RDF-based Semantics for Selective Disclosure and Zero-knowledge Proofs on Verifiable Credentials

Christoph H.-J. Braun[0000-0002-5843-0316],✉ and Tobias Käfer[0000-0003-0576-7457]

Karlsruhe Institute of Technology, Karlsruhe, Germany
`{braun,tobias.kaefer}@kit.edu`

Abstract. Our work connects the W3C Verifiable Credentials (VC) data model and zero-knowledge proofs (ZKPs) to allow for minimised information disclosure. More generally, as VCs are Resource Description Framework (RDF) datasets, our work enables the application of the following ZKPs on such an RDF dataset: selective disclosure, proof of numeric bounds, and proof of set non-membership – all on the level of RDF terms. To pre-process RDF datasets for such ZKP applications, we introduce schema-free and schema-based approaches, and highlight their differences. We then present a data model for credential presentation and its semantics, where we show that selective disclosure is equivalent to RDF's simple entailment, and that verifications of the presented proofs are validity checks on the processed RDF dataset.

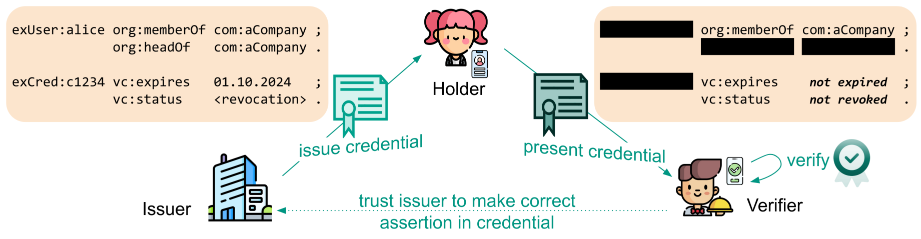
Keywords: RDF · Verifiable Credentials · Zero-knowledge Proofs

1 Introduction

Governments [19] and industry [6] are pushing for the introduction of digital credentials to digitise physical ID cards [23]. In the near future, using digital credentials in the physical world and on the Web will become part of daily life, e. g. for receiving discounts (Fig. 1): Alice is CEO of `aCompany`, which issues a credential attesting that Alice is both member and head of `aCompany`. For lunch, Alice visits a restaurant offering a company discount. To receive the discount, Alice presents her credential to the waiter for verification using her digital wallet. In the process, only the fact that Alice is indeed `aCompany`'s employee needs to be disclosed. No unnecessary information should be revealed, not even the fact that an employee ID card was used. Still, the waiter must be able to verify the credential's integrity, i. e. that `aCompany` indeed signed that Alice is an employee.

To model such credentials, W3C Verifiable Credentials (VCs) [36] provide a standard data model based on the Resource Description Framework (RDF) [17]. Thus, VCs are small personal knowledge graphs that are cryptographically signed by an issuer. Preserving a person's privacy as in our example can be achieved by using cryptographic methods such as zero-knowledge proofs (ZKPs) [22]. This has been strongly advocated by Europe's leading cryptographers [3], which sparked a major discussion[1] about the cryptographic mechanisms in the EU

[1] `https://github.com/eu-digital-identity-wallet/eudi-doc-architecture-and-reference-framework/discussions/211`



```
exUser:alice org:memberOf com:aCompany ;
             org:headOf   com:aCompany .

exCred:c1234 vc:expires   01.10.2024   ;
             vc:status    <revocation> .
```

```
[redacted] org:memberOf com:aCompany ;
           [redacted]   [redacted]   .

[redacted] vc:expires   not expired  ;
           vc:status    not revoked  .
```

Fig. 1: A high-level illustration of our running example.

Digital Identity Wallet [18]. In this paper, we therefore investigate the connection between VCs as RDF datasets and cryptographic methods for ZKPs.

Corresponding cryptographic methods are being implemented, e. g., in AnonCreds (cf. Sec. 6) which only focus on a JSON- and schema-based credential format. They do not consider an RDF-based perspective and its formal logical foundations as relevant. While there has been related work [38] on RDF-based credentials and ZKPs, their approach only allows selective disclosure of complete triples of an RDF graph. Upon presentation, the full triple is either hidden or revealed. To prove that e. g. the expiration date has not yet passed without revealing it is impossible, as the RDF literal is inseparable from its triple.

We thus present a more granular approach that enables ZKPs on RDF term level. Our approach allows to selectively disclose RDF terms within a quad of an RDF dataset, which allows, e. g. to cryptographically prove the equality of RDF terms across quads, or to conceal this equality. We also support proofs of numeric bounds (e. g. expiration date not yet passed) or set (non-)membership on RDF terms (e. g. credential id not revoked), while also proving that the values in both cases were signed by the issuer. Such proofs work on any term of a dataset's quads, and without a schema. Proof composition is not new in cryptography [12], yet to our knowledge, it has not been investigated for RDF, let alone at term granularity. Moreover for our approach, we prove using RDF's formal underpinnings [25] that a presented credential graph logically follows from its original credential graph. Thus, it is logically sound in our approach to apply querying and reasoning on VCs and their presentations, even when they use the advanced cryptography of ZKPs. We provide the following contributions:

– Two approaches to transform RDF datasets for ZKP application (Sec. 3): one schema-free and one schema-based,
– A data model and its semantics for credential presentation (Sec. 4): for selective disclosure of RDF terms, and for associated ZKP verification.
– We show in our evaluation (Sec. 5):
    · selective disclosure is equivalent to RDF's simple entailment
    · proof verification poses validity checks on the processed RDF dataset

We introduce VCs and ZKPs in Section 2. After presenting our approach in Sections 3-4 and evaluation in Section 5, we cover related work in Section 6, and close by highlighting aspects of our work in Section 7. A proof-of-concept implementation of our approach and our vocabulary are available online[2].

[2] https://github.com/uvdsl/rdf-zkp/

2 Preliminaires

In this section, we cover W3C Verifiable Credentials [36] and provide a non-technical introduction to Selective Disclosure using ZKPs [22].

2.1 W3C Verifiable Credentials are RDF Datasets

The Verifiable Credentials (VCs) data model [36] is a W3C Recommendation for modeling digitally signed information using JSON-LD, e. g. digital credentials such as an employee ID card. Such digital credential – and thus a VC as one way to model it – consists of at least two fundamental parts: the information which is claimed to be "true", and the digital signature by the credential's issuer which indicates that the issuer asserts the claims to be indeed "true".

This two-part structure is reflected in the VC data model [36]. A VC is an RDF dataset[3] comprised of two graphs: the *credential graph* and the *proof graph*. The credential graph contains claims and credential metadata. The credential graph on its own is unknown to be "true", i. e. asserted [15]. A claim that a person is a company's employee is meaningless without additional assurance. To decide if the credential graph is asserted/"true", the digital signature of the issuer, e. g. the company, is attached to the credential in the proof graph. Only if the digital signature is valid, and the verifier trusts the issuer to accurately attest the credential's data, the credential graph may be considered asserted/"true".

When showing a credential in the physical world, e. g. the employee ID card to get the lunch discount, the verifier/waiter also physically verifies that the person *presenting the credential* is the same person that the employee ID card was issued to. In the digital world, one way of achieving this is authenticating the credential holder by attaching their digital signature to the VC in presentation. Such additional attachment for presentation form two additional graphs: the *presentation graph* that may include e. g. a usage policy for the presented data and the *presentation proof graph* that indicates that the holder asserts the presented information, e. g. using a signature or cryptographic proof. These two graphs are linked to the VC, thereby forming a *Verifiable Presentation (VP)* [36].

We clarify a technical detail about the interconnections of the different graphs in a VC: The W3C VC data model mandates linking the proof graph (not the proof itself) from a node within the default graph. This implies that the proof specified in the proof graph was calculated over all triples in the (default) graph it was linked from. If two VCs (RDF datasets) would simply merge their default graphs, the proofs of neither would be verifiable anymore. Therefore, we make our choice of semantics [40] explicit: Blank nodes are scoped across graphs of an RDF dataset. When merging two RDF datasets, blank nodes between the two RDF datasets must be re-labelled to avoid co-references, and new graph names must be minted for the default graphs of the respective RDF datasets. We express the opinion that the W3C VC data model should mandate asserting claims in a named graph, and have the proof link to the graph it covers.

[3] For a common formalisation of RDF datasets, see Appendix A.

2.2 Selective Disclosure using Zero-knowledge Proofs

Selective disclosure refers to the idea of proving properties of credential data while hiding (some of) the actual data [7]. When showing a credential in the physical world, e. g. the employee ID card to get the lunch discount, the verifier/waiter sees all the information on the card. They actually only need to see that the presenting person is indeed an employee of an eligible company. Superfluous information should be kept secret such that (a) privacy of the presenting person is preserved and (b) the verifier only receives data that they are prepared and expected to handle. A restaurant would only be expected to handle information on how many customers belong to a company, e. g. receiving a linear payment to offer this benefit to company employees. Handling arbitrary personal information of customers should not be the restaurant's concern.

Considering VCs, a first step towards selective disclosure is to hide or reveal exact information in a VP, e. g. to reveal the employment relation while hiding all other data. A second step is adding proofs on hidden pieces, e. g. prove that the hidden expiration date has not yet passed. These kinds of proofs are enabled by cryptographic protocols that are already mature today: zero-knowledge proofs.

The concept of a *zero-knowledge proof (ZKP)* [22] is that a prover is able to convince a verifier that a statement is true without the verifier learning any additional information. Considering VCs for example, a prover needs to prove that their employee ID card is not expired. The verifier will gain zero knowledge beyond the fact that the prover's credential is still valid.

Technically, this non-expiration proof is a composition of multiple distinct yet connected proofs: As expected, there is a *proof of numeric bounds*. This proof specifically includes proving that the prover knows (a) a certain date and (b) that this date is in the future. But because the prover could just pick a suitable expiration date, there must also be a proof that the picked date is the same as in the employee ID card, signed by the issuing company. Proving the attestation of the expiration date is done using a *proof of knowledge of signature* of the VC (and its content) without revealing the date itself. The fact that both proofs are about the same secret value is ensured by cryptographically tying them together. One way of doing that is to use the *Schnorr Protocol*[4] [34]. The Schnorr Protocol allows us to prove knowledge of a secret value, and subsequently, that the same secret value must have been used in both proofs.

Figure 2 provides a non-technical illustration: Each proof is represented by a seal on an envelope. The values that the proof is about are the content of the envelope. The envelope is indestructible and cannot be opened such that the contents of the envelopes remain secret. The contents of the envelope induce a certain color pattern on the seal, e. g. the date `01.01.2030` induces a blue section in the bottom right-hand corner of the seal. A proof is validated by checking the integrity of the colored seal. When two seals share a specific colored section, the blue bottom-right hand corner, then they share a specific content value, the date `01.01.2030`. The mechanism that ensures that matching color pattern

[4] For a mathematical description of the Schnorr Protocol, see Appendix B.

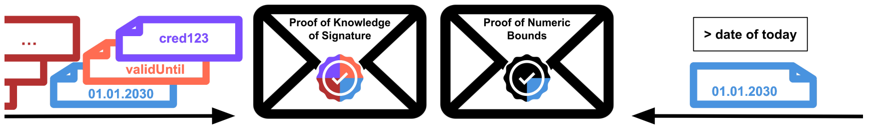


Fig. 2: A non-technical illustration of how two proofs (each represented as a seal on an envelope) are cryptographically tied together (blue color pattern in the seals' right-hand bottom quarter) by the Schnorr protocol (mechanism to ensure matching color pattern on seals) when the proofs are about the same value.

is the Schnorr Protocol. The blue section of the seal represents the so-called *Schnorr response value* that corresponds to a particular secret value. In this way, the Schnorr Protocol allows us to prove knowledge of a secret value – the date `01.01.2030` – and with that, that the same value must have been used in the two proofs – by using its Schnorr response value, the blue section in both seals. Note that a Schnorr response value is not only dependent on the secret value, called *witness*, but also an an additional random value called *blinding factor*, meaning that a single witness can have multiple Schnorr responses using different blinding factors. This allows us to prove or hide the fact that a specific RDF term occurs in multiple distinct RDF triples.

Using this technique, arbitrary combinations of proofs can be arranged. The above is just one example. To present multiple VCs together, proving e. g. that they are about the same person or share a secret value, two proofs of knowledge of signature can be combined. Or, maybe in a non-VC context, a proof of set-membership and a proof of numeric bounds may be combined on the same value.

Circling back to the two levels of selective disclosure: VCs provide information that is attested by the issuer's signature. Initially, using a proof of knowledge of that signature [2], single pieces of the attested information can be disclosed or remain hidden in a VP. Then, additional proofs such as proof of numeric bounds [12] or proof of set (non-)membership [37] can be performed on the hidden yet attested pieces. The glue that keeps it all together is the Schnorr Protocol [34]. On top: A verifier learns no additional information from the ZKPs.

There is, however, a step zero: Creating a VC signature that allows for such proofs to be performed. A simple digital signature covers only a single value (called *message*), e. g. the string value of the VC's credential graph as a whole. To allow for selective disclosure of specific parts of the VC, the digital signature must be a *multi-message* signature that allows for deriving a proof of knowledge of the signature value and of all the messages that were used in its creation. How our work dissects a VC into these multiple messages is presented in Section 3.

3 Preparing W3C Verifiable Credentials for ZKPs

To create a multi-message signature on a VC and then later be able to perform ZKPs on credential data, two conceptual steps are required: First the RDF dataset at hand must be transformed into a list of RDF terms (Sec. 3.1), i. e., a list of terms that we aim to create signature and proofs on. Second, the RDF terms must be transformed to numeric representations such that the cryptographic algorithms of the ZKPs can be applied on these terms (Sec. 3.2).

Recall our running example from Figure 1 and consider Listing 1.1 as the issued credential, which adheres to the W3C VC data model [36] and includes an expiration date and revocation status (Sec. 3.3). Note that Listing 1.1 already includes the signature, a BBS+ multi-message signature [10] (lines 14-18).

```
 1 # claims
 2 <http://example.org/users#alice>
 3   org:headOf <http://example.org/organisations#aCompany>;
 4   org:memberOf <http://example.org/organisations#aCompany>.
 5 # credential metadata
 6 <http://example.org/credentials#cred123>
 7   cred:credentialSubject <http://example.org/users#alice>;
 8   cred:issuer <http://example.org/organisations#aCompany>;
 9   cred:validUntil "2023-11-07T13:20:00Z"^^xsd:dateTimeStamp;
10   cred:credentialStatus <http://example.org/revoc#accumulatorId123>;
11   cred:proof _:signatureGraph.
12 # proof graph
13 GRAPH _:signatureGraph {
14   [] a bbsp16:BbbsPlus16GLDS;
15      cred:verificationMethod <http://example.org/keys#aCompanyKey>;
16      cred:proofValue [  bbsp16:A "g1Em1..."^^xsd:base64Binary ;
17                         bbsp16:e "gq1/f..."^^xsd:base64Binary ;
18                         bbsp16:s "FpwmJ..."^^xsd:base64Binary ] .
19 }
```

Listing 1.1: Our example VC (prefixes omitted for brevity) in TriG notation: Graphs are explicit; JSON-LD notation would hide them in the VC's `@context`.

3.1 Transforming RDF Datasets into a List of RDF Terms

To dissect an RDF dataset into a list of RDF terms, we present two foundational transformation approaches, each stemming from a different perspective on data modeling: VCs may be built with or without using credential schemas [36]. The first foundational approach is schema-free and reflects an RDF purist's perspective who cherishes RDF's flexibility. The second foundational approach is schema-based and reflects the perspective of many JSON-oriented credential system developers. A schema-based transformation is theoretically more efficient as it results in a smaller list of messages to sign; reducing computational expense. However, this approach requires a schema to be linked from the VC and thus may require a remote lookup by a verifier, to determine how to interpret the credential data. With that public schema, certain information, e. g., predicates or property names, cannot be hidden at all. This may lead to inferring information about a credential holder: e. g. from a driver's license schema, when only used for age verification, the verifier may presume that the holder is allowed to drive.

Schema-free Transformation. Our schema-free transformation is based on RDF canonicalisation and dissecting the resulting dataset into atomic elements. First, we canonicalise the dataset following the W3C Recommendation [30]. This includes ordering quads lexicographically in N-Quads serialisation. Then, we dissect the quads into atomic components. We consider the individual RDF

Table 1: Example schema-free (left) and schema-based (right) lists of RDF terms.

["http://example.org/credentials#cred123",	["http://example.org/credentials#cred123",
"https://www.w3.org/2018/credentials#credentialStatus",	"http://example.org/revoc#accumulatorId123",
"http://example.org/revoc#accumulatorId123",	"http://example.org/users#alice",
$\varnothing$, (omitted when trimmed)	"http://example.org/organisations#aCompany",
$\varnothing$, (omitted when trimmed)	"signatureGraph",
...	"2023-11-07T13:20:00Z",
"http://example.org/credentials#cred123",	"http://example.org/organisations#aCompany",
"https://www.w3.org/2018/credentials#validUntil",	"http://example.org/organisations#aCompany"]
"2023-11-07T13:20:00Z",	
"http://www.w3.org/2001/XMLSchema#dateTimeStamp",	
$\varnothing$, (omitted when trimmed)	
...]	

terms' values: URIs and blank nodes each consist of a simple bitstring value: The URI without the enclosing brackets and the blank node's label without `_:`. Literals have a lexical form, optionally a language tag [1], and a possibly implicit datatype. Literals consist therefore of two parts [25], a value and a suffix, which is either a data type or a language tag. Thus, the suffix always specifies the literal's data type. In summary, a quad consists of at most 5 components: subject, predicate, object value, object suffix, and graph name. Canonical ordering of quads and terms is preserved in the list of messages.

We emphasize the distinction between the literal value and the literal suffix because range proofs require numeric comparison. Inclusion of the literal suffix is required because agents need to be informed and assured on how to correctly interpret the literal's value *as intended by the issuer*. Thus, it would be unclear if e. g. value `0x41` should be interpreted as integer `65` or as string `"a"`.

A technical detail provides an optional optimisation: trimming zero elements from the message list to reduce its size. If a term in object position is not a literal, the object suffix component is the zero element ($\varnothing$). If there is no graph name, i. e. for the default graph, the graph name component is the zero element. The resulting list is thus reduced by (a) the number of triples in the default graph, and (b) the number of URIs and blank nodes in object position.

External Schema-based Transformation. Public schemas are commonly used to indicate which information to expect to be asserted in a VC [36]. Taking inspiration from [9] in adhering to W3C recommended specifications, we assume usage of SHACL shapes [29] to be used as RDF schemas. We outline a rough proposal on how to convert a SHACL-based credential schema with an associated claim schema to a value list: By convention in our work, the first element of the resulting value list is the identifier of the credential itself. For credential data, in lexicographical ordering of the predicate URIs the corresponding object values are appended, i. e., URIs and the values of RDF literals without their suffix. Object suffix information is provided by the schema. Similarly then for claim data, the corresponding object values are appended in lexicographical ordering of predicate URIs. For our example, a resulting list of RDF terms is in Table 1.

In this way, only the credential identifier and all the object values need to be signed, proven and verified. There is no need to re-iterate the subject for each triple. In addition, there is no need to include the predicate of each triple and object suffix if present, as these are provided by the public schema anyways. This reduces the number of messages to be signed significantly in trade-off for usage of a public external schema, i. e., the fixed structure of the credential and its predicates are public knowledge. These are then necessary to be obtained for proof verification, which may require a remote call.

3.2 Interpreting RDF terms for Signatures and ZKPs

After transforming the RDF dataset (Sec. 3.1), we need to consider the values of individual RDF terms, either in lexical space or in value space. Only then we can apply the cryptographic algorithms for creation of signature and proof.

URIs and blank node labels are considered in lexical space. Similarly as in a SPARQL `FILTER` expression [24], literals are considered according to the XML Schema Datatypes (XSD) definition [33]. All literals that exhibit the *numeric* fundamental facet [33] are considered in the numeric value space. We note that in an implementation, non-integer-based datatypes require specialised treatment, which does not limit the generality of our approach. Dates also need specialised treatment, see Appendix C. All other literals are considered in the lexical space.

More specifically, we map RDF term representations in the lexical space to sequences of bits (i. e. bitstrings of dynamic length) using UTF-8 encoding [39]. For the numeric value space, we map value representations of RDF Literals to unsigned 64-bit integers (i. e. bitstrings of 64 bits). These bitstrings are then further mapped to field representations of the elliptic curves used for signatures and proofs, as usual in elliptic curve cryptography.

While blank nodes are allowed in VCs, their label may pose a privacy risk, specifically on the security property of unlinkability [32]: A sufficiently unique blank node label may allow a verifier to track that the same credential is used in multiple presentations, and then to identify the presenting holder.

Re-labeling blank nodes for a presentation does not pose a solution: The original blank node labels are used in calculation of the signature value. When simply re-labeling blank nodes in presentation, the original blank node label is not available to a verifier (nor any cryptographic stand-in) and they cannot verify the proofs. There is just no connection between the new labels and the proof. Instead, we cryptographically hide all blank node labels in a VC when creating a presentation. In this case, there exists a Schnorr response value for each blank node which allows for verifying the proof of knowledge of signature (and of the values it was calculated on). Moreover, this enables to cryptographically prove and verify that two blank nodes in the presentation graph refer to the same node in the credential graph, or to cryptographically hide exactly that fact.

3.3 Expiration and Revocation

Once the VC has been signed and handed over from the issuer to the holder, expiration and revocation mechanisms ensure the issuer's control over a VC's validity. The expiration date is specified using a `cred:validUntil` relation (Listing 1.1 line 9) and should be hidden in presentation. If a holder proves this date has not yet passed, a verifier may assume the VC to not be expired.

For revocation, an issuer may manage a Universal Accumulator [37], where the identifiers of revoked VCs are accumulated. Using a corresponding witness, a holder is able to create a proof that the VC's id is not part of the accumulator, and thus that the VC is not yet revoked.

4 Data Model and Semantics in Credential Presentation

Alice presents her credential to the waiter to receive the discount using her digital wallet. The waiter only needs to know that she is indeed an eligible company's employee, and that the credential is not expired and not revoked. All other information should not be revealed; and thus remain hidden. To this end, Alice's wallet derives a presentation of the credential graph (Sec. 4.1) and corresponding proofs (Sec. 4.2) from the VC.

4.1 The Presentation's Credential Graph

Instead of presenting the VC in plain, to be able to selective disclose information of the credential graph, a named graph is derived from the VC's default graph (cf. Listing 1.1 lines 1-11). The terms that are to be hidden are replaced by blank nodes. The remaining terms are revealed. Listing 1.2 lines 2-10 illustrate this selective disclosure of RDF terms.

Recall that a presented credential is an RDF dataset [36]. To hide terms, we cannot simply omit them: If we omitted single terms, the quads would be incomplete. If we omitted entire quads if all their elements are hidden, we could not assert properties about the individual terms in the quad. Thus, when applying selective disclosure and hiding terms, we need a stand-in for the hidden terms.

We use blank nodes for this purpose: Blank nodes are treated as indicating the existence[5] of a thing [25]. Two blank nodes with the same label refer to the same thing. We note that two blank nodes with different labels may or may not refer to the same thing. This ambiguity aids the desired features in selective disclosure of RDF terms: Hidden terms, i. e., hidden messages, are represented by blank nodes in the presented credential graph. Preserving the semantics of blank nodes, we consider all blank nodes to indicate hidden messages. URIs and Literals are revealed messages. To allow for selective disclosure of predicates, we allow blank nodes in predicate position similar to SPARQL [24].

```
1 # the presented credential graph
2 GRAPH _:4 {
3  _:0 cred:credentialStatus <http://example.org/revoc#accumulatorId123>.
4  _:0 cred:credentialSubject _:7 .
5  _:0 cred:issuer <http://example.org/organisations#aCompany>.
```

[5] In a graph not deemed "true", the fact that a blank node indicates existence is meaningless. Recall Sec. 2.1: Whether or not a (presented) credential graph may be assumed to be asserted/"true", and with that the existence of a thing denoted by a blank node, depends on whether or not the verifier trusts the issuer to accurately attest information and on the validity of their digital proof/signature.

```
 _:0 _:16 _:17 .
 _:0 cred:validUntil _:22 .
 _:25 _:26 _:27 .
 _:7 org:memberOf <http://example.org/organisations#aCompany>.
}
# the presentation proof graph
GRAPH <#presentationProofGraph> {
  # Schnorr response values for hidden RDF terms
  _:0 spok:hasSchnorrResponse "YT+CO..."^^xsd:base64Binary.
  _:4 spok:hasSchnorrResponse "zuN2D..."^^xsd:base64Binary .
  _:7 spok:hasSchnorrResponse "QW+Hb..."^^xsd:base64Binary.
  _:7 spok:hasSchnorrResponseForSuffix "m761h..."^^xsd:base64Binary.
  ... # _:16, _:17, _:22, _:25, _:26 omitted for brevity
  _:27 spok:hasSchnorrResponse "hU3Nl..."^^xsd:base64Binary .
  _:27 spok:hasSchnorrResponseForSuffix "CGUG+..."^^xsd:base64Binary .
  # composite proof
  _:cproof a zkp:CompositeProof ;
      zkp:comprisedOf <#poks> , <#rp> , <#snmp> .
  # proof of knowledge of signature over original credential graph
  <#poks> a bbsp16:PoKS;
      bbsp16:hasVerificationKey <http://example.org/keys#aCompanyKey>;
      bbsp16:isProofOfKnowledgeOfSignatureOverGraph _:4 ;
      bbsp16:A_prime "og4GMd...e"^^xsd:base64Binary ;
      bbsp16:A_bar   "oh2fn..."^^xsd:base64Binary ;
      bbsp16:d       "ihCgp..."^^xsd:base64Binary ;
      bbsp16:pi      _:spk1, _:spk2 .
    _:spk1 a bbsp16:SPK1
      spok:hasCommitmentToRandomness "gi01T..."^^xsd:base64Binary ;
      bbsp16:hasResponseValueFor_e   "U2Rte..."^^xsd:base64Binary ;
      bbsp16:hasResponseValueFor_r2  "VrS/w..."^^xsd:base64Binary .
    _:spk2 a bbsp16:SPK2
      spok:hasCommitmentToRandomness     "jWnoZ..."^^xsd:base64Binary ;
      bbsp16:hasResponseValueFor_r3      "6fh2i..."^^xsd:base64Binary ;
      bbsp16:hasResponseValueFor_s_prime "vUfqW..."^^xsd:base64Binary .
  # range proof of expiration date
  <#rp> a lg16:PoRM ;
      lg16:hasVerificationKey <http://example.org/keys#verifierLg16VerificationKey> ;
      lg16:hasWitness _:22 ;
      lg16:hasLowerBound "1383830400"^^xsd:nonNegativeInteger ;          # transformed dates
      lg16:hasUpperBound "18446744073709551615"^^xsd:nonNegativeInteger; # see Appendix C
      lg16:hasProofValue [ ... ] ; # details omitted for brevity
      lg16:pok [ ... ] .           # details omitted for brevity
  # set non-membership proof of credential id (URI) in revocation accumulator
  <#snmp> a uacc:PoSNMP ;
      uacc:hasAccumulator <http://example.org/revocation#accumulatorId123> ;
      uacc:hasWitness _:0 ;
      uacc:hasProvingKey <http://example.org/keys#proverUaccProvingKey> ;
      uacc:hasRandomizedWitness [ ... ]; # details omitted for brevity
      uacc:hasCommitments [ ... ];       # details omitted for brevity
      uacc:hasResponses [ ... ].         # details omitted for brevity
}
```

Listing 1.2: The example credential presentation (in schema-free approach).

4.2 Modeling Zero-knowledge Proofs on W3C Verifiable Credentials

So far, in the *presented credential graph*, RDF terms are revealed or hidden. To be able to verify the validity of the presented information and to perform additional proofs on properties of hidden terms, the corresponding proofs are modelled in the *presentation proof graph*. Our data model[6] thus aims to provide verifier with all the information required for proof verification. In particular, we focus on proof

[6] https://github.com/uvdsl/rdf-zkp/tree/main/vocab/

composition: The proof of a credential presentation is a composition of distinct sub-proofs (Listing 1.2 line 22-23) that are interlinked by proving statements on the same secret witness values, and by shared parameters.

Center of proving that a hidden RDF term is witness across multiple proofs are *Schnorr response values* (see Sec. 2.2). For each hidden RDF term, there exists a Schnorr response value[7] (Listing 1.2 line 14-20).

For proof of knowledge of signature, Listing 1.2 line 25-39 provides a BBS+ proof tuple according to its definition in [10]: $(A', \bar{A}, d, \pi)$. The proof links to its hidden and revealed messages, i. e. the presented credential graph. For proof verification, parameters of the proof itself, verification key, the revealed RDF terms, and the hidden RDF terms' Schnorr response values are required.

To prove properties of the hidden RDF terms, proofs of numeric bounds or set (non-)membership, link to their particular witnesses in the presented credential graph: The proof of numeric bounds, a LegoGroth16 proof (Appendix H of [12]), links to the witness for which the specified upper and lower bounds are proven. The witness is the hidden object value of the `cred:validUntil` relation (Listing 1.2 line 7), and is thus part of the graph that the BBS+ proof of knowledge of signature covers. The proof of set non-membership, using a Universal Accumulator as defined in [37], applies the same principle and is thus linked to the BBS+ proof via the witness' blank node in the presented credential graph.

5 Evaluation

To evaluate our approach, we first look at the semantics for selective disclosure of RDF terms (Sec. 5.1). Then, we examine proof verification as validity checks on the RDF dataset (Sec. 5.2). Last, we compare the two transformations (Sec. 5.3).

5.1 Semantics for Selective Disclosure of RDF Terms

A verifier, who receives a presentation with hidden terms, will process the dataset using querying or reasoning techniques. Thus, a statement about the truthfulness in the logical sense of such query or reasoning results is an open question. We thus aspire a logical connection between the original graph and the selectively disclosing graph: If the truth of the selectively disclosing graph follows from the truth of the original graph, the said querying and reasoning results are true (in the logical sense). We prove[8]:

Theorem 1. *Every selectively disclosing graph can be entailed under simple entailment from the underlying original graph that it selectively discloses.*

Proof. Let $g2m : \mathcal{D} \to (t_1, t_2, \ldots, t_n) | t_i \in \mathcal{U} \cup \mathcal{B} \cup \mathcal{L}$, a transformation function as outlined in Sec. 3.1 from an RDF dataset to an array of RDF terms (the list

[7] There may also exist a Schnorr response value for a quad's object suffix (Sec. 3.1).
[8] See Appendix A and B as a refresher on the usual formalisation of RDF and Schnorr.

of messages). Let its inverse function be $m2g$. Let $\mathtt{I_D}$ be the set of indices of revealed (D̲isclosed) messages from the message list.

From graph G, create a presentation P such that P selectively discloses G:

1. Let $m := g2m(G)$, i. e., graph G to RDF term array (the list of messages).
2. Let $\forall i \notin \mathtt{I_D} : \exists r_i$ (blinding factor) s. t. $\exists i, j \notin \mathtt{I_D} : r_i = r_j \Rightarrow m_i = m_j$.
   We note that $\exists i, j \notin \mathtt{I_D} : m_i = m_j \not\Rightarrow r_i = r_j$ which means that an RDF term that occurs at different indicies may have two distinct blinding factors, e. g., to hide the fact that the same RDF term occurs in two quads.
3. Let $\forall i \notin \mathtt{I_D} : z_i = r_i + c\, m_i$ (the Schnorr response value),
   with c the overall proof challenge which is the same for all messages.
4. It follows immediately: $\exists i, j \notin \mathtt{I_D} : z_i = z_j \Rightarrow m_i = m_j$.
   We again note that $\exists i, j \notin \mathtt{I_D} : m_i = m_j \not\Rightarrow z_i = z_j$.
5. Let B be a set of fresh blank nodes (that are not in G): $\forall i \notin \mathtt{I_D} : \exists b \in B$.
6. If two messages have in the same Schnorr Response value z, then they map to the same blank node in B:
   $t2b : \{i \notin \mathtt{I_D} : m_i\} \to \mathcal{B} \mid \forall (m_i, m_j) : z_i = z_j \Rightarrow t2b(m_i) = t2b(m_j)$.
7. Then replace the hidden terms with their stand-in blank node:
   $m' := m$ and $\forall i \notin \mathtt{I_D} : m'_i := t2b(m_i)$.
8. Finally, convert the array back[9] to an RDF graph: $P := m2g(m')$.
   Thus, P is G with some terms replaced by blank nodes, where multiple blank node may refer to the same RDF term.

Recall the definition of a proper instance I of an RDF graph H, where in H two distinct blank nodes may refer to one term in I [25].

It follows: original graph G is proper instance of the presented graph P.

It follows: original graph G is instance of the presented graph P.

Recall that "a graph is entailed under simple entailment by any of its instances" [25].

It follows: original graph G entails under simple entailment presented graph P.

$\square$

In our example, the original credential graph G (Listing 1.1 lines 1-11) is a proper instance of the presented credential graph P (Listing 1.2 lines 2-10). Notice how blank nodes `_:7` and `_:25` are both mapped to the same node `http://example.org/users#alice`. The presented credential graph P can be entailed under simple entailment from the original credential graph G.

5.2 Proof Verification as Validity Check

Our vocabulary and model aims to supply a verifier with all the information to execute a validity check on the RDF dataset. That is, for an RDF dataset to be a valid representation of a presented credential, the modeled values must satisfy the proofs' mathematical verification equations. We thus ask:
Does our data model support validity checks using proofs' verification equations?

[9] In this step, there exists a matching from object suffix to the blank node of the corresponding object value: The blank node stands-in for the object as a whole.

In this evaluation, we only cover the proof of knowledge of BBS+ signature; the other proofs trivially follow the same line of argument. The BBS+ proof a tuple $(A', \bar{A}, d, \pi)$, and the prover proves as defined in [10]:

$$\pi \in SPK\{(\{m_i\}_{i \in \mathtt{I_D}}, e, r_2, r_3, s') : \bar{A}/d = A'^{-e} * h_0^{r_2} \quad \wedge \quad g_1 \prod_{i \in \mathtt{I_D}} h_i^{m_i} = d^{r_3} * h^{-s'} \prod_{i \notin \mathtt{I_D}} h_i^{-m_i}\}$$

where SPK is short for signature proof of knowledge. For the SPKs, we use the Schnorr protocol (see Appendix B) to prove knowledge of the discrete logs within the equations[10]. The resulting *verification equations* (among other checks) are

$$\mathtt{SPK1} \mid \quad A'^{\mathcal{S}(-e)} * h_0^{\mathcal{S}(r_2)} / (\bar{A}/d)^c = t_{\mathtt{SPK1}} \quad \mid$$

$$\mathtt{SPK2} \mid \quad d^{\mathcal{S}(-r_3)} * h_0^{\mathcal{S}(s')} * \textstyle\prod_{i \notin \mathtt{I_D}} h_i^{\mathcal{S}(m_i)} * (g_1 \prod_{i \in \mathtt{I_D}} h_i^{m_i})^c = t_{\mathtt{SPK2}} \quad \mid$$

where $\mathcal{S}(\cdot)$ is a Schnorr response, t the commitment to randomness and c the proof challenge. g_1, h_0 and h_i are verification key parameters [10]. $\mathtt{I_D}$ is the set of indices of revealed messages: $m_i | i \in \mathtt{I_D}$ is revealed; $m_i | i \notin \mathtt{I_D}$ is hidden.

For the RDF dataset to be a valid representation of the presented credential, every proof's verification equations must hold (including the above). We thus connect our proof model to the corresponding verification equations by creating a mapping from variables to values, e. g. trivially implemented using SPARQL (cf. values from Listing 1.2, e. g., line 34 for $\mathcal{S}(-e)$):

$$\mu = \{\mathcal{S}(-e) \mapsto \texttt{"U2Rte..."\^{}\^{}xsd:base64Binary}, \mathcal{S}(r_2) \mapsto ...\}$$

For all variables from the verification equations, there exists a corresponding value in our model. We then apply the mapping to the verification equations. If verification equations hold for all proofs, then the composite proof is successfully verified: The RDF dataset is a valid representation of the presented credential and its proofs. Our vocabulary thus provides a data model that supports validity checks on the RDF dataset using the proofs' mathematical verification equations.

5.3 Evaluating Transformation Approaches

We presented two foundational approaches to transform RDF datasets to a list of messages usable for ZKP application in Sec. 3.1. We look at potential trade-offs:

Efficiency. We compare the transformation approaches regarding the number of resulting messages to sign, prove and verify. Schema-free transformation always results in a message list whose length is 5 times the number of quads in the RDF dataset. Each quad is dissected into subject, predicate, object value, object suffix and graph name. The optional trimming removes any zero-elements from the message list resulting in a equal or smaller length list. Schema-based transformation results in a message list whose length depends on the schema. In our version, the resulting list is the credential id followed by all object values of the VC. This results in a list length of one fifth of the number of quads, plus one. However, our performance comparison indicates that the theoretical efficiency

[10] After inverting the equation for SPK2 to $g_1^{-1} \prod_{i \in \mathtt{I_D}} h_i^{-m_i} = d^{-r_3} * h^{s'} \prod_{i \notin \mathtt{I_D}} h_i^{m_i}$.

advantage does not translate to practical performance gains (see Appendix D). Therefore, we conclude that the schema-free transformation is not worse (and not impractical at all) compared to the schema-based transformation.

(Proof) Generality. We compare the transformation approaches regarding their support of different kinds of proofs on an RDF datasets' terms. Schema-free transformation, in a generic fashion, enable proofs on any RDF term of an RDF dataset: Full selective disclosure on RDF term level; on subject, predicate, objects and their suffix. The fact that the same term occurred in two RDF triples can also be hidden by assigning two different blank nodes. Proofs of numeric bounds on numeric RDF literals and set (non-)membership proofs on RDF terms are also supported. Schema-based transformation only allows proofs on focal terms of the dataset depending on the schema. Selective disclosure is therefore partial as e. g. predicates are known from the schema, and existence relations between nodes can be inferred from the schema. Numeric bounds and set proofs are supported on the focal terms. Therefore, we conclude that the schema-free transformation offers a more general and thus flexible approach to performing proofs on credential data compared to using a pre-defined schema.

Conformance. We compare the transformation approaches regarding their conformance with the W3C VC data model [36]. For the credential itself, all approaches are compliant. For the presentation, we discuss: Schema-free transformation require that the triples of the presented credential graph are provided in the correct order as the ordering of RDF terms need to be the same for verification as for proof creation. Additionally for proofs on terms in predicate position, the data model must allow blank nodes as predicates, e. g. as SPARQL or Generalised RDF. This is not necessarily an issue: The mandatory serialisation of the VC data model [36] is JSON-LD, which supports Generalised RDF by default[11]. Schema-based transformation does not impose additional requirements. Using the schema (known a priori or looked up), the list of RDF terms can be re-constructed. Predicates are public knowledge; they cannot be hidden. Therefore, we conclude that the schema-free transformation imposes minor syntactical requirements on presented credential graphs in general, while the schema-based transformation requires a specific schema on specific credential instances.

6 Related Work

We do not claim originality of proof composition: LegoSNARK [12] is a framework to combine multiple proofs on the same witness using a commit-and-prove scheme [13,28]. Our work applies such composition of ZKPs to W3C VCs.

Therefore, we now look at related work combining W3C VCs [36] and ZKPs.

First practical work on selective disclosure of RDF triples was introduced by MATTR, resulting in a community draft specification [31]. [38] presents a general formalisation of this approach to ultimately construct RDF-based verifiable digital credentials. As an application of this, [8] aspires privacy-preserving

[11] `https://www.w3.org/TR/json-ld11/#relationship-to-rdf`

authentication and attribute-based authorization on Solid Pods [14]. Recently, a candidate recommendation [5] to achieve selective disclosure based on the Data Integrity specification [35] was published by the W3C VC Working Group. These works are limited to selective disclosure on RDF triple level. They do not consider proofs on an RDF term level, other proof types like numeric bounds, or proof composition in general. Our work, on the other hand, enables these features.

Hyperledger AnonCreds [16] use CL signatures [11] to create “anonymous credentials”. Hyperledger AnonCreds can be constructed such that they can be transformed to a representation compliant with the W3C VC data model [36]. Proof-wise, AnonCreds allow for selective disclosure of credential values (the schema reveals the credential properties), predicate proofs like CL numeric bounds proofs, and revocation in zero-knowledge using CL accumulators. Recent work on the next version AnonCreds 2.0 aims to expand the feature set to support different signature schemes, statement proof protocols and verifiable encryption.

AnonCreds 2.0 is similar to the presented work in that it investigates flexible composition of ZKPs for digital credentials. It is different in that it focuses on a credential data format which is both JSON- and schema-based. Similar to AnonCreds 1.0, a transformation to the VC data model may be possible. But AnonCreds 2.0 does not recognise the RDF-based W3C VC Recommendation [36] as foundational. As such, there is no considerations of the logical connection between the original credential data and the selectively disclosed data – a connection that our work provides. Investigating selective disclosure in credential presentation from an RDF perspective allows our work to prove that applying querying and reasoning techniques on a presented credential is logically sound. On top and at the very least, our work provides an alternative perspective to AnonCreds: Our work shows that VCs can also be built without using schemas.

7 Conclusion

With our work, we seek to bridge the gap between the W3C recommended VC data model [36] and European cryptographers' recommendation for ZKPs [3]. In particular, we aim to contribute to the W3C's ongoing efforts in improving the securing mechanisms of VCs [5,35] by expanding their feature set while providing formal logical underpinnings: We highlighted the semantics for selective disclosure of RDF terms and proved that selective disclosure of terms from the credential graph is equivalent to simple entailment. Our work provides a foundational understanding of how compositions of ZKPs may be applied on terms in an RDF dataset – with or without using a schema.

Our work provides a foundation to build advanced semantic credential systems, e. g.: VC-based anonymous credentials that allow for user anonymity or pseudonymity while offering semantic querying and reasoning capabilities. But even using regular VCs, research on reasoning-enabled access control, i. e. based on entailed relations, would push the frontier of authorization mechanisms.

Acknowledgements This work is supported in part by the German federal ministry of education and research (BMBF) in MANDAT (FKZ 16DTM107B). We thank Ruben Verborgh for shepherding the paper and Sebastian Rudolph for his help with logics terminology. Icons (used in figures) created by Freepik.

A Common Formalisation of RDF Graphs and Datasets

We re-use the common formalisation: Let $\mathcal{U}$ denote the set of all HTTP URIs [4,21], $\mathcal{B}$ the set of all blank nodes, and $\mathcal{L}$ the set of all literals. Let $\mathcal{G}$ denotes the set of all RDF graphs. An RDF graph $G \in \mathcal{G}$ is defined as a set of triples. A triple t is defined as $t \in (\mathcal{U} \cup \mathcal{B}) \times \mathcal{U} \times (\mathcal{U} \cup \mathcal{B} \cup \mathcal{L})$. Let $\mathcal{D}$ denote the set of all RDF datasets. An RDF dataset $D \in \mathcal{D}$ is a set of named graphs and an unnamed *default graph*. A named graph [15] is a couple (n, G_n) where $n \in (\mathcal{U} \cup \mathcal{B})$ and $G_n \in \mathcal{G}$. The default graph does not have a graph name, $(\_, G)$ with $G \in \mathcal{G}$. A triple of a graph G_n within an RDF dataset is also referred to as a quad q with $q \in (\mathcal{U} \cup \mathcal{B}) \times \mathcal{U} \times (\mathcal{U} \cup \mathcal{B} \cup \mathcal{L}) \times \{n\}$, with n being the graph name.

B Schnorr Protocol – Proof of Knowledge of Discrete Log

A prover wants to convince a verifier that they know a secret $x \in \mathbb{Z}_q$, the group of integers modulo q. To this end, the prover calculates $h = g^x$ and makes h public. In general terms, we call $h = g^x$ a *statement* and x a *witness* to that statement. In this proof, h is used as a *verification key* for the proof.

Prover and verifier know that $\mathbb{G}_q$ is a cyclic group of prime order q, g is a generator of $\mathbb{G}_q$, and $h \in \mathbb{G}_q$. The non-interactive protocol under the Fiat-Shamir transformation [20] makes use of a hash function that serves as a *random oracle*. The protocol is as follows:

$$\begin{array}{ccc}
\textbf{Prover} & & \textbf{Verifier} \\
\hline
r \overset{\$}{\leftarrow} \mathbb{Z}_q & & \\
u = g^r & & \\
c = H(g, q, h, u) & & \\
z = r + x \cdot c & & \\
 & \xrightarrow{\pi=(u,c,z)} & \\
 & & u, h \overset{?}{\neq} 0 \;\texttt{\&\&}\; u, h \overset{?}{\in} \mathbb{G}_q \\
 & & z \overset{?}{\neq} 0 \mod q \\
 & & \text{- - - - - - - - - - -} \\
 & & c \overset{?}{=} H(g, q, h, u) \\
 & & g^z \overset{?}{=} u \cdot h^c
\end{array}$$

To prove knowledge of a secret $x \in \mathbb{Z}_q$, the prover has *verification key* $h = g^x$. The prover creates: A randomly sampled $r \leftarrow \mathbb{Z}_q$ referred to as *randomness* or *blinding factor*. Statement $u = g^r$ is called *commitment to randomness*. A *Schnorr response value* is $z = r + c \cdot x$ where c is called the proof *challenge*. The challenge is computed by $c = H(g, q, h, u)$. Its argument list (g, q, h, u)

is the proof's *challenge contribution*. The Schnorr proof of knowledge itself is $\pi = (u, c, z)$ which is provided by the prover to a verifier with the verification key h. The verifier knows (g, h, π) and validates their inputs. Then, the verifier checks the *verification equations* $g^z = u \cdot h^c$ and $c = H(g, q, h, u)$. If true, the verifier is convinced that the prover knows the secret x.

C Treatment of dates and time in signature creation

The W3C VC data model recommends using `xsd:dateTimeStamp` for indicating creation and expiration dates. `xsd:dateTimeStamp` does not exhibit the numeric fundamental facet and thus would be interpreted in the lexical space. With that, range proofs of "later/earlier than" will not work out-of-the-box. Consider `2004-04-12T13:20:00Z`, which is 1:20 pm on April 12, 2004, Coordinated Universal Time (UTC), and `2004-04-12T13:20:00-05:00`, which is the same time, albeit in different time zones. To compare the two, a common "reference point in time" is required. By convention, the time zone offsets could be resolved and all date time stamps could be transformed to UTC. Then, we could compare the lexical values of the adjusted dateTimeStamps as these values are now totally ordered. JSON Web Tokens [27] already solved this issue for their `iat` (issued at) and `exp` (expiration time) claims. Their values must be a `NumericDate` value. `NumericDate` is defined as the number of seconds from `1970-01-01T00:00:00Z UTC` until the specified UTC time ignoring leap seconds [26]. Similar to JWTs, we internally transform an `xsd:dateTimeStamp` value to a `NumericDate` value. In our example, `"2023-11-07T13:20:00Z"` is transformed to `1699363200`.

D Example Implementation

We implemented[12] our example with credential issuance, presentation and verification to evaluate the transformation approaches (Sec. 3.1). In theory, the distinguishing factor should be the message list length as the proof of knowledge of signature becomes more expensive with more messages. Table 2 indicates however that performance difference is negligible. The first number is the time in milliseconds of the overall process and the second number is only the part of creating / verifying a proof in the process. We presume the credentials are too small scale such that any algorithmic difference is not relevant given the hardware; a consumer laptop with an AMD Ryzen 7 PRO 5850U.

Table 2: Performance benchmark using `criterion.rs`.

Transformation	**Schema-free** (no trim)	**Schema-based**
Number of messages	35	8
Prove (incl. RDF transform) / proof creation time	∼158 ms / ∼59 ms	∼157 ms / ∼60 ms
Verify (incl. RDF transform) / verification time	∼114 ms / ∼18 ms	∼116 ms / ∼18 ms

[12] `https://github.com/uvdsl/rdf-zkp/`

References

1. Alvestrand, H.: Tags for the identification of languages. Best current practice, IETF (2001), https://www.ietf.org/rfc/rfc3066.txt
2. Au, M.H., Susilo, W., Mu, Y.: Constant-size dynamic k-taa. In: Proc. of the 5th SCN (2006)
3. Baum, C., Blazy, O., Camenisch, J., Hoepman, J.H., Lee, E., Lehmann, A., Lysyanskaya, A., Mayrhofer, R., Montgomery, H., Nguyen, N.K., Preneel, B., abhi shelat, Slamanig, D., Tessaro, S., Thomsen, S.E., Troncoso, C.: Cryptographers' Feedback on the EU Digital Identity's ARF (jun 2024), https://github.com/user-attachments/files/15904122/cryptographers-feedback.pdf, see also https://github.com/eu-digital-identity-wallet/eudi-doc-architecture-and-reference-framework/issues/200
4. Berners-Lee, T., Fielding, R., Masinter, L.: Uniform resource identifier (uri): Generic syntax. Internet standards track document, IETF (Jan 2005), https://www.ietf.org/rfc/rfc3986.txt
5. Bernstein, G., Sporny, M.: Data integrity bbs cryptosuites v1.0. W3c candidate recommendation draft, W3C Verifiable Credentials Working Group (2024), https://www.w3.org/TR/vc-di-bbs/
6. Bitkom Arbeitskreis Digitale Identitäten: EU Digital Identity Wallet (2025), https://www.bitkom.org/sites/main/files/2025-01/bitkom-whitepaper-organisationsidentitaeten.pdf, Bitkom e.V.. Accessed: 2025-03-11
7. Brands, S.: A technical overview of digital credentials (2002), https://api.semanticscholar.org/CorpusID:18284690
8. Braun, C., Käfer, T.: Attribute-based access control on solid pods using privacy-friendly credentials. In: Proc. of Posters & Demos at the 18th SEMANTiCS. CEUR Workshop Proceedings, vol. 3235. CEUR-WS.org (2022)
9. Braun, C.H.J., Papanchev, V., Käfer, T.: SISSI: an architecture for semantic interoperable self-sovereign identity-based access control on the Web. In: Proceedings of the 32nd Web Conference (WWW). p. 3011–3021. ACM, New York, NY, USA (2023). https://doi.org/10.1145/3543507.3583409
10. Camenisch, J., Drijvers, M., Lehmann, A.: Anonymous attestation using the strong diffie hellman assumption revisited. IACR Cryptol. ePrint Arch. p. 663 (2016), http://eprint.iacr.org/2016/663
11. Camenisch, J., Lysyanskaya, A.: A signature scheme with efficient protocols. In: Revised Papers of the 3rd SCN. LNCS, vol. 2576, pp. 268–289. Springer (2002)
12. Campanelli, M., Fiore, D., Querol, A.: Legosnark: Modular design and composition of succinct zero-knowledge proofs. IACR Cryptol. ePrint Arch. p. 142 (2019)
13. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. IACR Cryptol. ePrint Arch. p. 140 (2002), http://eprint.iacr.org/2002/140
14. Capadisli, S., Berners-Lee, T., Verborgh, R., Kjernsmo, K.: Solid protocol. Version 0.9.0, W3C Solid Community Group (Dec 2021), https://solidproject.org/TR/protocol
15. Carroll, J.J., Bizer, C., Hayes, P.J., Stickler, P.: Named graphs, provenance and trust. In: Ellis, A., Hagino, T. (eds.) Proceedings of the 14th international conference on World Wide Web, WWW 2005, Chiba, Japan, May 10-14, 2005. pp. 613–622. ACM (2005). https://doi.org/10.1145/1060745.1060835, https://doi.org/10.1145/1060745.1060835

16. Curran, S., Philipp, A., Yildiz, H., Curren, S., Jurado, V.M., Bhaduri, A., and, A.I.: Anoncreds specification. v1.0 draft, Hyperledger AnonCreds Working Group (2024), https://hyperledger.github.io/anoncreds-spec/
17. Cyganiak, R., Wood, D., Lanthaler, M.: Rdf 1.1 concepts and abstract syntax. W3C Recommendation, W3C (2014), https://www.w3.org/TR/rdf11-concepts/
18. European Commission: EU Digital Identity Wallet (2024), https://ec.europa.eu/digital-building-blocks/sites/display/EUDIGITALIDENTITYWALLET/, accessed: 2024-09-17
19. European Commission: Regulation (eu) 2024/1183 of the european parliament and of the council amending regulation (eu) no 910/2014 as regards establishing the european digital identity framework (2024), https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=OJ:L_202401183, accessed: 2024-09-17
20. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) Proceedings of the 6th Conferences on Advances in Cryptology (CRYPTO). Lecture Notes in Computer Science, vol. 263, pp. 186–194. Springer (1986). https://doi.org/10.1007/3-540-47721-7_12
21. Fielding, R., Reschke, J.: Hypertext transfer protocol (http/1.1): Message syntax and routing. Internet standards track document, IETF (Jun 2014), https://www.ietf.org/rfc/rfc7230.txt
22. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof-systems. In: Proc. of the 17th ACM STOC. pp. 291–304. ACM (1985)
23. Hanssens, S., Vonner, F., Lisoir, X.: Digital identity (2021), https://www.pwc.lu/en/smart-identity/docs/pwc-digital-identity.pdf, PricewaterhouseCoopers, Société coopérative. Accessed: 2025-03-11
24. Harris, S., Seaborne, A.: Sparql 1.1 query language. W3C Recommendation, W3C (2013), https://www.w3.org/TR/sparql11-query/
25. Hayes, P., Patel-Schneider, P.F.: Rdf 1.1 semantics. W3C Recommendation, W3C (2014), https://www.w3.org/TR/rdf11-mt/
26. IEEE, The Open Group: The open group base specifications issue 7, 2018 edition. IEEE Std 1003.1-2017, IEEE and The Open Group (2017), https://pubs.opengroup.org/onlinepubs/9699919799/basedefs/V1_chap04.html#tag_04_15
27. Jones, M., Bradley, J., Sakimura, N.: Json web token (jwt). Internet standards track document, IETF (May 2015), https://www.ietf.org/rfc/rfc7519.txt
28. Kilian, J.: Uses of randomness in algorithms and protocols. MIT Press (1990)
29. Knublauch, H., Kontokostas, D.: Shapes constraint language (shacl). W3C Recommendation, W3C (Jul 2017), https://www.w3.org/TR/shacl/
30. Longley, D., Kellogg, G., Yamamoto, D.: RDF dataset canonicalization a standard RDF dataset canonicalization algorithm. W3C recommendation, W3C (2024), https://www.w3.org/TR/rdf-canon/
31. Looker, T., Steele, O.: Bbs+ signatures 2020. Draft cg report, W3C Credentials Community Group (2023), https://w3c-ccg.github.io/ldp-bbs2020/#the-bbs-signature-suite-2020
32. National Security Agency: Common Criteria for information technology security evaluation (CCMB-2017-04-002) (2017), https://www.commoncriteriaportal.org/files/ccfiles/CCPART2V3.1R5.pdf
33. Peterson, D., Gao, S., Malhotra, A., Sperberg-McQueen, C.M., Thompson, H.S.: W3C XML schema definition language (XSD) 1.1 part 2: Datatypes. W3C recommendation, W3C (2012), https://www.w3.org/TR/xmlschema11-2/
34. Schnorr, C.: Efficient identification and signatures for smart cards. In: Brassard, G. (ed.) Proceedings of the 9th Annual International Conference on Advances in

Cryptology (CRYPTO). Lecture Notes in Computer Science, vol. 435, pp. 239–252. Springer (1989). https://doi.org/10.1007/0-387-34805-0_22

35. Sporny, M., Longley, D., Bernstein, G., Zagidulin, D., Crane, S.: Verifiable credential data integrity 1.0. W3c candidate recommendation draft, W3C Verifiable Credentials Working Group (2024), https://www.w3.org/TR/vc-data-integrity/
36. Sporny, M., Noble, G., Longley, D., Burnett, D.C., Zundel, B., Hartog, K.D.: Verifiable credentials data model v1.1. W3C recommendation, W3C (2022), https://www.w3.org/TR/vc-data-model/
37. Vitto, G., Biryukov, A.: Dynamic universal accumulator with batch update over bilinear groups. IACR Cryptol. ePrint Arch. p. 777 (2020), https://eprint.iacr.org/2020/777
38. Yamamoto, D., Suga, Y., Sako, K.: Formalising linked-data based verifiable credentials for selective disclosure. In: IEEE European Symposium on Security and Privacy, EuroS&P 2022 - Workshops, Genoa, Italy, June 6-10, 2022. pp. 52–65. IEEE (2022). https://doi.org/10.1109/EUROSPW55150.2022.00013, https://doi.org/10.1109/EuroSPW55150.2022.00013
39. Yergeau, F.: Utf-8, a transformation format of iso 10646. Internet standards track document, IETF (Nov 2003), https://www.ietf.org/rfc/rfc3629.txt
40. Zimmermann, A.: RDF 1.1: On semantics of RDF datasets. W3C working group note, W3C (Feb 2014), https://www.w3.org/TR/rdf11-datasets/