Chapter 6
Ultra-lightweight Authentication

Xavier Carpent, Paolo D'Arco, and Roberto De Prisco

Abstract In this chapter we provide a *critical look* at the state of the art in ultra-lightweight authentication protocols. We start by outlining the features of the current ubiquitous and pervasive computing environment that have motivated the development of the ultra-lightweight paradigm which uses only basic arithmetic and logical operations. We emphasize its goals and its main challenges. Then, we focus our attention on the authentication problem. We use an abstract framework for modeling the protocols proposed over the years, in order to discuss their design strategies and the security and privacy properties they aim to achieve. After that, we survey the weaknesses and the common pitfalls in both the design and the analysis of ultra-lightweight authentication protocols. Finally, we conclude the chapter by discussing some fundamental ideas and research directions.

6.1 Introduction

6.1.1 A Fully Connected World of Small Devices

Small and inexpensive devices are becoming increasingly important in today's technological infrastructures. Modern computing paradigms, pervasive in nature, involve methods for monitoring the status of physical objects, capturing meaningful data, and communicating the data through network channels to processing servers. In many cases, the endpoint elements of connected systems are small and inexpensive devices attached to physical objects. These devices carry identifying information, and are used to achieve certain functionalities: to open and lock doors, control a heating system, catalog items in a shopping basket, identify objects,

X. Carpent
University of California, Irvine, CA, USA

P. D'Arco · R. D. Prisco (✉)
University of Salerno, Fisciano, Italy
e-mail: robdep@unisa.it

G. Avoine, J. Hernandez-Castro (eds.), *Security of Ubiquitous Computing Systems*,
https://doi.org/10.1007/978-3-030-10591-4_6

operate anti-theft systems, and much more. Wireless communication plays an important role in this landscape, especially in dealing with moving objects where Radio and Near-Field frequencies are commonly used. In the specific case of Radio-Frequency Identification (RFID), there are "Tags" and "Readers". Tags are tiny devices used to label objects; they contain data and communicate with the readers. Readers are bigger devices that collect and forward information to a backend server that processes the data. RFID tags are already widely deployed to track objects (e.g., goods dispatched in a distribution hub). Tags, in their most basic form, the *passive* one, have no battery: they receive their energy wirelessly from the reader. Tags are extremely cheap, with costs in the order of few cents. They are severely constrained in terms of computing power.

In general, small devices, in all forms currently available, are the weak link in the system (e.g., see [579] for a recent attack), and good solutions to the security and privacy concerns are of paramount importance. In particular, *authentication*, the process through which two entities confirm their identities to each other, is a fundamental step for the development of secure applications.

6.1.2 Authentication: Protocol Classification and Physical Constraints

Unfortunately, the authentication problem, in the ultra-lightweight setting, is a challenging one. Indeed, the devices' limitations severely impact the design of the protocols. In [139] a coarse classification partitions authentication protocols into 4 categories: *full-fledged*, *simple*, *lightweight*, and *ultra-lightweight*. The division is based on the capabilities of the constrained devices. Full-fledged protocols allow the use of public-key and symmetric-key cryptography. Thus, they can fully exploit standard cryptographic tools. Simple protocols rely on a limited number of cryptographic functionalities like pseudo-random numbers generation and hashing. Lightweight protocols further restrict the usable cryptography. They avoid hashing, and resort to using simpler operations like CRC checksums. Finally, ultra-lightweight protocols rely only on basic arithmetic and logical operations (modular addition, `and`, `or`, `xor`, etc.).

Although the above classification does not provide an exact distinction among the various classes, we still adopt it since it has been used in several papers that have appeared in the literature. In this chapter we are concerned with very small computing elements, like passive RFID tags, and with ultra-lightweight authentication protocols for such devices. It is very likely that a large percentage of tomorrow's interconnected world will consist of ultra-lightweight computing elements. Indeed, as observed in [308], although technological advances allow us to build inexpensive devices with improved capabilities at the same price, usually the market dictates the use of increasingly cheaper devices with the same capabilities. Hence, we should expect to keep dealing with the least powerful ones.

What exactly are the limitations imposed by these inexpensive devices? In [26] the authors provide a detailed description of the constraints.[1] These constraints are mostly influenced by hardware factors: chip size, power consumption, and clock speed. A standard measure for the computing power of such devices is the number of Gate Equivalent (GE) elements, which reflects the number of logic gates that the circuit integrated on the device consists of.

Let us consider RFID tags as an example. An RFID tag can communicate at very slow rates (typically under 200 kb/s), and this imposes, assuming that authentication has to happen within a reasonable time limit (e.g., 150 ms), an upper bound on the size of the total communication that the protocol can use. RFID tags usually consists of no more than 2000 GEs. Such a limit is imposed by the available physical area and by the cost of the device. Most of the gates are used for the tag's basic functionalities, and only a small fraction of them remain available to implement an authentication protocol. The cheapest RFID tags are passively powered. They receive power through an electromagnetic field, radiated from the reader; this limits the total power consumption that can be used in a single run of the authentication protocol. The power available to the tag is inversely proportional to the maximum distance at which the tag and the reader have to operate: a greater distance implies less available power and this imposes limits on the clock speed (a typical limit is 100 kHz) and, consequently, on the number of instructions that the tag is allowed to execute to finish a run of the protocol within a given time bound. Another limitation of RFID tags is the total number of memory bits: a typical limit is 2048 bits.

Finally, notice that authentication protocols often rely on random or pseudo-random number generators. Passive RFID tags can hardly afford such a component. There exist low-cost pseudo-random generators, but they still pose a substantial burden for an RFID tag. A generator might require the use of more than 1000 GEs, which is more than half of the total number of GEs usually available on these devices.

6.1.3 Design Challenges

Authentication can be achieved in several ways. Standard authentication protocols exhibit a *challenge-and-response* structure, and exploit public-key or symmetric-key cryptography. Sometimes they require the presence of a trusted third party. In all cases, the parties involved in the protocols must be able to execute the required cryptographic algorithms (e.g., encrypting a piece of data using AES). So it goes without saying that standard authentication protocols are not tailored for ultra-lightweight devices. Thus, ultra-lightweight authentication protocols using

[1]The title of [26] uses the term "lightweight", but its authors do not use the classification proposed in [139]. The discussion provided in [26] is, indeed, about ultra-constrained devices, like RFID tags.

only elementary operations are needed. It should therefore not come as a surprise that achieving the same security levels as those offered by standard authentication protocols might be much more difficult, or perhaps even impossible.

Thus, the real challenge posed by ultra-lightweight authentication is obtaining the highest possible level of security, given the hardware constraints. Part of the challenge concerns the development of a formal model that can be used to assess the security and privacy achieved by ultra-lightweight authentication protocols.

Nowadays, security assertions are expressed in terms of formal mathematical models for describing problems and analyzing proposed solutions. In particular, security assertions are expressed in formal mathematical terms, cryptographic protocols are built upon computational hardness assumptions, and proofs assume the form of mathematical reductions. As we will argue in the following sections, ultra-lightweight cryptography should be tackled with a similar rigorous approach. We might have to rethink, or to appropriately adapt, the formal framework within which ultra-lightweight protocols are designed and security and privacy assertions about them are assessed.

6.1.4 Organization of the Chapter

In Sect. 6.2 we provide a general concise framework which captures the common structure of known ultra-lightweight authentication protocols, and we discuss the design strategies and properties they aim to achieve. Then, in Sect. 6.3, we point out the limits of achieving security by using very constrained computing devices which allow only simple operations. Specifically, we survey the weaknesses and the common pitfalls in the design of ultra-lightweight authentication protocols. In Sect. 6.4, we elaborate on the importance of using security and privacy models, and provide suggestions for sound design strategies. Finally, in Sect. 6.5 we provide some conclusions.

6.2 Ultra-lightweight Authentication Protocols

Ultra-lightweight mutual authentication protocols appeared in the literature around 2006. M^{2}AP [467], LMAP [466] and EMAP [465] were the first protocols designed to be executed on circuits equipped with only a few hundred gates. They were collectively identified as the *UMAP family*. In the following year, another protocol, called SASI [139], addressed some of the weaknesses present in those protocols. SASI received considerable attention both from cryptanalysts and designers. However, like its predecessors, it was quickly broken in a few months. Surprisingly, plenty of similar protocols followed, and its "structure" is still being used.

Almost all proposed ultra-lightweight mutual authentication protocols can be seen as instances of one general framework. Three entities are involved: a tag, a reader and a backend server. The channel between the reader and the backend server is assumed to be secure, but the channel between the reader and the tag is public and is susceptible to attacks. To simplify the description, we say that the reader performs some computations, even if the reader just forwards the messages and the backend server is the real entity that performs the computations.

Each tag has a *static identifier*, ID, which is hard-coded into the circuit at production time and is never revealed. Furthermore, the tag has a *pseudonym*, IDS, and a few *secret keys*, which are stored in the tag memory, and are usually updated after each successful execution of the protocol. All of these values are bit-strings of up to 100 bits. Common values are 64 and 96.

Readers are expected to be able to generate *random numbers* or *pseudo-random numbers*.

The backend server, for each tag with static identifier ID, stores in a table the pseudonym and the keys, which therefore are shared with the tag.

The authentication protocol consists in a few rounds. Typically, four.

Figure 6.1 depicts the structure of many ultra-lightweight authentication protocols. Here we provide a description of the messages:

- The *Hello* message is the starting message with which the reader activates the tag, providing it with the energy for the subsequent computation.

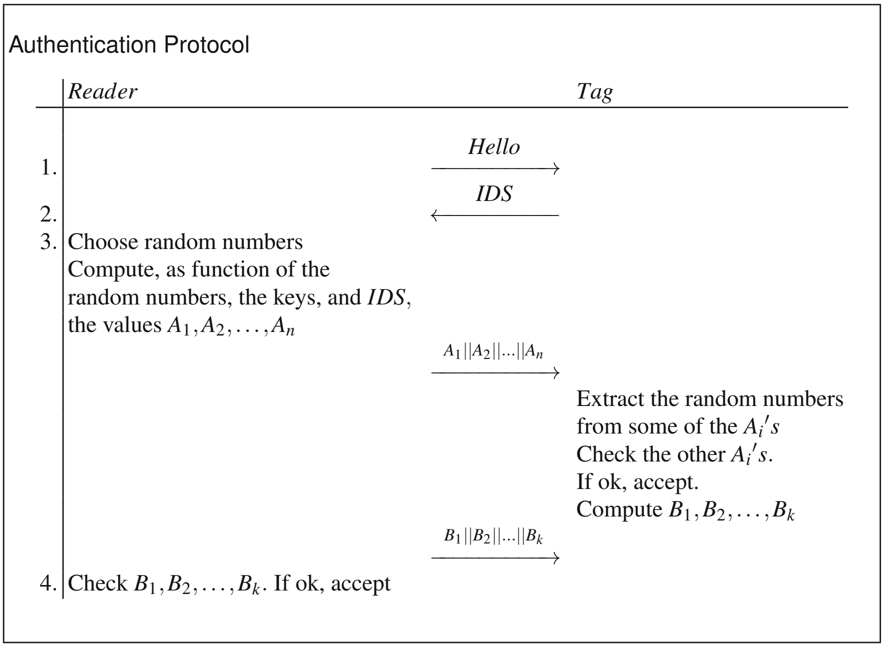
Authentication Protocol
Reader
Tag
Hello
1.
IDS
2.
3. Choose random numbers
Compute, as function of the
random numbers, the keys, and IDS,
the values $A_1, A_2, \ldots, A_n$
$A_1||A_2||...||A_n$
Extract the random numbers
from some of the $A_i's$
Check the other $A_i's$.
If ok, accept.
Compute $B_1, B_2, \ldots, B_k$
$B_1||B_2||...||B_k$
4. Check $B_1, B_2, \ldots, B_k$. If ok, accept

Fig. 6.1 General framework: steps of the authentication protocol

- The *IDS* is the current pseudonym, which the tag sends to the reader and initiates the subsequent authentication protocol.
- The sequence of values $A_1, A_2, \ldots, A_n$, computed by the reader, is usually used in the following form: the first values are a sort of *carriers* for fresh randomly or pseudorandomly generated numbers, while the last ones are the *true authenticators*, computed by using the random numbers, the secret keys and information shared between the reader and the tag. From some of the A_i's, the tag, by using the secret keys, retrieves the random numbers chosen by the reader. Then, by using the secret keys, the retrieved random numbers and some other shared information, the tag recomputes the remaining A_i's and checks that they match the ones received. Such a check aims at ensuring the integrity and the authenticity of the transmitted values.
- The values $B_1, B_2, \ldots, B_k$ are, finally, used by the reader as an acknowledgment that the tag has authenticated the reader, and to complete the authentication of the tag to the reader. They are generated and used in a similar way to the values $A_1, A_2, \ldots, A_n$.

At the end of a successful execution, the reader and the tag change the pseudonym *IDS* of the tag, and all the secret keys, by applying some updating functions. The updating functions use the *IDS* and the secret keys, as well as some of the random numbers, used in the last execution of the protocol. In particular, the updating function for the *IDS* uses also the static tag *ID*.

In practice, many protocols require the reader and tag to store both the new *IDS* and the sequence of secret keys, as well as the previous *IDS* and the previous sequence of secret keys. The reason is that the tag completes the protocol before the reader. If for some reason, adversarial or not, the reader does not complete the protocol, the tag updates the *IDS* and the secret keys while the reader does not. Then, at the subsequent execution, the reader and the tag do not recognize each other. Technically speaking, they are not *synchronized* anymore. By also keeping the old tuple of values, the authentication protocol can be modified in such a way that, if the reader does not reply to the new *IDS*, then the tag sends the old *IDS* again and the authentication protocol is executed using the old sequence of secret keys.

To exemplify the general framework, notice that in M^{2}AP and EMAP three values are sent from the reader to the tag, and two values are sent from the tag to the reader. In LMAP, SASI, and Gossamer [463], three values are sent from the reader to the tag and one value is sent from the tag to the reader. Moving ahead to more recent protocols, in KMAP [323], RCIA [431] and SASI$^+$ [431], three values are sent from the reader to the tag and one value is sent from the tag to the reader, while in SLAP [383] two values are sent from the reader to the tag, and one value is sent from the tag to the reader. However, some protocols slightly deviate from the general framework, e.g., RAPP [554] has one more round.

To get an idea of the computations, let us look at SASI. Denote by K_1 and K_2 two secret keys shared between the reader and tag, and by n_1 and n_2 two fresh random values generated by the reader. Moreover, we denote by $\oplus, \vee, +$ the `xor`, `or` and modular addition operators. Finally, denote with $Rot(s, \ell)$ a bit-string cyclic rotation function, which returns the string s rotated circularly to the left by ℓ positions. The three values, A_1, A_2 and A_3, computed by the reader, are:

$$A_1 = IDS \oplus K_1 \oplus n_1, \quad A_2 = (IDS \vee K_2) + n_2, \text{ and } A_3 = (K_1 \oplus \overline{K_2}) + (\overline{K_1} \oplus K_2),$$

where $\overline{K_1} = Rot(K_1 \oplus n_2, K_1)$ and $\overline{K_2} = Rot(K_2 \oplus n_1, K_2)$. Then, the value B_1, computed by the tag, is

$$B_1 = (\overline{K_2} + ID) \oplus ((K_1 \oplus K_2) \vee \overline{K_1})$$

The updating functions for the pseudonym and the keys are:

$$IDS = (IDS_{old} + ID) \oplus (n_2 \oplus \overline{K_1}), \quad K_1 = \overline{K_1}, \quad K_2 = \overline{K_2},$$

Having considered a sample computation, let us move to the basic requirement for an authentication protocol, that is, correctness: if the reader and tag initiate a protocol execution when they share at least one *IDS* and the corresponding sequence of secret keys, and no adversarial action or transmission error occurs, then they should successfully complete the execution and authenticate each other.

The main security and privacy goals in the design of ultralightweight authentication protocols are:

- **Resistance to desynchronization attacks.** An adversary should not be able to desynchronize the reader and tag.
- **Resistance to impersonation attacks.** An adversary should not be able to impersonate the reader to the tag or the tag to the reader.
- **Anonymity and resistance to tracking attacks.** The protocol should protect against any adversarial action aiming at identifying the tag, and should guarantee that the movements of a tag cannot be traced.
- **Resistance to replay attacks.** The protocol should be immune to attacks in which an adversary collects messages from protocol executions between the reader and tag and sends them again to the parties, in order to subvert some of the security and privacy properties.
- **Forward security.** Even if at a certain point the tag is compromised and the adversary gets the secret information stored in the tag's memory, the past communications should remain unaffected.
- **Resistance to leakage and disclosure attacks.** The protocol should not leak secret information under adversarial actions, and there is no way to get access to the secret information shared between the tag and reader.

Some of the above goals in certain applications should be guaranteed against a *passive* adversary, who just eavesdrops on the protocol executions, while others should hold with respect to an *active* adversary, who can intercept and modify the messages and interact with the parties.

In the next section we elaborate on the security and privacy properties. Indeed, in this area they are almost always expressed in an informal way, and as a *list of* *desiderata* that need to be achieved. No rigorous model is used to state clearly the goals and to prove the merits of a given protocol. The *proofs* are arguments aiming at convincing the reader of the goodness of the design. As shown in [169] in a case-study for the SASI protocol, this approach opens doors to unexpected consequences.

6.3 Weaknesses and Pitfalls

Ultra-lightweight protocols strive to achieve a strong level of both security and privacy while fitting extreme design constraints due to limited space, energy, and cost on RFID tags. Unsurprisingly, attacks on virtually all proposals in the literature have been published.[2]

As a result, no such protocol could reasonably be used in practice.[3] Although some lessons have been learned from these failures, and despite much advocacy for better screening from the research community, protocols repeatedly fall victim to common pitfalls, even in recent proposals. What follows is a short description of the prevailing weaknesses.

6.3.1 Poor Diffusion and Linearity

Many protocols use the so-called “T-functions” extensively. These are functions for which each bit in the output depends only on bits in the same or lower positions in the input. Binary operations (e.g., `and`, `or`, `xor`) and modular addition are T-functions.

By definition, in a T-function it is not possible that all output bits depend on all input bits, which is the ideal scenario for maximizing “diffusion”, an important property in cryptographic primitives. This is particularly dangerous in cryptographic applications, lightweight or otherwise. The only reasonable way to address this shortcoming is by combining these operations with others which do not exhibit this characteristic. Unfortunately many designers do not follow this simple combination rule, and have proposed schemes entirely based on T-functions which are doomed to fail. LMAP [466] is an example of a protocol that uses T-functions exclusively, which was exploited in its cryptanalysis [464].

[2]It has been observed that ultra-lightweight protocols are “broken” with relative ease, very shortly after their publication. Avoine et al. [43] shows a short statistical study and concludes conservatively that most are broken in under 4 months.

[3]To the best of our knowledge, the Gossamer protocol [463] is the sole instance to not have any published attacks, although a number of weaknesses in its construction have been identified [130]. In addition, Gossamer is definitely more involved and, arguably, could hardly be considered “ultra-lightweight”.

Linearity, i.e., the property that $f(a \odot b) = f(a) \odot f(b)$, is another source of trouble. The xor operation, rotations and other permutations are linear. Like T-functions, linearity is transitive (the composition of linear operations is linear), and some schemes have been shown to be entirely linear, which easily leads to attacks. Particularly notable and common examples are the many proposals in which security is based heavily on the use of Cyclic Redundancy Codes (CRCs). CRCs are designed to do channel error correction, but offer very little security if any at all.

6.3.2 Poor Message Composition

Securely designing the messages exchanged over an ultra-lightweight protocol is a difficult open problem. Keeping the secrets exchanged as secure as possible against any leakage is indeed a big challenge, particularly in such constrained environments. Generally speaking, the messages should guarantee good confusion (i.e., key mixing) and diffusion properties. That is, the secret key (or keys) should be thoroughly involved in the construction of the messages, and a subtle change in the secret should result in completely different messages. However, due to the constraints of ultra-lightweight protocols, messages are usually built using a handful of operations, and in many cases good confusion and diffusion levels are not obtained.

In LMAP for instance, the key update phase is defined by:

$$IDS^{(n+1)} = (IDS^{(n)} + (n_2^{(n)} \oplus K_4^{(n)})) \oplus ID,$$

where we can see that ID, a secret that the protocol is designed to protect, is simply xored with a mixture of public and secret values. This operation exhibits poor confusion and diffusion properties. Although exploitation of this varies in different attacks, this quite frequent feature heuristically leads to a major leakage of secret bits, as the rest of the message the ID is combined with may be biased, or be partially known by the adversary.

6.3.3 Biased Output

Another important weakness of many lightweight schemes is that some of the operations are biased, a property that in many cases leads to security vulnerabilities. This is typical of Boolean functions such as or ($\vee$) and and ($\wedge$), where $x \vee y$ and $x \wedge y$ have, for unbiased random bits x and y, heavily (75%) biased outputs, respectively, towards 1 and 0.

This can constitute a security weakness because these functions leak information for both of their arguments. For example, if $x \vee y = 0$, then $x = y = 0$, which discloses both the inputs. With a uniformly distributed input, this happens 25% of the time (similarly for `and`, of course). In some cases it is more than enough, after seeing some exchanges, to be able to completely recover all the inputs.

In LMAP, the reader sends $B = (IDS \vee K_2) + n_1$. The attacker can thus use $B + (2^L - 1)$ as a very good approximation to the unknown "shielded" nonce n_1 (on average, 75% of the bits are correct), and this approximation can be used later in other parts of the protocol to approximate the secret (see e.g. [44] for a full attack partially based on this).

6.3.4 Rotations

Rotations have been used for a long time in cryptography. Many modern block ciphers and hash functions such as BLAKE [37] or RC5 [502] rely on the ARX (addition, rotation, XOR) paradigm. Rotations are extremely cheap to implement in hardware, and they introduce diffusion, which complements nicely the modular addition and the XOR (which exhibit poor diffusion properties). Fixed-amount rotations are typically used in ARX designs, but data-dependent rotations, as first featured in the RC5 block cipher [502], also exist.

The SASI [139] protocol was the first ultra-lightweight authentication protocol to feature data-dependent rotations. Since then, most ultra-lightweight protocols have used them, and in many cases they are the weak spot for ad hoc attacks. In addition to linearity, the most important shortcoming of data-dependent rotations is that there are only L possible outputs. Mod n cryptanalysis [319] is also a promising tool for attacking schemes using rotations and additions, although it has never been applied in the cryptanalysis of an ultra-lightweight protocol, to the best of our knowledge. It has, on the other hand, been used to successfully attack block ciphers such as RC5P and M6, which use the same kinds of operation.

6.3.5 Vulnerability to Knowledge Accumulation

If partial leakage of a static secret occurs in a protocol, there is an obvious traceability issue. Indeed, it becomes possible for an attacker to correlate two leaked traces of an eavesdropped exchange. A typical example is recovering the least significant bit of the static identifier (see for instance [473], the first traceability attack on SASI). More importantly, an attacker is sometimes able to recover the full static secret after a few rounds. Indeed, different observations can be combined using Bayesian inference. An example of such an attack was the full cryptanalysis of SASI [44].

One of the initial goals of synchronized protocols[4] is to provide forward privacy. Forward privacy is a stronger notion than just privacy. Simply put, a protocol is said to be forward private if an attacker, having recovered the internal state (the dynamic values of the identifier and keys) of a tag, is not able to recognize the tag in past interaction traces. For a more formal definition, see [453]. Forward privacy cannot be achieved in a protocol if the secrets used in the exchange are all static. Indeed, if the attacker knows the secrets of a tag at some point, it also knows them in the past, since the secret does not change in the tag's lifetime. Therefore, messages sent by a tag in previous interactions can be recomputed, and recognized easily. Note that a changing secret is required for forward privacy, but it does not guarantee it (indeed, there are many synchronized protocols that are not private, and therefore not forward private).

A positive side effect of changing the secrets is that it might make it harder to obtain the full secret at any given time, if only a partial leakage is obtained at every authentication round. This seems to be a good feature as it is intuitively harder to hit a moving target than a static one. However, this does not necessarily make the full cryptanalysis impossible, just slightly harder, as has been demonstrated with the Tango attacks [273, 468].

6.3.6 Dubious Proofs of Security: Randomness Tests and Automated Provers

In many instances, some degree of security is allegedly claimed by verifying that the exchanged messages look random enough. For that, multiple sessions of the protocol are run and the exchanged messages are recorded and later analyzed using various randomness test batteries such as the well-known ENT [568], Diehard [394] and NIST [509]. Unfortunately this does not prove any security level (for instance, LMAP presented such a "proof" but was broken shortly after publication). Randomness may appear, not as a consequence of a well designed protocol, but simply as a result of employing nonces in message mixing. Randomness is not a sufficient condition, neither is it a necessary one. A trivial way of showing this is by thinking about highly formatted messages and how, even if a protocol is secure, due to formatting and padding of some or all of its messages these may not pass some randomness test.

Another popular but flawed way of proving security of proposed ultra-lightweight protocols is the use of logic modeling and formal protocol verification software.

A notable example is [492]. The scheme was broken in [464], despite being accompanied by a formal security proof in BAN logic. The authors mistakenly

[4]In synchronized protocols the parties, after each execution of the protocol, apply the same updating function to their secret keys and state information.

employed CRC (as recommended by the EPC-C1-G2 standard), but instead of using them as a simple error detection tool, employed them for encryption. In their idealized model, they identified their CRC usage as equivalent to encryption, so some of the BAN logic rules (for example R1: the message-meaning rule) did not hold anymore. This constitutes a common mistake, as an idealized scenario like the one modeled by BAN logic (with perfect, unbreakable and zero-leaking ciphers) never accurately models reality. The level of abstraction needed in the modeling phase basically makes it impractical for most realistic situations. This is, unfortunately, not only a limitation of BAN logic but, to different extents, is also in most formal models (GNY, etc.).

6.4 Towards a Sound Approach

6.4.1 State of the Literature

RFID technology has prompted many interesting challenges in the security and privacy research community, and designing a secure authentication protocol for very low-end tags is definitely one of them.

The field, however, has been the victim of an abundance of papers of dubious quality. Many research results either repeat mistakes (for new schemes) or past achievements (for attacks) or both. Recent protocols, with respect to previous ones, have been enhanced by using more involved transforms of the data stored in the tag's memory. However, the mistakes appear to be repeated: poor design choices, a lack of confusion and diffusion in the transforms, and informal fallacious security analyses to support the security claims [170]. This bad reputation, combined with a decline of interest in RFID security research as a whole, may have scared off many seasoned cryptographers, and contributed to the relative stagnation of the field.

Despite the current situation, which may seem to indicate that ultra-lightweight protocols are bound to fail, there is no clear evidence that designing a secure protocol with such constraints is impossible.

The field may nowadays be inactive, but there are many unanswered questions (and indeed, no practical, concrete, and trusted protocol emerged from it). While it is likely to reappear under a different guise, the problem of designing a secure authentication protocol while minimizing some aspects of its design (e.g., gate count), is not going away, and remains an interesting research question.

6.4.2 Promising Avenues

The need for cryptographic building blocks in low-end systems is definitely not unique to ultra-lightweight authentication protocols. A much larger research community is dedicated to so-called “lightweight cryptography”, with countless applications.

In particular, significant efforts have been made to develop ciphers and hash functions suitable for lightweight authentication. A notable example is the KECCAK hash function [81], winner of the SHA-3 competition, that has excellent hardware performance. Furthermore, there are many ongoing efforts to develop special primitives with a stronger focus towards hardware footprint/performance, possibly trading off “some” security (e.g., reducing 128 or 256-bit security to 80) or other aspects, such as reducing block size, or software performance. Examples include the PRESENT block cipher [101] or the hash functions PHOTON [251] and QUARK [33].

Some of these, like BLAKE [37] or RC5 [502] are so-called Add-Rotate-Xor algorithms, that use the very same set of operations as ultra-lightweight protocols.

While not quite fitting the same extreme constraints imposed on ultra-lightweight protocols just yet, they are a stepping stone in that direction. They also benefit from much wider exposure and scrutiny, which bodes better for their overall security.

Ultra-lightweight protocols take a unique approach in that the entire scheme is designed, for instance, without using cryptographic building blocks as black boxes. It seems instead perhaps more promising to use standard authentication protocols with these lightweight primitives.

6.4.3 The Reductionist Approach

A deeply studied approach to the design of lightweight authentication protocols for RFID tags is the one provided by the $HB+$ protocol [310], which builds on the earlier HB protocol [284], introduced to efficiently authenticate a human to a computer. The security of these protocols is based on the difficulty of solving the *learning parity with noise* (LPN) problem [284]. Subsequently, several variants of $HB+$ have been proposed but almost all of them present some problems, e.g., [231–233, 457]. Unfortunately, according to [26], the HB-like protocols are not suitable for implementation on ultra-constrained devices. However, the identification of hard problems which allow the design of ultra-lightweight authentication protocols is a research direction which should not be abandoned.

Another interesting approach to designing an authentication protocol for RFID tags was proposed in [520]. Therein, a lightweight hash function, which can be used in RFID authentication, was described. The security of such a hash function is related to the security of the Rabin public key scheme. The idea is to compute an excellent numerical approximation for a *short window of bits* in the middle of the ciphertext produced by the Rabin encryption function. The Rabin encryption function uses a modulus of a particular form, in such a way that computing these bits for an adversary is as hard as breaking the full Rabin scheme. A basic version of the scheme was analyzed in [458]. As far as we know, the approach of [520] has not been followed by other significant proposals. We believe that this research line, which aims at reducing the computational burden while maintaining the security of the full protocol, or even in a more realistic way by paying a small loss compared to the full protocol, is worth pursuing in order to get an improvement in the field.

6.5 Conclusions

We have provided a short overview of the field: from our excursus, it seems clear that ultra-lightweight authentication is a challenging task, and that the present solutions are insufficient. The current state of knowledge is overall quite poor.

Perhaps, the first important open problem is to come up with a reasonable model for the class of ultra-lightweight protocols, in order to get an in-depth understanding of the possibilities and limits of these protocols.

Moreover, we note that, while most of the ultra-lightweight authentication protocols are broken, some are *more broken than others*: if one can impersonate a tag after 10^6 eavesdropped sessions, or after 1 such session, the two attacks effectively "break" the protocol in theory, but the question is does in practice the former represent an "acceptable" or "sufficient" level of security in some settings? It is quite unique to have this tradeoff between security and complexity measured, for example but not exclusively in GEs. We should think about this.

Finally, positive, promising avenues, which build on well-known cryptographic approaches and practices, are available. They are a good starting point to get new findings and to realize suitable solutions for the application market.