

## Chapter 2

# Pair-Wise Key Establishment

In this chapter we start addressing the security of Wireless Sensor Networks (WSNs) from the point of view which is closest to the single node: We consider the authentication and the confidentiality of the communications with other nodes. In particular, this chapter presents the ECCE Protocol, a new distributed, probabilistic, cooperative protocol to establish a secure pair-wise communication channel between any pair of sensors in a WSN. The main contribution of the ECCE Protocol is: To allow the set up of a secure channel between two sensors (principals) that do not share any pre-deployed key. This feature is obtained involving a set of sensors (cooperators) in the channel establishment protocol to provide probabilistic authentication of the principals as well as the cooperators. In particular, the probability for the attacker to break authentication check decreases exponentially with the number of cooperators involved. We provide analytical analysis and extensive simulations of the ECCE, which show that the proposed solution increases both the probability of a secure channel set up and the probability of channel resilience with respect to other protocols.

### 2.1 Introduction

WSNs are expected to be the basic building block of pervasive computing environments, hence establishing secure pair-wise communications could be useful for many applications. In particular, it is a pre-requisite for the implementation of secure routing, and can be useful for secure group communications. Further, pair-wise secure communication allow in-network processing [247], or facilitate the establishment of a cluster key, hence enabling *passive participation*, in which a sensor node can take certain actions based on overheard messages. It was pointed out in [3, 36, 173], that asymmetric cryptography such as RSA or Elliptic Curve Cryptography (ECC) is unsuitable for most sensor architectures due to high energy consumption and

increased code storage requirements. However, it is worth noticing that evolution in technology allows to sparingly use asymmetric cryptography for a certain class of WSN [222]. For instance, in [169] the authors devise a protocol that, with the seldom use of ECC, thwarts the replication attack [79, 159]. However, it seems reasonable that there will be always some classes of WSNs in which asymmetric cryptography would rise an unfeasible cost due to either energy consumption or memory constraints. Indeed, as for energy consumption, in a mobile WSN if we have a secure key establishment protocol based on ECC whenever two sensors want to agree on a shared key for the first time, this would put high requirements on battery consumption. As for memory, it seems unfeasible that any node could host the public keys of all the other nodes in the network (for instance, the Mica mote is equipped with a 8 bit 4 MHz processor and has 4 KB of RAM and 128 KB of flash RAM only). Note that the constraint on memory stands even in a static WSN. Hence, while solutions that intend to address specific problems can directly benefit of the sparingly use of ECC [169], building communication channel based on symmetric algorithms, which are three order of magnitude more efficient than ECC [222], is still an attractive research field [36, 197].

This chapter presents the ECCE Protocol, a new protocol to establish a secure pair-wise communication channel between any pair of sensors in the WSN. The ECCE Protocol can be classified as probabilistic and cooperative. Unlike other protocols for channel establishment, ECCE allows to establish a secure channel between sensors that do not share any key, involving a set of cooperating sensors (cooperators) which are not required to share a key with both principals. The same feature is not present in actual protocols such as Multipath Key Reinforcement [37] and Cooperative [75]. The overhead required is limited and it is sustained just once during the sensor life-time. ECCE shows better performance in channel existence and channel resilience than existing protocols. The Protocol also guarantees implicit and probabilistic mutual authentication of principals and cooperators without any additional overhead and without the presence of a base station. Further, the proposed protocol could be used also between sensor that already share some secret keys to increase the resilience of these shared keys. The proposed protocol is also adaptive to the required security level: To achieve an higher level of security, it suffices to involve an higher number of cooperators in the channel set-up. Finally, the protocol allows to trade off the memory required to store pre-deployed keys with cooperators. In particular, it is possible to set the number of cooperators in order to have a reduced key ring that provides the same level of security and the same probability of channel existence of solutions that involve no cooperators but a large key ring size. For example, choosing a pool of size 1,000, a key ring of size 12, and involving 8 cooperators, provides the same probability of channel existence of a scenario in which every sensor has 20 pre-deployed keys but there are no cooperators. As for resiliency, with a pool of size 10,000, a key ring of size 100 and 8 cooperating sensors, the attacker is required to capture 110 sensors to corrupt a channel, while with the same parameters, but with no cooperators, the attacker has to corrupt only 75 sensors to corrupt the channel. Note that reducing the key ring size provides the possibility for sensors to store the cooperative keys set-up with the ECCE Protocol. Compared to several recently

proposed approaches such as [11, 66, 196] which fall in different categories of key establishment schemes, our analytical and experimental results show that the ECCE Protocol has better performance than the other protocols as for channel existence and channel resiliency to the attacker.

## Organization

The remainder of this chapter is organized as follows. In Sect. 2.2, we review the current contributions in the field. In Sect. 2.3, we report some preliminaries and define our system assumptions. In Sect. 2.4, we describe the ECCE Protocol, while in Sect. 2.5, we analyze the probability to establish a secure channel and the resilience of the established channel.

## 2.2 Related Work

Some research focus on key establishment protocol for WSN based on centralized solution. Examples of centralized protocols include [73, 143, 173]. Centralized protocols assume the presence of a Base Station (BS), which takes part in the process of establishing a pair-wise key between pairs of sensors. This kind of solution has some drawbacks, for instance the energy consumption experienced by the nodes close to the BS, and the presence of a single point of failure. Other research focus on distributed solution for pair-wise keys establishment. To better refine this classification, we can distinguish between *deterministic* and *probabilistic* solutions. As for deterministic solutions one can see [36, 144, 197]. However, each of these solutions suffer of a specific type of problem. In [144], the attacker only needs to corrupt a constant number of nodes to disrupt the confidentiality of the whole network. In [197], the authors recognise that given a fixed key-ring size, this limits the number of sensors in the network. Finally, in [36], each sensor is required to store  $O(\sqrt{N})$  keys; moreover, the number of sensors that belong to the same WSN (that is  $N$ ) must be known at design time. As for probabilistic solutions, the idea of probabilistic key sharing for WSN was firstly introduced in [85]. In the proposed solution, each of the  $N$  sensors of the WSN is assigned  $K$  symmetric encryption keys randomly selected without replacement from a common Pool of  $P$  keys (*key pre-deployment phase*). When two sensors need to communicate securely, they must first find out which keys (if any) of the Pool they share (*shared-key discovery phase*). Then, they compute a common key as a function of the shared keys (*pairwise-key establishment phase*). This latter key is used to secure the channel by using a symmetric key encryption algorithm. Some solutions based on pseudo-random key assignment are presented in [37, 74, 75, 80, 144, 249]. However, these solutions show limited resiliency to tampering, as highlighted in [74, 75], or just require cooperating nodes to share a key with both principals. For the shared-key discovery phase different mechanisms have been proposed. In [85], the challenge-response and key index notification are proposed. With  $K$  keys stored in each sensor, the challenge-response requires sending and receiving  $K$  messages, to perform  $K$  encryption and, in the worst case,  $K^2$  decryption. With the

pseudo-random key index transformation proposed in [249], no message exchange are needed between sensors that want to establish a secure channel. Moreover, the attacker can compute the IDs of the keys stored by each sensor just acquiring the sensor ID; this solution shows a weakness similar to that in [85] and above exposed. The problem related to the information leakage of the keys' ID is solved in [74, 75]. Here the authors introduce a mechanism (ESP) that requires no message exchange for the shared-key discovery phase and reveals to the attacker no information about the keys it does not hold yet. Further, ESP provides probabilistic node authentication: A sensor can prove its identity by proving knowledge of the keys it is supposed to hold. The *Efficient and Secure Pre-deployment (ESP) scheme* works as follows. Consider a sensor  $a$ . For every key  $k_i^P$  of the pool, compute  $z = f_y(a \parallel k_i^P)$ , where  $f_y$  is a *pseudo-random function*, that is an efficient (deterministic) algorithm which given an  $h$ -bit seed,  $y$ , and an  $h$ -bit argument,  $x$ , returns an  $h$ -bit string, denoted  $f_y(x)$ , so that it is infeasible to distinguish the responses of  $f_y$ , for a uniformly chosen  $y$ , from the responses of a truly random function. Then, put  $k_i^P$  into the key ring of  $a$ , if and only if  $z \equiv 0 \pmod{(|P|/K)}$ . ESP supports a very efficient key discovery procedure. Consider a sensor  $b$  that is willing to know which keys it shares with sensor  $a$ . For every key  $k_j^b$  in the key ring of  $b$  sensor  $b$  computes  $z = f_y(a \parallel k_j^b)$ . Then, by testing  $z \equiv 0 \pmod{(|P|/K)}$ ,  $b$  discovers whether sensor  $a$  also has key  $k_j^b$  or not. Indeed, whoever already knows key  $k_i^P$  is the only one who can know whether  $k_i^P$  is in the key ring of  $a$  or not. This is computationally impossible for all other entities, since  $f_y$ , being a pseudo-random function, is also one-way and thus hard to invert [99]. For this reason, from the ID of a node an attacker cannot acquire neither the keys stored by this node, nor the corresponding key indexes:  $f_y(x)$  is applied to the actual value of the key, not to the corresponding key index. This kind of ID-based security could be thwarted only with the random capture of a large number of nodes (as shown later in this chapter) or via a node replication attack [52]. Finally, it is important to note that another property a WSN is required to enforce is connectivity. The connectivity problem was initially addressed in [85]; however, Ganeriwal et al. [76] revised and extended the model of connectivity in WSN.

## 2.3 Preliminaries and Assumptions

This section reports the notation and the assumptions that will be used in the following. For clarity, in Table 2.1 we list the symbols used in the chapter.

### 2.3.1 Security Requirements and Threat Model

We assume the following working hypothesis:

- **Communication infrastructure:** We assume an underlying routing mechanism such that any node can send a message (leveraging multi-hop) to any other node in the

**Table 2.1** Pair-wise key establishment: Notations

Symbol	Meaning
$N$	Number of sensors in the WSN
$P$	Size of the pool from which the keys are drawn
$K$	Number of keys assigned to each sensor (key-ring size)
$\mathcal{C}$	Set of cooperating sensors
$a, b$	Principals
$c_i$	$i$ th cooperating sensor, where $1 \leq i \leq  \mathcal{C} $
$w$	Number of corrupted sensor in the WSN
$k_i^a$	$i$ th key assigned to sensor $a$ , where $1 < i < k$
$k_{h,l}$	Key between sensors $h$ and $l$
$K_{h,l}$	Key computed with Direct Protocol [75], if possible, string of $0_s$ otherwise
$K_{h,l}^{\mathcal{C}}$	Key computed with Cooperative protocol [75]
$\bar{K}_{h,l}^{\mathcal{C}}$	Key computed with ECCE Protocol
$E_k(x)$	Encryption of string $x$ with key $k$
$D_k(x) = E_k^{-1}(x)$	Decryption of string $x$ with key $k$
$H(x)$	Hash function
$DH(x)$	Hash function doublehash, $DH(x) : \{0, 1\}^{ x } \rightarrow \{0, 1\}^{2 x }$
$LS(x)$	Less significant bits of string $x$ , where $ LS(x)  = \frac{ x }{2}$
$MS(x)$	Most significant bits of string $x$ , where $ MS(x)  = \frac{ x }{2}$

network. An example of such a globally addressable communications infrastructure is in [31];

- Sensors are randomly scattered in an unattended and often adversarial environments. To preserve their low cost, as well as to save power [9], they are not tamper proof [247]. Hence, we can assume that sensors can be physically captured and the attacker can acquire all the information stored within captured sensors;
- Good security engineering practice [8]: The algorithms, protocols and mechanisms that are employed to secure the WSN are publicly known. Only the keys in the sensors' key rings and in the Pool are initially secret. Moreover, the cryptographic primitives that are employed are at least computationally secure;
- Attacker model: We assume the strong node-compromise attacker model adopted in [36]. Specifically, we assume that the attacker is capable of compromising a fraction of the total number of nodes in the network and exposing the secret information contained within them. There can be two forms of node compromise.

In *passive node compromise* we assume that after node compromise, the attacker can only launch passive attacks such as eavesdropping. In *active node compromise* we assume that the attacker is also able to perform active attacks such as providing false routing metrics through the compromised node. We assume that the goal of the adversary is the exposure of the keys stored by the sensors, including those established with the ECCE Protocol.

- Channel establishment: The ECCE Protocol requires three phases, that is, key pre-distribution, shared-key discovery and channel establishment. We assume keys are assigned to sensors according to the ESP procedure [74], hence the first two phases are carried out as described in Sect. 2.2.

As for channel establishment, we will cope with this issue exploiting cooperating sensors, as described in Sect. 2.4. In the remainder of this chapter we assume that two sensors sharing one or more pre-deployed keys can compute a shared key via the Direct Protocol [75], that is the channel is built combining the keys the two principals share. The existence of a key established via the Direct Protocol translates into the existence of a Direct channel (that is a *link*) between the two sensors. Finally, we will use the term *corrupted* channel to refer either the fact that the keys the channel is built with are known to the attacker, or that the channel does not exist (that is the sensors do not share any key).

## 2.4 The ECCE Protocol

The ECCE Protocol involves, beyond the principals, other sensors (cooperating sensors). It is based on the fact that each distinguished cooperating sensor  $c_i$  can efficiently compute the keys it shares with each other cooperator. This can be efficiently done assuming that the key pre-deployment procedure is carried out according to ESP scheme, detailed in Sect. 2.2. Based on the keys cooperating sensor  $c_i$  shares with each other cooperator,  $c_i$  can compute  $(s_1, \dots, s_{i-1}, s_{i+1}, \dots, s_{|\mathcal{C}|-1})$ . These shared information are further combined to compute two values  $(v_1, v_2)$ . Each value is then sent to the two principals  $a$  and  $b$ . Each principal, upon receiving all the values provided by each cooperator, computes the key  $\bar{K}_{a,b}^{\mathcal{C}}$  that will be employed to secure communication between the two principals. The details of the protocol follow.

If sensor  $a$  wants to establish a secure channel with sensor  $b$ ,  $a$  chooses a set  $\mathcal{C} = \{c_1, \dots, c_m\}$  of cooperating sensors such that  $a, b \notin \mathcal{C}$  and  $m \geq 1$ . Then,  $a$  sends a request of cooperation to  $c_i$ , for each  $c_i \in \mathcal{C}$ . If a Direct key  $K_{a,c_i}$  between  $a$  and  $c_i$  exists, the request of cooperation is sent encrypted with  $K_{a,c_i}$ , else the request is sent not encrypted. The request carries the ID of  $b$  and the IDs of the sensors in  $\mathcal{C}$ .

Each sensor computes two different values  $(v_1, v_2)$ ,  $v_1$  to be sent to the sensor  $a$  and  $v_2$  to be sent to the sensor  $b$ . In particular, every cooperating sensor  $c_i$  computes the value  $v_1$  to be sent to sensor  $a$  as follows. The key  $K_{c_i,b}$  is computed and then hashed with the ID of  $a$  ( $ID_a$ ):  $H(ID_a, K_{c_i,b})$ . The keys shared with all other cooperating sensors are computed via the ESP protocol. For each cooperating sensor  $c_j$  where  $ID_{c_i} < ID_{c_j}$ , the hash  $H(ID_a \oplus ID_b, LS(DH(K_{c_i,c_j})))$  is computed; for each

cooperating sensor  $c_j$ , where  $ID_{c_i} > ID_{c_j}$ , the hash  $H(ID_a \oplus ID_b, MS(DH(K_{c_i,c_j})))$  is computed. The XOR of all the computed hash is executed and the resulting string  $v_1$  is sent to  $a$  encrypted with Direct key  $K_{c_i,a}$ . Note that, as exposed in Table 2.1, we have assumed that the Direct key procedure always returns a key. This key is  $K_{c_i,a}$  if it exists, or an appropriate string of 0s -a publicly known key- otherwise.

Every cooperating sensor  $c_i$  computes the value  $v_2$  to be sent to sensor  $b$  as follows. Key  $K_{c_i,a}$  is computed and then hashed with the ID of  $b$ :  $H(ID_b, K_{c_i,a})$ . The keys shared with all other cooperating sensors are computed and for each cooperating sensor  $c_j$  satisfying  $ID_{c_i} < ID_{c_j}$  the hash  $H(ID_a \oplus ID_b, MS(DH(K_{c_i,c_j})))$  is computed. For all cooperating sensors  $c_j$ , with  $ID_{c_i} > ID_{c_j}$ , the hash  $H(ID_a \oplus ID_b, LS(DH(K_{c_i,c_j})))$  is computed. The XOR of all the computed hash are executed and the resulting value  $v_2$  is sent to  $b$  encrypted with Direct key  $K_{c_i,b}$ .

For every cooperating sensor  $c_i$  from which sensor  $a$  receives a reply message  $v_1$  before time-out expires (let  $\mathcal{C}_r$  the set of replying cooperating sensors), sensor  $a$  computes the hash  $H(ID_b, K_{a,c_i})$  and then  $g_i = v_1 \oplus H(ID_b, K_{a,c_i})$ . When  $a$  either receives all the reply messages from the cooperating sensors, or the last time-out expires,  $a$  computes the ECCE key as follows:  $\tilde{K}_{a,b}^{\mathcal{C}} = K_{a,b} \oplus_{i=1}^{|\mathcal{C}_r|} g_i$ . The ECCE key  $\tilde{K}_{a,b}^{\mathcal{C}}$  is finally hashed and sent to  $b$ . The hashed ECCE key, when received by  $b$  could be used by  $b$  to check whether the locally computed ECCE key matches the ECCE key computed by  $a$ . Sensor  $b$  can set a time-out to limit the delay of expected messages. When the time out expires,  $b$  computes the ECCE key in a way similar to  $a$ , and will finally check whether the hash of the ECCE key received by  $a$  matches with the hash of the ECCE key locally computed.

Algorithm 1 shows the detailed pseudo-code of the ECCE Protocol. Algorithms 2 and 3 illustrate the functions COMPUTEMSGFORSOURCE and COMPUTEMSGFORDESTINATION used by every cooperating sensor to compute the message to be sent to  $a$  and  $b$  respectively. The  $H$  function used in the Protocol is employed to produce a non-invertible image of the keys, to avoid information leakage [37, 75]. The algorithm SELECTRANDOM selects the cooperating sensors in a pseudo random fashion among all the possible sensors in the network. However, note that this choice of cooperators could be unsatisfactory. For instance, the number of cooperators that share a key with the principals or with other cooperators might be small. This could affect the channel resilience, as we will see in Sect. 2.5.2. To cope with this problem, we could select cooperators according to some other policy. For instance, we could accept a selected cooperator only if it shares a Direct key with both principals. Note that it is not required that the key shared with principal  $a$  is the same key shared with principal  $b$ .

Involving cooperators allow the ECCE Protocol to be adaptive to different security requirements: If it is required to increase the channel resilience, than this objective can be achieved involving more cooperating sensors. The use of cooperators in the ECCE Protocol further allows to balance the burden of protocol execution among all the cooperators and the principals. Indeed, each cooperator computes  $|\mathcal{C}| + 1$  Direct keys and hash, while sending only 2 messages. If principal  $a$  wants to set up an ECCE key with principal  $b$ ,  $a$  has to forward  $|\mathcal{C}| + 1$  messages and to receive  $|\mathcal{C}|$  messages. Sensor  $b$  only needs to collect the  $|\mathcal{C}|$  messages sent to it by the cooperating sensors.

If a cooperating sensor in  $\mathcal{C}$  is not available (for instance, due to a node failure), using the time-out the protocol will not fail or deadlock, granting sensor failure resilience. Observe also that if there is not a Direct key between some of the cooperators involved in the protocol, this does not imply the failure of the protocol.

**Input** :  $b$  : ID of the receiving sensor.  
**Output**:  $\tilde{K}_{a,b}^{\mathcal{C}}$

```

1 begin
2    $\mathcal{C} = \text{SELECTRANDOM}(\text{NeededCooperators})$  ;
3   Set time-out  $\Delta$  ;
4    $\tilde{K}_{a,b}^{\mathcal{C}} = 0_s$  ;
5   forall the  $c_i \in \mathcal{C}$  do
6      $K_{a,c_i} = \text{DIRECT\_PROTOCOL}(c_i)$  ;
7      $a \rightarrow c_i :< a, c_i, E_{K_{a,c_i}}(\text{req\_coop} \parallel C \parallel b) >$  ;
8   end
9    $\mathcal{C}' = \mathcal{C}$  ;
10  while  $\mathcal{C}' \neq \emptyset$  and ( not elapsed( $\Delta$ ) ) do
11     $a \leftarrow c_i :< c_i, a, E_{K_{a,c_i}}(b, \text{COMPUTEMSGFORSOURCE}(a, b, c_i, C)) >$  ;
12     $b \leftarrow c_i :< c_i, b, E_{K_{c_i,b}}(a, \text{COMPUTEMSGFORDESTINATION}(a, b, c_i, C)) >$  ;
13     $s = E_{K_{a,c_i}}^{-1}(E_{K_{a,c_i}}(\text{COMPUTEMSGFORSOURCE}(a, b, c_i, C')))$  ;
14     $\mathcal{C}' = \mathcal{C}' - \{c_i\}$  ;
15     $\tilde{K}_{a,b}^{\mathcal{C}} = \tilde{K}_{a,b}^{\mathcal{C}} \oplus s \oplus H(ID_b, K_{a,c_j})$  ;
16  end
17   $\tilde{K}_{a,b}^{\mathcal{C}} = K_{a,b} \oplus \tilde{K}_{a,b}^{\mathcal{C}}$  ;
18   $a \rightarrow b :< a, b, H(E_{\tilde{K}_{a,b}^{\mathcal{C}}}) >$  ;
19 end

```

**Algorithm 1: ECCE Protocol**

**Input** :  $a$  : ID of the sensor (cooperator).  $b$  : ID of the source.  $c_i$  : ID of the receiving sensor.  $\mathcal{C}$  : Set of IDs of the others cooperating sensors.  
**Output**: Compute the message that the cooperator send to source sensor

```

1 begin
2    $\text{Msg} = H(ID_a, K_{c_i,b})$  ;
3   forall the  $c_j \in C_i (i \neq j)$  do
4     if ( $ID_{c_i} < ID_{c_j}$ ) then
5        $\text{Msg} = \text{Msg} \oplus H(ID_a \oplus ID_b, \text{LS}(\text{DH}(K_{c_i,c_j})))$  ;
6     end
7     if ( $ID_{c_i} > ID_{c_j}$ ) then
8        $\text{Msg} = \text{Msg} \oplus H(ID_a \oplus ID_b, \text{MS}(\text{DH}(K_{c_i,c_j})))$  ;
9     end
10  end
11 end

```

**Algorithm 2: COMPUTEMSGFORSOURCE**



<p><b>Input</b> : <math>a</math> : ID of the sensor (cooperator). <math>b</math> : ID of the source. <math>c_i</math> : ID of the receiving sensor. <math>\mathcal{C}</math> : Set of IDs of the others cooperating sensors.</p> <p><b>Output</b>: Compute the message that the cooperator send to receiving sensor</p> <pre> 1 begin 2   <math>Msg = H(ID_b, K_{c_i, a})</math>; 3   forall the <math>c_j \in \mathcal{C}_i (i \neq j)</math> do 4     if <math>(ID_{c_i} &lt; ID_{c_j})</math> then 5       <math>Msg = Msg \oplus H(ID_a \oplus ID_b, MS(DH(K_{c_i, c_j})))</math>; 6     end 7     if <math>(ID_{c_i} &gt; ID_{c_j})</math> then 8       <math>Msg = Msg \oplus H(ID_a \oplus ID_b, LS(DH(K_{c_i, c_j})))</math>; 9     end 10  end 11 end </pre>
---

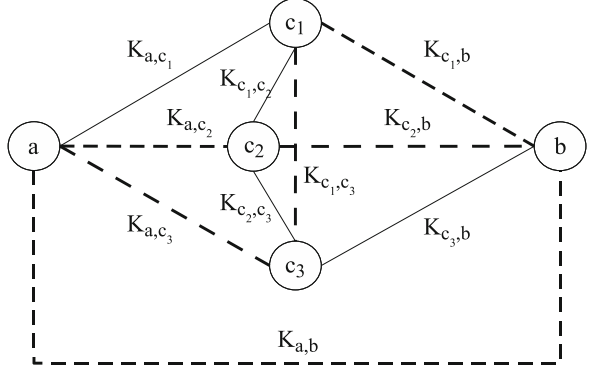
**Algorithm 3:** COMPUTEMSGFORDESTINATION

Sending the list of all cooperators to each  $c_i \in \mathcal{C}$  can help the attacker: It will be sufficient to corrupt a channel between the sender and one of the cooperators, and the set of cooperators would be disclosed. However, employing the ESP mechanism, the attacker cannot know the set of keys the cooperators hold, other than the subset of keys it already knows. Hence, the attacker is still forced to corrupt cooperating nodes if it wants to reduce its efforts [75]. Further, to decrease the possibility for the attacker to acquire the list of all the cooperators, the set  $\mathcal{C}$  of cooperators could be partitioned in subset  $\mathcal{C}_1, \dots, \mathcal{C}_q$ : Corrupting a channel or a cooperator within a specific subset  $\mathcal{C}_i$  does not reveal any information about the cooperators belonging to other subsets. This countermeasure has been implemented in a version of the ECCE Protocol that we will refer to as Partitioned ECCE.

## 2.5 Security Analysis

The condition that must be verified to guarantee the confidentiality of keys set-up using the ECCE Protocol, is the existence of a non corrupted path between the principals  $a$  and  $b$  ( $a - b$ ), where each link of this path is built with a Direct key and the intermediate nodes between  $a$  and  $b$  are the cooperating sensors. As an example, in Fig. 2.1 the sensors  $a$  and  $b$  use the ECCE Protocol to build a confidential key. In Fig. 2.1 the path composed of continuous lines signals a Direct key unknown to the attacker, while the dashed line signals a corrupted link or a non existing channel. In this example, the confidentiality of the established key  $\tilde{K}_{a,b}^{\mathcal{C}}$  is guaranteed by the existence of the path  $(a, c_1, c_2, c_3, b)$ . If the attacker does not hold the Direct keys used to secure the links  $(a, c_1)$ ,  $(c_1, c_2)$ ,  $(c_2, c_3)$ , and  $(c_3, b)$  there is no way to build the ECCE key.

**Fig. 2.1** Example of ECCE channel not corrupted:  
Existence of path  $(a, c_1, c_2, c_3, b)$



### 2.5.1 Channel Existence

In this section we analyze the probability that a pair of sensors succeeds to establish a confidential key using the ECCE Protocol. This probability depends on the probability of existence of Direct keys shared between all the possible pairs of sensors in  $\mathcal{C} \cup \{a\} \cup \{b\}$ . The probability to establish a Direct key is given by the probability that two sensors share at least one of the assigned keys of the Pool. From [85], it follows that:

$$\Pr[\text{link exists}] = 1 - \frac{\binom{P-K}{K}}{\binom{P}{K}} = 1 - \frac{K!(P-K)!(P-K)!}{P!K!(P-2K)!} \quad (2.1)$$

In the following to ease exposition, we assume that the existence of each link is independent from each other [37, 85]. We indicate with  $p$  the probability of existence of a single link. Further,  $path_{ECCE}(|\mathcal{C}|)$  represents the event of a path between principals (that can be established also via a direct link between principals), while  $path_{ECCE}(|\mathcal{C}|, nodir)$  accounts for the event of a path between principals but note that this path can be formed only through cooperating sensors (it is assumed that the direct link between principals does not exist). Then, we have:

$$\Pr[path_{ECCE}(|\mathcal{C}|)] = p + (1 - p) \Pr[path_{ECCE}(|\mathcal{C}|, nodir)] \quad (2.2)$$

The existence of a not corrupted direct link between the principals implies the existence of a non corrupted ECCE channel. Should this direct link do not exist, then the existence probability of a not corrupted channel is equal to the probability that at least one non corrupted path involving cooperating sensors do exists.

To compute this probability, take into consideration all the possible links of type  $(a, c_i)$  (grouped in the set  $l_{src}$ ) and  $(c_i, b)$  (grouped in the set  $l_{dst}$ ). Hence:

$$\begin{aligned}
 & \Pr[path_{ECCE}(|\mathcal{C}|, nodir)] \\
 &= \sum_{A=0}^{|\mathcal{C}|} \sum_{B=0}^{|\mathcal{C}|} \Pr[|l_{src}| = A] \Pr[|l_{dst}| = B] \\
 & \quad \cdot (\Pr[path_{ECCE}(|\mathcal{C}|, nodir) \mid |l_{src}| = A, |l_{dst}| = B]) \\
 &= \sum_{A=0}^{|\mathcal{C}|} \sum_{B=0}^{|\mathcal{C}|} p^A (1-p)^{|C|-A} p^B (1-p)^{|C|-B} \\
 & \quad \cdot (\Pr[path_{ECCE}(|\mathcal{C}|, nodir) \mid |l_{src}| = A, |l_{dst}| = B]) \quad (2.3)
 \end{aligned}$$

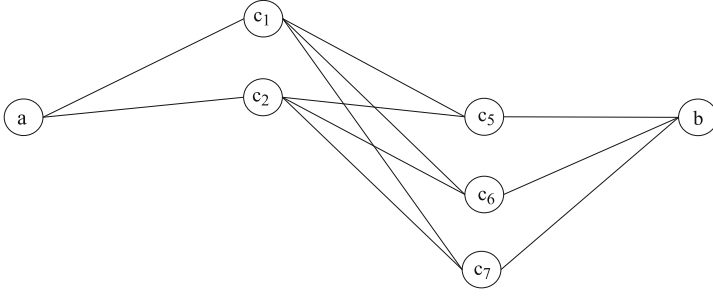
We remark that when  $A = 0$  we have a null probability of having a path between  $a$  and the set of cooperators; if  $B = 0$  then there are no paths between  $b$  and the sensors in  $\mathcal{C}$ . Let  $C(l_{src})$  and  $C(l_{dst})$  be the sets of cooperating sensors in  $\mathcal{C}$  that share a key with sensor  $a$  and  $b$  respectively. A path between the principals exists with probability 1 if  $C(l_{src}) \cap C(l_{dst}) \neq \emptyset$ . Fixing  $|l_{src}| = A$  and  $|l_{dst}| = B$  we have that:

$$\Pr[C(l_{src}) \cap C(l_{dst}) \neq \emptyset] = 1 - \frac{\binom{|\mathcal{C}|-A}{B}}{\binom{|\mathcal{C}|}{B}}$$

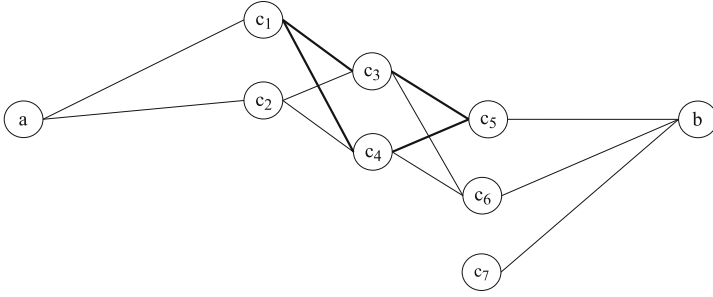
Equation. 2.3 can then be expressed as:

$$\begin{aligned}
 & \sum_{A=0}^{|\mathcal{C}|} \sum_{B=0}^{|\mathcal{C}|} p^A (1-p)^{|C|-A} p^B (1-p)^{|C|-B} \cdot \left( \left( 1 - \frac{\binom{|\mathcal{C}|-A}{B}}{\binom{|\mathcal{C}|}{B}} \right) \right. \\
 & \quad \left. + \Pr[path_{ECCE}(|\mathcal{C}|, nodir) \mid C(l_{src}) \cap C(l_{dst}) = \emptyset, |l_{src}| = A, |l_{dst}| = B] \right) \quad (2.4)
 \end{aligned}$$

We must now calculate the existence probability of the paths  $(a - b)$  that use more than one cooperating sensor; see for instance Fig. 2.1. In order not to incur in problems of dependency when dealing with probability, we will consider a modified scheme of the ECCE Protocol. This scheme helps the attacker because it excludes some cases in which the path would exist, hence this analysis will provide a lower bound to the security provided. In particular, we will only use the paths that join cooperating sensors in  $C(l_{src})$  (called  $s$ ) to cooperating sensors in  $C(l_{dst})$  (called  $d$ ), of length 1 or 2. As an example, Figs. 2.2 and 2.3 show the paths  $(s - d)$  of length



**Fig. 2.2** Paths  $(s - d)$  of length 1



**Fig. 2.3** Paths  $(s - d)$  of length 2

1 and 2 respectively, for one particular choice of  $\mathcal{C}$ ,  $l_{src}$  and  $l_{dst}$ . Finally, Eq. 2.2 can be expressed as:

$$\begin{aligned}
 \Pr[path_{ECCE}(|\mathcal{C}|)] &= p + (1 - p) \\
 &\cdot \left( \sum_{A=0}^{|\mathcal{C}|} \sum_{B=0}^{|\mathcal{C}|} p^A (1 - p)^{|C|-A} p^B (1 - p)^{|C|-B} \cdot \left( \left( 1 - \frac{\binom{|\mathcal{C}|-A}{B}}{\binom{|\mathcal{C}|}{B}} \right) \right. \right. \\
 &\quad \left. \left. + \left( 1 - \left( (1 - p)^{AB} (1 - p^2)^{(|C|-A-B) \cdot \min\{A, B\}} \right) \right) \left( \frac{\binom{|\mathcal{C}|-A}{B}}{\binom{|\mathcal{C}|}{B}} \right) \right) \right) \quad (2.5)
 \end{aligned}$$

These analytical results will be compared with simulation results in Sect. 2.6. The simulations performed support the behaviour predicted by the analytical model.

### 2.5.2 Channel Resilience

In this section we analyze the probability that the key established by the principals, using the ECCE Protocol, is not corrupted, that is not computable by the attacker through the information it holds. As discussed for the channel existence probability, also the resilience probability depends on the probability of resilience of the single Direct keys used in the construction of ECCE key. The probability that a Direct key is corrupted depends on the probability that a single key  $k_i$  of the Pool is corrupted. Considering  $w$  compromised sensors:

$$\Pr [\text{key } k_i \text{ is corrupted}] = 1 - \left(1 - \frac{K}{P}\right)^w \quad (2.6)$$

If one knows the probability that a key is corrupt, then it could be possible to calculate the probability that an existing link is corrupted.

$$\begin{aligned} \Pr [\text{link is corrupted} \mid \text{link exists}] &= \frac{\Pr [\text{link is corrupted} \cap \text{link exists}]}{\Pr [\text{link exists}]} \\ &= \frac{\sum_{i=1}^K (\Pr [\text{key is corrupted}])^i \Pr [i \text{ shared keys}]}{\Pr [\text{link exists}]} \\ &= \frac{\sum_{i=1}^K \left(1 - \left(1 - \frac{K}{P}\right)^w\right)^i \frac{\binom{K}{i} \binom{P-K}{K-i}}{\binom{P}{K}}}{1 - \frac{\binom{P-K}{K}}{\binom{P}{K}}} \quad (2.7) \end{aligned}$$

Replacing the probability given by Eq. 2.7 in Eq. 2.5 (Eq. 2.7 gives the value for probability  $p$ ), it is possible to obtain the probability that an ECCE channel is corrupted, assuming that all the pairs of cooperators share a Direct key. In a similar way, to assess the probability that an ECCE channel exists and is not corrupted, it is sufficient to replace the parameter  $p$  in Eq. 2.5 with the following formula:

$$\begin{aligned} &\Pr [\text{link exists}] \cdot \Pr [\text{link not corrupted} \mid \text{link exists}] \\ &= \Pr [\text{link exists}] \cdot (1 - \Pr [\text{link corrupted} \mid \text{link exists}]) \end{aligned}$$

Again, in Sect. 2.6 we will show that simulation results support the derived analytical model.

### 2.5.3 Probabilistic Authentication

Another important issue of WSN security is node authentication. For instance, any scheme for the revocation of misbehaving nodes has its basis on the certainty of

the nodes identities. Authentication can mitigate many dangerous attacks, like the replication of malicious sensors [79, 159]. In the following we discuss how the ECCE Protocol provides probabilistic authentication.

Based on the ECCE Protocol, every cooperating sensor  $c_i$  generates the messages to send by XORing the strings derived from the keys shared with the other cooperating sensors and the strings:

- $H(ID_a, K_{c_i,b})$ , for the message destined to the sender sensor;
- $H(ID_b, K_{a,c_i})$ , for the message destined to the receiving sensor.

Note that all cooperating sensors implicitly verify that both the principals have the keys that the principals should possess. This verification is possible due to the ESP mechanism. If a principal declares a false ID, cooperating sensors will use the keys that they should share with the sensor identified by ID. If just one of these keys is not possessed by the malicious principal that provided the fake ID the ECCE key cannot be established [74]. However, if the malicious principal possesses all the keys shared by the cooperators with the sensor identified by ID, the authentication process succeeds. For this reason the authentication is only probabilistic. Observe that, with respect to the Direct channel in which only one principal verifies the identity of the other party, in ECCE all the cooperating sensors verify the same identity with possibly different key-rings, hence the probability that a malicious sensor is not detected is smaller than in the Direct channel. In particular, since the authentication check performed by cooperators is carried out independently from each other, the probability that a fake principal succeeds in the authentication process, decreases exponentially with the number of cooperators involved. Further, note that the same mechanism supports authentication among the cooperating sensors as well. We remark that this authentication mechanism does not involve messages overhead other than the (limited) overhead required for the creation of the confidential channel.

## 2.6 Simulations and Discussion

In order to supply an experimental support to the analytical results developed in the previous section, we have performed extensive simulations. In particular, the ECCE Protocol has been compared with the following protocols:

- Direct [74];
- Cooperative [74];
- Extended Cooperative;
- MKR (Multipath Key Reinforcement) [37];
- Extended MKR;
- Partitioned ECCE (we have divided the set  $\mathcal{C}$  in independent subsets of size 2, 3 and 4).

We assume that in all the considered protocols the ESP mechanism [74] is used in the *shared-key discovery phase*. We introduce the Extended version of both the

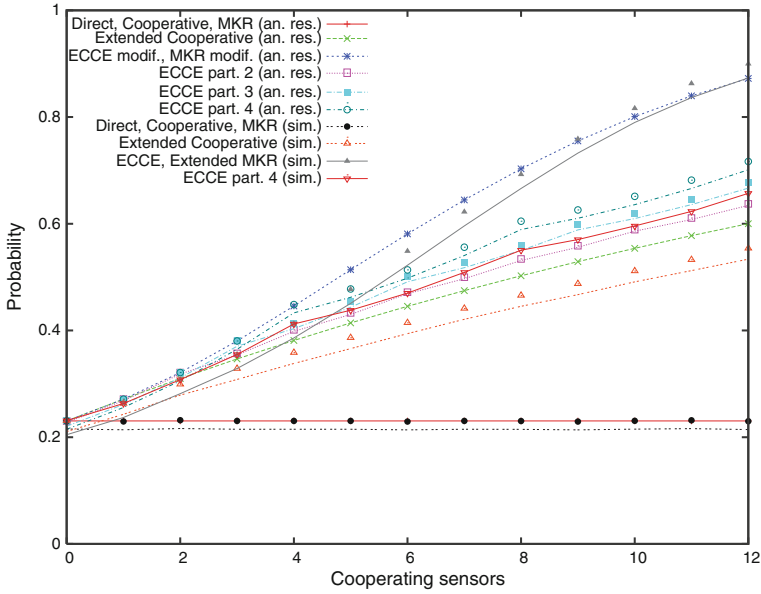
Cooperative and the MKR protocol, in which we assume that the existence of the Direct channel between the principals is not necessary. This optimization is based on the observation that, in the Cooperative and the MKR channel construction, the existence of the direct link between principals is only used to send some information to the receiving sensor; however, this information can be sent via a threshold scheme  $(t, c)$  [201] through the cooperating sensors.

We remark that a channel built according to the Cooperative Protocol, which requires that there is at least one shared key between sender and receiver, has the same existence probability of a channel established with the Direct Protocol. In fact, in both the Direct and the Cooperative Protocol, the necessary condition for the channel existence is the existence of the Direct link between the principals. Hence, the use of cooperating sensors in the Cooperative Protocol is only useful to increase confidentiality resiliency against the attacker, while the existence probability does not increase. This observation holds for the MKR Protocol as well.

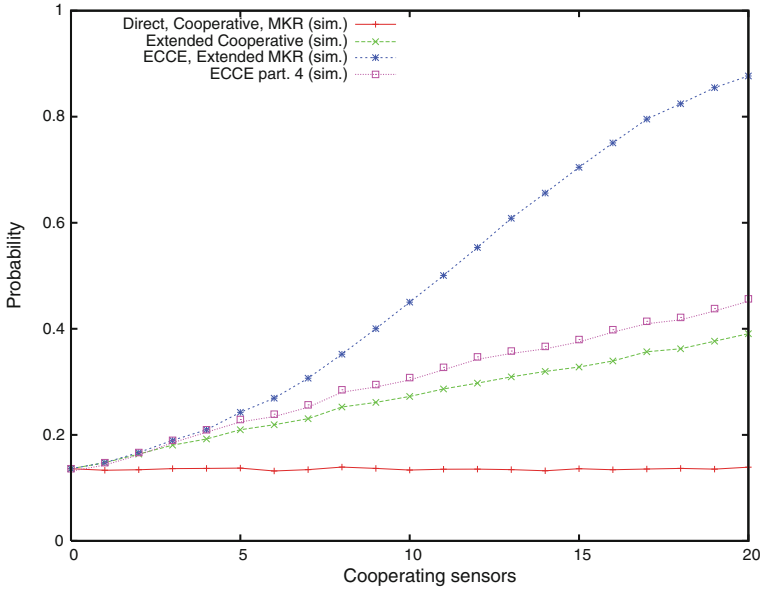
The Extended Cooperative needs only a direct link or a 2-hop path between principals realized through the cooperators in  $\mathcal{C}$  for the channel to exist. The behaviour of the Extended Multipath Key Reinforcement (MKR) [37], assuming a 2-hop MKR scheme as in [37], provides the same probability of channel existence and channel resiliency as the Cooperative. In both ECCE and Extended MKR the necessary condition for channel existence is the existence of a direct link between the principals or a path through the cooperators in  $\mathcal{C}$ . Hence, we can state that the existence probability of ECCE and the Extended MKR is roughly the same but, as discussed in Sect. 2.5.2, this equivalence does not hold as for the resilience, where ECCE performs better.

Figure 2.4 compares the analytical and experimental results as for channel existence. We have fixed  $P = 100$ ,  $K = 5$ , while  $|\mathcal{C}|$  ranges from 0 to 12. As expected, the assumption of independence among the links, used to ease the analysis in Sect. 2.5, implies an upper bound on the estimation of channel existence, as simulation results show (*an. res.* refers to the analytical results of Sect. 2.5 while *sim.* refer to simulation results). However, the simplified model used to analytically study the behaviour of the ECCE Protocol did not take into consideration some cases in which a channel between cooperators could exist. For this reason, when more than 9 cooperating sensors are involved in the channel establishment, the analytical results for ECCE are superseded by the simulation results. However, note that the difference between the simulation and the analytical results between the two slopes is tiny for the whole range of cooperating sensors considered. Observe that using no cooperating sensors, the behaviour of the ECCE, the Cooperative and MKR Protocol is similar to that of the Direct Protocol, while increasing the number of cooperators, the existence probability of the cooperative protocols (Cooperative, MKR and ECCE) increases as well. In particular, the ECCE Protocol provides better channel existence probability, and this probability improves with the number of cooperators.

Figures 2.5, 2.6, and 2.7 plot the existence probability of a secure channel established with the ECCE protocol, together with the same probability for the other protocols. These figures show the results obtained varying the number of cooperating sensors for  $P = 1000$  and  $K = 12, 15$  and  $20$  respectively. From these three figures it is possible to notice that increasing the key-ring size, the channel exis-

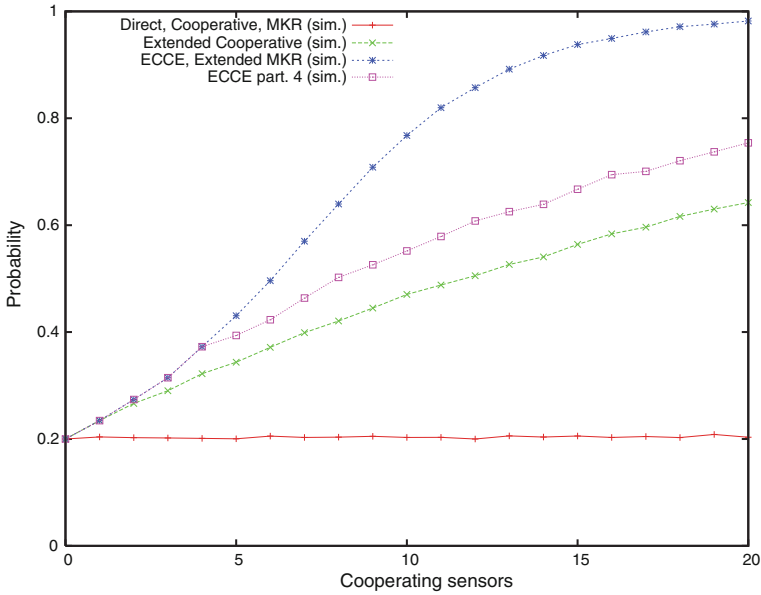


**Fig. 2.4** Channel existence: Comparison between analytical and simulation results for  $P = 100$ ,  $K = 5$



**Fig. 2.5** Channel existence:  $P = 1000$ ,  $K = 12$





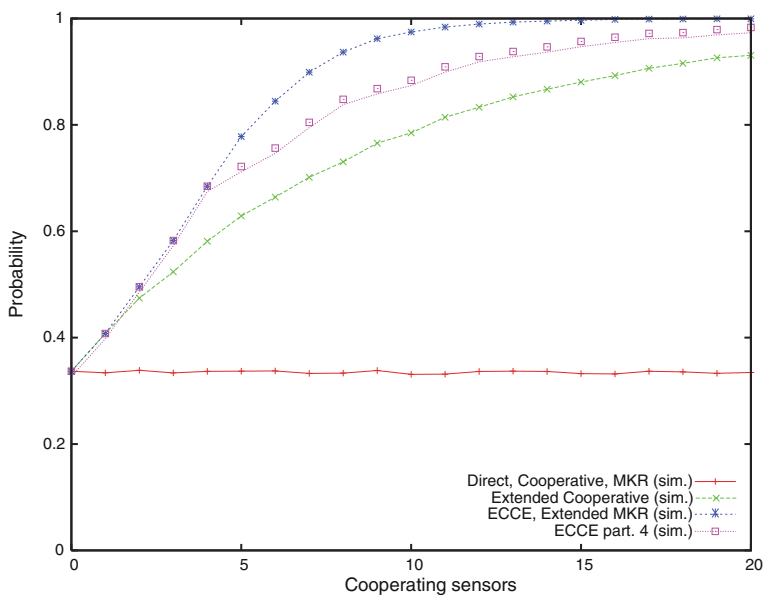
**Fig. 2.6** Channel existence:  $P = 1000$ ,  $K = 15$

tence probability increases as well. Hence, it is possible to obtain the same existence probability with different key-ring size, varying the number of involved cooperating sensors. The better performance of the ECCE Protocol, compared to the Cooperative one is due to the greater number of possible paths between  $a$  and  $b$  generated by the ECCE protocol. Indeed, the higher the number of possible paths, the higher the probability of channel existence, as analytically exposed in Sect. 2.5. For instance, in Fig. 2.1, the principals cannot set-up a secure channel via the Cooperative Protocol, while this is possible adopting the ECCE Protocol.

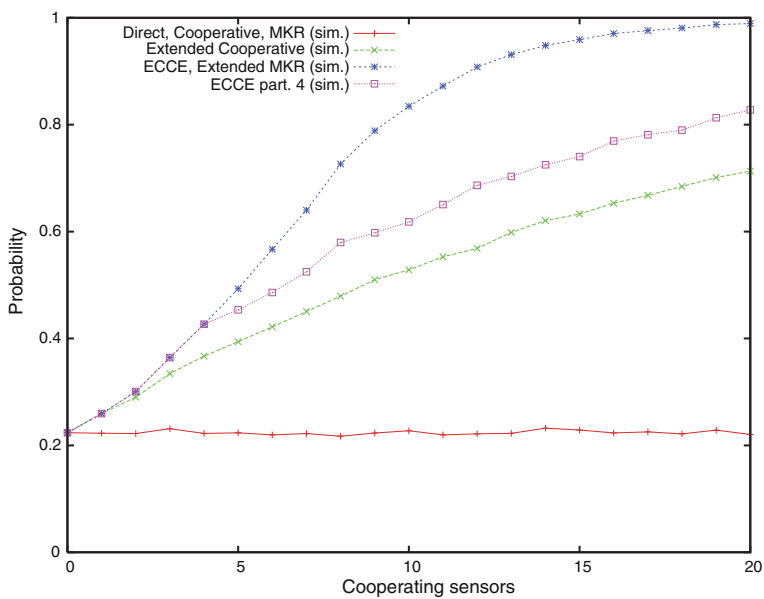
Figure 2.8 shows the existence probability of a channel for the different compared protocols, when  $P = 10,000$  and  $K = 50$ , while  $|\mathcal{C}|$  ranges from 0 to 20. We can notice that the curves behaviour is similar to that obtained in Fig. 2.6. This is because, as noticed in Sect. 2.5, the overall channel probability existence strictly depends on the existence probability of a single link.

Furthermore, we inquired the resilience of the established channels. In particular, we have performed our analysis assuming the existence between the two principals of at least the Direct channel, while the cooperating sensors are randomly selected. In Figs. 2.9, 2.10, 2.11, and 2.12 we report on the x axis the key ring size, while on the y axis the number of sensors to corrupt to compromise a channel, considering  $P = 10,000$  and 4 and 16 cooperating sensors, respectively.

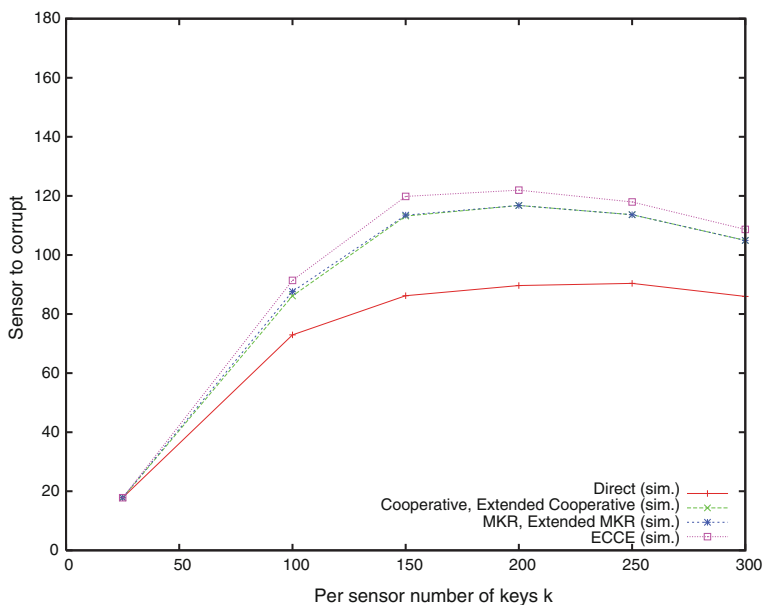
From Fig. 2.9, with 150 keys stored per sensor, the attacker has to capture about 82 sensors to corrupt a Direct channel; if 4 cooperating sensors are involved, the attacker has to corrupt about 113 sensors to corrupt a Cooperative channel, a little



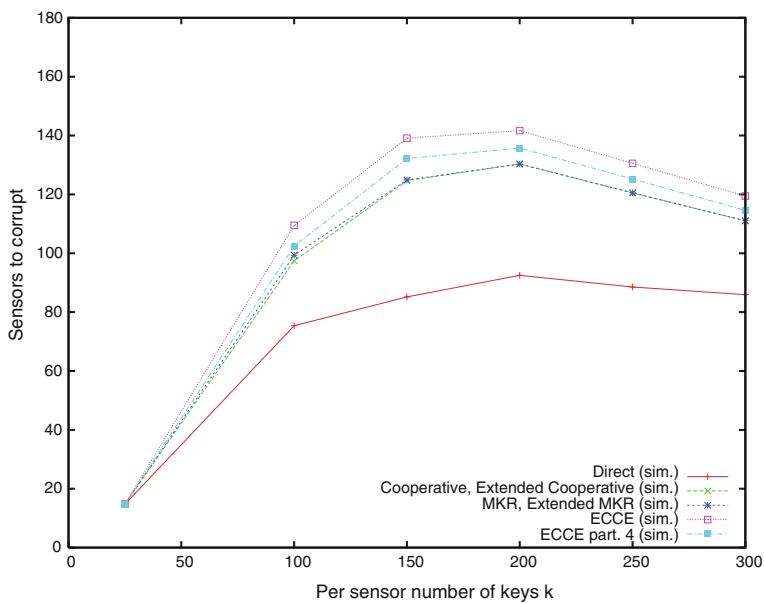
**Fig. 2.7** Channel existence:  $P = 1000$ ,  $K = 20$



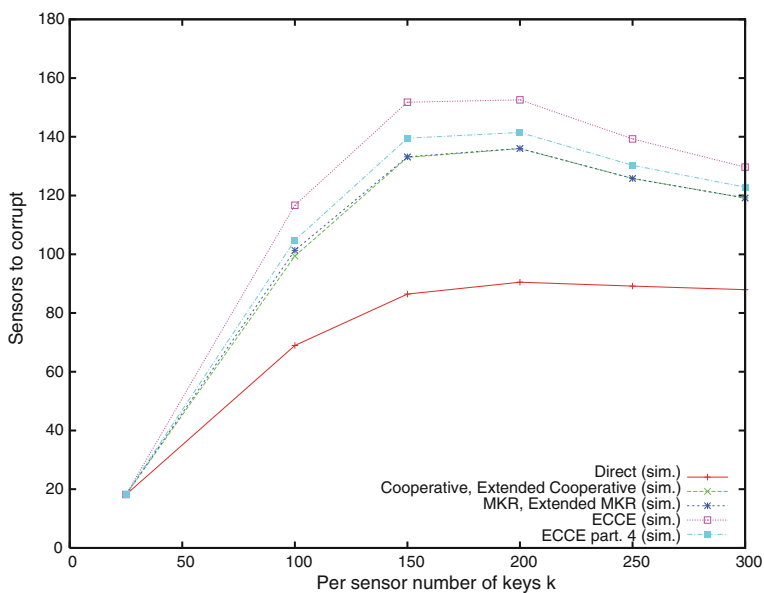
**Fig. 2.8** Channel existence:  $P = 10,000$ ,  $K = 50$



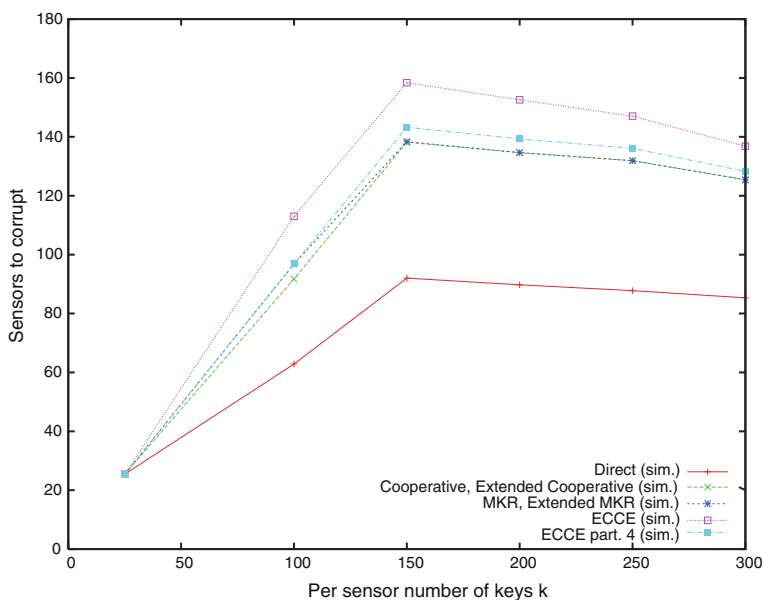
**Fig. 2.9** Resiliency:  $P = 10,000$ ,  $\mathcal{L} = 4$



**Fig. 2.10** Resiliency:  $P = 10,000$ ,  $\mathcal{L} = 8$



**Fig. 2.11** Resiliency:  $P = 10,000$ ,  $\mathcal{C} = 12$



**Fig. 2.12** Resiliency:  $P = 10,000$ ,  $\mathcal{C} = 16$

more to corrupt the MKR channel, while 120 sensors are required to corrupt an ECCE channel. Increasing the number of cooperating sensors also improves the resilience of these protocols. Indeed, in Fig. 2.12, with 16 cooperating sensors, for a key-ring size of 150, the attacker needs to corrupt about 138 sensors to corrupt the Cooperative, a little more to corrupt the MKR channel, while the attacker is required to compromise about 159 sensors to corrupt an ECCE channel. It is worth noticing that for all the simulated scenarios, the ECCE protocol performs better than the Cooperative protocol. In Figs. 2.9, 2.10, 2.11, and 2.12 the behaviour of the Direct Protocol is the same: The resilience of the Direct channel is not influenced by the number of cooperating sensors. From these figures we can also notice that the resilience to corruption of these protocol increases as the key ring size increases up to a certain value, after that value, the resiliency to corruption decreases. As observed in [85], this is due to the number of keys that the attacker can acquire tampering with a sensor.

Table 2.2 outlines the features of the ECCE Protocol, compared with the others protocols reported in Sect. 2.2. As can be seen, the ECCE Protocol benefits from all the features introduced by the use of cooperation among sensors. This explains why the ECCE Protocol performs better than the other protocols, as shown in the previous figures.

**Table 2.2** Features comparison of different protocols

Protocol	Features				
	Involve cooperating sensors	Principals mutual authentication	Usable in the case of no secret shared between principals	Authentication between cooperating sensors	Cooperating sensors that do not share keys with principals helps channel establishment
Direct [74]	No	–	–	–	–
Multipath key reinforcement [37]	Yes	No	No	No	Yes
Cooperative [74]	Yes	Yes	No	No	No
Extended cooperative (this chapter)	Yes	Yes	Yes	No	No
ECCE (this chapter)	Yes	Yes	Yes	Yes	Yes

## 2.7 Concluding Remarks

In this chapter we presented ECCE, a new cooperative protocol to establish a secure pair-wise communication channel between any pair of sensors in a WSN. The contributions are the following: This protocol does not require cooperating sensors to share a key with both principals for the channel between principals to be established. Also cooperating sensors that do not share any key with any of the two principals can help in the set up of the secure channel; cooperating sensors implement a probabilistic authentication of both principals as well as other cooperators. The probability that a fake principal or a fake cooperator could escape the authentication procedure decreases exponentially with the number of cooperators involved in the protocol; it is possible to trade-off key ring size with the number of cooperating sensors while preserving the same level of security. Note that this feature gives the possibility to have some memory available to store the ECCE keys, that could be used later for further use, hence amortizing the (limited) overhead incurred in the ECCE key set-up; the security provided by the protocol is adaptive with the level of threat in the WSN, on one hand, the higher the security threat, the more cooperators can be involved to enhance the resiliency of the channel; on the other hand a required downgrade on the required level of security can be implemented involving less cooperators, hence improving performances. Finally, in comparison with other protocols, ECCE shows better performances in channel existence and channel resilience even when the number of involved cooperators is small.

While the aim of this chapter is to propose an efficient way to involve cooperating nodes in the pair-wise key establishment, we note that it is also interesting to study how the network density influences the availability of neighbour nodes; a further detailed study on the energy consumed by the protocol would be of interest as well. As for comparing the proposed protocol with the current solutions in the literature, we only considered other probabilistic algorithms: we did not consider deterministic solutions because of the drawbacks described in Sect. 2.2. A broader comparison (e.g. for the channel resilience) of the proposed solution against the deterministic solutions like the one in [22] would also be of interest. We leave these points as future works.

As discussed in Sect. 2.5.2, the main threat to the protocol presented in this chapter comes from the fact that the attacker can capture nodes—acquiring the secret material stored in their memory. If the network were able to know the ID of the captured node it could react somehow. As an example, the pre-deployed keys associated with the captured ID could be revoked—not considered secure anymore. Nodes that do not share anymore a secure channel would stop communicating or will just use a multihop path to communicate or to share a new secret key. Next chapter provides a mechanism to deal with the detection of the node capture.



<http://www.springer.com/978-1-4939-3458-4>

Secure Wireless Sensor Networks

Threats and Solutions

Conti, M.

2015, XIII, 169 p., Hardcover

ISBN: 978-1-4939-3458-4