Reviewing ISO/IEC Standard for Time-stamping Services

Long Meng, Liqun Chen

***Abstract*—Time-stamping services are used to prove that a data item existed at a given point in time. This proof is represented by a time-stamp token that is created by a time-stamping authority. ISO/IEC 18014 specifies time-stamping services and requires them holding the following two properties: (1) The data being time-stamped is not disclosed to the time-stamping authority, hash values of the data are provided to the authority instead. (2) A time-stamp token can be renewed, as a result the validity duration of a time-stamp token is not restricted by the lifetimes of underlying algorithms or policies. In this paper, we review this standard and discover several issues: Due to inconsistent writing or information missing, a time-stamping service, following the standard specification, may not be able to achieve these designed properties. We provide a solution to each issue.**

***Index Terms*—Time-stamping, Time-stamping standards, Data nondisclosure, Data integrity, Long-term security**

Introduction

Time-stamping services provide time-stamp tokens on data items to prove that the data existed at certain points in time. A time-stamping service involves a time-stamp requester, a time-stamping authority and a time-stamp verifier. The authority creates a time-stamp token for a data item after the requester asks for, and the token can be verified by the verifier. Time-stamp tokens are created by binding the data item and time together using cryptographic algorithms, such as digital signatures and hash functions. A time-stamp token is valid when this binding between the data and time is true and can be verified. This security property is referred to as *data integrity* of a time-stamp token. For privacy reasons, the requester may not reveal the data being time-stamped to the authority. This security property is referred to as *data nondisclosure* of a time-stamp token and it is also achieved by using cryptographic algorithms, such as hash functions.

It is well-known that any commonly used cryptographic algorithm has a limited lifetime due to its operational lifecycle and the increasing computational power of attackers [1]. Time-stamp tokens only hold the property of data nondisclosure or data integrity as long as the underlying cryptographic algorithms remain secure. For the purpose of this paper, if a security property, data integrity or data nondisclosure, of a time-stamp token relies on the continuing validity of the underlying cryptographic algorithms that are used to generate the token, we say that this time-stamp token holds this property in *short-term*; otherwise, we say that it holds this property in *long-term*.

Long Meng and Liqun Chen are with the Department of Computer Science, the University of Surrey (e-mail: lm00810@surrey.ac.uk; liqun.chen@surrey.ac.uk).

Several standards bodies have specified time-stamping services, including the US National Institute of Standards and Technology (NIST) [2], the Internet Engineering Task Force (IETF) [3], the European Telecommunications Standards Institute (ETSI) [4], [5], the American National Standards Institute (ANSI) [6] and the International Organization of Standardization and International Electrotechnical Commission (ISO/IEC) [7]–[10]. In those standardized time-stamping services, the property of data nondisclosure may or may not be required. If it is required, a simple solution is used, in which instead of sending the actual data being time-stamped to the time-stamping authority, the requester sends hash values of the data. Obviously, with this solution, the data nondisclosure property can only hold when the underlying hash function achieves data confidentiality, therefore, it is short-term data nondisclosure. The property of data integrity is mandatory and this property may be achieved in a short-term or a long-term.

In order to achieve long-term data integrity, time-stamp tokens need to be regularly renewed. When an underlying cryptographic algorithm used to generate a token becomes weak, the *time-stamp token renewal* process will replace this algorithm with a stronger cryptographic algorithm. The renewal process needs to carry out before the weak algorithm is broken. We now give a brief overview of how these time-stamping standards mentioned before support the properties of data nondisclosure and data integrity.

- The NIST standard specifies a signature-based time-stamping application for proving time evidences of digital signatures [2], in which the data nondisclosure requirement and time-stamp token renewal are outside the scope of the standard.
- The IETF standard specifies signature-based time-stamping protocols [3]. This standard supports data nondisclosure, which is based on hash functions. The time-stamp token renewal is mentioned but not specified in detail, so the IETF standard holds short-term data nondisclosure and expects to achieve long-term data integrity.
- The ETSI standard specifies policy and security requirements for issuing time-stamps [4], and defines what a time-stamp requester and a time-stamping authority should support [5]. In both documents the time-stamping protocols follow the IETF standard [3], in which the data nondisclosure is protected by hash functions and the time-stamp token renewal is not in their scopes.
- The ANSI standard specifies time-stamping services [6] and includes the specification of mechanisms for both

data nondisclosure and time-stamp token renewal. It also makes use of hash functions for data nondisclosure. Three applications of time-stamps are specified: (1) time-stamp tokens generated on the hash values of an actual data item, which is able to achieve long-term data integrity and short-term data nondisclosure; (2) time-stamp tokens generated on the hash values of a signature of a data item, which is able to achieve short-term data integrity and data nondisclosure; (3) time-stamp tokens generated on the hash values of a (data, signature) pair, which is able to achieve long-term data integrity and short-term data nondisclosure.

- The ISO/IEC standard [7]–[10] specifies time-stamping services, which supports both data nondisclosure and time-stamp token renewal. The same as the IETF and ANSI standards, for data nondisclosure, the ISO/IEC standard requires only revealing the hash values of the data being time-stamped to the time-stamping authority, so it holds short-term data nondisclosure. This standard aims to achieve long-term data integrity by using time-stamp token renewal.

In the remaining part of this paper, we will first briefly introduce more related works in this research field and then focus on reviewing the ISO/IEC 18014 standard [7]–[10] in more details. The major contribution of this paper is that we will report our discovery of several issues: Due to inconsistent writing or information missing, a time-stamping service, following the standard specification, may not be able to achieve these designed properties. We will suggest how to improve the standard in order to remove these issues.

Related works

In 1990 [11], Haber and Stornetta introduced the first concept of digital time-stamping with two techniques: linear linking and random witness. In this paper, they also proposed a solution for time-stamp token renewal, in which a time-stamp token could be renewed by time-stamping the token with a new implementation before the old implementation is compromised.

In 1993 [12], Bayer, Haber and Stornetta proposed another time-stamping technique: publish linked trees into a widely visible medium (e.g. newspapers). Besides, they spotted that the renewal idea in the 1990 paper [11] is insufficient to time-stamp a digital certificate alone (without the original data being certified). They proposed a corrected renewal solution: time-stamping a (data, signature) pair or a (data, time-stamp) pair to extend the signature or time-stamp's lifetime.

As we have mentioned in the previous section, several standards have specified time-stamping services, including NIST [2], IETF [3], ETSI [4], [5], ANSI [6] and ISO/IEC [7]–[10]. The major technologies of these standards follow the suggestions of [12].

The technique of time-stamping renewal in [12] has been extended into several long-term integrity schemes. However, the security of these schemes were not given until 2016, Geihs et al. formalised the property of long-term data integrity in time-stamping services. Their works were reported in two separate papers, focusing on a signature-based long-term integrity scheme based on time-stamping techniques [13] and a long-term hash-based time-stamping scheme [14], respectively. These two schemes provide substantial frameworks for analysing the long-term data integrity property in time-stamping services.

An Overview of the ISO/IEC Standardised Time-stamping Services

The ISO/IEC 18014 standard specifies time-stamping services in four parts: the framework in Part 1 [7], mechanisms producing independent tokens in Part 2 [8], mechanisms producing linked tokens in Part 3 [9], and traceability of time sources in Part 4 [10].

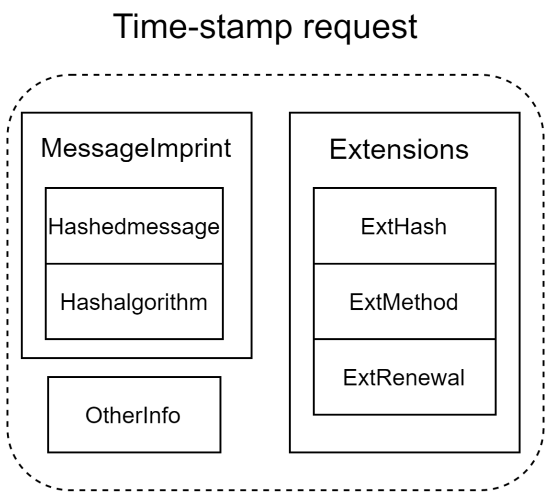
A time-stamping service specified in ISO/IEC 18014 makes use of a protection mechanism and generates one of the following two types of time-stamp tokens:

1) *Independent tokens*: An independent time-stamp token can be verified without involving other time-stamp tokens. The protection mechanism used to generate this type of tokens can be digital signatures, message authentication codes or archives. For example, for signature-based time-stamping, a Time-Stamping Authority (TSA) digitally signs a data item and a time value that results a cryptographic binding between the data and time. The data, time and the corresponding signature together form a time-stamp token.
2) *Linked tokens*: A linked time-stamp token is associated with other time-stamp tokens produced by the same methods. The protection mechanism used to generate this type of tokens can be hash functions and a public repository, therefore a time-stamping service generating this type of tokens is referred to "hash-based time-stamping" or "repository-based time-stamping". In specific, a TSA hashes a data item and a time value together and aggregates the hash value with other data items produced at the same time, (e.g., uses a Merkle Tree [15]). The aggregation result can be linked to other data produced at previous times, (e.g., uses linear chain linking [11]). Eventually, the aggregation or linking result is published at a widely visible media (e.g. newspapers). The data, time record, published information and group values, which are contributed to determine the published information, together form a time-stamp token.

In a time-stamping service, the following two *time-stamping transactions* are performed between a requester and one or more TSAs, or between a requester and a verifier, respectively:

1) Time-stamp request transaction: A requester sends a *time-stamp request* to a TSA and the TSA returns a *time-stamp response* to the requester.
2) Time-stamp verification transaction: A requester sends a *verification request* to a verifier and the verifier returns a *verification response* to the requester.

The data formats of a time-stamp request and response, as shown in Figure 1, are specified in ISO/IEC 18014-1 [7]. *A time-stamp request* contains a "messageImprint" field, which is



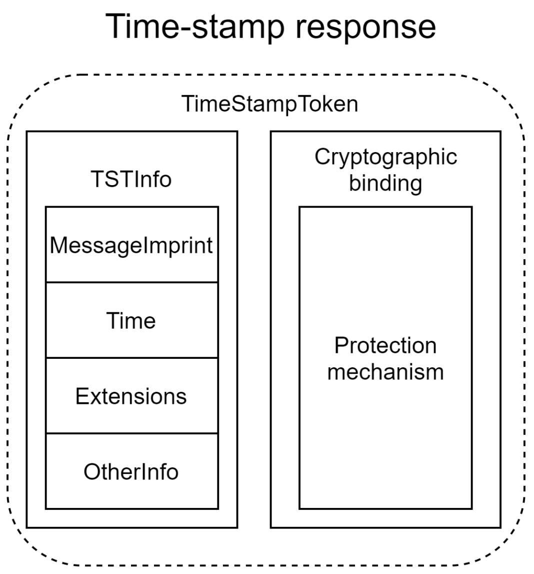


Fig. 1: Data formats of time-stamp request and time-stamp response

comprised of a hash value of a data item and its hash function identifier, an "extensions" field and other information.

More specifically, the "extensions" field contains three types of additional information: ExtHash, ExtMethod and ExtRenewal, which work as follows:

1) ExtHash: In this field, a requester could submit multiple "messageImprint" fields, in which each hash value could be computed from a different hash function so that it prevents the failure of any single hash function.
2) ExtMethod: In this field, a requester could indicate a specific protection mechanism (e.g., a digital signature scheme) to bind the data item and time.
3) ExtRenewal: In this field, a requester could submit an existing time-stamp token on the data item in the purpose of extending the validity period of the time-stamp token.

After the TSA receives the request, it adds the current time to the request content to form a "TSTInfo" structure, and produces a cryptographic binding on the TSTInfo by using the indicated protection mechanism or a default one if it is not indicated. The TSTInfo and the cryptographic binding together form a time-stamp token, then the TSA returns a *time-stamp response* with the time-stamp token to the requester.

In order to validate the time-stamp token, the requester could send a *verification request* that contains the time-stamp token to a verifier at time t_v. For a single time-stamp token that has not been renewed, the verifier checks the following:

- The token is syntactically well-formed.
- Every hash value of the data item is correctly computed through the corresponding hash function.
- At least one of the hash function that is used to generated hash values of the data item is collision resistant at t_v.
- The protection mechanism of the time-stamp token is not broken at t_v.
- The cryptographic binding is correctly computed on the data and time.

If all above conditions are held, the time-stamp token is valid at time t_v, so the verifier returns a *verification response* with a "true" result to the requester, or a "false" result otherwise.

For a renewed time-stamp token, the verifier checks the validity of each nested time-stamp token at the time it was generated or renewed, and validity of the latest time-stamp token at t_v following the above checking steps. The verifier returns a *verification response* with a "true" result to the requester if all verifications are successful, or a "false" otherwise.

Issues in ISO/IEC 18014

In this section, we discuss five issues that we have found in the ISO/IEC 18014 [7]–[10] standard for time-stamping services.

Inconsistency of data format

The data format for a data item being time-stamped is addressed inconsistently in different locations of this standard. The details are given in the following sources.

Source 1: in Section 4 of ISO/IEC 18014-1 [7], symbols and abbreviated terms:

- "D: data to be time-stamped."
- "TS$(x_1,\ x_2,\ ...,\ x_n)$: generation of time-stamp token for the data $x_1,\ x_2,\ ...,\ x_n$."

Source 2: in Section 5.1 of the same document, background and summary:

- "Data shall be provided in a way that it is not disclosed."
- "The time-stamping methods specified in this standard solve these requirements by time-stamping the hash value of the data, which allows for the control of integrity and nondisclosure. The data themselves are not exposed."

Source 3: in Section 5.7 of the same document, time-stamp renewal:

- "Let data D be time-stamped at time T_0
  TS$(D,\ \text{(other info)},\ T_0)$
  At time T_1, while the time-stamp is trusted, a renewal such as

$\text{TS}(D,\ (\text{TS}(D,\ (\text{other info}),\ T_0),\ \text{other info}),\ T_1)$ still proves the existence of D at T_0, given that the first time-stamp is valid at T_1. "

Source 4: in Section 7.1 of the same document, Time-stamp request:

- "Type MessageImprint is used to encapsulate the message imprint data along with an indicator of the algorithm used to generated the message imprint.
```
    MessageImprint ::= SEQUENCE {
        hashAlgorithm DigestAlgorithmIdentifier,
        hashedMessage OCTET STRING
    } "
```
- "hashAlgorithm: hash algorithm identifier and parameter value."
- "hashedMessage: the corresponding hash value of a message to be time-stamped, as calculated with the hash-function specified in the hashAlgorithm data field. "

Discussion: Source 2 indicates that the standard requires that the data to be time-stamped is not disclosed to the time-stamping authority and it should be hashed into a hash value before being submitted to the authority for time-stamping. Source 4 provides the implementation, in which the format of the data to be time-stamped is defined in "MessageImprint" and it only comprises of a hash value of a message and a hash algorithm identifier, the actual data is not contained. However, in Source 3, the data to be time-stamped, D, is directly presented as input to the generation of a time-stamp token. This data format is addressed in both the original token generation at time T_0 and the token renewal at time T_1. Clearly, Source 3 is inconsistent with Sources 2 and 4. Following Source 3, the data nondisclosure property cannot be achieved.

Proposed Solution: We would like to suggest replacing Source 3 with the following text:
"Let D denote data to be time-stamped and H_0 be a hash function or a set of hash functions at time T_0. A time-stamp token is generated by

$$\text{TS}(\text{H}_0(D),\ (\text{other info}),\ T_0).$$

Let H_1 be a hash function or a set of hash functions at time T_1, while the time-stamp token generated at time T_0 is trusted. A time-stamp token renewal at time T_1, which is generated by

$$\text{TS}(\text{H}_1(D),\ (\text{TS}(\text{H}_0(D),\ (\text{other info}),\ T_0),\ \text{other info}),\ T_1),$$

still proves the existence of D at T_0, given that the first time-stamp is valid at T_1."

Short-term Data Integrity for Signed Data

ISO/IEC 18014 aims to provide long-term data integrity to time-stamping services. In Section 5.4 (use of time-stamps) of ISO/IEC 18014-1 [7], three use cases for time-stamping a requester's signature on a data item are specified. We observe that by following the specification, every case can only achieve short-term data integrity. To achieve long-term data integrity, it is necessary to use carefully designed renewal processes, but this information is missing in the document. To discuss this issue, let us quote the following source from that section:

"Time-stamps also play an important role for the validity of signed documents. There exist three possibilities for the time at which time-stamping and signing of data may occur. Data may be time-stamped before the requester of the time-stamp signs it, after the provision of the signature of the document's sender, and before and after the signature. This leads to different results when examining the timely validity of the signature."

These three use cases are explained in the same section as follows:

- Case 1: At time t_1, the TSA generates a time-stamp for a data item, the requester then signs the data together with the provided time-stamp. The requester's signature, including the time-stamp, does not exactly define the point in time when data was signed. It states that the signature was provided after t_1.
- Case 2: The requester first signs data, then at time t_2, the TSA time-stamps signed data. This case expresses that the data was signed prior to the stated point in time t_2.
- Case 3: At time t_1, the TSA generates a time-stamp for a data item, the requester then signs the data together with the provided time-stamp, and at later time t_2, TSA time-stamps signed data. This case defines an interval, between t_1 and t_2, during which the document was signed.

Discussion: We now discuss the property of long-term data integrity in these use cases. Obviously, without time-stamp renewal, neither of these cases can achieve long-term data integrity. Although we do not consider that this is a design flaw, we think that it would be helpful to add a clear note to indicate that without renewal these use cases can only achieve short-term data integrity.

What we are more interested in is how the time-stamp in each case can be renewed and after renewal whether distinguishing of these three use cases still exists. To make the discussion clear, three types of algorithms that will require renewal are considered:

- Hash functions used by a requester to hash the data items into hash values.
- Requester's signature algorithm.
- TSA's time-stamp algorithm.

In Case 1, the requester creating a signature is the last step of this service. If the requester's signature algorithm becomes weak but the hash function and the TSA time-stamp algorithm are still strong, the requester can renew its signature by re-signing the data and the provided time-stamp, using a stronger signature algorithm. The statement that "the signature was provided after the data was time-stamped" can still be claimed. However, if either the hash function or the TSA time-stamping algorithm becomes weak, the requester needs to request for time-stamp renewal. As a result, a TSA will provide a new time-stamp on the requester's signature. Now, this updated Case 1 becomes more or less like Case 3.

In Cases 2 and 3, a TSA generating a time-stamp token is the last step of this service. If the requester's signature algorithm at t_2, the hash function used at t_2 or the TSA's time-stamping algorithm used at t_2 becomes weak, the requester should request a time-stamp renewal for the time-stamp token generated on the signature before any of these algorithms is actually broken.

Proposed Solution: We would like to suggest adding the above discussion in Section 5.4 of ISO/IEC 18014-1 [7] as an informative note to address the security issues.

Missing Renewal Motivations

As mentioned before, time-stamp renewal is a necessary procedure to achieve long-term data integrity of time-stamping services. ISO/IEC 18014-1 [7] provides a list of the reasons why a time-stamp token needs to be renewed. In this list one important reason is missing.

The Source 2 and Source 4 of Section "Inconsistency of data format" have shown that the data submitted to a TSA for being time-stamped is one or more hash values of the actual data item, the data itself is not exposed to the TSA. The hash functions used to transfer data into hash values have limited lifetimes and need to be renewed before they are compromised. Otherwise, once the collision resistance of the hash functions are broken, the corresponding time-stamp token is invalid because the data integrity is compromised. This motivation of time-stamp renewal is not listed in the following source in Section 5.7 of ISO/IEC 18014-1 [7], time-stamp renewal:

"Time-stamped data may be time-stamped again at a later time. This process is called time-stamp renewal and may optionally be implemented by the TSA. This may be necessary for example for the following reasons:

- The mechanism used to bind the time value to the data is near the end of its operational life cycle (e.g. when using a digital signature and the public key certificate is about to expire).
- The cryptographic function used to bind the time value to the data is still trusted; however, there is strong evidence that it will become vulnerable in the near future (e.g. when a hash function is close to begin broken by new attacks or available computing power).
- The issuing TSA is about to terminate operations as a service provider."

Discussion: In this source, neither the "mechanism used to bind the time value to the data" nor the "cryptographic function used to bind the time value to the data" covers the hash functions that are used to protect data nondisclosure. If these hash functions are not renewed, they will become a bottle neck to break the long-term data integrity, that means the data integrity property is protected only within the lifetime of these hash functions, even if other mechanisms and cryptographic functions are renewed correctly.

Proposed Solution: We would like to suggest adding an item in this source:

- "The hash functions used for nondisclosure are nearly to be broken (i.e. either it is not one way or not collision resistant)."

Ambiguity for Renewal Mechanism

In ISO/IEC 18014, it is clearly specified that the "ExtHash" extension allows multiple hash values of a data item to be submitted, but there is not a specification in the whole standard about whether new hash values of a data item are allowed in time-stamp renewal. ExtHash extension is specified in Section 7.4.1 of ISO/IEC 18014-1 [7] as follows:

- "A requester of time-stamping services may wish to submit for time-stamping more than one hash value derived from a single data item."
- "Submitting multiple hash values derived from a single data item using different hash functions allows the requester to insulate the resulting time-stamp token from the cryptographic failure of any single hash function."
- "To enable the submission of multiple hash values the following extension is defined:
  ExtHash ::= SEQUENCE SIZE (1..MAX) OF MessageImprint
  tsp-ext-hash ::= OBJECT IDENTIFIER {tsp-ext 1}
  extHash EXTENSION ::= {
  SYNTAX ExtHash IDENTIFIED BY tsp-ext-hash }"
- "If this extension is present and the TSA is able to process it, then the TSA shall bind both the hash value in the time-stamp request message specified in the messageImprints field and those included in this extension to the time value it assigns to the resulting time-stamp token."

Discussion: The same reason as discussed in section "Missing Renewal Motivations", those hash functions used to transfer data into hash values have limited lifetimes, the renewal of these hash functions are necessary. If multiple hash values are allowed but these hash values must be the same in each time-stamp request, then the hash functions cannot be renewed. The integrity of the data could be only protected before all the applied hash functions are no longer collision resistant. After that, time-stamp tokens generated on these hash values are invalid, which are not able to prove the existence of the data item at a certain point in time.

Proposed Solution: We would like to suggest adding an item as follows in the above source:

- "A requester shall replace the hash value(s) of the data item using a stronger hash function, before the current hash function(s) in the time-stamp token is not collision resistant or preimage resistant."

Inconsistency of the Time-stamp Token Format

In ISO/IEC 18014-2 [8], the time-stamp token format is addressed with several formats that are inconsistent to each other. We now take a look at four sources to check the details.

Source 1: in Section 3, terms and definitions, item 3.18:

- "Time-stamp token: data structure containing a verifiable cryptographic binding between a data item's representation and a time-value."

Source 2: in Section 6.1:

- "A time-stamp token is a data structure containing a verifiable cryptographic binding between a data item's representation and a time-value. A time-stamp token may also bind additional items to the data item's representation and the time-value."

Source 3: again in Section 6.1:

- "A time-stamp token shall contain
  - one or more hash-values of the data that is to be time-stamped, hash functions are specified in the multi-part standard ISO/IEC 10118;
  - a point in time (a time-value);
  - a reference to the policy under which the time-stamp token is generated,

  together with any additional information that may be regarded as helpful for the practical provision of the service, such as
  - identification of the time-stamping service provider (to help verifiers in looking for further evidence);
  - an indication of the accuracy of the time point (that is, the maximum error in the time representation);
  - an indication of ordering (that is, whether the service provider guarantees the relative ordering of generated tokens);
  - identification of the version of the format (foreseeing syntax changes in the future);
  - a serial number (to enable reference to be made to the token);
  - a reference to the user's request, to help users in matching requests and responses."

Source 4: in Section 6.2, notation:

- "The time-stamp token $TST(t)$ may be further decomposed into its parts:

$$TST(t) :=< \{H_i(D)\},\ t,\ P >,$$

  where $\{H_i(D)\}$ is the set of one or more hash-values on data D. P indicates the policy under which the token was generated."

Discussion: Sources 1 and 2 indicate that a time-stamp token is a data structure containing a verifiable cryptographic binding between a data item's representation and a time-value. However, Sources 3 and 4 show that the information contained in a time-stamp token does not include the verifiable cryptographic binding, which is a contradiction with Sources 1 and 2. This contradiction breaks the consistency of the format of time-stamp tokens.

Proposed Solution: We would like to suggest adding the following item in Source 3:

- "A cryptographic binding that binds between a data item's representation and a time-value."

In Source 4, we would like to suggest adding a notation of a verifiable cryptographic binding in the components of time-stamp tokens. For example,

- "The time-stamp token $TST(t)$ may be further decomposed into its parts:

$$TST(t) :=< \{H_i(D)\},\ t,\ C,\ P >,$$

  where C is a cryptographic binding between $H_i(D)$ and t."

Conclusion

In this paper, we discuss several issues in ISO/IEC 18014, the international standard for time-stamping services. We demonstrate that these issues may affect the data nondisclosure and long-term data integrity properties, which are required in this standard. We propose modifications to fix these issues. Moreover, it would be helpful to the users of this ISO/IEC standard if we could add an informative annex in ISO/IEC 18014-1 [7] to discuss data nondisclosure and data integrity, and the concept of short-term and long-term security.

Acknowledgements

This work is supported by the European Union's Horizon 2020 research and innovation program under grant agreement No.779391 (FutureTPM) and grant agreement No. 952697 (ASSURED). The authors would like to thank the anonymous reviewers for their helpful comments.

References

[1] A. K. Lenstra. "Key Length. Contribution to The Handbook of Information Security", 2004.

[2] National Institute of Standards and Technology (NIST). "Recommendation for Digital Signature Timeliness". Standard, 2009.

[3] C. Adams *et. al.* "RFC 3161: Internet X. 509 Public Key Infrastructure Time-Stamp Protocol (TSP)", 2001.

[4] European Telecommunications Standards Institute (ETSI). "Electronic Signatures and Infrastructures (ESI); Policy and Security Requirements for Trust Service Providers issuing Time-Stamps". Standard, 2015.

[5] European Telecommunications Standards Institute (ETSI). "Electronic Signatures and Infrastructures (ESI); Time-stamping protocol and time-stamp token profiles". Standard, 2016.

[6] American National Standard Institute (ANSI). "ANSI X9.95-2016 – Trusted Timestamp Management and Security", 2016.

[7] ISO/IEC 18014-1:2008. "Information technology – Security techniques – Time-stamping services – part 1: Framework". Standard, 2008.

[8] ISO/IEC 18014-2:2009. "Information technology – Security techniques – Time-stamping services – part 2: Mechanisms producing independent tokens". Standard, 2009.

[9] ISO/IEC 18014-3:2009. "Information technology – Security techniques – Time-stamping services – part 3: Mechanisms producing linked tokens". Standard, 2009.

[10] ISO/IEC 18014-4:2015. "Information technology – Security techniques – Time-stamping services – part 4: Traceability of time sources". Standard, 2015.

[11] S. Haber and W. S. Stornetta. "How to Time-Stamp a Digital Document". In Conference on the Theory and Application of Cryptography, pages 437–455. Springer, 1990.

[12] D. Bayer, S. Haber, and W. S. Stornetta. "Improving the Efficiency and Reliability of Digital Time-Stamping". In Sequences Ii, pages 329–334. Springer, 1993.

[13] M. Geihs, D. Demirel, and J. Buchmann. "A security analysis of techniques for long-term integrity protection". In 2016 14th Annual Conference on Privacy, Security and Trust (PST), pages 449–456. IEEE, 2016.

[14] A. Buldas, M. Geihs, and J. Buchmann. "Long-Term Secure Time-Stamping using Preimage-Aware Hash Functions". In International Conference on Provable Security, pages 251–260. Springer, 2017.

[15] R. C. Merkle. "A Certified Digital Signature". In Conference on the Theory and Application of Cryptology, pages 218–238. Springer, 1989.

BIOGRAPHIES

Long Meng (lm00810@surrey.ac.uk) received his B.Sc degree from Beijing University of Technology in 2017 and M.Sc degree from the University of Surrey in 2018. He is currently a PhD student at the University of Surrey under the supervision of Professor Liqun Chen. His research interests include security analysis of long-term applications and cryptography updating in blockchain.

Liqun Chen [SM] (liqun.chen@surrey.ac.uk) is a Professor in Secure Systems at the University of Surrey. Prior to taking this position in 2016, she was a principal research scientist at Hewlett-Packard Laboratories, Bristol, UK. She developed several cryptographic schemes that were adopted by International Standards bodies, ISO/IEC, IEEE and TCG. In particular, she designed several cryptographic algorithms (including direct anonymous attestation and multiple signature interfaces) used in the Trusted Platform Module (TPM). She co-authored the paper "Direct anonymous attestation", which was originally published at ACM CCS 2004 and received a Test of Time award at ACM CCS 2014. Her current research interests are applied cryptography, trusted computing and network security. She has served as editor or co-editor for seven ISO/IEC standard documents.