Two Improvements of Random Key Predistribution for Wireless Sensor Networks

Jiří Kůr, Vashek Matyáš*, and Petr Švenda

Masaryk University, Brno, Czech Republic
{xkur,matyas,svenda}@fi.muni.cz

Abstract. Key distribution is of a critical importance to security of wireless sensor networks (WSNs). Random key predistribution is an acknowledged approach to the key distribution problem. In this paper, we propose and analyze two novel improvements that enhance security provided by the random key predistribution schemes. The first improvement exploits limited length collisions in secure hash functions to increase the probability of two nodes sharing a key. The second improvement introduces hash chains into the key pool construction to directly increase the resilience against a node capture attack. Both improvements can be further combined to bring the best performance. We evaluate the improvements both analytically and computationally on a network simulator. The concepts used are not limited to the random key predistribution.

Keywords: hash function collision, key management, random key predistribution, security, wireless sensor network.

1 Introduction

A wireless sensor network (WSN) consists of resource-constrained and wireless devices called sensor nodes. WSNs monitor some physical phenomenon (e.g., vibrations, temperature, pressure, light) and send measurements to a base station. There are several classes of sensor nodes available – ranging from high-end nodes that can easily employ public-key cryptography down to nodes that can barely make use of any cryptography at all. In our work, we consider cheap and highly constrained nodes that can use only symmetric cryptography and their storage is just a few kilobytes.

Key distribution is one of the greatest challenges in WSNs. Since network topology is usually not a priori known, every node should be able to establish a link key with a large portion of other nodes to ensure the connectivity of the network. To achieve this requirement, nodes may pre-share a single network-wide master key and use it to establish link keys. However, if a single node with the master key is captured, then the whole network gets compromised. In an alternative approach, each node pre-shares a unique link key with every node.

* Final work on this paper undertaken as a Fulbright-Masaryk Visiting Scholar at Harvard University.

This offers much better security, yet hits the memory limits as number of nodes in the network increases.

A suitable trade-off between the two approaches comes with the random key predistribution [1]. Every node is preloaded with a fixed number of keys randomly selected from a given key pool. After the network deployment, two nodes establish a link key if they share at least one key from the key pool. The scheme can be extended to require at least q shared keys [2].

In this paper, we propose two improvements of the basic random key predistribution schemes. In the first improvement, we increase a probability that any two nodes establish a link key while maintaining memory requirements fixed. For this purpose we construct the key pool using limited length (e.g., 80-bit) collisions in a secure hash function. We also provide an evidence that such collisions can be found in a reasonably short time on today's personal computers.

The second improvement introduces hash chains into the key pool structure to directly enhance the resilience against a node capture attack. Both the improvements can be further combined together to bring the best performance. These improvements are particularly advantageous for situations in which the attacker manages to capture a significant number of nodes.

The structure of the paper is following – we review the basic random key predistribution schemes and other related work in Section 2. We present and evaluate the first improvement in Section 3 and the second improvement in Section 4. Then we evaluate their combination in the following section of this paper. We provide computational results from a network simulator and proof of concept for the collision search in Section 6, and then the last section concludes the paper. Proofs of selected equations can be found in the Appendix.

2 Background and Related Work

In this section, we provide a background knowledge on the basic random key predistribution schemes and other related work.

Our proposals modify the basic random key predistribution schemes proposed by Eschenauer and Gligor [1] and Chen et al. [2]. We refer to the schemes as to the *EG scheme* and the *q-composite scheme*, respectively, in our paper.

The *EG scheme* works as follows: in the *(i) key setup* phase, a key pool S is created by randomly taking $|S|$ keys from the possible key space. Then, for every node, m keys are randomly drawn from the key pool S without replacement and uploaded into the node. These keys form a key ring for the given node. If $|S|$ and m are chosen properly, any two nodes in the network share at least one key with a high probability. E.g., for a key pool size $|S| = 10,000$ and a ring size $m = 83$ the probability that any two nodes share a key is $p = 0.5$.

In the *(ii) shared key discovery* phase, every two neighboring nodes try to identify shared keys among their key rings. This can be done by various methods. E.g., every key can be assigned a short unique identifier that is broadcasted by the nodes that have the corresponding key in their key rings. If a shared key is found, it is used as a link key for the communication between the two nodes. If not, the link key can be established in the *path-key establishment* phase.

The *(iii) path-key establishment* phase is optional. It uses already secured links to establish link keys between two neighboring nodes that could not establish a link key directly as they had no shared key or their keys were compromised [3].

The *q-composite scheme* is a generalization of the EG scheme. In the *shared key discovery* phase, two nodes establish a link key only if they share at least q keys in their key rings. The resulting link key is derived from all the shared keys.

2.1 Node Capture Resilience

The performance of random key predistribution schemes is evaluated with respect to the node capture resilience [2]. It can be defined as the probability that a given secured link between two uncaptured nodes can be compromised by an attacker using keys extracted from already captured nodes. In other words, the node capture resilience is a fraction of secured links between uncaptured nodes that can be compromised by an attacker.

The node capture resilience is mostly influenced by the following three factors – the ring size m, the key pool size $|S|$ and the probability that any two nodes in the network can establish a link key. These values are to some extent determined by properties of the network concerned. The ring size m is limited by a storage capacity of the network sensor nodes. If we want the network to be connected by secure links, the minimum required probability of a link key establishment is given by the size of the network and by the average number of neighboring nodes (for details see [1]). Given the m and the minimum required probability, the $|S|$ is uniquely determined. Note that in the q-composite scheme also the q influences the node capture resilience and the key pool size $|S|$.

2.2 Other Related Work

The basic schemes have been modified by Ren et al. [4]. The key pool in their scheme consists of a large number of keyed hash chains where every hash chain element is considered a unique key. Every node is then assigned a number of such keys and a number of whole keyed hash chains represented by their hashing keys and chain starting points. Deterministic and hybrid approaches how to select keys for key rings based on *combinatorial design* are proposed in [5]. These approaches enhance the performance of the basic schemes and similarly to them can be also further improved with our proposals. For other key distribution schemes in WSNs see the survey [6].

Our first proposal is based on hash collisions. Rivest and Shamir took an advantage of hash collisions for a security gain in the MicroMint micro payment scheme, where an electronic coin was represented by a hash collision [7]. However, their scheme relies on the security economics rather than on computational complexity per se. An attacker with a computational power equal to the broker is able to cheat by finding a collision with given properties. In our scheme, an attacker needs to find a pre-image for a given hash.

As far as we are aware, the first usage of hash-chains for key agreement appeared in [8].

3 Collision Key Improvement

We propose a modification to the basic random key predistribution schemes. Keeping the key ring size m and the key pool size $|S|$ same as for the basic schemes, this modification additionally increases the number of keys that two nodes may share.

In our scheme, two nodes X and Y can share a key directly as in the basic schemes. Furthermore, an additional shared key C can be constructed if two nodes carry two different, but related keys K_A and K_B such that the condition $C = H(K_A) = H(K_B)$, where H is a suitable cryptographic hash function, is fulfilled. We call such related keys *colliding keys* and the resulting value C a *collision key*. Probability of two randomly chosen values for K_A and K_B being colliding keys is generally very low due to the collision resistance of the hash function. Therefore, we modify the process how keys for the key pool are selected. Instead of randomly selecting $|S|$ keys from the possible key space, $\frac{|S|}{2}$ colliding key pairs are taken to form a key pool S. Thus, the total number of keys in the key pool remains $|S|$ and the key pool gets the structure depicted in Figure 1.

$$\begin{tikzcd}
K_{A_1} \arrow[r, "H"] & C_1 & K_{B_1} \arrow[l, "H"'] \\
K_{A_2} \arrow[r, "H"] & C_2 & K_{B_2} \arrow[l, "H"'] \\
K_{A_3} \arrow[r, "H"] & C_3 & K_{B_3} \arrow[l, "H"'] \\
\vdots & \vdots & \vdots \\
K_{A_{|S|/2}} \arrow[r, "H"] & C_{|S|/2} & K_{B_{|S|/2}} \arrow[l, "H"']
\end{tikzcd}$$

Fig. 1. Key pool structure in the collision key improvement. Colliding keys from the key pool are denoted K_A and K_B. Collision keys are depicted as C. H denotes a secure hash function.

Colliding keys long enough to withstand a brute-force attack can be efficiently generated due to the birthday paradox. In Section 6, we demonstrate that for key length of $N = 80$ bits thousands of colliding key pairs can be generated with a moderate computational power.

Beside the key pool structure, we also slightly modify the way how keys are selected to a key ring. The keys are still picked from the key pool randomly without replacement, however, we do not allow two colliding keys to be in the same key ring. Thus, if a key is picked, not only itself but also its colliding counterpart is temporarily marked off the key pool. Once the key ring is complete, all keys are put back to the key pool and the next node is processed.

In the shared key discovery phase, similarly to the q-composite scheme, two nodes X and Y can establish a link key if they share at least q keys. The shared keys can be both colliding keys drawn directly from the key rings or collision keys computed using a hash function H. Since every node has m keys in its key ring, it is also able to establish m collision keys. Thus our improvement significantly increases the effective size of the key rings as evaluated in the following subsection. Therefore, we can expand the key pool accordingly while keeping the probability of a link key establishment at the desired level. This expansion increases the node capture resilience. In the rest of the paper we refer to this improvement as to the *collision key improvement*.

3.1 Probability of Link Key Establishment

In this subsection we show how to calculate the probability that any two nodes in the network are able to directly establish a link key in the shared key discovery phase. Let us define the following notation to support readability of the subsequent analysis.

Definition 1

$$\left\{ \begin{matrix} |S| \\ m \end{matrix} \right\} = \frac{|S| \cdot (|S|-2) \cdot (|S|-4) \cdot ... \cdot (|S| - 2 \cdot (m-2)) \cdot (|S| - 2 \cdot (m-1))}{m!}$$

The formula expresses the number of all possible key rings of size m selected from a key pool of size $|S|$ where no colliding key pair is present in the key rings. Thus it can be viewed as $|S|$ choose m, where the choice has to respect the above mentioned constraint. For justification see the Appendix.

The probability that any two nodes in the network share exactly i keys from the key pool S and exactly j collision keys that do not result from the i shared keys can be calculated as follows (see the Appendix for proof):

$$P_{SharedExactly}(i,j) = \frac{\binom{m}{i}\binom{m-i}{j}\left\{ \begin{matrix} |S|-2m \\ m-i-j \end{matrix} \right\}}{\left\{ \begin{matrix} |S| \\ m \end{matrix} \right\}} \tag{1}$$

Two nodes can establish a link key if they share at least q independent keys, no matter whether these are colliding or collision keys. The collision keys are counted only if their pre-images are not. Thus the probability that any two nodes in a network are able to establish a link key is, among m and $|S|$, dependent also on the parameter q and can be calculated as:

$$P_{LinkEstablishI} = \sum_{i=0}^{m} \sum_{j=0}^{m} P_{SharedExactly}(i,j) \tag{2}$$

where $i + j \geq q$ and $i + j \leq m$.

3.2 Resulting Node Capture Resilience

In this subsection we evaluate the collision key improvement with respect to the node capture resilience. The resilience is dependent on the number of captured nodes x, the key pool size $|S|$, the key ring size m and the desired probability of a link key establishment $P_{LinkEstablishI}$. We assume that an attacker has selected the nodes to capture in a random fashion and calculate the node capture resilience as

$$P_{LinkComprI} = \sum_{i=0}^{m}\sum_{j=0}^{m}(1-(1-\frac{m}{|S|})^x)^i(1-(1-\frac{2m}{|S|})^x)^j \frac{P_{SharedExactly}(i,j)}{P_{LinkEstablishI}} \quad (3)$$

where $i + j \geq q$, $i + j \leq m$. For proof see the Appendix.

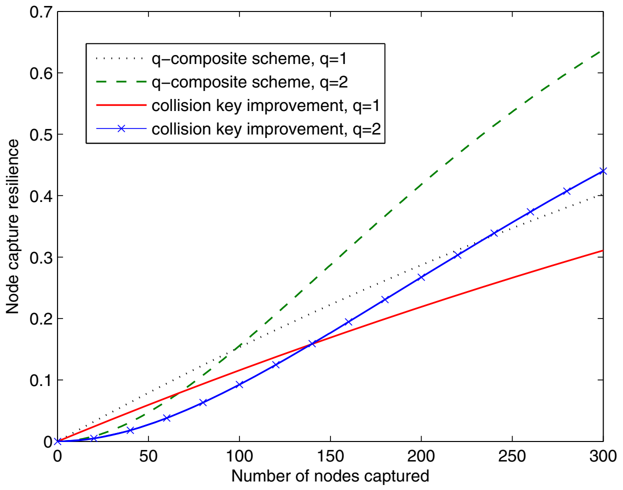

Fig. 2. Node capture resilience after x randomly selected nodes have been captured, key ring size $m = 200$, probability of link key establishment $P_{LinkEstablishI} = 0.33$

Figure 2 presents a comparison of the q-composite scheme and the collision key improvement. It is clear that our improvement provides a better node capture resilience for both values of q. E.g., if $q = 2$ and 50 nodes are captured, the resilience of the q-composite scheme is 4.7%, whereas the collision key improvement gives us the resilience of 2.7%.

4 Key-Chain Improvement

The second proposed modification of the basic random key predistribution introduces hash chains into the key pool construction. We will refer to this modification as to the *key-chain improvement*. The key-chain improvement was loosely inspired by previous work of Ren et al. [4], but our scheme utilizes hash chains in a different manner. Furthermore, we employ basic hash chains instead of the keyed ones.

In our scheme, the key pool consists of $|S|$ hash chains of a length L and every value in the chains is considered as a potential key. Thus, we refer to the hash chains as to the key-chains. The structure of the key pool is depicted in the Figure 3.

$$\begin{tikzcd}
K_{11} \arrow[r, "H"] & K_{12} \arrow[r, "H"] & \cdots \arrow[r, "H"] & K_{1L} \\
K_{21} \arrow[r, "H"] & K_{22} \arrow[r, "H"] & \cdots \arrow[r, "H"] & K_{2L} \\
K_{31} \arrow[r, "H"] & K_{32} \arrow[r, "H"] & \cdots \arrow[r, "H"] & K_{3L} \\
\vdots & \vdots & & \vdots \\
K_{|S|1} \arrow[r, "H"] & K_{|S|2} \arrow[r, "H"] & \cdots \arrow[r, "H"] & K_{|S|L}
\end{tikzcd}$$

Fig. 3. Key pool structure in the key-chain improvement. The knowledge of a key K_{ij} enables one to compute keys K_{ik} for every $k \geq j$.

In the key-setup phase, every node is randomly assigned a key from m randomly selected key-chains. If two nodes were assigned keys from the same key-chain, they are able to calculate a shared key. A node with a value closer to the beginning of the key-chain can traverse the chain downwards to find the shared key carried by the second node.

In the shared key discovery phase, two nodes can establish a link key when sharing at least q independent keys.

The actual size of the key pool is $|S| \cdot L$, although in the subsequent analysis we shall consider the number of key-chains $|S|$ as the key pool size. The length of a key-chain L shall be taken as an independent parameter that influences the scheme security. The key-chain improvement can be further combined with the collision key improvement to get even better performance. The combination is considered in Section 5.

4.1 Probability of Link Key Establishment

The probability that any two nodes share exactly i independent keys is equal to the same probability for the basic EG and q-composite schemes. To calculate the probability we can use the formula from [2]:

$$P_{SharedExactly}(i) = \frac{\binom{|S|}{i}\binom{|S|-i}{2(m-i)}\binom{2(m-i)}{m-i}}{\binom{|S|}{m}^2} \qquad (4)$$

Note that the probability is independent of the key-chain length L as the length influences only the node capture resilience provided by the scheme. The probability of a link key establishment for a given q is then

$$P_{LinkEstablishII} = \sum_{i=q}^{m} P_{SharedExactly}(i) \qquad (5)$$

4.2 Resulting Node Capture Resilience

The node capture resilience is in this case dependent (among other parameters) also on the key-chain length L. To evaluate the node capture resilience, we first calculate the probability that a key from a given key-chain is compromised after an attacker captured x random nodes as follows (see the Appendix for proof):

$$P_{ChainCompr} = \sum_{i=1}^{L} \frac{2 \cdot i - 1}{L^2}(1 - (1 - \frac{m}{|S|}\frac{i}{L})^x) \tag{6}$$

Assuming an attacker has selected the nodes to capture in a random fashion, the node capture resilience can be calculated as

$$P_{LinkComprII} = \sum_{i=q}^{m}(P_{ChainCompr})^i \frac{P_{SharedExactly}(i)}{P_{LinkEstablishII}} \tag{7}$$

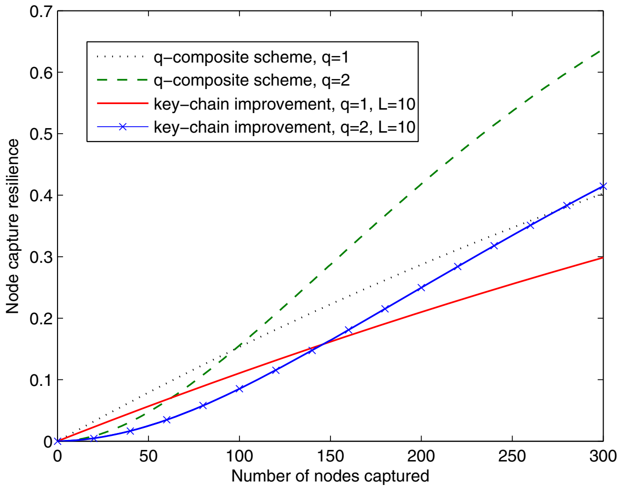

Fig. 4. Node capture resilience after x randomly selected nodes have been captured, key ring size $m = 200$, probability of link key establishment $P_{LinkEstablishII} = 0.33$, effective key-chain length $L = 10$

Figure 4 presents a comparison of the q-composite scheme and the key-chain improvement. Again, our improvement provides a better node capture resilience for both values of q. E.g., if $q = 2$ and 50 nodes are captured, the resilience of the q-composite scheme is 4.7%, whereas the key-chain improvement provides the resilience of 2.5%. Such a resilience is even better than the resilience provided by the collision key improvement proposed in Section 3.

4.3 Key-Chain Length

An important security parameter of the key-chain improvement is the length of the key-chains. It holds that the longer the key-chain, the better the node capture resilience. However, as the length of the key-chain increases, the security gain obtained for a single unit increment decreases rapidly as demonstrated in Figure 5. Also, when evaluating the node capture resilience, we have to consider the *effective length* of the key-chain, not the actual one. The effective length of a key-chain is the number of different keys from the key-chain that are actually assigned to some key ring. The effective length is dependent on the number of nodes in the network n, the size of a key ring m and the size of the key pool $|S|$. The average effective key-chain length cannot exceed $\frac{n \cdot m}{|S|}$, which is the expected number of nodes that will be assigned a key from a given key-chain. If we set the actual length to be equal to this number, the average effective key-chain length will be shorter. We can get close to the bound by setting the actual length artificially long. Yet this would increase the computational complexity of the key establishment as nodes would need to perform more hashing to establish a shared key. In practice, it is not necessary to reach the maximum length available due to the steep decrease in additional gain demonstrated in the Figure 5.

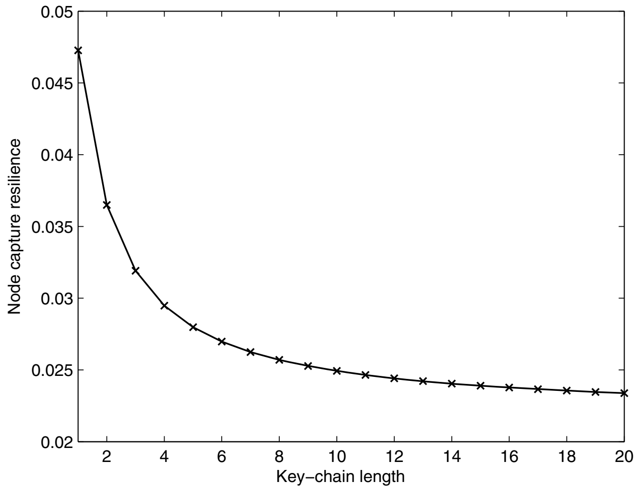

Fig. 5. Relationship between an effective length of a key-chain and node capture resilience. Key ring size $m = 200$, probability of link key establishment $P_{LinkEstablishII} = 0.33$, $q = 2$ and the number of captured nodes $x = 50$.

For most networks, a practical value of the effective key-chain length would be around $L = 10$. Such an effective length is achievable with only a slightly higher actual key-chain length for sufficiently large networks.

5 Combined Results

The key-chain improvement can be directly combined with the collision key improvement to obtain even better results with respect to the node capture resilience. To combine both improvements, the end points of the key chains should be the colliding keys. This requirement can be easily fulfilled due to the nature of the parallel collision search algorithm. If the collision is found, also the two hash chains that precede it are obtained, see Section 6.

The *probabilities of a link key establishment* between any two nodes in the network are calculated similarly as in the case of the collision key improvement using Equations 1 and 2. The size of the key pool $|S|$ is in this case defined as the number of key-chains. Thus $|S|$ has a similar meaning as in the key-chain improvement.

For given arguments, one obtains the same probability of a link key establishment for both the collision key improvement and for the combined solution. Yet there is a difference in the achieved node capture resilience, which is higher for the combined solution. The difference is dependent on the effective length L of the key-chains.

The *node capture resilience* of the combined scheme can be calculated as

$$P_{LinkComprIII} = \sum_{i=0}^{m}\sum_{j=0}^{m}(P_{ChainCompr})^{i}(1-(1-\frac{2m}{|S|})^{x})^{j}\frac{P_{SharedExactly}(i,j)}{P_{LinkEstablishI}} \quad (8)$$

where $i+j \geq q$, $i+j \leq m$.

Figure 6 demonstrates that the combined improvement outperforms the q-composite scheme and the key-chain improvement. E.g., if $q = 2$ and 50 nodes are captured, q-composite scheme scores 4.7%, collision key improvement 2.7%, key-chain improvement 2.5% and the combined improvement 2.2%. The comparison gets even better for the combined improvement as the number of captured nodes grows.

The scheme of Ren et al. [4] provides a slightly better node capture resilience than our combined improvement for a small number of captured nodes. However, as this number grows our combined improvement starts to outperform the Ren's scheme. For $P_{LinkEstablishI} = 0.5$, $q = 2$ and $m = 161$, the turning point is around 125 of captured nodes. Since we were not able to fully reproduce Ren's analytical results (and did not get any response from the contacted authors), we did the comparison only for the parameters used in their paper.

The communication overhead of the shared-key discovery phase, when our combined improvement is used, is dependent on the discovery procedure itself. For some procedures it is similar to the overhead of the basic random key predistribution schemes. E.g., if the pseudo-random predistribution [9] is used, identifiers of keys assigned to a particular node can be calculated from the node ID. These identifiers can carry all the information necessary to discover shared keys – the key's position in a hash-chain and the hash-chain identifier. Additionally, the shared collision key can be figured out through the hash-chain identifiers when these identifiers (assigned to the colliding hash-chains) differ only in the

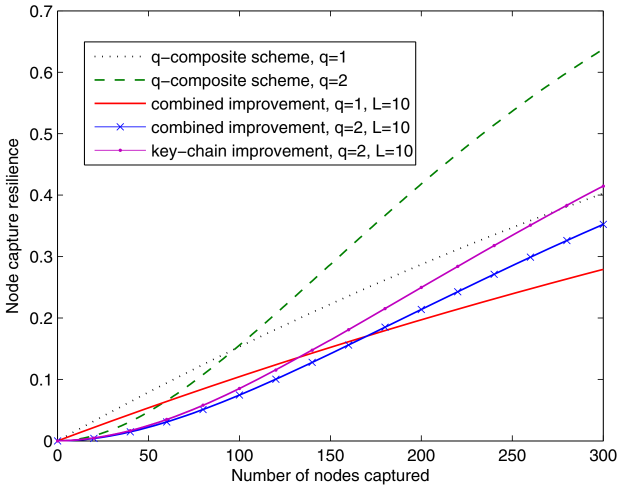

Fig. 6. Node capture resilience of the combined improvement. Key ring size $m = 200$, probability of link key establishment $P_{LinkEstablishI} = 0.33$, effective key-chain length $L = 10$.

Table 1. Probabilities that two nodes share exactly i keys from hash chains and j additional collision keys. Value i is dependent on the row and value j on the column of the table. $P_{LinkEstablishI} = 0.33$, $q = 1$, $m = 200$.

$i \backslash j$	0	1	2	3
0	0.67	0.1344	0.0134	0.0009
1	0.1344	0.0267	0.0026	0.0002
2	0.0134	0.0026	0.0003	0
3	0.0009	0.0002	0	0

least significant bit. Thus the communication overhead only covers transmission of the node IDs. Another advantage of the pseudo-random approach is that the nodes do not need to store their own key identifiers as these can be computed when actually needed.

Another interesting information concerns the composition of link keys established, e.g., what fraction of keys is based solely on the collision keys or solely on the keys from the hash chains. Such information can be calculated using Equation 1. The equation gives us the probability that two nodes share exactly i keys from hash chains and j additional collision keys. The sample probabilities for $P_{LinkEstablishI} = 0.33$, $q = 1$, $m = 200$ and different combinations of i and j are summarized in Table 1. E.g., the probability that a link key is based on exactly two keys from hash chains and a single collision key is given in the row 2, column 1. The probability that two nodes do not share any key is in the row 0, column 0. Note that the table is symmetric, i.e., both types of keys are used with an equal probability. The table is not complete, yet the probabilities of other combinations of i and j are negligible.

6 Computational Results

Analytical results presented in the previous sections were computationally verified using our network simulator. We have simulated the q-composite scheme and all the proposed improvements using various settings for critical parameters and networks of different sizes and topologies. For every setting, an average over 10 different simulation runs was taken as a result. The reference simulated network had 10,000 nodes, though we have tested also other sizes. It showed that obtained analytical and simulated results for node capture resilience are consistent. The simulator and its source codes are available for download along with sample configuration scripts that enable the verification of results[1].

The important part of the collision key and the combined improvement is a search for collisions of the cryptographic hash function. This search can be efficiently performed due to the birthday paradox. In order to find an N-bit collision in a cryptographic hash function, one needs to perform approximately $2^{\frac{N}{2}}$ hashing operations. Furthermore, to find c^2 such collisions for $1 \leq c \leq 2^{\frac{N}{2}}$, one needs to perform "only" approximately $c \cdot 2^{\frac{N}{2}}$ hashing operations [7]. Thus, once the first collision is found, additional collisions can be found increasingly efficiently. If we assume 80-bit keys are used, to create the key pool for $|S| = 2^{16}$ one needs to find 2^{15} collisions which requires approximately $2^{47.5}$ hashing operations. This can be reached with a moderate computational power. Note that 80-bit keys can be still considered as secure and appropriate for use in wireless sensor networks as attacker needs to try approximately 2^{79} values to brute force the key.

We have conducted our collision search using the parallel collision search method proposed by van Oorschot and Wiener [10]. This method is based on Hellman's time-memory trade-off approach and calculates long hash-chains. We have searched for 80-bit collisions of the SHA-256 hash function, 80-bit values were taken as an input and 80 most significant bits of the SHA-256 function as an output. We used the Gladman's implementation[2] of the hash function. The aggregate time to find over 5,000 suitable collisions was approximately 19,000 hours on a single 3GHz CPU core. The search was distributed using the BOINC infrastructure [11] to around 1,000 CPU cores so the search took less than a day. Approximately 2^{23} hash chains with an average length of 2^{24} were computed, thus about 2^{47} hashing operations were performed. The time spent and resources invested are moderate and within reasonable bounds since this procedure takes place only once in a network lifetime. The speed of the search could be significantly increased using GPUs or special purpose hardware like FPGA.

7 Conclusions

The key distribution stands among the most critical security issues for wireless sensor networks. In this paper, we proposed and analyzed two improvements

[1] http://www.fi.muni.cz/~xsvenda/papers/SecureComm2012/

[2] http://gladman.plushost.co.uk/oldsite/cryptography_technology/sha/index.php

(and their combination) of the random key predistribution schemes. The first improvement exploits limited length collisions in secure hash functions to increase the probability of two nodes sharing a key. The second improvement introduces hash chains into the key pool construction to directly enhance the node capture resilience. Both these improvements can be further combined to bring the best performance. Our analytical results were supported by simulations.

Our improvements are particularly advantageous for networks where the attacker manages to capture a significant number of nodes. However, the benefits of our improvements are not limited to the basic random key predistribution schemes. The improvements could be employed, e.g., in the deterministic or hybrid approach proposed in [5]. We leave the investigation of such combination for the future work. Another challenge is to analyze the improvements face to face with a clever attacker who does not capture the nodes in a random fashion. Yet the impact of such an attacker could be limited by a deterministic selection of keys to be placed into key rings.

Acknowledgements. We are grateful to the anonymous reviewers and to Luděk Smolík for their suggestions that improved the paper and to Tobiáš Smolka for his help with the BOINC infrastructure. This work was supported by the Czech research project VG20102014031, programme BV II/2 - VS. The first author is additionally supported by the project GD102/09/H042 "Mathematical and Engineering Approaches to Developing Reliable and Secure Concurrent and Distributed Computer Systems" of the Czech Science Foundation. The second and third authors are also supported by the research center Institute for Theoretical Computer Science (ITI), project No. P202/12/G061. The access to computing and storage facilities owned by parties and projects contributing to the National Grid Infrastructure MetaCentrum, provided under the programme "Projects of Large Infrastructure for Research, Development, and Innovations" (LM2010005) is highly appreciated/acknowledged.

References

1. Eschenauer, L., Gligor, V.D.: A key-management scheme for distributed sensor networks. In: 9th ACM Conference on Computer and Communications Security, CCS 2002, pp. 41–47. ACM, New York (2002)
2. Chan, H., Perrig, A., Song, D.: Random key predistribution schemes for sensor networks. In: Symposium on Security and Privacy, pp. 197–213. IEEE (2003)
3. Švenda, P., Sekanina, L., Matyáš, V.: Evolutionary design of secrecy amplification protocols for wireless sensor networks. In: 2nd ACM Conference on Wireless Network Security, pp. 225–236. ACM, New York (2009)
4. Ren, K., Zeng, K., Lou, W.: A new approach for random key pre-distribution in large-scale wireless sensor networks. Wireless Communications and Mobile Computing 6(3), 307–318 (2006)
5. Camtepe, S.A., Yener, B.: Combinatorial design of key distribution mechanisms for wireless sensor networks. IEEE/ACM Transactions on Networking 15(2), 346–358 (2007)

6. Xiao, Y., Rayi, V.K., Sun, B., Du, X., Hu, F., Galloway, M.: A survey of key management schemes in wireless sensor networks. Computer Communications 30(11-12), 2314–2341 (2007)
7. Rivest, R.L., Shamir, A.: PayWord and MicroMint: Two Simple Micropayment Schemes. In: Crispo, B. (ed.) Security Protocols 1996. LNCS, vol. 1189, pp. 69–87. Springer, Heidelberg (1997)
8. Leighton, T., Micali, S.: Secret-Key Agreement without Public-Key Cryptography. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 456–479. Springer, Heidelberg (1994)
9. Di Pietro, R., Mancini, L.V., Mei, A.: Random key-assignment for secure wireless sensor networks. In: 1st ACM Workshop on Security of Ad Hoc and Sensor Networks (SANS 2003), pp. 62–71. ACM, New York (2003)
10. van Oorschot, P.C., Wiener, M.J.: Parallel collision search with cryptanalytic applications. Journal of Cryptology 12(1), 1–28 (1999)
11. Anderson, D.P.: BOINC: A system for public-resource computing and storage. In: 5th IEEE/ACM International Workshop on Grid Computing, pp. 4–10. IEEE Computer Society (2004)

Appendix: Proofs and Calculations

In this appendix we provide proofs of the selected equations and also justify the following statement that relates to the Definition 1. The formula in Definition 1 expresses the number of all possible key rings of size m selected from a key pool of size $|S|$ where no colliding key pair is present in the key rings.

Proof. We have $|S|$ possibilities how to select the first key for a key ring. After this selection, we mark the selected key and its colliding key off the key pool. Thus we have only $|S| - 2$ possibilities how to select the second key. The keys are selected in this fashion until we select the m-th key for which only $|S| - 2 \cdot (m - 1)$ possibilities remain. Since the order in which the keys were selected is not important, we divide the result by the number of permutations $m!$. $\square$

Proof (Equation 1). We have $\left\{ {|S| \atop m} \right\}$ possibilities how to select m keys into a key ring for any given node. Given these m keys, we have $\binom{m}{i}$ ways to select the i shared keys. Similarly, once these i shared keys have been picked, we have $\binom{m-i}{j}$ ways to select j shared collision keys that do not result from the i shared keys. Finally, we have to pick the remaining $m-i-j$ keys for the second key ring that are not the keys from the first key ring nor their colliding counterparts. Hence we pick them from the key pool without m colliding key pairs ($2m$ keys). This can be done by $\left\{ {|S| - 2m \atop m - i - j} \right\}$ ways. Thus the number of key ring assignments for two nodes such that they share exactly i keys and are able to calculate exactly j collision keys excluding the collision keys resulting from the i shared keys is $\left\{ {|S| \atop m} \right\} \binom{m}{i} \binom{m-i}{j} \left\{ {|S| - 2m \atop m - i - j} \right\}$. The total number of key ring assignments for any two nodes is $\left\{ {|S| \atop m} \right\}^2$. Thus the resulting probability is the fraction of these two values. $\square$

Proof (Equation 3). We follow and extend the proof from [2]. Since every node contains m keys out of $|S|$, the probability that an attacker obtains a particular key after a single node is captured is $\frac{m}{|S|}$. The probability that the attacker does not obtain the particular key after x nodes have been captured is thus $(1-\frac{m}{|S|})^x$. Finally, the probability that the attacker compromises a link key that is based on exactly i shared keys is $(1-(1-\frac{m}{|S|})^x)^i$.

Similarly, the probability that the attacker obtains a particular collision key after a single node is captured is $\frac{2m}{|S|}$, because we have only $\frac{|S|}{2}$ distinct collision keys and every node is able to calculate exactly m such keys. Hence, the probability that the attacker compromises a link key based on exactly j collision keys is $(1-(1-\frac{2m}{|S|})^x)^j$.

Assuming a link is secured with a link key, the probability that the link key is based on exactly i shared keys and j collision keys is $\frac{P_{SharedExactly}(i,j)}{P_{LinkEstablishI}}$. □

Proof (Equation 6). Assume that two nodes were assigned keys from a given key-chain, then the probability that they establish i-th key of the key-chain as a shared key is $\frac{2\cdot i-1}{L^2}$. Furthermore, the probability that i-th key of a given key-chain was compromised after a random node was captured is $\frac{m}{|S|}\frac{i}{L}$. The probability that an attacker has compromised i-th key of a given key-chain after he captured x nodes is $1-(1-\frac{m}{|S|}\frac{i}{L})^x$. □