A Brief Survey of Nonces and Nonce Usage

Geir M. Køien

Faculty of Engineering and Science
University of Agder
Grimstad, Norway
Email: geir.koien@uia.no

***Abstract*—Last year the European Union Agency for Network and Information Security (ENISA) published a report on cryptographic protocols. A main verdict was that we still have not reached maturity for the design and analysis of cryptographic protocols. This is bad news for a society that has become dependent on well-functioning information- and communication technology (ICT) infrastructures. In this paper, we address this by investigating the *nonce*. The nonce, a number-used-once, is but one small element of a cryptographic protocol. That is, a small, but nevertheless a critically important element. Yet, there is relatively little to be found in the literature regarding the properties of the nonce. This is thus an attempt to improve on this, by providing an initial analysis of nonces and classifying types of nonces used in different cryptographical protocols.**

Keywords–Number-used-once; Nonce; Randomness; Freshness; Timeliness; Uniqueness; Non-repeatability; Cryptographic protocol.

I. INTRODUCTION

A. Background

The ENISA report "Study on cryptographic protocols" [1] published in 2014 investigates the state-of-the-art for cryptographic protocols. The conclusion is clear and points out that cryptographic protocols are not well understood:

> Whilst the security of basic cryptographic building blocks, such as primitives and protocols, is well studied and understood, the same cannot be said of cryptographic protocols. The scientific study of such protocols can be said to be still not mature enough.
>
> *– from the executive summary of [1]*

Also, we have the ENISA report "Algorithms, key size and parameters report 2014" [2]. This is an important report, in a yearly series, as it represents the nearest thing to consensus about state-of-the-art in cryptographic algorithms and levels of protection. Here we learn, amongst others, that when using a nonce as an Initialization Vector (IV), there are several traps to fall in. Section 4, "Basic Cryptographic Schemes", goes into detail about concerns with using nonces as IVs for block cipher modes and in hash-based schemes. The worries are mainly about non-randomness and predictability. Yet, the report is surprisingly vague about what a nonce really is:

> Many modes make use of either a nonce or an random IV. A nonce is a number used once, it is a non-repeating value but not necessarily random. Thus a nonce could be a non-repeating sequence number. On the other hand a random IV should be random, and unpredictable to the adversary.
>
> – from Section 4.1 in [2]

Nonces are an important part of many, if not most, cryptographic protocols. They do serve different purposes and the requirements on the nonces are not always explicit, making it hard to determine exactly what properties the nonces must have and to verify that they indeed to have the required qualities.

B. Randomness, Computationally Infeasible, The Birthday Paradox and Collisions

We shall use the terms randomness and pseudo-randomness. Actual bit-field randomness in a nonce information element (IE) is not always necessary or even desirable. For our purpose, we want the term merely to mean that the value of the "random" IE is uniformly distributed over the whole value range and that a priori guessing the value of a random IE is computationally infeasible. That is, even when knowing very large series of prior values, the adversary shall not have gained any practical advantage in guessing the next value.

In this paper, we will use the term *computationally infeasible*. We intend it to mean that it will not, under any circumstance, be practical for an adversary to brute-force guess the random IE. That is, pre-computation attacks of dictionaries and rainbow tables must be utterly impractical and beyond the reach of even the most powerful adversary. This implies that the range of IE values must be so large that pre-computation indeed becomes impractical.

The birthday paradox is well known in cryptography (Section 2.1.5 in [3]). In our context, we shall worry about it's application to collisions. Nonce collisions will, in some cases, be a severe security problem. The collision-free property may therefore be required. It is well known that collision probability for a random draw is roughly $2^{N/2}$, where N is the bit-field size of the random IE. This means that one will need the random IE to have a bit-field size of 2 x N. It seems generally accepted that "128 bit complexity" is enough to provide the *computationally infeasible* property. If collisions are a security problem, then one need a 256 bit field for the IE in question in order to reach the required security level. On the other hand, if collisions are not a worry, then a 128 bit nonce should suffice.

II. BRIEF LITERATURE SURVEY

There is a considerable body of literature concerning security protocols and cryptographic methods available. In this paper, we have focused mostly on investigating nonce usage as described in security protocols. That is, our focus is from a "user's perspective". In this respect we have investigated some classic security papers, some papers concerning formal verification of security protocols and some papers that explicitly mention and discusses nonces.

 ISBN: 978-1-61208-427-5

A. Older Literature

Nonces are a very important part of BAN logic, as reported in in the seminal research reports "A Logic of Authentication" [4]. In [4], there is substantial emphasis on the *freshness* as property of a nonce.

Nonces are also explicitly mentioned in "Prudent Engineering Practice for Cryptographic Protocols" [5]. Here, the authors clearly relate authentication *challenges* with nonces. We here also find a description of what a nonce may be:

- Timestamps
- Serial numbers (which must be *recent* somehow)
- Random numbers

These three examples are also found elsewhere in the literature. In [5], the authors also a provide a principle for nonces:

> Be clear what properties you are assuming about nonces. What may do for ensuring temporal succession may not do for ensuring association – and perhaps association is best established by other means.

We concur with the above principle (`Principle 6`).

The paper "A Survey of Authentication Protocol Literature: Version 1.0" [6] is not about nonces per se, but nonce use is mentioned frequently throughout the survey.

B. Books Covering Nonces

In the book "Handbook of Applied Cryptography" (HAC) [3] we find a fair amount of material on nonces. Here the authors have explicitly highlighted the time-variant properties. They put emphasis on the role nonces play in preventing replay attacks. This is of course in line with the freshness property of [4]. We furthermore note that the authors highlight the following (Section 10.3.1 [3]):

> The term nonce is most often used to refer to a random number in a challenge-response protocol, but the required randomness properties vary. Three main classes of time-variant parameters are discussed in turn below: random numbers, sequence numbers, and timestamps. Often, to ensure protocol security, the integrity of such parameters must be guaranteed (e.g., by cryptographically binding them with other data in a challenge-response sequence). This is particularly true of protocols in which the only requirement of a timevariant parameter is uniqueness, e.g., as provided by a never-repeated sequential counter.

The "time-variant" property seems to be as much about ordering as about time per se. In this sense it is more concerned with uniqueness and freshness than about temporal properties.

In the book "Security Engineering" [7], by Anderson, we find only a few brief passages. This, otherwise quite comprehensive text, mentions that nonces may be random numbers, serial numbers or timestamps, but does not say much more. That is, the problem with synchronizing clocks are mentioned as a drawback to timestamps. Another comprehensive book, "Computer Security: Art and Science" [8], also provides next to nothing on nonces. A brief passage is all, with a few extra sentences for timestamps added to it. The book "Applied Cryptography" [9] is also mostly silent on the topic.

The book "Protocols for Authentication and Key Establishment" [10] does cover nonces and nonce usage. In Chapter 1.5 "Freshness" is handled. The text recognizes *Timestamps*, *Nonces (Random Challenges)* and *Counters* as ways to achieve freshness. This classification is somewhat different from the one found in the HAC or in [5], [7]. The authors also places emphasis on key freshness, which may be assured by use of nonces.

The book "Cryptography Engineering" [11] barely mentions nonces, in conjunction with Initialization Vectors (IVs), and have only a brief mention of timestamps. Here, they highlight what they call the "same-state" problem, which in effect seems to be lack of uniqueness.

C. Some Research Papers Covering Nonces

In the paper "Nonce-Based Symmetric Encryption" [12] the author discusses encryption mode-of-operations using IVs. The IVs in question are nonce based, and here the authors emphasises the uniqueness property. In the "Encode-then-encipher encryption: ..." paper [13], the general assumption is that the nonce is a counter or a random value. Collisions are problematic, and so the collision probability must be kept low.

An analysis of nonces used for authentication in a smart-card context can be found in [14]. The authors seems only to be concerned about "random" nonces, although it is not clear from the text whether they intend this to be random or pseudo-random. Still, it appears to be a trivial application where unpredictability and uniqueness seems the essential characteristics. Freshness is not directly mentioned, but it is probably assumed. Use of nonces in SIP is studied in [15]. The basis is a Diffie-Hellman exchange [16] and the use of random nonces, one for the client and one for the server. The scheme is relatively complicated, and somewhat bewilderingly, there are no other requirement on the nonces than they be "random". We assume this implies freshness and uniqueness.

The lack of preciseness and explicitness as demonstrated by the two latter papers seems to be fairly common, and informal browsing of the literature indicates that this state of affairs are the norm for many conference papers.

D. The Key Wrap Problem and Synthetic IVs

During the late 1990s, the National Institute of Standards and Technology (NIST) posed the so-called "Key Wrap" problem. The essential problem is how to develop secure and efficient cipher-based key encryption algorithms. This call caused quite a lot of research activity and then standardization. Amongst the notable papers are [17], in which the authors discusses use of the so-called *Synthetic IV (SIV)* and how to have misuse-resistant nonce-based authenticated encryption. Use of SIVs are also captured in RFC 5297 [18].

E. Keeping Time in the Face of An Intruder

Maintaining synchronized clocks in a distributed network is no small problem in itself, and it gets worse when one may expect a dishonest party. As highlighted in the Network Time Protocol (NTP) ver. 4 [19], there are serious security considerations for synchronized global clocks. Section 15 in [19] goes into considerable detail about the security implications.

F. Same-State and Time Resolution

Timestamps may suffer from resolution problems too, also known as the the same-state problem. That is, a timestamp nonce is not unique within the time resolution period. The

 ISBN: 978-1-61208-427-5

problem can be solved by combining timestamps and counters. One such scheme is presented in a paper by Mitchell [20], with a hybrid timstamp/counter based nonce. The Mitchell approach was the input to the sequence number scheme used in the UMTS Authentication and Key Agreement (AKA) protocol (Annex C in [21]). To avoid tracking by the sequence numbers, the actual sequence number is masked by an anonymity key in the UMTS AKA protocol.

III. Nonces and Formal Verification

This is just a few examples of the assumptions made by formal verification tools with regard to nonce properties.

A. BAN Logic

The research report "A Logic of Authentication" [4] did in many ways bootstrap the field of formal security protocol analysis. The logic devised in the report, commonly known as the BAN logic, is a belief logic and it is, by today's standard, both incomplete and flawed. However, it is interesting to note that the so-called "freshness formula", is essential to the logic.

1) The Freshness Formula:

$$\#(X) \tag{1}$$

> The formula X is *fresh*, that is, X has not been sent in a message at any time before the current run of the protocol. This is usually true for nonces, that is, expressions generated for the purpose of being fresh. Nonces commonly include a timestamp or a number that is used only once, such as a sequence number.

2) The Nonce-Verification Rule: The rule is concerned with freshness, and if and only if the message is fresh/recent will the receiver *believe* in the message. It is noteworthy that [4]:

> "This is the only postulate that promotes from $|\sim$ to $|\equiv$."

That is, the *nonce-verification* rule is the only rule that promotes *once said* to *believes*. This again means that the freshness property is required in BAN logic in order to successfully prove that the target protocol was successful. We observe that (pseudo-) randomness is not actually mentioned here, but it not excluded either. Timestamps and serial numbers are explicitly mentioned. Formal verification tools for verifying security protocols must necessarily have some way to capture what the nonce is to achieve. In BAN logic [4], it is clear that *freshness* is the main nonce objective.

B. Process Algebra

The book "Modelling and analysis of security protocols" [22] covers formal modelling and analysis in the context of the process algebra "Communication Sequential Processes" (CSP). There is an associated compiler, Casper, and a model checker, the "Failures-Divergence Refinement" (FDR) tool.

The authors discuss nonces and nonce properties. The Station-to-Station (STS) protocol is one of the protocols discussed, and interestingly they note that "*Note how the Diffie-Hellman terms double as nonces, so providing assurances of freshness.*" (Section 0.2 in [22]). This dual role can be problematic, unless explicitly defined and unless the properties matches both roles. Nonces is defined as (Section 0.7 in [22]):

> In the context of security protocols a nonce can informally be taken to be a fresh, random value. It is created as required in a way that is supposed to guarantee that it is unpredictable and unique. Giving precise formal meaning to terms like unpredictable, fresh and unique is itself rather subtle and we will see later how this can be done in a number of frameworks, including our own.

The authors differentiates between nonces, where randomness is involved, and timestamps. They also notes that "*If we are using time-stamps to help maintain the security of the protocols we need to bear in mind ways that an intruder might try to subvert the mechanisms.*". There is no mentioning of counters or serial numbers. That is, the authors briefly mention "identifiers for runs" and this may indeed be a counter.

As a modelling strategy, a nonce must be strictly unique in CSP. This in effect precludes the use of timestamps, and it explains why the authors distinguishes between nonces and timestamps. Also, a nonce can only be used once. This shouldn't be a surprise, but it is quite common to allow a nonce to remain within a protocol sequence as a way of binding the message sequence together (providing association). But, as noted in `Principle 6` in [5], this is a task that perhaps is best solved by other means. There are also schemes to differentiate Initiator (I) and Responder (R) nonces, although this has no real bearing on the nonce properties. Nonces are defined by the $Nonce$ constructor, where $Nonce.n$ defines n as a nonce. It is possible to capture the notion that n should be secret, but this is not a characteristic of n per se.

C. The AVISPA and AVANTSSAR Toolsets

The AVISPA project was funded by the European Union during the early/mid 2000. The acronym AVISPA stands for Automated Validation of Internet Security Protocols and Applications. The project goals was to develop formal modeling techniques for the analysis of security protocols. The formal verification tools rely on an abstract notion of the nonce, where freshness and uniqueness are the main properties [23].

The follow-up AVANTSSAR project provided considerable improvements to the AVISPA formal modelling tools, in particular the somewhat awkward High Level Protocol Specification Language (HLPSL) was substituted with the AVANTSSAR Specification Language (ASLan++) [24], which more closely resembles the familiar Alice-Bob notation. However, the semantics for the nonce is the same, and symptomatically, the way to initialize a nonce in ASlan++ is by assignment with a value from the pre-defined `fresh()` function.

While freshness may appear to be the only aspect captured here, we note that secrecy may additionally be defined as a goal for the nonce. This would captured as a protocol goal rather than as an intrinsic aspect of the nonce.

IV. Nonce Attributes

In the following section we try to capture the attributes we may require of a nonce. It is a synthesis of requirements, explicit and implicit, found in the literature.

A. Base Attributes

We need to define the attributes that a nonce may or may not have in order to properly classify the nonce types. The definitions are given with respect to a nonce of a given bit field size. The bit field size is assumed to be large enough to exceeds any practical computational means for exhaustive guessing or testing with respect to any given property.

 ISBN: 978-1-61208-427-5

B. Uniqueness Properties

We have two uniqueness variants.

Attribute 1. ***Uniqueness***

Uniqueness is the property that that the value of the nonce will never be repeated.

This is another way of expressing the once-only property. It also implies a collision-free property. The uniqueness may be expressed within a given context, and if so, it is prudent practice to make this constraint explicit. The uniqueness property effectively precludes the use of a random function for assigning the nonce value. Pseudo-random numbers will generally satisfy this property, as will sequence numbers. Uniqueness, if it can be demonstrated, is one way to ensure the never-used-before aspect of freshness.

Attribute 2. ***Statistical Uniqueness***

A statistically unique nonce shall be computationally indistinguishable from a unique nonce.

The probability of a collision for a statistically unique nonce must be exceedingly low. It must be computationally infeasible to distinguish between a unique nonce and a statistically unique nonce. If nonces are truly randomly generated, then one *may* experience collisions. However, with uniformly distributed random draws with a very large outcome space, collision will be exceedingly infrequent.

C. Freshness and Recentness

Uniqueness, or statistical uniqueness, does not quite capture timeliness or recentness. This is a problem since in many protocols it is important to demonstrate that all principals, including certificate authorities and authentication centers, are online during the protocol run. Ideally, we want to capture recentness directly in the nonce, but this harder than one might initially envision. Proving that all principal parties are online may therefore best be carried out by application of nonces, rather than being an inherent property of "fresh" nonces. We therefore exclude recentness as an inherent property of freshness.

Attribute 3. ***Freshness***

Freshness is the attribute that a certain nonce value has never been used before in the given context.

Freshness is an essential property and it is likewise essential that the freshness is verifiable by the receiver. If freshness verification, for the receiver, is not implicit in the scheme, we prefer to label it as *weak freshness*. Then, to attain *verifiable freshness* may require additional steps in the protocol itself. This is acceptable, provided that the verification is indeed carried out.

D. Unpredictable Nonces

Uniqueness and freshness are fundamental properties, but there are cases where they are insufficient or unnecessary. For instance, unpredictability is not intrinsic to uniqueness or freshness. Nonces that only require uniqueness and freshness may be produced by a sequential number scheme, but they will certainly not be unpredictable.

For some cases, one must explicitly require that the nonce is unpredictable. That is, the nonce must appear to be randomly drawn. We generally assume, but do not require, that the randomness is pseudo-randomness. The pseudo-random number generator ($prng()$) function must have uniform distribution over the whole outcome space and it must be computationally infeasible to guess the next value even when given very long series of previous values. The quality of the $prng()$ function and security associated with the initialization of the function is essential. In [25], we find many examples of security being compromised due to what the authors call "bad randomness". We define unpredictable here as compliant with our statistical uniqueness requirement, and hence it is really a "statistically unpredictable" attribute.

Attribute 4. ***Unpredictable***

This is the property that predicting the value of the next nonce must be computationally infeasible. This must hold even if the adversary have observed a large number of previous nonces and the associated contexts.

E. Predictable Nonces

Freshness can be assured without randomness, and timestamps and counters are obvious solutions. Encoding-wise, a timestamp may be seen as a special case of a counter. The counters must be monotonically increasing, although discrete continuity seems not to be required. This allows for timestamps to be interpreted as counters. It is also possible to abandon the requirement for monotonicity, but then obviously one can no longer guarantee sequential order. Schemes exits where a window mechanism is used (See SEQ use in UMTS [21]), but this opens up to the risk of replays and one looses the possibility to guarantee that the principals are all online during the transaction. Use of window mechanisms may be permissible under some circumstance, but we generally argue against it.

Counter wrap-around is a potential hazard for sequentially ordered nonces. We mention this explicitly since it is not uncommon to use protocol frame numbers and similar as "nonce-like" inputs, and these counters will generally be allowed to wrap-around. Special care must be taken for cases where this may occur.

Attribute 5. ***Sequentially ordered***

This is the property that the nonce is a counter with monotonically increasing values.

Since sequential order alone cannot provide timeliness, we have added timeliness as a separate attribute.

Attribute 6. ***Timeliness***

Timeliness is the property that the nonce is encoded as a unique timestamp. The clock source(s) must be trusted and clock synchronization must be protected.

We note that the collision-free property is satisfied for the predictable nonces. However, be warned that counter wrap-around and period resolution problems may re-introduce the same-state problem, and thereby the potential for collisions.

F. Secrecy and Authenticity

Privacy is a concern and sequential nonces may be used for tracking purposes. To avoid this one may require the nonce to be encrypted (data confidentiality).

 ISBN: 978-1-61208-427-5

Attribute 7. ***Secrecy***

Secrecy is defined to be data confidentiality for the nonce.

In some protocols, there is also a need for verified nonces. We believe that this should be explicitly captured, and hence that a integrity/authenticy attribute is needed.

Attribute 8. ***Authenticity***

Authenticity is defined to be data integrity for the nonce.

Authenticity is here equivalent to the *once said* property of BAN logic.

G. Basic Nonce Types

We now define a few basic nonce types. These will be defined in the context of the base attributes. We have qualified the freshness attribute with *weak* and *verifiable*, depending on whether the responder is directly able to verify the freshness.

Definition 1. *Random Nonce*

- *Statistical Uniqueness*
- *Weak Freshness*
- *Unpredictable*

Definition 2. *Pseudo-random Nonce*

- *(Guaranteed) Uniqueness*
- *Weak Freshness*
- *Unpredictable*

Definition 3. *Sequential Nonce*

- *(Guaranteed) Uniqueness*
- *Sequentially ordered*
- *Freshness*

Definition 4. *Timestamp Nonce*

- *Uniqueness (with resolution constraints)*
- *Verifiable Freshness*
- *Sequentially ordered*
- *Timeliness*

H. Complex Nonce Types

A *Timestamp Nonce* may suffer from the same-state problem. It may be mitigated with a high-resolution timestamp, but this introduces its own problems. For strict real-time environments this may be irrelevant. Alternatively, one may relax the timeliness guarantee to be within a slightly longer period, while maintaining the requirement for strict sequential order. That is, one may augment a timestamp nonce with a serial number to alleviate the same-state problem while avoiding a high-resolution timestamp.

Definition 5. *Augmented Timestamp Nonce*

- *(Guaranteed) Uniqueness*
- *Verifiable Freshness*
- *Sequentially ordered*
- *(relaxed) Timeliness*

We define a *Encoded Nonce* to be a sequential, timestamp or augmented timestamp nonce that has been "encoded". With encoded we here mean that the target nonce has been transformed by some cryptographic function such that the output will appear to be random and unpredictable to an external observer. However, to the initiator/responder, it is possible to reverse the encoding and retrieve the original sequential nonce. One such scheme is mentioned in Section V-A. Note that such schemes may be convoluted, and that decoding may potentially only be possible in the later phases of the protocol. One may also envisage encoded nonces that are based on random nonces, but we have not found reason to investigate this further.

Definition 6. *Encoded Nonce*

- *(Guaranteed) Uniqueness*
- *Freshness*
- *Sequentially ordered*
- *Unpredictable*
- *Optionally: Timeliness*

I. Nonce Qualifiers

The above nonce types, which should be well in line with suggestions in the literature, may optionally be augmented by what we call the nonce qualifiers. The use of qualifiers should be based on requirements on the nonce in the protocol. The previously defined basic nonce types may then be qualified as:

- **Secret** – possessing the secrecy attribute
- **Authentic** – possessing the authenticity attribute

Note that an *Encoded Nonce* does not necessarily achieve secrecy, and cannot be used in this way unless the required properties are explicitly confirmed.

V. Some Nonce-based Information Elements

A. Initialization Vector

An IV is a value used as the initial input to a cipher function mode-of-operation. For instance, in *cipher block chaining* (CBC) mode, one uses the ciphertext output of the previous stage as input to be mixed (XOR'ed) with the plaintext block. That is, the IV is the input to the initial stage of a CBC chain, where there is no previous output available.

In Section 4.3 in [11], the authors describes IV use in CBC and how to choose the IV. Specifically, they argue that one may use a "Nonce-Generated IV". The specific scheme proposed involves using a message number, which needs to be unique within its context, and then encrypting the message number to produce the nonce. The nonce is then used as the IV. However, message numbers are generally predictable and visible to the intruder, and so one must be careful not to allow the intruder to use the message number as a source for known plaintext attacks. We note that a counter/serial number that is encrypted and then used as an IV, would be a kin to a *Encoded Nonce*.

B. Random Challenge

By definition a normal random challenge information element may be a *Pseudo-random Nonce*. Potentially, it may also be *Random Nonce*, but this would be the exception. Depending on the protocol, it may be desirable to have an *Authentic Pseudo-random Nonce*.

 ISBN: 978-1-61208-427-5

C. Key Agreement

It is quite common to have a key derivation function, $kdf(\cdot)$, that accepts nonces as inputs. One example is the $f4()$ function in UMTS [26].

$$f4_K(RAND) \rightarrow IK \tag{2}$$

In (2), the random challenge is accepted as input and the integrity key, IK, is the output. The key, K, is the permanent pre-shared subscriber symmetric-key credential. In this case the $RAND$ is equivalent to an *Pseudo-random Nonce*.

D. Pseudonymous Identifiers

Privacy is of growing concern, and *identity privacy* is a pronounced concern for authentication protocols, where corroboration of a claimed identity (or identifier) is a primary goal (Chapter 4 in [27]).

With symmetric-key schemes one has no option but to present the claimed identifer in plaintext form. This invariably exposes the identifier and identity privacy is lost. For a mobile subscriber, *location privacy* is simultaneously lost. One way to solve the problem is to use asymmetric cryptography and hide the presented identifer so that only the intended recipient is able to decrypt the message. It is quite common with such protocols to also forward (or agree on) a temporary identifier, to be used subsequently [21], [28]. This temporary identifier must then be free of any apparent association with the primary identifier. That is, to any external observer there must not appear to be any correlation between the primary identifier and the temporary identifier. This, of course, can only be done if the temporary identifier appear to be random with respect to the primary identifier.

The temporary identifier must be unique, or at least statistically unique, within the given context. A *Pseudo-random Nonce* will nicely fit this description.

VI. An Example of Nonce Usage

We have chosen to use the UMTS AKA protocol as an example protocol. It should be a relevant example, as it is both a fairly simple protocol and it is a widely used protocol.

A. Case Study: The UMTS AKA Protocol

The UMTS AKA protocol includes a random challenge and a sequence number scheme. The random challenge is clearly a nonce and the sequence number, the SQN field, fits the *number used once* requirement. The primary reference for the UMTS AKA protocol are the 3GPP technical specifications TS 33.102 [21] and TS 33.105 [26]. See also [29] for an overview.

The UMTS AKA protocol is a 3-way protocol with the following principal entities:

- **User Equipment (UE)** – represents the subscriber.
- **Visited Network (VN)** – the local access network.
- **Home Network (HN)** – UE subscription point.

A central component of the protocol is the Authentication Vector (AV):

$$AV = \{RAND, XRES, CK, IK, AUTN\} \tag{3}$$

The $RAND$ is the pseudo-random challenge (128 bit), $XRES$ is the expected response (64 bit), CK and IK are the session keys (128 bit each) and the $AUTN$ is the authentication token (128 bit) compound IE. $XRES$ and RES are identical, but that the former is the expected response while the latter is the actual response (as computed by the USIM). The Authentication Token ($AUTN$) is defined as:

$$AUTN = \{SQN \oplus AK, AMF, MAC{-}A\} \tag{4}$$

Here AK is an anonymity key, AMF is the authentication management field and $MAC{-}A$ is the cryptographic check sum for the challenge.

The Protocol:

1) VN $\rightarrow$ HN: SendAuthInfo-req($IMSI$)
2) HN: Generate AV
3) HN $\rightarrow$ VN: SendAuthInfo-resp(AV)
4) VN $\rightarrow$ UE: AuthReq($RAND, AUTN$)
5) UE: Compute $MAC{-}A, RES, CK/IK$ and AK
6) UE: Verify validity of challenge
7) UE: Verify timeliness of challenge
8) UE $\rightarrow$ VN: AuthResp(RES)
9) VN: Verify that $RES = XRES$

The challenge is verified by checking $MAC{-}A$. The timeliness is verified based on the sequence number. The SEQ may be seen as a *Secret Sequential nonce*. The secrecy is provided by the used of the anonymity key (AK).

The $RAND$, in conjunction with the associated $AUTN$, may be seen as an *Authentic Pseudo-random Nonce*.

VII. Discussion and Analysis

A. Being Explicit

We note that Principle #6 in [5], which said "*Be clear what properties you are assuming about nonces.*", tend to be ignored. That is, the requirements on the nonce may have been clear to the designer(s) of the protocol, but the requirements must be made explicit and well documented. This will not only remind the designers about what assumptions there are on the properties of the nonce, but it will also be essential in order to carry out formal verification of the protocol. And, clearly, it is vital for the implementors that all assumptions are explicit.

B. Multiple Usages

As exemplified by the UMTS AKA protocol, the random challenge nonce is used for different purposes simultaneously, including entity authentication and key derivation. This may be permissible, but then certainly these nonce requirements must be explicit. It is also not clear that it is advisable to use a nonce for different purpose, even if the nonce properties does match the nonce usage well. That is, this may be detrimental to security in much the same manner as using a shared secret key for different algorithms may lead to added vulnerability.

C. Formal Verification and Type Systems

In [30], [31] there are deep and broad background coverage of type theory and type systems in computer languages. The last part is quite relevant, as formal modelling of security protocols is captured in a modelling language. To enhance a modelling language to capture more aspects of nonces and perhaps even to have a set of different nonce types would clearly be useful. The properties can then at least be checked for consistency and one may even prove that certain properties are adhered to (or not).

 ISBN: 978-1-61208-427-5

However, as noted by Gollmann in [32], proving a protocol correct may not be the most important aspect (Gollmann even declares this to be a non-goal). In fact, the most important aspect may indeed be to adhere to `Principle 6` [5]. This would make it easier to define exactly what the protocol is to achieve in the first place, and it would make it easier to construct the protocol. In [33] this is recognized, and the authors does see formal verification tools in the context as a design aid. Correctness is one thing, but improved clarity and less ambiguity in the description may also lead to more reliable and robust implementations.

We agree with this view and would welcome further research in capturing nonce properties in type systems for use with formal verification tools. This would represent one step in the right direction for designing better and more secure protocols.

VIII. Summary and Concluding Remarks

In this paper, we have made a brief survey of how nonces are discussed in the literature. The survey is by no means exhaustive, but then even our selected sources demonstrates quite well that description of nonce properties in cryptographic protocols often are vague or even missing. And, certainly, the descriptions are almost invariably incomplete.

We have not strived for completeness, or even rigorousness, but we have attempted to further the field by providing improved nonce attribute characteristics and better and more useful nonce definition. However, more work is needed here and we intend to extend our study of nonce usage by investigating more cryptographic protocols. We will also continue our work with type systems for nonces and investigate ways to verify the properties.

Ultimately, we believe greater awareness about nonce properties should lead to less ambiguity and thereby better designs for security protocols. Clarity in what a security protocol should achieve is obviously essential and a goal in itself. Protocol designs may certainly benefit from being captured in a formal languages, not only because it may allow for formal verification of selected properties, but also because increased awareness and preciseness ultimately may lead to better implementation too. In the end then, we may have hope for better, more reliable and robust security protocols.

References

[1] N. P. Smart, V. Rijmen, M. Stam, B. Warinschi, and G. Watson, "Study on cryptographic protocols," ENISA, Report TP-06-14-085-EN-N, 11 2014.

[2] N. P. Smart *et al.*, "Algorithms, key sizes and parameters report - 2014," ENISA, Report TP-05-14-084-EN-N, 11 2014.

[3] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone, *Handbook of applied cryptography; Fifth Printing (August 2001)*. CRC press, 2001.

[4] M. Burrows, M. Abadi, and R. Needham, "A logic of authentication," DEC System Research Center, Research Report 39, 2 1990.

[5] M. Abadi and R. Needham, "Prudent engineering practice for cryptographic protocols," DEC System Research Center, Research Report 125, 6 1994.

[6] J. Clark and J. Jacob, "A survey of authentication protocol literature: Version 1.0," 1997.

[7] R. Anderson, *Security engineering*. John Wiley & Sons, 2008.

[8] M. Bishop, *Computer Security: Art and Science*. Addison-Wesley, 2003.

[9] B. Schneier, *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, 2nd ed. john wiley & sons, 1996.

[10] C. Boyd and A. Mathuria, *Protocols for Authentication and Key Establishment*, 1st ed. Springer Science & Business Media, 2003.

[11] N. Ferguson, B. Schneier, and T. Kohno, *Cryptography Engineering: Design Principles and Practical Applications: Design Principles and Practical Applications*. John Wiley & Sons, 2011.

[12] P. Rogaway, "Nonce-based symmetric encryption," in *Fast Software Encryption*. Springer, 2004, pp. 348–358.

[13] M. Bellare and P. Rogaway, "Encode-then-encipher encryption: How to exploit nonces or redundancy in plaintexts for efficient cryptography," in *Advances in Cryptology (ASIACRYPT 2000)*. Springer, 2000, pp. 317–330.

[14] N. Junghyun, K. Seungjoo, P. Sangjoon, and W. Dongho, "Security analysis of a nonce-based user authentication scheme using smart cards," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 90, no. 1, pp. 299–302, 2007.

[15] J. L. Tsai, "Efficient nonce-based authentication scheme for session initiation protocol," *International Journal of Network Security*, vol. 9, no. 1, pp. 12–16, 2009.

[16] W. Diffie and M. E. Hellman, "New directions in cryptography," *Information Theory, IEEE Transactions on*, vol. 22, no. 6, pp. 644–654, 1976.

[17] P. Rogaway and T. Shrimpton, "A provable-security treatment of the key-wrap problem," in *Advances in Cryptology - EUROCRYPT 2006*, ser. Lecture Notes in Computer Science, S. Vaudenay, Ed. Springer Berlin Heidelberg, 2006, vol. 4004, pp. 373–390.

[18] D. Harkins, "Synthetic Initialization Vector (SIV) Authenticated Encryption Using the Advanced Encryption Standard (AES)," IETF, RFC 5297, 10 2008. [Online]. Available: https://tools.ietf.org/html/rfc5297

[19] D. Mills, J. Martin, J. burbank, and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification," IETF, RFC 5905, 06 2010. [Online]. Available: https://tools.ietf.org/html/rfc5905

[20] C. J. Mitchell, "Making serial number based authentication robust against loss of state," *ACM SIGOPS Operating Systems Review*, vol. 34, no. 3, pp. 56–59, 2000.

[21] 3GPP, TS 33.102, "3G Security; Security architecture," 3GPP, France, TS 33.102 (3G), 2014.

[22] P. Ryan, S. A. Schneider, M. Goldsmith, G. Lowe, and B. Roscoe, *The Modelling and Analysis of Security Protocols: the CSP Approach*. Addison-Wesley Professional, 2001.

[23] AVISPA project, "Automated Validation of Internet Security Protocols and Applications (AVISPA); AVISPA v1.1 User Manual," AVISPA IST-2001-39252, Tech. Rep., 06 2006.

[24] AVANTSSAR project, "Aslan++ specification and tutorial," FP7-ICT-2007-1, Deliverable 2.3 (update), 01 2008.

[25] B. Schneier, M. Fredrikson, T. Kohno, and T. Ristenpart, "Surreptitiously weakening cryptographic systems," Cryptology ePrint Archive, Report 2015/097, 2015.

[26] 3GPP, TS 33.105, "3G Security; Cryptographic algorithm requirements," 3GPP, France, TS 33.105 (3G), 2014.

[27] G. Danezis *et al.*, "Privacy and data protection by design from policy to engineering," ENISA, Report TP-05-14-111-EN-N, 12 2014.

[28] G. M. Køien, "Privacy enhanced cellular access security," in *Proceedings of the 4th ACM workshop on Wireless Security*. ACM, 2005, pp. 57–66.

[29] ——, "An introduction to access security in UMTS," *Wireless Communications, IEEE*, vol. 11, no. 1, pp. 8–18, Feb 2004.

[30] B. C. Pierce, *Types and programming languages*. MIT press, 2002.

[31] L. Cardelli, *Computer Science Handbook*, 2nd ed. CRC press, 6 2004, ch. Type Systems.

[32] D. Gollmann, "Analysing security protocols," in *Formal Aspects of Security*. Springer, 2003, pp. 71–80.

[33] J. A. Clark and J. L. Jacob, "Protocols are programs too: the meta-heuristic search for security protocols," *Information and Software Technology*, vol. 43, no. 14, pp. 891–904, 2001.

 ISBN: 978-1-61208-427-5