Recognition in a Low-Power Environment

Jonathan Hammell* André Weimerskirch† Joao Girao‡ Dirk Westhoff‡

* Faculty of Mathematics, University of Waterloo, Canada, jfhammell@uwaterloo.ca
† escrypt GmbH, 44801 Bochum, Germany, aweimerskirch@escrypt.com
‡ Network Laboratories, NEC Europe Ltd., 69115 Heidelberg, Germany,
{joao.girao,dirk.westhoff}@ccrle.nec.de

***Abstract*— This paper formally defines recognition as a new security principle closely related to authentication. Low-power environments with no pre-deployment information require the less authoritative security of recognition. We give general properties of recognition protocols based on the method of key disclosure. We examine previously proposed low-power protocols according to the environment and security model presented. Finally, we give measurements from an implementation of a recognition protocol called Zero Common-Knowledge and discuss how well this proof-of-concept satisfies the properties of the environment.**

***Index Terms*— Ad hoc Networks, Security, Authentication, Identification, Low-Power**

I. Introduction

Low-power environments, such as sensor networks, present a difficulty in performing traditional security protocols. Sensor motes, as the devices involved in a sensor network are called, are intended to be small and cheap. The continued desire to make these motes smaller offsets the technological advancements of increasing computational power in a smaller area. Thus, the extremely low computational power of such devices has severe performance decreases for asymmetric cryptography and exponential mathematics. Moreover, the dynamic network topology and self-organizing properties of this environment prevent any sort of pre-deployment information like shared secrets or network addresses of trusted third parties.

In this paper, we formally define a new security classification called entity recognition whereby after the first communications with an entity, future recognitions assure that one is communicating with the same entity. This is closely related to the commonly studied term entity authentication. Similarly, we define message recognition along the same thread but without the guarantee that participants are actively involved. Using these weaker security requirements allows one to draw security conclusions in low-power environments where the stronger requirements are unnecessary and often even impossible to achieve.

Under these new security classifications, we examine the satisfiability of some previously proposed low-power protocols. Specifically, these include the Resurrecting Duckling [1], TESLA [2], μTESLA [3], Guy Fawkes [4], Remote User Authentication [5], and Zero Common-Knowledge (ZCK) [6].

This research was conducted while A. Weimerskirch was a member and J. Hammell was a visitor in the Communication Security Group at Ruhr-University Bochum.

By listing common properties of low-power application environments, we attempt to show that ZCK[1] is an improvement in this limited environment. Furthermore, we relate the theory to measurements from an actual implementation of ZCK to prove that our hypotheses are correct.

We present the new definitions of recognition, target environment properties and motivation in Section II. Previous work is presented in Section III. Next, we outline two key disclosure techniques important to recognition protocols, and some consequences of the target environment in Section IV. We examine common low-power protocols in Section V. Section VI describes our proof-of-concept, our implementation of ZCK. Results and measurements are provided in Section VII, concluding by discussing issues and observations based on the timings. Finally, conclusions and a summary of key results are given in Section VIII.

II. Definitions and Preliminaries

A. Entity Recognition

In terms of classical cryptography, the following definition is widely accepted [7]:

Definition 1: Entity authentication is the process whereby one party is assured (through acquisition of corroborative evidence) of the identity of a second party involved in a protocol, and that the second has actually participated (*i.e.* is active at, or immediately prior to, the time the evidence is acquired).

We define a *low-power* environment as a situation where one or more of the devices involved in a communication protocol have a list of constraints that include the following:

Axiom 1. Low computational power
Axiom 2. Low code space
Axiom 3. Low communication bandwith
Axiom 4. Low energy resources

In these low-power environments, entity authentication might be too costly and in many scenarios, it is unnecessary. We formally define the following term to refer to a new classification of protocols:

Definition 2: Entity recognition is the process whereby one party after having initially met, can be assured in future contacts that it is communicating with the same second party involved in a protocol, and that the second has actually

[1] ZCK was designed by some of the authors of this paper.

participated (*i.e.* is active at, or immediately prior to, the time the evidence is acquired).

Note that entity recognition is a weaker form of entity authentication. Hence, a protocol which performs entity authentication has, implicitly, performed entity recognition. The reverse of this is not true.

Consider a Diffie-Hellman key agreement without exchanging certificates but only the public keys in order to agree on a symmetric key later on used to authenticate messages by means of a MAC. Informally spoken, such a scheme provides entity recognition but no entity authentication since there is no identity binding to the key. The above example shows the properties and limitations of an entity recognition scheme.

Message authentication is closely related to entity authentication and recognition. Definition 3 is also provided from [7].

Definition 3: Message authentication provides data origin authentication and data integrity with respect to the original message source, but no uniqueness and timeliness guarantees.

The difference between message authentication (Def. 3) and entity authentication (Def. 1) is the lack of a timeliness guarantee in the former with respect to when a message was created [7].

Similar to entity recognition, the following defines message recognition from message authentication.

Definition 4: Message recognition provides data integrity with respect to the original message source and assures the data origin is the same in future contacts, but no uniqueness and timeliness guarantees.

Message recognition has a similar relation to entity recognition as message authentication has to entity authentication with respect to timeliness as described above.

B. Adversarial Model

We assume the Dolev-Yao threat model. In short, we assume that an adversary, has full control over the connection between Alice and Bob. Thus, Eve controls all the messages sent. That means Eve can do the following:

1) Read all messages sent by Alice or by Bob;
2) Modify messages, delay messages, or replay messages; and
3) Insert messages generated by herself to Alice or Bob or both.

Recognition requires one exception, however. At the beginning of the communication between Alice and Bob a trustworthy relay channel needs to be available, similar to that required for the Diffie-Hellman protocol. The initial contact between Alice and Bob then determines the future relationship. Without such a trustworthy initial relay channel, the whole notion of entity recognition does not make sense since Alice and Bob do not actually know who they are communicating with. This modification can be defined as an *adversarial model*.

Thus, in the adversarial model we assume an initial phase where the adversary can read messages but relays them faithfully. Based on the assumptions we do not consider denial of service attacks here—the adversary could just delay all messages. The adversary aims to forge a message, that is, to make Bob accept a message believing that it originated from Alice. Considered protocols are expected to be sound and robust. Hence, if Alice and Bob behave as intended, Bob will accept messages that Alice sent, but he will not accept messages that Bob did not send or that were manipulated. Furthermore, all protocols are expected to be recoverable, so if Bob refuses to accept a message, soundness is regained for future messages.

We assume that the adversary is not able to compromise Alice or Bob in a sense that he gains knowledge of secret key material. In particular, if the adversary compromises Alice then all relationships of Alice to further entities are compromised. This is not an unusual drawback but a usual property of today's deployed systems.[2]

III. Related Work

When outlining the "resurrecting duckling" protocol, Stajano and Anderson give a simplified version of entity recognition which they call *imprinting* and *reverse metempsychosis* [1]. Imprinting establishes a shared secret between two parties. They then have the capability to recognize each other until reverse metempsychosis, whereby the parties involved discard this secret and become ready to be imprinted again.

Seigneur *et al.* describe how entity recognition applies to ubiquitous computing environments [8]. However, the paper fails to give a concrete definition for entity recognition.

Finally, Lucks *et al.* defines the *entity recognition problem* as how the receiver of anonymous messages is to determine whether the messages have been sent from the same (anonymous) source [9]. This definition is closer to message recognition rather than entity recognition since it does not consider the timeliness guarantee of an active participant (see Definitions 4 and 2).

IV. Recognition and Low-Power

A. Basis of Identification

Authentication requires the binding of an identity to a key. Classical techniques for this binding include the following [7]:

1) *Known secrets* such as passwords or Personal Identification Numbers (PINs);
2) *Trusted possessions* such as a passport or smart card; and
3) *Inherent properties* such as biometrics or a geographical location.

To accept any of these bindings, a mechanism must be in place to verify the authenticity. This mechanism is typically a pre-deployment list (a user/password list or valid passport numbers) or uses a trusted third party (TTP) to do the validation.

A low-power environment does not have access to such mechanisms. In our environment we have assumed that there exists no pre-deployment information in the network nodes and also have refused access to a trusted third party. In

[2]For instance, if the agreed key of Alice and Bob for message authentication is compromised an attacker can impersonate both Alice and Bob.

this environment it is impossible to perform identity binding. In consequence, complete entity authentication as defined is unable to succeed. In such an environment, recognition is the best we can achieve.

B. Basis of Security

Since the computational requirement of asymmetric cryptography causes performance decreases in low-power environments, many low-power protocols use a delayed key disclosure technique to achieve similar security (see Table I). This section briefly examines two common methods for key disclosure used in low-power protocols.

TABLE I
KEY DISCLOSURE OF RECOGNITION PROTOCOLS

Protocol	Key Disclosure		Other
	Time Delay	Interactive	
Resurrecting Duckling			X
TESLA	X		
μTESLA	X		
Guy Fawkes	X	X	
Remote User Auth.		X	
ZCK		X	

1) Time Delayed Key Disclosure: For an authentication or recognition scheme to use any notion of time, the two entities involved must have loosely synchronized clocks.

The Network Time Protocol (NTP), introduced by Mills [10], provides scalable clock synchronization over the wired network of the Internet. However, Elson and Römer point out that the network assumptions that NTP makes are not true in sensor networks [11]. The main issues are the low energy, multi-hop, self-organizing, and dynamic topology properties of the environment.

One of the first algorithms for distributed clock synchronization was provided by Lamport [12]. Other algorithms for reliable clock synchronization have been proposed, but few consider a malicious node in the network attempting to modify the time for its own needs.

Gong points out that maintaining the synchronization of clocks requires a secure authentication scheme [13]. Consequently, basing an authentication (or recognition) scheme on clock synchronization requires another authentication scheme to validate the time.

Moreover, time delayed disclosure requires an upper bound on the message delivery delay to be certain that the message is delivered before the key is disclosed. In multi-hop wireless networks, this message delay could be significant.

2) Interactive Key Disclosure: By disclosing the key after an interactive exchange of messages, an algorithm does not need to worry about the issues involved with clock synchronization. However, there are still a couple trade-offs that must be addressed.

Note that this method of key disclosure requires at least three messages between entities A and B (see Protocol 1). In the first message, A sends the authenticated message to B. The second message is an acknowledgement by B for receiving the message from A. Message three discloses the key to B.

The three message exchange could be a problem in an environment with high packet loss. In such a situation, time delayed disclosure may be a better solution.

A more important issue is, however, that in order to prevent an adversary E from causing A to disclose the key too early, the second message must have the data origin authenticated as coming from B.

V. RECOGNITION PROTOCOLS

The security of many of the low-power protocols we are examining is based on delayed key disclosure (see Section IV-B) and a method of hash chaining, first introduced by Lamport [14]. Coppersmith and Jakobsson [15], [16] provide efficient methods for storing and traversing single-level hash chains.

In the following, we examine some of these protocols in more detail. For the purposes of this discussion, we assume entity A is approached by entity B to be recognized.

A. The Resurrecting Duckling

The Resurrecting Duckling introduced by Stajano and Anderson [1], [17] is described as a security policy for low-power environments. We point out that the method of key exchange during the imprinting phase is open to an attack by a passive observer E. If E observes this key-sharing phase between A and B, E can impersonate B to A at any point in the future. The protocol attempts to address this by recommending this imprinting phase take place over a secure channel like direct contact. In a distributed sensor network, it is unlikely that this requirement could be satisfied.

B. TESLA

Perrig *et al.* define TESLA as a broadcast authentication protocol [2], [18]. TESLA sends messages with a MAC keyed according to time intervals. The receiver can verify the message when the key is sent in a future time interval based on a key-disclosure delay. Clock synchronization is negotiated using a digital signature algorithm, like RSA or DSA.

The time delayed key disclosure of TESLA relies on a loose, but bounded clock synchronization between the two involved parties. Clock synchronization requires authenticated synchronization messages as described in Section IV-B.1. TESLA attempts to deal with this issue by suggesting the use of digital signatures to authenticate the time response. The computational, bandwidth and memory requirements do not make it a viable solution for a sensor network environment. These issues are outlined in more detail in [3].

C. μTESLA

Recognizing the limitations of TESLA in a low-power environment, Perrig *et al.* modified TESLA to address the issues above and named the result μTESLA [3]. Like TESLA, it uses time intervals for disclosing keys, however, clock synchronization is performed in negotiation with a base station.

Now, instead of authenticating the clock synchronization messages with a digital signature, μTESLA assumes a master pre-shared secret between the base station and authenticating nodes. This pre-deployment information might not be possible in some deployment scenarios as in our target environment.

D. Guy Fawkes

The first variant of the Guy Fawkes protocol by Anderson *et al.* [4], uses codewords to publish messages and future codewords in a hash so that the codeword can be revealed later to prove that you are communicating with the same party. In the original scheme the commitment would be published in a newspaper such that the commitment would be stored in a public directory with a time-stamp and could be verified at any time. However, in an *ad hoc* network, in most cases there is no such central directory that provides time-stamps, so an explicit acknowledgement of the receipt is necessary for the security of the protocol. However, this requires the acknowledgement data to be authenticated as coming from the receiver. A second variant of the Guy Fawkes protocol was presented that fixes this issue. Here, basically both parties publish messages and future codewords in a hash that are revealed later on. Note that this scheme only requires a secure relay channel for the initial step but no confidential channel. This requirement is satisfied in the adversarial model.

The Guy Fawkes scheme requires negligible computations. However, the Guy Fawkes protocol also requires quite some bytes to be exchanged and it is more complex. To clarify, if a pair Alice and Bob only wants to authenticate a single message m_0, they need to perform two iterations of the Guy Fawkes protocol since the key for the authenticated message is opened in the next iteration.

E. Remote User Authentication

Mitchell developed a protocol called Remote User Authentication [5] which has an initial set up phase where the party wishing to be authenticated passes a random string X and a list of MACs on X keyed by a set K of secrets. Each time it wishes to be authenticated, a new set K' of secrets must be generated for new MACs for the future as well as a request for a subset R of K keys are passed back and forth, the authenticator verifying the MACs using R.

The Remote User Authentication protocol requires a high communication bandwidth. Given a random string X, A sends X to B. Then B sends t MACs to A. Furthermore, B sends r keys, t more MACs, and another random string X'. It is recommended that $t \geq 35$ and $r = t/2$. We give rough sizes to these figures: X and X' as 4 bytes each, and each MAC and key 10 bytes. Then each recognition costs $10 \cdot (2t+r)+4 \cdot 2 = 878$ bytes. This result is significantly greater than the 34 bytes calculated for ZCK, below. In low-bandwidth situations, this is will be a disadvantage.

F. Zero Common-Knowledge

Weimerskirch and Westhoff introduced a protocol called Zero Common-Knowledge (ZCK) [6], [9], [19] that uses hash chains and an initial hash image exchanged between two parties to verify messages by revealing consecutive hash pre-images in subsequent exchanges.

ZCK is similar to Guy Fawkes in efficiency, but it solves the authenticated acknowledgement problem (see Section IV-B) in only one iteration. By using a hash chain, each message is authenticated as coming from the same entity using a pre-image in the hash chain. Moreover, the protocol has far lower bandwidth requirements than remote user authentication. It passes a message X, one MAC, and two hash chain elements. If we give similar values for sizes as the Remote User Authentication: X 4 bytes, MAC and the hash chain elements 10 bytes each. Then each recognition only costs $10 \cdot 3+4 = 34$ bytes. This result is significantly less than the 878 bytes calculated for Remote User Authentication, above.

Furthermore, ZCK is not susceptible to the passive key-sharing attack found in the Resurrecting Duckling. If E observes the Initialization phase between A and B, E does not have (and cannot calculate without breaking the hash function) the hash chain pre-image necessary for messages 2 or 3 of Protocol I to impersonate either entity.

However, since ZCK uses interactive key disclosure, it could have problems in an environment with high packet loss (see Section IV-B.2). Also, the message and MAC sent in the first exchange must be buffered by B until the key is received in the third transmission. These issues are examined in more detail in Section VII.

PROTOCOL I

ZCK UNILATERAL PROTOCOL

GOAL. B recognizes A and message M to originate from A

1) *One-time setup (exchange of hash chain results).*
   a) A and B choose a random values x_0 and y_0, respectively.
   b) Each compute a hash chain $x_0 \rightarrow x_1 \rightarrow \ldots \rightarrow x_n$ (resp. $y_0 \rightarrow y_1 \rightarrow \ldots \rightarrow y_m$)
   c) A sends x_n to B
   d) B receives x_n and stores it as x'. Set $i := m$.
   e) B sends y_m to A
   f) A receives y_m and stores it as y'. Set $j := n$.
2) *Protocol messages.*

$$A \rightarrow B \quad : \quad M, d := \mathrm{MAC}_{x_{j-1}}(M) \tag{1}$$
$$A \leftarrow B \quad : \quad y_{i-1} \tag{2}$$
$$A \rightarrow B \quad : \quad x_{j-1} \tag{3}$$

3) *Protocol actions.*
   a) A chooses a message M, computes $d := \mathrm{MAC}_{x_{j-1}}(M)$ and sends B message 1.
   b) B stores d and M and sends A message 2.
   c) A checks if $h(y_{i-1}) = y'$ and if it verifies, sends B message 3 and sets $j := j - 1$. Otherwise, B waits for a new y_{i-1}.
   d) B checks if $h(x_{j-1}) = x'$ and if it verifies, sets $x' := x_{j-1}$ and verifies $\mathrm{MAC}_{x'}(M) = d$. If the MAC verification succeeds, B ACCEPTS message M as originating from a now recognized A. If the MAC fails, REJECT message M. Either way, set $i := i - 1$.

VI. PROOF OF CONCEPT

We implemented ZCK (Protocol I) as a proof that a recognition protocol can satisfy the requirements of a low-power environment. Our implementation was written in NesC, an extension to the C programming language specifically for deeply networked systems. It is developed by the University of California at Berkeley Wireless Embedded System (WEBS)

Project. We used components from another sub-project of WEBS called TinyOS, an event-driven operating system designed for sensor network nodes.

Our target platform was the Mica2 motes developed by Crossbow Technologies. We tested and performed initial analysis using a modified version of the Atemu AVR emulator developed by the Center for Satellite and Hybrid Communication Networks (CSHCN) at the University of Maryland. Furthermore, our code used in simulation also ran on actual Mica2 motes without modification.

For a one-way function, we used the MD5 hash algorithm[3] [20]. This was re-used for MAC generation according to the HMAC specification [21]. For storing and traversing hash chains, we implemented the algorithm described by Jakobsson in [15].

The code declares static space for five incoming connections and one outgoing connection. When each mote starts up it randomly chooses another mote to perform unilateral recognition with. A little overlap in the chosen partners is expected, hence the larger incoming connection array.

VII. Results

In this section we present and examine results based on our implementation of the ZCK protocol. Tables III and II give an overview of our measurements. We now evaluate the ZCK protocol with respect to a low-power environment. The properties of such environments are determined by Axioms 1 to 4. Hence we relate our measurements to these axioms.

1) Axiom 1 *(low computational power):* An MD5 hash operation takes about 30,000 clock cycles. The initial generation of a hash chain of 128 elements requires 127 MD5 iterations that run in 3,030,000 clock cycles. The computation of an hash chain element requires on average only $\lceil \log_2 128 \rceil = 7$ MD5 iterations by applying the optimizations of Jakobsson [15].

TABLE II
Performance breakdown

Operation	Clock Cycles†
One-Way Function (MD5)	30,039
Hash Chain Generation	3,026,524
Hash Chain Iteration	23,554

† these values are averaged and could suffer small variations

2) Axiom 2 *(low code space):* The program code of ZCK requires approximately 1,991 bytes of RAM and 29,032 bytes of ROM in total. This includes approximately 578 bytes of RAM and approximately 12,088 bytes of ROM for TinyOS specific code such that 1,413 bytes RAM and 16,944 bytes of ROM are induced by the ZCK overhead. For comparison, the Mica2 Motes are equipped with 4 KB SRAM and 128 KB programmable flash memory so that ZCK occupies 35% of the total RAM and 13% of the total ROM. These numbers seem to be inappropriately high. However, implementing our protocol in a highly optimized assembly we expect from our experience to achieve far smaller code sizes, namely around 100 bytes of RAM and 3 KB of ROM such that less than 3% of the total RAM are occupied by our implementation.

The hash chain represents a large portion of the required memory for ZCK. As mentioned earlier, we implemented Jakobsson's algorithm [15] which requires $2 \cdot \lceil \log_2 128 \rceil + 128 = 142$ bits per stored hash-chain element (including addition information about its location in the chain). That works out to 124 bytes in total for each hash chain. Note that this memory requirement is in the flash memory and, hence, a very small portion of the 128 KB available. Obviously, a mote must keep space for multiple hash chains since a separate one is needed for each communication partner.

TABLE III
Code and RAM size breakdown in bytes

Module	Code Size	RAM Size
One-Way Function (MD5)	9,266	0
Hash Chain	416	0
ZCK Negotiation	6,784	1,394
TinyOS	12,088	578
Total	29,032	1,991
ZCK Overhead	16,944	1,413

3) Axiom 3 *(low communication bandwidth):* The maximum size of a TinyOS packet is 36 bytes with a 7 byte header. In this payload, our implementation has 3 bytes of overhead (the message source and a sequence number). This leaves 26 bytes for the actual payload.

As we described earlier, our implementation of ZCK uses MD5 for a hash function. MD5 gives a hash result of length 16 bytes. The initialization phase exchanges two hash results (see Protocol 1). Thus the total transmitted data would be $(7+19) \cdot 2 = 52$ bytes.

The ZCK recognition phase has three transmissions. The first includes the message and a MAC. The MAC is 16 bytes, leaving 10 bytes for the message (see the discussion in Section VII-A.4). The second transmission replies with a 16 byte hash, and similarly in the third transmission. Thus the total transmission overhead for each message (not including the message itself which can vary in size) would be $(7+19) \cdot 3 = 78$ bytes.

4) Axiom 4 *(low energy resources):* We validated that our ZCK implementation fits to Axiom 1 and Axiom 3. Since the energy consumption of a protocol is composed of the computational effort and the data transmission, one can infer that our implementation requires only little energy overhead.

Finally, note that most functions can be built in hardware. Nevertheless, a hardware solution only reduces the computational overhead and not the communication bandwidth which is agreed to be the dominant factor in performance overhead.

A. Remarks and Observations

1) Security Model: The ZCK protocol satisfies the above presented adversarial model (see Section II-B). In particular, the scheme is secure if there is a reliable relay channel

[3]Recent collision attacks against MD5 do not affect the hash chain requirement for pre-image resistence. However, collision attacks do affect the security of the HMAC. We recognize that SHA-1 should be used in future implementations.

available for sending data during the initialization phase. During the first recognition Alice cannot be sure that it is communicating with the correct entity Bob, however, the more information that Alice receives that it expected Bob to have, the more certain Alice can be that it is correct. Note that entity recognition prevents a man-in-the-middle attack after the first recognition, hence if an entity John was modifying information, John would have to have been in the middle from the beginning (and hence, Alice has been recognizing John, not Bob).

Plainly spoken, the ZCK protocol is as powerful as a scenario involving Diffie-Hellman key agreement based on public keys with no certificates in order to agree on a shared key for authenticating messages by means of a MAC.

2) Denial of Service: In reference to Protocol I, ZCK requires the receiver B to buffer message M and MAC d from message 1 until they can be verified after message 3. As discovered in our implementation, this issue could require a high amount of memory if many nodes attempt to negotiate at the same time. Moreover, in TinyOS this memory requirement must be declared statically at compile time, so this could open up the protocol to denial of service attacks if a mote is flooded with recognition requests.

3) Pairwise Memory Complexity: Similar to protocols involving pairwise symmetric keys, each communicating partnership requires a shared key. In the case of ZCK, separate shared keys are needed for transmitting and receiving. That shared key is in the form of hash chains which have a significant, though not unmanageable, memory requirement. However, this property does mean that there could be a limit on the number of partners that one node negotiates with. In practice this is not expected to be a limiting factor. The multi-hop network organization of typical sensor networks allows nodes to limit their communications to a small local neighborhood. Most sensors networks are static and consequently direct neighbors do not change and are restricted. Moreover, most often in sensor network deployments, the sensor is transmitting data in one direction, for instance to a base station, rather than both transmitting and receiving. Under these assumptions a limited number of authentication possibilities is not a restriction.

4) Message Fragmentation: When describing the communication bandwidth, we mentioned that the transmission payload available for the message was 10 bytes. However, a message m that is larger than 10 bytes can easily be fragmented into packets $m = (m_0, m_1, \ldots m_n)$ and transmitted separately. The ZCK protocol is then only applied to the MAC over m. Typical monitored data in sensor networks can be transmitted with only a few bits which reduces the risk of fragmentation.

VIII. Conclusions

We introduced a new security principle called recognition that is closely related to authentication. We showed that such a new principle is appropriate for low-power environments where identification is not possible. We discovered that few proposed protocols satisfy the requirements of this limited-resource environment. Zero Common-Knowledge (ZCK) fits the requirements best out of the protocols we analyzed. By presenting our implementation as a proof-of-concept we showed that the environment can be satisfied, but we also identified areas for concern including denial-of-service, pairwise memory complexity, and message fragmentation. Future work in recognition protocols to satisfy low-power protocols will improve on our presented techniques.

References

[1] F. Stajano and R. Anderson, "The resurrecting duckling: Security issues for ad-hoc wireless networks," in *Security Protocols, 7th International Workshop Proceedings, Lecture Notes in Computer Science*, 1999.

[2] A. Perrig, R. Canetti, J. D. Tygar, and D. Song, "The TESLA broadcast authentication protocol," *Cryptobytes*, vol. 5, no. 2, pp. 2–13, 2002, RSA Laboratories, Summer/Fall 2002.

[3] A. Perrig, R. Szewczyk, V. Wen, D. E. Culler, and J. D. Tygar, "SPINS: Security protocols for sensor networks," in *Mobile Computing and Networking*, 2001, pp. 189–199.

[4] R. Anderson, F. Bergadano, B. Crispo, J. Lee, C. Manifavas, and R. Needham, "A new family of authentication protocols," *ACM Operating Systems Review*, vol. 32, 1998.

[5] C. Mitchell, "Remote user authenticaion using public information," in *Cryptography and Coding, 9th IMA International Conference on Cryptography and Coding Proceedings*, 2003.

[6] A. Weimerskirch and D. Westhoff, "Zero common-knowledge authentication for pervasive networks," in *Proceedings of Selected Areas of Cryptography 2003 (SAC 2003)*, 2003.

[7] A. Menezes, P. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*. CRC Press, 1997.

[8] J. Seigneur, S. Farrell, and C. D. Jensen, "Secure ubiquitous computing based on entity recognition," in *Workshop on Security in Ubiquitous Computing (UBICOMP2002)*, 2002.

[9] S. Lucks, E. Zenner, A. Weimerskirch, and D. Westhoff, "Efficient Entity Recognition for Low-cost Devices," Communication Security Group, Ruhr-University Bochum, Tech. Rep., 2004.

[10] D. L. Mills, "Internet time synchronization: The network time protocol," in *Global States and Time in Distributed Systems*, Z. Yang and T. A. Marsland, Eds. IEEE Computer Society Press, 1994.

[11] J. Elson and K. Römer, "Wireless sensor networks: A new regime for time synchronization," in *Proceedings of the First Workshop on Hot Topics in Networks*, Princeton, New Jersey, Oct. 2002.

[12] L. Lamport, "Time, clocks, and the ordering of events in a distributed system," *Communications of the ACM*, vol. 21, no. 7, pp. 558–565, July 1978.

[13] L. Gong, "Variations on the themes of message freshness and replay," in *Proceedings of the IEEE Computer Security Foundations Workshop VI*, Franconia, New Hampshire, June 1993, pp. 131–136.

[14] L. Lamport, "Password authentication with insecure communication," *Communications of the ACM*, vol. 24, no. 11, pp. 770–772, Nov. 1981.

[15] M. Jakobsson, "Fractal hash sequence representation and traversal," in *IEEE International Symposium on Information Theory (ISIT '02)*, 2002.

[16] D. Coppersmith and M. Jakobsson, "Almost optimal hash sequence traversal," in *Proceedings of the Fifth Conference on Financial Cryptography (FC '02)*, 2002.

[17] F. Stajano, "The resurrecting duckling — what next?" in *Security Protocols, 8th International Workshop Proceedings, Lecture Notes in Computer Science*, 2000.

[18] A. Perrig, R. Canetti, D. Song, and J. D. Tygar, "Efficient and secure source authentication for multicast," in *Network and Distributed System Security Symposium Conference Proceedings*, 2001.

[19] A. Weimerskirch, D. Westhoff, S. Lucks, and E. Zenner, "Efficient Pairwise Authentication Protocols for Sensor and Ad-hoc Networks," *Sensor Network Operations*, 2004.

[20] R. Rivest, "RFC 1321: The MD5 message-digest algorithm," Internet Activities Board, 1992.

[21] H. Krawczyk, M. Bellare, and R. Canetti, "RFC 2104: HMAC: Keyed-hashing for message authentication," Network Working Group, Feb. 1997.