

# Secure Path-Key Revocation for Symmetric Key Pre-distribution Schemes in Sensor Networks

Tyler Moore and Jolyon Clulow

Computer Laboratory, University of Cambridge  
15 JJ Thomson Avenue, Cambridge CB3 0FD United Kingdom  
{Tyler.Moore, Jolyon.Clulow}@cl.cam.ac.uk

**Abstract.** Path keys are secrets established between communicating devices that do not share a pre-distributed key. They are required by most key pre-distribution schemes for sensor networks, because topology is unknown before deployment and storing complete pairwise-unique keys is infeasible for low-cost devices such as sensors. Unfortunately, path keys have often been neglected by existing work on sensor network security. In particular, proposals for revoking identified malicious nodes from a sensor network fail to remove any path keys associated with a revoked node. We describe a number of resulting attacks which allow a revoked node to continue participating on a network. We then propose techniques for ensuring revocation is complete: universal notification to remove keys set up with revoked nodes, path-key records to identify intermediaries that are later revoked, and blacklists to prevent unauthorized reentry via undetected malicious nodes. Path keys also undermine identity authentication, enabling Sybil attacks against random pairwise key pre-distribution.

## 1 Introduction

A number of symmetric key management and distribution schemes have been proposed to address the trust bootstrapping problem for sensor networks. Most notable are the seminal papers of Eschenauer and Gligor [1] proposing pools of keys and Chan et al.'s [2] random key pre-distribution scheme. These papers have inspired many subsequent proposals balancing storage, computational and communication overhead while retaining reasonable security levels [2, 3, 4, 5, 6].

Most key distribution schemes pre-load a limited number of secret keys into permanent memory so that nodes can either communicate directly using a shared key or, failing that, set up a *path key* using intermediaries they do share keys with. Path keys are a necessity for any scheme that minimizes storage costs prior to deployment. But path keys must also be considered in the later stages of credential revocation. Existing revocation proposals [1, 2, 7] fail to remove path keys established during operation. This oversight enables attackers to wreak havoc in a number of ways: rejoining the network after dismissal, issuing spoofed revocation messages, and retaining access to path keys established for others. Safeguarding revocation mechanisms from these attacks is essential.

To our knowledge, this is the first paper in the literature of key pre-distribution schemes and revocation mechanisms for sensor networks to identify the need for and difficulty in revoking path keys. We demonstrate that existing proposals used in conjunction with path keys are vulnerable to number of attacks, defeating attempts to revoke bad nodes and enabling Sybils [8] where one node pretends to be many. We propose *path-key records*, which detail the identifiers of proxy nodes that help establish each path key. These records are used to identify and remove path keys tainted by a bad node. We show that the combined use of path-key records and blacklisting can secure a centralized revocation mechanism such as Eschenauer and Gligor’s. We also show how to modify decentralized revocation schemes to make revocation decisions verifiable to the entire network. Finally, we show that naïve instantiations of Sybil detection mechanisms where results are not verifiable to third parties leave the network vulnerable to path-key-enabled Sybil attacks.

## 2 Background

There are four basic events in the life cycle of a distributed, wireless, sensor network: *pre-deployment*, *initialization*, *operation* and *revocation*. In pre-deployment, the network owner programs nodes with keys and authentication values. This is regarded as a secure operation occurring away from the attacker under the owner’s control. Nodes are then deployed into the environment where attackers may be present and initialized by establishing keys with their neighbors. When nodes are mobile, key setup is ongoing as they establish links with new neighbors and break links with old ones. At any stage, one or more nodes may find another node misbehaving, prompting a decision mechanism to determine whether the node should be removed from the system. Revocation makes invalid any credentials shared between the revoked node and honest nodes.

In the pre-deployment phase, keys and authentication values are computed by the owner and stored on the nodes. The keys assigned to nodes are effectively also their identities. As a result, the uniqueness of a node’s identity is tied to the secrecy of the keys it has been assigned. A message encrypted under a symmetric key assigned to a group of nodes could have originated from any node in the group. Encrypting under a pairwise unique key, by contrast, unambiguously demonstrates a node’s identity to the other node that shares the key.

### 2.1 Key pre-distribution schemes

The simplest architecture is a single shared key known to all nodes. This scheme is vulnerable to the compromise of a single node, and revocation is impossible. At the other extreme is the *complete pairwise scheme*, where every node stores a unique pairwise key for each of the  $n - 1$  other nodes in the network. Here all nodes can confidentially communicate with each other, and any individual node can be revoked. However the scheme is infeasible when considering large

networks of low cost nodes with limited storage space. We now review a number of proposals seeking a middle ground where a limited number of keys are assigned to nodes while maintaining a high likelihood of node confidentiality.

Eschenauer and Gligor [1] propose two related techniques for reducing the number of keys pre-loaded onto nodes: *key pools* and *random key assignment*. Here each node is randomly assigned  $m$  keys from a large pool  $P$  of  $l$  keys (where  $m \ll n$  and  $l \gg n$ ). Nodes determine which keys are shared between them by querying each other for the identifiers of keys held. These *link keys* are used to secure and authenticate messages between nodes. Using results from random graph theory to identify suitable choices of  $l$  and  $m$ , the network is probabilistically guaranteed to be connected, with nodes sharing a link key with an average of  $d$  neighbors in communication range. This pooling mechanism and random key assignment have inspired several extensions and variations. [2] generalizes the scheme to require  $q$  shared secrets to establish a link key. In [4, 5], the authors propose creating a large polynomial pool to offer threshold secrecy.

Eschenauer and Gligor randomly assign keys to nodes; thus the only way for nodes to determine whether they share keys is to exchange lists of key identifiers, or by challenge-response. Zhu et al. [3] propose a deterministic algorithm that calculates the key identifiers known to a node from the node identifier. This increases efficiency, but it also helps an attacker obtain any desired keys by targeting nodes for compromise. To counter this, [9] describes a way for nodes to check whether they share keys without revealing all keys held by every node.

While undoubtedly reducing storage requirements, key pools also introduce a new set of security challenges. First, it is impossible to authenticate identity based on the keys held by a node, since several nodes may legitimately possess the same keys. Second, pool keys make it hard to revoke a misbehaving node's keys without negatively impacting legitimate nodes. Removing compromised keys is onerous since many nodes across the network could be assigned some keys in common with a revoked node; yet removing too many keys could deplete the key pool, causing inadvertent denial-of-service attacks. Finally, pool keys make harvesting attacks attractive, where an attacker compromises enough nodes to increase the chance of reusing keys to eavesdrop other nodes' communications.

Chan, Perrig and Song propose a *random pairwise* scheme [2] as an alternative to key pools combining aspects of the complete pairwise scheme with the storage-saving random distribution approach of [1]. Nodes are pre-loaded with pairwise unique keys, but rather than allocate  $n - 1$  keys per node, a fraction of the keys are randomly assigned. Pre-distributing pairwise unique keys prevents a key-harvesting attacker from compromising the confidentiality of any pre-assigned key shared between uncompromised nodes. It also enables mutual authentication between nodes sharing a key. This forms the basis of a revocation scheme whose details we describe below. One disadvantage of the random pairwise scheme is increased storage cost: nodes are pre-loaded with keys totaling a significant fraction of  $n$  (e.g.,  $\frac{1}{5}$  to  $\frac{1}{3}$ ). Since very few keys are used for neighbors in communication range, a small number of colluding nodes can set up many fake link keys across the network to drown out legitimate links [10].

## 2.2 Path-key establishment

In all of the schemes outlined above, there must exist neighboring nodes that are not pre-assigned a common key but wish to communicate. These nodes must discover a path between each other using a number of intermediate nodes, where each hop is secured by a link key. One of the nodes chooses a new *path key* and sends it to the other node, encrypted using link keys between intermediaries.

Intermediate nodes are selected for setting up path keys in two ways. In *random* path-key establishment, nodes discover paths to other nodes using locally broadcast messages. The average path length depends on the scheme used. In random pool-key deployment with plausible values for  $l$ ,  $m$  and  $n$ , path keys to most neighboring nodes can be established within three hops [1]. Path keys to distant nodes are more expensive, however, requiring an average of eleven link keys for the simulations in [1]. Random path-key establishment is simple but has relatively high communication costs. Schemes using random path-key establishment include [1, 2, 4, 5]. Alternatively, in *deterministic* path-key establishment, link keys are assigned based on a node's identifier so that nodes can unilaterally select the intermediaries used to set up path keys. This eliminates the communication cost of searching for suitable intermediate nodes. Schemes using deterministic path-key establishment include [6, 3].

Path-key establishment is also vulnerable to malicious intermediaries, since only link-level encryption is used to establish an end-to-end key. Several papers explore multi-path key reinforcement for strengthening path keys [11, 2, 3].

## 2.3 Revocation mechanisms

Few papers on sensor networks consider the revocation phase at all. Eschenauer and Gligor describe a centralized scheme [1] where the base station determines which keys are tied to a compromised node and instructs all nodes holding these keys to delete them. In [2], Chan, Perrig and Song propose a distributed revocation mechanism for the random pairwise scheme, where nodes sharing pre-assigned pairwise keys vote to remove a node. Their scheme is extended and generalized in [7]. Here, each node  $B$  that shares a pairwise key with  $A$  is assigned to the set of *participants of A*,  $V_A$ . Every node  $A$  is assigned a unique revocation secret  $\text{rev}_A$ , which is divided into secret shares and given to every  $B \in V_A$  along with an authentication value for the revocation secret,  $h^2(\text{rev}_A)$ . Nodes vote for another's removal by revealing their share. If enough shares are revealed, then  $\text{rev}_A$  is reconstructed and  $h(\text{rev}_A)$  broadcast across the network. Every node  $B \in V_A$  deletes its key shared with  $A$  upon verifying the broadcast.

## 2.4 The Sybil attack and countermeasures

Sybil identities [8], where a malicious node pretends to be multiple distinct nodes in the network, can facilitate attacks against routing, voting, misbehavior detection, distributed storage, resource allocation and data aggregation in

sensor networks. As discussed earlier, a node’s identity is determined by the the keys it holds. So key pool schemes, for example, are especially vulnerable to Sybil attacks: an attacker can create many fake identities from a few known pool keys since many nodes would be expected to hold these keys.

Newsome et al. propose a Sybil detection mechanism where honest nodes challenge each other for the expected pre-loaded keys associated with claimed identities [12]. Two nodes that share a common key can directly challenge each other to prove they have knowledge of the key. As more nodes challenge a given node in this way, confidence increases that this node is not a Sybil identity. However, the authors do not specify how to aggregate the results of a number of direct challenges in a manner verifiable to other nodes in the network. This they term *indirect validation* and describe as a challenging problem in the absence of a trusted central authority because malicious nodes must be prevented from vouching for each other. Potential approaches to indirect validation include reputation or voting schemes. However, both are prone to manipulation and increase the computational and communication overhead significantly.

### 3 Path-key-enabled attacks

In this section we describe two classes of attack that exploit path keys: circumventing revocation mechanisms and Sybil attacks.

We use the following threat model. Honest nodes adhere to their programmed strategy including algorithms for routing and revocation. The attacker can compromise a small minority of nodes  $M_1, M_2, \dots, M_i$  since devices may be unprotected and deployed in hostile environments. All such malicious nodes can communicate with each other and with honest nodes. Malicious nodes have access to any secret information, including keys, of all other malicious nodes, and can use their identifiers if desired. They do not have to correctly execute revocation mechanisms to identify misbehavior or delete keys shared with revoked nodes. Notably, we do not assume active node compromise is immediately detected. In fact, node compromise may never be detected.

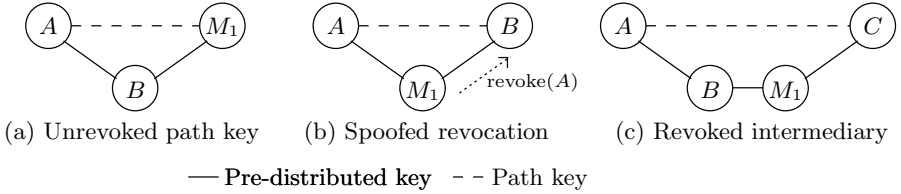
In particular, we consider two threat models:

**T.0** We assume a global passive adversary upon deployment. However, no nodes are actively compromised until path-key establishment is complete.

**T.1** Again, we assume a global passive adversary upon deployment. Here we allow the adversary to actively compromise a small minority of nodes prior to path-key establishment. This threat model is adopted by most key distribution schemes for sensor networks [1, 2, 4, 5, 6].

#### 3.1 Path-key attacks on revocation mechanisms

**Incomplete revocation of path keys** In a centralized revocation scheme, the base station issues revocation orders verifiable by all other nodes which then delete any paths keys shared with the revoked node. However, under the



**Fig. 1.** Path-key attacks on revocation mechanisms.

distributed schemes described above, the only nodes that can verify votes are those that can participate in a revocation vote. Therefore, only these nodes, which have been pre-assigned keys, know to revoke keys; therefore, only the pre-assigned keys are removed during revocation. Notably, *nothing is done to remove any path keys established with the revoked node.*

Any revocation scheme that does not remove path keys is vulnerable to the attacks in Figure 1. Consider the network in Figure 1(a). Suppose a malicious node  $M_1$  has been identified and a revocation order issued to all nodes sharing pairwise keys with  $M_1$ .  $B$  knows to remove the key shared with  $M_1$ , but  $A$  does not so the path key established between  $M_1$  and  $A$  continues to function.

It is not possible to counter this attack by allowing  $A$  to accept forwarded revocation claims from  $B$ , since  $A$  cannot verify the veracity of the claim from  $B$  of  $M_1$ 's revocation (apart from that  $B$  made it). This lack of authentication would enable the attack shown in Figure 1(b) where the undetected malicious node  $M_1$  could lie to  $B$ , falsely claiming that honest  $A$  had been revoked.

**Malicious intermediaries and path keys** The threat of malicious intermediaries during the establishment of path keys has been investigated by a number of authors [11, 2, 3]. These authors have focused on making the path key establishment mechanism as robust as possible under threat model T.1 and include techniques such as using multiple disjoint paths. These methods make it harder, but not impossible, for an attacker to compromise a path key, requiring that more intermediary nodes be compromised.

However, these papers do not consider what to do if, as a result of such an attack, a path key is, or might be, compromised. For example, suppose that  $M_1$  has served as an intermediary to establish a path key between  $A$  and  $C$  as in Figure 1(c). Suppose further that  $M_1$  is subsequently identified as malicious and revoked from the network. While  $C$  could observe a revocation order for  $M_1$ , it is unaware that its path key to  $A$  was set up via  $M_1$ .  $M_1$  could use its knowledge of the path key to eavesdrop on or rejoin the network. Clearly, the path key should also be revoked.

The use of multiple disjoint paths during path key establishment simply changes the threshold at which the path key should be revoked. Once an intermediary on each path has been compromised, the resulting path key should also be revoked. A conservative network may require this to happen earlier (e.g., once half of the paths are compromised or when just one path remains secure).

Note that both threat models T.0 and T.1 are relevant. In the latter case, the attacker has actively compromised  $M_1$  prior to path-key setup and can immediately determine  $k_{AC}$ . However, a path-key recovery attack is still possible under threat model T.0, where an adversary eavesdrops traffic during path-key setup but does not compromise the intermediary until after path-key setup. Suppose  $A$  establishes a path key to  $C$ , using  $M_1$  as an intermediary. Here,  $A$  sends  $\{k_{AC}\}_{k_{AM_1}}$  to  $M_1$ , which then transmits  $\{k_{AC}\}_{k_{M_1C}}$  to  $C$ . When the attacker subsequently compromises  $M_1$ , she can recover  $k_{AM_1}$  or  $k_{M_1C}$  and decrypt the message containing  $k_{AC}$ .

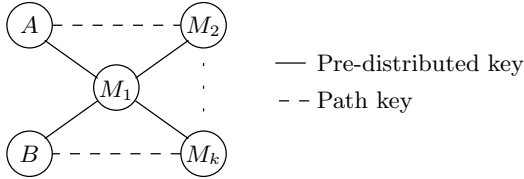
**Compromised but unrevoked pool keys** The centralized revocation scheme of Eschenauer and Gligor [1] is susceptible to an additional path key attack. Since generating truly random noise is hard for sensor nodes to do, the authors advocate that nodes select unused keys from their key rings as path keys. These keys are, of course, pool keys. A malicious node can establish as many path keys with neighbors as possible, requiring them to provide an unused pool key (and therefore, a key that the attacker does not already possess).

However, the network owner is never informed that the node knows additional pool keys. Therefore, should the network owner subsequently revoke the node, none of the path keys are removed. The malicious node retains not only the path-key-enabled links to its neighbors, but also the pool keys for establishing communications back into the system. However, addressing this attack by tracking and removing path keys (and thus pool keys) could enable a denial-of-service attack whereby the adversary deliberately depletes the key pool by setting up many unused pool keys as path keys prior to revocation.

**Unauthorized reentry of revoked nodes** One problem with implementing revocation by simply deleting shared keys is that it is not permanent when multiple undetected compromised nodes are present. Suppose malicious nodes  $M_1$  and  $M_2$  both share link keys with honest node  $A$ , and a revocation order is issued for  $M_1$  but not  $M_2$ .  $A$  deletes its link key with  $M_1$ , as do all honest neighbors of  $M_1$ . Yet  $M_1$  can rejoin the network by establishing path keys using  $M_2$  as an intermediary. Unless honest nodes are required to maintain a network-wide blacklist of all revoked nodes,  $A$  does not remember that  $M_1$  has already been revoked. Under random path-key establishment,  $M_1$  can rejoin via any colluding node. For deterministic path-key establishment schemes,  $M_1$  can only rejoin via colluding nodes pre-assigned a key.

### 3.2 Path-key-enabled Sybil attacks

Pool-key-based pre-distribution schemes are susceptible to Sybil attacks since shared keys are not guaranteed to be unique. Pairwise-key-based pre-distribution schemes, by contrast, should be Sybil-resistant since the keys are unique, which enables authentication between nodes sharing keys. However, consider the scenario given in Figure 2. Node  $M_1$  shares a pairwise key with  $A$  but creates several fake nodes  $M_2..M_k$  and requests path keys for each of them. Under Chan et



**Fig. 2.** Path-key-Sybil attacks on revocation mechanisms.

al.’s distributed revocation protocol, these Sybil nodes are unrevokable! Their sets of voting members,  $V_{M_2} \dots V_{M_k}$ , are all empty since the identities are fake, yet  $A$  has no way of knowing this. Node  $M_1$  can use each of these fake identities to carry out attacks, while otherwise behaving honestly to its real neighbors.

The fact that path keys enable Sybil attacks is important because it has been claimed in [12] that ‘an adversary cannot fabricate new identities’ under the random pairwise scheme, making it immune to Sybil attack. As we have demonstrated, Sybil attacks remain viable under the random pairwise scheme due to the use of path keys. So a scheme such as Newsome et al.’s must be employed to detect Sybils for random pairwise schemes. But their direct key validation Sybil defense [12] cannot detect these path-key-Sybil attacks for the random pairwise scheme. What we require is the missing indirect validation protocol (that is, a protocol that allows nodes which don’t share keys with the target node to be able to verify claims from nodes that do share keys). So while Chan et al.’s revocation scheme [7] assumes the existence of adequate Sybil attack detection and refers to Newsome et al.’s techniques as an example, path-key-enabled Sybil attacks remain an unaddressed impediment to effective revocation, and to pairwise key pre-distribution in general.

Note also that the degree-counting mechanism proposed in [2] cannot detect Sybil attacks. It detects attackers using more pairwise keys than allowed, but these attacks create path keys without using any pre-assigned pairwise keys.

## 4 Secure path-key revocation

We now present three techniques for securing revocation mechanisms from the path-key attacks outlined above: (1) complete notification of node revocation, (2) path-key records to identify malicious intermediaries and (3) blacklists to prevent unauthorized reentry via path keys. We propose both centralized and decentralized solutions where appropriate.

### 4.1 Complete notification of node revocation

Every node eligible to establish a path key with a revoked node must be notified of its compromise. In sensor networks where the topology is unknown before deployment, path keys could conceivably be established between any two nodes. Thus, every node must be notified of every revocation.

A centralized revocation mechanism for removing pre-distributed keys, similar to the one proposed by Eschenauer and Gligor, can be trivially augmented to revoke path keys. Here, the central authority unicasts a message to every node (signed by the pairwise key shared between the authority and the node) instructing them to remove any path keys established with the node.

For a decentralized revocation mechanism, nodes must verify revocation messages sent by other nodes even when they do not necessarily trust them. To do so, each node is loaded with an authentication value for the revocation secrets of all  $n$  nodes in the network. (Recall from Section 2.3 that revocation secrets are reconstructed from voting shares broadcast by nodes eligible to decide when it is best to revoke a node.) This is in contrast to Chan et al.'s distributed revocation mechanism, which equips nodes with only the ability to verify the revocation of nodes sharing pre-assigned keys. This  $O(n)$  storage cost could impede deploying revocation with symmetric keys for large networks. While costs may be reduced by storing authentication values as leaves in a Merkle tree [13], one must be careful in distributing the  $\log n$  path-authentication values. We cannot rely on the node being revoked to provide the information to verify its own removal; a safer alternative is for every voting member to keep a copy of the path-authentication values for each node it might revoke.

## 4.2 Path-key records to identify malicious intermediaries

Recall that under threat model T.0, an attacker may collect traffic that passes through a node, then compromise it and determine all path keys established with the node as intermediary (Figure 1(c)). However, if nodes periodically update their link keys using a one-way function, e.g.,  $k'_{AB} = h(k_{AB})$ , then an attacker cannot recover any path keys established prior to node compromise.

To address the case where nodes are compromised during path-key establishment (under threat model T.1), nodes must keep track of the intermediate nodes used to establish each path key so that affected path keys can be removed once node compromise is detected. To do so, nodes can build a list of all node identifiers used as intermediaries in conjunction with path-key establishment. When node  $A$  establishes a path key with node  $B$ , it stores a *path-key record*

$$B, K_{AB}, N_1, N_2, \dots, N_l$$

where  $K_{AB}$  is the path key, and  $N_1, N_2, \dots, N_l$  are the identifiers for the  $l$  intermediate nodes. Whenever a revocation order is issued, nodes must check their path-key records for the revoked nodes, discard affected path keys and reinitiate transmission to discover a new path key.

Path-key record generation and verification should remain decentralized, even when access to a central authority is used for other components of path-key revocation. Because path-key records are constructed every time a path key is established, it is unreasonable to always consult a base station. If this frequency of communication with base stations is allowed, then nodes are better off using the base station to set up the path keys in the first place.



originating from different places. Similarly, nodes could be required to remember only a subset of revoked nodes (e.g., each node remembers nodes it has decided to revoke). Nodes check the identifiers transmitted in the node replication detection messages against their own subset of the blacklist. If detected, nodes forward the message to the base station, which issues a new revocation order. This scheme can be less expensive than requiring nodes to maintain a complete blacklist when node replication detection is already in frequent use.

#### 4.4 Cost summary

In summary, path keys impose the following additional costs to those outlined in [7] for revocation to be effective:

- Authenticated revocation secrets for all nodes in the network
- Maintain path-key record listing all intermediaries on every path key
- Maintain blacklist of all revoked nodes in network

These previously unaccounted costs reflect the difficulty in designing efficient revocation mechanisms using pre-distributed symmetric-key cryptography. We believe these costs are unavoidable whenever anything less than complete pairwise keys are used. Furthermore, path keys necessitate Sybil attack detection using indirect validation [12] to secure pairwise key pre-distribution schemes.

## 5 Conclusions

Any symmetric key management scheme pre-distributing less than complete pairwise keys necessarily weakens notions of identity. Complications inevitably ensue. In this paper we identified problems with revocation mechanisms and Sybil identities caused by path keys. We proposed effective countermeasures to ensure that keys shared with or exposed to revoked nodes are removed, and blacklists to prevent the unauthorized reentry of revoked nodes. We note that exposure to path-key attacks may be limited by employing deterministic path-key establishment mechanisms and minimizing the number of intermediaries used. We also showed that, contrary to prior understanding, path keys make incomplete pairwise key-distribution schemes vulnerable to Sybil attacks. Pairing key pre-distribution schemes with Newsome et al.'s Sybil detection scheme is not convincing; the authors themselves note they do not provide a practical way to detect Sybils by nodes not sharing pre-distributed keys, the case for path keys. It remains an open problem whether an efficient Sybil detection mechanism can be created for this scenario.

More generally, the efficiency gains made at one stage in the life cycle of a network may cause unforeseen problems that are expensive to remedy at other stages. We have shown that trade-offs made to improve the efficiency of bootstrapping keys to sensor nodes open the door to devastating attacks that are costly to handle during the maintenance phase of revocation, counteracting the

gain from the earlier trade-offs. This resonates with Anderson's argument that protocol designers have long underestimated the maintenance costs of security mechanisms [15]. One could continue to layer on patchwork mechanisms for mitigating these attacks, accepting the costs as unavoidable. In contrast, we question whether efficient key establishment coupled with inefficient or insecure revocation is desirable. Instead, we should perhaps consider selective uses of asymmetric cryptography or develop more innovative revocation mechanisms.

## References

1. L. Eschenauer and V. D. Gligor, A Key-Management Scheme for Distributed Sensor Networks, *ACM Conference on Computer and Communications Security*, 2002, pp. 41–47.
2. H. Chan, A. Perrig, and D. X. Song, Random Key Predistribution Schemes for Sensor Networks, *IEEE Symposium on Security and Privacy*, 2003, pp. 197–213.
3. S. Zhu, S. Xu, S. Setia, and S. Jajodia, Establishing Pairwise Keys for Secure Communication in Ad Hoc Networks: a Probabilistic Approach, *IEEE International Conference on Network Protocols*, 2003, pp. 326–335.
4. W. Du, J. Deng, Y. S. Han, and P. K. Varshney, A Pairwise Key Pre-distribution Scheme for Wireless Sensor Networks, *ACM Conference on Computer and Communications Security*, 2003, pp. 42–51.
5. D. Liu and P. Ning, Establishing Pairwise Keys in Distributed Sensor Networks, *ACM Conference on Computer and Communications Security*, 2003, pp. 52–61.
6. H. Chan and A. Perrig, PIKE: Peer Intermediaries for Key Establishment in Sensor Networks, *IEEE INFOCOM*, 2005, pp. 524–535.
7. H. Chan, V. D. Gligor, A. Perrig, and G. Muralidharan, On the Distribution and Revocation of Cryptographic keys in Sensor Networks, *IEEE Trans. Dependable Secur. Comput.* **2**(3), 233-247 (2005).
8. J. R. Douceur, in: *Lecture Notes in Computer Science 2429*, edited by P. Druschel, M. Kaashoek, and W. Rowstron (Springer, Heidelberg, 2002), pp. 251–260.
9. R. Di Pietro, L. V. Mancini, and A. Mei, Energy Efficient Node-to-Node Authentication and Communication Confidentiality in Wireless Sensor Networks, *Wireless Networks* **12**(6), 709–721, 2006.
10. T. Moore, A Collusion Attack on Random Pairwise Key Predistribution Schemes for Distributed Sensor Networks, *IEEE International Workshop on Pervasive Computing and Communications Security*, 2006, pp. 251–255.
11. R. J. Anderson, H. Chan, and A. Perrig, Key Infection: Smart Trust for Smart Dust, *IEEE International Conference on Network Protocols*, 2004, pp. 206–215.
12. J. Newsome, E. Shi, D. X. Song, and A. Perrig, The Sybil Attack in Sensor Networks: Analysis and Defenses, *Information Processing and Sensor Networks*, 2004, pp. 259–268.
13. R. C. Merkle, Protocols for Public-Key Cryptosystems, *IEEE Symposium on Research in Security and Privacy*, 1980, pp. 122–134.
14. B. Parno, A. Perrig, and V. D. Gligor, Distributed Detection of Node Replication Attacks in Sensor Networks, *IEEE Symposium on Security and Privacy*, 2005, pp. 49–63.
15. R. Anderson, The Initial Costs and Maintenance Costs of Protocols, *International Workshop on Security Protocols*, 2005.