# New Adversary and New Threats: Security in Unattended Sensor Networks

**Di Ma, Claudio Soriente, and Gene Tsudik, University of California, Irvine**

**Abstract**

Some wireless sensor networks preclude the constant presence of a centralized data collection point, that is, a sink. In such a disconnected or unattended setting, nodes must accumulate sensed data until it can be off loaded to an itinerant sink. Furthermore, if the operating environment is hostile, there is a very real danger of node and data compromise. The unattended nature of the network makes it an attractive target for attacks that aim to learn, erase, or modify potentially valuable data collected and held by sensors. We argue that adversarial models and defense techniques in prior WSN literature about security are unsuitable for the unattended WSN setting. We define a new adversarial model by taking into account special features of the UWSN environment. We show that in the presence of a powerful mobile adversary, securing data stored on unattended sensors presents interesting challenges and opens an exciting new line of research.

Within the last decade, sensors and sensor networks have been extremely popular in the research community. In particular, security issues in wireless sensor networks (WSNs) have received a lot of attention. Because of the low cost of individual sensors and due to meager resources, security poses unique and formidable challenges.

One common assumption in prior WSN security research was that data collection is performed in (or near) real time: a trusted entity — usually called a sink — is assumed to always be present. Presence of an online sink enables nodes to submit measurements soon after sensing. As a consequence, an adversary capable of compromising nodes and corrupting data has relatively little time to attack.

Although many WSNs operate in this mode, there are WSN scenarios and applications that do not fit into the real-time data collection model. We refer to such networks as unattended WSNs (UWSNs). Consider the following examples:

- A tree-mounted WSN composed of noise sensors, installed in a protected area (e.g., in a national park) to monitor firearm discharge (to detect poaching) or sawing (to detect illegal tree logging). The size of the protected area, its inaccessibility, and/or the difficulty of hiding a sink, can motivate the requirement for an itinerant sink.
- A subterranean WSN deployed along an international border to record illegal crossings. The scale, in terms of both the number of sensors and the area they must cover, might make it too costly to install a multitude of stationary sinks, one per border segment. Instead, periodic visits by a mobile sink (e.g., mounted on a border patrol vehicle) might be more realistic.

One common feature in these examples is that constant physical access to the entire network is impossible, and sink visits are sporadic. Consequently, sensors cannot off load data in real time: they must accumulate data in situ and wait for an explicit upload signal.

We further narrow the scope to UWSNs operating in hostile environments. Unattended sensors deployed in such environments represent an attractive and easy target for an adversary. The inability of the sensors to off load data in real time exposes them and their data to increased risk. Without external connectivity, sensors can be compromised (without detection), and collected data can be read, altered, or simply erased. Sensor compromise is a realistic threat because a typical sensor is a mass-produced commodity device with no specialized secure hardware or tamper-resistant components.

In prior security research, often it was assumed that a number of sensors can be compromised during the entire operation of the network. (Thus, the main challenge is to detect such compromise.) This is a reasonable assumption because — given a constantly present sink — attacks can be detected and isolated. The sink can take appropriate action immediately to prevent further compromise.

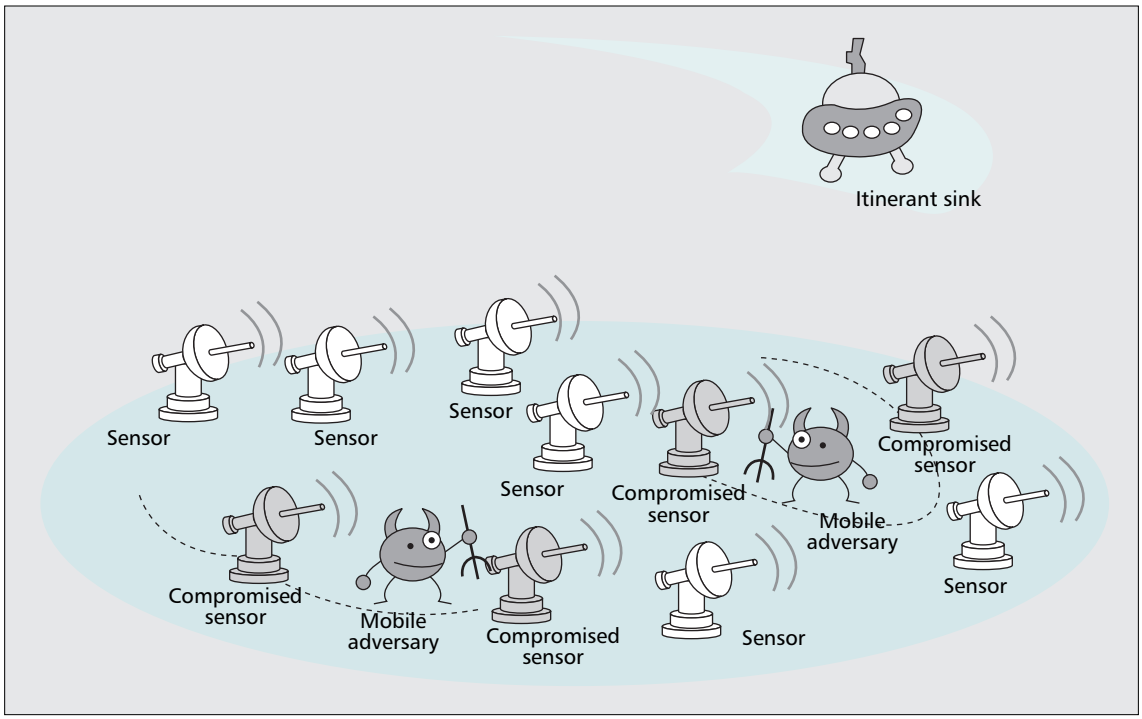
In contrast, in a UWSN, the adversary can compromise a maximum number of sensors within a particular time interval. This interval can be much shorter than the time between successive visits of the sink. Given a sufficient number of intervals, the adversary can subvert the entire network as it moves between sets of compromised nodes, gradually undermining security (Fig. 1).[1]

In the rest of this article, we discuss a number of security

[1] *A similar type of adversary is well-known in the cryptographic literature as the mobile adversary [1]. An entire branch of cryptography, called proactive cryptography, has been developing cryptographic techniques that preserve security in the presence of such an adversary [2, 3]. However, the goal of the cryptographic mobile adversary is to discover a system-wide secret (usually, a decryption or a signature key) that is pre-distributed through secret-sharing techniques among the system components; whereas, in a UWSN, the goal of the mobile adversary is to read, erase, or modify data collected by unattended sensors. Consequently, research results in proactive cryptography do not apply to the problem at hand.*

 0890-8044/09/$25.00 © 2009 IEEE 

Authorized licensed use limited to: Univ Politecnica de Madrid. Downloaded on December 9, 2009 at 03:24 from IEEE Xplore. Restrictions apply.




■ Figure 1. *μADV in a UWSN setting.*

challenges unique to UWSNs. In doing so, our main goal is to bring the problem to light and engender interest from the research community.

## Envisaged Network Environment

We assume a UWSN consisting of a multitude of homogeneous low-cost sensor nodes distributed over a certain geographical area. The term "unattended" means, as discussed previously, that sensors cannot communicate with the sink at will and that the network is not under constant supervision. The unattended nature of the network might be caused by the design requirement to avoid a central/single point of failure. Alternatively, as illustrated in the two examples in the previous section, it might be caused by poor or sporadic accessibility of the deployment area. In any case, the sink is assumed to be mobile.

We also assume that sensors are scheduled to obtain measurements from the environment periodically.[2] Each sensor has enough power and storage to collect measurements for up to a fixed maximum number of sensing intervals.

## The Mobile Adversary

We anticipate a powerful mobile adversary, hereafter referred to as an μADV. One important feature that separates it from other adversarial models is its *mobility*. We assume that a μADV can compromise a subset (up to a certain size) of sensors within a given time interval. The subset of compromised nodes might not be clustered or contiguous, that is, concurrently compromised nodes can be spread throughout the entire UWSN topology. Furthermore, in the next interval, a μADV can migrate and compromise a different subset of sensors.[3]

The time it takes a μADV to compromise a set of nodes is much shorter than the time between two successive visits of the sink. Thus, given enough compromise intervals, a μADV can gradually subvert the entire network. While a μADV occupies a given node, it can read and possibly write to the storage, memory, and all of the communication interfaces of the node. At the very least, it can learn all node secrets, as well as eavesdrop on all relevant communication.

Of course, nothing prevents a μADV from physically destroying or damaging sensors, in particular because the network is unattended most of the time. However, such crude behavior leaves evidence. Another one of our assumptions is that a μADV is subtle and prefers to operate in a stealthy manner.[4] Therefore, because it wants to "leave no trail," the movements of a μADV are unpredictable, and they also are untraceable. Specifically, it is impossible to detect if or when a μADV compromised a particular sensor.

We now consider several types of μADVs, each with slightly different goals:

- **Curious** μADV — Aims to learn as much sensed data as possible. It is not hard to read data from RAM and/or ROM of a commodity sensor [4]. With no countermeasures, a μADV can compromise nodes and read their data directly. Of course, a μADV might be especially interested in learning some specific measurements that represent critical or high-value data.
- **Search-and-erase** μADV — Aims to prevent certain target data from reaching the sink. Consider, for example, a sensor network that monitors nuclear emissions, where the sink raises an alarm if one of the nodes reports a value above a certain threshold. The goal of the μADV might be to find that value and erase it before it ever reaches the sink. If we assume that the sink tolerates some missing measurements (due to occasional errors or malfunctions), a μADV remains undetected even if it succeeds.
- **Search-and-replace** μADV — If we assume that the sink

[2] *Event-driven sensing also is possible in the unattended setting; however, we do not consider it for the time being.*

[3] *There is no requirement for a μADV to move; however, mobility is clearly in its interest.*

[4] *That is, no matter what kind of attack it mounts, a μADV strives to remain undetected.*



Authorized licensed use limited to: Univ Politecnica de Madrid. Downloaded on December 9, 2009 at 03:24 from IEEE Xplore. Restrictions apply.

has no tolerance for lost data, the corresponding adversary model changes from search-and-erase to search-and-replace. In this version, a μADV also aims to prevent target data from reaching the sink. However, it wants to replace the target data with concocted value(s).

It is not hard to envision other types of adversarial behavior relevant to UWSNs. For example, we can imagine a *polluter* μADV whose goal is to confuse or mislead the sink by introducing many fraudulent measurements into the UWSN; or, one that aims to indiscriminately erase as much data as possible, the main goal being denial-of-service. However, at least initially, we do not consider these types of behavior because they violate our previously stated assumption that a μADV wishes to remain stealthy.

We further distinguish between a *proactive* and a *reactive* adversary. The latter is assumed to be dormant (inactive) until it receives a signal to respond to certain target data. As soon as this happens, the μADV *reacts* and starts compromising nodes in order to accomplish its goal. In contrast, a proactive μADV roams the network ahead of time, compromises subsets of nodes, and waits for a signal to respond to certain target data. A proactive μADV, as we discuss below, has certain advantages over its reactive counterpart.

## General Concepts

In this section, we introduce preliminary general concepts.

*Forward and Backward Security* — Considering that the compromise of a given sensor has a certain duration, we can partition sensor-collected data into the following three categories, based on the time of compromise:

1 Before compromise
2 During compromise
3 After compromise

The security of category 1 data is referred to as forward security. *Forward security* means that even if a μADV obtains the current secrets of a sensor, it cannot decrypt (or forge authentication tags for) data collected and encrypted (or authenticated) before compromise; whereas, the security of category 3 data is referred to as backward security. *Backward security* means that a μADV that obtains the current secrets of a sensor cannot decrypt (or forge authentication tags for) data in category 3. Of course, nothing can be done about the security of category 2 data because during that time, a μADV is in full control of the sensor.

Modern cryptography offers a number of tools and techniques to achieve either forward security only [5–7] or both forward and backward security (sometimes referred to as *key insulation*) [8–12]. Key evolution is a common theme in all of these techniques. Time is divided into fixed intervals. If public cryptography is used, the public key remains fixed throughout the entire lifetime of the system; whereas, the private key is updated in each interval. After the private key of the next interval is computed, the current private key and other intermediate values are deleted. With symmetric cryptography, the pairwise (shared) key is evolved at the end of each interval.

Forward-secure techniques rely on key evolution alone, without resorting to any trusted third party. That is, usually, the private key is updated by its owner through a one-way function. In contrast, in key-insulated schemes, key evolution is performed collectively by the owner and an outside helper called a *base* — a separate secure entity, typically in the form of a remote trusted server or local tamper-resistant hardware. To update its private key, the owner contacts the base. This way, without simultaneously compromising both the owner and the base, a μADV cannot learn future keys, and backward security is achieved.

*Random Number Generators* — We distinguish between two types of random number generators: a true random number generator (TRNG) and a pseudo-random number generator (PRNG). The former extracts randomness from physical phenomena and generates information, theoretically independent values. That is, given an arbitrarily long sequence of consecutive TRNG-generated numbers, removing any one number from the sequence makes any guess of the missing number equally likely. In other words, learning a value in a given position of the TRNG-generated sequence provides no information on the values in the previous nor in the following positions.

A PRNG is an algorithm that starts with an initial value — *seed* — and uses some function(s) to produce a sequence of values that appear random. Often, one-way functions (functions that are computationally infeasible to invert) are used as PRNGs. Thus, a typical PRNG provides forward security. However, given access to the current state of the PRNG, it is possible to compute all subsequent values. Thus, a typical PRNG does not offer backward security.

*Randomized Encryption* — Randomized encryption uses a nondeterministic, probabilistic algorithm to encrypt data. Informally, given two ciphertexts encrypted under the same key, it is infeasible to determine whether the corresponding plaintexts are the same. A form of randomness is always involved in the randomized encryption process. In the context of symmetric encryption, different keys can be used to achieve randomized encryption. In the context of public-key encryption, because the public key used to encrypt data usually remains constant, a random value is required as one of the inputs (along with plaintext) to the encryption function [13].

## Potential Defense Strategies

We now describe some potential defense strategies against the aforementioned μADV types. As can be expected, encryption and authentication, as well as data dissemination can mitigate attacks to some extent.

There is an important difference between a curious μADV and the other two types (search-and-erase and search-and-replace) that is related to data location. With a curious μADV, moving data around the network (to keep it away from compromised nodes) does not help. Because a μADV is mobile, it eventually *catches up* with all data, whether the data is moved around or not. Thus, to protect against a curious μADV, nodes might not be required to communicate among themselves, except perhaps for the purposes of encryption or authentication techniques that might require cooperation. Whereas, if a μADV *searches* the network for specific data, cooperation among sensors might be desirable. In particular, sensors might benefit from off loading their data to other sensors in an effort to hide it from a search-and-erase or search-and- replace adversary. Note that we cannot assume that sensors are aware of the exact data that is targeted by an μADV. Therefore, all data must be considered as a potential target.

### Curious μADV

Because a curious μADV is interested only in learning actual sensed data, encryption seems like an effective defense strategy; without knowing a sensor's encryption key, the μADV cannot learn the data. Any secure encryption scheme can be used, whether symmetric or public key. However, as discussed



Authorized licensed use limited to: Univ Politecnica de Madrid. Downloaded on December 9, 2009 at 03:24 from IEEE Xplore. Restrictions apply.

above, encryption must be *randomized* [13]. This is because the range of possible sensor readings (i.e., plaintext) might be quite restricted or small. If encryption is not randomized, the same sensor readings, once encrypted, would produce identical ciphertexts, and a μADV would be able to infer and correlate plaintexts by simply analyzing the corresponding ciphertexts.

However, randomized encryption alone is not sufficient. After a sensor is compromised and a μADV learns its secrets, collected data — even if encrypted —is exposed immediately. This holds regardless of where encrypted data is stored — on the sensor that produced it or elsewhere. To defend against potential key exposure, both forward and backward security are required.

With symmetric encryption, assuming that each sensor shares a distinct initial key with the sink, forward security can be obtained easily by periodically evolving the encryption key through a one-way function. A curious μADV who compromises a sensor and obtains its current key cannot compute pre-compromise keys (and decrypt pre-compromise data) due to the one-wayness property of the key update function. Unfortunately, this does not help with backward security because a μADV can mimic the key update function easily and calculate all future keys.

With public-key encryption, a sensor can use the sink's public key to encrypt data.[5] Because a μADV does not know the corresponding private key, the only way it can learn data is by the guessing attack, that is, trying to encrypt all possible data values with the sink's public key. Two identical values will result in the same ciphertext. However, as discussed above, encryption should be randomized — a sensor must produce a random number for each encryption operation. Assuming that random numbers are sufficiently long (e.g., at least 160 bits), the guessing attack becomes computationally infeasible.

Next, the problem becomes more subtle: we must zoom in on the source of random numbers. If sensors generate random numbers using a TRNG, both forward and backward security are attained trivially. If sensors use PRNGs, a μADV still cannot obtain pseudo-random numbers used before compromise.[6] Unfortunately, given a snapshot of the current state of the PRNG, computing future pseudo-random numbers is trivial, and no backward security is provided.

We observe that the distinction between TRNGs and PRNGs applies only to public-key encryption. In case of symmetric encryption, *backward security is unattainable*, regardless of the source of randomness. This holds as long as a sensor relies only on itself for security. However, as we discuss later, backward security might be possible if sensors cooperate [14, 15].

At this point, it might seem natural to consider more complex — and seemingly relevant — key-insulated [9] and intrusion-resilient [8] encryption schemes. They are designed to provide both forward and backward security to mitigate the effect of exposure of decryption keys. However, such techniques are completely unsuitable for our problem because in our case, nodes do not possess decryption keys. They use the sink's public key only to encrypt data. Therefore, a key-insulated or an intrusion-resilient scheme can help only against the sink's private key compromise — a problem irrelevant in our context.

[5] *Note that in this case, no other keys, private or symmetric, are required.*

[6] *Recall that a PRNG typically produces pseudo-random numbers based on an internal state and a one-way function. Thus, computing prior values from the current state requires inverting a one-way function.*

*Summary* — Based on the above discussion, we conclude that except for public-key encryption used in conjunction with TRNGs, no basic encryption scheme achieves the desired level of protection: forward and backward security. We believe that the combination of public-key encryption and per-sensor TRNG is not realistic for many current and emerging sensor-network settings. Public-key encryption involves high computation costs and consumes higher storage and bandwidth than symmetric encryption. Similarly, node-specific TRNGs often are not realistic.

*Challenge* — We identify a major challenge in developing effective and inexpensive techniques that eschew both public-key encryption and TRNGs while enabling UWSN nodes to recover from compromise by a curious μADV. We anticipate that such techniques must involve cooperation among nodes. It is not hard to imagine potential directions, for example, nodes periodically soliciting or volunteering random contributions from/to other nodes. However, the difficulty is in proposing optimal defense strategies while keeping in mind that the counterstrategy of the adversary is attuned to the specifics of these defense strategies.

## Search-and-Erase μADV

Recall that the main goal of a search-and-erase μADV is to prevent certain target data from reaching the sink. We assume that between any two successive sink visits, a search-and-erase μADV can erase no more than a fixed number of measurements. Erasing more exposes the presence of a μADV to the sink and contradicts its goal of remaining undetected. To succeed, a μADV must erase the target data before the next visit of the sink. However, sensors are unaware of the exact data that a μADV is targeting, thus, all data must be protected equally.

When sensors are on the very low end of the cost spectrum, they might not have cryptographic capabilities. In this case, the only possible defense is to move all of the data around the network attempting to hide its location. Although not entirely useless, this strategy is ultimately futile as demonstrated in [16]. Given enough rounds, a μADV is almost guaranteed to find and erase the target data. This is independent of the particulars of the data migration strategy (e.g., whether data is moved once or kept moving constantly).

If replication is used, the survival probability of the target data increases appreciably, as long as the maximum number of nodes a μADV can compromise (i.e., as a fraction of the total number of nodes) within a single interval is small. As shown in [16], the benefits of replication vanish after some rounds, confirming that data migration and data replication are not sufficient to assure data survival in the presence of a search-and-erase μADV.

The natural next step is to explore the design space of cryptographic techniques. Obviously, encrypting data might prevent a μADV from recognizing it. If a μADV cannot distinguish (by decrypting) the ciphertext that contains the target data from among a large pool of ciphertexts, it can only *guess* which ciphertext to erase. Thus, a larger set of indistinguishable ciphertexts directly implies a higher data survival probability.

The first distinction is based on the type of encryption used: symmetric or public key.

With symmetric encryption, each node shares a unique key with the sink. A node uses this key to encrypt data and at a later time, the sink uses it to decrypt. If the shared node-sink key is constant, a μADV has no need to be proactive. Once it receives an external signal specifying the target data, a μADV compromises the target sensor, learns its key, decrypts all



Authorized licensed use limited to: Univ Politecnica de Madrid. Downloaded on December 9, 2009 at 03:24 from IEEE Xplore. Restrictions apply.

ciphertexts, and erases the target data. Even if all data is moved around the network, knowledge of the encryption key enables the adversary to recognize the target data when it encounters it (in encrypted form) elsewhere. A natural countermeasure is to require sensors to evolve their keys (as shown earlier) to gain forward security. This defends against a reactive μADV; key evolution makes it impossible to decrypt/recognize target data, especially if it is moved away from the sensor that generated it. However, a proactive μADV that has sufficient time to "visit" all nodes before receiving an external signal can compute any key used by any node starting from the time of the compromise of that node. Defending against a proactive μADV requires considering more sophisticated defense strategies.

With public-key encryption, nodes have no secret keys. Each node has the public key of the sink that it uses to encrypt data. After collection, the sink uses its private key to decrypt all ciphertexts. However, as shown earlier, the important issue is the source of randomness for public-key encryption. Use of PRNGs enables a μADV to recompute all randomness used by a node since compromise (i.e., no backward secrecy). As expected, if each node is equipped with a TRNG, both forward and backward security become viable.

An encryption technique (whether symmetric or public key) can be combined with a data migration strategy to achieve a better chance of data survival. To succeed, a μADV must first identify and compromise the node hosting the ciphertext containing target data and then, decrypt or otherwise recognize the target data before erasing it. To hinder the goals of the μADV, data migration cannot be deterministic; for example, each node should pick a random peer node to host its data. Also, other techniques such as re-encryption (in the case of symmetric encryption) or re-randomization (in the case of public key) might help further to obscure ciphertexts and make it much harder for the μADV.

Ideally, even in the presence of a proactive μADV, the network should be capable of forcing the μADV to erase data blindly, that is, its best strategy is to select at random and erase some unrecognizable ciphertext. However, the only way to assure this is to require the use of public-key encryption in conjunction with TRNGs.

*Summary* — The use of cryptographic tools is imperative to defend against a search-and-erase μADV. Simply moving data around to conceal its location, with or without replication, is effective only if the sink visits the network very frequently, or the number of nodes that a μADV can compromise at any time is very low. There are interesting similarities in defense strategies against a curious and a search-and-erase μADV. The highest level of security is achieved with public-key cryptography and a per-sensor TRNG. Unfortunately, both are not likely to be found on current commodity sensors.

*Challenge* — The initial problem is how to effectively combine lightweight (i.e., symmetric) cryptographic techniques that involve node collaboration (to obtain backward secrecy) with bandwidth-conscious data migration methods. These issues were explored (but not thoroughly enough) in some recent work [14, 15]. The next step is a thorough understanding — most likely to be gained through careful analysis and simulations — of the efficacy and the cost of such defense strategies.

### Search-and-Replace μADV

A *search-and-replace* μADV appears to be very similar to the search-and-erase type. The main difference is that the former must not only identify and erase the target data but also must replace it with a "credible" substitute. It is easy to see that most of our discussion in the previous section also applies to the search-and-replace μADV. Specifically, the use of data migration combined with forward and backward secrecy still provides the best possible countermeasure to μADV recognition of the target data. However, there are some additional issues.

In particular, data authentication, which was not relevant in the case of a search-and-erase μADV, now must be taken into account. Note that to construct a credible replacement for the erased data, a μADV must produce an appropriate authentication tag, such as a signature or a message authentication code (MAC), depending on the type of cryptography used. Once again, we must consider both forward and backward security, however, this time in the context of authentication.

Forward-secure MAC [6] and forward-secure signature schemes [5, 7] can be found in the literature. Although it is normal to expect that public-key signature schemes offer better security than symmetric MACs, this is not the case here. In reality, forward-secure signatures are no more secure than forward-secure MACs. The reason is that a signing (private) key periodically must evolve in order to obtain forward security. If the evolution process is random, signature verification would be impossible. Thus, the signing key must evolve in essentially the same manner that other secrets would — through a one-way function. Therefore, after a μADV compromises a sensor, it does not matter whether signatures or MACs are used for data authentication; in either case, a μADV can evolve only the current (MAC or signature) key into the future and cannot roll it back due to the one-wayness of the key evolution process.

*Summary* — Intrusion-resilient authentication in UWSNs is harder to achieve than intrusion-resilient confidentiality. As we discussed above, authentication does not benefit from public-key cryptography (signatures) and TRNGs. One conclusion is that in the case of a search-and-replace μADV, there is absolutely no incentive to use public-key signatures over symmetric MACs.

*Challenge* — Unfortunately, all current MAC and signature schemes that offer both backward and forward security require either per-node secure hardware or an online trusted third party. This makes them impractical for UWSNs. Although there are no alternatives with an equivalent level of security, we expect that some progress can be made by using collaborative techniques to help nodes recover from past compromise and to evolve MAC keys in a way that cannot be mimicked by an μADV. Some initial progress recently was made in [17].

## Related Work

Because of similarities between WSNs and mobile ad hoc networks (MANETs), related work comes from research on data availability in MANETs.

Hara *et al*. [18] proposed several algorithms for data replication in MANETs, whereby data remains available (with high probability) in the event of network partitions. This solution keeps replicas consistent (in case of update) and includes a location-management protocol guaranteeing that requesting nodes access the closest replica. Gianuzzi *et al*. [19] also studied data availability through replication in the presence of MANET partitions. Chakrabarti *et al*. [20] defined various conditions (e.g., node death rate and retransmission rate) for sensitive data survival in peer-to-peer (P2P) systems and WSNs. This work derived simple survivability conditions and



Authorized licensed use limited to: Univ Politecnica de Madrid. Downloaded on December 9, 2009 at 03:24 from IEEE Xplore. Restrictions apply.

validated the proposed model through simulations with several realistic topologies. Other results [21, 22] take advantage of multipath routing between nodes to statistically improve data confidentiality and availability in hostile MANET environments.

Another body of related research focused on data availability in WSNs. In contrast to our UWSN setting, there is an assumption of a constantly present (online) trusted entity, that is, a sink. Benenson *et al*. [23] investigated techniques to prevent a μADV from learning certain collected data, as well as retaining access to target data over the long term. Chessa *et al*. [24] presented a P2P-based, distributed data storage solution for MANETs that guarantees data confidentiality and dependability. A more recent related result that addresses data availability in WSNs is [25]. It constructs a scheme to maximize the amount of data recovered at the sink when a portion of the network is affected by natural disasters. Subramanian *et al*. [26] provide secure, distributed data storage for WSNs, where only authorized entities can access collected data.

## Conclusions

In this article, we presented our vision of an emerging type of unattended WSN and a model of an adversary more powerful than that traditionally encountered in other network (and sensor-network) settings. We argue that the unattended nature of the environment lends itself to a nimble μADV that aims to gradually undermine the security of the entire network. Our initial analysis shows that current techniques do not offer sufficient protection against a μADV in UWSNs. The limited resources of individual sensors combined with the power of the μADV make the problem very challenging. By investigating the problem further, we identified two properties — forward security and backward security — crucial for mitigating μADV threats. However, current encryption and authentication techniques that provide these properties are unsuitable for the UWSN model due to sensor computational limitations, as well as lack of trusted hardware components and/or online trusted parties. One conclusion is that to defend against the envisaged adversary, sensors must collaborate to:

- Help each other recover from compromise (to regain a secure state)
- Hide the origin, contents, and locations of sensed data until the next visit of the itinerant sink

New ideas and techniques are clearly required to address the challenge posed by the UWSN μADV model. We hope that this exploratory article offers a new research direction and stimulates some discussion within the research community.

## References


[1] R. Ostrovsky and M. Yung, "How to Withstand Mobile Virus Attacks," *ACM PODC '91*, 1991.
[2] Y. Frankel *et al.*, "Proactive RSA," *Crypto'97*, 1997.
[3] T. Rabin. "A Simplified Approach to Threshold and Proactive RSA," *Crypto '98*, 1998.
[4] J. Deng *et al.*, "A Practical Study of Transitory Master Key Establishment for Wireless Sensor Networks," *IEEE SecureComm '05*, 2005.
[5] M. Bellare and S. Miner, "A Forward Secure Digital Signature Scheme," *Crypto '99*, Aug. 1999.
[6] M. Bellare and B. Yee, "Forward Integrity for Secure Audit Logs," UCSD CSE Dept. tech. rep. 1997 no. 23, 1997.
[7] G. Itkis and L. Reyzin, "Forward-Secure Signatures with Optimal Signing and Verifying," *CRYPTO '01*, 2001.
[8] Y. Dodis *et al.*, "A Generic Construction for Intrusion-Resilient Public Key Encryption," *CT-RSA '04*, Feb. 2004.
[9] Y. Dodis *et al.*, "Key-Insulated Public Key Cryptosystems," *Eurocrypt '02*, May 2002.
[10] Y. Dodis *et al.*, "Strong Key-Insulated Signature Schemes," *PKC'03*, 2003.
[11] G. Itkis, "Intrusion-Resilient Signatures: Generic Constructions," *SCN '02*, 2002.
[12] G. Itkis and L. Reyzin, "SiBIR: Signer-Base Intrusion-Resilient Signatures," *CRYPTO '02*, 2002.
[13] V. Shoup, "OAEP Reconsidered," *Crypto '01*, 2001.
[14] R. Di Pietro *et al.*, "POSH: Proactive Cooperative Self-Healing in Unattended Wireless Sensor Networks," *IEEE SRDS '08*, 2008.
[15] D. Ma and G. Tsudik, "DISH: Distributed Self-Healing in Unattended Sensor Networks," *SSS '08*, 2008.
[16] R. Di Pietro *et al.*, "Catch Me (If You Can): Data Survival in Unattended Sensor Networks," *IEEE PERCOM '08*, Mar. 2008.
[17] R. Di Pietro *et al.*, "Collaborative Authentication in Unattended Sensor Networks," *ACM WiSec '09*, 2009.
[18] T. Hara and S. Madria, "Data Replication for Improving Data Accessibility in Ad Hoc Networks," *IEEE Trans. Mobile Comp.*, vol. 5, 2006, pp. 1515–32.
[19] V. Gianuzzi, "Data Replication Effectiveness in Mobile Ad Hoc Networks," *ACM PE-WASUN '04*, 2004.
[20] D. Chakrabarti *et al.*, "Information Survival Threshold in Sensor and P2P Networks," *IEEE ICC '07*, 2007.
[21] V. Berman and B. Mukherjee, "Data Security in MANETs Using Multipath Routing and Directional Transmission," *IEEE ICC '06*, 2006.
[22] P. Papadimitratos and Z. Haas, "Secure Data Communication in Mobile Ad Hoc Networks," *IEEE JSAC*, vol. 24 no. 2, 2006, pp. 343–56.
[23] Z. Benenson, P. Cholewinski, and F. Freiling, "Simple Evasive Data Storage in Sensor Networks," *PDCS'05*, 2005.
[24] S. Chessa and P. Maestrini, "Dependable and Secure Data Storage and Retrieval in Mobile, Wireless Networks," *DSN '03*, 2003.
[25] A. Kamra *et al.*, "Growth Codes: Maximizing Sensor Network Data Persistence," *ACM SIGCOMM '06*, 2006.
[26] N. Subramanian, C. Yang, and W. Zhang, "Securing Distributed Data Storage and Retrieval in Sensor Networks," *Pervasive Mobile Comp.*, vol. 3, no. 6, 2007.


## Biographies


DI MA (dma1@ics.uci.edu) received a B.Eng. degree from Xi'an Jiaotong University, China, and an M.Eng. degree from Nanyang Technological University, Singapore. She is a Ph.D. candidate in computer science at the University of California, Irvine. Her research interests include computer and network security, data and storage security, and applied cryptography. She was with the Institute for Infocomm Research, Singapore from 2000 to 2005.

CLAUDIO SORIENTE (csorient@ics.uci.edu) received a Laurea degree in computer science in 2004 from the University of Salerno. He is a Ph.D. candidate in computer science at the University of California, Irvine. His research interests include wireless sensor networks and applied cryptography.

GENE TSUDIK (gtsg@ics.uci.edu) received his Ph.D. in computer science from the University of Southern California in 1991. He is a professor of computer science at the University of California, Irvine (UCI). His research interests include computer and network security/privacy, as well as applied cryptography. Before coming to UCI in 2000, he was at the IBM Zurich Research Laboratory (1991–1996) and the USC Information Science Institute (1996–2000).





Authorized licensed use limited to: Univ Politecnica de Madrid. Downloaded on December 9, 2009 at 03:24 from IEEE Xplore. Restrictions apply.