Chapter 23
Key Management in Sensor Networks

Dahai Xu, Jeffrey Dwoskin, Jianwei Huang, Tian Lan, Ruby Lee, and Mung Chiang

Abstract Secure communications in wireless ad hoc networks require setting up end-to-end secret keys for communicating node pairs. It is widely believed that although being more complex, a probabilistic key predistribution scheme is much more resilient against node capture than a deterministic one in lightweight wireless ad hoc networks. Supported by the surprisingly large successful attack probabilities (SAPs) computed in this chapter, we show that the probabilistic approaches have only limited performance advantages over deterministic ones. We first consider a static network scenario as originally considered in the seminal paper by Eschenauer and Gligor [9], where any node capture happens after the establishment of all pairwise links. In this scenario, we show that the deterministic approach can achieve a performance as good as the probabilistic one. In a mobile network scenario, however, the probabilistic key management as described in [9] can lead to a SAP of one order of magnitude larger than the one in a static network due to node fabrication attacks.

The above analysis motivates us to propose two low-cost secure-architecture-based techniques to improve the security against such attacks. Our new architectures, specifically targeted at the sensor-node platform, protect long-term keys using a root of trust embedded in the hardware System-on-a-Chip (SoC). This prevents an adversary from extracting these protected long-term keys from a captured node to fabricate new nodes. The extensive simulation results show that the proposed architecture can significantly decrease the SAP and increase the security level of key management for mobile ad hoc networks.

Finally, we develop an analytical framework for the on-demand key establishment approach. We propose a novel security metric, the REM resilience vector, to quantify the resilience of any key establishment schemes against Revealing, Erasure, and Modification (REM) attacks. Our analysis shows that previous key establishment schemes are vulnerable under REM attacks. Relying on the new security metric, we prove a universal bound on achievable REM resilience vectors for any on-demand

D. Xu (✉)
AT&T Labs - Research, 180 Park Ave, Building 103, Florham Park, NJ 07932, USA
e-mail: dahaixu@research.att.com

The AT&T Labs - Research, The work was done when Xu was in Princeton University.

key establishment scheme. This bound that characterizes the optimal security performance analytically is shown to be tight, as we propose a REM-resilient key establishment scheme which achieves any vector within this bound. In addition, we develop a class of low-complexity key establishment schemes which achieve nearly optimal REM attack resilience.

23.1 Introduction

23.1.1 Motivation

Lightweight ad hoc networks typically consist of nodes that are distributed and have very limited computation and energy resources. Examples include portable mobile devices and tiny low-cost sensors used for environment surveillance and emergency response. Providing secure communication over this kind of network is challenging. Various key management schemes have been proposed with an attempt to provide a highly secure communication environment in lightweight ad hoc networks against various malicious attacks. Among the proposed schemes, *symmetric* key predistribution schemes (e.g., [1, 5, 15, 16, 23–25, 27]) are more suitable to the lightweight ad hoc network than *asymmetric* public-key schemes, because the former schemes require less resources (e.g., battery, memory, and computation power) and there is no need for a trusted third party for authorization.

There are two main approaches for symmetric key predistribution: *probabilistic* (e.g., [4, 5, 9, 17, 22, 27]) and *deterministic* (e.g., [2, 19, 30, 31]). In a probabilistic approach, the keys in each node's key ring are randomly chosen from a large key pool. In a deterministic approach, the key ring is chosen deterministically. In general, probabilistic approach end up with a large key pool, a larger key ring per node, and poorer network connectivity than a deterministic one.[1] On the other hand, a typical deterministic algorithm preloads each node with a single common key and reaches connectivity of 100%. More related references can be found in the survey [3].

It is often believed that a typical probabilistic scheme is much more resilient against node captures than a typical deterministic approach [4, 5, 9, 32], thus making probabilistic schemes popular despite their clear disadvantage on many other metrics. *In this chapter, we show that the probabilistic approaches have only limited advantages over deterministic approaches even when considering node capture*. Our performance measurement is the *Successful Attack Probability* (SAP). In particular, we consider an attack on a pairwise link between two authorized nodes to be successful if a compromised node can intercept and decipher the information transmitted through that link.

[1] For example, the probabilistic scheme in [9] requires preloading each node with 83 keys out of a key pool size of 10, 000 and achieves a local direct connectivity of 50%.

23.1.2 Summary of Our Study Between Representative Probabilistic and Deterministic Schemes

The probabilistic scheme was first proposed in the seminal and widely cited paper by Eschenauer and Gligor [9], and we call the corresponding scheme the *EG scheme*. It consists of three phases: *key distribution*, *shared key discovery*, and *path-key establishment*. In the key distribution phase, each node is loaded with k keys randomly chosen from a large key pool of size m, where $k \ll m$. The shared key discovery is the process of establishing a pairwise link between two neighbor nodes if they share one or more key(s). Finally, in the path-key establishment phase, a pairwise link is established between any two neighboring nodes who do not share any key but can establish a path between them through one or more relay nodes. In this case, a path-key is sent from one node to its neighbor through the relay(s), and then a link is established similarly to the shared-key discovery phase.

A typical deterministic scheme uses only a single common key, and each node is preloaded with the same initial key. After the deployment, each pair of neighbor nodes exchanges the messages encrypted by the common initial key to derive a unique (and are often random) key for all later communications between them.

Throughout the chapter, we will compare the performance of probabilistic and deterministic key management schemes based on the EG scheme [9] and single common key scheme. *We will show that the probabilistic scheme is not significantly better than the deterministic scheme measured in terms of SAP.* Since the single common key is one of the simplest deterministic schemes, any further improvement over it (e.g., [31]) will only reinforce our conclusion.

We consider two network scenarios: *static network* and *mobile network*. In a static network, all pairwise links have been established before an adversary captures any node. This is the case previously considered in [9]. This could happen, for example, if all nodes are deployed almost at the same time and remain stationary after deployment. In contrast, in a mobile network, an adversary can capture a node before all pairwise links have been established. This is true for a network where nodes are constantly on the move and need to establish new links. This includes, for example, a sensor network of buoys floating freely on the ocean to gather environmental data[28], or a network consisting of sensors moving around in an unknown environment to form reasonable coverage [13].

In a static network, the single common key deterministic scheme can achieve almost perfect resiliency against node capture (i.e., SAP ≈ 0). This is because the initial common key can be deleted permanently from all nodes after the establishment of all pairwise keys (as in [30]). Since all pairwise keys are randomly generated and known only to the corresponding two neighbor nodes, they cannot be deduced by a captured node even if the common initial key is disclosed. In the EG scheme, however, the SAP equals k/m with only one captured node where each neighbor node pair uses one of the shared keys to encrypt the communication. It is possible to reduce the SAP to almost 0 as in the single common key case if two neighbor nodes also generate a random key for future communication. In short, the deterministic scheme can achieve performance as good as the probabilistic approach in a static network, but with much lower complexity.

In a mobile network, the single common key deterministic scheme could lead to an SAP as high as 100% if the common initial key is obtained by an adversary before any link is established. However, we show that the EG algorithm is also quite vulnerable in this case and may lead to a value of SAP one order of magnitude larger than in the static network case (e.g., as high as 60%), especially when the adversary can fully utilize the keys obtained from several compromised nodes. The intuition for the surprising result is as follows. In the static network, there is only one way to attack a link successfully, i.e., knowing the key with which the communications on that link is encrypted. In a mobile network, however, a compromised node can also attack a link by acting as a relay during the path-key establishment phase. By intercepting the key information that is being relayed, a compromised node can figure out the key which the two authorized nodes will use for future mutual communication. This new *man-in-the-middle* attack opportunity can significantly increase the value of SAP for a probabilistic approach, since nodes frequently use a relay for link establishment.

After thus re-examining the performance difference between probabilistic and deterministic key predistribution schemes, we propose two secure hardware-based techniques, specifically targeted to the sensor-node platform, which protect long-term keys. Such techniques can be used to improve the performance of both deterministic and probabilistic key management schemes for mobile networks. They ensure that protected secrets cannot be extracted from a captured node. This is the first step toward building a comprehensive low-cost secure-hardware design for sensor nodes.

The rest of this chapter is organized as follows. In Section 23.2, we calculate the values of SAP in both static and mobile networks, with a focus on the probabilistic approach (i.e., EG scheme). In Sect. 23.3, we propose processor architecture-based techniques for securing secret keys and critical software on a node. In Sect. 23.4 we analyze the security of the proposed architecture under several specific attacks. In Sect. 23.5, we validate the analytical results from Sect. 23.2 and the security improvement of the proposed architecture with simulations based on a C++ simulator. In Sect. 23.6, we reexamine other probabilistic key predistribution schemes. In Sect. 23.7, we introduce our REM resilient key establishment framework and a low-complexity protocol. We conclude in Sect. 23.8.

23.2 Fragility Analysis for Probabilistic Key Management

In this section, we first review the results in [9], where the successful attack probability (SAP) is calculated for a static network. We then consider a mobile network and show how the value of SAP is significantly larger in that case. We only consider the attacks on the *pairwise* link between two authorized nodes that are within each other's communication range. The SAP will be even higher if A and B are far away and can only be connected with a multi-hop path, since a successful attack on any hop will jeopardize the confidentiality of the whole communication.

The establishment of a link requires two neighbor nodes, A and B, to be able to encrypt the communication over such a link using a common key. This could be achieved in two ways:

(i) A and B share a key within their preloaded key rings, thus can establish the link directly.
(ii) A and B do not share a key initially and need to exchange additional information through one or more relay nodes, with whom the pairwise links have already been established. For example, A can randomly choose an unused key from its key ring and send it to B through the relay node(s). Then A and B can use this key to encrypt the pairwise key between them.

In either case, SAP of the link between A and B is defined as

$$SAP \triangleq P(A \otimes B | A \leftrightarrow B)$$

where $A \otimes B$ denotes the event that the link between A and B is successfully attacked, and $A \leftrightarrow B$ denotes the event that A and B establish a link between them. Since a link can only be attacked if it has been established, we have (23.1) and (23.2):

$$P(A \otimes B \cap A \leftrightarrow B) = P(A \otimes B) \tag{23.1}$$

$$SAP = \frac{P(A \otimes B)}{P(A \leftrightarrow B)} \tag{23.2}$$

All the notation used in this section is defined in Table 23.1 to enable a cleaner presentation of later derivations. A, B, and C denote three generic nodes, and $\mathbb{C}^h$ denotes a set of h nodes. Each node is preloaded with a key ring of k randomly chosen keys out of a key pool of size m.

Table 23.1 Summary of notation

Notations	Meaning
$A \leftrightarrow B$	A and B establish a pairwise link between them
$A \leftrightarrow \mathbb{C}^h \leftrightarrow B$	A and B communicate through one node in $\mathbb{C}^h$
$A \otimes B$	The link between A and B is successfully attacked
$A \sharp B$	A and B share at least one key
$(A \sharp B) \lhd C$	C has all the keys (≥ 1) shared by A and B
$(A \sharp B) \lhd \mathbb{C}^h$	At least one node of $\mathbb{C}^h$ has all the keys (≥ 1) shared by A and B
$(A, B) \sharp \mathbb{C}^h$	At least one node in $\mathbb{C}^h$ shares at least one key with A and at least one key with B
$(A, B) \sharp \mathbb{C}^h_r$	Exactly r nodes out of $\mathbb{C}^h$, each of which shares at least one key with A and at least one key with B

23.2.1 SAP for a Static Network

If a compromised node wants to attack an established link, it needs to know the key that is used to encrypt the link. Therefore, a compromised node can successfully attack an existing link with probability k/m, as stated in [9].

23.2.2 SAP for a Mobile Network

In a mobile network, a compromised node C can attack the link between A and B in three ways:

(i) If A and B share a key initially and establish the link directly, then C needs to know the key chosen by A and B to encrypt the link.
(ii) If A and B do not share a key initially and use C as a relay, then C can get the desired information while relaying the information between A and B. A first communicates with C via encrypted messages protected by shared key K_{ac}. C decrypts this with K_{ac} giving it access to the plain text message, re-encrypts it with K_{cb}, a key it shares with node B, and then sends the re-encrypted message to B. This sets C up as a man-in-the-middle eavesdropper between A and B, since C can see the plain text of all messages going from A to B.
(iii) If A and B do not share a key and do not choose C within the relay path, C can still attack the communication between A and B by either eavesdropping on the links along the relay path or attacking the eventual pairwise link established between A and B, if it has any of the keys used for these links.

Overall, the value of SAP depends on the number of compromised nodes and authorized nodes within both A and B's communication range, as well as how A and B choose the relay nodes. To simplify the analysis, we only consider cases (i) and (ii), and further assume only one node relay in case (ii). In the simulation in Sect. 23.5, we calculate SAP for all three cases.

It will be useful to know the probability of sharing at least one key between any two nodes in the network. Denote δ_m^k as the probability that any two nodes A and B do *not* share any key, then

$$\delta_m^k \triangleq P\left(\overline{A\sharp B}\right) = \binom{m-k}{k} \Big/ \binom{m}{k}$$

where $A\sharp B$ denotes A and B share at least one key. The value of δ_m^k can be either accurately calculated as $\prod_{i=0}^{k-1}(m-k-i)/(m-i)$, or approximated using Stirling's approximation for $n!$ as in [9], i.e.,

$$\delta_m^k = \frac{\binom{m-k}{k}}{\binom{m}{k}} \approx \frac{\left(1-\frac{k}{m}\right)^{2(m-k+0.5)}}{\left(1-\frac{2k}{m}\right)^{m-2k+0.5}}$$

Then the probability of A and B sharing at least one key is

$$P(A \sharp B) = 1 - \delta_m^k \tag{23.3}$$

For example, if $k = 83,\ m = 10000,\ P(A \sharp B) \approx 50\%$.

Next, we derive the value of SAP based on the number of authorized users and compromised users within both A and B's communication range. We start with the simplest case, where there is only one compromised node available. We then consider the case where there are h compromised nodes. Finally, we consider the case with h compromised nodes and g authorized nodes.

23.2.2.1 Scenario I: Only One Compromised Node C is Within Both A and B's Communication Range

Depending on whether A and B share a key initially, they may establish the pairwise link with or without the relay of C. The probability of successfully establishing the link is (23.4) and the probability of attacking the link is (23.5).

$$P(A \leftrightarrow B) = P(A \sharp B) + P((A \sharp C \cap B \sharp C) \cap \overline{A \sharp B}) \tag{23.4}$$

$$P(A \otimes B) \geq P((A \sharp B) \lhd C) + P((A \sharp C \cap B \sharp C) \cap \overline{A \sharp B}) \tag{23.5}$$

Here $((A \sharp B) \lhd C)$ means that A and B share at least one key, and all the shared keys between A and B are within the key ring of node C. Since we ignore the case where C only knows a subset of the shared keys between A and B, where C still has a chance to successfully attack the link between A and B, we have an inequality in (23.5) instead of an equality.

Let us calculate each term in (23.4) and (23.5). We know the value of $P(A \sharp B)$ from (23.3). Also,

$$\begin{aligned}
&P(A \sharp C \cap B \sharp C | \overline{A \sharp B}) \\
&= 1 - P(\overline{A \sharp C}) - P(\overline{B \sharp C}) + P(\overline{A \sharp C} \cap \overline{B \sharp C} | \overline{A \sharp B}) \\
&= 1 - 2\delta_m^k + \binom{m-2k}{k} \Big/ \binom{m}{k} \\
&= 1 - 2\delta_m^k + \binom{m-k}{k} \Big/ \binom{m}{k} \cdot \binom{m-2k}{k} \Big/ \binom{m-k}{k} \\
&= 1 - 2\delta_m^k + \delta_m^k \cdot \delta_{m-k}^k
\end{aligned}$$

Define

$$\phi_m^k \triangleq P(A\sharp C \cap B\sharp C | \overline{A\sharp B})$$

we then have

$$P(A\sharp C \cap B\sharp C \cap \overline{A\sharp B}) = P(\overline{A\sharp B}) \cdot P(A\sharp C \cap B\sharp C | \overline{A\sharp B}) = \delta_m^k \phi_m^k \tag{23.6}$$

Thus from (23.3), (23.4), and (23.6),

$$P(A \leftrightarrow B) = 1 - \delta_m^k + \delta_m^k \phi_m^k$$

Meanwhile,

$$\begin{aligned}
&P((A\sharp B) \lhd C) \\
&= \sum_{i=1}^{k} \left(\binom{k}{i} \cdot \left(\frac{\binom{m-k}{k-i}}{\binom{m}{k}} \right) \cdot \left(\frac{\binom{m-i}{k-i}}{\binom{m}{k}} \right) \right) \\
&\geq \binom{k}{1} \cdot \left(\frac{\binom{m-k}{k-1}}{\binom{m}{k}} \right) \cdot \left(\frac{\binom{m-1}{k-1}}{\binom{m}{k}} \right) \\
&= k \left(\frac{k}{m-2k+1} \cdot \frac{\binom{m-k}{k}}{\binom{m}{k}} \right) \cdot \left(\frac{\dfrac{(m-1)!}{(k-1)!(m-k)!}}{\dfrac{m!}{k!(m-k)!}} \right) \\
&= \frac{\delta_m^k k^3}{m(m-2k+1)}
\end{aligned} \tag{23.7}$$

whereas in (23.7), for simplicity we ignore the event that A, B, and C share more than one key. Define

$$\gamma_m^k \triangleq P((A\sharp B) \lhd C)$$

we then have

$$SAP = \frac{P(A \otimes B)}{P(A \leftrightarrow B)} \geq \frac{\gamma_m^k + \delta_m^k \phi_m^k}{1 - \delta_m^k + \delta_m^k \phi_m^k}$$

23.2.2.2 Scenario II: *h* Compromised Nodes are Within Both A and B's Communication Range

We use $\mathbb{C}^h$ to denote the set of h compromised nodes. Since

$$\begin{aligned}
&P((A,B)\sharp\mathbb{C}^h \cap \overline{A\sharp B}) \\
=&P(\overline{A\sharp B}) \cdot P((A,B)\sharp\mathbb{C}^h|\overline{A\sharp B}) \\
=&P(\overline{A\sharp B}) \cdot (1-(1-P(A\sharp C \cap B\sharp C|\overline{A\sharp B}))^h) \\
=&\delta_m^k \cdot \left(1-\left(1-\phi_m^k\right)^h\right)
\end{aligned}$$

then using a similar argument as in Scenario I, we have

$$\begin{aligned}
SAP &\geq \frac{P((A\sharp B) \lhd \mathbb{C}^h) + P((A,B)\sharp\mathbb{C}^h \cap \overline{A\sharp B})}{P(A\sharp B) + P((A,B)\sharp\mathbb{C}^h \cap \overline{A\sharp B})} \\
&\geq \frac{1-\left(1-\gamma_m^k\right)^h + \delta_m^k \cdot \left(1-\left(1-\phi_m^k\right)^h\right)}{1-\delta_m^k + \delta_m^k \cdot \left(1-\left(1-\phi_m^k\right)^h\right)}
\end{aligned}$$

23.2.2.3 Scenario III: *h* Compromised Nodes and *g* Authorized Nodes are Within Both A and B's Communication Range

In this case, if A and B do not share any key initially and need to communicate through a relay, a successful attack can happen if one compromised node is chosen as the relay. Assuming there are a total of *a qualified* relays (i.e., nodes who can establish pairwise links with both A and B), b out of which are compromised nodes. Denote μ_a^b as the probability of A and B picking a compromised node as the relay, which can have different values depending on the specific attack models (details in the next section).

The probability of having r useable relays out of all h compromised nodes when A and B do not share keys is

$$\begin{aligned}
P\left((A,B)\sharp\mathbb{C}_r^h|\overline{A\sharp B}\right) &= \binom{h}{r}\left(P(A\sharp C \cap B\sharp C|\overline{A\sharp B})\right)^r \left(1-P(A\sharp C \cap B\sharp C|\overline{A\sharp B})\right)^{h-r} \\
&= \binom{h}{r}\left(\phi_m^k\right)^r \left(1-\phi_m^k\right)^{h-r}
\end{aligned}$$

Similarly, the probability of having w useable relays out of all g authorized nodes when A and B do not share keys is

$$P\left((A,B)\sharp\mathbb{C}_w^g|\overline{A\sharp B}\right) = \binom{g}{w}\left(\phi_m^k\right)^w \left(1-\phi_m^k\right)^{g-w} \tag{23.8}$$

Then the probability of sending a message through a compromised node given the existence of h compromised nodes, g authorized nodes, and A and B do not share any key is

$$\begin{aligned}
&P(A \leftrightarrow \mathbb{C}^h \leftrightarrow B|\overline{A\sharp B}) \\
&= \sum_{r=1}^{h}\sum_{w=0}^{g} \mu^r_{r+w}\left(P\left((A,B)\sharp\mathbb{C}^h_r|\overline{A\sharp B}\right)\cdot P\left((A,B)\sharp\mathbb{C}^g_w|\overline{A\sharp B}\right)\right) \\
&= \sum_{r=1}^{h}\sum_{w=0}^{g} \mu^r_{r+w}\left(\binom{h}{r}\binom{g}{w}\left(\left(\phi^k_m\right)^{r+w}\left(1-\phi^k_m\right)^{h+g-(r+w)}\right)\right)
\end{aligned}$$

Since

$$P(A \leftrightarrow \mathbb{C}^h \leftrightarrow B \cap \overline{A\sharp B}) = P(A\sharp B)\cdot P(A \leftrightarrow \mathbb{C}^h \leftrightarrow B|\overline{A\sharp B})$$

we have the following lower bound on SAP:

$$\begin{aligned}
SAP &= \frac{P(A\otimes B)}{P(A\leftrightarrow B)} \\
&\geq \frac{P((A\sharp B)\lhd\mathbb{C}^h) + P(A\leftrightarrow\mathbb{C}^h\leftrightarrow B\cap\overline{A\sharp B})}{P(A\sharp B)+P(A\leftrightarrow\mathbb{C}^{h+g}\leftrightarrow B\cap\overline{A\sharp B})} \\
&= \frac{1-\left(1-\gamma^k_m\right)^h + \delta^k_m\cdot\left(\sum_{r=1}^h\sum_{w=0}^g \mu^r_{r+w}\left(\binom{h}{r}\binom{g}{w}\left(\left(\phi^k_m\right)^{r+w}\left(1-\phi^k_m\right)^{h+g-(r+w)}\right)\right)\right)}{1-\delta^k_m+\delta^k_m\cdot\left(1-\left(1-\phi^k_m\right)^{h+g}\right)}
\end{aligned}$$

23.2.2.4 Numerical Results

Table 23.2 shows the SAP for different values of h and g based on the previous analysis. The key ring size is $k = 83$, with a key pool size of $m = 10,000$.

Table 23.2 Successful attack probability (SAP) for different numbers of authorized nodes (g) and compromised nodes (h). We assume there are a total of a qualified relays, b out of which are compromised nodes. μ^b_a is the probability of picking a compromised node as the relay. The key pool size $m = 10000$, the preloaded key ring size $k = 83$, and the original SAP estimation is hk/m

h	$g = 0$	$g = 10$		$g = 20$		hk/m
		$\mu^b_a = b/a$	$\mu^b_a = 1$	$\mu^b_a = b/a$	$\mu^b_a = 1$	
1	20.4%	4.7%	13.0%	2.7%	12.8%	0.8%
2	31.1%	8.8%	22.7%	5.1%	22.4%	1.7%
3	37.6%	12.3%	30.0%	7.4%	29.7%	2.5%
4	41.9%	15.3%	35.5%	9.5%	35.2%	3.3%
5	44.8%	18.0%	39.7%	11.4%	39.5%	4.2%
6	46.9%	20.4%	42.9%	13.2%	42.7%	5.0%
7	48.5%	22.5%	45.4%	15.0%	45.2%	5.8%
8	49.7%	24.4%	47.3%	16.6%	47.2%	6.6%
9	50.6%	26.2%	48.8%	18.1%	48.7%	7.5%

Several observations are in order. When the probability of picking a compromised node as the relay $\mu_a^b = b/a$, the SAP increases with h (the number of compromised nodes) under a fixed g (the number of authorized nodes). When $\mu_a^b = 1$, the general trend is similar, but the SAP is not very sensitive to g between the cases of $g = 10$ and $g = 20$, since A and B will always choose a compromised node as relay if possible. Comparing with the value of SAP estimated in [9], which is approximated as hk/m, the SAP in Table 23.2 is much larger. For example, with $\mu_a^b = b/a$, $h = 9$ and $g = 20$, we have an SAP of 18.1%, as opposed to $hk/m = 7.5\%$. The value of SAP further increases when $\mu_a^b = 1$.

The value of μ_a^b depends heavily on the attack model used by the compromised nodes. We define two attack models, *honest attack* and *smart attack*. In an honest attack, the relay nodes are randomly chosen and $\mu_a^b = b/a$. In a smart attack, however, the compromised nodes will improve the value of μ_a^b by various methods. In a *smart attack with incentive*, the compromised nodes provide incentives for nodes A and B to choose one of them as a relay. If the choice of relay is determined by a shortest path routing protocol, the compromised nodes can announce distance metrics of the links connected to them smaller than the actual values. If the choice of relay is based on energy efficiency, the compromised nodes can pretend to be very energy efficient. In most cases, the incentives provided by the compromised nodes can make the value of μ_a^b very close to 1. In *a smart attack with virtual node fabrication*, each compromised node is able to collect the keys from all other compromised nodes and can then fabricate up to $\binom{hk}{k}$ nodes with distinct key rings. The number will be very large if $h \geq 2$. For example, when two nodes are captured with non-overlapping key rings, then

$$\binom{2k}{k} = \frac{(2k)!}{k!} \approx \frac{\sqrt{2\pi}(2k)^{2k+0.5}e^{-2k}}{\left(\sqrt{2\pi}(k)^{k+0.5}e^{-k}\right)^2} = \frac{2^{2k+0.5}}{\sqrt{2\pi k}} \tag{23.9}$$

which is around 5.8×10^{48} if $k = 83$. As a result, the value of μ_a^b will be closer to 1 as the number of fabricated nodes increases.

23.3 Secret-Protecting Processor Architecture

The analysis in the previous section is based on the assumption that an adversary can obtain the long-term key information from the captured nodes. In particular, we assume an adversary with physical access to the device, so software protections are easily bypassed. The keys are accessible to an adversary if the existing software is exploited or if the software is replaced entirely with malicious code. He might also read the keys directly from a flash memory chip or other permanent storage when the device is offline.

We propose a solution that protects secrets by storing them inside the System on a Chip (SoC) [8]. The chip includes the processor core and main memory. It is quite expensive for an adversary to remove the packaging and directly probe the registers and memory. The SoC chip can further implement physical tamper-resistance mechanisms, which will clear the secrets whenever probing attempts are detected, cut the power supply to the chip, and erase any intermediate data based on those secrets. Therefore, the assumption of protected on-chip secrets is valid for a large class of attacks.

Our solution is to provide a Secret Protecting (SP) architecture which minimizes the trusted computing base (TCB) of hardware and software that has to be fully correct, verified, and trusted in order to protect the long-term secrets. Our TCB comprises some SP hardware features (described below) and a small Trusted Software Module (TSM) that performs the key management on the node.

We first present the Reduced Sensor-mode SP that is suitable for the simplest sensors. We then extend the solution for slightly more capable sensors. Our work is inspired by the SP architecture proposed for general-purpose microprocessors [7, 20], but stripped to the bare minimum for sensors with very constrained computing and storage resources.

23.3.1 Reduced Hardware Architecture

The simplest version of our architecture, Reduced Sensor-mode SP, is shown in Fig. 23.1. It only requires one new register — the Device Key—and one bit to indicate protected mode. A Trusted Software Module (TSM) is stored in the on-chip instruction EEPROM. The long-term keys for the probabilistic key management scheme, provided by a central authority, are stored in the on-chip data EEPROM. A portion of the main memory of the node is reserved for the TSM Scratchpad Memory.

The key feature of our design is that the TSM is the only software module that can use the Device Key and the protected long-term keys. Since the TSM code is stored within the trusted SoC chip in ROM, it cannot be changed by other software—whether by a malevolent application or by a compromised operating system. Similarly, the long-term keys never leave the SoC chip. Any intermediate data (which may leak key bits) generated during TSM execution are placed in the TSM scratchpad memory and also never leave the SoC chip. We will discuss how this prevents node fabrication attacks in Sect. 23.4.

The TSM code is stored on-chip in a segment of the existing instruction EEPROM along with other system software for the node. Correspondingly, the long-term keys from the authority are stored in a TSM segment of the data EEPROM. The keys are encrypted with the device key or with another encryption key derived from it by the TSM.

The device key is the SP master key and is protected by the processor hardware; it can only be used by the TSM running in protected mode and can never be read by any other software.

When the unprotected software wants to make use of protected keys, it calls the TSM. The TSM functions access the protected keys, perform the requested

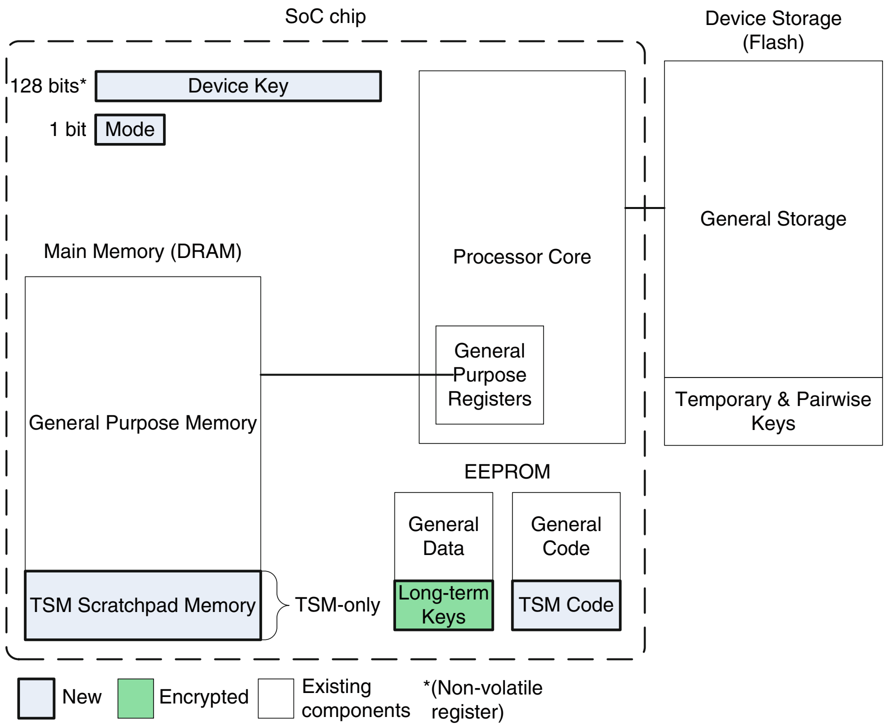

Fig. 23.1 Reduced sensor-mode SP

operation, and return the results, never revealing the protected keys themselves to the unprotected software. Each TSM function starts with a *Begin_TSM* instruction, which disables interrupts, sets the protected mode bit, and enters protected mode for the next instruction. *Begin_TSM* is only valid for code executing from the instruction-EEPROM; any code executed from main memory or off-chip storage cannot enter the protected mode at all. The end of the TSM code is indicated by the *End_TSM* instruction which clears the mode bit and re-enables interrupts. Table 23.3 shows the set of instructions used only by the TSM and for initialization in the Sensor-mode SP architectures.

The TSM Scratchpad Memory is a section of main memory reserved for the exclusive use of the TSM. It is addressed separately from the regular on-chip memory and accessed only with special *Secure_Load* and *Secure_Store* instructions (see Table 23.3). These new instructions are available only to the TSM, making it safe for storing sensitive intermediate data in the TSM scratchpad memory. The TSM can also use this extra space to spill general registers, to decrypt and store keys, and to encrypt data for storage in regular unprotected memory.

Initialization of a new device takes place at the authority's depot. First the authority must generate a new random device key. Long-term keys and other secrets are encrypted with it are then stored along with the TSM code on the on-chip EEPROM.

Table 23.3 New sensor-mode SP instructions

Instruction	Description
Begin_TSM	Begins execution of the TSM
End_TSM	Ends execution of the TSM
Secure_Store	Secure store from processor to TSM scratchpad memory (TSM only)
Secure_Load	Secure load from TSM scratchpad memory to processor (TSM only)
DeviceKey_Read	Read the Device Key (TSM only)
DeviceKey_Set	Sets the Device Key register. First clears the TSM scratchpad memory
ASH_Set	Sets the ASH register. First clears the device key and TSM scratchpad memory

Next it uses the *DeviceKey_Set* instruction to store the device key. Finally, any other unprotected software and data can be copied to the flash storage.

Any time the Device Key register is set (or cleared), the processor will automatically clear the TSM scratchpad memory, wiping any intermediate data that are protected by the old key. If in protected mode at the time, the mode bit is also cleared along with the general-purpose registers. Similarly, the processor will clear the device key upon writing to either the instruction or data EEPROM; this in turn clears the other intermediate data.

23.3.2 Expanded Sensor-Mode SP Architecture

The Reduced Sensor-mode SP architecture is ideal for the smallest sensor nodes which use minimal software and have very limited resources. In slightly larger lightweight sensor nodes, the software will be more complex. The additional applications that run on this sensor combined with the TSM and long-term keys will be too large to store on-chip. This greater flexibility in the sensor also requires additional support for security. Hence, we propose the Expanded Sensor-mode SP architecture shown in Fig. 23.2.

The TSM code and encrypted long-term keys are moved to the off-chip device storage. This makes them susceptible to modification by other software or through physical attacks. Therefore we must verify their integrity before they can be used. To do this, we add a new register—the Authority Storage Hash (ASH), a hardware hashing engine (implementing SHA-1, MD5, or another cryptographic hash function), a small ROM, and an additional initialization instruction.

The ASH register contains a hash over the entire memory region of the TSM code and long-term keys. It is set by the authority during initialization and is rechecked by the processor each time the TSM is called. The checking code is stored in the on-chip ROM and is fixed and therefore safe from modification; it uses the hardware hashing engine to compute the hash over the TSM code and the encrypted keys. When *Begin_TSM* is called, the processor disables interrupts and jumps to the TSM-checking routine. If the hash check succeeds, the protected mode bit is set, and execution jumps to the newly verified TSM code. If the check fails, an exception is triggered. The *ASH_Set* instruction sets the ASH register and also causes the device

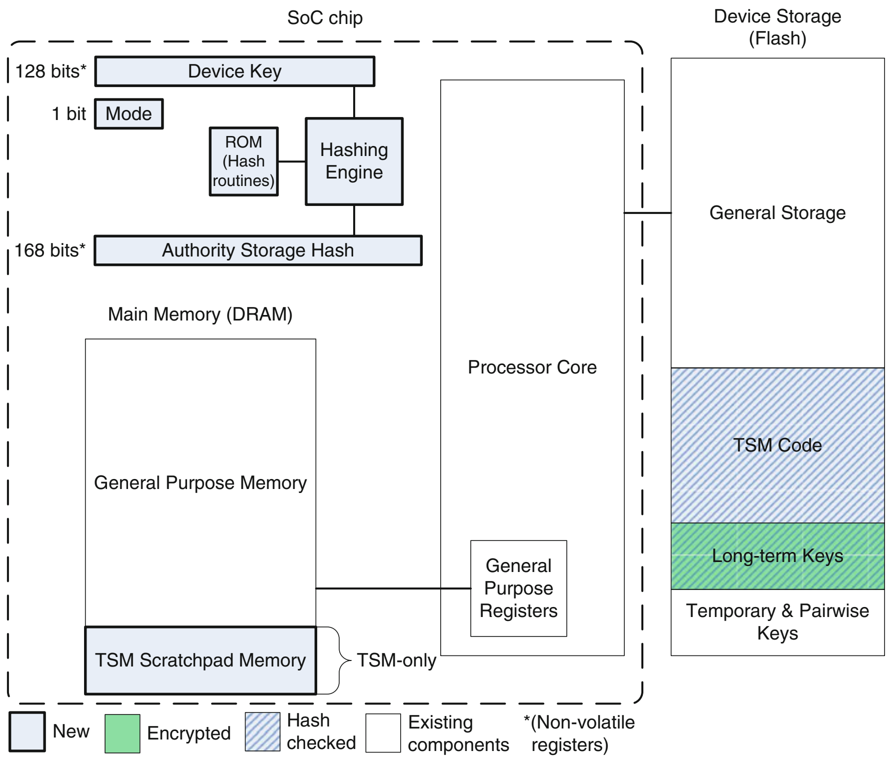

Fig. 23.2 Expanded sensor-mode SP

key register to be cleared. As the ASH is used to verify the TSM code, installing a new TSM requires writing a new value to the ASH value. Clearing the device key therefore ensures that a new TSM cannot be installed and still have access to the protected keys that belonged to another TSM.

23.4 Security and Economics Analysis of SP Architecture-Based Solution

23.4.1 Attacks on Protected Keys

Our new Sensor-mode SP architectures safeguard a sensor node's long-term keys, preventing extraction by an adversary in the event of node capture. The keys are always stored in encrypted form in the permanent storage in either on-chip EEP-ROM or off-chip storage. The adversary cannot obtain the device key needed to decrypt them. The device key never leaves the SP processor or its protected software environment. Therefore, rather than access the keys directly, regular software must call TSM functions which perform operations with the keys on its behalf. Thus, software can use the keys in any way permitted by the TSM, but can never extract the keys themselves even under physical attacks.

23.4.1.1 Node Fabrication Attacks

Without the SP protection, an adversary maximizes his SAP by cloning multiple copies of compromised nodes and combining their long-term keys. This increases his ability to observe link establishment and the likelihood of being used as a relay. With the SP protection, he cannot create any clones and is limited to using only the keys originally stored on the captured node.

23.4.1.2 Node Capture Attacks

Node capture attacks use long-term keys in the node to observe pairwise links between other nodes in the network. With SP, an adversary can no longer extract the keys. However, he can still change unprotected software which calls the TSM. A simple TSM might provide functions like *Encrypt(key, data*) and *Decrypt(key, data)*. The adversary can use the keys through this TSM interface to observe or attack pairwise links without ever seeing the actual keys. While we do not prevent node capture attacks outright, such attacks are limited since the adversary can only observe links within the communication range of the compromised node. We show in Sect. 23.5 that this severely limits the SAP, which is constrained by the number of captured nodes.

23.4.2 Attacks on Changing the TSM or the Device Key

The security of the long-term keys relies on the correctness and proper design of the authority's TSM. As part of the trusted computing base of the system, this software must not leak secrets it has access to. This includes any intermediate data written to general-purpose memory, placed in off-chip storage, or left in general registers when it exits. The TSM runs with interrupts disabled, so no other software will have an opportunity to observe its registers or modify its code or data while it is executing. If the TSM ever exists abnormally due to an exception, the processor clears the general registers before ending protected mode. Any other sensitive data will be in the TSM scratchpad memory which other software cannot access.

In order to circumvent the access control provided by the authority's TSM, the attacker might try to replace it with his own TSM or modify the existing TSM. In the Reduced Sensor-mode, the TSM and long-term keys are stored in on-chip EEPROM where they cannot be modified without also clearing the device key. In the Expanded Sensor-mode, the attacker could modify or replace the TSM code in off-chip storage. The hash checking routine will detect any such modifications made to the TSM before execution. We assume that the data in off-chip storage cannot be modified through a physical attack during execution. If this is not the case, the TSM and keys should first be copied to the general-purpose memory on-chip before being verified, where they will be safe from physical attacks.

Finally, if the attacker tries to modify the ASH register to match the new TSM code, the device key will be cleared, irrevocably cutting off his access to all of the keys that were encrypted with that device key. Clearing or setting the device key also clears the TSM scratchpad memory, so any intermediate data stored there that might have leaked secrets are also unavailable to the new TSM.

23.4.3 Economics Analysis

When considering low-cost sensors, any new hardware must be designed for high volume in order to keep down fabrication costs. Our Sensor-mode SP provides basic security primitives and a hardware root of trust using a design that is easily integrated into the SoC of standard embedded processors. It therefore supports a wide range of software protection mechanisms with only a slight increase in chip area.

Our hardware also provides physical security. SP prevents attacks by an adversary with physical control over a captured node, who tries to modify the code or data in storage while the device is in operation or offline. The physical integrity of the SoC itself is sufficient to prevent adversaries from probing the SP registers inside the chip, without requiring more costly tamper-proofing mechanisms in most cases.

23.5 Simulation Results

To verify our probability computations in Sect. 23.2 and demonstrate the improvement of security performance of the proposed architecture in Sect. 23.3, we evaluate the SAP of the probabilistic and deterministic key predistribution scheme (the EG scheme) through a simulator written in C++.

23.5.1 Comparison of Probabilistic and Deterministic Key Predistribution

To compare probabilistic and deterministic key predistribution schemes, we consider a unit disk network model, as shown in Fig. 23.3. A total of g authorized nodes (denoted by symbol A) are uniformly distributed in the unit disk. All the compromised nodes (including any virtually fabricated nodes) are placed at the center of the unit disk and denoted with symbol C. All nodes are assumed to have the same transmission range equal to the radius of the disk. This means an adversary can eavesdrop on any communication in the unit disk through the compromised nodes as long as it has the right key(s). Two neighbor nodes will set up a pairwise link directly if they share one or more keys. Otherwise, they will try to find a relay path through one or more nodes to exchange additional key information, so that

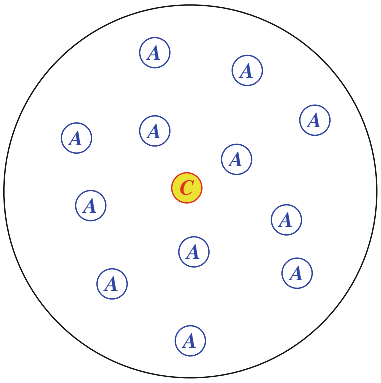

Fig. 23.3 Unit disk network model with a unit radius. The authorized nodes are uniformly distributed in the unit disk and denoted by the symbol A. The compromised and virtually fabricated nodes are placed in the center of the network and denoted by the symbol C. All nodes have the same communication range equal to the disk radius

they can set up pairwise link between them. When there is more than one qualified relay node available, the authorized nodes will choose a relay randomly in the case of $\mu_a^b = b/a$ (i.e., honest attack or finite virtual node fabrication), or search for a shortest relay path in an attack with incentive.[2] Any two nodes that are not neighbors cannot establish pairwise links among themselves. The main reason of using the above unit disk network model is to derive a uniform and fair metric (i.e., SAP) among various approaches where failing to attack is only due to the lacking of appropriate keys rather than the limitation of transmission range.

The SAP is calculated as the fraction of the links (among all the pairwise links) that can be eavesdropped by the compromised nodes. As we explained in Sect. 23.1, a basic deterministic scheme like single common key either enables nearly zero SAP in a static network, or leads to 100% SAP for the unit disk model in a mobile network. Hence, our focus here is to determine the SAP for the probabilistic key predistribution scheme (i.e., the EG scheme). All the simulation results are averaged over 10 sets of random seeds which affect the distribution of the authorized nodes within the unit disk, the key ring preloaded to each node and the choices in case of multiple qualified relays.

Figures 23.5, 23.6, and 23.7 illustrate the values of SAP under different assumptions on the number of compromised nodes (h), number of authorized nodes (g), and different attack models (honest attack, smart attack with incentive, or smart attack with fabrication). Unless otherwise specified, each node is preloaded with a key ring consisting of $k = 83$ keys that are randomly chosen from a key pool of size $m = 10,000$.

Fig. 23.5 shows the SAP for various values of h and g under the *honest attack*. For a fixed value of h, the SAP decreases when the density of authorized nodes increases. This is because in a denser network, there are more qualified relay nodes available between any two neighbor nodes; thus the probability of choosing a compromised node as the relay is smaller under honest attack. For a fixed number of

[2] In the simulation, the smart attack with incentive is approximated as setting the cost of the links adjacent to the compromised nodes as 0.9999 instead of as 1 unit (hop) for other authorized nodes.

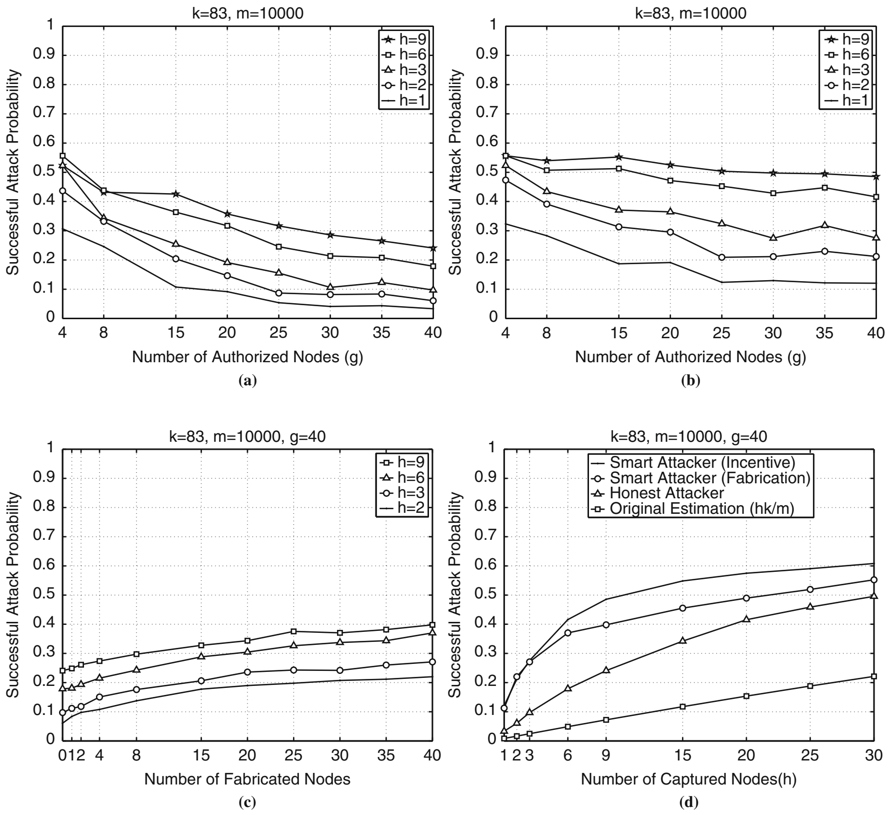

Fig. 23.4 Successful attack probability with various numbers of captured nodes (h) and authorized nodes (g), (**a**) Honest attack; (**b**) Smart attack; (**c**) Smart attack (node fabrication); and (**d**) Different attack models

authorized nodes g, a higher value of h increases the probability of picking a compromised node as the relay, thus leads to a higher value of SAP. In a network with 9 compromised nodes and 15 authorized nodes, the SAP could be as high as 42%.

Figure 23.4(b) shows the SAP for various values of h and g under the *smart attack with incentive*. In this case, two neighbor nodes without a common key will have a high chance to pick a compromised node as relay if it is qualified. There is a high probability of finding a qualified relay node among the compromised nodes when h is large, in which case the SAP is insensitive to the number of authorized nodes g. Similarly as in Fig. 23.5, a higher value of h also leads to a higher value of SAP. In a network with 40 authorized nodes and 9 compromised nodes, the SAP would be around 50%.

Figure 23.6 shows the SAP for the smart attack of various numbers of compromised nodes and different total numbers of virtually fabricated nodes. The total number of authorized nodes is kept at 40. The node fabrication is achieved as follows. All the keys collected from the h compromised nodes will constitute a

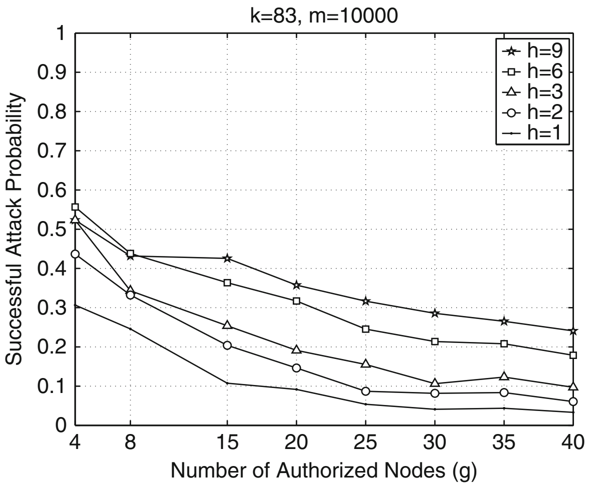

Fig. 23.5 Successful attack probability for various numbers of captured nodes (**h**) and authorized nodes (**g**) under an honest attack

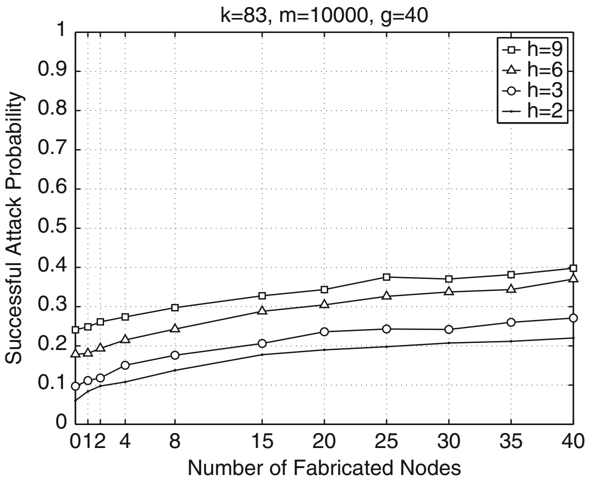

Fig. 23.6 Successful attack probability for various numbers of captured nodes (**h**) and different numbers of node fabrications in smart attack

compromised key pool. Then each fabricated node will be loaded with $k = 83$ keys randomly chosen from the compromised key pool. A larger number of fabricated nodes increase the chance of such a node being chosen as a relay node, thus increasing SAP. A larger value of h leads to a larger compromised key pool, which again increases the chance of a fabricated node serving as a qualified relay.

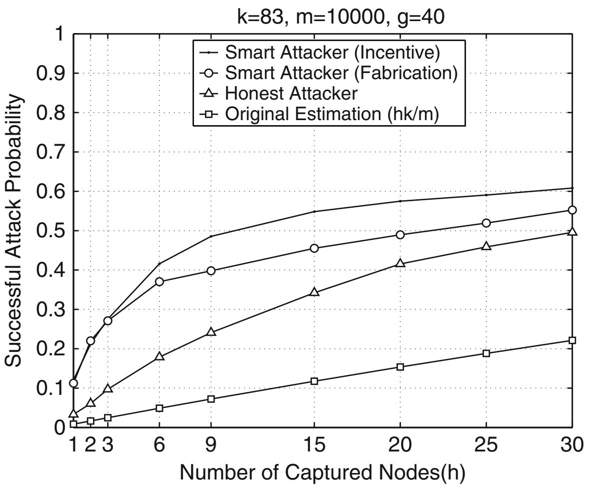

Fig. 23.7 Successful attack probability under different numbers of captured nodes (**h**), for different attack models as well as the estimation in [9].

Figure 23.7 shows the SAP under different numbers of captured nodes, for different kinds of attacks, as well as the estimation based on the result in [9][3]. The number of authorized nodes is fixed at 40. It is clear that the results in [9] significantly underestimate the SAP in mobile networks. With a large enough number of compromised nodes, the SAP can easily reach an unacceptably high value of 50% with all attack models.

Figure 23.9 shows the SAP under different sizes of the preloaded key ring, k, for different attack models as well as the estimation based on the results in [9]. We also plot the link connectivity (i.e., the probability that two neighbor nodes share at least one key, $1 - \delta_m^k$) under different values of k. With the increase in k, the link connectivity increases, as well as the SAP estimation based on the analysis in [9], which is linear in k. On the other hand, the SAPs for all three attack models actually decrease with an increasing k, due to less need of going through a relay to establish a pairwise link. However, they are still much higher than the original estimation of SAP in [9] and the nodes need more memory to store so many keys.

23.5.2 Security Improvement with SP architecture

In this section, we show the security performance of the proposed SP architecture for lightweight ad hoc networks. We focus on the evaluation of the basic probabilistic

[3] When the network is static, an adversary captures h nodes, then its successful attack probability on a link is $1 - \left(1 - \frac{k}{m}\right)^h \approx \frac{hk}{m}$. if $\frac{k}{m}$ is small

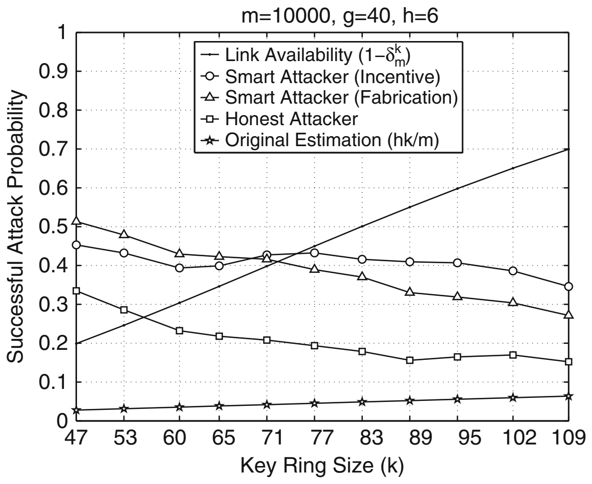

Fig. 23.8 Successful attack probability for different key ring sizes (**k**), for different attack models as well as the estimation in [9].

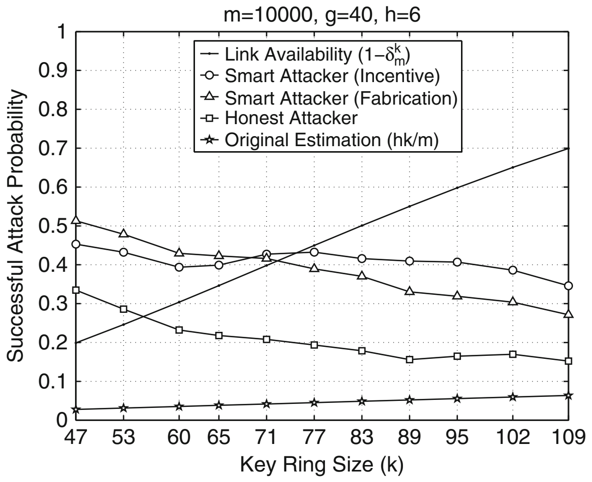

Fig. 23.9 Successful attack probability for different key ring sizes (k), for different attack models, as well as the estimation in [9]

key predistribution approach (the EG scheme) since the deterministic approach (e.g., single common key) is a special case of the probabilistic approach. In addition, many advanced versions of probabilistic key predistribution (e.g., [4, 6]) are also vulnerable to node capture attacks and can benefit from the proposed architecture.

We have run the simulation for a 10×10 grid network, with all nodes assumed to have the same (1 unit) transmission range. A total of 400 nodes are randomly placed in the network. Network-wide SAP is calculated as the fraction of links that can be intercepted by the compromised nodes among all the pairwise links established among the authorized nodes. Again, all simulation results are averaged over 10 sets of random seeds that affect the distributions of the location of each node, the key rings preloaded to nodes, and the relay choices.

We consider several possible attack models depending on whether the SP architecture is used. If every node is equipped with the Sensor-mode SP architecture, the adversary can only launch a *node capture attack*, where the adversary utilizes the captured nodes themselves to intercept pairwise-key establishment. Without the SP architecture, the adversary can further launch *node fabrication attacks* where he can turn the captured nodes into super-nodes by loading each of them with all of the keys from all captured nodes. Each super-node can mimic multiple nodes. A straightforward method to achieve this is to let each super-node stay at its original location but announce the existence of all the captured nodes. The adversary can even make more copies of the super-nodes and deploy them into the network to eavesdrop additional communication. We note that it is difficult to detect the duplication of nodes within the network, since it requires knowledge of the location of each node (possibly using Global Positioning System) and non-trivial communication and memory overhead [26].

Figure 23.10 shows the network-wide SAP under different numbers of captured nodes for different kinds of attacks. "SP" means launching only the node capture attack with the SP architecture. "0 copies" means changing captured nodes into super-nodes (i.e., node fabrication attack) due to the lack of the SP architecture. "*x* copies" means making *x* extra copies of these super-nodes elsewhere in the network. We note that the effect of node capture can be serious without SP. When only 3% of the nodes are captured, the SAP for the network will be 9.7% even with "0 copies", and becomes 42.6% if the adversary makes six copies of the captured nodes to cover more area. Whereas the SAP for the nodes with SP is only 2.1%—a reduction by

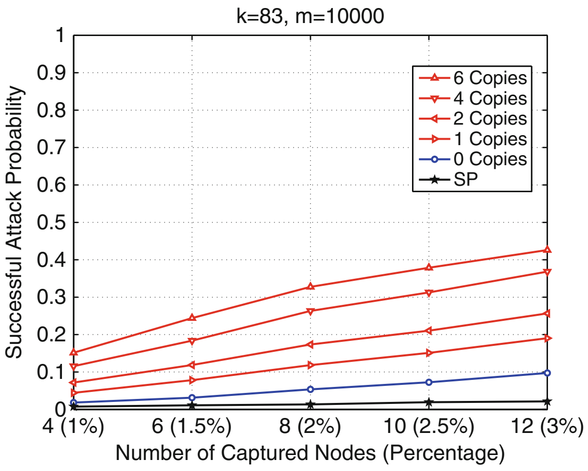

Fig. 23.10 Network-wide successful attack probability under different numbers of captured nodes for different attack models

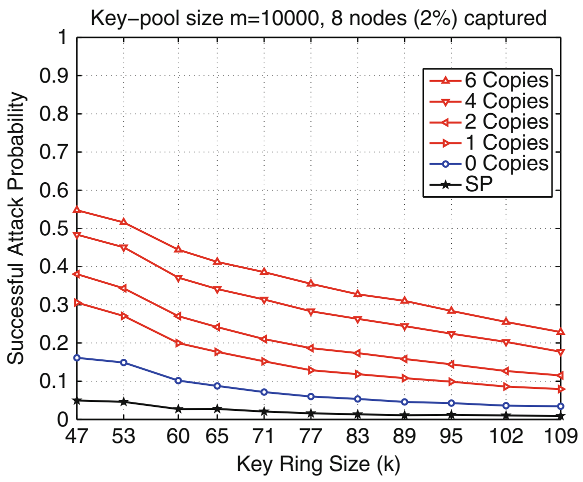

Fig. 23.11 Network-wide successful attack probability for different key ring sizes (k) with different attack models

roughly an order of magnitude. Therefore, SP provides significant benefits in terms of alleviating node fabrication attacks.

Figure 23.11 shows the network-wide SAP under different sizes of the preloaded key ring, k, for different attack models, assuming 2% of nodes have been captured. An increasing value of k has two effects on the network. First, the link connectivity increases; this reduces the probability of two neighboring nodes establishing a pairwise link through a relay node, and thus can improve the network security. Second, each node captured by the adversary contains more keys, which will increase the chance of intercepting the communications on other pairwise links. This is detrimental to the network security. Figure 23.11 shows that the advantage of the first effect dominates and the overall SAP decreases with an increasing value of k. Notice that the SP architecture offers significant advantages over the other schemes for all values of k.

Finally, the single common key scheme also benefits from SP. This is because the adversary, without the ability to learn the common key, can only eavesdrop on the information exchanged within the communication range of the captured nodes.

23.6 Implications to Related Work

23.6.1 Reinforcements on the Basic EG Scheme

Many probabilistic schemes based on the EG scheme have been proposed, e.g., [4, 5, 23, 27, 32]. We show that many of them are also very vulnerable to node capture in the mobile networks, or that the proposed improvements in those schemes can benefit deterministic approaches as well.

In the q-composite key scheme [4], two nodes can only establish a pairwise link between them if they share at least q keys initially (i.e., within the preloaded key rings). The real key for encrypting the communication is a hash result of all q keys. Though this approach can reduce SAP in a static network, it is almost as fragile as the EG scheme in a mobile network, following similar analysis as in Sect. 23.2. In addition, to keep the link connectivity comparable to the EG scheme, the key ring size has to be substantially increased, which means fewer node captures are needed to disclose a sufficiently large key space to the adversary.

The concept of multiple disjoint path key reinforcement was proposed in [4] and [32]. A node will send partial keys to its neighbor through several disjoint paths. The counterpart then regenerates the original key after receiving all these partial keys. A compromised node can only regenerate the key if it intercepts all partial keys. However, this approach requires each node to maintain a global network topology to calculate disjoint paths, and the ability to do source routing. In addition, if virtual node fabrication is possible, there is still a high probability that every path passes through a compromised node. Finally, the approach is also fragile to the attack of deliberate modification by a compromised node along the path, so that the neighbor node cannot successfully regenerate the key.

Some schemes utilize the "partial" deployment information to increase the resilience against node capture (e.g., [5, 23]). In particular, two nodes have a higher probability of sharing keys if they are supposed to be deployed in a group (e.g., in the same geographic area), or have a lower probability if they will be deployed in different groups. Therefore, a node captured in a particular group will have little effect on the security of nodes in other groups. However, the same technique can also be used to enhance the security of a deterministic scheme. For example, a different common key can be assigned to the nodes deployed in the same group, and a node is then equipped with all the keys of the groups it belongs to. Thus capturing one node will not have much adverse effect on the nodes in other groups. Therefore, such improvements do not change the nature of our comparisons of deterministic and probabilistic key management.

23.6.2 Selective Node Capture

So far we have only considered *random node capture*, i.e., the adversary randomly captures nodes to collect sufficient keys to attack the whole network. Another new attack mode, called *selective node capture*, can further weaken the security levels of the probabilistic approaches. As originally proposed in [27], the adversary can listen to the information exchanges when each node tries to identify the keys to be shared with its neighbors. By identifying the key indices in each node and physically locating any node, the adversary can selectively capture nodes with the least overlap in keys. Compared with random node capture, fewer selective node captures are needed to disclose a certain number of unique keys. Although several methods have been proposed to reduce the communication overhead and avoid unnecessary key index disclosure, none of them can completely preclude selective node capture.

One class of key discovery methods is “key indices notification.” A basic approach is that each node announces all of the key indices, with a message size of $O(k)$, as in [1]. Zhu et al. [32] proposed a refined approach that uses a publicly known pseudo-random key index generation function. In the key predistribution phase, the authority’s server chooses a random seed for each node, which calculates a set of k outputs (i.e., key indices) using the key index generation function. The node then uses the random seed as its ID and announces this ID to its neighbors in the key discovery phase. Each of its neighbors can figure out the key indices stored in the node from its ID, since they all have the same key index generation function. As a result, the communication overhead is only of order $O(1)$. However, since the adversary can also figure out all the key index information from the announcements, the scheme is still fragile to selective node capture.

A more complicated challenge-response-like key discovery technique was proposed in [6, 9, 23], where a node can determine whether its neighbor has a particular key if and only if it also has the key. In this scheme, a node announces k challenges separately encrypted by its own k keys. A neighbor node then tries to decrypt the k challenges with its own k keys. Only after the successful decryption of a challenge, can the neighbor node figure out the key from the node who announced the challenge. The process involves k^2 decryption operations and $O(k)$ message exchanges between any two neighbor nodes. To further reduce the computation and communication overhead, Pietro et al. [27] designed a verification function $\Phi(ID\|key)$ using a one-way hash function, like SHA-1, where $\|$ is the concatenation operation. The function $\Phi(ID\|key)$ returns “true” for any argument (i.e., $ID\|key$) with a (pseudo-random) probability of k/m. In the key distribution phase, each node uses $\Phi(ID\|key)$ to test each key k_i from the key pool with its ID and uses that key only if the result is “true”. Therefore, around k keys out of an m sized key pool will be stored on each node. In the key discovery phase, the node just needs to announce its ID, and a neighbor node will execute $\Phi(ID\|k_i)$ k times (i.e., with all its own k keys and the ID it hears) to discover the shared key(s). In short, the procedure involves only k hashing operations and $O(1)$ message exchange between any two neighbor nodes. However, these challenge-response-like methods are still subject to selective node capture. The major difference here is that the selective node capture is only more meaningful than random node capture in sequential node capture mode. That is, in each step, the adversary identifies and captures the node with the fewest keys existing in the compromised key pool. While in “key indices notification” methods, the adversary can identify a set of nodes with the least key overlap and capture them concurrently.

23.7 Key Establishment Approach

As discussed in Sect. 23.1, key pre-distribution schemes have to struggle with the conflicts among node resource limits, desired key-connectivity probability, scalability in network size, and resilience against malicious attacks. Due to the limitation of node memory and computation ability, key predistribution schemes scale poorly in very large networks and the resulting pairwise key-connectivity probability is relatively low. In order to provide an end-to-end key to any communicating node pair, on-demand key establishment becomes a necessary approach. From a security perspective, most key predistribution schemes are designed to protect only the confidentiality of secret keys, while two other security components, integrity and availability, are not accounted for. Key pre-distribution schemes are vulnerable when various attacks occur simultaneously.

To address these issues, a key establishment approach that employs pre-distributed keys as local link keys has been proposed in [4, 14, 32]. The problem is similar to the verifiable secret sharing [10, 11] in cryptography literature, where most existing algorithms rely on complicated algebraic operations, and thus are unsuitable for ad hoc network applications under computation constraints. In the key establishment approach, to set up an end-to-end secret key between two nodes, the source node generates a set of keying messages, from which a secret key can be derived. Each keying message is sent through a different communication path from the source node to the destination node, which then computes the secret key locally. The transmission is protected by existing link keys at each hop. Since it is difficult to attack a large fraction of keying messages simultaneously in an ad hoc network, the key establishment approach using multi-path is able to guard against various attacks efficiently. In particular, an XOR-based key establishment scheme was proposed in [4, 32], where a secret key is derived by the XOR of all keying messages. This scheme prevents malicious attackers from deriving the secret key if not all keying messages are revealed. In [14], Huang et al. proposed a Reed-Solomon code-based scheme that allows node pairs to derive secret keys when both erasure and modification of keying messages occur. In a closely related problem known as secret sharing [29], it is shown that there exists a scheme to divide a secret into n messages in such a way that the key is easily reconstructable from any $r + 1$ pieces, but even complete knowledge of r pieces reveals no information about the secret. When applied to sensor networks, this technique enables the construction of a key establishment scheme that can guard against both revealing and erasure of keying messages.

However, these key establishment schemes only deal with a subset of the following three attacks, in which malicious nodes (i.e., compromised or fabricated nodes by attackers) can (a) reveal the keying messages passing through them to make secret keys computable to the attackers; (b) erase and not-forward keying messages to prevent other nodes from establishing secret keys; or (c) modify the forwarded keying messages to prevent other nodes from deriving the correct secret keys. These attacks violate the three security properties, confidentiality, availability, and integrity of the keying messages, respectively. To provide a unifying analytical framework for key establishment, in [18] the authors proposed a novel security metric, called the REM resilience vector to quantify the resilience of any key establishment scheme against Revealing, Erasure, and Modification (REM) attacks. Relying on the new security metric, a universal bound on achievable REM resilience vectors was proven for any on-demand key establishment scheme. This bound that characterizes the optimal security performance analytically is shown to be tight, using a REM-resilient key establishment scheme which achieves any vector within this bound. In addition, a low-complexity key establishment scheme which achieves nearly optimal REM attack resilience has been developed in [18].

The remaining parts of this section are organized as follows: In Sect. 23.7.1, we first introduce the analytical framework for key establishment developed in [18], and analyze the security of [4, 14, 29, 32] under the framework. The result in [18] that characterizes the optimal security performance is summarized in Sect. 23.7.2, followed by the low-complexity key establishment scheme in Sect. 23.7.3. Section 23.7.4 contains a simulation, which compares the security performance of all key establishment algorithms.

23.7.1 An Analytical Framework for Key Establishment

Consider a wireless ad hoc network where nodes are not tamper resistant. Compromised or fabricated nodes may reveal all their forwarded keying messages to attackers and also try to disrupt normal key establishment in the network.

In Fig. 23.12, an end-to-end secret key is provided for nodes S and D, who do not share a common key from key predistribution. The procedure is described as follows. After receiving a request message, source node S first employs a network routing protocol and finds m paths (which can be non-disjoint) to the destination node D. Then n keying messages, denoted by $M_1, \ldots, M_n$, are generated by the source node and sent to the destination node, each via a different path, i.e., message M_i is send via path $(S, R_{i,1}, R_{i,2}, \ldots, D)$. To secure keying messages during transmission, encryptions by existing link keys are performed at each intermediate node before forwarding keying messages, and nodes at the next hop decrypt the messages with the same link keys. More precisely, the following message is sent from node $R_{i,j}$ to node $R_{i,j+1}$:

$$R_{i,j} \to R_{i,j+1} : E\left[M_i, K_{i,j}^{i,j+1}\right]$$

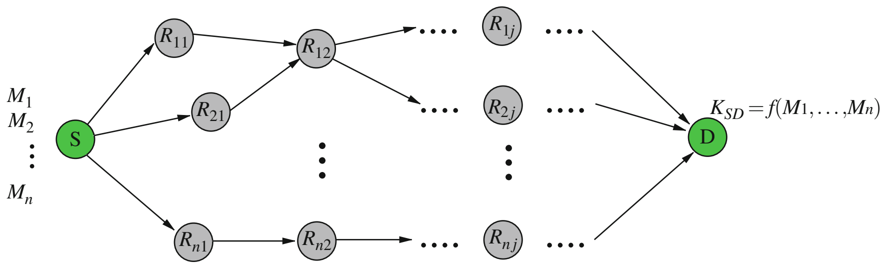

Fig. 23.12 A general key establishment where n messages are sent from the source node S to the destination node D

where $E[\cdot]$ denotes the encryption function and $K_{i,j}^{i,j+1}$ is a link key from key predistribution. Upon receiving the keying messages, node D employs a function $f(\cdot)$ to reconstruct the secret key $K_{SD} = f(M_1, \ldots, M_n)$ for future communication with node S. Since secret keys are set up on demand, the key establishment approach allows rekeying or key refreshing to be easily implemented in wireless ad hoc networks.

In [18], a REM attack is defined as any arbitrary combination of revealing, erasure, and modification attacks. Each type of attack targets at a different security property:

- *Revealing attacks on keying message confidentiality*: Compromised or fabricated nodes reveal to attackers the content of keying messages traveling through them. To quantify the resilience against this attack, we define a threshold value $r \geq 0$, such that if no more than r messages are revealed to attackers, the resulting secret keys remain *completely unknown* even if all attackers collude.

  Definition 1 A secret key generated by a key establishment scheme with function $f(\cdot)$ is *completely unknown* under r revealed messages if

  $$\text{Prob}\left\{f(M_1, \ldots, M_n) = \hat{K} \middle| M_{i_1}, \ldots, M_{i_r}\right\} = \text{Prob}\left\{f(M_1, \ldots, M_n) = \hat{K}\right\}. \quad (23.10)$$

  for any $i_1, \ldots, i_r$ and any choice of key $\hat{K}$.

  Definition 1 implies that revealing any set of no more than r keying messages does not change the original probability distribution of Prob$\{f(M_1, \ldots, M_n)\}$. Thus, attacks obtain 0 information by knowing r out of n keying messages. However, neither S nor D will know if the key is revealed as long as sufficient messages are passed through intact to generate a successful key.
- *Erasure attacks on keying message availability*: In an attempt to prevent the end-to-end secret key from being established, compromised or fabricated nodes make keying messages unavailable to the destination by not forwarding keying messages or jamming the forwarding link. We define $e \geq 0$ to be a threshold such that the secret key can be recovered at the destination node if no more than e messages are erased or dropped.
- *Modification attacks on keying message integrity*: Since complicated authentication methods (e.g., digital signatures using public-key cryptography) are impractical in ad hoc networks, keying messages are subject to modification attacks, in which compromised or fabricated nodes forward modified keying messages to cause confusion. A threshold value $m \geq 0$ is chosen to denote the maximum number of modified messages that can be corrected by a key establishment scheme.

Definition 2 A REM attack in wireless ad hoc networks is defined as any arbitrary combination of the revealing, erasure, and modification attacks, defined above.

Although erasure and modification attacks can also be regarded as transmission erasures and errors from a classical error control coding perspective, this REM attack model is different, because providing confidentiality (which is irrelevant to error control coding applications) jointly with integrity and availability is a must for establishing secret keys. Given that n keying messages are used for establishing a secret key in a key establishment scheme, we quantify its REM attack resilience by introducing a new security metric $(r, e, m)_n$ denoted as a REM resilience vector.

Definition 3 A key establishment scheme using n messages achieves REM resilience $(r, e, m)_n$ if a secret key can be successfully established under no more than e erasure attacks and m modification attacks, and at the same time, the key is completely unknown to attackers for up to r revealed keying messages.

For a key establishment scheme using n keying messages, the set of achievable REM resilience vectors lies in a three-dimensional region, which illustrates security of the particular scheme along three axes: confidentiality, availability, and integrity (see Fig. 23.13).

We can use this unifying framework to analyze the security of any key establishment schemes. In [4, 32], secret keys of length k are derived at destination nodes by the bitwise XOR of all keying messages, each being exactly k bits, i.e.,

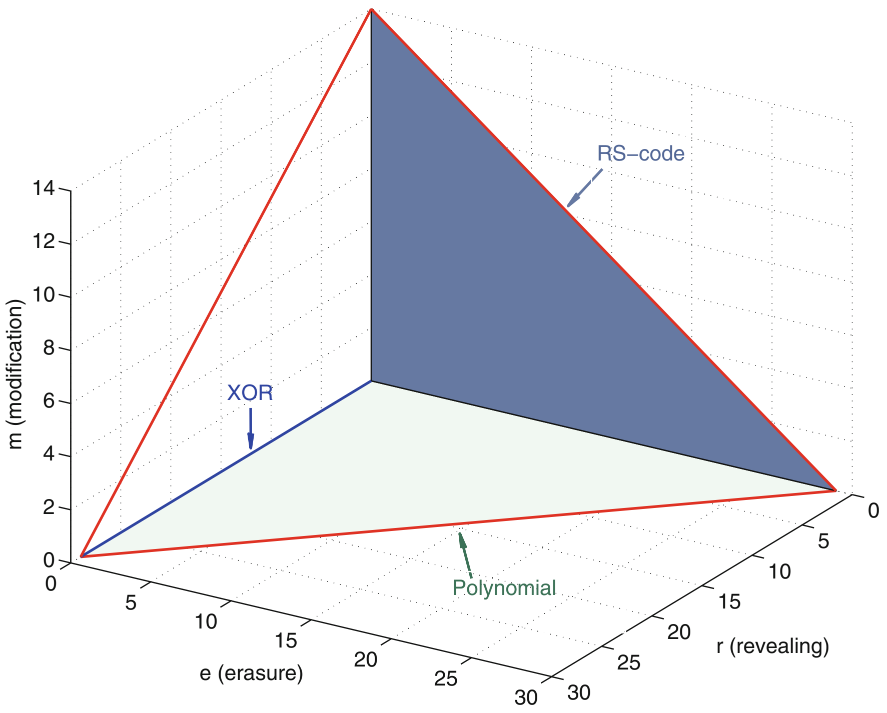

Fig. 23.13 For $n = 30$, this figure plots the 3D optimal REM resilience region (i.e., the tetrahedron defined by $r + e + 2m \leq n - 1$) and 2D sub-planes achieved by previous schemes

$K_{SD} = M_1 \oplus \cdots \oplus M_n$. It is easy to verify that a secret key remains completely unknown if not all keying messages are revealed to attackers. Thus, this scheme achieves REM resilience $(r = n - 1, e = 0, m = 0)_n$. In another scheme based on secret sharing [29], a secret key is regarded as an integer coefficient of a degree t random polynomial in GF_{2^k}, such that it can be recovered from any $t + 1$ evaluations of the polynomial and remains completely unknown if only t evaluations are given. Thus, it achieves $(r = t, e = n - t - 1, m = 0)_n$. By varying the degree t, we denote the set of achievable REM resilience vectors by $(r + e = n - 1, m = 0)_n$.

Another scheme in [14] employs Reed–Solomon (RS) codes to deal with keying message erasures and modifications. The Reed–Solomon codes (RS codes) are non-binary cyclic codes in $GF(2^q)$. RS codes have length $n = 2^q - 1$ with dimension k and minimum Hamming distance $s = n - k + 1$ [21]. Using a secret key of size kq as an input, keying messages are constructed by dividing the output codeword into n pieces, such that the key can be recovered if no more than e and m keying messages are erased and modified respectively, given that $2m + e \leq s - 1$. Since each keying message is a linear combination of the secret key, revealing any keying message makes some choices of keys impossible. Consider a simple scheme with three-bit secret keys $K_{SD} = [b_1 b_2 b_3]$ and a (7,4,3) binary code. If an attack obtains just one bit of the codeword $b_1 \oplus b_2 = 1$, it immediately derives that the secret key cannot be $[00b_3]$ or $[11b_3]$. According to Definition 1, the secret key is not completely unknown to the attacker, and he can remove four possible keys from his entire search space. Thus, we have $r = 0$ for the RS code scheme. Further, by extending this scheme to general non-binary error control codes, a REM resilience of $(r = 0, e + 2m = n - 1)_n$ can be achieved. Table 23.4 summarizes the security analysis of previous key establishment schemes, whose vulnerabilities under REM attacks (i.e., entries with zero resilience) are marked by * in the table.

Table 23.4 Security analysis for key establishment schemes [4, 14, 29, 32]. This shows that these schemes are designed to deal with only a subset of possible attacks

Previous schemes	Resilience vector $(r, e, m)_n$		
	r	e	m
XOR [4, 32]	$r = n - 1$	$e = 0^*$	$m = 0^*$
Polynomial [29]	$r + e = n - 1$		$m = 0^*$
RS code [14]	$r = 0^*$	$2m + e = n - 1$	

23.7.2 Characterization of Optimal Resilience

We summarize the results in [18], which analyzes the optimal REM resilience for arbitrary key establishment schemes. For n paths and n keying messages, it is shown that no matter what keying-message construction and function $f(\cdot)$ are used, it is impossible to achieve any REM resilience vector with $r + e + 2m > n - 1$. This result states that $r+e+2m \leq n-1$ is a universal upper bound on achievable REM resilience vectors. The upper bound is also tight, as the authors in [18] proposed an optimal key establishment scheme which can achieve any REM resilience vector within this bound.

At first glance, it may appear that both optimality and achievability of bound $r + e + 2m \leq n - 1$ can be readily proved by encoding secret keys using an $(n, r + 1, s)$ linear error control code, since the keys are undecodable from r pieces of output codewords, and a direct application of the Hamming distance gives $2m+e \leq n-r-1$. However, the result in [18] is much stronger and requires more interesting proofs. First, the definition of security for key establishment requires secret keys to be completely unknown, not even partially decodable. Any piece of output codeword from a simple $(n, r + 1, s)$-encoding reveals certain linear constraints of the secret keys, and thus violates the desired security. Second, our upper bound $r + e + 2m \leq n - 1$ is applicable to any key establishment schemes with an arbitrary keying-message construction and function $f(\cdot)$, while a linear error control code is just one possible approach. The following analysis provides a fundamental limit for the security performance of key establishment, quantified by the proposed REM resilience vector.

Theorem 1 *(*Optimal Resilience of Key Establishment Approach [18]) *Let each keying message be the same length as the secret key. For n paths and n keying messages, a REM resilience vector $(r, e, m)_n$ can be achieved if and only if $r + e + 2m \leq n - 1$. When the length of keying messages is less than that of the secret key (i.e., length$(M_i) < k$, $\forall i$), it can be proven that a REM resilience $(r, e, m)_n$ can be achieved if and only if $r + e + 2m \leq n - \left\lceil \frac{k}{\text{length}(M_i)} \right\rceil$.*

See Sect. 23.7.5 for proof. Theorem 1 states that for $n > 1$, the set of all achievable REM resilience vectors $(r, e, m)_n$ form a three-dimensional tetrahedron $r + e + 2m \leq n - 1$ as shown in Fig. 23.13, while key establishment schemes in Sect. 23.7.1 only explored certain two-dimensional sub-planes in the tetrahedron: the polynomial-based approach based on [29] achieves $\{r + e \leq n - 1, m = 0\}$, the Reed–Solomon code-based approach in [14] achieves $\{r = 0, e + 2m \leq n - 1\}$, and the XOR-based approach in [4] only achieves a single line $\{r \leq n - 1, e = 0, m = 0\}$. Theorem 1 for key establishment includes all previous results as lower-dimensional special cases.

23.7.3 Low-Complexity Algorithm for Key Establishment

It is shown in [18] that achieving REM resilience vectors on the optimal bound requires multiplications of large integers in GF_p with $p > 2^k$ for constructing keying messages and a complicated sphere decoder. This complexity is prohibitive for wireless ad hoc networks. Therefore, the authors also derived a class of low-complexity key establishment schemes that makes use of binary linear error control codes, and only requires bitwise XOR operations and simple table lookups. The algorithm can achieve a nearly optimal REM resilience. In this section, we first explain the basics of binary linear error control codes, describe the low-complexity algorithm in [18], and then provide a security analysis.

Classical linear coding theory focuses on error correcting. A linear code $\mathcal{C}$ over a finite field with q elements is a linear subspace of the field GF_q^N. If $\mathcal{C}$ is an (n, k, s)-code, then it encodes a vector of length k as a codeword of length n. Let G of size $k \times n$ be the generating matrix for this linear code. Codewords can be obtained by linear combinations of the rows of G, i.e., if $\vec{x}$ is a vector of length k, then $\mathbf{y} = G^T\mathbf{x}$ has length N and is the codeword for $\mathbf{x}$. Parameter s is the distance of the linear code, which is equal to the minimal weight (i.e., number of non-zero components) among all non-zero codewords and measures the error correcting capability of code $\mathcal{C}$. In this chapter, we focus on binary linear codes in GF_2 such that each component is either 0 or 1, although most results can be extended to linear codes in general.

To describe the error correcting procedure, we first introduce the concept of dual code and parity check matrix. The orthogonal complement of $\mathcal{C}$, i.e., the set of all vectors in GF_q^n which are orthogonal to every vector in $\mathcal{C}$, is also a subspace and thus another linear code called the dual code of $\mathcal{C}$, denoted by $\mathcal{C}^\perp$. It is easy to see that if $\mathcal{C}$ is an (n, k, s)-code, then $\mathcal{C}^\perp$ is an $(n, n-k, s')$-code. A generating matrix, denoted by H, for $\mathcal{C}^\perp$ is called a parity check matrix for $\mathcal{C}$ and has size $(n-k) \times n$. A parity check matrix H can be used to recover the codewords of $\mathcal{C}$ because they must be orthogonal to every row of H. Suppose $\hat{\mathbf{y}} = \mathbf{y} + \mathbf{t}$ is a faulty codeword with an error vector $\mathbf{t}$. Then we can compute $r = H\hat{y} = Hy + Ht = Ht$. The vector r is called the syndrome of $\hat{y}$, which voices information about the error vector $\mathbf{t}$, since $H\mathbf{y} = 0$ for all codeword $\mathbf{y} \in \mathcal{C}$. To recover the original codeword $\mathbf{y}$ from the faulty codeword $\hat{\mathbf{y}}$, we only need to store a syndrome table containing all possible syndromes together with corresponding error patterns. In decoding, when $\hat{\mathbf{y}} = \mathbf{y} + \mathbf{t}$ is received, we first calculate the syndrome $\mathbf{r} = H\hat{\mathbf{y}}$, look up the syndrome table with index $\mathbf{r}$ to the error vector $\mathbf{t}$, and then recover codeword $\mathbf{y}$ by $\mathbf{y} = \hat{\mathbf{y}} - \mathbf{t}$.

When both error and erasure occur, the following syndrome decoding procedure for binary linear codes is employed: We first replace the erased coordinates by all zeros and ones and compute two different syndromes (i.e., $\mathbf{r}^0$ and $\mathbf{r}^1$) respectively. After looking up $\mathbf{r}^0$ and $\mathbf{r}^1$ in the syndrome table to obtain two different error vectors $\mathbf{t}^0$ and $\mathbf{t}^1$, the one that contains less number of errors on non-erased coordinates gives us the correct syndrome that should be chosen. More precisely, if $\mathbf{r}^0$ (or $\mathbf{r}^1$ instead) gives less error, then the original codeword can be recovered by inserting 0 (or ones) on the erased coordinates and then abstracting the error vector $\mathbf{t}^0$ (or $\mathbf{t}^1$). In classical coding theory, it has been proven that an (n, k, s)-code is able to correct any e erasures and m errors at the same time, given that $2m + e \leq s - 1$. The following example contains a generating matrix and a parity check matrix for a (8, 2, 5) linear binary code

$$G = \begin{bmatrix} 1\,1\,1\,1\,1\,0\,0\,0 \\ 0\,0\,0\,1\,1\,1\,1\,1 \end{bmatrix}$$

$$H = \begin{bmatrix} 1\,1\,0\,0\,0\,0\,0\,0 \\ 0\,1\,1\,0\,0\,0\,0\,0 \\ 0\,0\,0\,1\,1\,0\,0\,0 \\ 0\,0\,0\,0\,0\,1\,1\,0 \\ 0\,0\,0\,0\,0\,0\,1\,1 \\ 1\,1\,1\,1\,0\,1\,1\,1 \end{bmatrix}$$

For an input vector $\mathbf{x} = [1\ 1]^T$, the corresponding codeword is given by $\mathbf{y} = G^T\mathbf{x} = [1\ 1\ 1\ 0\ 0\ 1\ 1\ 1]^T$. Now, suppose that the fist two bits of $\mathbf{y}$ are erased and the third bit is flipped, i.e., $\hat{\mathbf{y}} = [*\ *\ 0\ 0\ 0\ 1\ 1\ 1]^T$. In order to recover the original codeword from $\hat{\mathbf{y}}$, we compute two syndromes respectively, $\mathbf{r}^0 = [0\,0\,0\,0\,0\,1]^T$ and $\mathbf{r}^1 = [0\ 1\ 0\ 0\ 0\ 1]^T$. By looking up the syndrome table for this (8, 2, 5)-code, we get $\mathbf{t}^0 = [0\ 0\ 0\ 1\ 1\ 0\ 0\ 0]$ and $\mathbf{t}^1 = [0\ 0\ 1\ 0\ 0\ 0\ 0\ 0]$. Since $\mathbf{t}^0$ contains two errors on non-erased coordinates, while $\mathbf{t}^1$ contains only one error, we choose all ones on the erased bits in $\hat{\mathbf{y}}$ and subtract $\mathbf{t}^1$ from it. This gives us the correct codeword $\mathbf{y}$. In the next, we generalize this syndrome decoding method and derive an algorithm for secure key establishment. The proposed algorithm not only corrects modifications and erasures, but also makes secret keys completely unknown to attackers.

Now, we summarize the protocol in [18], for key establishment in wireless ad hoc networks—the low-complexity algorithm relying on linear binary codes. The protocol is divided into four phases: *(1) Request and Path-discovery*, *(2) Sending Keying Messages*, *(3) Recovering Key*, and *(4) Verification*. Packets transmitted in the protocol have the structure

ID_1	ID_2	Payload	CmdType

where ID_1 and ID_2 are the IDs of the source node and the destination node, respectively. In phase 1, any standard ad hoc network routing, such as the Zone Routing Protocol [12], is employed to discover n paths, after receiving a request for key establishment. In phase 2, a $(n+1, t, s)$ error control code is used to generate n keying messages. Let G be a generating matrix for the code

$$G = \begin{bmatrix} g_{01} & g_{02} & \cdots & g_{0t} \\ g_{11} & g_{12} & \cdots & g_{1t} \\ \vdots & \vdots & \ddots & \vdots \\ g_{n1} & g_{n2} & \cdots & g_{nt} \end{bmatrix}_{(n+1)\times t} \tag{23.11}$$

In order to add freshness to the algorithm, the source node constructs t length-k pseudo-random vectors $X_1, \ldots, X_t$ and encodes each column of matrix $[X_1, \ldots, X_t]$ using G:

$$[K_{SD}, M_1, \ldots, M_n]^T = G \cdot [X_1, \ldots, X_t]^T \tag{23.12}$$

where the first row of the output codeword is chosen as a secret key and the $(i+1)$th row as keying message $\mathbb{M}_i$ for $i = 1, \ldots, n$. Since linear binary codes are used, all operations required in this phase are simply binary XORs, denoted by $\oplus$.

Without loss of generality, assume that the last e keying messages are unavailable to the destination node due to erasure attacks and the remaining $n - e$ keying messages contain m faulty ones due to modification attacks. Let H be a parity check matrix of size $(n+1) \times (n+1-t)$ for the generating matrix in (23.11). In phase 3, the destination node implements a key-recovery algorithm based on the syndrome decoding for linear binary codes, as described in Sect. 23.7.3.A. Since the secret key K_{SD} is just the first row of the codeword in (23.12), the algorithm only needs to restore the first row of the codeword, rather than to decode all random vectors $X_1, \ldots, X_t$. In phase 4, the secret key is verified between the source and destination node. Our protocol for establishing a secret key between two nodes S and D is summarized as follows:

Phase 1 *Request and Path-discovery*

1. Node D broadcasts a request for key establishment:

$$D\colon \boxed{D}\boxed{S}\boxed{\text{Void}}\boxed{\text{ReqKey}}$$

2. Node S responses to the request and starts a routing query for node D using the standard Zone Routing Protocol [12].
3. Node S recodes the first n replies to its routing query and prepares n paths to D:

$$(S, R_{i,1}, R_{i,2}, R_{i,3}, \ldots, D), \text{ for } i = 1, \ldots, n$$

Phase 2 *Sending Keying Messages*

1. Node S constructs t length-k pseudo-random vectors $X_1, \ldots, X_t$.
2. The secret key is derived by

$$K_{SD} = (g_{01}X_1) \oplus (g_{02}X_2) \oplus \ldots \oplus (g_{0t}X_t)$$

3. Initialize $i = 1$.
4. Node S generates keying message $\mathbb{M}_i$:

$$\mathbb{M}_i = (g_{i1}X_1) \oplus (g_{i2}X_2) \oplus \ldots \oplus (g_{it}X_t)$$

5. Node S sends $\mathbb{M}_i$ to node $R_{i,1}$ and erases $\mathbb{M}_i$ locally

$$S \to R_{i,1}\colon \boxed{D}\boxed{R_{i,1}}\boxed{E\left[M_i, K_s^{i,1}\right]}\boxed{\text{EstKey}}$$

6. If $i < n$, let $i = i + 1$ and go to step 4.
7. Node S erases $X_1, \ldots, X_t$ from his memory.
8. Messages are forwarded to node D, for $i = 1, \ldots, n$:

$$R_{i,1} \rightarrow R_{i,2}\text{:} \boxed{R_{i,1}}\boxed{R_{i,2}}\boxed{E\left[M_i, K_{i,1}^{i,2}\right]}\boxed{\text{EstKey}}$$

$$R_{i,2} \rightarrow R_{i,3}\text{:} \boxed{R_{i,2}}\boxed{R_{i,3}}\boxed{E\left[M_i, K_{i,2}^{i,3}\right]}\boxed{\text{EstKey}}$$

$$\vdots$$

$$R_{i,j} \rightarrow D\text{:} \boxed{R_{i,j}}\boxed{D}\boxed{E\left[M_i, K_{i,j}^{D}\right]}\boxed{\text{EstKey}}$$

Phase 3 *Recovering Key*

1. Node D receives at least $n - e$ keying messages $\hat{M}_1, \ldots, \hat{M}_{n-e}$.
2. Define a mask vector A according to the indices of received keying messages: $A_1 = 0$ and

$$A_{i+1} = \begin{cases} 1, \text{ if } \hat{M}_i \text{ is received} \\ 0, \text{ otherwise} \end{cases} \forall i = 1, \ldots, n$$

3. Node D computes a submatrix $\tilde{H}$, consisting of the $n - e$ non-erased rows of H:

$$\tilde{H}_i = H_{i+1}, \text{ for } i = 1, \ldots, n - e$$

4. Node D computes a syndrome perturbation vector $\tilde{r}$ as the XOR of the $e + 1$ erased rows of H:

$$\tilde{r} = H_1 \oplus H_{n-e+2} \ldots \oplus H_{n+1}$$

5. Node D computes $R^0 = \tilde{H}^T \cdot \left[\hat{\mathbb{M}}_1, \ldots, \hat{\mathbb{M}}_{n-e}\right]^T$.
6. Initialize $i = 1$. Let $ADDR$ be the base address of the syndrome table stored at the destination node.
7. Retrieve $\mathbf{t^0}$ from address $ADDR + R_i^0$.
8. Retrieve $\mathbf{t^1}$ from address $ADDR + \left(R_i^0 \oplus \tilde{r}\right)$.
9. The ith bit of K_{SD} is given by

$$K_{SD,i} = \begin{cases} \mathbf{t}_1^{\mathbf{0}}, & \text{if } popcnt(\mathbf{t^0} \bigwedge A) < popcnt(\mathbf{t^1} \bigwedge A) \\ 1 \oplus \mathbf{t}_{\mathbf{1}}^{\mathbf{1}}, & \text{otherwise} \end{cases}$$

10. If $i < k$, let $i = i + 1$ and go to step 5.

Phase 4 *Verifying Key*

1. Node D generates a random message R and computes its hash value $h(R)$.
2. Node D broadcasts a challenge using secret key K_{SD}:

$$D\colon \boxed{D}\boxed{S}\boxed{E[(R, h(R)), K_{SD}]}\boxed{\text{GotKey}}$$

3. Node S decrypts $E[(R, h(R)), K_{SD}]]$ using its version of secret key K_{SD} and obtains $\hat{R}$.
4. Node S broadcasts an acknowledgement

$$S\colon \boxed{S}\boxed{D}\boxed{\hat{R}}\boxed{\text{ACK}}$$

5. Node D accepts K_{SD} if it receives $\hat{R} = R$.

In Step 5 of phase 3 above, each row of $\left[\hat{M}_1, \ldots, \hat{M}_{n-e}\right]$ is a valid codeword generated by (23.11) with $e + 1$ erasures and m modifications. According to the syndrome decoding procedure described in Sect. 23.7.3.A, if we assume that the erased keying messages are all zero vectors, we can compute a syndrome matrix $R^0 = \tilde{H}^T \cdot \left[\hat{M}_1, \ldots, \hat{M}_{n-e}\right]^T$, where each column of R^0 is a syndrome vector. On the other hand, if we assume that the erased keying messages are all one vectors, it is easy to show that the syndrome for the ith row of $\left[\hat{M}_1, \ldots, \hat{M}_{n-e}\right]$ becomes $\tilde{r} \oplus R^0_i$, with $\tilde{r}$ as a perturbation vector defined in Step 4. Therefore, by looking up the syndrome table and comparing resulting error vectors, we can recover the first bit of the secret key, and thereafter bit by bit. In Step 9 of Phase *Recovering Key*, *popcnt* is a population count instruction which counts the number of "1" bits in a word.

The low-complexity key establishment algorithm is able to achieve nearly optimal REM resilience vectors $(r, e, m)_n$ by choosing different linear error control codes. For n paths and n keying messages, the security performance of the algorithm is characterized as follows.

Theorem 2 *(*Resilience of Low-Complexity Key Establishment [18]) *For a linear binary error control code $(n+1, t, s)$ with dual code $(n+1, n+1-t, s')$, the low-complexity key establishment algorithm in [18] achieves a REM resilience vector $(r, e, m)_n$ for $r = s' - 2$ and $2m + e = s - 2$. In particular, when both codes are maximum distance separable (MDS), the algorithm achieves an optimal REM resilience of $2m + e + r = s + s' - 4 = n - 3$.*

See Sect. 23.7.6 for proof.

23.7.4 Numerical Simulations

Consider a wireless ad hoc network with $Z = 1000$ nodes, uniformly distributed in a square area of size $L = 100$. We assume that nodes in the neighborhood of communication range $R = 15$ share pre-installed keys with probability p. These pre-installed link keys are used to secure keying messages during transmission. The standard Zone Routing Protocol (ZRP) [12] with a zone radius of $\rho = 2$ hops is employed to discover n paths for each node pair. Due to the page limitation, we focus on security comparisons in this section and do not provide a network-aspect simulation with complexity evaluations. In all numerical examples, compromised nodes are randomly selected from the Z nodes such that the locations of compromised nodes are uniformly distributed in the area. All security performance is evaluated over 40,000 different realizations and node selections.

We define the probability of secure and successful key establishment as the average probability that two nodes can successfully establish a secret key, and at the same time, the secret key remains completely unknown to attackers. For $p = 0.5$ and optimal key establishment, Fig. 23.14 plots the probability of secure and successful key establishment for the use of $n = 1, 5, 10, 20, 30, 40$ keying messages, under REM attacks with equal probability of each type of attack. It can be observed that the optimal key establishment with $n \geq 20$ can safeguard secret keys with a probability of over 80% for as many as 80 (i.e., 8%) malicious nodes, and its security performance benefits from the increase in keying messages as more path diversity is exploited. This figure provides an important benchmark for the design of practical key establishment algorithm for given security requirements and expected fractions of compromised nodes.

For the same network model with $p = 0.5$ and $n = 30$, we compare in Fig. 23.15 the security performance of different schemes: the optimal key establishment algorithm in Sect. 23.7.5, the low-complexity key establishment algorithm in Sect. 23.7.3, key establishment using single path, and the three previous multi-path key establishment schemes. Our low-complexity algorithm proposed in Sect. 23.7.3, which is based on a (31, 11, 11) linear code and its dual (31,20,6) code for achieving resilience $(r = 4, e + 2m = 9)_31$, has a performance that is close to the optimal one and is more suitable for practical implementations. This comparison highlights the importance of defending against multiple attacks simultaneously: under REM attacks, the overall security performance of a key establishment algorithm is largely determined by the worst individual-attack resilience (i.e., $\min(r, e, m)$). It also demonstrates the excellent security-complexity properties of our proposed key establishment protocol.

For the same network model with $p = 0.5$ and $n = 30$, we compare the security performance of different schemes: the optimal key establishment algorithm in Theorem 1, the low-complexity key establishment algorithm in [18], key establishment using single path, and the three multi-path key establishment schemes discussed in Sect. 23.7.1. The low-complexity algorithm in [18] has a performance that is close to the optimal one and is more suitable for practical implementations. This comparison highlights the importance of defending against multiple attacks simultaneously: under REM attacks, the overall security performance of a key establishment algorithm is largely determined by the worst individual-attack resilience (i.e., $\min(r, e, m)$).

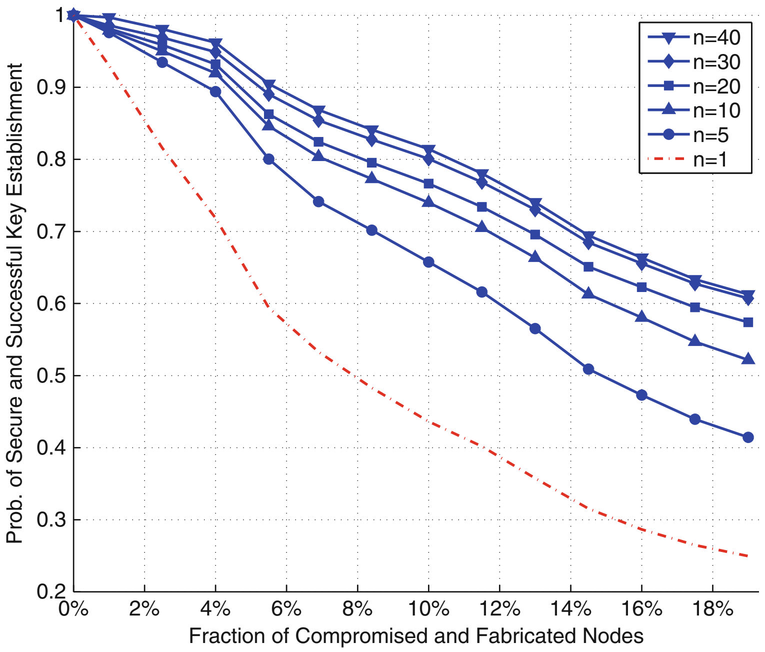

Fig. 23.14 Probability of secure and successful key establishment vs. number of compromised nodes for $n = 1, 5, 10, 20, 30, 40$ keying messages. A diminishing security improvement is observed when more messages are used for key establishment

23.7.5 Proof of Theorem 1

Proof The theorem states that the bound $r + e + 2m = n - 1$ is both optimal and tight. In the following, we start by showing the optimality and then propose a new key establishment scheme to prove the achievability.

To show $r + e + 2m = n - 1$ is optimal. If $e = m = 0$, then we immediately have $r \leq n - 1$, since the secret key becomes deterministic given all n keying messages. For $e + m > 0$, we denote $[M_1, \ldots, M_n]$ as a *feasible message vector*, in which $M_1, \ldots, M_n$ are a set of allowable keying messages that can be used to establish a secret key $K_{SD} = f(M_1, \ldots, M_n)$. Without loss of generality, we assume that the first r keying messages $M_1, \ldots, M_r$ are revealed to attackers who are able to collude. Then, with this information, the attackers can rule out any feasible message vector whose first r keying messages are not equal to $M_1, \ldots, M_r$. To guarantee that the secret key remains completely unknown, it is necessary that the number of remaining feasible message vectors with the first r messages in common must be no less than 2^k, i.e., the number of all possible secret keys of length k. Formally, if $H(\cdot)$ denotes the entropy function and feasible message vectors are random, we derive

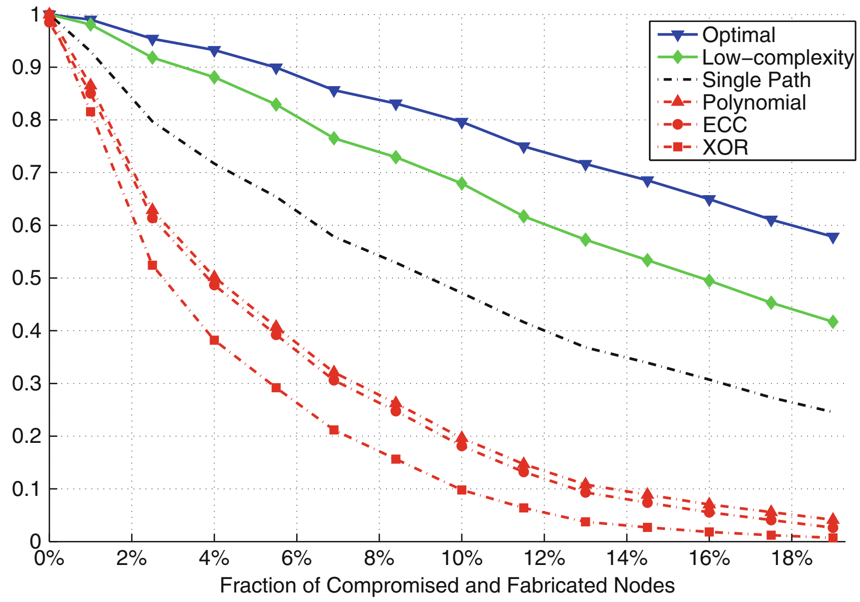

Fig. 23.15 Compare security performance of key establishment schemes under our REM attacks. Our low-complexity algorithm is based on a $(1, 31)$ linear code and its dual (6,20,31) code, and achieves REM resilience $(r = 4, e + 2m = 9)_{31}$

$$\begin{aligned} &H([M_1, \ldots, M_n]|M_1, \ldots, M_r) \\ &\quad \geq H(f(M_1, \ldots, M_m)|M_1, \ldots, M_r) \\ &\quad = H(K_{SD}|M_1, \ldots, M_r) \\ &\quad = H(K_{SD}) = k \end{aligned} \tag{23.13}$$

where K_{SD} is the secret key. The second step is from the information processing inequality and the last step holds because all keys are equally likely due to the definition of completely unknown (23.10). Equation (23.13) implies that with the first r messages fixed, there exists at least 2^k feasible message vectors. These 2^k feasible message vectors are different only in the last $m - r$ messages, each of length k. Thus, the minimum Hamming distance of these feasible message vectors (i.e., the minimum number of different messages in any two feasible message vectors) can be no more than $m - r$. According to error control coding theory, given e erasures and m modifications, two feasible message vectors with a Hamming distance of $m - r$ remain distinct and separable only if

$$2m + e + 1 \leq n - r \quad \Leftrightarrow \quad r + e + 2m \leq n - 1 \tag{23.14}$$

This gives the optimality of bound $r + e + 2m \leq n - 1$.

For achievability of the bound, we propose a new key establishment scheme that achieves any REM resilience vector $(r, e, m)_n$ satisfying the upper bound

$r + e + 2m + 1 = n$. The proposed algorithm for generating n keying messages is similar to the polynomial evaluation used in [29]. However, we employ a different decoding strategy and show that the algorithm can deal with revealing, erasure, and modification attacks at the same time. Let $p > 2^k$ be a prime number. Thus the desired secret key can be regarded as an integer in the field GF_p, i.e., $K_{SD} \in [0, 2^k - 1]$. We generate a random degree r polynomial in GF_p as follows:

$$q(z) = K_{SD} + A_1 z + \cdots + A_r z^{rv} \tag{23.15}$$

where $A_i \in GF_p$ for $i = 1, \ldots, r$ are randomly chosen integers. Then n keying messages are computed by evaluating $q(x)$ at n distinct points for $z = 1, \ldots, n$, i.e.,

$$[M_1, M_2, \ldots, M_n] = [q(1), q(2), \ldots, q(n)] \tag{23.16}$$

Since the polynomial has degree r, it has been shown in [29] that revealing no more than r keying messages would leave the secret key K_{SD} completely unknown. So we only need to show that the destination node can recover key K_{SD} under e erasures and m modifications, given that $2m + e = m - r - 1$. Toward this end, we rewrite (23.16) using a matrix representation:

$$\begin{bmatrix} M_1 \\ M_2 \\ \vdots \\ M_n \end{bmatrix} = \begin{bmatrix} 1 & 1^1 & 1^2 & \ldots & 1^r \\ 1 & 2^1 & 2^2 & \ldots & 2^r \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & m^1 & m^2 & \ldots & m^r \end{bmatrix} \cdot \begin{bmatrix} K_{SD} \\ A_1 \\ \vdots \\ A_r \end{bmatrix}$$

It is easy to verify that the $n \times (r+1)$ coefficient matrix (denoted by G) on the right hand side is a Vandermonde matrix, whose any $r + 1$ rows are full rank. Thus, any non-zero vector $\mathbf{x}$ in $GF_p^{(r+1)}$ of size $1 \times (r+1)$ can be orthogonal to at most r rows of matrix G. We have

$$\forall \mathbf{x} \neq \mathbf{0},\ \text{Hamming}(G\mathbf{x}, \mathbf{0}) \geq n - r \tag{23.17}$$

where $\mathbf{0}$ is a zero vector and Hamming$(\cdot)$ is the Hamming distance function. This implies that matrix G is a generating matrix for a $(n, r + 1, s)$ linear error control code in GF_p with a minimum Hamming distance of at least $n - r$. According to error control coding theory, given that $2m + e + 1 \leq n - r$, any m modifications and e erasures of the keying messages can be corrected at the destination node using a sphere decoding algorithm which finds the closest feasible message vector to the received one [21]. We summarize the optimal key establishment algorithm as follows:

This complete the proof of Theorem 1. ⊠

Optimal Key Establishment Algorithm

1. Source node generates a random key K_{SD} and r random integers $A_1, \ldots, A_r$.
2. Source node generates $\mathbb{M}_i = K_{SD} + A_1 i + \ldots + A_r i^r$ and sends it to destination node, for $i = 1, \ldots, n$.
3. Destination node employs sphere decoding to derive K_{SD} upon receiving the keying messages.

23.7.6 *Proof of Theorem 2*

Proof We first prove that the proposed algorithm can recover the secret key under e erasure and m modification attacks and then show that attacks have absolutely no information about the secret key with r revealing attacks.

Since each row of the codeword matrix $[K_{SD}, M_1, \ldots, M_n]$ is a valid codeword for the $(n+1, t, s)$ error control code, classical coding theory shows that up to $\left\lfloor \frac{s-1}{2} \right\rfloor$ errors can be corrected by syndrome decoding. In Algorithm 3, we choose the $e+1$ erased keying messages to be all zeros and all ones respectively. Because the error control code is binary, one of the two choices introduces no more than $\left\lfloor \frac{e+1}{2} \right\rfloor$ new errors, and thus leads to no more than $m + \left\lfloor \frac{e+1}{2} \right\rfloor$ errors totally. These errors can be corrected by the syndrome decoding in Algorithm 3 if the following is satisfied:

$$m + \left\lfloor \frac{e+1}{2} \right\rfloor = \left\lfloor \frac{2m+e+1}{2} \right\rfloor \le \left\lfloor \frac{s-1}{2} \right\rfloor \tag{23.18}$$

This establishes $2m + e \le s - 2$ as a sufficient condition for recovering secret key K_{SD}.

To show that secret key K_{SD} remains completely unknown to attacks, without loss of generality, we assume that keying messages $M_1, \ldots, M_r$ are revealed to attackers. According to the construction of messages, attackers have $r+1$ equations in the following matrix representation:

$$\begin{bmatrix} K_{SD}^T \\ M_1^T \\ \vdots \\ M_r^T \end{bmatrix} = \begin{bmatrix} g_{01} & g_{02} & \cdots & g_{0t} \\ g_{11} & g_{12} & \cdots & g_{1t} \\ \vdots & \vdots & \ddots & \vdots \\ g_{r1} & g_{r2} & \cdots & g_{rt} \end{bmatrix} \cdot \begin{bmatrix} X_1^T \\ X_2^T \\ \vdots \\ X_t^T \end{bmatrix} \tag{23.19}$$

Because the dual error control code $(n+1, n+1-t, s')$ has distance s', classical coding theory shows that any $s'-1$ rows of the G matrix are linearly independent. Further, s' is upper bounded by $s' \le t+1$. When $r \le s'-2$ as claimed in the statement of Theorem 2, we also have $r+1 \le t$. This implies that the first matrix on the right-hand side of (23.19) is full row-rank.

Thus, when $M_1, \ldots, M_r$ are fixed in (23.19), for each possible choice of secret key K_{SD}, (23.19) defines a system of $r + 1$ linear equations with t unknowns, i.e., $\mathbb{X}_1, \ldots, \mathbb{X}_t$. There exists 2^{t-r-1} possible $\mathbb{X}_1, \ldots, \mathbb{X}_t$ vectors such that (23.19) is satisfied. More precisely, since vectors $\mathbb{X}_1, \ldots, \mathbb{X}_t$ are generated randomly by a uniform distribution, we have

$$\begin{aligned}
&\text{Prob}\left\{K_{SD} = \hat{K} \,\middle|\, [M_1, \ldots, M_r] = \hat{M}\right\} \\
&= \frac{\text{Prob}\left\{K_{SD} = \hat{K}, [M_1, \ldots, M_r] = \hat{M}\right\}}{\sum_K \text{Prob}\left\{K_{SD} = K, [M_1, \ldots, M_r] = \hat{M}\right\}} \\
&= \frac{\text{Prob}\left\{[X_1, \ldots, X_t] \in \mathcal{X}_{\hat{K},\hat{M}}\right\}}{\sum_K \text{Prob}\left\{[X_1, \ldots, X_t] \in \mathcal{X}_{K,\hat{M}}\right\}} \\
&= \frac{1}{2^k}
\end{aligned} \tag{23.20}$$

where $\mathcal{X}_{\hat{K},\hat{M}}$ is the set of all $X_1, \ldots, X_t$ satisfying (23.19) for $K_{SD} = \hat{K}$ and $[M_1, \ldots, M_r] = \hat{M}$. Equation (23.20) used the fact that $\left|\mathcal{X}_{\hat{K},\hat{M}}\right| = 2^{t-r}$ for all $\hat{K}$ and $\hat{M}$ and that $\mathbb{X}_1, \ldots, \mathbb{X}_t$ are uniformly distributed. From (23.20), we conclude that given keying messages $\mathbb{M}_1, \ldots, \mathbb{M}_r$, unconditional secrecy as defined in (23.10) is achieved if $v \le s' - 2$.

In addition, according to classical coding theory, for binary error control codes, we have $s + s' = n + 1$ when both the primal and the dual codes are maximum distance separable.Thus, we derive $r + 2m + e = s + s' - 4 = n - 3$, which is the desired result. ⊠

23.8 Concluding Remarks

In this chapter, we discuss key management in lightweight mobile ad hoc networks. Backed up by the large successful attack probabilities computed in this chapter, we show that the probabilistic key predistribution schemes are in fact quite vulnerable to node captures in many practical cases. Considering the large key pool and key ring sizes, complex key predistribution, low network connectivity, and complex pairwise link establishments, the advantage of the probabilistic approach over the deterministic approach is not as much as people have believed. We also generalize the re-examination to other probabilistic key predistribution schemes, including the q-composite key scheme, the multiple disjoint path key reinforcement scheme, and the scheme based on partial deployment information. All of these schemes are vulnerable to node capture in a mobile network. A selective node capture will further weaken the performance of a probabilistic approach.

We then propose two low-cost hardware-based architectures to enhance the security of key management schemes against the attack of sensor node fabrication for a lightweight mobile ad hoc network, which can benefit both probabilistic and deterministic key management.

Finally, we propose a unifying framework for analyzing the security of any key establishment scheme, quantified by a new metric we call a REM resilience vector. A universal bound on achievable REM resilience vectors is derived in closed-form and is shown to be attained by an optimal key establishment algorithm. For practical implementations, we also develop a low-complexity XOR-based key establishment protocol that achieves nearly optimal REM resilience. Our analysis and simulation show that the capability of simultaneously defending against multiple attack classes, critical for the security of wireless ad hoc networks, can indeed be achieved with provable REM resilience and low complexity.

References

1. R. Blom. An optimal class of symmetric key generation system. In *Advanced in Cryptology—Eurocrypt'84*, LNCS, vol. 209, pages, 335–338, 1984.
2. S. Çamtepe and B. Yener. Combinatorial design of key distribution mechanisms for wireless sensor networks. In: *European Symposium On Research in Computer Security (ESORICS'04)*, Sophia Antipolis, France, 2004.
3. S. A. Çamtepe and B. Yener. Key distribution mechanisms for wireless sensor networks: a survey. Technical Report TR-05-07, Rensselaer Polytechnic Institute, Computer Science Department, 2005. Available at http://www.cs.rpi.edu/research/pdf/05-07.pdf. Accessed in 2006
4. H. Chan, A. Perrig, and D. Song. Random key predistribution schemes for sensor networks. In *IEEE Symposium on Security and Privacy*, pages 197–213, Oakland, CA, 2003.
5. W. Du, J. Deng, Y. Han, S. Chen, and P. Varshney. A key management scheme for wireless sensor networks using deployment knowledge. In: *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, Hong Kong, 2004.
6. W. Du, J. Deng, Y. S. Han, and P. K. Varshney. A pairwise key pre-distribution scheme for wireless sensor networks. In: *CCS'03: ACM conference on Computer and communications security*, New York, NY, pages. 42–51, 2003.
7. J. Dwoskin and R. Lee. Hardware-rooted trust for secure key management and transient trust. In: *CCS'07: ACM conference on Computer and communications security*, Alexandria, VA, pages 389–400, 2007.
8. J. Dwoskin, D. Xu, J. Huang, M. Chiang, and R. Lee. Secure key management architecture against sensor-node fabrication attacks. In: *IEEE GlobeCom*, Washington, D.C., pages 166–171, 2007.
9. L. Eschenauer and V. D. Gligor. A key-management scheme for distributed sensor networks. In: *CCS'02: ACM conference on Computer and communications security*, New York, NY, pages, 41–47, 2002.
10. P. Feldman. A practical scheme for non-interactive verifiable secret sharing. In: *IEEE Symposium on Foundations of Computer Science*, Los Angeles, CA, Pages 427–438, 1987.
11. M. Fitzi, J. Garay, S. Gollakota, C. Rangan, and K. Srinathan. Round-optimal and efficient verifiable secret sharing. In: *Third Theory of Cryptography Conference*, Volume 3876 of Lecture Notes in Computer Science, pages 329–342, 2006.
12. Z. Haas and M. Pearlman. The performance of query control schemes for the zone routing protocol. In: *IEEE/ACM Transactions on Networking*, 9(4): 427–438, 2001.
13. A. Howard, M. J. Mataric, and G. S. Sukhatme. Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. In: *Distributed Autonomous Robotic Systems*, Fukuoka, Japan, pages 299–308, 2002.

14. D. Huang and D. Medhi. A byzantine resilient multi-path key establishment scheme and its robustness analysis for sensor networks. In: *19th IEEE International Parallel and Distributed Processing Symposium*, Washington, DC, pages, 4–8, 2005.
15. D. Huang and D. Medhi. Secure pairwise key establishment in large-scale sensor networks: An area partitioning and multigroup key predistribution approach. *ACM Transactions on Sensor Networks* 16:1–34, 2007.
16. D. Huang, M. Mehta, D. Medhi, and L. Harn. Location-aware key management scheme for wireless sensor networks. In: *Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*, ACM New York, NY, pp. 29–42, 2004.
17. J. Hwang and Y. Kim. Revisiting random key pre-distribution schemes for wireless sensor networks. In; *ACM workshop on Security of ad hoc and sensor networks*, Washington, DC, USA, pages, 43–52, 2004.
18. T. Lan, M. Chiang, and R. Lee. Multi-path key establishment against REM attacks in wireless ad hoc networks. In: *IEEE Globecom*, Honolulu, HI, 2009.
19. J. Lee and D. Stinson. Deterministic key predistribution schemes for distributed sensor networks. 11th Annual Workshop on Selected Areas in Cryptography, Waterloo, Ontario, Canada, 2004.
20. R. Lee et al. Architecture for protecting critical secrets in microprocessors. In: *International Symposium on Computer Architecture (ISCA 2005)*, pages, 2–13, 2005.
21. S. Lin and D. Costello. *Error Control Coding: Fundamentals and Applications*. Prentice Hall, NJ, USA, 1983.
22. D. Liu and P. Ning. Establishing pairwise keys in distributed sensor networks. In: *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS '03)*, Washington, DC, pages, 52–61, 2003.
23. D. Liu and P. Ning. Location-based pairwise key establishments for static sensor networks. In: *Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks*, ACM New York, NY, pages. 72–82, 2003.
24. T. Matsumoto and H. Imai. On the key predistribution systems: a practical solution to the key distribution problem. In: *Advances in Sryptology–Crypto'87*, Santa Barbara, CA, 1987.
25. C. S. R. Murthy and B. S. Manoj. *Ad Hoc Wireless Networks: Architectures and Protocols*. Prentice Hall Communications Engineering and Emerging Technologies Series, NJ, USA, 2004.
26. B. Parno, A. Perrig, and V. Gligor. Distributed detection of node replication attacks in sensor networks. In: *Proceedings of the 2005 IEEE Symposium on Security and Privacy*, pages 49–63, California, USA, 2005.
27. R. D. Pietro, L. V. Mancini, and A. Mei. Random key-assignment for secure wireless sensor networks. In: *SASN'03: ACM workshop on Security of ad hoc and sensor networks*, New York, NY, pages. 62–71, 2003.
28. S. Seys and B. Preneel. The wandering nodes: Key management for lower-power mobile ad hoc netowrks. In: *IEEE International Conference on Distributed Computing Systems Workshops, ICDCSW'05*, 2005.
29. A. Shamir. How to share a secret. In: *Communications of the ACM*, 1979.
30. S. Zhu, S. Setia, and S. Jajodia. LEAP: Efficient security mechanisms for large-scale distributed sensor networks. In: *CCS'03: ACM conference on Computer and communications security*, New York, NY, pages. 62–72,, 2003.
31. S. Zhu, S. Setia, S. Jajodia, and P. Ning. An interleaved hop-by-hop authentication scheme for filtering of injected false data in sensor networks. Security and Privacy, 2004. In: *Proceedings*. IEEE Symposium, pages, 259–271, California, USA, 2004.
32. S. Zhu, S. Xu, S. Setia, and S. Jajodia. Establishing pairwise keys for secure communication in ad hoc networks: A probabilistic approach. In: *Proceedings of the 11th IEEE International Conference on Network Protocols (ICNP'03)*, pp. 326–335, 2003.