Secure Key Management Architecture Against Sensor-node Fabrication Attacks

Jeffrey Dwoskin Dahai Xu Jianwei Huang Mung Chiang Ruby Lee
{jdwoskin, dahaixu, jianweih, chiangm, rblee}@princeton.edu
Department of Electrical Engineering, Princeton University, NJ 08544, USA

***Abstract*— In lightweight mobile ad hoc networks, both probabilistic and deterministic key management schemes are fragile to node fabrication attacks. Our simulation results show that the Successful Attack Probability (SAP) can be as high as 42.6% with the fabrication of only 6 copies from captured nodes comprising only 3% of all nodes. In this paper, we propose two low-cost secure-architecture-based techniques to improve the security against such node fabrication attacks. Our new architectures, specifically targeted at the sensor-node platform, protect long-term keys using a root of trust embedded in the hardware System-on-a-Chip (SoC). This prevents an adversary from extracting these protected long-term keys from a captured node to fabricate new nodes. The extensive simulation results show that the proposed architecture can significantly decrease the SAP and increase the security level of key management for mobile ad hoc networks.**

I. Introduction

Security is one of the critical requirements for the deployment of lightweight wireless ad hoc networks, which consist of nodes such as low-cost sensors or embedded devices. These networks are ideal for applications like environmental surveillance and emergency response. The nodes in such networks are typically highly distributed without a centralized controller, and each node has very limited computation and energy resources. It is thus a challenging task to maintain a high security level in such networks.

In this paper, we consider the network security issues related to key management, which is the cornerstone of secure communication. For lightweight ad hoc networks that are typically deployed in hostile environments, adversaries can easily capture the nodes and try to extract the keys from them, leading to severe security threats to the network. We propose a secure-architecture-based technique to protect the keys in the captured nodes, thus making various key management schemes robust even in the face of node capture.

A. Key Management in Lightweight Ad Hoc Networks

We first introduce some background on key management in lightweight ad hoc networks, including different classes of key management schemes, several performance metrics, and two network scenarios upon which we will focus our analysis.

There are two main classes of key management schemes in lightweight ad hoc networks: *deterministic* (e.g., [1]–[4]) and *probabilistic* (e.g., [5]–[12]). A typical deterministic algorithm preloads each node with a single common (long-term) key, while in a probabilistic approach, the long-term keys in each node's key ring are randomly chosen from a large key pool [5]. Once deployed in a mobile network, long-term keys are used for mutual authentication between pairs of neighboring nodes (i.e., nodes within each other's communication range) to establish pairwise keys for future communication. When neighboring nodes do not share a long-term key, they will relay through another node that is within their communication range to set up a pairwise key. Pairwise key establishment and future communication can be eavesdropped by other nodes within the communication range, provided those nodes also have the corresponding long-term or pairwise key.

There are several metrics for evaluating various key management schemes. One metric is *link connectivity*, which is defined as the probability of being able to set up pairwise communication directly between two neighboring nodes that are within communication range. Obviously, the link connectivity of the single-common-key deterministic scheme is 100%, while that of a probabilistic approach is the probability of sharing at least one long-term key between two neighboring nodes. In general, probabilistic approaches end up with a larger key pool, many more keys per node, and poorer link connectivity than the deterministic approaches. More related references can be found in [13], [14].

Another important metric is *Successful Attack Probability* (SAP) for node-capture attacks [14]. An attack on a pairwise link between two authorized nodes is successful if a compromised node can intercept and decipher the information transmitted through that link. SAP will be dependent on network scenarios, which can be categorized as *static networks* or *mobile networks*.

In a static network where sensors do not move after deployment, both the deterministic approach (e.g. single common key scheme [1]) and the probabilistic approach (e.g. EG scheme [5]) can provide perfect resilience if nodes are captured after all pairwise links have been established. By exchanging messages encrypted with the initial common key (deterministic) or a shared key (probabilistic), two neighboring nodes can generate a random pairwise key, which is known only to them. The pairwise keys cannot be deduced by a captured node even if the initial long-term keys are later disclosed. Thus the SAP is close to 0. To ensure further security, the initial long-term keys can be deleted from memory permanently after deployment [1].

In contrast, in a mobile network, nodes are constantly on the move and often need to establish new links. Examples include networks of buoys floating freely on the ocean to gather environmental data [15], and networks of sensors moving around in an unknown environment to form reasonable coverage [16].

In a mobile network, the single common deterministic key scheme could lead to an SAP as high as 100%, if the common key is obtained by an adversary before any link is established. However, the EG probabilistic scheme is also quite vulnerable as shown in [14]. The value of SAP for the EG scheme can be as high as 60% if the adversary can fully utilize the keys obtained from several compromised nodes (i.e., a node fabrication attack as explained in Section II). The reason is that in a probabilistic approach, to increase link connectivity, key-relay is required. By intercepting the key information that is being relayed, a compromised node can figure out the key which the two authorized nodes will use for future mutual communication. This man-in-the-middle attack opportunity can significantly increase the value of SAP for a probabilistic approach, since there is a high chance of using a relay for link establishment in a mobile environment. By combining the keys from multiple captured nodes to *fabricate* new nodes in the network, the adversary increases his likelihood of being used as a relay in this scheme and succeeding in an attack.

B. Contributions and Structure of This Paper

Hence, preventing node fabrication from long-term keys stored in captured nodes is critical to improve the security levels in both deterministic and probabilistic key management schemes. In this paper, we propose two secure-hardware-based techniques, specifically targeted to the sensor-node platform, that protect long-term keys for both deterministic and probabilistic key management schemes for mobile networks. This ensures that protected secrets cannot be extracted from a captured node. This is the first step towards building a comprehensive low-cost secure-hardware design for sensor nodes.

The contributions of this paper are:

- We analyze various security attacks on the secure key management in mobile lightweight ad hoc networks.
- *Sensor-mode SP*: Two new secure architectures that defend against node fabrication attacks for sensors with very limited or moderate capabilities, enhancing the security of mobile sensor network key management.
- *SAP reduction*: Extensive simulation results showing how node fabrication attacks increase the Successive Attack Probability (SAP) and how our new architecture reduces SAP to an insignificant level.

The rest of the paper is organized as follows. In Sec. II we describe the scenario under which we protect sensor nodes and the threat models for node-capture attacks. In Sec. III, we propose processor architecture based techniques for securing secret keys and critical software on a node. In Sec. IV we analyze the security of the proposed architecture under several specific attacks. In Sec. V we provide simulation results showing the reduced SAP with our architecture. We conclude in Sec. VI.

II. Sensor Network Scenario and Threat Model

For our analysis, we consider a probabilistic key management scheme in a mobile network. We assume that sensor nodes are initialized at a secure depot by an authority and that attacks are not possible during this process. As it receives devices from a manufacturer, the authority is responsible for initializing the software and the security mechanisms of the node. The authority serves as the primary trusted party that generates and installs long-term keys and authorizes devices for the sensor network.

In a *node capture* attack, an adversary compromises one or more sensor nodes and extracts their long-term keys after deployment. The adversary then tries to obtain the pairwise keys used by other nodes so that it can later monitor their links. If it shares the long-term key used by two nodes for key-establishment, it can observe the corresponding negotiated pairwise key between those two nodes. Alternatively, if the two nodes do not share a long-term key, they may choose to relay through the compromised node which can save the resulting pairwise key. In both cases, the adversary is limited to attacking nodes within its communication range.

In a *node fabrication* attack, the adversary uses the extracted keys to fabricate new nodes. One method is to simply clone the compromised node, using additional sensor devices loaded with an exact copy of the keys from the compromised node. Another method pools the keys from multiple compromised nodes; it then either makes fabricated nodes with unique subsets of the combined key pool [14] or fabricates *super-nodes* using all of the extracted keys in each copy. Cloning and node fabrication allow the adversary to significantly increase his SAP compared to a node capture attack.

The crucial observation is that the attacks succeed because the long-term keys are not protected when a node is captured. We assume an adversary with physical access to the device, so software protections are easily bypassed. The keys are accessible to the software on the node which the adversary can exploit or replace entirely. He might also read the keys directly from a flash memory chip or other permanent storage when the device is offline.

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

III. Secret-Protected Processor Architecture

Our solution is to provide a Secret Protected (SP) architecture which minimizes the trusted computing base (TCB) of hardware and software that has to be fully correct, verified and trusted. Our TCB comprises some SP hardware features (described below) and a small Trusted Software Module (TSM), that does the key management.

A. Sensor-mode SP

To prevent node fabrication attacks, we must tackle the problem of key extraction from a captured node. We first

present the simplest solution, which we call Reduced Sensor-mode SP, suitable for the simplest sensors. We then extend the solution for slightly more capable sensors. Our work is inspired by the SP architecture proposed for general-purpose microprocessors [17], [18], but stripped to the bare minimum for sensors with very constrained computing and storage resources.

B. Reduced Hardware Architecture

The simplest version of our architecture, Reduced Sensor-mode SP, is shown in Fig. 1. It only requires one new register — the Device Key and a bit to indicate protected mode. Additionally, a Trusted Software Module (TSM) is stored in the on-chip instruction EEPROM and the long-term keys for the probabilistic key management scheme are stored in the on-chip data EEPROM. Also, a portion of the main memory of the node is reserved for the TSM Scratchpad Memory.

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

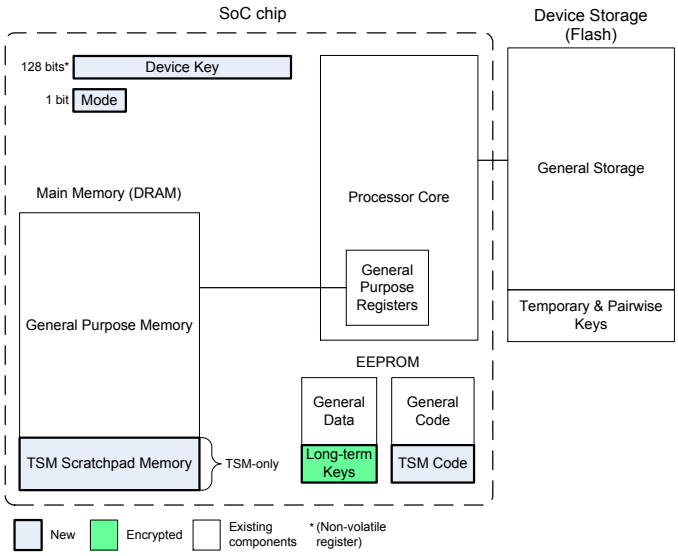

Fig. 1. Reduced Sensor-mode SP

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

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

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

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

Initialization of a new device takes place at the authority's depot. First it must generate a new random device key. Long-term keys and other secrets are encrypted with it are then stored along with the TSM code on the on-chip EEPROM. Next it uses the *DeviceKey_Set* instruction to store the device key. Finally, any other unprotected software and data can be copied to the flash storage.

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

C. Expanded Sensor-mode SP Architecture

The Reduced Sensor-mode SP architecture is ideal for the smallest sensor nodes which use minimal software and have very limited resources. In slightly larger lightweight sensor nodes, the software will be more complex. The additional applications that run on this sensor combined with the TSM and long-term keys will be too large to store on-chip. This greater flexibility in the sensor also requires additional support for security. Hence, we propose the Expanded Sensor-mode SP architecture shown in Fig. 2.

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

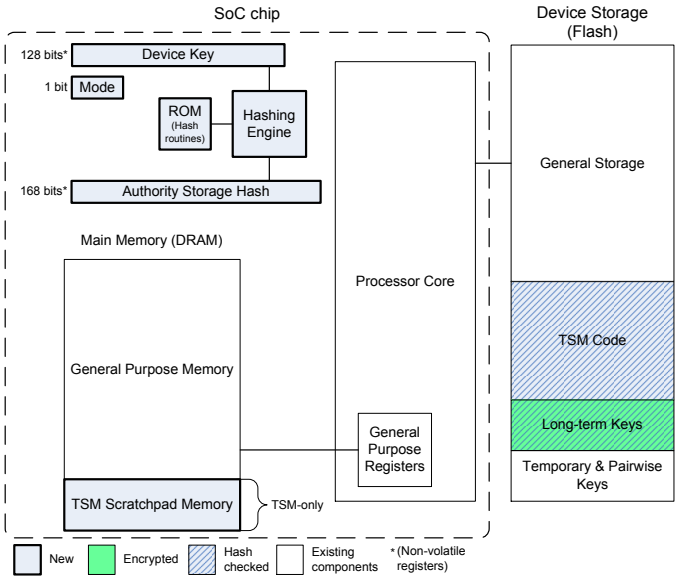

Fig. 2. Expanded Sensor-mode SP

TABLE I
NEW SENSOR-MODE SP INSTRUCTIONS

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

The ASH register contains a hash over the entire memory region of the TSM code and long-term keys. It is stored by the authority during initialization and is rechecked by the processor each time the TSM is called. The checking code is stored in the on-chip ROM and is fixed and therefore safe from modification; it uses the hardware hashing engine to compute the hash over the code of the TSM and the encrypted keys. When *Begin_TSM* is called, the processor disables interrupts and jumps to the TSM-checking routine. If the hash check succeeds, the protected mode bit is set, and execution jumps to the newly-verified TSM code. If the check fails, an exception is triggered. The ASH register is set using the *ASH_Set* instruction, which first clears the device key, ensuring that the TSM can't be replaced and still have access to the protected keys.

IV. SECURITY ANALYSIS

A. Attacks on Protected Keys

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

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

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

B. Attacks on Changing the TSM or the Device Key

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

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

Finally, if the attacker tries to modify the ASH register to match the new TSM code, the device key will be cleared,

irrevocably cutting off his access to all of the keys that were encrypted with that device key. Clearing or setting the device key also clears the TSM scratchpad memory, so any intermediate data stored there that might have leaked secrets is also unavailable to the new TSM.

V. Simulation Results

In this section, we show the security performance of the proposed secret-protected processor (SP) architecture for lightweight ad hoc networks based on numerical results obtained through a C++ simulator. We focus on the performance evaluation of the probabilistic key predistribution approach (the EG scheme) since the deterministic approach (e.g., single common key) is a special case of the probabilistic approach. In the EG scheme, each node is equipped with k keys randomly chosen from a key pool of size m.

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

If every node is equipped with the Sensor-mode SP architecture, the adversary can only launch the *node capture attack*, where the adversary utilizes the captured nodes themselves to intercept pairwise-key establishment. Without the SP architecture, the adversary can further launch the *node fabrication attack* where he can turn the captured nodes into super-nodes by loading each of them with all of the keys from all captured nodes. Each super-node can mimic multiple nodes. A straightforward method to achieve this is to let each super-node stay at its original location but announce the existence of all the captured nodes. Without the help of Global Positioning System (GPS), it is difficult for the authorized nodes to identify the duplication of nodes within the network. The adversary can even make more copies of the super-nodes and deploy them into the network to eavesdrop additional communication.

Fig. 3 shows the network-wide SAP under different numbers of captured nodes, for different kinds of attacks. "SP" means launching only the node capture attack with the SP architecture. "0 copies" means changing captured nodes into super-nodes (i.e., node fabrication attack) due to the lack of the SP architecture. "x copies" means making x extra copies of these super-nodes. Note that, without SP, the effect of node capture can be serious. When only 3% of the nodes are captured, the SAP for the network will be 9.7% even with 0 copies, and it becomes 42.6% if the adversary makes 6 copies of the captured nodes to cover more area. Whereas the SAP for the nodes with SP is only 2.1%. Using SP can reduce SAP by roughly an order of magnitude. Therefore, the benefit of using SP is significant in terms of alleviating the attack of node fabrication.

Fig. 4 shows the network-wide SAP under different sizes of the preloaded key ring, k, for different attack models,

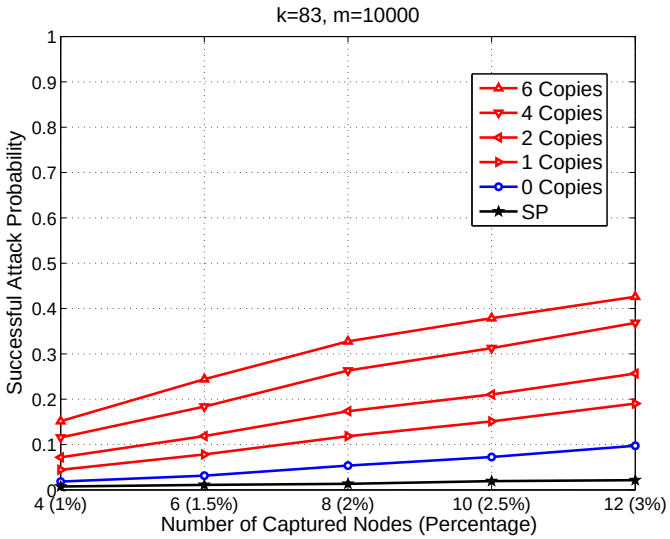

Fig. 3. Network-wide successful attack probability under different numbers of captured nodes for different attack models

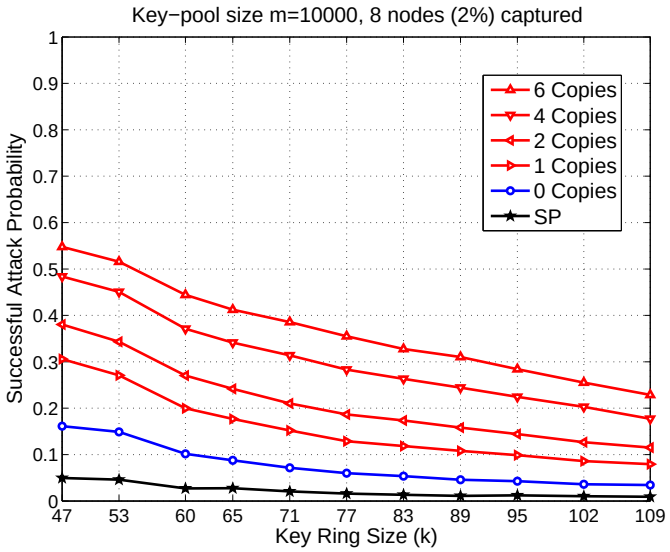

Fig. 4. Network-wide successful attack probability for different key ring size (k) with different attack models

assuming 2% of nodes have been captured. An increasing value of k has two effects on the network. First, the link connectivity increases; this reduces the probability of two neighboring nodes establishing a pairwise link through a relay node, and thus can improve the network security. Second, each node captured by the adversary contains more keys, which will increase the chance of intercepting the communications on other pairwise links. This is detrimental to the network security. Fig. 4 shows that the advantage of the first effect dominates and the overall SAP decreases with an increasing value of k. Notice that the SP architecture offers significant advantages over the other schemes for all values of k.

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

VI. Concluding Remarks

In this paper, we propose two low-cost hardware-based architectures to enhance the security of key management schemes against the attack of sensor node fabrication for a lightweight mobile ad hoc network. The simulation results show that the proposed architectures can decrease the Successful Attack Probability on pairwise links by an order of magnitude. Future work includes investigating the advantage of hardware-based techniques for end-to-end security and public-private-key-based key management schemes.

References

[1] S. Zhu, S. Setia, and S. Jajodia, "LEAP: efficient security mechanisms for large-scale distributed sensor networks," in *CCS'03: ACM conference on Computer and communications security*, New York, NY, 2003, pp. 62–72.

[2] J. Lee and D. Stinson, "Deterministic key predistribution schemes for distributed sensor networks," *Selected Areas in Cryptography*, 2004.

[3] S. Camtepe and B. Yener, "Combinatorial design of key distribution mechanisms for wireless sensor networks," *European Symposium On Research in Computer Security (ESORICS'04)*, 2004.

[4] S. Zhu, S. Setia, S. Jajodia, and P. Ning, "An interleaved hop-by-hop authentication scheme for filtering of injected false data in sensor networks," *Security and Privacy, 2004. Proceedings. 2004 IEEE Symposium on*, pp. 259–271, 2004.

[5] L. Eschenauer and V. D. Gligor, "A key-management scheme for distributed sensor networks," in *CCS'02: ACM conference on Computer and communications security*, New York, NY, 2002, pp. 41–47.

[6] H. Chan, A. Perrig, and D. Song, "Random key predistribution schemes for sensor networks," in *IEEE Symposium on Security and Privacy*, 2003.

[7] R. D. Pietro, L. V. Mancini, and A. Mei, "Random key-assignment for secure wireless sensor networks," in *SASN'03: ACM workshop on Security of ad hoc and sensor networks*, New York, NY, 2003, pp. 62–71.

[8] W. Du et al., "A key management scheme for wireless sensor networks using deployment knowledge," in *INFOCOM'04, Hong Kong*, Mar. 2004.

[9] J. Hwang and Y. Kim, "Revisiting random key pre-distribution schemes for wireless sensor networks," *ACM workshop on Security of ad hoc and sensor networks*, pp. 43–52, 2004.

[10] S. Zhu, S. Xu, S. Setia, and S. Jajodia, "Establishing pairwise keys for secure communication in ad hoc networks: A probabilistic approach," in *ICNP'03*, 2003.

[11] W. Du, J. Deng, Y. S. Han, and P. K. Varshney, "A pairwise key predistribution scheme for wireless sensor networks," in *CCS'03: ACM conference on Computer and communications security*, New York, NY, 2003, pp. 42–51.

[12] D. Liu and P. Ning, "Location-based pairwise key establishment for static sensor networks," in *1st ACM Workshop on Security of Ad Hoc and Sensor Networks*, 2003.

[13] S. A. Çamtepe and B. Yener, "Key distribution mechanisms for wireless sensor networks: a survey," Rensselaer Polytechnic Institute, Computer Science Department, Tech. Rep. TR-05-07, Mar. 2005, available at http://www.cs.rpi.edu/research/pdf/05-07.pdf.

[14] D. Xu, J. Huang, J. Dwoskin, M. Chiang, and R. Lee, "Re-examining probabilistic versus deterministic key management," in *ISIT'07, Nice, France, http://www.princeton.edu/∼dahaixu/pub/key/key.pdf*.

[15] S. Seys and B. Preneel, "The wandering nodes: Key management for lower-power mobile ad hoc netowrks," in *IEEE International Conference on Distributed Computing Systems Workshops, ICDCSW'05*, 2005.

[16] A. Howard, M. J. Mataric, and G. S. Sukhatme, "Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem," in *Distributed Autonomous Robotic Systems*, 2002.

[17] R. Lee et al., "Architecture for protecting critical secrets in microprocessors," in *International Symposium on Computer Architecture (ISCA 2005)*, June 2005, pp. 2–13.

[18] J. Dwoskin and R. B. Lee, "Processor architecture for remote, transient, policy-controlled secrets," Princeton University Department of Electrical Engineering Technical Report CE-L2006-007, Tech. Rep., Nov. 2006.