Reviewing the ISO/IEC Standard for Timestamping Services

Long Meng and Liqun Chen

Abstract

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

Introduction

Timestamping services provide timestamp tokens on data items to prove that the data existed at certain points in time. A timestamping service involves a timestamp requester, a timestamping authority, and a timestamp verifier. The authority creates a timestamp token for a data item after the requester asks for it, and the token can be verified by the verifier. Timestamp tokens are created by binding the data item and time together using cryptographic algorithms, such as digital signatures and hash functions. A timestamp 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 timestamp token. For privacy reasons, the requester may not reveal the data being timestamped to the authority. This security property is referred to as data nondisclosure of a timestamp 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 life cycle and the increasing computational power of attackers [1]. Timestamp 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 article, if a security property, data integrity, or data nondisclosure of a timestamp token relies on the continuing validity of the underlying cryptographic algorithms that are used to generate the token, we say that this timestamp token holds this property in the *short term*; otherwise, we say that it holds this property in the *long term*.

Several standards bodies have specified timestamping services, including the U.S. 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 timestamping 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 timestamped to the timestamping 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 the short or long term.

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

The NIST standard specifies a signature-based timestamping application for proving time evidence of digital signatures [2], in which the data nondisclosure requirement and timestamp token renewal are outside the scope of the standard.

The IETF standard specifies signature-based timestamping protocols [3]. This standard supports data nondisclosure, which is based on hash functions. The timestamp 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 timestamps [4], and defines what a timestamp requester and a timestamping authority should support [5]. In both documents, the timestamping protocols follow the IETF standard [3], in which the data nondisclo-

Digital Object Identifier:
10.1109/MCOMSTD.011.2000083

The authors are with the University of Surrey.

sure is protected by hash functions and the timestamp token renewal is not in their scope.

The ANSI standard specifies timestamping services [6] and includes the specification of mechanisms for both data nondisclosure and timestamp token renewal. It also makes use of hash functions for data nondisclosure. Three applications of timestamps are specified:

- Timestamp 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
- Timestamp 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
- Timestamp 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 timestamping services, supporting both data nondisclosure and timestamp 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 timestamped to the timestamping authority, so it holds short-term data nondisclosure. This standard aims to achieve long-term data integrity by using timestamp token renewal.

In the remaining part of this article, we first briefly introduce more related works in this research field and then focus on reviewing the ISO/IEC 18014 standard [7–10] in more detail. The major contribution of this article is that we report our discovery of several issues. Due to inconsistent writing or information missing, a timestamping service, following the standard specification, may not be able to achieve these designed properties. We 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 timestamping with two techniques: linear linking and random witness. In this article, they also proposed a solution for timestamp token renewal, in which a timestamp token could be renewed by timestamping the token with a new implementation before the old implementation is compromised.

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

As mentioned in the previous section, several standards have specified timestamping 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 timestamping renewal in [12] has been extended into several long-term integrity schemes. However, the security of these schemes were not given until 2016, when Geihs *et al.* formalized the property of long-term data integrity in timestamping services. Their works were reported in two separate papers, focusing on a signature-based long-term integrity scheme based on timestamping techniques [13] and a long-term hash-based timestamping scheme [14], respectively. These two schemes provide substantial frameworks for analyzing the long-term data integrity property in timestamping services.

An Overview of the ISO/IEC Standardized Timestamping Services

The ISO/IEC 18014 standard specifies timestamping 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 *timestamping service* specified in ISO/IEC 18014 makes use of a protection mechanism and generates one of the following two types of timestamp tokens.

Independent Tokens: An independent timestamp token can be verified without involving other timestamp tokens. The protection mechanism used to generate this type of token can be digital signatures, message authentication codes, or archives. For example, for signature-based timestamping, a Timestamping Authority (TSA) digitally signs a data item and a time value, which results in a cryptographic binding between the data and time. The data, time, and the corresponding signature together form a timestamp token.

Linked Tokens: A linked timestamp token is associated with other timestamp tokens produced by the same methods. The protection mechanism used to generate this type of token can be hash functions and a public repository; therefore, a timestamping service generating this type of token is referred to as "hash-based timestamping" or "repository-based timestamping." Specifically, 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., using a Merkle Tree [15]). The aggregation result can be linked to other data produced at previous times (e.g., using linear chain linking [11]). Eventually, the aggregation or linking result is published in a widely visible medium (e.g., newspapers). The data, time record, published information, and group values, which are contributed to determine the published information, together form a timestamp token.

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

- Timestamp request transaction: A requester sends a *timestamp request* to a TSA, and the TSA returns a *timestamp response* to the requester.
- Timestamp 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 timestamp request and response, as shown in Fig. 1, are specified in ISO/

The aggregation result can be linked to other data produced at previous times, (e.g., using linear chain linking [11]). Eventually, the aggregation or linking result is published in a widely visible medium (e.g., newspapers). The data, time record, published information and group values, which are contributed to determine the published information, together form a timestamp token.

The ISO/IEC standard specifies timestamping services, which supports both data nondisclosure and timestamp 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 timestamped to the timestamping authority, so it holds short-term data nondisclosure. This standard aims to achieve long-term data integrity by using timestamp token renewal.

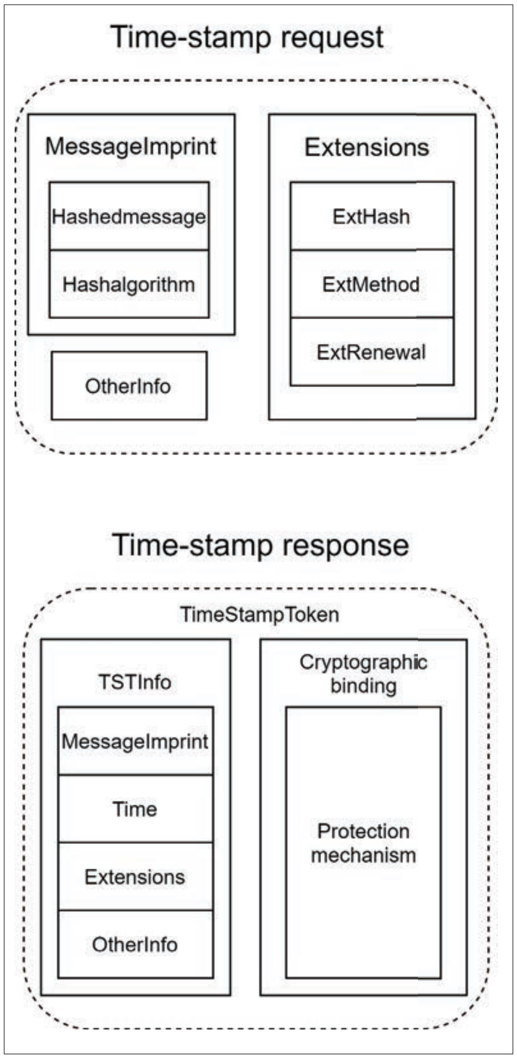

FIGURE 1. Data formats of timestamp request and timestamp response.

IEC 18014-1 [7]. *A timestamp request* contains a "messageImprint" field, which comprises 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:

- ExtHash: In this field, a requester could submit multiple "messageImprint" fields, in which each hash value could be computed from a different hash function, preventing the failure of any single hash function.
- 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.
- ExtRenewal: In this field, a requester could submit an existing timestamp token on the data item for the purpose of extending the validity period of the timestamp 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 timestamp token; then the TSA returns a *timestamp response* with the timestamp token to the requester.

In order to validate the timestamp token, the requester could send a *verification request* that contains the timestamp token to a verifier at time t_v. For a single timestamp 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 functions used to generated hash values of the data item is collision-resistant at t_v.
- The protection mechanism of the timestamp 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 timestamp 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 timestamp token, the verifier checks the validity of each nested timestamp token at the time it was generated or renewed, and the validity of the latest timestamp 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 timestamping services.

Inconsistency of Data Format

The data format for a data item being timestamped 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 timestamped"
- "TS(x_1, x_2, ..., x_n): generation of timestamp 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 timestamping methods specified in this standard solve these requirements by timestamping 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, "Timestamp Renewal":

- "Let data *D* be timestamped at time T_0
  TS(*D*, (other info), T_0)."
  "At time T_1, while the timestamp is trusted, a renewal such as TS(*D*, (TS(*D*, (other info), T_0), other info), T_1) still proves the existence of *D* at T_0, given that the first timestamp is valid at T_1. "

Source 4: in Section 7.1 of the same document, "Timestamp Request":

- "Type MessageImprint is used to encapsulate the message imprint data along with an indicator of the algorithm used to generate 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 timestamped, 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 timestamped is not disclosed to the timestamping authority, and it should be hashed into a hash value before being submitted to the authority for timestamping. Source 4 provides the implementation, in which the format of the data to be timestamped is defined in MessageImprint, and it only comprises 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 timestamped, D, is directly presented as input to the generation of a timestamp 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 timestamped and H_0 be a hash function or a set of hash functions at time T_0. A timestamp token is generated by

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

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

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

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

Short-Term Data Integrity for Signed Data

ISO/IEC 18014 aims to provide long-term data integrity to timestamping services. In Section 5.4 ("Use of Timestamps") of ISO/IEC 18014-1 [7], three use cases for timestamping 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:

"Timestamps also play an important role for the validity of signed documents. There exist three possibilities for the time at which timestamping and signing of data may occur. Data may be timestamped before the requester of the timestamp 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 timestamp for a data item; the requester then signs the data together with the provided timestamp. The requester's signature, including the timestamp, 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 timestamps 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 timestamp for a data item, the requester then signs the data together with the provided timestamp, and at a later time, t_2, the TSA timestamps 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 timestamp renewal, neither of these cases can achieve long-term data integrity. Although we do not consider this 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 timestamp 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 timestamp 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 timestamp algorithm are still strong, the requester can renew its signature by re-signing the data and the provided timestamp using a stronger signature algorithm. The statement that "the signature was provided after the data was timestamped" can still be claimed. However, if either the hash function or the TSA timestamping algorithm becomes weak, the requester needs to request timestamp renewal. As a result, a TSA will provide a new timestamp on the requester's signature. Now, this updated Case 1 becomes more or less like Case 3.

The statement that "the signature was provided after the data was timestamped" can still be claimed. However, if either the hash function or the TSA timestamping algorithm becomes weak, the requester needs to request timestamp renewal. As a result, a TSA will provide a new timestamp 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 timestamp 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 timestamping algorithm used at t_2 becomes weak, the requester should request a timestamp renewal for the timestamp 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, timestamp renewal is a necessary procedure to achieve long-term data integrity of timestamping services. ISO/IEC 18014-1 [7] provides a list of the reasons why a time-

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 no specification in the whole standard about whether new hash values of a data item are allowed in timestamp renewal. ExtHash extension is specified in Section 7.4.1 of ISO/IEC 18014-1.

stamp token needs to be renewed. In this list, one important reason is missing.

Source 2 and Source 4 of "Inconsistency of Data Format" have shown that the data submitted to a TSA for being timestamped 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 is broken, the corresponding timestamp token is invalid because the data integrity is compromised. This motivation of timestamp renewal is not listed in the following source in Section 5.7 of ISO/IEC 18014-1 [7], "Timestamp Renewal":

"Timestamped data may be timestamped again at a later time. This process is called timestamp 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 being 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 bottleneck to break the long-term data integrity, which 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 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 no specification in the whole standard about whether new hash values of a data item are allowed in timestamp renewal. ExtHash extension is specified in Section 7.4.1 of ISO/IEC 18014-1 [7] as follows:

- "A requester of timestamping services may wish to submit for timestamping 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 timestamp 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 timestamp request message specified in the messageImprints field and those included in this extension to the time value it assigns to the resulting timestamp token."

Discussion: For the same reason discussed in "Missing Renewal Motivations," those hash functions used to transfer data into hash values have limited lifetimes, so the renewal of these hash functions is necessary. If multiple hash values are allowed, but these hash values must be the same in each timestamp request, the hash functions cannot be renewed. The integrity of the data could only be protected before all the applied hash functions are no longer collision-resistant. After that, timestamp tokens generated on these hash values are invalid, and 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 timestamp token is not collision-resistant or preimage-resistant."

Inconsistency of the Timestamp Token Format

In ISO/IEC 18014-2 [8], the timestamp token format is addressed with several formats that are inconsistent with 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:

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

Source 3: again in Section 6.1:

- "A timestamp token shall contain
  – one or more hash-values of the data that is to be timestamped, 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 timestamp 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 timestamping 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 timestamp 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 timestamp 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 timestamp token does not include the verifiable cryptographic binding, which is a contradiction to Sources 1 and 2. This contradiction breaks the consistency of the format of timestamp 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 timestamp tokens. For example:

- "The timestamp 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 article, we discuss several issues in ISO/IEC 18014, the international standard for timestamping services. We demonstrate that these issues may affect the data nondisclosure and long-term data integrity properties 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.

Acknowledgments

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.

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.

References

[1] A. K. Lenstra, "Key Length: Contribution to *The Handbook of Information Security*," 2004.
[2] NIST Std., "Recommendation for Digital Signature Timeliness," 2009.
[3] C. Adams *et al.*, "RFC 3161: Internet X. 509 Public Key Infrastructure Timestamp Protocol (TSP)," 2001.
[4] ETSI Std., "Electronic Signatures and Infrastructures (ESI); Policy and Security Requirements for Trust Service Providers Issuing Timestamps," 2015.
[5] ETSI Std., "Electronic Signatures and Infrastructures (ESI); Timestamping Protocol and Timestamp Token Profiles," 2016.
[6] ANSI, "ANSI X9.95-2016 — Trusted Timestamp Management and Security," 2016.
[7] ISO/IEC 18014-1:2008, "Information Technology — Security Techniques — Timestamping Services — Part 1: Framework," 2008.
[8] ISO/IEC 18014-2:2009, "Information Technology — Security Techniques — Timestamping Services — Part 2: Mechanisms Producing Independent Tokens," 2009.
[9] ISO/IEC 18014-3:2009, "Information Technology — Security Techniques — Timestamping Services — Part 3: Mechanisms Producing Linked Tokens," 2009.
[10] ISO/IEC 18014-4:2015, "Information Technology — Security Techniques — Timestamping Services — Part 4: Traceability of Time Sources," 2015.
[11] S. Haber and W. S. Stornetta, "How to Timestamp a Digital Document," *Proc. Conf. Theory and Application of Cryptography*, Springer, 1990, pp. 437–55.
[12] D. Bayer, S. Haber, and W. S. Stornetta, "Improving the Efficiency and Reliability of Digital Timestamping," *Sequences Ii*, Springer, 1993, pp. 329–34.
[13] M. Geihs, D. Demirel, and J. Buchmann, "A Security Analysis of Techniques for Long-Term Integrity Protection," *Proc. 2016 14th IEEE Annual Conf. Privacy, Security and Trust*, 2016, pp. 449–56.
[14] A. Buldas, M. Geihs, and J. Buchmann, "Long-Term Secure Time-Stamping Using Preimage-Aware Hash Functions," *Proc. Int'l. Conf. Provable Security*, Springer, 2017, pp. 251–60.
[15] R. C. Merkle, "A Certified Digital Signature," *Proc. Conf. Theory and Application of Cryptology*, Springer, 1989, pp. 218–38.

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 Ph.D. 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, United Kingdom. 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.