Zero-Knowledge Validation for an Offline Electronic Document Wallet using Bulletproofs

Michael Brand
School of Computing Technologies, RMIT University
Melbourne, Vic, Australia
INCERT GIE
Leudelange, Luxembourg
michael.brand@rmit.edu.au

Benoît Poletti
INCERT GIE
Leudelange, Luxembourg
bpoletti@incert.lu

ABSTRACT

We describe designs for an electronic wallet, meant for the housing of official government documents, which solves the problem of displaying document data to untrusted parties (e.g., in order to allow users to prove that they are above the drinking age). The wallet attains this goal by employing Zero-Knowledge Proof technologies, ascertaining that nothing beyond the intended information is ever shared. In order to be practically applicable, the wallet has to meet many additional constraints, such as to be usable in offline scenarios, to employ only widely-accessible communication methods which, themselves, must not impinge on the user's privacy, and to be constructed solely over standard, widely-studied cryptographic algorithms, offering appropriately high levels of cryptographic security. We explain how our design was able to successfully meet all such additional constraints.

KEYWORDS

electronic wallet, zero-knowledge proof, bulletproof

1 INTRODUCTION

While traditionally physical certificates such as one's identity card, driver's license, student card or pensioner's card have been one's methods to ascertain one's identity and one's eligibility for particular services, the advent of electronic document wallets aims for such documents to be storable, managed and retrieved in their digital forms. New legislation, such as for example the European electronic IDentification, Authentication and trust Services (eIDAS) framework [17, 22, 32] and its counterpart projects in the various EU countries, are driving the wide-spread adoption of digital tools for identity verification, and as an extension of this also the management of academic diplomas [2, 5, 6] and beyond [7].

The introduction of such novel technologies is difficult, in part for two reasons. First, the digital technologies must demonstrate that they provide at least the same levels of protection, versatility and manageability as their physical counterparts. Second, they must be able to function within the same legal environment as their physical counterparts [31].

The main thrust of digital wallet research currently focuses on blockchain technology, such as in the case of the European Blockchain Services Infrastructure (EBSI) [14, 47], but this choice, too, is not without its inherent challenges [19].

For example, single-repository approaches, of which the blockchain-based approach is one, inherently assume that actors can access the ledger. What if we want to enable one to present and validate one's documents offline, in scenarios in which neither document presenter nor document validator have Internet access, and where they can only rely on the data stored locally, on their own smart devices?

In this paper we describe an alternative approach, which overcomes some of the issues present in the traditional solution methods. Not only does this alternative method support offline verification, and therefore greater versatility in the potential scenarios in which the technology can be put to use, it was also specifically designed to address the complications presented by "untrusted" document verifiers, and by this provides inherently more privacy than traditional solutions.

To explain: while many standard solutions may be appropriate to handle a situation in which, e.g., a traffic police officer requests a motorist to present their license, not all such solutions are appropriate in order to handle a scenario in which a bartender asks a customer to verify their age.

The latter scenario is one in which the customer has very little trust of the verifying bartender. Not only would the customer want to minimise the amount of information exposed to the bartender (i.e., merely prove that they are above the drinking age, rather than present their full date of birth, or even a complete ID card containing their name, address and other details), one should also take into account the possibility that such verifications may be recorded and collated, possibly across many venues. The customer may want to avoid such recording from enabling anyone to track their whereabouts and/or profile them.

In this paper we describe an algorithmic solution that tackles all these additional challenges, and can serve as the basis for a production-ready digital wallet.

We describe this solution by presenting a succession of three systems. The first system describes the building blocks and interactions that the design is meant to support. We use it in order to introduce these in a simple environment. We show that this simple solution meets the vast majority of our requirements, but point out where it does not. Namely, the simple solution does not completely eliminate a document verifier's ability to track a document holder from one verification to another.

The second system showcases the main idea of our solution, which is namely the use of general-purpose Zero-Knowledge Proof

PoPETs YYYY(X), 1–10

https://doi.org/XXXXXXX.XXXXXXX

(ZKP) protocols [24, 26] to eliminate entirely a verifier's ability to track a document holder from one verification to another.

While the second system ticks all our boxes from a theoretical point of view, on a practical level the implementation is a little too slow for commercial use. The third system solves this through the use of a number of optimisations, of both algorithmic (number theoretical) and programmatic nature, dramatically improving the system's performance.

We note that many ZKP implementations in the literature rely on the re-implementation of standard protocols but with slight alterations that make them more amenable to ZKP. For example, it is common to replace general hash functions, which traditionally rely on significant amounts of bit mixing and mixed-field arithmetic, with more ZKP-friendly hash functions, such as ones based on Pedersen hashes [3, 18, 27]. In our case, however, because our aim is to enable the practical use of these cryptographic tools in large-scale deployments, we restricted ourselves to the use of only long-standing cryptographic algorithms in their original, unaltered and heavily studied forms.

In Section 2, we describe the general context for the system, including the relevant actors, terminology, etc.. In Section 3, we describe the initial "placeholder" system. In Section 4, we describe the basic ZKP-based solution. In Section 5, we then describe the final design. Follow up research plans are described in the final summary section, Section 6.

2 CONTEXT

2.1 The actors

For our purposes, a digital wallet system involves three parties:

- A *user*, who installs the digital wallet app on their smart device and uses it to host virtual documents, as well as to present them, or some data derived from them, to relying parties;
- A *relying party*, who accepts digitally-signed data served by the user's app, validates it, and provides some required service on the basis of the data presented; and
- A *document issuer*, acting also as a *wallet authority*, which provides the data used by the user app.

Each of the actors listed above interacts with the system by electronic means. This includes the user's device, the relying party's device, and the wallet authority's system.

The *user's device* is a smart device running the digital wallet app. This app supports the following functionalities.

Onboarding: The generation of an *ID certificate* that identifies the user within the system;

Normal usage: Allowing the user to present their data to relying parties; and

Administration: Management of the documents within the wallet, such as by adding more documents or removing existing ones.

Here, the generation of an ID certificate may involve multiple steps and actors, but one essential step is the generation of a *private/public identifier pair*, such that the private identifier is only known to the user's app, and the public identifier is communicated to the wallet authority and serves as the user's identifier within the system. Note that we refer to these as *identifiers* rather than *keys*. It is not necessary for a user to be able to use these identifiers for functions such as encryption or decryption. The only requirement is that the private identifier is the pre-image of the public identifier in a one-way, collision-resistant hash function.

Similar to the user's device, the *relying party's device* is a smart device used by the relying party. The relying party's device runs dedicated certificate verification software to verify the data presented by users in the course of normal usage. In our design, relying parties need no onboarding, no authentication, and no identifiers of their own. Anyone can become a relying party and request to see documents: it is the user's choice who they show which documents to. This is in no way a limitation of the system, because the same technology that identifies a user can also be used to identify relying parties. This makes the management of approvals for relying parties a deployment choice rather than a design choice.

Lastly, the *wallet authority's system* is the software system used by the wallet authority to

- Onboard users;
- Issue certified documents; and
- Publish any additional information required for the running of the digital wallet.

2.2 Assumptions and constraints

To be viable, our solution must meet the following conditions.

- Though users and relying parties may interact with the wallet authority for onboarding and administration, normal usage must be supported without any online interaction. A user must be able to present certificates to a relying party without the presence of any network connection. (We refer to this as *the offline assumption*.)
- As hinted by the offline assumption, the system is meant for physically-present parties to exchange information. A design constraint is that the system must ascertain, when a document is presented, that the user presenting the document is physically present.
- The device and the app of the relying parties are not considered trusted. No sensitive information, beyond what a user explicitly intends to share, should be transferred to the relying party. In particular, to the degree that the documents shared are not personally identifiable, the system must support anonymity in document presentation, protecting the user against any potential attempts by relying parties to track the user based on their document presentation data. This protection must continue even if the relying party uses malicious software on its device.
- The system should be designed for intensive use, allowing users to present documents frequently. As a result, it is imperative that the act of document presentation and validation itself is as lightweight and convenient as possible to both the users and the relying parties.

The constraints above are balanced by the following assumptions.

- We assume that the user app can generate a private identifier, and that the software and hardware protections provided by the smart device are enough to keep this private identifier

secure, even from the device's own user. In other words, if a software is able to access this private identifier, this is enough to ascertain that it is the legitimate, untampered, user app.
- We assume that the initial installation of the user app, including the generation of the private identifier, can be performed in such controlled conditions so as to ensure that the public identifier delivered to the wallet authority was generated by the authentic user software (and its corresponding private identifier is therefore kept securely on the user's device, not accessible even to the device's own user).
- As a simplifying assumption for the purposes of the present system, we do not concern ourselves with resilience against quantum attacks. The system described uses classical cryptographic primitives.

3 A BASIC SYSTEM

3.1 General

Our initial system implementation, presented here for clarity rather than as a true contender, utilises a variety of cryptographic tools.

For public/private key operations, it uses points in the additive Curve25519 [9] twisted Edwards elliptic curve group [10]. For the user, the private identifier is an integer, x, modulo the group size, and the public identifier is Gx, where G is the default group generator. We use Ristretto [42], a method to eliminate cofactors extending on Decaf [29], as our method to represent elliptic group members, thus establishing a one-to-one connection between x and our representation for Gx.

The wallet authority uses a similar pair, (w, Gw), as its private and public key. The public key of the wallet authority is known to all users and all relying parties. (In our implementation, this public key is available as part of the software apps for these parties. More generally, one can think of such public keys as available via the wallet authority's website, and downloaded by users and relying parties ahead of any document verification.)

Our implementation of algorithms was in C, and for cryptographic operations the libsodium library [11] and OpenSSL [48, 49] were used. In particular, we used the Ristretto implementation available in libsodium.

These C algorithms were later wrapped up in a Kotlin language [30, 35, 40] implementation for deployment on Android devices.

One of the critical design constraints of the system was the need for all communication between a user's app and a relying party's app to be fully privacy-preserving. In addition to Internet-based communication being disallowed by the offline assumption, this requirement for anonymity in document presentation excludes any protocol in which the parties need to electronically identify themselves in any way. Thus, WiFi, Bluetooth and many other potential communication protocols are all off the table.

To meet this constraint, all communication used by the system in document presentation and verification is solely based on optical means: the communicating device displays an image on the device's screen, which is then captured by the reading device's camera. This ensures that no data other than what is explicitly visible can be captured in any information exchange. The initial system uses QR codes [33, 44, 46], specifically, for communication in both directions between the user and relying party.

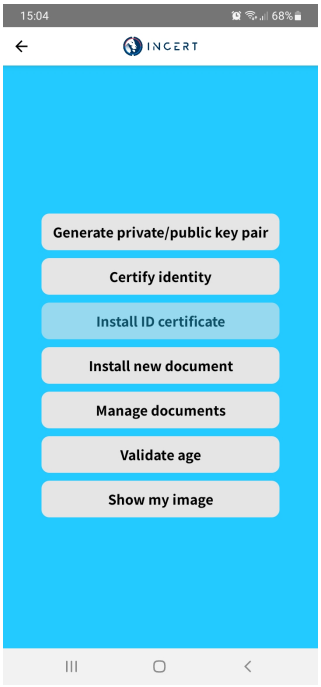

Figure 1: Screenshot of the main menu in an Android implementation of the software.

Balancing the privacy advantages of the optical communication system, QR codes have the drawbacks that they are fairly cumbersome to use, and that even the largest QR codes cannot transfer more than 2953 bytes. The former of these drawbacks forces the design to use only the minimal number of data transfers in the protocol, while the latter restricts what can be communicated in any one transfer.

3.2 Document verification

The core functionality of the digital wallet is to enable a user to present certified data about themselves to a relying party. For the relying party to be convinced of the veracity of the information, it must be able to ascertain all of the following:

(1) That the document presented is authentic;
(2) That the document presented relates to the true owner of the user's device; and
(3) That the presenting user is the true owner of the user's device.

To attain these goals, the system employs the following techniques.

First, all documents in the wallet are digitally signed by the wallet authority using EdDSA [12, 13] before they are issued to the user. A relying party can ascertain the document's authenticity by verifying this signature.

Second, recall that at onboarding the user app generates a private identifier and shares the corresponding public identifier with the

wallet authority. Later iterations of the design presented in this paper eliminate the need for this public identifier to be shared with the relying party, but in this initial design the public identifier is part of what is signed by the wallet authority and transmitted (via QR code) to the relying party. To convince a relying party that the document presented relates to the true owner of the user's device, we only need to prove that the device holds the private identifier corresponding to the public identifier in the signed document that is presented. As a first use of ZKP in the design, we utilise the Chaum-Evertse-van-de-Graaf (CEG) ZKP protocol [16] for the zero-knowledge ascertainment of the knowledge of a discrete logarithm. The constraints of QR code size allow us to run the CEG protocol in parallel 80 times, thus providing 80 bits of confidence to the relying party that the user's app has access to the private identifier corresponding to the public identifier in the signed document. Not only does this prove that the document presented refers to the owner of the presenting device, it also proves that the presenting user's app is authentic and untampered. (In order to make sure also in other wallet-related operations that only the authentic user app can be used, all wallet documents are stored using AES-256 [41] in CBC mode [20], where the encryption key used is based on a SHA3-512 [28] hash of the private identifier. Thus, only the authentic user app, in possession of the user's private identifier, can manipulate the wallet data in any way.)

This leaves us only with the need to authenticate that the user physically presenting the document is, indeed, the user to whom the presented document was issued, i.e. the true owner of the user's device. How this is verified is discussed in Section 3.3.

3.3 User identification

So far, we have used cryptographic primitives in order to verify every aspect required of the presented documents that is in the digital domain. Our one remaining issue to be validated is in the physical domain: whether the person physically presenting a device, which in turn is communicating a piece of knowledge, is truly the owner of the device. This, in turn, is equivalent to asking whether the person who is the object of the document presented is the same person as the one presenting the document.

To resolve this issue we use two types of controls:

(1) The user app itself identifies the user by biometric means, and
(2) Optionally, the relying party may request to see an identifying picture of the person, for manual verification of the same.

Note that because of our offline assumption, it is not possible for this type of validation to be performed over any network, so it must be the responsibility of one of the two devices participating.

Because biometric identifiers (including a person's image) are highly sensitive, personal identifiers (e.g., because they can never be changed by the user), we chose not to entrust them at any point to the "untrusted" relying party. Instead, once we have verified, in the digital domain, the integrity and authenticity of the user's app, we trust this app to

(1) Perform biometric identification of the user based on biometric information that was stored at enrolment time, and
(2) At request of the relying party, present on the user's device an image of the app's true user, based on information that was, as before, stored at enrolment time, and which was signed by the wallet authority.

This separation between digital authentication and physical authentication is critical, because it allows us to connect a user's physical identity solely to one, secret item of information, namely the user's private identifier, from which point all other data stored in the wallet can merely refer to this one identifier and contain no other user identifier.

As a comparison, consider what a customer must do in the physical-document world in order to convince airport personnel that they have a boarding card to a particular flight. It is not enough to present a boarding card that states "I am seated in 23C", because the verifier cannot ascertain from this who this "I" refers to. Instead, the customer must present a boarding card that identifies them by their full name, and this name is then validated against yet another document, such as a passport, containing even more personal details.

Instead, in our implementation the wallet authority needs to ascertain with high confidence only once the connection between the physical user and their digital identifier. Once validation of this connection through biometric means has been set up, the only information the user needs on their documents in order to demonstrate that a document is about them is their one public identifier (knowledge of the private identifier to which they can prove, digitally, through zero-knowledge protocols).

This techniques allows us to shard the data from any official document to its atomic constituents, meaning the minimal pieces of information that a user may wish to present. Thus, in our system, a customer wishing to prove to a bartender that they are above the drinking age only needs to present that one bit of information, without having to expose even their full birth date, let alone any other identifying details.

This connection between the user's physical identity and their public digital identifier, which is documented and signed by the wallet authority and installed by the user on their device, is what comprises the user's ID certificate. It is produced, signed and installed as part of the onboarding process, which happens before the wallet can be activated for normal use. Producing it may require the user to physically identify themselves to the wallet authority. Notably, for onboarding, the wallet authority only needs the user's public identifier. The private identifier is never shared, and, indeed, is not known even to the users themselves.

In our implementation, the wallet authority, which is a trusted party, is given access to the user's image in order to digitally sign it. This is important in order to establish with high confidence this critical link between the physical user and their public identifier. Technologically, however, it should be noted that it would have also been possible to implement for this purpose a "blind signature" protocol [34], in which the wallet authority signs the data without being digitally exposed to it, thus alleviating even the need for the wallet authority to have access to the user's biometrics. As long as sufficient confidence in the veracity of the signature is maintained, blind signatures remain a possibility.

Either way, in our protocol the wallet authority does not (and should not) retain any copy of the user's physical identifiers after the initial creation of the identity certificate. Having said this, the privacy implications of exposing the wallet authority to the user's image are minimal, given that the digital documents stored in the wallet are meant to replace physical documents, traditionally issued by the same authority, and traditionally already containing in many cases images of the user.

3.4 Improvements

When presenting this first system initially, we stated that it is presented here as a basis to build on, not as a true contender in its own right.

In the next sections, we describe our heavier, full-ZKP solutions, which address the initial system's issues in more thorough, qualitatively-better ways. However, before doing so we close the present discussion by presenting several fairly lightweight, point-modifications, that were implemented over the initial system and which, nevertheless, substantially improve its overall appeal.

3.4.1 Multiple public identifiers. The one Achilles heel of the design, as presented so far, is that in order for the user to convince the relying party of the veracity of their documents, said user needs to produce a signed document that contains not just the information they wish to convey but also their public identifier.

This is problematic because an untrusted relying party may end up harvesting the public identifier, using it to track the user, profile them, etc., specifically violating one of our design constraints.

In Section 4 we will show how such public identifiers can be omitted altogether from the information presented to the relying party. However, we will also discuss the costs that such an approach has.

In our initial implementation, which is extremely lightweight when it comes to certificate sizes, response times, etc., we nevertheless were able to reduce dramatically the problems associated with having such a public identifier, without this costing in system performance.

The way to do so was to establish, at enrolment time (i.e., when creating the identity certificate) not one private/public identifier pair but a large number of them. In our specific implementation, at enrolment $1,000$ such pairs are created.

At document verification, a user still needs to present a public identifier, but can now choose randomly any one of the $1,000$ available pairs.

Even if a user presents their certificates daily through the system, any attempt to track the user via their identifiers will only see identifiers recurring once every 3 years. This makes data harvesting essentially useless in this case, noting that the individual identifiers are all randomly chosen, so on their own carry no information.

3.4.2 A two-message Fiat-Shamir protocol. So far, of the assumptions and constraints listed in Section 2.2, the only one that has not yet impacted the design is the assumption of intensive use. This stipulates that using the system should not be cumbersome.

In fact, for privacy reasons we have opted to use QR codes, which are a more cumbersome way to communicate than the more common alternatives.

In order to satisfy the intensive use assumption, we aimed to implement our algorithms in a way that minimises the number of QR codes used per document validation.

To explain, let us first consider what communication happens during document validation. This can be divided into two parts:

(1) The shared information is transferred from the user device to the relying party's device, and
(2) The veracity of this information is ascertained by use of a zero-knowledge protocol.

In this first system, the protocol used is CEG. In the later sections, this protocol is replaced by a Bulletproof protocol [15]. In both cases, however, the protocols share a similar structure:

(1) The prover (in our case the user) performs a zero-knowledge cryptographic commitment;
(2) The verifier (in our case the relying party) responds with a random challenge; and
(3) The prover sends a response to the challenge.

In CEG, one such pass over the three steps is enough to provide one bit of confidence in the proof, and multiple passes (of which we perform 80) can be run in parallel. This results in a protocol with 3 communication steps (noting that the user can also send the shared document details during the first step without this increasing the total number of communication steps).

In the Bulletproof protocol there are significantly more steps, but the structure is the same: at any point in the protocol, all the prover's past communications can be viewed as commitments, and the verifier sends additional random challenges in order to probe whether the prover can respond to them correctly. In Bulletproofs, once the steps of the protocol are done, this proves the claim to all but a negligible probability of error.

In recent years, in part due to the rising popularity of blockchain and the wish to encode zero-knowledge proofs on the blockchain ledger, demand has risen for non-interactive ZKPs. Of these, the currently most popular ones are zk-SNARKs [25, 37] and zk-STARKs [4].

For ZKP protocols following the template of CEG and Bulletproofs as presented above, a common way to transform them from interactive to noninteractive is through the use of the Fiat-Shamir heuristic [23]. Essentially, the prover can complete the entire protocol on its own, substituting in place of the verifier's random challenges deterministic challenges that are based on a cryptographic hash of the entire transcript of the ZKP protocol up to this point.

A verifier looking at the proof can ascertain the correctness of the hash, and by this be convinced that the challenges could not have been chosen maliciously, nor could they have been chosen before the protocol's prior commitments.

In our case, non-interactivity is not an option because we are expressly trying to ascertain the physical presence of the user (as well as the user's device and the app aware of the user's private identifier). We therefore need interactivity. However, we still want to minimise the amount of it, in the sense of minimising the number of communication steps.

Unfortunately, if even one challenge by the verifier is not a Fiat-Shamir challenge, this induces a minimum of 3 communication steps, as per the list above.

To overcome this, we have implemented what we refer to as a *two-message Fiat-Shamir protocol*. In this protocol we introduce a brand new random verifier challenge, which does not exist in the CEG or Bulletproof protocols, or in other zero-knowledge protocols following the same template. This challenge is sent by the verifier to the prover prior to any other communication, i.e. without any prior commitment by the prover.

After this one true challenge by the prover, the verifier continues the protocol as usual, using the normal Fiat-Shamir heuristic for all later verifier challenges. The critical point, however, is that the initial, real challenge is part of the protocol transcript, and therefore impacts all later Fiat-Shamir challenges (because these are all based on hashes computed over the protocol transcript).

In this way, we are able to reduce our communication to only two steps: we begin with the relying party sending a random challenge as one QR code, and the only other QR code needed is the prover's response, containing all other required information.

This is clearly the minimal number of communication passes needed for any interactive protocol.

We use this optimisation both here and in the later Bulletproof-based solutions.

Notably, the first communication step, in addition to the random challenge, may include any other information the relying party may need to convey to the user. In our implementation, the relying party uses this communication step to also specify electronically what document type it is requesting.

3.4.3 JAB codes. In a two-message Fiat-Shamir protocol, the first challenge does not need to be particularly large. We use a 128-bit challenge. It is only the response that carries the important information, and fills the majority of what a QR code can carry.

Unfortunately, in practical experimentation we discovered that scanning a large QR code that is displayed on a device's screen can only be done when the reading device is a fairly high-end device, and even then may be somewhat fiddly in difficult lighting conditions. This makes such large QR codes less suited for the intensive use scenario we wish to support.

Even though this only affects the choice of device for the relying party, not for the user, we decided to solve this problem altogether, which we did by switching from QR codes to JAB codes [8]. JAB codes are an 8-colour variant of QR codes, because of which they require only 1/3 the number of screen pixels in order to communicate the same number of bits. JAB codes, like QR codes, are based purely on visually scanning a user's device, so changing from one code type to the other has no implications on the privacy of the design, but their more compact bit representation allows even low-end devices to be used in our protocol for communicating our challenges and challenge responses.

This improvement, too, was carried over to all our later designs as well.

4 A FULL ZKP IMPLEMENTATION

4.1 General

The second version of the digital wallet which we implemented aims to capitalise on all the advantages of the original design (and, indeed, reuses as many as possible of the ideas presented in the previous section), but does so without ever exposing the user's public identifier (or any other traceable identifier) to the relying party.

The way to do so is to implement the document presentation as a single Bulletproof protocol, implemented (as explained in Section 3.4.2) as a two-message Fiat-Shamir protocol. Bulletproofs, specifically, were chosen because they are essentially unique in that they are general-purpose ZKP protocols whose entire proofs nevertheless fit comfortable even on a QR code.

The basic idea is that instead of a CEG protocol, which can only determine whether a user knows the private identifier matching a given public identifier, we use a Bulletproof, which is a general ZKP protocol, able to prove any claim in the computational class NP [45].

With Bulletproofs, the user can therefore now claim, in zero knowledge, the statement "I have a document, signed by the wallet authority, certifying X to the individual with public identifier Y, and I know the private identifier matching Y (proving that I am this individual)," and can do so whilst only divulging X (i.e., the information the user wants to share) and not Y (the public identifier which we wish to keep known only to the user and the wallet authority).

Because this statement is in the computational class NP, we know that a Bulletproof can be devised for it. Unfortunately, such a Bulletproof, formulated in this native form, would not be efficient. Bulletproofs work on what is known as a *rank-1 constraint system (R1CS)* representation of NP problems. This is a system of equations involving linear relationships between variables, as well as triplets of the form "$x_R x_L = x_O$", encapsulating a multiplicative relationship between variables, where variables are integers modulo the size of the elliptic curve used in the implementation (which, in our case, is Curve25519). The costs associated with a Bulletproof are largely to do with the number of multiplication gates used. The problem is that not all NP problems are equally amenable to being represented as an R1CS with a small number of multiplications.

For example, in the system described in Section 3, the relationship between a user's private identifier and their public identifier was that if the private identifier is x, the public identifier is Gx, where G is a base element in an elliptic curve. Unfortunately, computing Gx using an R1CS is quite heavy.

In this section, we describe the specific changes that were made to the initial design in order to allow convenient R1CS representation, and how this was then implemented.

We remark that while the "proof" part of a Bulletproof is extremely succinct, Bulletproofs do require the prover and verifier to agree on substantial amounts of public randomness as part of a one-time setup. In many scenarios this is a drawback, because no trusted third party exists to ensure that the public randomness is generated fairly. In our case, however, no such problem exists because the wallet authority can provide the public random bits (e.g., by publishing them on its website, or, as in our implementation, by bundling them as part of the wallet software). The reason that there is no cryptological concern around the wallet authority generating the public randomness is that even if the wallet authority chooses this public randomness maliciously, the only possible adverse effect of this, from a cryptological perspective, is that the wallet authority can allow a user to make a false "proof" that they own a specific

certificate; however, such a malicious use of the public randomness would be entirely unnecessary, because, if the wallet authority wishes to allow a user to make such a claim, it can simply provide said user with the relevant certificate, instead.

4.2 The basic idea

In general, any relationship that can be represented through Boolean logic can be transformed into an R1CS. Unfortunately, this requires a multiplication for every AND gate and every OR gate. For efficiency, one wants to turn to an algorithm that natively represents itself through arithmetic operations.

Even more specifically, one wants to turn to an algorithm that natively expresses itself through arithmetic operations in a field that has the same size as the underlying elliptic curve.

For this reason, many ZKP implementations of classic cryptographic algorithms tweak them in some ways in order to rely less on Boolean logic and more on arithmetic in the given field. Replacing general hash functions with functions based on the Pedersen hash is a classic example of this.

In our case, however, because we want to use only untweaked, longstanding algorithms, we were not able to follow in this route.

Instead, we built our protocols based on algorithms that are natively low-degree arithmetic functions.

Specifically:

- For private and public user identifiers, we used the hardness of the factorisation problem: the private identifiers are chosen as two prime numbers of length 1024 bits, u_p and u_q, and the public identifier, u_n, is their product.
- Instead of EdDSA, we use RSA signing [39].

Specifically, let the document information be I (represented in our case as 132 bytes of data) and let P be an additional 992 bits of random padding. Furthermore, let M be the concatenation of u_n, I, and P, amounting in total to 4096 bits.

The wallet authority maintains a 4096-bit RSA key n, and signs M by means of the signature $S = M^d \mod n$, where d is chosen such that $M = S^3 \mod n$.

In integer arithmetic, this allows representing the equation that the user needs to prove they know a solution to merely as

$$S^3 = u_p u_q + I \times 2^{2048} + P \times 2^{3104} + Dn, \tag{1}$$

where S, u_p, u_q, P and D are all private integer variables, known only to the user, whereas I is communicated to the relying party and n is public knowledge. Together, as long as the user can also prove the bit lengths of all private variables, this equation encapsulates both that the wallet authority signed the document with the appropriate public identifier $u_n = u_p u_q$, and that the user knows the corresponding private identifier (u_p, u_q).

4.3 Bulletproof formulation

Unfortunately, we are unable to depict (1) directly as a Bulletproof equation. The reason for this is that this is an integer equation, but Bulletproof R1CS systems work only in modular integer arithmetic on fields the size of the underlying curve.

To resolve this, note that the entire equation is of $4096 \times 3 = 12288$ bits, as determined by its largest element, S^3. Thus, it is enough to ascertain that the difference between the left-hand side and the right-hand side is zero modulo some large value N, greater than 2^{12288}.

To prove this, we add to the shared public data (e.g., data published by the wallet authority) the identity of 174 distinct primes, each 71 bits in length, $P_1 \ldots, P_{174}$, such that $N = \prod_i P_i$ is sufficiently large. The computation of (1) can then be performed modulo each P_i separately, and if the equation holds under all moduli, then by the Chinese Remainder Theorem, it must also hold modulo N.

The reason for the modular representation, and the reason for the choice of 71-bit-long primes, is that it allows us to compute the modular equivalent of (1) without overflows.

To assert this, we must first make sure that all our variables are of the appropriate bit length. We do this by defining separate variables for each bit of S, u_p, u_q, P and D. This is a common technique used in ZKP in order to ascertain the bit-size of a value. For example, let us define L_S^j to be the j'th bit of S. Then the following equations

$$R_S^j = L_S^j - 1,$$

$$L_S^j \times R_S^j = 0,$$

ensure that all L_S^j are bits, while the equation

$$S = \sum_{j=0}^{4095} L_S^j 2^j$$

ensures that S is of the appropriate bit length.

In our case, we modify this paradigm by computing a new value, S_i, as

$$S_i = \sum_{j=0}^{4095} L_S^j (2^j \mod P_i), \tag{2}$$

to act as a convenient stand-in for $S \mod P_i$, and we create in the same way also modular variants for the equation's other variables.

This "modular representation" works because

$$S_i \equiv S \pmod{P_i}.$$

However, unlike $S \mod P_i$ the value of S_i is not constrained to be in $[0, P_i)$. Instead, we can only be sure that it does not exceed $4096P_i$.

This is enough to ensure that S_i^3 is still within the 252 bit range of the finite field in which we are computing, allowing us to effectively compute all of (1) modulo each P_i.

If we refer to the modular version of the left-hand side of (1) as LHS_i and the right-hand side as RHS_i, then the final R1CS equation for each P_i becomes

$$LHS_i + C_i = RHS_i + div_i P_i. \tag{3}$$

(The constants C_i, known to the relying party, are multiples of P_i designed to make sure that the div_i values proving the modular equation are nonnegative.)

In total, this approach requires 4096 variables in order to store the bits of S, twice 1022 bits in order to store the bits of u_p and u_q (noting that their most and least significant bits are both known to be 1s), a further 992 variables to store the bits of P, 8192 variables to store the bits of D, and 178 bits for each of the 174 div_i variables. Each such bit requires one multiplication gate.

Additionally, for each of the 174 P_i values, 2 multiplication gates are required to compute S_i^3 and 1 for the modular equivalent of $u_p u_q$.

In total, these are 46,818 multiplication gates.

Bulletproofs, by their design, require a number of multiplication gates that is a power of 2, for which reason this number must be rounded up to 65,536.

4.4 Performance

The Bulletproof design is relatively frugal in variables (65,536), relatively lightweight in total number of equations (94,854), and meets the main design goal of proving the veracity of the user's documents without divulging to the relying party any traceable user identifier.

The total size of the challenge response sent by the user to the relying party is also quite small: at merely 1,572 bytes, it is only half the size of what fits on a single QR code, and is a much smaller challenge response than was needed even for our initial system as described in Section 3.

Unfortunately, despite all this, the system's performance is, in practice, a little too slow for practical usage. In contrast to the system of Section 3, whose performance is for all practical purposes instantaneous, generating a Bulletproof based on a challenge by the relying party takes an average of 45 seconds, and verifying the response takes an additional 15 seconds.

In the next section, we show how we were able to optimise the system further, in order to bring these values down to interactive times.

5 THE FINAL DESIGN

In Section 4 we described the basic layout of a Bulletproof-based solution that meets the problem's privacy constraints, but its practical performance was too slow for the needed intensive use.

To resolve this, we have improved the design by adding both algorithmic and programmatic optimisations. We describe these in this section.

5.1 A new basic formula

In order to introduce the new optimisations, we must first undo one optimised design choice in the Section 4 design. Specifically, originally we established (3), which in turn encapsulates (1), by means of a single equation.

This was a choice that minimised the total number of multiplication gates used, but forced us to work in arithmetic modulo 71-bit primes, largely because the computation of S_i^3 had to fit into the 252-bit field which the Bulletproof was using, as determined by the size of the underlying elliptic curve.

For the present design, we require our computation to be modulo higher numbers, so instead of encapsulating (1) by a single equation, we break it down as

$$S' = S^2,$$
$$S'S = u_p u_q + I \times 2^{2048} + P \times 2^{3104} + Dn,$$

where the first of these, in turn, translates to the modular equation

$$S_i^2 = S_i' + div_i' P_i$$

thus requiring us to only use our 252-bit-sized field for squaring rather than for cubing numbers. By setting each P_i to be at most 111 bits in length, we can ensure that all S_i and other "modular" representations of our various integer variables all stay within 123 bits, making products of two such variables limited to 246 bits, giving us plenty of headroom in order to ensure that all of (1) can be computed in less than the available 252 bits.

5.2 Number-theoretic optimisations

The time costs of our Section 4 Bulletproof implementation can largely be traced back to two sources. First, setting up the Bulletproof equations requires much large-integer arithmetic, which was implemented using OpenSSL's BIGNUM functionality. The bulk of these computations need to be repeated 174 times, one for each prime modulus.

Second, the Bulletproof itself requires a one-time amount of computation related to the total number of nonzero coefficients in the Bulletproof equations, followed by a sequence of computations whose complexity is linear in the number of multiplication gates (which in our case is 65,536). These computations are in both Curve25519 Ristretto and in the modular integer field of the same size, and were implemented using libsodium's Ristretto implementation.

Of our Bulletproof variables (whose number is linear in the multiplication gates needed), only 15,324 describe the underlying integer equation. The bulk of our variables are those added for the modular computation. Specifically, 181 new variables are needed for each of the 174 moduli tested.

A common idea in Bulletproofs (and ZKP protocols beyond) is that instead of testing each of a large number of linear equations separately, one can choose a random combination of the linear equations and only test it. If all linear equations separately are satisfied, so will their linear combinations, but if not all linear equations are met, one can prove that one has only a negligible probability of choosing a combination that is satisfied.

In our case, we want to prove that the difference between the left-hand side and the right-hand side of (1) is zero. By construction, we know this difference to be nonnegative and at most 2^{12288}. Previously we proved it to be zero by computing its value modulo multiple numbers, P_i, noting that their least common multiple is greater than 2^{12288} and so if the difference is not zero, this would necessarily show up in at least one of the moduli's residues.

Consider, however, what would happen if we were to compute the difference, Δ, modulo some randomly-chosen 111-bit number, Q, not necessarily prime.

It turns out that merely choosing a random Q is not good enough. The reason for this is that Δ may be chosen adversarially to be, for example, a highly composite number [1, 38]. In such a case, the number of Q values dividing Δ will be large, so merely choosing randomly one Q value will not be enough to definitively conclude that Δ is, in fact, zero.

There is, however, a better way. Consider choosing Q not at random from all 111-bit numbers, but only uniformly from those numbers that are 2200-rough [36], defined as those numbers all of whose divisors are greater than 2200.

If Δ is nonzero and at most 12288 bits long, such a number can have at most 1106 prime divisors over 2200.

For a 111-bit number to be 2200-rough, it must be the product of at most 9 primes over 2200. Note that if some subset of the prime divisors of Δ is 111 bits in length, no product of a subset or superset

of this set can be in the same range (as these products must differ by at least 11 bits of length). Thus, the maximum number of divisors of Δ that are 2200-rough integers 111 bits in length can be bounded at C_9^{1106}, i.e. the number of ways to choose 9 primes out of 1106. (This can be attained in the worst case, which is when Δ is square free [43], meaning that it factorises into only unique primes.)

This number of possibilities is just under $2^{72.5}$.

The total number of 2200-rough 111-bit numbers is well-approximated as $2^{110}\prod_i \frac{p_i-1}{p_i}$, where p_i is the i'th smallest prime. This is approximately 0.073×2^{110}.

In total, the chance for a randomly-chosen Q to divide a non-zero Δ can be bounded from above by $2^{-33.7}$. By choosing 2 such Q candidates at random, we ensure that the probability for false acceptance is bounded from above by $2^{-67.4}$, which was deemed an acceptable rate.

In this optimisation, only 2 moduli need to be checked, rather than 174, reducing the bulk of the big-number computations required for the algorithm.

Moreover, the number of multiplication gates required in this design is only $4096 + 1022 \times 2 + 992 + 8192 + 2 \times 360 < 2^{14}$. This is a reduction to a quarter of the number of variables required for the initial Bulletproof, and also reduces the number of Bulletproof iterations required, bringing the entire algorithm solidly to interactive reaction times.

5.3 Programmatic optimisations

The question remains how best to allot a uniform 2200-rough number that is 111 bits long, recalling that the procedure to do so must be deterministic from one or more Fiat-Shamir challenges (so that it can be repeated by both prover and verifier, ascertaining that the number was not chosen adversarially). The distribution will not be completely uniform, but will have high enough entropy for our purposes.

To do this, we employ two techniques.

First, in order to allot a number that is not divisible by 2, 3, 5, 7 and 11, we prepare in advance a table, T, of all 480 residues modulo $2310 = 2 \times 3 \times 5 \times 7 \times 11$ that do not divide by any of these primes.

Our Fiat-Shamir challenges are outputs of SHA3-512 [21]. This provides us, for a single challenge, with 512 pseudorandom bits. Of these, we can use 17 bits in order to compute the 480 modulus of a uniformly-chosen 17-bit number, and this already provides us with a close-enough approximation of a uniform value in the range $[0, 479]$, which we can use as an index to the look-up table, T, of residues modulo 2310, thus uniformly choosing a residue class for our chosen number, Q, modulo 2, 3, 5, 7 and 11. Let m_{2310} be the value retrieved from the table of residues.

Next, we choose uniformly a value d_{2310} in the range

$$[\lceil 2^{110}/2310 \rceil, \lfloor 2^{111}/2310 \rfloor).$$

This provides us with a Q candidate $2310 \times d_{2310} + m_{2310}$ which we know does not divide by any prime smaller than 13.

The range for d_{2310} is just under 99 bits. We can allot a d_{2310} value uniformly by choosing 99 bits from the SHA3 output, then using rejection sampling to reject the choice if the value of d_{2310} it produces (after adding $\lceil 2^{110}/2310 \rceil$) is not smaller than $\lfloor 2^{111}/2310 \rfloor$. Such rejection happens with probability around 1/8.

Next, we need to make sure that Q does not divide by any of the 322 remaining primes below 2200. This we do by checking, i.e. by more rejection sampling: if our Q candidate divides by any of the remaining "small" primes, we eliminate it, and use the next 99 bits of the SHA3 output in order to allot another d_{2310},

The probability that a Q candidate does not divide any of the remaining 322 primes is approximately 1/3, and even when accounting also for the probability of rejection due to a too-large d_{2310}, the probability of success is still above 30%. In expectation, roughly 3 attempts are needed in order to find a successful candidate. Having started with 512 random bits, we can afford a full 5 attempts before another Fiat-Shamir challenge needs to be computed, so in high probability the procedure can be completed using only the first challenge.

This leaves us with the question of how to efficiently verify that a candidate value Q does not divide by any of the remaining primes. To do this, we have partitioned the primes to 49 buckets, such that the primes in each bucket have a product smaller than 2^{64}. This can be done simply by greedy assignment.

The trick here is that much of the reason for the slow computation time is the need to handle large numbers (which we do using OpenSSL's BIGNUM functionality). Instead, for each Q candidate we only compute in this way the 49 residues $\{Q \bmod \Pi_i\}_{i=1}^{49}$, where Π_i is the product of all primes in the i'th bucket. The result of each such modulo operation can then be stored in a 64-bit unsigned "long" integer, allowing all remaining checks to be done at much higher speeds.

6 NEXT STEPS

Research on the described systems is continuing. The designs described in this paper are for demonstration prototypes, not for production-ready systems.

Additional features that will need to be supported for real-world usability include

(1) Support for multiple document issuing authorities, as well as a separation between the document issuers and the authority managing the wallet system itself,
(2) Support for automatic life-cycle management by recognising document expiration and revocation of documents, and
(3) Potentially, support for resilience against quantum attacks.

New designs that integrate such abilities are currently under investigation.

Even so, the present system serves as a powerful proof point that ZKP systems have matured to the point that they can support complex, real-world usage, powering systems that afford better privacy management than traditional solutions can.

ACKNOWLEDGMENTS

This work was funded by INCERT GIE.

REFERENCES

[1] Leonidas Alaoglu and Paul Erdős. 1944. On highly composite and similar numbers. *Trans. Amer. Math. Soc.* 56 (1944), 448–469.
[2] Álvaro Alonso, Alejandro Pozo, Aldo Gordillo, Sonsoles López-Pernas, Andrés Munoz-Arcentales, Lourdes Marco, and Enrique Barra. 2020. Enhancing university services by extending the eIDAS European specification with academic attributes. *Sustainability* 12, 3 (2020), 770.

[3] Mario Alessandro Barbara, Lorenzo Grassi, Dmitry Khovratovich, Reinhard Lüftenegger, Christian Rechberger, Markus Schofnegger, and Roman Walch. 2021. Reinforced Concrete: Fast hash function for zero knowledge proofs and verifiable computation. *IACR Cryptol. ePrint Arch.* 2021 (2021), 1038.

[4] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. 2019. Scalable zero knowledge with no trusted setup. In *Advances in Cryptology – CRYPTO 2019: 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2019, Proceedings, Part III 39*. Springer, 701–732.

[5] Diana Gratiela Berbecaru and Antonio Lioy. 2018. On integration of academic attributes in the eIDAS infrastructure to support cross-border services. In *2018 22nd International Conference on System Theory, Control and Computing (ICSTCC)*. IEEE, 691–696.

[6] Diana Gratiela Berbecaru, Antonio Lioy, and Cesare Cameroni. 2019. Electronic identification for universities: Building cross-border services based on the eIDAS infrastructure. *Information* 10, 6 (2019), 210.

[7] Diana Gratiela Berbecaru, Antonio Lioy, and Cesare Cameroni. 2021. On enabling additional natural person and domain-specific attributes in the eIDAS network. *IEEE Access* 9 (2021), 134096–134121.

[8] Waldemar Berchtold, Huajian Liu, Martin Steinebach, Dominik Klein, Tobias Senger, and Nicolas Thenee. 2020. JAB code – A versatile polychrome 2D barcode. *Electronic Imaging* 32 (01 2020), 1–7.

[9] Daniel J Bernstein. 2006. Curve25519: New Diffie-Hellman speed records. In *International Workshop on Public Key Cryptography*. Springer, 207–228.

[10] Daniel J Bernstein, Peter Birkner, Marc Joye, Tanja Lange, and Christiane Peters. 2008. Twisted Edwards curves. In *International Conference on Cryptology in Africa*. Springer, 389–405.

[11] Daniel J Bernstein and Frank Denis. 2019. Libsodium – A modern, portable, easy to use crypto library. https://github.com/jedisct1/libsodium. Retrieved 3 February 2023.

[12] Daniel J Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. 2011. High-speed high-security signatures. In *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 124–142.

[13] Daniel J Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. 2012. High-speed high-security signatures. *Journal of Cryptographic Engineering* 2, 2 (2012), 77–89.

[14] Soeren Bittins, Gerhard Kober, Andrea Margheri, Massimiliano Masi, Abdallah Miladi, and Vladimiro Sassone. 2021. Healthcare data management by using blockchain technology. In *Applications of Blockchain in Healthcare*, Suyel Namasudra and Ganesh Chandra Deka (Eds.). Springer Singapore, Singapore, 1–27.

[15] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. 2018. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 315–334.

[16] David Chaum, Jan-Hendrik Evertse, and Jeroen van de Graaf. 1988. An improved protocol for demonstrating possession of discrete logarithms and some generalizations. In *Advances in Cryptology – EUROCRYPT'87: Workshop on the Theory and Application of Cryptographic Techniques Amsterdam, The Netherlands, April 13–15, 1987 Proceedings 6*. Springer, 127–141.

[17] CMKC Cuijpers and Jessica Schroers. 2014. eIDAS as guideline for the development of a pan European eID framework in FutureID. In *The Open Identity Summit 2014, Gesellschaft für Informatik*, Vol. 237. Bonn: Bonner Köllen Verlag, 23–38.

[18] Ivan Damgård, Torben Pedersen, and Birgit Pfitzmann. 1994. On the existence of statistically hiding bit commitment schemes and fail-stop signatures. In *Advances in Cryptology – CRYPTO'93*. Springer, 250–265.

[19] Dirk Draheim. 2020. Blockchains from an e-governance perspective: Potential and challenges. In *Electronic Governance and Open Society: Challenges in Eurasia, 7th International Conference, EGOSE 2020, St. Petersburg, Russia, November 18–19, 2020, Proceedings*, Andrei Chugunov, Igor Khodachek, Yuri Misnikov, and Dmitrii Trutnev (Eds.). Springer International Publishing, Cham, 75–89.

[20] Morris J Dworkin. 2001. *Recommendation for block cipher modes of operation: Methods and techniques*. Technical Report SP800-38A. National Institute of Standards and Technology, U.S. Department of Commerce, Gaithersburg, MD.

[21] Morris J Dworkin. 2015. *SHA-3 standard: Permutation-based hash and extendable-output functions*. Technical Report Federal Information Processing Standards Publication (FIPS PUBS) 202. National Institute of Standards and Technology, U.S. Department of Commerce, Washington, D.C.

[22] Nils Engelbertz, Nurullah Erinola, David Herring, Juraj Somorovsky, Vladislav Mladenov, and Jörg Schwenk. 2018. Security analysis of eIDAS – The cross-country authentication scheme in Europe. In *12th USENIX Workshop on Offensive Technologies (WOOT 18)*.

[23] Amos Fiat and Adi Shamir. 1986. How to prove yourself: Practical solutions to identification and signature problems. In *Crypto*, Vol. 86. Springer, 186–194.

[24] Uriel Fiege, Amos Fiat, and Adi Shamir. 1987. Zero knowledge proofs of identity. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*. ACM, 210–217.

[25] Oded Goldreich and Hugo Krawczyk. 1996. On the composition of zero-knowledge proof systems. *SIAM J. Comput.* 25, 1 (1996), 169–192.

[26] Oded Goldreich, Silvio Micali, and Avi Wigderson. 1991. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM (JACM)* 38, 3 (1991), 690–728.

[27] Lorenzo Grassi, Dmitry Khovratovich, Christian Rechberger, Arnab Roy, and Markus Schofnegger. 2021. Poseidon: A new hash function for zero-knowledge proof systems. In *USENIX Security Symposium*.

[28] Shay Gueron, Simon Johnson, and Jesse Walker. 2011. SHA-512/256. In *2011 Eighth International Conference on Information Technology: New Generations*. IEEE, 354–358.

[29] Mike Hamburg. 2015. Decaf: Eliminating cofactors through point compression. In *Advances in Cryptology – CRYPTO 2015: 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I 35*. Springer, 705–723.

[30] Dmitry Jemerov and Svetlana Isakova. 2017. *Kotlin in Action*. Simon and Schuster.

[31] Mirosław Kutyłowski and Przemysław Błaśkiewicz. 2023. Advanced electronic signatures and eIDAS – Analysis of the concept. *Computer Standards & Interfaces* 83 (2023), 103644.

[32] Silvia Lips, Nitesh Bharosa, and Dirk Draheim. 2020. eIDAS implementation challenges: The case of Estonia and the Netherlands. In *Electronic Governance and Open Society: Challenges in Eurasia: 7th International Conference, EGOSE 2020, St. Petersburg, Russia, November 18–19, 2020, Proceedings 7*. Springer, 75–89.

[33] Yue Liu, Ju Yang, and Mingjun Liu. 2008. Recognition of QR code with mobile phones. In *2008 Chinese Control and Decision Conference*. IEEE, 203–206.

[34] Nikolay A Moldovyan. 2011. Blind signature protocols from digital signature standards. *Int. J. Netw. Secur.* 13, 1 (2011), 22–30.

[35] Marcin Moskala and Igor Wojda. 2017. *Android Development with Kotlin*. Packt Publishing Ltd.

[36] David Naccache and Igor E Shparlinski. 2009. Divisibility, smoothness and cryptographic applications. In *Algebraic Aspects of Digital Communications*, Tanush Shaska and Engjell Hasimaj (Eds.). Vol. 24. Ios Press.

[37] Charles Rackoff and Daniel R Simon. 2001. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *Advances in Cryptology – CRYPTO'91, Proceedings*. Springer, 433–444.

[38] Srinivasa Ramanujan. 1997. Highly composite numbers. *Ramanujan Journal* 1 (1997), 119–119.

[39] Ronald L Rivest, Adi Shamir, and Leonard Adleman. 1978. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM* 21, 2 (1978), 120–126.

[40] Stephen Samuel and Stefan Bocutiu. 2017. *Programming Kotlin*. Packt Publishing Ltd.

[41] C Sanchez-Avila and R Sanchez-Reillol. 2001. The Rijndael block cipher (AES proposal): A comparison with DES. In *Proceedings IEEE 35th Annual 2001 International Carnahan Conference on Security Technology (Cat. No. 01CH37186)*. IEEE, 229–234.

[42] Peter Schwabe and Amber Sprenkels. 2019. The complete cost of cofactor $h = 1$. In *Progress in Cryptology – INDOCRYPT 2019: 20th International Conference on Cryptology in India, Hyderabad, India, December 15–18, 2019, Proceedings (Lecture Notes in Computer Science, Vol. 11898)*, Feng Hao, Sushmita Ruj, and Sourav Sen Gupta (Eds.). Springer, 375–397.

[43] Harold N Shapiro. 2008. *Introduction to the Theory of Numbers*. Courier Corporation.

[44] Tan Jin Soon. 2008. QR code. *Synthesis Journal* 2008 (2008), 59–78.

[45] Larry J Stockmeyer. 1976. The polynomial-time hierarchy. *Theoretical Computer Science* 3, 1 (1976), 1–22.

[46] Sumit Tiwari. 2016. An introduction to QR code technology. In *2016 International Conference on Information Technology (ICIT)*. IEEE, 39–44.

[47] Muhamed Turkanović and Blaž Podgorelec. 2020. Signing blockchain transactions using qualified certificates. *IEEE Internet Computing* 24, 6 (2020), 37–43.

[48] John Viega, Matt Messier, and Pravir Chandra. 2002. *Network Security With OpenSSL: Cryptography for Secure Communications*. O'Reilly Media, Inc.

[49] Eric A Young, Tim J Hudson, and Ralf S Engelschall. 2022. OpenSSL. https://www.openssl.org/.