BulletID: ePassport Possession Attestation Using Bulletproofs

Yannick Abouem
Queen's University
School of Computing
Kingston, ON, Canada
24hd7@queensu.ca

Abstract

In the digital age, it has become more common the need to provide IDs to a website in order to access some service. However, doing so is risky as you can not fully trust the website operator with such information. To solve this problem, we utilized zero-knowledge proofs to construct proofs using the information in the machine-readable zone of a passport. Zero-knowledge proofs are a cryptographically secure way of constructing proofs of knowledge of some secret piece of information. Using the Bulletproofs protocol, we constructed a program that construct a proof from the information in a passport and verifies said proof.

1 Introduction

With the advent of the digital age, it has become more common the need for a user to prove their legal identity over the internet. This tasks often involves filling a form containing your own personal information, such as name, date of birth, sex etc. and, in occasions, providing a picture or a scan of one of your IDs. Modern cryptographic techniques will protect your information in transit and when stored, but it would be of little help if the other party were to misuse, either by malice or by incompetence, your own information.

To illustrate this problem, assume a private company P that provides electricity to a city. This company has a simple sign-up page to become a costumer, but in order to complete the form a prospective costumer has to provide a picture of one of their IDs to prove that they are actually a person. Using the current techniques the user will have to send said picture over the internet, which will then be reviewed by the company P. However, the user has no longer control on what happens to that picture. Let's consider the case where P, due to cost-cutting measures, has poor cybersecurity practices and a malicious entity is able to take control of their systems. If, let's say, the ID pictures sent to P were not to be properly stored with encryption, the attacker will be able to steal the ID pictures and use them for their nefarious purposes. This would cause a massive data breach that would affect all of P's costumers.

There exists many ways to improve upon this scenario. For example, P might only store the ID pictures for no longer than they need them for. We decided to take a different approach from most of the solutions and explored the use of zero-knowledge proofs to create proofs of ID possession, specifically passports, to solve this problem. As we will see, we created a system to convert the machine-readable zone information of a passport into a proof of knowledge of this information using the inner-product proofs offered by Bulletproofs.

2 Background

In this section we will provide background on electronic IDs (eIDs), zero-knowledge proofs (ZKP) and Bulletproofs.

2.1 Electronic ID

To enhance the identity verification process, many countries issue identity documents (IDs) with an embedded RFID chip [10]. This chip contains the same information as displayed on the physical card or, it may, in addition, store biometric information of the holder such as fingerprint and facial recognition data [10].

EIDs are commonly issued in the form of national IDs but, the most common form of eIDs are ePassports. According to the International Civil Aviation Organization (ICAO)[NA], "[t]here are more than 140 States and non-state entities (e.g. United Nations, European Union) currently issuing ePassports, and over 1 billion ePassports in circulation". EPassports are a form of electronic Machine Readable Travel Documents (eMRTD) and are standardized by ICAO under the Doc 9303. [7, 9]. Each passport possesses a machine-readable zone (MRZ) located in the passport data page, which is mandatory for all ICAO members to include [7]. This MRZ contains all information about the passport and the individual who possesses it. This data is formatted, according to the ICAO [2021], in the following manner:

On the first line:

(1) Document code. One character P followed by a character denoting the passport type or a "filler" character <
(2) Issuing state or organization. The three-letter ISO code of the state or organization issuing the passport
(3) Name. The name of the individual who owns the passport of length at most 39 characters. Last name and given names might be separated by "filler" characters. Extra unused characters will be filled with <

On the second line:

(1) Passport number. Nine characters long passport number
(2) Check digit. Check digit of the passport number
(3) Nationality. Three-letter ISO code of the holder's nationality
(4) Date of birth. Date of birth of the holder of length 6
(5) Check digit. Check digit for the date of birth
(6) Sex. Single character denoting the sex of the holder. Either one of M, F or <
(7) Date of expiry. Six digit expiriy date of the passport
(8) Check digit. Check digit for the date of expiry
(9) Personal number or other data element. This section has 14 characters in length and can be used by a country as they see fit. If unused replaced with filler characters.
(10) Check digit. Check digit for the previous section. If previous section is empty could be 0 or filler character
(11) Composite check digit. Check digit for all check digits in the second line

We will be using the data contained in the MRZ in this project as the information we desire to create a zero-knowledge proof of.

2.2 A Note About Notation

In this report we will use the same notation as used by Bunz et al. [2017] in the original Bulletproofs paper. In particular, we will denote vectors in bold, i.e. $\mathbf{a} \in \mathbb{F}^n$ is a vector with elements in $\mathbb{F}$, and the inner product notation $c = \langle \mathbf{a}, \mathbf{b} \rangle$. We will also denote cyclic groups of prime order p with $\mathbb{G}$ and the ring of integer modulo p as $\mathbb{Z}_p$.

2.3 Zero-Knowledge Proofs

Zero-knowledge Proofs (ZKP) are a cryptographic technique to prove the knowledge of a secret piece of information without revealing this information to the verifier of the proof. This is particularly useful if we do not fully trust the verifier with information such as personal identifiable information (PII) or IDs. A ZKP protocol has, usually, two entities, a Prover, who constructs a proof of knowledge over some secret information, and a Verifier that must be convinced by the Prover that their proof is correct.

There are three properties a ZKP protocol possesses:

(1) **Correctness**, if an honest verifier is able to convince a Prover that they know the secret information [5]
(2) **Zero-knowledge**, no secrete information is revealed to the Verifier other than the necessary information [5]
(3) **Soundness**, if the verification of the proof correlates to proving knowledge of the secret information [5]

Protocols rely on one-way functions or circuits to create the proof to ensure that it is not computationally feasible for a dishonest Prover or Verifier to violate any of the properties.

There exists many ZKP protocols in the literature. Among these the most popular and well-developed are zk-SNARK, zk-STARKS and Bulletproofs. In the following paragraphs we will briefly describe the first two protocols and in the next subsection we will explore more in depth Bulletproofs and how does the protocol work.

2.3.1 zk-SNARK. Short for zero-knowledge succinct non-interactive argument of knowledge, zk-SNARK is an argument of a NP statement (such as a computation) that is verifiable with a small computational complexity [1]. This is achieved through the use of collision-resistant hash functions and probabilistically-checked proofs[1].

2.4 Bulletproofs

Bulletproofs is a non-interactive zero-knowledge protocol that possesses short proofs and does not require a trusted setup, which distinguish it from both zk-SNARKS, which require a trusted setup, and zk-STARKS, which proofs are larger but require no trusted setup [2]. Before explaining how Bulletproofs work it is necessary to talk about Pedersen commitments.

2.4.1 Pedersen Commits. A cryptographic commitment scheme is a pair of algorithms (Setup, Com) such that the algorithm Setup generates the public parameters for the scheme and the algorithm Com defines a function that given a message and a random value produces a commitment to that message [2]. A commitment scheme must abide by two properties:

- Hiding, a commitment does not reveal the message [5]
- Binding, there is no message m' such that $\mathrm{Com}(m) = \mathrm{Com}(m')$ [2, 5]

An example of a commitment scheme is cryptographic hash functions since they possess the binding property and it is also possible to obtain the hiding property through the use of a random value in the input. However, Bulletproofs require a third property for its commitments, Homomorphism.

A homomorphic commitment is a commitment such that the equation

$$\mathrm{Com}(m_1, r_1) + \mathrm{Com}(m_2, r_2) = \mathrm{Com}(m_1 + m_2, r_1 + r_2) \tag{1}$$

is true [2]. One such commitment scheme is the Pedersen commitment scheme.

A vector Pedersen commitment C is a point on an elliptic curve such that:

$$C = rH + \sum_{i=1}^{n} v_i G_i \tag{2}$$

where $\mathbf{v}$ is a secret vector for which we are building the commitment of, r is a randomly generated scalar, $\mathbf{G}$ is a generator vector of the elliptic curve previously agreed upon by both parties, and H is a point on the curve for which the discrete logarithm q is not known by anyone and $H = q\mathbf{G}$ [5].

2.4.2 The Inner-Product Argument. The core component of Bulletproofs is the inner-product argument, which is an argument of knowledge that proves that the Prover knows the secret vectors in the Pedersen commitment that satisfy a specific inner product relation [2]. We will show, in brief, how the argument works, as shown by Groth [2009] and Gibson [2022].

Assuming that the Prover has two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_p^n$ of size n, and a scalar z such that $z = \langle \mathbf{x}, \mathbf{y} \rangle$ [5]. We construct three commitments:

$$C_z = tH + zG \tag{3}$$

$$C_x = rH + \sum_{i=1}^{n} x_i G_i \tag{4}$$

$$C_y = sH + \sum_{i=1}^{n} y_i G_i \tag{5}$$

The algorithm will work in three steps:

(1) **Commitment** The Prover creates four additional commitments:

$$A_d = r_d H + \sum_{i=1}^{n} d_{xi} G_i \tag{6}$$

$$B_d = s_d H + \sum_{i=1}^{n} d_{yi} G_i \tag{7}$$

$$C_1 = t_1 H + (\langle \mathbf{x}, \mathbf{d}_y \rangle + \langle \mathbf{y}, \mathbf{d}_x \rangle) \tag{8}$$

$$C_0 = t_0 H + (\langle \mathbf{d}_x, \mathbf{d}_y \rangle) G \tag{9}$$

A_d and B_d are commitments for two nonce vectors $\mathbf{d}_x$ and $\mathbf{d}_y$ one for each of $\mathbf{x}$ and $\mathbf{y}$ [5]. C_1 and C_0 are commitments to the inner product of the *blinded* form of our vectors, this will return useful later in the response phase [5].

(2) **Challenge** The challenge phase simpy consists of the Verifier sending a scalar value e as challenge [5].

(3) **Response** The Prover computes the following and send them to the Verifier:

$$\mathbf{f}_x = e\mathbf{x} + \mathbf{d}_x \tag{10}$$

$$\mathbf{f}_y = e\mathbf{y} + \mathbf{d}_y \tag{11}$$

$$r_x = er + r_d \tag{12}$$

$$s_y = es + s_d \tag{13}$$

$$t_z = e^2 t + et_1 + t_0 \tag{14}$$

The verifier will then check that:

$$eC_x + A_d = r_x H + \sum_{i=1}^{n} (\mathbf{f}_x)_i G_i \tag{15}$$

$$eC_y + B_d = s_y H + \sum_{i=1}^{n} (\mathbf{f}_y)_i G_i \tag{16}$$

If the two sides of the equations are equal then we proven honest behaviour from the prover's side [5]. Finally, we need to perform a third check to verify the correctness of the inner product:

$$\langle \mathbf{f}_x, \mathbf{f}_y \rangle = e^2 z + e(\langle \mathbf{x}, \mathbf{d}_y \rangle + \langle \mathbf{y}, \mathbf{d}_x \rangle) + (\langle \mathbf{d}_x, \mathbf{d}_y \rangle) \tag{17}$$

2.4.3 Non-Interactability. As shown above, Bulletproofs require interaction between the Prover and the Verifier. We can render this protocol non-interactive through the use of a Fiat-Shamir Heuristic and generating the challenge using the public data of the proof.

3 Methodology

In this section we will discuss our implementation as well as the library used in this project.

3.1 Design

The program functions in three stages:

(1) **Passport Encoding**
(2) **Proof Construction**
(3) **Proof Verification**

In the first stage the program constructs a `Passport` object from string values. The `Passport` object has one variable for each field in the MRZ of a phisical passport. This data will later be used to construct the proof.

The second stage consists in first computing the hash of the `Passport` object. This is achieved by converting all variables in the objects into arrays of bytes and concatenating together. The final array will be passed to a hash function. The hash function chosen for this project is `SHA3-256`, which was chosen due to the small size of the resulting hash and its resistance against attacks. Once the hash is computed, the proof is then constructed. The program begins by randomly selecting the generator vector $\mathbf{G}$ of size $n = 32$. Then, we create the Pedersen generator F and the blinding value B. In the next step, we have to determine a public vector $\mathbf{b}$ that will be used in computing the inner product c with our secret vector $\mathbf{a}$. We determine such vector randomly using the cryptographically secure random number generator (CSRNG) offered by the `rand` crate. For our secret vector $\mathbf{a}$, we use the hash value of the passport to fill in all entries. Next, the program initializes the `merlin` transcript that is used to render the protocol non-interactive, and the blinding value of the commitment r. Finally, we compute the inner product c using our secret vector $\mathbf{a}$ and the public vector $\mathbf{b}$ and the commitment C using the formula

$$C = \sum_{i=0}^{n-1} a_i G_i + rB + cF \tag{18}$$

The last step in the proof creation is to invoke the `create` function from the `bulletproofs` crate to create the actual Bulletproofs proof and create the object `ProofEnv`, which contains all public values that will be used in the verification step.

The verification step simply consists in creating the transcript for the Verifier calling the `verify` method of the `LinearProof` object and passing in the public values computed in the proof creation. If the proof succeeds then the program will display true, otherwise it will throw an `VerificationException`.

3.2 Tools Used

To implement our solution we used the Rust programming language as it supports the most modern and well-maintained Bulletproofs implementation. This library relies on Ristretto to generate the elliptic curve needed and Merlin, a transcript construction for ZKP that automates the Fiat-Shamir heuristic [3, 4].

During the initial stages of this project we consider other Bulletproofs implementations. One such library is the Bulletproofs library available for Haskell, which was originally planned for this project. However, we had to reconsider our choice due to the age of the library and the last published update of it. This library has not been maintained for the past 6 years making it outdated and thus unfit to be used for this project.

4 Future Work

In this section we will touch upon future work that could expand on this project.

To start, we would like to expand on the current implementation and add the ability of, not only using passports, but also any form of eID that abides by ICAO Doc 9303 and has a MRZ. This would improve on the usability of this techniques by allowing individuals that do not possess a passport to use it.

Another expansion that in our opinion is vital for this project, is the addition of a TCP server and client to the program to showcase the functionality of the system better.

Finally, another addition could be the ability of a user of entering passport data by reading the RFID chip on the ePassport. Originally this was one of the goals of the project, but it was later discarded due to difficulties in reading the chip as ePassports use sophisticated protections to avoid unauthorized reading of the data.

5 Conclusion

As we have discussed, this project is aimed at improving the security of sharing IDs over the internet, specifically to entities we might not fully trust with our PII by using ZKP protocols. We discussed background knowledge of MRTD such as passports and the information contained in their MRZ. We presented a background of ZKP in brief and explored Bulletproofs, as well as its component, more in depth.

Finally, we outlined the work we carried out by presenting the design of our solution and how it functions. Later, we discussed the tools used to implement our program and future work necessary to improve upon our proposed solution.

References

[1] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. 2012. Recursive Composition and Bootstrapping for SNARKs and Proof-Carrying Data. Cryptology ePrint Archive, Paper 2012/095. https://eprint.iacr.org/2012/095

[2] Benedikt Bunz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. 2017. Bulletproofs: Short Proofs for Confidential Transactions and More. (2017). Retrieved April 6, 2025 from https://web.stanford.edu/~buenz/pubs/bulletproofs.pdf

[3] Henry de Valence. 2018. Merlin. Retrieved April 6, 2025 from https://crates.io/crates/merlin

[4] Henry de Valence, Cathie Yun, , and Oleg Andreev. 2018. Bulletproofs. Retrieved April 6, 2025 from https://crates.io/crates/bulletproofs

[5] Adam Gibson. 2022. From Zero (Knowledge) to Bulletproofs. (27 Jun 2022). Retrieved April 5, 2025 from https://github.com/AdamISZ/from0k2bp

[6] Jens Groth. 2009. Linear Algebra with Sub-linear Zero-Knowledge Arguments. (2009). Retrieved April 6, 2025 from http://www0.cs.ucl.ac.uk/staff/J.Groth/MatrixZK.pdf

[7] International Civil Aviation Organization. 2021. *Doc 9303, Machine Readable Travel Document* (8 ed.).

[8] International Civil Aviation Organization. NA. *ePassport Basics*. Retrieved April 5, 2025 from https://www.icao.int/Security/FAL/PKD/Pages/ePassport-Basics.aspx

[9] Wikipedia. 2025. Biometric passport. Retrieved April 5, 2025 from https://en.wikipedia.org/wiki/Biometric_passport

[10] Wikipedia. 2025. Electronic identification. Retrieved April 5, 2025 from https://en.wikipedia.org/wiki/Electronic_identification