76$. It yields to a reduction in the Fail function by a factor of $\alpha^2 = 0.58$. In contrast, we can use $l = 10$ in the chain improvement technique to reach the same level of complexity. In this situation, we have $\alpha = 1 - 0.29 = 0.71$. The Fail function is reduced by a factor of $\alpha = 0.71$. Hence, the reduction in 2DHC scheme is much better than the reduction in the chain improvement technique with the same level of complexity.

4.3.2. Comparison with the RSA and the ECC systems

We suggested modular exponentiation as a one-way function with commutative property for using in our scheme. However, it is not mandatory to use this function and the administrator of the system is free to select any function with the mentioned properties.

Here, one question may arise: we know that modular exponentiation is a part of the RSA system, which is impractical to implement on sensor nodes with such restricted resources. So, how can we use it in our technique?

The answer is in the difference between exponents of both systems. In the RSA system, while the exponent in the encryption phase is small, for decryption it uses a very huge number. Thus, it is not practical to implement decryption in the sensor nodes. On the other hand, in 2DHC technique, the exponent could be a small value that makes it possible to be implemented in a restricted resource gadget like a sensor node.

The same argument can be made about the power consumption. In [6], the power consumption for RSA-1024 signature generation and verification is calculated. While signature generation, which corresponds to a large exponent, consumes 304 mWs, signature verification which corresponds to a small exponent (like our one-way functions) consumes less than 12 mWs. Therefore, we can conclude that our one-way function consumes less than 4% of RSA decryption power (equivalent to signature generation).[1]

Another question that may arise is that why not using ECC to encrypt the secret key, exchange it, and then decrypt it in the destination node, instead of multiple modular exponentiations? We first give a rough comparison between the execution time of 2DHC and ECC. For

[1] Some may argue that the power consumption is very high in comparison with typical one-way functions such as SHA-1. While this argument is true, it could be a trade-off between security and power consumption. Since this power consumption occurs once in the neighbor discovery phase of network, it is considered as a reasonable trade-off.

the comparison, we use the data in [7], where the total time for ECC is around 4.1 s while the execution time for RSA public key operation is around 0.79 s If we set $l = 5$ in 2DHC technique, which yields to a reduction in Fail function by a factor of 0.58, the average number of required RSA public key operations is 1.6 times (less than two times) in each node (see Appendix B). Then, the total required time for 2DHC is around 2.5 s, which is better than ECC required time. Additionally, the most significant point is that in the asymmetric cryptography systems like ECC, there should be a trusted center to certify node public keys. Establishing such a trusted center forces an unavoidable and extortionary overhead to the network. Thus, it is much better to run RSA public operation a few times in the beginning of network establish time rather than handling a trusted center for all the network life-time.

4.3.3. Non-RSA one-way functions

According to our best knowledge, examples of one-way functions other than ones based on RSA have not been found yet. If an example will be found with less computation, it could readily be used in 2DHC technique. However, in this part, we introduce a more effective, less secure function, to create the chain. If the network administrator decides that RSA public operation is too expensive to obtain the security, we can use another more effective function that can supply lower security level than modular exponentiation.

Define the first function, $h(x)$, as $x_1|f(x_2)$ and the second function, $h'(x)$, as $f(x_1)|x_2$, where x_1 and x_2 are the first and the second half of the x bit string, respectively, and f is an arbitrary one-way function such that its output length is equal to the length of x_1. This way, h and h' are two functions with commutative property. However, they are not cryptographic one-way functions; because, knowing $h(x)$, one can determine half of the x bit string (but not all of x bits).

Using the above h and h' functions, we can create 2D Hash Chains of keys. From information theoretical point of view, these 2D chains have lower security rather than chains created by modular exponentiation. However, in practice, their security levels are almost the same. If the adversary does not have overwhelming power to brute-force check all the possible values for the unrevealed half of the key, we can assume that h and h' are almost one-way functions with commutative property.

4.3.4. The nDHC technique

We can extend the idea to an n-Dimensional Hash Chain. Assume that we have n one-way functions, say $h_1, h_2, \ldots, h_n$, with commutative property, such that when applying to an input, the result would be independent of the order of indices.

Using such one-way functions, one can create an n-Dimensional Hash Chain (nDHC) from each key in the primary key pool, and distribute keys among sensors in the same manner. The gain for using an nDHC is that the adversary chance can be reduced by α^n.

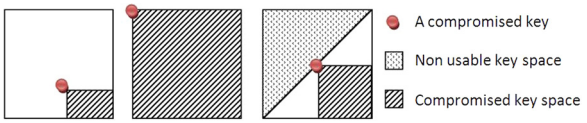

Fig. 3. An idea for mitigating the effect of node capture by using half of the key space.

4.3.5. Using half chain

In 2DHC, if the adversary compromises a key, say $h^i(h'^j(k_0))$, all the keys in the same chain with the indices after (i, j) (below and right of (i, j) position) would be compromised, too. If the adversary succeeds in compromising the chain root, i.e. the key with the index $(0, 0)$, all the chain will be compromised completely. While this is a small probability, it leads to a big loss. Thus, it has a considerable risk. Here, one may think to use a part of key space, e.g. half of the chain, such that compromising neither of the keys yield to compromising all the chain. Fig. 3 shows the motivation for this idea. As illustrated in the figure, if we use half of the key space (the keys on the secondary diagonal of the matrix), the adversary can get at most half of the usable space by compromising a key. This will be discussed in the future work.

4.3.6. Applying on other schemes

2DHC technique is generally applicable to other schemes since it can improve resilience of every RKD scheme. The Fail function for the q-composite scheme can be written as [3]:

$$Fail_{q\text{-}comp}(s) = \sum_{i=q}^{m} (1 - (1 - \frac{m}{|P|})^s)^i \times \frac{p(i)}{p_c} \tag{11}$$

By applying 2DHC technique, we have:

$$\begin{aligned} &Fail_{q\text{-}comp,2DHC}(s) \\ &= \sum_{i=q}^{m} (1 - (1 - \alpha^2 \times \frac{m}{|P|})^s)^i \times \frac{p(i)}{p_c} \\ &\simeq \sum_{i=q}^{m} \alpha^{2i} \times (1 - (1 - \frac{m}{|P|})^s)^i \times \frac{p(i)}{p_c} \\ &\leq \alpha^{2q} \times \sum_{i=q}^{m} (1 - (1 - \frac{m}{|P|})^s)^i \times \frac{p(i)}{p_c} \end{aligned} \tag{12}$$

Eq. (12) shows for the best case, the improvement in the resilience becomes $1 - \alpha^{2q}$. Therefore, the 2DHC has a better effect on the q-composite than the basic scheme.

5. Conclusion

We considered the problem of increasing the resilience of RKD schemes to node capture. We proposed a new technique, 2D Hash Chain (2DHC), which can significantly increase the resilience of RKD schemes. Without increasing the memory storage or mitigating the connectivity level, it decreases the value of fail function. Finally, we have shown that all these benefits come at a few extra communications/computations for the sensors. In upcoming work,

we will investigate the combination of 2DHC with other RKD-based schemes.

Appendix A

The formula for α parameter:

Consider a link between A and B that is secured by $h^{max(i_a,i_b)}(k_0)$. Assume that the adversary captures $h^{i_c}(k_0)$. $1-\alpha$ denotes the probability that i_c, is greater than the other two indices and hence, the adversary cannot obtain the common key $h^{max(i_a,i_b)}(k_0)$.

There are l choices for every index, i.e. from 0 to $l-1$. Thus, the total number of possible outcomes for three random indices is l^3.

For the number of desired outcomes, we set i_c to all possible values and count the desired outcomes in each case. If $i_c = i$, for any i from 1 to $l-1$, the other indices could be any thing from 0 to $i-1$, i.e. i^2 desired outcomes. Hence, we have:

$$1-\alpha = \frac{\sum_{i=1}^{l-1} i^2}{l^3} \tag{13}$$

Appendix B

The number of required hash: Consider a 1D Hash Chain with length l and two nodes that have keys from this chain. The closer node to the root has to run some hashes to reach the other node's key. The expected number of hashes is calculated as following:

Assume that the first node has the ith key in the chain and the other node has the jth key in the chain, $0 \le i, j \le l-1$. The required number of hashes in this situation is $abs(i-j)$, where abs denotes the absolute value. Since there is l^2 possibilities for (i, j) pair, the expected number of hashes is:

$$\frac{1}{l^2}\sum_{i=0}^{l-1}\sum_{j=1}^{l-1} abs(i-j) \tag{14}$$

For example, for $l = 5$, the expected number of hashes is $40/25 = 1.6$. Note that in a 2D Hash Chain, the expected number of all required hashes in both nodes is doubled.

References

[1] P. Papadimitratos, J. Deng, Stealthy pre-attacks against random key pre-distribution security, in: Proceedings of the IEEE International Conference on Communications – Communication and Information Systems Security Symposium, ICC'12 CISS, Ottawa, Canada, 2012, pp. 251–260.

[2] L. Eschenauer, V.D. Gligor, A key-management scheme for distributed sensor networks, in: Proceedings of the 9th ACM Conference on Computer and Communications Security, CCS, 2002, pp. 41–47.

[3] H. Chan, A. Perrig, D. Song, Random key predistribution schemes for sensor networks, in: Proceedings of the 2003 IEEE Symposium on Security and Privacy, Washington, DC, USA, 2003, pp. 197–213.

[4] J. Kur, V. Matyas, P. Svenda, Two improvements of random key predistribution for wireless sensor networks, in: Proceedings of the International Conference on Security and Privacy in Communication Networks, 2012.

[5] J. Benaloh, M. De Mare, One-way accumulators: a decentralized alternative to digital signatures, in: Advances in Cryptology, EUROCRYPT'93, vol. 765, Springer, 1994, pp. 274–285.

[6] A. Wander, N. Gura, H. Eberle, V. Gupta, S. Shantz, Energy analysis of public-key cryptography for wireless sensor networks, in: PERCOM'05: Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications, 2005, pp. 324–328.

[7] H. Wang, Q. Li, Efficient implementation of public key cryptosystems on mote sensors, in: Proceedings of the 8th International Conference on Information and Communications Security, 2006, pp. 519–528.