Establishing secure links in low-rate wireless personal area networks

Maurizio Adriano Strangio

Abstract This paper presents a provably secure and efficient key agreement protocol (SNKE) using private key authentication. The distinguishing features of protocol SNKE are: (a) ease of implementation in the 802.15.4 stack (it makes use of the cryptographic services provided by the media access layer); (b) availability of two operation modes (hash-chain and key-renewal modes) with forward secrecy achieved (in key-renewal mode) with a modest computational effort.

In addition, the key distribution scheme, which may be either based on group keys or pairwise keys, combined with both operation modes offers effective levels of protection against long-term key compromise.

The protocol was designed to meet the strict power and energy constraints of low-rate wireless personal area networks (802.15.4 WPANs). Indeed, the foreseeable applications include the deployment of standard-compliant secure wireless sensor networks (WSNs).

1.1 Introduction

The latest version of the IEEE 802.15.4 standard [16] provides a comprehensive specification for the physical and media access control layers of low-rate wireless personal area networks (LR-WPAN). LR-WPANs are essentially aimed at supporting low cost, low power and reliable applications such as industrial monitoring and control, home automation, sensor networks and medical solutions. The standard specifies two different device types: (a) full-function devices (FFDs) and (b) reduced-function devices (RFDs). A FFD may operate in PAN coordinator, coordinator or device modes and interacts with both RFDs and FFDs. On the other hand, an RFD is intended for lightweight applications and can communicate only with FFDs. Three networking topologies are generally supported, i.e. star, peer-to-peer

Department of Mathematics, University of Rome “Roma Tre”, Italy
e-mail: strangio@mat.uniroma3.it

or cluster-tree. RFDs can associate with a single FFD at a time thus forming only star network topologies. A network shall include at least one PAN coordinator which may have greater c! omputational resources than any other node in the PAN.

Over-the-air networks are inherently less secure than wired networks since attackers with the technology can intercept protocol transcripts with little effort (e.g. by making use of a portable computer with a 802.15.4 compliant radio interface or a scanner). Therefore, many applications need to ensure the confidentiality and integrity of the data flowing among the communicating nodes in the LR-WPAN. Furthermore, in hostile environments the opponent may succeed in obtaining the long-term keying material stored on a node. As a result, the attacker not only may learn confidential data exchanged in past communications (secured with previously established sessions keys) but, even worse, could modify and inject messages to influence the events for her own advantage (e.g. sensor networks deployed on a battlefield).

In this paper we present protocol SNKE, a secure and efficient key agreement scheme for establishing secure links in a LR-WPAN. The protocol makes use of private key cryptographic primitives to account for the energy constraints of devices forming the network and uses long-term shared keys to authenticate the communicating principals. In terms of energy consumption it is well known that private key algorithms are superior to public key cryptography.

Recently, many schemes based on pre-distributed keys have been proposed to avoid the computational overhead required by key agreement protocols; the general approach is to use either a unique network-wide shared key or a set of keys randomly chosen from a key pool so that two nodes share at least one key with high probability. However, the main drawbacks of these schemes is the lack of scalability and a higher vulnerability to node exposure.

To avoid the above shortcomings, protocol SNKE incorporates a key renewal mechanism; the long term private key shared by any two principals is replaced by a new value at the completion of the protocol execution. As a result, the protocol enjoys forward secrecy while requiring only a modest computational load on low resource devices (RFD nodes). As an additional benefit, protocol SKNE can be effectively implemented on top of the security services offered by the 802.15.4 media access control layer (WPAN-MAC).

1.2 Related work

There is a relatively small number of key agreement protocols based on symmetric key techniques. This is so mainly because key management is considered troublesome with symmetric key cryptosystems although the resulting algorithms are much more efficient. On the other hand, public-key cryptosystems can simplify the key management process but it turns out that ensuring the authenticity of the public keys is not less trivial than distributing symmetric keys.

Table 1.1 considers the most significant properties of key agreement protocols and compares several well known schemes found in the literature with protocol SNKE. As shown in column one, protocol SNKE competes with the others in terms of the required message flows. Column two enumerates the cryptographic primitives used as the basic building blocks, with ENC standing for "symmetric encryption" and MAC for "message authentication code" (the number in the parenthesis counts the invocations of the primitive for each side in a run of the protocol). Column three reveals whether the protocols provide key confirmation while column four clearly indicates that protocol SNKE is the only one providing forward secrecy (FS). Finally, column five indicates whether there are successful attacks against the protocols published in the literature.

Note that we do not consider key agreement protocols that are not strictly based on private key cryptography (even if they are more recent) or that require a trusted on-line third party (e.g. Kerberos key distribution); for example, SEKEN [18] albeit being an interesting proposal makes use of public keys (it is a key transport protocol) and also requires a base station to maintain an updated map of the network topology.

Table 1.1 Comparison of symmetric-cryptography key agreement protocols

↓*Protocol*/*Property*→	*Flows*	*Primitives*	*Key. Conf.*	*FS*	*Attacks*
Andrew RPC[28]	4	ENC(4)	-	No	Yes
2PKDP[19]	3	ENC(2)	A,B	No	No
ISO/IEC Mech. 5[17]	2	ENC(2)	A	No	No
ISO/IEC Mech. 6[17]	3	ENC(2)	A	No	No
AKEP1 [5]	3	ENC(1),MAC(2)	A,B	No	Yes
SNKE	3	ENC(1),MAC(1)	A,B	Yes	No

1.3 Protocol Specification

1.3.1 Preliminaries

If X is a finite set then $x \xleftarrow{R} X$ denotes the sampling of an element uniformly at random from X. If α is neither an algorithm nor a set $x \leftarrow \alpha$ represents a simple assignment statement. The symbol $\oplus$ denotes bitwise *exclusive or* on strings of equal length. Let $\mathtt{H} : \{0,1\}^* \rightarrow \{0,1\}^\ell$ denote a cryptographic hash function. We assume that hash functions behave like random functions (random oracles [6]).

A *keyed message authentication code* (MAC) is a 3-tuple of polynomial time algorithms $\langle \mathtt{key}(\cdot), \mathtt{tag}_\kappa(\cdot), \mathtt{ver}_\kappa(\cdot) \rangle$ where (a) $\mathtt{key}(\cdot)$ is a randomized key generation algorithm that returns a cryptographic key κ when the input is 1^ℓ (ℓ is the security parameter); (b) $\mathtt{tag}_\kappa(m)$ is a (randomized) algorithm that accepts message $m \in \{0,1\}^*$, key κ in input and returns a tag τ; (c) $\mathtt{ver}_\kappa(\tau)$ is a (deterministic) tag verification algorithm that accepts (m, τ) in input and returns 1 iff $tag_\kappa(m) = \tau$

otherwise it returns 0. We also require a correctness condition where for all m in the message space if $\mathtt{tag}_\kappa(m) = \tau$ then $\mathtt{ver}_\kappa(m,\tau)$ should output 1. The standard security notion for MACs requires resistance to strong unforgeability against chosen-message ! attacks (SUF-CMA, [4]).

A *private key encryption scheme* (ENC) is a 3-tuple of polynomial time algorithms $\langle \mathtt{key}(\cdot), \mathtt{enc}_\kappa(\cdot), \mathtt{dec}_\kappa(\cdot)\rangle$ where (a) $\mathtt{key}(1^\ell)$ is a randomized key generation algorithm that returns a cryptographic key κ on input the security parameter ℓ; (b) $\mathtt{enc}_\kappa(m)$ is a (randomized or stateful) algorithm that accepts a message $m \in \Sigma^*$ and key κ in input and returns a ciphertext c; (c) $\mathtt{dec}_\kappa(c)$ is a (deterministic and stateless) algorithm that on input c, κ returns m (iff $\mathtt{dec}_\kappa(\mathtt{enc}_\kappa(m)) = m$). For a private key encryption scheme the standard security notions include *indistinguishability* of encryptions [15] and *non-malleability* [13] under chosen plaintext attacks (CPA) and chosen ciphertext attacks (CCA).

1.3.2 Protocol SNKE

Protocol SNKE (Figure 1.1) is an efficient key agreement protocol requiring only three messages to be exchanged. A symmetric authentication setting is assumed, i.e. a principal believes that its communication peer holds the shared key (securely) distributed prior to the actual communication and trusts that its partner is honest (private keys are not deliberately revealed to malicious third parties). The protocol provides key confirmation, implying that only the two principals involved in the communication should be able to establish the session key (since computation of the session key by each party requires knowledge of the shared secret), provided they were not corrupted before the protocol run. This appealing property, which makes the protocol resistant to *adaptive corruptions* [29, 9] and allows universal composability [10], is achieved by making use of a MAC to (explicitly) authenticate the transcripts with a (confirmation) key derived f! rom the actual session key. The protocol is endowed with two modes of operation:

hash-chain mode: principals A, B running the protocol establish a pair of session keys (respectively sk_A, sk_B) that are used to seed a hash chain of encryption keys. In other words, the ith message sent from A to B is encrypted under the key $sk_{A,i} = \mathtt{H}(sk_{A,i-1})$ and key $sk_{A,i-1}$ is discarded (vice versa, the jth message from B to A is encrypted with $sk_{B,j}$). With this option enabled, protocol SNKE may be run once to seed the scheme of Mauw *et al*. [23] thus allowing bidirectional communication (the protocol in [23] supports only unidirectional communications);

key-renewal mode: principals A, B running the protocol receive in their local output a session key sk for subsequent encryption and a new shared long-term key $\mathtt{K}'$. As a consequence, communications secured with previously established session keys are inaccessible if the parties are eventually corrupted (thus offering forward secrecy).

We point out two distinguishing features of protocol SNKE: (a) it is easily implemented in the 802.15.4 stack by making use of the cryptographic services provided by the media access layer; (b) with key-renewal mode forward security is achieved at the expense of a small resource expenditure (since it makes use of symmetric key cryptography). Observe that if the pre-shared key $\mathtt{K}$ is discarded by A, B in hash-chain mode the protocol also maintains forward secrecy. The main actions of the protocol are outlined below (refer to figure 1.1 for further details):

1. Principal A selects r_A (challenge nonce) at random from $\{0,1\}^\ell$ where $\ell \geq 64$, computes the ciphertext c_A under the symmetric key $\mathtt{K}$ and sends it to B;
2. Principal B decrypts message c_A, checks whether the resulting cleartext is correct (i.e if it contains the identity of its intended peer A), carries on by selecting r_B at random from $\{0,1\}^\ell$ and computes the 3-tuple of keys κ_B, χ_B, η_B. Thereafter, B computes the ciphertext c_B, the tag t_B authenticating message c_B, sends c_B, t_B to A and erases r_B, κ_B from its memory;
3. On receipt of c_B, t_B principal A decrypts message c_B and checks whether the resulting cleartext is correct (i.e if it contains the identity of its intended partner B). Subsequently she computes the 3-tuple of keys κ_A, χ_A, η_A verifies the tag t_B (rejecting the connection request in case of failure), sends the tag t_A to B and discards r_A, κ_A from its memory;
4. On receipt of t_A, provided the tag verification procedure is successful, principal B returns either $\mathtt{K}, \eta_B$ or χ_B, η_B depending on the operation mode (otherwise it returns null). Analogously for principal A;

Observe that in key-renewal mode when protocol SNKE returns, the calling application erases key $\mathtt{K}$ and replaces it with the new value $\mathtt{K}'$. For additional security, long-term keys may be stored in tamper-proof modules (when available); in this case all operations involving such keys would be performed inside the module (e.g. to compute χ_A, χ_B). On-line computations for both principals (regardless of the operation mode) involve one encryption and one decryption, computing and verifying a MAC tag and a hash value calculation.

1.4 Remarks on the Key Management Model

In this section we briefly comment on the key management models that may be applied with protocol SNKE. In general, adopting the appropriate scheme requires the designer to seek for an acceptable trade-off between security (resilience versus the adversarial model) and the required resource expenditure (e.g. memory usage).

The most common types of keying models are those wherein either: (1) a *network shared key* is distributed to the entire network; (2) *pairwise keys* are established among the nodes or (3) the network is partitioned into sets of nodes which are assigned *group keys*.

The network shared key model is the simplest scheme since it allows full connectivity with only one key to manage and small memory usage. However, the all

A : $\mathtt{K} = \mathtt{K}_{AB} = \mathtt{K}_{BA}$
B : $\mathtt{K} = \mathtt{K}_{AB} = \mathtt{K}_{BA}$
$\mathtt{OpMode}$: $0 = \mathtt{key\text{-}renewal}, 1 = \mathtt{hash\text{-}chain}$

A : $r_A \xleftarrow{R} \{0,1\}^\ell$
$c_A \leftarrow \mathtt{enc}_{\mathtt{K}}(r_A, A)$
$A \rightarrow B$: c_A
B : $r_A, B \leftarrow \mathtt{dec}_{\mathtt{K}}(c_A)$
$r_B \xleftarrow{R} \{0,1\}^\ell$
$c_B \leftarrow \mathtt{enc}_{\mathtt{K}}(r_B, B)$
$\kappa_B, \chi_B, \eta_B \leftarrow \mathtt{H}(r_B, r_A, A, B, \mathtt{K})$
$t_B \leftarrow \mathtt{tag}_{\kappa_B}(c_B, r_A, A)$
$B \rightarrow A$: c_B, t_B
A : $r_B, A \leftarrow \mathtt{dec}_{\mathtt{K}}(c_B)$
$\kappa_A, \chi_A, \eta_A \leftarrow \mathtt{H}(r_B, r_A, A, B, \mathtt{K})$
if $\mathtt{ver}_{\kappa_A}(t_B) \neq 1$ then $\mathtt{reject}$ endif
$t_A \leftarrow \mathtt{tag}_{\kappa_A}(c_A, r_B, B)$
$A \rightarrow B$: t_A
B : if $\mathtt{ver}_{\kappa_B}(t_A) \neq 1$ then $\mathtt{reject}$ endif
if $\mathtt{OpMode}$=0 then return $\mathtt{K}' \leftarrow \mathtt{K} \oplus \chi_B, sk_B \leftarrow \eta_B$
elseif $\mathtt{OpMode}$=1 then return $sk_A \leftarrow \chi_A, sk_B \leftarrow \eta_B$
endif
A : if $\mathtt{OpMode}$=0 then return $\mathtt{K}' \leftarrow \mathtt{K} \oplus \chi_A, sk_A \leftarrow \eta_A$
elseif $\mathtt{OpMode}$=1 then return $sk_A \leftarrow \chi_A, sk_B \leftarrow \eta_B$
endif

Fig. 1.1 Protocol SNKE

powerful adversary can gain complete control of the entire network by compromising any one of the nodes. Therefore, this scheme is detrimental if there is reason to believe that nodes may be compromised with non negligible probability (e.g. in an extremely hostile environment).

Many applications are aimed at monitoring the interactions between objects and the surrounding premises; the resulting network topology is often hierarchical with data paths optimized for the computation of aggregate data from subsets of nodes (in other words a node needs no interaction with all other nodes in the network). To this end, it may be advantageous to employ a key distribution scheme to establish pairwise links among communicating nodes. Networks established with pairwise keys are more resilient to attacks since the adversary that is able to compromise a node only learns the set of keys used by that node to communicate with neighboring devices. However, the degree of network connectivity determines the number of keys to be stored in each node (a fully connected 1-hop network having n nodes requires storing $n-1$ pairwise keys per node).

In the group key model a shared key is used by a set of nodes to establish secure links between any two nodes in the group. With this scheme one achieves a reasonable trade off between the worst case resilience to attacks displayed by the network shared key model and the large resource expenditure required by the pairwise key model.

Protocol SNKE may be profitably used either with pre-distributed group keys or pairwise keys. We assume that all nodes sharing a pre-defined key are within communication range (either because they were statically deployed over the target area or an additional self-organising mechanism is employed to bring key-coupled nodes in each others neighborhood).

A group key may be assigned to a set of RFD nodes and to one or more coordinator nodes[1] For example, the group of RFD nodes N020, N021, N022 and the FFD coordinator N02 in Figure 1.2 may have in common a group key; the nodes N0, N01, N02, N03, N012 also constitute a group based on a different key (unique in the whole network with N0 acting as the PAN coordinator).

When protocol SKNE is run in key-renewal mode, resilience to node compromise (within a group) increases with time; optimum resilience is achieved after each node has run the protocol with the coordinator node at least once (thereafter the protocol is invoked when the nodes need to communicate again with the coordinator). In hash-chain mode, the group key does not change and it suffices that each device runs the protocol only once with the coordinator since both will use the established keys for subsequent communications. Note that in this case the network (group) is not resilient to node compromise (unless as discussed before the base key is discarded; however, devices will no longer be able to communicate with other nodes in the group). Now assume that pairwise keys are pre-distributed to nodes in the WPAN. If protocol SNKE is run in key-renewal mode, compromise resilience of network nodes remains constant through out the lifetime of the WPAN (there is no need to wait until! all RFDs have run the protocol at least once with the coordinator as with group keys); i.e., forward secrecy is ensured for all sessions. Analogously, in hash-chain mode the main difference with respect to group keys is the stronger resilience to node compromise. We conclude this section by mentioning some noteworthy schemes for the key distribution problem (there is a large body of recent work in the literature concerning this problem). In [1] the authors introduce the concept of *pebbles* which are large ad-hoc networks of small resource-constrained devices that are assigned a secret group key. Communication among the devices of the same pebble is encrypted using the Traffic Encryption Key (TEK) which is derived from the group key. However, the proposed mechanism lacks a detailed analysis of the resources needed in the computation and the overhead for the subsequent distribution of the keys. Another open issue concerns the effectiveness of the probabilistic algorithm used to select the manager at each round of the key updating process. The probabilistic key distribution scheme of [14] is based on the assumption that every node in the network needs to communicate with all other nodes; this is a somewhat unp! ractical scenario for real world networks. Their scheme was later extended by [11] to allow more sophisticated probabilistic algorithms for establishing pairwise keys among sensor devices. Moreover, the authors in [22] developed a general framework called pool-based key pre-distribution wherein the schemes of [14, 11] can be considered as particular instances. These papers are all inspired by the work of [8] which describes a protocol for group key distribution based on the evaluation

[1] RFDs must store a shared key for each potential coordinator FFD they need to associate with.

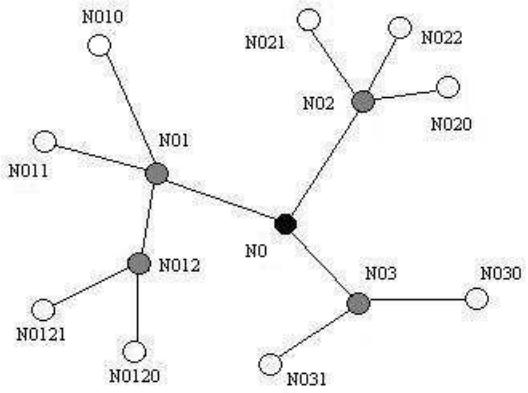

Fig. 1.2 A simple cluster tree WPAN. The black circle designates the PAN-coordinator, the gray circles indicate FFD nodes and the white circles RFD nodes. The lines between two nodes indicate that a parent-child relationship has been established due to a previous communication.

of shared polynomials. A disadvantage with all the preceding schemes is that they imply heavy computational loads on devices. The work of [26] mandates the use of a base station to distribute keys. This is not always convenient since it introduces a single point of failure into the network.

1.5 Security of Protocol SNKE

Rather than informally arguing about the desirable security properties (e.g. key independence (KI), forward secrecy (FS), etc) we adopt a more rigorous approach by proving the security of protocol SNKE in the formal model of distributed computing of Canetti-Krawczyk [9]. This model is currently considered the most comprehensive complexity-theoretic framework for protocol security analysis.

1.5.1 The Canetti-Krawczyk model

In this section we recall the main concepts of the Canetti-Krawczyk model (the presentation is mostly adapted from [9]). A `key-exchange` (`KE`) protocol is run in an interconnected network of machines (principals) executing instances of the protocol called `KE-sessions`. The set of principals are usually denoted by the letters P_i with $i = 1, \ldots, n$ (P_i's unique id corresponds to the subscript i). The input

to the kth running instance of a protocol (`KE` session) within principal P_i is of the form $(i, j, s_{ik}, role)$ where j is the identity of the partner, s_{ik} is the session identifier and $role$ is either `initiator` or `responder` (the tuple $i, j, s_{ik}, role$ identifies the session within P_i). A session within P_i and a session within P_j are `matching` if their inputs are respectively of the form $(i, j, s_{ik}$,`initiator`) and $(j, i, s_{jl}$,`responder`) with $s_{ik} = s_{jl}$.! Matching sessions play an important role in the definition of security.

The adversary is an active "man-in-the-middle" malicious party that can intercept and modify messages, delay or prevent their delivery, inject messages of her own choice, interleave messages from different sessions, etc (i.e., the adversary is the network). To account for the potential disclosure of secret information, the adversary is allowed to perform the following types of operations on a session:

1. (`session-key query`,i, s_{ik}): the adversary learns the session key sk_{ik} corresponding to the complete session s_{ik};
2. (`party corruption`,i): the adversary learns *all* the information stored in the memory of P_i (including long-term private keys, session specific data and session keys). The result of this query depends on the current state of the protocols running within the principal; if P_i has not invoked any sessions then it returns the long-term private key otherwise it returns long-term private keys, session-specific data and session keys of completed and unexpired sessions (which have not been explicitly erased). From thereon all the actions of a corrupted principal are controlled by the attacker;
3. (`session-state reveal`,i, s_{ik}): this query is asked of a non-complete session s_{ik}. The adversary learns the (internal) session state for that particular session (which may include, for example, the secret ephemeral nonces but not the long-term private key used across all session at the party). The need for this query, in addition to party-corruption and session-key queries, stems from the possibility that private keys are given better protection than ephemeral session-specific data (e.g by using tamper-proof security modules);
4. (`session expiration`,i, s_{ik}): this action can be scheduled by the adversary for any session s_{ik} which is complete. As a result, the session key s_{ik} is erased from P_i's memory. The adversary is not allowed to perform a `session-key query` query of an expired session;

A `KE-session` $(i, j, s_{ik}, role)$ is *exposed* if the adversary has scheduled any one of the following actions on the session: (a) a `session-state reveal` query; (b) `session-key query`; (c) `party corruption` (of P_i) before the session has expired. Also, the session is exposed if its matching session $(j, i, s_{jl}, role')$ has been exposed (since matching sessions output the same session key).

The above model refers to the *unauthenticated-links* (UM) model and the corresponding adversary $\mathscr{U}$ is called the UM-adversary. There is also the *authenticated-links* (AM) model, wherein the AM-adversary [3], denoted $\mathscr{A}$, cannot inject or modify messages or neither deliver a specified message more than once. The rationale for having two models is explained by the notion of protocol *emulation*. Informally, a protocol Σ' *emulates* protocol Σ in the UM if for any adversary that interacts with Σ' in the UM there exists an adversary that interacts with the Σ in the AM such

that the two interactions are computationally indistinguishable. Specific protocols, called *authenticators* (or compilers), on input the description of a protocol Σ, output a description of a protocol Σ' such that Σ' emulates Σ in the UM. Examples of authenticators are given in [3].

Formalisation of the security of a KE protocol follows the definitional approach of [5]. The resultant notion of an SK-secure protocol captures the requirement that the adversary is unable to obtain the session key of an unexposed session. An SK-game is defined wherein the goal of an UM-adversary $\mathscr{U}$ (the definition is analogous for the AM-adversary) is to distinguish the session key of a KE-session $(i, j, s_{ik}, role)$ with an additional test-session query. This query may be scheduled by the adversary only of a complete session. The adversary is provided with a value sk_{ik} as follows: an unbiased coin is tossed, if the result b equals 0 then sk_{ik} is the real value of the session key, otherwise it is a random value chosen from the same distribution of the session keys produced by the protocol but independent of the value of the real session key. After receiving sk_{ik} the adversary may continue with oth! er queries against the protocol; at the end of its game $\mathscr{U}$ outputs its guess b' of b. The adversary succeeds in its distinguishing attack if (1) the test session is not exposed; (2) the probability that $b' = b$ is significantly larger than $1/2$. More formally:

Definition 1 ([9]) *A* KE *protocol is* SK-secure in the UM *(resp. AM) if for any PPT* UM*-adversary* $\mathscr{U}$ *(resp.* AM*-adversary* $\mathscr{A}$*) the following conditions hold:*

a. if two uncorrupted parties complete matching sessions in a protocol run then, except for a negligible probability, they output the same session key;
b. $\mathscr{U}$ *succeeds (in its test-session distinguishing attack) with probability no more than 1/2 plus a negligible function in the security parameter.*

An important property of KE protocols not captured by the above definition is forward secrecy (FS), i.e., the assurance that once a session key is erased from the memory of the principals that have established the key, it cannot be learned by the adversary even if the principals are subsequently corrupted and (a polynomial number of) the protocol transcripts were observed (note that a passive adversary is subsumed here). Formally, this is captured via the notion of session expiration. A key-exchange protocol is said to be FS-secure if the above definition holds even when the adversary is allowed to corrupt a peer to the test session *after the test-session key expired* at that peer.

1.5.2 Security analysis of protocol SNKE

To prove that protocol SNKE (in both operational modes) is SK-secure in the UM we first show that the protocol of Figure 1.3 which we denote by SNKE-ENC[2]

[2] The values s_A, s_B represent session identifiers which are omitted from the specification of protocol SNKE for simplicity.

(since it is essentially equivalent to protocol SNKE without MACs) is secure in the AM. We then apply well known techniques and results to prove our thesis using the following MAC-authenticator ([9]): principal P_i sends a challenge nonce $N_i \xleftarrow{R} \{0,1\}^\ell$ to P_j and P_i responds by sending message m along with the authentication tag $\mathrm{MAC}_{k_{ij}}(j, N_i, m)$. This authenticator can be proven secure analogously to the signature-based authenticator from [3].

Theorem 1 *Assuming the encryption scheme* ENC *is* (t,ε)-`IND-CPA-secure` *and* `H` *is a random oracle then protocol* SNKE-ENC *is* `SK-secure in the UM`.

The proof of this theorem is in the appendix. We stress that the theorem is valid for both operation modes of protocol SNKE. By using the above MAC-authenticator to compile protocol SNKE-ENC, by virtue of Theorem 6 [9], we obtain that protocol SNKE is secure in the UM.

$A:$	$\mathtt{K} = \mathtt{K}_{AB} = \mathtt{K}_{BA}$
$B:$	$\mathtt{K} = \mathtt{K}_{AB} = \mathtt{K}_{BA}$
$A:$	$r_A \xleftarrow{R} \{0,1\}^\ell$
	$c_A \leftarrow \mathtt{enc}_{\mathtt{K}}(r_A, A)$
$A \rightarrow B$:	s_A, c_A
$B:$	$r_A, A \leftarrow \mathtt{dec}_{\mathtt{K}}(c_A)$
	$r_B \xleftarrow{R} \{0,1\}^\ell$
	$c_B \leftarrow \mathtt{enc}_{\mathtt{K}}(r_B, B)$
	$\chi_B, sk_B \leftarrow \mathtt{H}(s_A, s_B, r_A, r_B, A, B, \mathtt{K})$
	$\mathtt{K}' \leftarrow \mathtt{K} \oplus \chi_B$
$B \rightarrow A$:	s_B, c_B
$A:$	$r_B, B \leftarrow \mathtt{dec}_{\mathtt{K}}(c_B)$
	$\chi_A, sk_A \leftarrow \mathtt{H}(s_A, s_B, r_A, r_B, A, B, \mathtt{K})$
	$\mathtt{K}' \leftarrow \mathtt{K} \oplus \chi_A$

Fig. 1.3 Protocol SNKE-ENC

1.6 Implementation Issues

The 802.15.4 media access control layer offers three basic security services: (a) *message integrity*, (b) *data confidentiality* and (c) *replay protection*. The standard recommends the secure implementation of cryptographic operations and also that mechanisms should be employed to guaranteee the authenticity of stored keying material (which may be shared on a pairwise or group basis).

There are also eight different security suites in the standard which can be applied to the single frame that offer increasing grades of cryptographic protection: (a) no protection; (b) message (frame) integrity with tags of length 32, 64 and 128 bits; (c) encryption only (d) authenticated encryption with tags of length 32, 64 and 128 bits.

The upper protocol layers are responsible for setting up the keys and determining the security level to use.

We estimate the power consumption of a node in the key establishment process; of course, there are minimum energy requirements necessary to preserve an adequate level of security. By inspection of Table 1.1 (column two - *Primitives*) it is clear that protocol SNKE performs better than all the others in terms of the energy requirements involved in the execution of the cryptographic primitives and radio communications.

The practical security of any hash function lies in the appropriate choice of the output size m in order to prevent collisions (which are easier to find than pre-images and 2^{nd} pre-images). A number of computations of size $O(2^{m/2})$ are required to find a collision for an m-bit hash function according to the "birthday attack"; whereas it requires a $O(2^m)$ effort to find either pre-images or 2^{nd} pre-images by the brute-force attack [24]. For this reason we consider the SHA-1 construction a suitable choice for the hash function `H`. According to [27] the energy cost of computing SHA-1 ($m = 160$) is approximately 0.76 μJ per byte.

The 802.15.4 standard indicates the AES block cipher [12] with 128-bit keys as the designated encryption scheme (consider also the construction of [25]). In recent work it was reported that the energy cost of performing 128-bit AES encryption/decryption operations on a Rockwell WINS node (equipped with a 133 MHz StrongARM processor) is less than 0.1 mJ [21]. For greater efficiency one-time pads may be used (significant energy savings can be achieved on RFD devices); in this case the shared key should be twice the standard length (i.e. 256 bits) with each principal using a different half (128 bits) to encrypt the exchanged nonces r_A, r_B (this method can be applied when the protocol is run in key-renewal mode).

The HMAC-SHA1 [2] construction is the recommended choice for the MAC since it enjoys the SUF-CMA security property and requires a modest computational load (the underlying hash function (SHA) is applied twice with some additional padding); furthermore, it allows efficient reuse of the chip area dedicated to the hash function. Alternatively, the UMAC construction [7] which is based on a universal hash function family may offer greater efficiency. Recently, Kaps *et al*. [20] have improved the universal hash function family used in UMAC (NH) and have obtained a new construction (WH) with significant energy savings (it is shown that WH outperforms NH with respect to dynamic and leakage power, circuit area and delay at 100 MHz).

References

1. S. Basagni, K. Herrin, E. Rosti, and D. Bruschi. Secure Pebblenets. *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pages 156–163, 2001.
2. M. Bellare, R. Canetti, and H. Krawczyk. Keying hash functions for message authentication. *Advances in Cryptology - CRYPT0 1996*, LNCS 1109:1–15, 1996.

3. M. Bellare, R. Canetti, and H. Krawczyk. A modular approach to the design and analysis of authentication and key exchange protocols. *In 30th Symposium on Theory of Computing*, pages 419–428, 1998.
4. M. Bellare and C. Nampempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. *Advances in Cryptology - Asiacrypt 2000*, LNCS 1976, 2000.
5. M. Bellare and P. Rogaway. Entity authentication and key distribution. *In Proceedings of CRYPTO 1993*, LNCS 773:232–249, 1993.
6. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. *In 1st Conference on Computer and Communications Security*, pages 62–73, 1993.
7. J. Black, S. Halevi, H. Krawczyk, T. Krovetz, and P. Rogaway. UMAC: Fast and secure message authentication. *Advances in Cryptology - CRYPTO '99*, LNCS 1666:216–233, 1999.
8. C. Blundo, A. D. Santis, A. Herzberg, S. Kutten, S. Vaccaro, and M. Yung. Perfectly-secure key distribution for dynamic conferences. *Advances in Cryptology - CRYPTO 1992*, LNCS 740:471–486, 1993.
9. R. Canetti and H. Krawczyk. Analysis of key exchange protocols and their use for building secure channels. *Advances in Cryptology-EUROCRYPT 2001*, LNCS 2045:453–474, 2001.
10. R. Canetti and H. Krawczyk. Universally composable notions of key exchange and secure channels. *Advances in Cryptology-EUROCRYPT 2002*, LNCS 2332:337–351, 2002.
11. H. Chan, A. Perrig, and D. Song. Random key predistribution schemes for sensor networks. *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 197–213, 2003.
12. J. Daemen and V. Rijmen. *The Design of Rijndael*. Springer-Verlag, 2002.
13. D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography. *Proceedings of the 23rd Annual Symposium on the Theory of Computing*, 1991.
14. L. Eschenauer and V. Gligor. A key management scheme for distributed sensor networks. *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 41–47, 2002.
15. S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computing and System Sciences*, 28:270–299, 1984.
16. IEEE-802.15.4-2006. Standard for information technology-telecommunications and inofrmation exchange between systems-local and metropoolitan area networks-specific requirements-part 15.4: Wireless medium access control (mac) and physical layer (phy) specifications for low rate wireless personal area networks (wpann). Institute of Electrical and Electronics Engineers, 2006.
17. ISO/IEC-11770-2. Information technology-security techniques-key management-part 2: Mechanisms using symmetric techniques. International Standards Organization, 1996.
18. K. Jamshaid and L. Schwiebert. SEKEN (Secure and Efficient Key Exchange for Sensor Networks. *Proceedings of IEEE Int.l Conference on Performance, Computing and Communications*, pages 415–422, 2004.
19. P. Janson and G. Tsudik. Secure and Minimal Protocols for Authenticated Key Distribution. *Computer communications*, 18(9):645–653, 1995.
20. J. Kaps, K. Yuksel, and B. Sunar. Energy Scalable Universal Hashing. *IEEE Transactions on Computers*, pages 1484–1495, 2005.
21. H. Krawczyk. The energy cost of cryptographic key establishment in wireless sensor networks. *http://eprint.iacr.org/2007/003*, 2007.
22. D. Liu, P. Ning, and R. Li. Establishing pairwise keys in distributed sensor networks. *Proceedings of the 10th ACM conference on Computer and communications security*, pages 52–61, 2003.
23. S. Mauw, I. Van Vessen, and B. Bos. Forward Secure Communication in Wireless Sensor Networks. *3rd International Conference on Security in Pervasive Computing*, 2006.
24. A. Menezes, P. V. Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.
25. M. Ohkubo, K. Suzuki, and S. Kinoshita. Cryptographic Approach to "Privacy-Friendly" Tags. *In RFID Privacy Workshop*, 2003.

26. A. Perrig, R. Szewczyk, V. Wen, D. E. Culler, and J. Tygar. SPINS: Security protocols for sensor networks. *In Mobile Computing and Networking*, 2001.
27. N. Potlapally, S. Ravi, A. Raghunathan, and N. Jha. Analyzing the energy consumption of security protocols. *Proceedings of Int.l Symposium on Low power electronics and Design*, pages 30–35, 2003.
28. M. Satyanarayanan. Integrating Security in a Large Distributed System. *ACM Transactions on Computer Systems*, 7(3):247–280, 1989.
29. V. Shoup. On Formal Models for Secure Key Exchange. Technical Report RZ 3120, IBM Research, 1999.

1.7 Appendix - Proof of theorem 1

The first condition of Definition 1 is easily verified in the AM (recall that AM-adversaries cannot modify or inject messages unless the sender is corrupted or the messages belong to an exposed session).

We now prove condition two of Definition 1. Recall that an encryption scheme is (t,ε)-IND-CPA-secure if the advantage of the adversary (running in time at most t) at the end of the IND-CPA-game (which captures the indistinguishability of encryptions under chosen plaintext attacks) is less than ε. The IND-CPA-game played by $\mathscr{B} = (\mathscr{B}_1, \mathscr{B}_2)$ (against challenger algorithm $\mathscr{C}$) is outlined below:

1. (Setup): $\mathscr{C}$ generates a shared key $\mathtt{K} \stackrel{R}{\leftarrow} \mathtt{key}(1^\ell)$ and gives 1^ℓ to $\mathscr{B}_1$;
2. (Non-adaptive queries): $\mathscr{B}_1$ asks the encryption oracle queries of its own choice;
3. (Challenge ciphertext): $\mathscr{B}_1$ chooses plaintexts m_0, m_1 (with $|m_0| = |m_1|$) and gives them to the challenger, who selects a bit b and returns the encryption of m_b under $\mathtt{K}$ (i.e. ciphertext c). $\mathscr{B}_1$ hands over c and state information to $\mathscr{B}_2$;
4. (Adaptive queries): $\mathscr{B}_2$ may issue additional queries to the encryption oracle;
5. (Output): $\mathscr{B}_2$ stops and outputs a bit b_{CPA} as its guess for b: $\mathscr{B}$ wins if $b_{CPA} = b$.

The advantage of the adversary is calculated as the difference between the probability that $\mathscr{B}$ outputs the correct value of bit b and a random guess (with probability $1/2$). Consider adversary $\mathscr{A}$ attacking the protocol in the AM. Observe that under the hypothesis that $\mathtt{H}$ is a random oracle the only one way $\mathscr{A}$ can obtain significant information (and therefore win the SK-game) about a particular session key $sk^s_{ij} = \mathtt{H}(s_i, s_j, r_i, r_j, i, j, \mathtt{K}_{ij})$ is by querying the oracle at the point $(s_i, s_j, r_i, r_j, i, j, \mathtt{K}_{ij})$. Suppose that the challenger $\mathscr{C}$ starts (the setup phase) by generating key $\mathtt{K}^* \stackrel{R}{\leftarrow} \mathtt{key}(1^\ell)$. Algorithm $\mathscr{B}_1$ is given in input 1^ℓ and access to the encryption oracle under key $\mathtt{K}^*$. At some stage of its game $\mathscr{B}_1$ chooses at random two principals P_{i^*}, P_{j^*} among all the n principals, selects $! \stackrel{R}{\leftarrow} r_i^{(0)}, r_i^{(1)} \{0,1\}^{\ell_1}$ and outputs $m_0 = (r_i^{(0)}, j^*), m_1 = (r_i^{(1)}, j^*)$. The challenger $\mathscr{C}$ chooses a random bit b and computes $c = \mathtt{enc}_{\mathtt{K}^*}(m_b)$. Algorithm $\mathscr{B}_2$ is given in input $c, 1^\ell$ and state information (which contains the values $r_i^{(0)}, r_i^{(1)}, i^*, j^*$). $\mathscr{B}_2$ generates the keying material (pairwise keys) $\mathtt{K}_{ij} \leftarrow \mathtt{key}(1^\ell)$ for any two parties i, j except for P_{i^*}, P_{j^*}; their shared key $\mathtt{K}_{i^*j^*}$ is assumed to be equal to $\mathtt{K}^*$ (of course, $\mathscr{B}_2$ does not know the value of $\mathtt{K}^*$). Note that

P_{i^*} and P_{j^*} will eventually share pairwise keys with other principals $P_k, k \neq i^*, j^*$. $\mathscr{B}_2$ also chooses at random a session id s_i^* among those where P_{i^*} is the initiator and P_{j^*} the responder (i.e. the chosen session is (i^*, j^*, s_i^*)—we omit the indi! cation of the role and the subscript counting the number of se! ssions f or simplicity). Algorithm $\mathscr{B}_2$ tries to guess b in its IND-CPA-game while running $\mathscr{A}$ as a subroutine. In particular, $\mathscr{B}_2$ must simulate a virtual SK-game played by $\mathscr{A}$ in such a way that $\mathscr{A}$'s view is indistinguishable from a real SK-game. All sessions scheduled by $\mathscr{A}$ are simulated by $\mathscr{B}_2$ according to the specification of the protocol. In particular, for sessions $s_i \neq s_i^*$ involving two principals P_i, P_j with $i \neq i^*$ or $j \neq j^*$ it outputs the transcript $\mathtt{enc}_{\mathtt{K}_{ij}}(r_i, j) \| \mathtt{enc}_{\mathtt{K}_{ji}}(r_j, i)$[3] for random r_i, r_j and sets $sk_{ji}^s = sk_{ij}^s \stackrel{R}{\leftarrow} \{0,1\}^{|\mathtt{H}(\cdot)|}$. When (eventually) session s_i^* is invoked, $\mathscr{B}_2$ submits to $\mathscr{A}$ the challenge c! iphertext $y = \mathtt{enc}_{\mathtt{K}}(m_b, j^*)$ as the message sent by the initiator P_{i^*}. In addition, $\mathscr{B}_2$ chooses $r_j^* \stackrel{R}{\leftarrow} \{0,1\}^{\ell_1}$, obtains the value $z = \mathtt{enc}_{\mathtt{K}^*}(r_j^*, i^*)$ from its encryption oracle, submits z to $\mathscr{A}$ as P_{j^*}'s response and sets $sk_{j^*i^*}^{s^*} = sk_{i^*j^*}^{s^*} \stackrel{R}{\leftarrow} \{0,1\}^{|\mathtt{H}(\cdot)|}$. For queries of the random oracle $\mathtt{H}$ at the point $(s_i, s_j, r_i, r_j, i, j, \mathtt{K}_{ij})$ submitted by $\mathscr{A}$ the answer is $v \stackrel{R}{\leftarrow} \{0,1\}^{|\mathtt{H}(\cdot)|}$. If $i = i^*$ and $j = j^*$ the record $\langle \widehat{s_i}, (s_i, s_j, r_i, r_j, i, j, \mathtt{K}_{ij}) \rangle$ is stored in the list L. When $\mathscr{A}$ invokes `session-state reveal` or `party corruption` queries that do not involve both P_{i^*}, P_{j^*} these are easily answered by $\mathscr{B}_2$ since it knows the required information (see above). If at any point $\mathscr{A}$ issues a `session-state reveal` of session s^*, a `party corruption` query of either P_{i^*} or P_{j^*} or chooses a test-session different than s^* then $\mathscr{B}_2$ aborts. At the end of its game (and when $\mathscr{A}$ has stopped) $\mathscr{B}_2$ outputs the element from the list L having $\widehat{s_i} = s_i^*$ (if it exists).

We analyze the success probability of $\mathscr{B}_2$. If there exists an entry in the list L having $\widehat{s_i} = s_i^*$ we say that event $\mathtt{qry}^*$ has occurred. We have

$$|\Pr_{\mathscr{B}_2}[b_{CPA}{=}b] - \frac{1}{2}| = |\Pr_{\mathscr{A}}[b_{SK}{=}b] - \frac{1}{2}| \leq \Pr_{\mathscr{A}}[\mathtt{qry}^*] + \frac{1}{2}\Pr_{\mathscr{A}}[\overline{\mathtt{qry}^*}] \leq \frac{1}{2}\Pr_{\mathscr{A}}[\mathtt{qry}^*]$$

where we used the fact that $\Pr_{\mathscr{A}}[b_{SK} = b | \overline{\mathtt{qry}^*}] = \frac{1}{2}$ The probability that $\mathscr{B}_2$ outputs the correct answer is (at least) $\Pr_{\mathscr{A}}[\mathtt{qry}^*]/q_s$ where q_s is the polynomial bound on the number of sessions run in the SK-game. Therefore, since the running time of $\mathscr{B}_2$ is essentially equal to the running time of $\mathscr{A}$ we have

$$|\Pr_{\mathscr{B}_2}[b_{CPA}{=}b] - \frac{1}{2}| \leq q_s \varepsilon$$

and this proves the theorem.

[3] The actual order of the messages in the transcript depends on the role (initiator or responder) played by each principal in the run of the protocol.