

CAB: A Cellular Automata-Based Key Management Scheme for Wireless Sensor Networks

Amin Y. Teymorian, Liran Ma, Xiuzhen Cheng

Department of Computer Science
George Washington University,
Washington DC, 20052, USA.

Email: {amin,lrma,cheng}@gwu.edu

Abstract— We develop a cellular automata (CA) based key management scheme for wireless sensor networks termed CAB. Our proposed scheme allows sensors to establish pairwise keys during any stage of the network operation using pre-loaded CAs. Additionally, CAB has the following nice properties: i) it is computationally efficient because operations can be as simple as bitwise OR and XOR; ii) it achieves quasi-perfect resilience against node compromise because the computed pairwise keys are unique with high probability; iii) it is the first scheme that inherently provides rekeying capabilities.

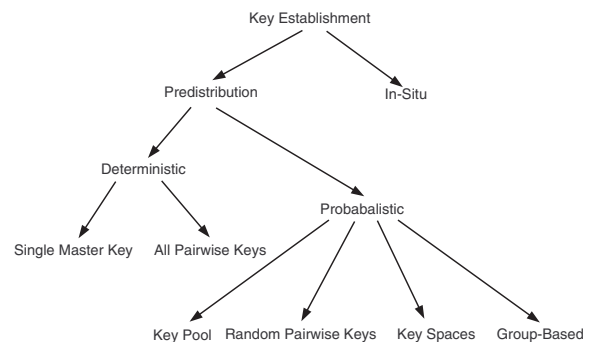


Fig. 1. A taxonomy of existing key establishment schemes.

I. INTRODUCTION

Secure communication is a required property for the wide-spread deployment of wireless sensor networks (WSNs). A fundamental prerequisite for secure communication is the existence of a pairwise secret key between neighboring sensors at any phase of network operation. These phases include initial sensor deployment, sensor elimination (e.g., resource depletion or node compromise), and future sensor deployment. In addition, the pairwise keys must be established under extreme sensor resource constraints (limited battery supply, CPU, memory, etc.) and harsh deployment environments. Such strong demands make key management a non-trivial problem.

One approach to solving this problem is to pre-distribute keys to each sensor prior to deployment. The pioneering work on key pre-distribution was carried out by Eschenauer and Gligor [1]. After the introduction of this concept, a variety of other key establishment schemes [2]–[7] have been reported, as illustrated in Fig. 1. In these solutions, keys or keying information (e.g., shares of symmetric key matrix [3] or symmetric bivariate polynomial [4]) are pre-loaded into each sensor prior to deployment. Then, neighboring sensors

establish shared keys after deployment through partial information exchanges. For this reason, these schemes are categorized as key pre-distribution schemes.

All key pre-distribution schemes must cope with the unpredictability of the network topology prior to deployment. Thus, a key pre-distribution scheme requires extra keying information to be pre-loaded in order to achieve desirable key-sharing probability between neighboring sensors. As a side effect, part of the keying information may not be utilized during the network lifetime. Further, this uncertainty can degrade the scalability of key pre-distribution schemes.

To achieve better scalability, three in-situ key establishment schemes [8]–[10] have been proposed. Instead of pre-loading key information, sensors compute shared keys with their neighbors after deployment. Subsequently, the schemes all scale well with the size of the network, and each of them can obtain a highly connected key-sharing graph with low storage overhead. However, a drawback of these schemes is that the shared key computation consumes a lot of energy compared to key pre-distribution schemes that do not employ random key spaces [1], [2].

Despite the contributions of key pre-distribution schemes and in-situ schemes, performing the rekeying

activities that cope with sensor elimination and support future sensor deployments is still an unsolved problem. To that end, we propose CAB, the first cellular automata (CA) based key management scheme for wireless sensor networks. A CA consists of a grid of cells and a rule function. Each cell can be in one of a fine number of states, and is updated in an iterative fashion according to a CA rule function. In CAB, a number of CAs are preloaded into each sensor. Once deployed, neighboring sensors can compute a pairwise key using the cell values of a shared CA. The proposed scheme has the following nice properties: i) it is computationally efficient because the operations involved can be as simple as bitwise OR and XOR; ii) it can achieve quasi-perfect resilience against node compromise in that all pairwise keys are unique with high probability; iii) it is the first scheme that inherently supports rekeying and phased deployment capabilities.

The remaining portion of this paper is organized as follows. Section II provides a brief overview of related research. In Section III, assumptions and background knowledge are presented. Our novel cellular automata-based key pre-distribution scheme is proposed in Section IV, and its performance is analyzed in Section V. Finally, we conclude our paper and discuss future extensions in Section VI.

II. RELATED WORK

Due to its computational efficiency, symmetric key based security mechanisms have received much attention in the context of WSNs. For example, Eschenauer and Gligor [1] proposed the basic *random keys scheme* for key pre-distribution in WSNs. This scheme consists of three phases: key pre-distribution, shared-key discovery, and path-key establishment. In the first phase, a large key pool is computed offline and each sensor is preloaded with keys selected randomly without replacement from the key pool. These keys form a sensor's key ring. A pair of sensors can establish a secure communication channel as long as their key rings have at least one key in common. If there is no common key, a path key needs to be established with the help of an intermediary node that shares a key with each node in the pair.

This pioneering work described above has inspired many followup proposals. For example, an enhanced scheme is presented in [2] that requires $q > 1$ number of common keys for two nodes to establish a shared key. However, because the keys used by key pre-distribution schemes are typically known to more than one pair of nodes, the resiliency of these schemes degrades dramatically as nodes become compromised. Moreover, they also struggle with scalability problems when faced with

sensor memory constraints. This is due to the number of keys that need to be stored to achieve adequate network connectivity when no topology information is available prior to node deployment. Other schemes have been developed to further address these issues.

In order to improve resilience to node compromise attacks, two *random key space schemes* [3], [4] have been proposed. These two schemes are very similar in nature; the main difference is in the way the key spaces are defined (i.e., symmetric matrices and symmetric polynomials, respectively). Sensors pre-load multiple pieces of keying information, each of which belongs to a particular key space. Two sensors can compute a shared key if they have keying information from the same key space. However, the process involves expensive modular multiplications. To further improve the resilience of the random key schemes against node compromise, PIKE [11] has been proposed. In this scheme, unique pairwise keys are used. If two nodes do not share a key, they use a trusted intermediary to facilitate the key establishment process between them. This communication intensive event occurs frequently because the key sharing probability of neighboring sensors is not satisfactory.

On the other hand, to address scalability issues, the *group-based schemes* [5]–[7] are proposed. Du *et al.* [5] associate each group of sensors with a portion of the overall key pool. Sub-key pools overlap if the corresponding groups have adjacent deployment points. In [6] and [7], sensors are grouped based on IDs, and nodes with the same deployment group or the same cross group are preloaded with pairwise keys. These two schemes inherit many of the nice features of [5], but release the strong topology assumption that it adopts. The tradeoff for this flexibility exists in the form of higher communication overhead when two neighboring sensors try to establish a path key. Another approach to improving scalability is to employ location information. Hence, location-aware key establishment schemes such as [12] and [13] are developed. The two schemes rely on preloaded unique pairwise keys between neighboring nodes, and identity-based cryptography [14], respectively.

In addition to improving resilience and scalability, in-situ key establishment is introduced in [8]–[10] to enhance the local neighborhood key sharing probability. These schemes enable sensors to compute shared keys with their neighbors after deployment, hence obviating the need to pre-load keying information. Subsequently, all of the in-situ schemes can obtain high local connectivity in the key-sharing graph while maintaining low storage overhead. One drawback of these schemes is that the shared key computation consumes a lot of

energy compared to most key pre-distribution schemes (excluding the random key space schemes [3], [4]). Also, with the assistance of a base station, schemes such as SPINS [15] and LEAP [16] may outperform in-situ key distribution schemes in terms of energy consumption. Nevertheless, the assumption of a trusted always-on basestation does not always hold.

In addition to these keying mechanisms, TinySec [17] is the first fully implemented link layer security architecture for sensor networks. It seeks a balance between different cryptographic primitives and the inherent limitations of wireless sensor networks.

CAB is different from the work mentioned above in that it is the first scheme that supports rekeying activities. In CAB, a small number of CAs are pre-loaded into each sensor. The CAs allow generation of highly random pairwise keys between neighboring sensors using operations as simple as bitwise OR and XOR. Moreover, since different key generation methods are used by each pair of sensors, the scheme enables the establishment of quasi-all-pairwise unique keys. Subsequently, CAB achieves a quasi-perfect resilience to node capture attacks that is unavailable to schemes based on pre-loaded shares of a global key pool. More importantly, keys can be computed on demand by sensors that have a common CA. This feature allows CAB to provide inherent support for rekeying activities.

III. PRELIMINARIES

A. Cellular Automata

A cellular automaton (CA) is a discrete structure consisting of a grid of cells, each of which can be in one of a finite number of states. The “grid” is typically 1-dimensional (a line) or 2-dimensional (a matrix), and the state values are usually 0 and 1, or white and black. All cells are updated together in an iterative fashion. The updates are made according to a rule (e.g., Eq. (1) in Subsection IV-B) or a set of rules based on the previous state of a cell and the states of its neighbors. The process of producing successive generations of the grid by updating its cells is called *evolution*.

The evolution of a CA is influenced by its initialization and rules. There are multiple ways to initialize the cells that comprise a CA. One method is to select a few cells near the center of the grid. Alternatively, each cell can be initialized randomly. The combination of these factors impact the randomness of the values that a CA outputs. As a CA evolves, the same rule can be applied uniformly to all cells, or different rules can be applied to different cells. For an expansive collection of results on CAs, we refer the readers to [18].

B. Network Model and Security Assumptions

We consider a large-scale homogeneous sensor network whose nodes are randomly distributed over a region. There is no neighborhood information available to any sensor before deployment. Therefore, a sensor discovers its neighbors and their CA information via local wireless broadcast after deployment.

The broadcast feature of wireless communication allows adversaries to perform a variety of passive and active attacks. In passive listening mode, adversaries silently listen to radio transmissions in order to capture data, security credentials, or other relevant information. For active attacks, adversaries may insert, modify, replay, or delete traffic, or jam part of the network. As a result, adversaries are capable of performing attacks that include session hijacking and man-in-the-middle attacks.

Adversaries equipped with powerful communication devices may access any spot of the network from a remote location. However, they cannot monitor the entire deployment region simultaneously at all times. They can gain mobility through the use of robotics or vehicles, and can move inside or outside the network. Also, adversaries can deploy their own sensors and base stations in uncontrolled wireless environments. Further, they are able to capture, replace, compromise, and physically damage existing sensors.

Since we assume that sensors are not tamper resistant, the compromise or capture of a sensor releases all of its security information to the attacker. This information, as explained in subsection IV-A, consists of several uniquely identified CAs from a global collection of CAs used by our scheme. The CAs used should be capable of generating high quality randomness. Possible methods to obtain these CAs are described in [19] and [20].

IV. A CELLULAR AUTOMATA-BASED KEY MANAGEMENT SCHEME

In this section, we present the basic features of our cellular automata-based key management scheme, deferring its analysis to the next section. CAB consists of three phases: *CA pre-loading*, *shared key derivation*, and *rekeying*. In the first phase, a collection of CAs are generated and loaded into each sensor. The second phase allows neighboring sensors to compute a shared key using a shared CA. Lastly, the rekeying process allows current shared keys to be refreshed, and pairwise keys to be established between sensors that are deployed during different phases.

A. CA Pre-Loading

The CA¹ pre-loading phase of our scheme consists of three offline steps. The steps are as follows:

- 1) Generate a large collection of CAs (e.g., about 2^{17} - 2^{20} CAs) denoted as C and a unique identifier for each CA. The only requirement for these CAs is that they be capable of producing output that exhibits high quality randomness. Since the collection of CAs is necessarily large, different types of CAs (e.g., 1-dimensional, 2-dimensional, etc.) may be needed to form the entire CA collection.
- 2) Randomly draw r CAs without replacement from the collection C to establish the subset of preloaded CAs that are assigned to a sensor. If multiple types of CAs, say m , are available, the CA selection process needs to be carefully performed so that the set of r CAs drawn by each sensor have a similar total size. One easy solution is to regard the collection of CAs as being comprised of m subpools. Then, each sensor can be randomly assigned at least $\lfloor r/m \rfloor$ CAs from each subpool.
- 3) Load the CAs with their identifiers into the memory of each sensor.

As explained in Subsection V-A, the CA pre-loading phase ensures that only a small number of CAs need to be pre-loaded into each sensor to ensure that any two sensors share (at least) one CA with a specified probability.

B. CA-Based Pairwise Session Key Computation

The shared key derivation phase consists of two steps, namely CA information exchange and key computation. After deployment, sensors exchange CA information with their immediate (i.e., 1-hop) neighbors. The simplest way to accomplish this is to have each sensor perform a local broadcast of its list of CA identifiers.

Note that this approach does not give an adversary any attack opportunity that she does not already have because the keys are never transmitted. Furthermore, when an adversary captures a node, she can only obtain the CAs stored in the memory of the node. Other node pairs that use the same CA likely selected a different initialization and number of rounds. Hence, the adversary will not necessarily know other keys computed with the same CA.

After obtaining a list of the CAs possessed by neighboring nodes, two sensors select a CA that they have

¹To keep definitions concise, the terms ‘‘CA’’ or ‘‘CA rule’’ may be used interchangeably throughout this paper. For example, pre-loading a CA is equivalent to pre-loading a CA rule.

in common. Then, the sensors agree on the values of two parameters used to configure their common CA. Specifically, the sensors choose an initial state for the CA (e.g., a single cell in state 1) and the number of rounds to apply the rule. Together with these parameters, the rule is applied to generate the session key. Note that the incorporation of additional parameters in computing a session key differs from selecting a key based on some partial information (e.g., a key ID) as done in other pre-distribution work [1], [2].

One example rule for a CA is given by the following equation:

$$a'_i = a_{i-1} \oplus (a_i \vee a_{i+1}). \quad (1)$$

This rule computes the new value of cell a_i (denoted as a'_i) to be the XOR of the cell’s left neighbor, a_{i-1} , with the OR of the current value of the cell and its right neighbor, a_{i+1} . With the initial state of its center cell set to 1 and other neighboring cells set to 0, the output of a CA that uniformly applies the above rule is shown in Fig. 2. The values that make up the center column of cells are believed to be highly random [21], and are hence used to compose a shared key.

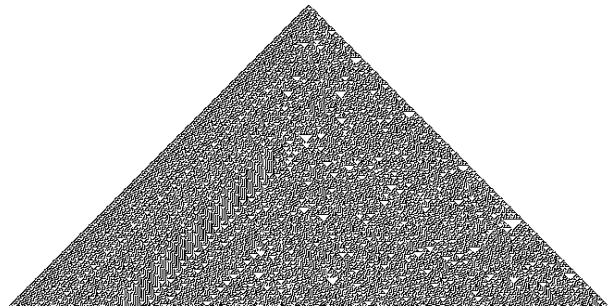


Fig. 2. The output of a CA that has its center cell initialized as 1, and uniformly applies the rule in Eq. (1) for 500 iterations.

As a result, the highly random CA output coupled with the independently agreed upon initialization parameters make it unlikely that multiple pairs of nodes will share the same key. Thus, CAB is able to achieve quasi-all-pairwise unique keys between each pair of sensors.

In the case of neighboring nodes not having a common CA, they can use a path key establishment phase to have a common neighbor transmit key related information. The only constraint on the intermediate node is that it must already share a key with both of neighboring nodes that desire to establish a key. The path-key establishment procedure is widely adopted in key pre-distribution schemes. Hence, we omit an explanation of this step, and refer the interested reader to any of the other works that contain this phase (e.g., [1]–[4]).

C. Rekeying

Nodes may decide to rekey for multiple reasons, e.g., possible key disclosure, prudent key refreshment, or newly deployed sensors. In CAB, the high level of randomness available from CAs forms the foundation of the rekeying process. It allows nodes to rekey at any time during the lifetime of the network, and independent of any previous rekeying phase. This property is critical to the reliability and longevity of networks comprised of resource-constrained devices. For example, sensors with depleted batteries will reduce global connectivity unless replaced. Another nice feature of CAB’s rekeying capability is the flexibility that it provides to the network controller. Specifically, the number of phases (or “generations”) of node deployments does not need to be known prior to the initial sensor deployment.

In order to rekey, two nodes agree on a new initial configuration for their shared CA and the number of iterations that the CA rule will be applied. The decision as to which node will initiate the process is arbitrary. If rekeying activity happens between two nodes who already share a key (in the case of key refreshment), the rekeying parameters agreed upon by the nodes can be encrypted for added protection. The to-be-refreshed key can be used to secure transmission of these parameters. In the case of phase deployment, the new sensors do not share keys with existing sensors. Hence, the rekeying parameters are transmitted in plaintext. As explained in Subsection V-D, this does not necessarily cause exposure of the established key. Since the newly deployed sensors are equipped with CAs drawn from the same collection of CAs as previously deployed sensors, they may perform session key establishment.

One may argue that one-way functions could be used as an alternative to the CAs in the rekeying process. However, CAs are a more appropriate choice for the following reasons.

- Many of these functions and associated applications (e.g., one-way hash chains) require expensive instructions such as modular multiplications. Thus, the implementation overhead required for these functions may be unsuitable for resource-constrained devices. We note that a recent attempt at addressing this inefficiency is made in [22].
- One-way functions are also compression functions, i.e. they map a larger arbitrary size input to a smaller fixed-size output. CAs exhibit behavior that is the exact opposite of this. Specifically, CAs have the ability to generate an arbitrarily long (and variable) length keys from a few initial bits.

V. EVALUATION

In this section, we evaluate CAB in terms of key sharing probability, resilience to node capture, computational overhead, communication expense, storage overhead, and rekeying ability. Additionally, we discuss some unique features of CAB (e.g., the connectivity-to-storage ratio), and their relationship to the overall performance of the scheme. All computations are performed using Mathematica.

A. Key Sharing Probability

Key establishment in CAB relies on the probability that two subsets of a collection of CAs overlap. Further, by assuming that the network topology is a random graph, we can analyze our global connectivity using random graph theory [23]. Hence, we obtain key sharing probabilities and connectivity results that are consistent with other pre-distribution schemes based on key sharing between nodes of a random graph (e.g., [1], [2], etc.). For example, with a collection of $C = 10,000$ CAs and preloading each sensor with $r = 150$ CAs, the probability of two nodes sharing at least 1 CA is $p = 1 - \frac{((C-r)!)^2}{(C-2r)!C!} \approx 90\%$. Note that even though the quantity of information (i.e., the number of keys or CAs) stored in each sensor’s memory may be the same, the storage overhead of each may be different.

B. Storage Requirement

The storage overhead of CAB is determined by the memory space needed for the preloaded CAs. Since the size of each CA rule is small and there is limited number of rules loaded, required storage can be significantly less than the space needed for storing a key. For example, consider the rule shown in Eq. (1). The next state of any particular cell is determined by the current value of the cell, and the values of its left and right neighbors. Since each cell is in one of two possible states (0 or 1), there are $2^3 = 8$ combinations of state values that dictate a cell’s next state. As illustrated in Table I, the mapping from current state combination $a_{i-1}a_i a_{i+1}$ to next state a'_i of the rule in Eq. (1) can be easily represented with 1 byte.

Current	000	001	010	011	100	101	110	111
Next	0	1	1	1	1	0	0	0

TABLE I
MAPPING OF CURRENT STATE COMBINATION TO NEXT STATE FOR
THE RULE IN EQ. (1).

If the required key sharing probability is 90%, about 150 CAs need to be loaded (according to Subsection V-A). As a result, the total memory consumption is 150 bytes.² It is worth of mentioning that CAB achieves

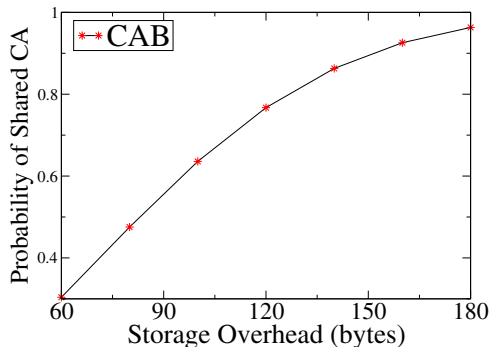


Fig. 3. Storage overhead vs. probability of sharing at least one CA (assuming 8-bit rules).

a higher efficiency of connectivity under same storage quota compared to key pre-distribution schemes. That is, CAB is able to employ a small number of bits of storage to generate a key, e.g., an 8-bit rule can be used to generate a 128-bit key. In other pre-distribution schemes (e.g., [1], [2], [5], [11], [12]), the amount of memory needed to store a piece of keying information is equivalent to the key length. Fig. 3 gives a taste of CAB’s efficiency of storage with respect to local key derivation probability.

C. Computation and Communication Overheads

Depending on the rules used by neighbor nodes, the computation needed to derive keys can be as little as a single bitwise OR and XOR operation (e.g., the rule in Eq. (1)). This expense is reasonable compared to schemes using similar keying information (e.g., [1] and [2]), and significantly lower than modular computation needed for schemes using symmetric key spaces [3], [4], [8]–[10]. Table II summarizes the main sources of computational overhead for these schemes.

Although each sensor’s keying information (i.e., the CA rules) is different than that used in the basic scheme [1], the communication overhead remains similar. This is attributed to there being only a list of identifiers are exchanged. Additionally, the cost of transmitting the CA initial state and the number of iterations is minor compared to the list of CA IDs. Even with this expense, CAB still achieves better communication efficiency than

²All of the CAs may not be able to be represented by 1 byte. However, careful off-line generation of CAs can produce CA rules that meet the space requirements of a given application.

Scheme	Source
Basic [1]	none
Random Pairwise [2]	none
CAB (under study)	OR and XOR
q -Composite [2]	q XORs
Symmetric Key Space [3], [4], [8]–[10]	modular multiplications

TABLE II
SUMMARY OF COMPUTATIONAL OVERHEAD SOURCES.

in-situ schemes such as iPAK [8], SBK [9], and LKE [10] whose communication involves transmitting the share of a symmetric matrix or a symmetric bivariate polynomial.

D. Resilience

In key pre-distribution schemes such as [1], keys revealed to the adversary from compromised nodes may overlap with those of uncompromised nodes. Therefore, the communication between two uncompromised nodes can become insecure. CAB inherently achieves higher resilience because sensors share the key generation methods (i.e., CAs) not keys. The high randomness of the CAs employed by CAB allow sensors to efficiently compute quasi-pairwise-unique keys, which enables CAB to offer the unique property that we term as quasi-perfect resilience.

To be specific, if an adversary captures some sensors, the keys used by other sensors will not be disclosed. That is, even though the CAs used by two uncompromised sensors may be known to the adversary, the key used by them is not automatically revealed. Specifically, the key shared by the uncompromised nodes is derived using an initial state and number of iterations agreed upon independently of other nodes, including the compromised ones.

Similarly, CAB’s resilience is better than the random key space schemes [3], [4]. For these schemes, once an adversary captures more than λ nodes with a share of particular space, the entire key space is broken. Subsequently, the communication between uncompromised nodes (using shares of the broken key space) becomes compromised.

E. Rekeying

CAB is the only scheme that naturally comes with a simple and efficient rekeying solution. Since computing a new key is low cost and convenient it can be done frequently with little overhead. Further, no centralized server is necessary for assigning new keys, and sensors may locally refresh or initiate a key when needed. Note

that this differs from previous attempts at rekeying that involve a base station, such as the scheme in [1].

Additionally, the rekeying information (e.g., the initial configuration or number of rounds) does not necessarily have to be protected by the current shared key. In fact, this can be an operational decision made by sensors in an ad-hoc manner. As long as the CA used by the two nodes has not been compromised, a passive listener cannot conclude the new key based on any eavesdropped rekey message. However, since the encryption process incurs a minimal amount of overhead (i.e., the cost of a simple XOR operation), we suggest sensors protect the rekeying parameters with their current shared key if they have one. Therefore, even if the CA has been compromised, but the initial state and/or the number of iterations of the CA rule have been undetected by the adversary, the keying parameters remain secure.

VI. SUMMARY AND FUTURE WORK

In this paper we present a cellular automata based key management scheme for wireless sensor networks, whose task is to setup pairwise keys among neighboring sensors at various operational stages of a sensor network. In this scheme, a number of CA rules are preloaded into sensors. A sensor computes pairwise keys with its neighbors after deployment by applying some initial parameters to their shared CAs. The proposed scheme is shown to possess the following nice properties: i) it is the first scheme that inherently provides rekeying capabilities; ii) it is computationally efficient because operations can be as simple as bitwise OR and XOR; iii) it achieves quasi-perfect resilience against node compromise in that the pairwise keys are unique with high probability. As part of our future work, we plan to reduce the randomness of the proposed scheme such that the scalability of CAB can be further improved.

REFERENCES

- [1] L. Eschenauer and V. D. Gligor, "A key-management scheme for distributed sensor networks," in *CCS '02: Proceedings of the 9th ACM conference on Computer and communications security*. ACM Press, 2002, pp. 41–47.
- [2] H. Chan, A. Perrig, and D. Song, "Random key predistribution schemes for sensor networks," in *SP '03: Proceedings of the 2003 IEEE Symposium on Security and Privacy*, 2003, p. 197.
- [3] W. Du, J. Deng, Y. S. Han, and P. K. Varshney, "A pairwise key pre-distribution scheme for wireless sensor networks," in *CCS '03: Proceedings of the 10th ACM conference on Computer and communications security*. ACM Press, 2003, pp. 42–51.
- [4] D. Liu and P. Ning, "Establishing pairwise keys in distributed sensor networks," in *CCS '03: Proceedings of the 10th ACM conference on Computer and communications security*, 2003, pp. 52–61.
- [5] W. Du, J. Deng, Y. S. Han, S. Chen, and P. Varshney, "A key management scheme for wireless sensor networks using deployment knowledge," in *Proceedings of IEEE Infocom*, Hong Kong, March 2004, pp. 586–597.
- [6] D. Liu, P. Ning, and W. Du, "Group-based key pre-distribution in wireless sensor networks," in *WiSe '05: Proceedings of the 4th ACM workshop on Wireless security*, 2005, pp. 11–20.
- [7] L. Zhou, J. Ni, and C. V. Ravishankar, "Efficient key establishment for group-based wireless sensor deployments," in *WiSe '05: Proceedings of the 4th ACM workshop on Wireless security*, 2005, pp. 1–10.
- [8] L. Ma, F. Liu, X. Cheng, and F. An, "ipak: An in-situ pairwise key bootstrapping scheme for wireless sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, 2006.
- [9] F. Liu and X. Cheng, "SBK: A self-configuring framework for bootstrapping keys in sensor networks," in *MASS 2006: IEEE International Conference on Mobile Ad-hoc and Sensor Systems*, 2006.
- [10] —, "Location aware key establishment in wireless sensor networks," in *IWCMC 2006: International Wireless Communications and Mobile Computing Conference*, 2006.
- [11] H. Chan and A. Perrig, "PIKE: Peer intermediaries for key establishment in sensor networks," in *Infocom 2005: The 24th Conference of the IEEE Communications Society*, 2005.
- [12] D. Liu and P. Ning, "Location-based pairwise key establishments for static sensor networks," in *SASN '03: Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks*, 2003, pp. 72–82.
- [13] Y. Zhang, W. Liu, W. Lou, and Y. Fang, "Location-based compromise-tolerant security mechanisms for wireless sensor networks," in *IEEE Journal on Selected Areas in Communications*, 2006.
- [14] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Proceedings of CRYPTO 84 on Advances in cryptology*. New York, NY, USA: Springer-Verlag New York, Inc., 1985, pp. 47–53.
- [15] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler, "SPINS: security protocols for sensor networks," *Wirel. Netw.*, vol. 8, no. 5, pp. 521–534, 2002.
- [16] S. Zhu, S. Setia, and S. Jajodia, "LEAP: efficient security mechanisms for large-scale distributed sensor networks," in *ACM Conference on Computer and Communications Security*, 2003, pp. 62–72.
- [17] C. Karlof, N. Sastry, and D. Wagner, "TinySec: a link layer security architecture for wireless sensor networks," in *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, 2004, pp. 162–175.
- [18] S. Wolfram, "A new kind of science." Champaign, Illinois, USA: Wolfram Media Inc, 2002.
- [19] M. Tomassini, M. Sipper, and M. Perrenoud, "On the generation of high-quality random numbers by two-dimensional cellular automata," *IEEE Trans. Comput.*, vol. 49, no. 10, pp. 1146–1151, 2000.
- [20] F. Seredynski, P. Bouvry, and A. Y. Zomaya, "Secret key cryptography with cellular automata," in *IPDPS '03: Proceedings of the 17th International Symposium on Parallel and Distributed Processing*, 2003, p. 149.
- [21] S. Wolfram, "Cryptography with cellular automata," in *Lecture notes in computer sciences; 218 on Advances in cryptology—CRYPTO 85*, 1986, pp. 429–432.
- [22] D. Mukhopadhyay, P. Joshi, and D. RoyChowdhury, "An efficient design of cellular automata based cryptographically robust one-way function," *VLSID*, vol. 00, pp. 842–853, 2007.
- [23] J. Spencer, "The strange logic of random graphs." Springer, 2001.