W3C Standardization: RDF Dataset Canonicalization

2.1 Introduction

In this article, I describe RDF Dataset Canonicalization[*1], which became a World Wide Web Consortium (W3C) recommendation in May 2024. I was involved in the standardization process at W3C. RDF Dataset Canonicalization is an algorithm for canonicalizing (i.e., normalizing or generating a serial canonicalization of) data represented using the Resource Description Framework (RDF). I explain what RDF is, what the process of RDF canonicalization entails, and when it is needed. I also go over the path we took to standardization at W3C and describe the canonicalization procedure in some detail.

2.2 What is RDF?

RDF is a W3C standard framework for describing information (resources) on the web. RDF makes it easy to link data between different databases and applications. It is widely used in areas such as the life sciences, pharmacology, and libraries for this reason. The first version became a W3C recommendation in 1999, and RDF 1.1[*2] became a recommendation in 2004. As of this writing (May 2024), work on the RDF 1.2[*3] standard is underway.

RDF represents information with three elements: a subject, a predicate, and an object. A set of these three elements is called an RDF triple. By way of example, the following is an RDF triple on the classic Japanese literary work *The Pillow Book (Makura no Soshi*, rendered below as "Makuranosoushi" in keeping with the Japan Search records) obtained from Japan Search and slightly modified for this article[*4], a site that lets you search through a wide range of Japanese content.

- Subject: <https://jpsearch.go.jp/data/bibnl-20853658>
- Predicate: <https://www.w3.org/2000/01/rdf-schema#label>
- Object: "Makuranosoushi"

RDF triples can be read like a normal sentence, as "the predicate of the subject is the object." That is, the RDF triple here can be read as "the label of bibnl-20853658 is Makuranosoushi." Here, the subject https://jpsearch.go.jp/data/bibnl-20853658 is an identifier assigned to a book by Japan Search. The predicate <https://www.w3.org/2000/01/rdf-schema#label> is a term defined in the W3C RDF Schema[*5], and the object that follows it, "Makuranosoushi", is the label (brief description) of the subject. Hence, this RDF triple indicates that the information with identifier <https://jpsearch.go.jp/data/bibnl-2085658> takes the label "Makuranosoushi".

So, RDF triples represent a lot of information using URLs[*6] like <https://jpsearch.go.jp/data/bibnl-20853658>[*7]. URLs are used so as to accurately convey the information that the data creator is trying to represent. If the subject and predicate were expressed without a URL as simply "20853658" and "label", readers would find it difficult to correctly understand where the 20853658 identifier comes from and what the meaning of the predicate label is.

RDF triples can also be drawn as a diagram with two nodes (information contained in ovals or boxes) connected by an arrow, as in Figure 1. For ease of reading, part of the URL in Figure 1, https://jpsearch.go.jp/data/, is abbreviated to "data:". Similarly, http://www.w3.org/2000/01/rdf-schema# is replaced by "rdfs:". I use this abbreviated notation below as well.

A collection of RDF triples is called an RDF graph. Retrieving additional RDF triples on The Pillow Book from Japan Search enables us to create an RDF graph like that in Figure 2.

Figure 1: Example of an RDF Triple Referring to Makuranosoushi

*1 Dave Longley, Gregg Kellogg, Dan Yamamoto: RDF Dataset Canonicalization. W3C Recommendation, May 21, 2024 (https://www.w3.org/TR/rdfcanon/).
*2 Richard Cyganiak, David Wood, Markus Lanthaler: RDF 1.1 Concepts and Abstract Syntax. W3C Recommendation, February 25, 2014 (https://www.w3.org/TR/rdf11- concepts/).
*3 Olaf Hartig, Pierre-Antoine Champin, Gregg Kellogg, Andy Seaborne: RDF 1.2 Concepts and Abstract Syntax. W3C Working Draft, May 2, 2024 (https://www.w3.org/TR/2024/WD-rdf12-concepts-20240502/).
*4 Japan Search (https://jpsearch.go.jp/).
*5 Dan Brickley, R.V. Guha: RDF Schema 1.1. W3C Recommendation, February 25, 2014 (https://www.w3.org/TR/rdf11-schema/).
*6 To be precise, an Internationalized Resource Identifier (IRI), which is a generalization of a URL, is used.
*7 In this example, the object is expressed as a string rather than as a URL, but it is common for triples to have a URL as the object.

In this RDF graph, the label for data:bibnl-20853658 is "Makuranosoushi", and we can also see that Sei shounagon was involved in the book's production and that Moriya Shinsuke was involved in its translation.

As mentioned, a collection of RDF triples forms an RDF graph. Additionally, a collection of RDF graphs is called an RDF dataset. RDF Dataset Canonicalization is, as the name suggests, a method of canonicalizing RDF datasets. For simplicity, however, I will not distinguish between RDF graphs and RDF datasets in this article.

2.3 Blank RDF Nodes

In the example in Figure 2, nodes with the strange names _:b152539105 and _:b152573899 appear. These are a special type of node called blank nodes and do not have an identifier (URL). When creating large RDF graphs, for example, it can be cumbersome to assign URLs to all nodes. So blank nodes without URLs are sometimes used for intermediate nodes not connected to other graphs.

Names given to blank nodes are only temporary. Within the same RDF graph, the names of blank nodes may change depending on the system or environment in which they are handled. For example, while the RDF graph in Figure 3 has _:hoge and _:fuga instead of Figure 2's _:b152539105 and _:b152573899, respectively, it is treated as being the same as the RDF graph before those name replacements (more precisely, it is isomorphic to that graph).

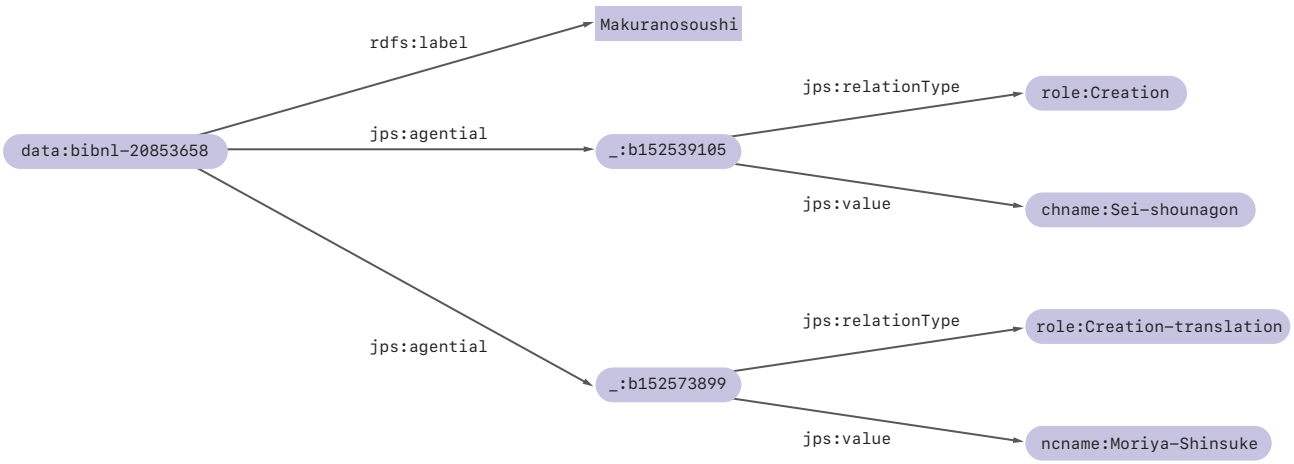

Figure 2: An RDF Graph for *The Pillow Book*

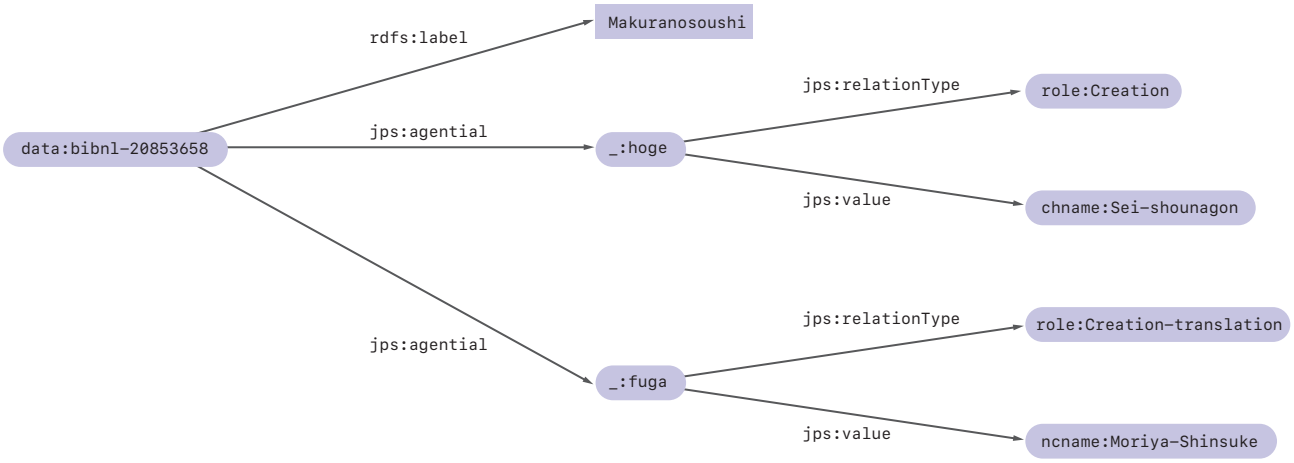

Figure 3: Another RDF Graph for *The Pillow Book*

An advantage of this is that RDF graph creators need not worry about naming blank nodes. Another advantage is that these names can be omitted when converting the RDF graph into data. For example, the RDF graph in Figure 3 can be expressed as follows using the JSON-LD[*8] specification. Here, there is no need to worry about the names of blank nodes.

```
{
    "@context": { ... },
    "@id": "data:bibnl-20853658",
    "rdfs:label": "Makuranosoushi",
    "jps:agential": [
        {
            "jps:relationType": "role:Creation",
            "jps:value": "chname:Sei-shounagon"
        },
        {
            "jps:relationType": "role:Creation-translation",
            "jps:value": "ncname:Moriya-Shinsuke"
        }
    ]
}
```

2.4 Canonicalization

Blank nodes are useful, but their lack of a fixed name can sometimes cause problems. For example, handling blank nodes can be problematic when you want to check whether two RDF graphs are isomorphic, determine the differences between graphs, or determine whether an RDF graph has been updated. Also, if an RDF graph is digitally signed by its creator, verification will fail if the names of blank nodes when the graph is signed differ from the names of blank nodes when the verification attempt is made, but due to the nature of blank nodes, it is not possible to guarantee that the names will be the same.

We therefore needed a method of assigning fixed names to blank nodes that would be independent of the system or environment. That's where RDF Dataset Canonicalization, the subject of this article, comes in. When canonicalized, the two RDF graphs in Figures 2 and 3, for example, are converted into the same graph, which is shown in Figure 4.

Once canonicalized, the blank nodes take the new names _:c14n0 and _:c14n1[*9]. These names are calculated using a predetermined method based on the URLs and strings that appear in the graph and the structure of the graph, and they are not influenced by the values originally assigned to the blank nodes, i.e., _:b152539105, _:hoge, and so forth. Thus, the same graph can be

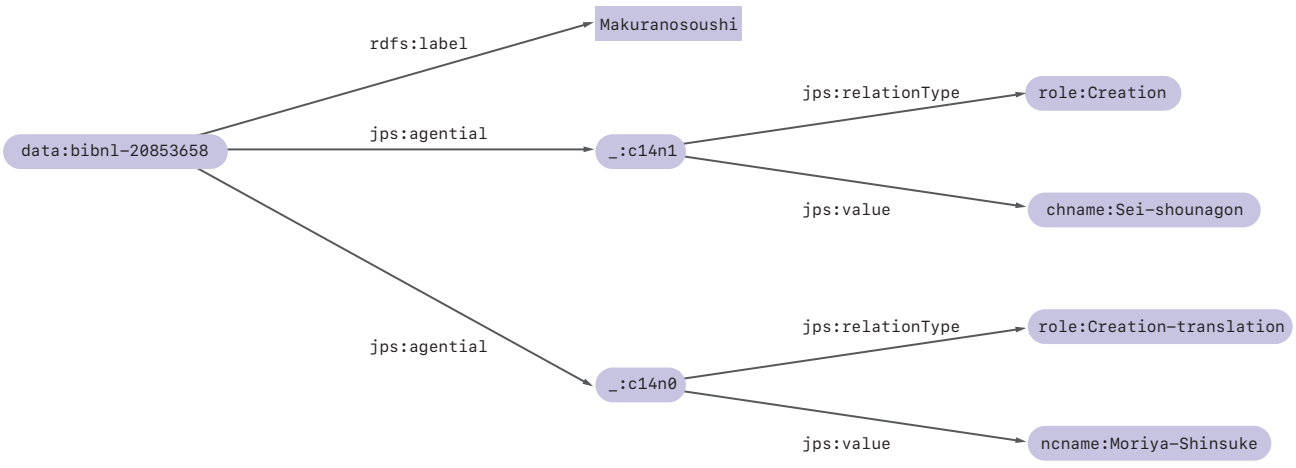

Figure 4: Example of a Canonicalized RDF Graph

*8 Gregg Kellogg, Pierre-Antoine Champin, Dave Longley: JSON-LD 1.1. W3C Recommendation, July 16, 2020 (https://www.w3.org/TR/json-ld11/).

*9 "c14n" is an abbreviation of "canonicalization".

obtained regardless of what names were assigned to the original blank nodes.

Once the blank node names are determined, a representation of the dataset called its canonical n-quads form[*10] can be generated, which, in our case, gives the canonicalized data shown in Figure 5. Using the canonicalized data, it is easy to calculate RDF graph differences, check for updates, and calculate digital signatures and hashes.

W3C Verifiable Credentials, a form of digital credentials that we covered in IIR Vol.52 (https://www.iij.ad.jp/en/dev/iir/052.html)[*11], are RDF datasets with a digital signature. Canonicalizing blank node names using RDF Dataset Canonicalization before applying the digital signature guarantees that the data that is signed will be the same as the data that is verified.

2.5 The Standardization Effort

Standardization is typically a lengthy process. The standardization of RDF Dataset Canonicalization took over a decade. Although the discussion around it started early on, one reason it took so long is that it took ages to arrive at a consensus on the need for standardization and what the optimal method would be[*12].

Discussions on the canonicalization specification first began at W3C from 2009 to 2010. In 2012, Dave Longley and Manu Sporny of Digital Bazaar proposed the Universal RDF Graph Normalization Algorithm (URGNA2012). This was followed three years later by a revised version, the Universal RDF Dataset Normalization Algorithm (URDNA2015), which became the basis for the now standardized specification.

The discussion around verifiable credentials subsequently ramped up at W3C, and 2021 saw the proposed formation of the Linked Data Signatures Working Group to work on methods of digitally signing RDF data. This effort was terminated, however, after it failed to reach consensus on the standardization of the overall signing process. The RDF Dataset Canonicalization and Hash Working Group (RCH WG)[*13], focused on RDF canonicalization, was proposed as an alternative and approved in July 2022.

And so the work to make RDF Canonicalization a W3C Recommendation finally began. On May 21, 2024, this standardization effort reached its goal of producing a W3C Recommendation[*14].

Following an invitation from the WG chair, I joined RCH WG as an Invited Expert in August 2022, and I also served as an Editor from November that year. The invitation was prompted by an article on verifiable credentials written by me and colleagues for an international conference[*15], which caught the interest of the WG co-chair.

```
<https://jpsearch.go.jp/data/bibnl-20853658> <https://jpsearch.go.jp/term/property#agential> _:c14n0 .
<https://jpsearch.go.jp/data/bibnl-20853658> <https://jpsearch.go.jp/term/property#agential> _:c14n1 .
<https://jpsearch.go.jp/data/bibnl-20853658> <rdfs:label>"Makuranosoushi".
_:c14n0 <https://jpsearch.go.jp/term/property#relationType> <https://jpsearch.go.jp/term/role/Creation-translation>.
_:c14n0 <https://jpsearch.go.jp/term/property#value> <https://jpsearch.go.jp/entity/ncname/Moriya-Shinsuke>.
_:c14n1 <https://jpsearch.go.jp/term/property#relationType> <https://jpsearch.go.jp/term/role/Creation>.
_:c14n1 <https://jpsearch.go.jp/term/property#value> <https://jpsearch.go.jp/entity/chname/Sei-Shounagon>.
```

Figure 5: The Result of Canonicalization (several URLs modified for translation purposes in this article)

*10 Generally, data in n-quads form does not need to have been sorted, and there is no limit on the number of whitespace or line feed characters used as delimiters, but here we use data sorted in lexicographical order and limit the number of delimiter characters to one. This is called the canonical n-quads form.

*11 Internet Infrastructure Review Vol. 52, "2. Focused Research (1): Verifiable Credentials and BBS+ Signatures" (https://www.iij.ad.jp/en/dev/iir/052.html). The LD Canonicalization referred to in Vol. 52 is the former name of RDF Dataset Canonicalization discussed here.

*12 Email by Phil Archer (https://lists.w3.org/Archives/Public/semantic-web/2024May/0030.html).

*13 W3C RDF Dataset Canonicalization and Hash Working Group (https://www.w3.org/groups/wg/rch/).

*14 RDF Dataset Canonicalization and Hash Working Group Charter (https://w3c.github.io/rch-wg-charter/).

*15 Dan Yamamoto, Yuji Suga, Kazue Sako, Formalising Linked-Data based Verifiable Credentials for Selective Disclosure. 2022 IEEE European Symposium on Security andPrivacy Workshops (EuroS&PW) (https://doi.org/10.1109/EuroSPW55150.2022.00013).

RCH WG mainly conducts its activities via GitHub discussions and fortnightly conference calls. Anyone can raise issues and give proposed solutions via GitHub, while on the conference calls, working group members engage in discussion to resolve issues and reach a consensus. The results of all this go through some editing on GitHub and then eventually end up in the specifications. This was my first involvement in standardization, and while I found it hard to keep up with the expert discussion, I made an effort to contribute in any way I could: proposing wordings, reviewing pull requests, and providing reference implementations, and so on.

It is crucial that W3C specifications are created in such a way that readers are able to correctly implement the content. As of this of writing (May 2024), nine open-source implementations of RDF Dataset Canonicalization have been released, having been developed in a wide range of languages: C++, Elixir, Java, JavaScript, Ruby, Rust, and TypeScript[*16]. I have also released an open-source implementation in Rust[*17].

2.6 Canonicalization Procedure

The algorithm defined in the RDF Dataset Canonicalization specification is named RDF Canonicalization algorithm version 1.0, commonly known as RDFC-1.0. Here, I give an overview of RDFC-1.0.

RDFC-1.0 consists of two steps: canonicalization, in which blank nodes in the input RDF graph are labeled, and serialization, in which the canonical n-quads form of the canonicalized RDF graph is generated.

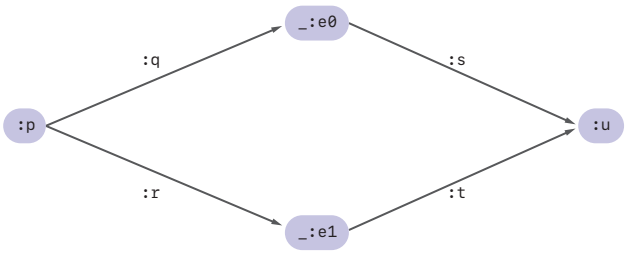

Figure 6: Example of an RDF Graph with Two Blank Nodes

In the canonicalization step, a value called the first degree hash is calculated for each blank node in the graph. This is done by passing the information around the blank node into a special function called a hash function to obtain a fixed-length block of data called the hash value. Intuitively, this constitutes giving a name to blank nodes using the information surrounding it.

If the first degree hashes assigned to the blank nodes are all different, then they can be sorted in lexicographical order[*18] so as to assign an order to the blank nodes. Labeling the blank nodes in this order—i.e., _ :c14n0, _ :c14n1, _ :c14n2, and so on—completes the canonicalization process.

I will now explain this process in detail using the example in Figure 6. This RDF graph contains four nodes, two of which (:p and :u) are normal nodes that have URLs, and the remaining two (_ :e0 and _ :e1) are blank nodes.

Extracting just the RDF triple containing blank node _ :e0 and representing it in canonical n-quads form yields this:

```
:p :q _:e0 .
_:e0 :s :u .
```

This corresponds to the information surrounding _ :e0. Here, we replace the blank node's "temporary" name of e0 with a, which yields the following string:

```
:p :q _:a .
_:a :s :u .
```

*16 Gregg Kellogg: RDF Dataset Canonicalization and Hash 1.0 Processor Conformance (https://w3c.github.io/rdf-canon/reports/).
*17 zkp-ld/rdf-canon (https://github.com/zkp-ld/rdf-canon).
*18 To be precise, they are sorted in Unicode code point order.

This is passed into the hash function, and the resulting hexadecimal bit string, 21d1dd5ba21f3dee9d76c0c00c260fa6f5d5d65315099e553026f4828d0dc77a, is used as the first degree hash of blank node _ :e0. Information about _ :e0 is embedded in this first degree hash value, and it can be used to distinguish _ :e0 from other blank nodes.

Similarly, extracting the RDF triple containing _ :e1 gives,

```
:p :r _:e1 .
_:e1 :t :u .
```

and with e1 replaced by a, as before,

```
:p :r _:a .
_:a :t :u .
```

and we then generate a first degree hash value for _ :e1 of 6fa0b9bdb376852b5743ff39ca4cbf7ea14d34966b2828478fbf222e7c764473.

When sorted in lexicographical order, the first degree hash of _ :e0, which starts with 2, comes before the first degree hash of _ :e1, which starts with 6. So we have been able to determine an ordering for e0 and e1. We then follow this order and assign the canonicalization identifier _ :c14n0 to _ :e0 and _ :c14n1 to _ :e1, completing the canonicalization process.

Crucially, the canonicalization result is always the same, regardless of what names are given to the blank nodes before canonicalization. Indeed, using the example in Figure 6, we can check that the first degree hash values do not change even if we replace _ :e0 with _ :hoge and _ :e1 with _ :fuga. As part of the process of calculating the first degree hash values, all blank node names are replaced by a, and this means that the end result is independent of the names originally given to the blank nodes.

RDF graphs that are not too complicated, such as the example in Figure 6, can be canonicalized by calculating just the first degree hashes, and this is relatively easy to do. But depending on the RDF graph, you can end up assigning the same first degree hash to different blank nodes. The graph in Figure 7, for example, contains blank nodes surrounded by information that is exactly the same, and in such situations, the nodes will be assigned the same first degree hash.

Looking at _ :e0 and _ :e1, we can see that both are objects reached from subject :p via predicate :q, and furthermore, that both are subjects that lead to a blank node object via predicate :p. Because of this, their first degree hash values will be exactly the same.

In RDFC-1.0, therefore, n-degree hashes are calculated as the next viable identification method only in cases of blank nodes being assigned the same first degree hash. The process by which n-degree hashes are calculated in RDFC-1.0 is complicated, so I will not explain it here. The interested reader is directed to the specification.

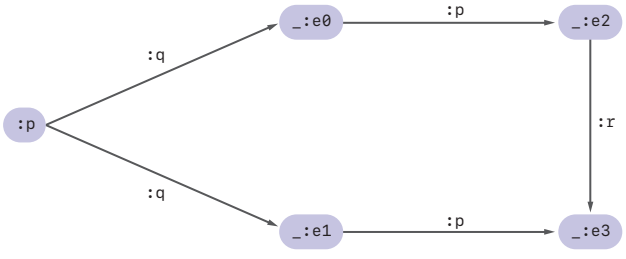

Figure 7: Example of a More Complicated RDF Graph

2.7 Canonicalization Challenges and Solutions

As should now be evident, RDF Dataset Canonicalization is, in essence, a way of obtaining canonicalized names by ordering blank nodes. So to canonicalize an RDF graph that does not contain any blank nodes, there is no need to calculate any first degree or n-degree hashes; all you have to do is perform the simple (serialization) process of representing the RDF graph in n-quads form and then sorting.

There is a misconception that RDF Dataset Canonicalization involves unnecessarily complicated processing, but the complexity of canonicalization depends on the number of blank nodes in the input RDF graph and the structure of the graph if it contains blank nodes. In practice, the process can be completed quickly with just the simple first degree hash value calculations.

Even so, there are RDF graphs with special structures containing many blank nodes for which it can take an extremely long time to calculate the n-degree hashes[*19].

For this reason, RDFC-1.0 implementations are required to place an upper limit on the number of n-degree hash calculations and terminate the process prematurely with an error if the limit is exceeded.

You also need to keep in mind that if an RDF graph contains personal data or confidential information, it may be possible to partially infer what that information is based on the canonicalization results. Since the canonicalization calculations are based on data in the graph, the canonicalized names (_:c14n0 etc.) will "partially" contain information from the graph.

Consider, for example, the RDF graph in Figure 8 showing that Alice's spouse is Bob. Say this graph is canonicalized, digitally signed, and saved as shown in Figure 9.

Say that at some point, for whatever reason, Alice wants to communicate that she is married while keeping her spouse's name hidden. Using the selective disclosure mechanism in Verifiable Credentials, she can hide her

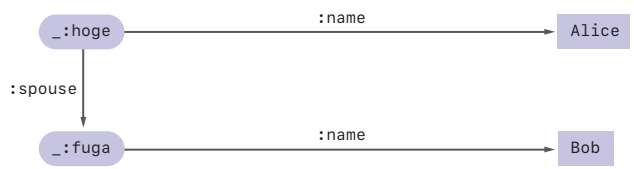

Figure 8: RDF Graph about Alice and Bob

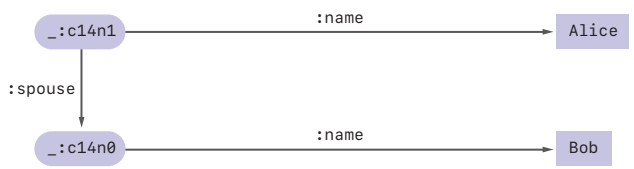

Figure 9: Canonicalized Form of FIgure 8

*19 The specification refers to these as poison datasets. The canonicalization of RDF graphs is known to be as difficult as the graph isomorphism problem, so depending on the input, there will inevitably be cases that require extremely long calculation times.

spouse's name while ensuring her digital signature remains valid, and thus create the verifiable RDF graph shown in Figure 10.

But suppose that a snooper who sees this graph knows only that Alice's spouse is either Bob or Charlie (but not which). This snooper could insert the names Bob and Charlie, first one and then the other, into the hidden part marked by *** and run the canonicalization process again, yielding two graphs with different canonicalized labels, as in Figure 11. By comparing these graphs with the graph that Alice published, the snooper would be able to determine that Alice's spouse is Bob.

This assumes special circumstances, but it is a property that you need to be aware of if using RDF Dataset Canonicalization with verifiable credentials. There is a discussion on avoiding this sort of problem within W3C Verifiable Credentials Data Integrity[*20], a specification for protecting the security and privacy of verifiable credentials.

2.8 Conclusion

I have provided an overview of RDF Dataset Canonicalization, now a W3C recommendation, looking at the specification itself and the standardization effort, which I was involved in. RDF Dataset Canonicalization makes it easier to calculate differences in RDF graphs, check for graph updates, calculate hashes, and generate digital signatures. This can streamline data management and make it possible to imbue RDF graphs with unforgeability and authenticity. As a user of the specification myself, I utilize it in research and development on verifiable credentials and their applications. I hope that this article has piqued your interest in the topic.

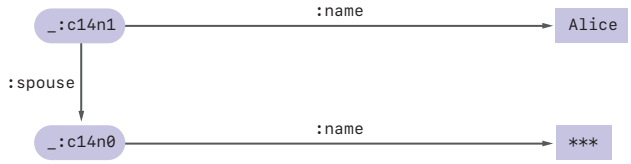

Figure10: RDF Graph with Bob's Name Hidden

Figure 11: Two Different Results

Dan Yamamoto
Senior Engineer, Office of Emergency Response and Clearinghouse for Security Information, Advanced Security Division, IIJ
Dr. Yamamoto has been in his current role since 2021. He is engaged in research on digital identity and information security.

*20 Manu Sporny, Dave Longley, Greg Bernstein, Dmitri Zagidulin, Sebastian Crane: Verifiable Credential Data Integrity 1.0. W3C Candidate Recommendation Draft, April 28, 2024 (https://www.w3.org/TR/2024/CRD-vc-data-integrity-20240428/).