ZKPMP: Zero-Knowledge Proofs for Anomaly and Similarity Detection in Time Series via Matrix Profile

Anonymous Author(s)

Abstract

Analyzing time-series databases in a privacy-preserving manner has gained significant attention, especially when the data contains sensitive personal information such as medical records or spatio-temporal data such as trajectories. Motivated by scenarios in which marathon organizers must verify the absence of cardiac irregularities in athletes' electrocardiogram (ECG) data, we study how to prove the presence or absence of anomalies or similarities without revealing the underlying time series. We leverage Matrix Profile (MP), a state-of-the-art data-mining structure, to detect anomalies and similarities in time series *at subsequence level* (small granularity), in contrast to many works that compute distance on the *complete time series* (large granularity). We adopt a strong adversarial model in which aggregated statistics (subsequence distances or MP values) and locations of potential similarities and anomalies are treated as sensitive, consistent with recent findings showing that these quantities can leak information comparable to the original data. To address these challenges, we propose a non-interactive zero-knowledge cryptographic scheme, allowing users to prove claims about anomalies or similarities in their private time series while revealing no information beyond the validity of the claim. We further implement and evaluate the performance of the proposed scheme.

CCS Concepts

• **Security and privacy**;

Keywords

Do, Not, Use, This, Code, Put, the, Correct, Terms, for, Your, Paper

ACM Reference Format:

Anonymous Author(s). 2018. ZKPMP: Zero-Knowledge Proofs for Anomaly and Similarity Detection in Time Series via Matrix Profile. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (Conference acronym 'XX).* ACM, New York, NY, USA, 23 pages. https://doi.org/XXXXXXX.XXXXXXX

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
Conference acronym 'XX, Woodstock, NY
© 2018 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Time-series applications today increasingly rely on cloud infrastructures for data storage and computation. The rapid growth of sensor-rich environments, such as wearable devices, medical monitors, and mobile endpoints, has led to massive volumes of personal and behavioral data being continuously collected. Outsourcing the processing of such data to untrusted cloud services raises serious privacy concerns. This makes secure computation techniques essential to ensure that sensitive information remains hidden from the service provider while computations are performed on the data.

A general task in this setting is performing similarity or anomaly queries over encrypted time series. For example, a marathon organizer may want to verify that participants do not exhibit abnormal patterns in their electrocardiogram (ECG) signals. To do so, a widely used approach in data analytics is to use a technique called Matrix Profile (MP) [21] (explained in detail in Sec. 2.1). MP is a transformation of a time series which produces a structure that captures all pairwise similarity relationships between subsequences and enhances the detection of anomalies or absence of similarities, in an unsupervised way. To motivate the uses of MP, we illustrate (see Figure 2) a classical application: detecting ECG anomalies using MP. It is important to stress that MP is used in an unsupervised manner, *i.e.,* by comparing a time series with itself in order to find anomalies. No other data is needed, contrary to an approach where *e.g.,* the time series would be compared to many "template" time series. For instance, the Statewide California Earthquake Center [17] explains that they use the self-join MP to identify previously uncatalogued seismic events by matching all sub-windows (of a set length) in a continuous stream against the remainder of the stream, which cannot be accomplished using template-based matching methods.

In parallel, other works have recently explored privacy-preserving data sharing with MP. While MP transformation may obfuscate raw time-series, recent evidence demonstrates that MP is not a privacy-preserving structure [22]: adversaries can exploit subsequence distance patterns to reconstruct substantial portions of the original time series or infer sensitive attributes. This reveals a fundamental gap that while MP is a highly expressive and widely adopted tool for unsupervised time-series mining, it cannot be directly shared or computed in the clear without privacy protection.

Objectives. We aim to bridge this gap by proposing a verifiable and privacy-preserving computation of the pairwise distance inspired by MP, in order to achieve our ultimate goal: to design **a cryptographic protocol that enables users to prove, with zero knowledge (ZKP), and in**

a non-interactive manner, the presence or absence of anomalies or similarities in their personal time series without revealing any additional information about the data or the distance values, *i.e.,* without directly sharing neither the time series itself nor its MP. Concretely, we consider four classic privacy-critical scenarios, that are:

- **No anomaly ($\perp_A$)**: proving and verifying that an athlete's ECG contains no cardiac anomalies before a marathon;
- **Anomaly (A)**: proving and verifying the existence of an unusual event in medical data in order to obtain a medical certificate;
- **No similarity ($\perp_S$)**: proving and verifying the absence of similarity between one's trajectory and that of a contaminated patient;
- **Similarity (S)**: proving and verifying that two biomedical signals share a clinically relevant motif to confirm a medical diagnosis.

We consider a setting with three parties, illustrated in Figure 1. The device provides the commitments on the time series data and signs. The prover then generates a proof of one of the four possible claims described before using his data; the proof reveals no information beyond the validity of the claim itself. Finally, the verifier checks the proof against the signed data and accepts the claim if the verification succeeds. Note that the proving and verification phases can be performed offline, thus real-time efficiency is not expected, as long as the approach scales to realistic time series sizes (*i.e.,* 1000 points).

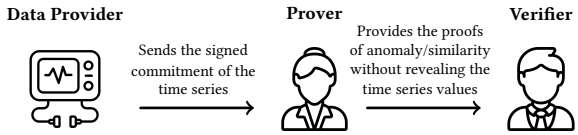

Figure 1: Scenario description

Contributions. To address the challenges, we leverage Schnorr-based non-interactive zero-knowledge proofs (NIP), to prove a distance-based algorithm, corresponding to one of the scenarios. This paper thus makes the following contributions:

- **Zero-knowledge proof for subsequence level anomaly/similarity analysis in time series.** We introduce a set of unit proofs that serve as building blocks for four protocols to prove presence or absence of anomaly or similarity over time series at subsequence level, by leveraging MP. To our knowledge, this is the first work to apply zero-knowledge proofs with subsequence distance metric in time series.
- **Security and verifiability guarantees.** We formalize the threat model, and provide proofs of correctness, zero knowledge, and extractability for our protocols.
- **Implementation and evaluation.** We present a full Rust implementation of our protocols and evaluate the scalability on realistic time-series workloads.

Paper organization. The paper is organized as follows: Section 2 introduces the concepts of MP and ZKP. Section 3 presents our problem statement. Section 4 presents the formal model used to define our cryptographic scheme. Section 5 present the scheme, and Section 6 its Non-Interactive Proof (NIP) instantiation. Section 7 presents our performance results on our RUST implementation, Section 8 discusses related works, and Section 9 concludes.

2 Preliminaries and Notations

Notation. We denote by $[\![n]\!]$ the set $\{1, \cdots, n\}$ and by $[\![i; j]\!]$ the set $\{i, \cdots, j\}$. We write $x \xleftarrow{\$} S$ to mean that x is picked uniformly at random from a set S. We use the acronym s.t. for such that, *PT* for polynomial time and *PPT* for probabilistic polynomial time.

2.1 Background on Matrix Profile

A *Time Series* can be represented by a vector $x = (x_i)_{i \in [\![n]\!]}$ together with a timestamp vector $tmp = (tmp_i)_{i \in [\![n]\!]}$. In what follows, we assume that temporal dependency is implicitly given by the ordering of x, and we omit explicit use of tmp for clarity.

Matrix Profile (*MP*) [21] is a data mining structure composed of two vectors *MPD* and *MPI*, parameterized by two time series (x, x'), a distance measure Dist and a subsequence length m noted:

$$MP_{\mathsf{Dist},m}(x, \hat{x}) = (MPD_{\mathsf{Dist},m}(x, \hat{x}), MPI_{\mathsf{Dist},m}(x, \hat{x}))$$

In some situations, one computes the *MP* of a time series with itself (*i.e.,* , $x = \hat{x}$) to identify anomalies or similarities without a separate reference. We refer to this as the **self-join Matrix Profile** and denote it by $MP_{\mathsf{Dist},m}(x) = (MPD_{\mathsf{Dist},m}(x), MPI_{\mathsf{Dist},m}(x))$.

Subsequence notation. Consider a time series x of length n. We denote a subsequence $s^x_{i,m}$ as a continuous subset of values of x of length m, starting at index i. We denote the set of all the subsequences (deduced by a sliding window) of length m of a time series x of length n, as $Sub^x_m = \{s^x_{1,m}, ..., s^x_{n-m+1,m}\}$, and $\mathcal{I}$ as the set of indices $[\![n - m + 1]\!]$.

We introduce next the 1-NN subsequence, which will be used in the definition of *MP*:

Definition 1 (First Nearest Neighbor (1NN) subsequence). *Let x and $\hat{x}$ be two time series of lengths n and $\hat{n}$, respectively, and let* Dist *be a distance function. For a given subsequence length m, consider a subsequence $s^x_{i,m} \in x$ and all subsequences $s^{\hat{x}}_{k,m} \in Sub^{\hat{x}}m$ of $\hat{x}$. We say that $s^{\hat{x}}_{j,m}$ is the* 1-NN subsequence *of $s^x_{i,m}$ if it is the subsequence of $\hat{x}$ that is closest to $s^x_{i,m}$ under* Dist. *Formally, $1NN(s^x_{i,m}, \hat{x}) = s^{\hat{x}}_{j,m}$ iff*

$$\mathsf{Dist}(s^x_{i,m}, s^{\hat{x}}_{j,m}) = min \left\{ \mathsf{Dist}(s^x_{i,m}, s^{\hat{x}}_{k,m}) \right\}_{s^{\hat{x}}_{k,m} \in Sub^{\hat{x}}_m}$$

When $x = \hat{x}$, we additionally require $|j - i| > m$ to avoid trivial self-matches.

Now we give the definition of Matrix Profile:

Definition 2 (Matrix Profile Distance). *Let x be an input time series and $\hat{x}$ a reference time series of lengths n and $\hat{n}$, respectively. For a distance function* Dist *and a subsequence length m, the* Matrix Profile Distance *is the vector*

$$MPD_{\mathsf{Dist},m}(x, \hat{x}) = (MPD_i)_{i \in I}$$

where each entry MPD_i gives the distance between the subsequence $s^x_{i,m}$ and its first nearest neighbor (1NN) subsequence in $\hat{x}$. Formally,

$$MPD_{\mathsf{Dist},m}(x, \hat{x}) = \left(\mathsf{Dist}(s^x_{i,m}, 1NN(s^x_{i,m}, \hat{x}))\right)_{i \in I}$$

Another component of the Matrix Profile is the Matrix Profile Index (MPI), which stores the indices of the 1NN neighbors of the subsequences in Sub^x_m within $\hat{x}$. Although this information is useful in certain applications, it is not relevant for the scenarios considered in this article; therefore, its formal definition is omitted. Concisely, We combine the notions of Matrix Profile Distance and Matrix Profile into a unified representation $MP_{\mathsf{Dist},m}(x, \hat{x})$ and $MP_{\mathsf{Dist},m}(x)$ for self-join Matrix Profile. We use the classic (unnormalized) Euclidean distance for Dist, which is common in Matrix Profile applications, especially in health care. Although most MP algorithms were developed for the z-normalized Euclidean distance, normalization can be inappropriate in many settings, such as ECG or heart-rate analysis, where the raw scale carries clinical meaning [1, 19, 24]. A table that concludes all the notions used in the paper is provided in Table 1.

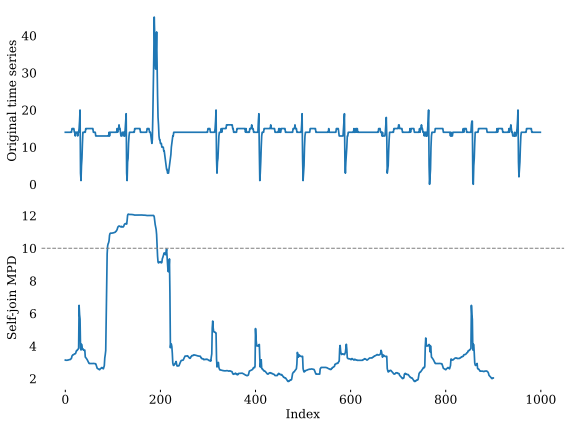

Figure 2: Detecting a self-anomaly using the Matrix Profile. Top: ECG with an anomaly in the third heartbeat. Bottom: self-join MPD (m = 100), where the anomaly onset appears as distances exceeding the threshold. The location of the anomaly onset is left-shifted by the length of the subsequence window used in the distance computation.

2.2 Background on Cryptography

A Non-Interactive Zero Knowledge Proof (NIZKP) allows a prover to prove the knowledge of a witness w s.t. $(w, s) \in \mathcal{R}$ where s is a statement and $\mathcal{R}$ a relation without revealing information on w.

Definition 3 (Non-Interactive Proofs (NIP) [2]). *Let $\mathcal{R}$ be a binary relation and let $\mathcal{L}$ be a language s.t. $s \in \mathcal{L} \Leftrightarrow (\exists w, (s, w) \in \mathcal{R})$. A* Non-Interactive Proof *NIP for $\mathcal{L}$ is a couple of algorithms* (NIP, NIPVerify) s.t.*:*

- $\underline{\mathsf{NIP}\{w : (s, w) \in \mathcal{R}\}}$**:** *on input a statement s and a witness w, it outputs a proof π.*
- $\underline{\mathsf{NIPVerify}(s, \pi)}$**:** *on input a statement s and a proof π, it outputs a bit b.*

A Non-Interactive Zero Knowledge Proof (NIZKP) is a NIP having the following properties:

- **Correctness.** For any $s, w, \mathcal{R}$ s.t. $(s, w) \in \mathcal{R}$ and $\pi \leftarrow \mathsf{NIP}\{w : (s, w) \in \mathcal{R}\}$, $\mathsf{NIPVerify}(s, \pi)$ returns 1.
- **Extractability.** There exists a PPT knowledge extractor Ext and a negligible function ϵ_{ext} s.t. for any algorithm $\mathcal{A}^{\mathsf{Simulator}(\cdot)}(\lambda)$ having access to a simulator that forges proofs for chosen statement and that outputs a fresh pair (s, π) with $\mathsf{NIPVerify}(s, \pi) = 1$, the extractor $\mathsf{Ext}^{\mathcal{A}}(\lambda)$ outputs w s.t. $(s, w) \in \mathcal{R}$ having access to $\mathcal{A}(\lambda)$ with probability at least $1 - \epsilon(\lambda)$.
- **Zero-knowledge.** A proof π *leaks no information, i.e.,* there exists a *PT* algorithm Simulator (called the *simulator*) s.t. $\mathsf{NIP}\{w : (s, w) \in \mathcal{R}\}$ and $\mathsf{Simulator}(s)$ follow the same probability distribution.

Definition 4 (Commitment Scheme). *A commitment scheme is a pair of two algorithms* (Com, Ope)*, s.t. on input a message, the algorithm* Com *produces a commitment using a secret key. The algorithm* Ope *on input a commitment and a secret key allows to reveal the committed message.*

A commitment scheme required to have the following property:

- **Hiding.** Given a message of its choice, a PPT algorithm $\mathcal{A}$ is not able to distinguish whether the commitment was generated from that message or is random, with a probability negligibly close to 1/2.
- **Binding.** An PPT algorithm $\mathcal{A}$ is not able to generate a commitment that can be open on two different messages with a non negligible probability.

Definition 5 (Discrete Logarithm (DL) assumption). *Let λ be a security parameter and $\mathbb{G} = \langle g \rangle$ be a group of prime order p, the Discrete Logarithm (DL) assumption states that there exists a negligible function ϵ s.t. for any PPT adversary $\mathcal{A}$, the following holds:*

$$\Pr[h \xleftarrow{\$} \mathbb{G}; x \leftarrow \mathcal{A}(h) \ : \ h = g^x] \leq \epsilon(\lambda).$$

3 Problem Statement

We denote a private time series x and a public time series $\hat{x}$, two sequences of data in $\mathbb{N}$, of length n and $\hat{n}$ respectively. Note that discretization is required when the original time series contain floating-point values.

Notation	Description
x	a private time series
$\hat{x}$	a public time series
n	length of the private time series x
$\hat{n}$	length of the public time series $\hat{x}$
m	subsequence length
$s^x_{i,m}$	subsequences $x_i, \ldots, x_{i+m-1}$
ϵ	value of the anomaly threshold
δ	value of the similarity threshold
A ($\perp_{\mathsf{A}}$)	(no) presence of anomaly
S ($\perp_{\mathsf{S}}$)	(no) presence of similarity
η	value in $\{\mathsf{A}, \perp_{\mathsf{A}}, \mathsf{S}, \perp_{\mathsf{S}}\}$
pp	public parameter
$\mathcal{I}$	set $[\![n-m+1]\!]$
$\mathcal{J}$	set $[\![\hat{n}-m+1]\!]$
$\mathcal{J}_i$	set $[\![1; i-m/2]\!]$
ℓ	encoding bit length
g	generator of the group $\mathbb{G}$
h	random element in $\mathbb{G}$ s.t. $\log_g(h)$ is unknown
x_i	i-th value of the time series
$x_{i,j}$	value $x_i - x_j$
c_i	Pedersen commitment of x_i
k_i	secret key to open the commitment c_i
$c_{i,j}$	Pedersen commitment of $x_{i,j}$
$k_{i,j}$	secret key to open the commitment $c_{i,j}$
$\tilde{c}_{i,j}$	Pedersen commitment of $x_{i,j}$
$\tilde{k}_{i,j}$	secret key to open the commitment $c_{i,j}$
$d_{i,j}$	squared Euclidean distance between $s_{i,m}$ and $s_{j,m}$
$D_{i,j}$	Pedersen commitment of $d_{i,j}$
$w_{i,j}$	secret key to open the commitment $D_{i,j}$
$d_{i,j,u}$	value of the uth bit of binary encoding of $d_{i,j}$
$D_{i,j,u}$	Pedersen commitment of $d_{i,j,u}$
$w_{i,j,u}$	secret key to open $D_{i,j,u}$
MPD_i	Matrix Profile Distance at index i
M_i	Pedersen commitment of MPD_i
z_i	secret key to open M_i
$MPD_{i,u}$	value of the uth bit of binary encoding of MPD_i
$M_{i,u}$	Pedersen commitment of $MPD_{i,u}$
$z_{i,u}$	secret key to open $M_{i,u}$

Table 1: List of some notations and their descriptions

We denote $\mathsf{Dist} : \mathbb{N}^m \times \mathbb{N}^m \to \mathbb{N}$ the function to calculate the distance between two sequences of length m.

We define the Matrix Profile function as:

$$MP : \mathsf{Dist} \times \mathbb{Z} \times \mathbb{N}^n \times (\mathbb{N}^{\hat{n}}) \to \mathbb{N}^{n-m+1}$$

which takes as input a distance function (*e.g.,* Euclidean distance), a subsequence length $m \in \mathbb{Z}$, a private time series $x \in \mathbb{N}^n$, and an optional public time series $\hat{x} \in \mathbb{N}^{\hat{n}}$. It returns a sequence of values in $\mathbb{N}$ of length $n - m + 1$, as defined in Definition 2.

We define the anomaly function $\mathsf{Ano} : \mathbb{N}^n \to \mathbb{B}$ (resp., the similarity function $\mathsf{Sim} : \mathbb{N}^n \to \mathbb{B}$), which takes a time series as input and returns 1 if there exists a value in Matrix Profile bigger than (resp., smaller than) a predefined threshold ϵ (resp., δ), and 0 otherwise. That is,

$$\mathsf{Ano}(x, (\mathsf{Dist}, m, \epsilon)) = \begin{cases} 1 & \text{if } \exists d \in MPD_{\mathsf{Dist},m}(x) \text{ s.t. } d > \epsilon, \\ 0 & \text{if } \forall d \in MPD_{\mathsf{Dist},m}(x),\ d \leq \epsilon, \end{cases} \tag{1}$$

$$\mathsf{Sim}(x, (\mathsf{Dist}, m, \delta)) = \begin{cases} 1 & \text{if } \exists d \in MPD_{\mathsf{Dist},m}(x) \text{ s.t. } d < \delta, \\ 0 & \text{if } \forall d \in MPD_{\mathsf{Dist},m}(x),\ d \geq \delta. \end{cases} \tag{2}$$

Here, $\epsilon, \delta \in \mathbb{R}$ are threshold values typically determined using expert knowledge or empirical analysis [5, 7], and can also be computed dynamically from the time series data [13, 27]. For the cases where $\mathsf{Ano} = 1$ or $\mathsf{Sim} = 1$, the detection should work in an unsupervised way that is the specific locations of anomalies or similarities are not known in advance and must not be revealed to the verifier, as disclosing them could leak timing information about sensitive events.

Specifically, some cases of the problem can be stated differently by considering the involved function *min* in MP. That is, for cases where $\mathsf{Ano} = 0$ and $\mathsf{Sim} = 1$,

$$\forall d \in MPD_{\mathsf{Dist},m}(x),\ d \leq \epsilon \Leftrightarrow \forall i \in I, \exists j \in \mathcal{J}_i \text{ s.t. } d_{i,j} \leq \epsilon \tag{3}$$

$$\exists d \in MPD_{\mathsf{Dist},m}(x) \text{ s.t. } d < \delta \Leftrightarrow \exists i \in \mathcal{I}, j \in \mathcal{J}_i \text{ s.t. } d_{i,j} < \delta \tag{4}$$

Our goal is to generate proofs that conceal the Ano and Sim functions and by using the verification algorithms, one can verify that the functions have been honestly calculated (or not), without knowing any other information.

4 Formal Model

In this section, we describe the formal definition of our proposed scheme.

4.1 Formal Definition

A Verifiable Anomaly or Similarity Computation (VASC) scheme allows a data provider to commit with a secret key the time series of a user using the algorithm Commit. Afterwards, the user can open the commitment using the key and can perform the computation of anomalies or similarities. The user who wishes to prove the existence (or absence) of an anomaly or a similarity uses the algorithm Prove to release a specific proof depending on a parameter pp, which is customized according to the context (*e.g.,* scenarios listed in Introduction). The generated proof demonstrates the correct computation of an anomaly or a similarity while revealing no information about the time series values beyond the claim. We denote by η the variable representing the type of claim, which takes values in A, S to indicate the existence of an anomaly or a similarity, and in $\perp_{\mathsf{A}}, \perp_{\mathsf{S}}$ to indicate their non-existence.

Definition 6 (**Verifiable Anomaly or Similarity Computation Scheme**). *A Verifiable Anomaly or Similary Computation* (VASC) *Scheme is a tuple of PPT algorithms* (Setup, Commit, Open, Prove, Verify) *defined as follows:*

Setup(λ): *takes as input a security parameter λ, and returns a setup* set *which contains the length of the time series vector.*

set *is implicitly taken in input of algorithms below.*

Commit(x): *takes as input a vector of values $x \in \mathbb{Z}_p^n$, generates a secret key K, a commitment C, and returns (K, C).*

Open(K, C): *takes as input a secret key K and a commitment C. It returns a vector of values x.*

Prove(K, C, η, pp): *takes as input a secret key K, a commitment C,a parameter $\eta \in \{\mathsf{A}, \perp_{\mathsf{A}}, \mathsf{S}, \perp_{\mathsf{S}}\}$ and a public parameter* pp $\in \Theta$. *It returns a proof π.*

Verify(C, π, η, pp): *takes as input a commitment C, a proof π, a parameter $\eta \in \{\mathsf{A}, \perp_{\mathsf{A}}, \mathsf{S}, \perp_{\mathsf{S}}\}$, a parameter* pp $\in \Theta$, *and returns a bit $b \in \{0, 1\}$.*

5 Our Verifiable Anomaly or Similarity Computation Scheme

In this section, we present $\mathsf{VASC}_{\mathsf{Ped}}$, our construction of a VASC scheme. Please refer to the table 1 for a comprehensive list of the notations and their corresponding descriptions.

Setup(λ): generates a group $\mathbb{G} = \langle g \rangle$ of prime order p, picks $h \xleftarrow{\$} \mathbb{G}$, sets n the length of the time series and returns set $= (\mathbb{G}, p, g, h, \lambda, n)$.

The Commit algorithm simply commit each value x_i of the time series using Pedersen commitment [15], which supports additive homomorphic operations on the commitments while preserving perfect hiding and computational binding guarantees.

Commit(x): parses x as $(x_i)_{i \in [\![n]\!]}$. For each $i \in [\![n]\!]$, picks $k_i \xleftarrow{\$} \mathbb{Z}_p^*$, and computes $c_i = g^{x_i} h^{k_i}$. Sets $C = (c_i)_{i \in [\![n]\!]}$ and $K = ((x_i)_{i \in [\![n]\!]}, (k_i)_{i \in [\![n]\!]})$, and returns (K, C).

Open(K, C): parses K as $((x_i)_{i \in [\![n]\!]}, (k_i)_{i \in [\![n]\!]})$, and returns $(x_i)_{i \in [\![n]\!]}$.

The algorithm Prove is based on the idea that, first, the algorithm compute commitments of distance between two subsequences of a time series. In order to achieve this, it takes as input the commitment c_i of each value x_i, it computes $c_{i,j}$, a commitment of $x_i - x_j$ by computing c_i / c_j for all $i, j \in [\![n]\!]$. Subsequently, it produces $\tilde{c}_{i,j}$, Pedersen commitment of the value $(x_i - x_j)^2$ by randomly picking $\tilde{k}_{i,j}$, and generates a non-interactive proof of knowledge to prove that the committed value in $\tilde{c}_{i,j}$ is the square of that committed in $c_{i,j}$. To computes the commitment $D_{i,j}$ of the (square of) Euclidean distance $d_{i,j}$ between two subsequences $s_{i,m}$ and $s_{j,m}$ given m the selected length of the subsequence, it performs homomorphic operations by computing the product between $(A_{i+r,j+r})_{r \in [\![0;m-1]\!]}$ for each $i \in \mathcal{I} = [\![n - m + 1]\!]$ and $j \in \mathcal{J}_i = [\![1; i - m/2 - 1]\!] \cup [\![i + m/2 + 1; n - m + 1]\!]$. We can remark that,

$$\begin{aligned}
\prod_{r=0}^{m-1} \tilde{c}_{i+r,j+r} &= \prod_{r=0}^{m-1} (c_{i+r,j+r})^{x_{i+r,j+r}} h^{\tilde{k}_{i+r,j+r}} \\
&= \prod_{r=0}^{m-1} (g^{x_{i+r,j+r}} h^{k_{i+r,j+r}})^{x_{i+r,j+r}} h^{\tilde{k}_{i+r,j+r}} \\
&= \prod_{r=0}^{m-1} g^{x_{i+r,j+r}^2} h^{k_{i+r,j+r} \cdot x_{i+r,j+r} + \tilde{k}_{i+r,j+r}} \\
&= g^{\sum_{r=0}^{m-1} x_{i+r,j+r}^2} h^{\sum_{r=0}^{m-1} k_{i+r,j+r} \cdot x_{i+r,j+r} + \tilde{k}_{i+r,j+r}} \\
&= g^{d_{i,j}} h^{w_{i,j}} \\
&= D_{i,j}.
\end{aligned}$$

Additionally, as we have $d_{i,j} = d_{j,i}$ and $d_{i,i} = 0$, then the algorithm Prove simply calculates the commitment of $D_{i,j}$ for $j < i$, and we set $\mathcal{J}_i$ as $[\![1; i - m/2 - 1]\!]$.

Secondly, the algorithm decomposes the distance value $d_{i,j}$ into binary and produces Pedersen commitment $D_{i,j,u}$ of the bit value $d_{i,j,u}$ using the secret key $w_{i,j,u} \in \mathbb{Z}_p^*$ where $u \in [\![\ell]\!]$. We notice that:

$$\begin{aligned}
\prod_{u=1}^{\ell} (D_{i,j,u})^{2^u} &= \prod_{u=1}^{\ell} \left(g^{d_{i,j,u}} h^{w_{i,j,u}}\right)^{2^u} \\
&= \prod_{u=1}^{\ell} \left(g^{d_{i,j,u}}\right)^{2^u} (h^{w_{i,j,u}})^{2^u} \\
&= \prod_{u=1}^{\ell} \left(g^{d_{i,j,u}}\right)^{2^u} \prod_{u=1}^{\ell} (h^{w_{i,j,u}})^{2^u} \\
&= g^{\sum_{u=1}^{\ell} d_{i,j,u} \cdot 2^u} h^{\sum_{u=1}^{\ell} w_{i,j,u} \cdot 2^u} \\
&= g^{d_{i,j}} h^{w_{i,j}} \\
&= D_{i,j}.
\end{aligned}$$

To compute the Matrix Profile Distance (MPD_i) for a fixed $i \in \mathcal{I}$, which consists of the minimum distance between $(d_{i,j})_{j \in \mathcal{J}_i}$, the algorithm uses the bit comparison described in 6.2.5. The algorithm then generates non-interactive proof of knowledge to prove that the value committed in M_i is well MPD_i.

Third, depending on η, the algorithm decomposes the threshold into binary and compares each bit of $d_{i,j}$ with each bit of MPD_i. (Keep in mind that as we consider the square of the Euclidian distance, we need to compare the distances with the square of the selected threshold.)

Please remember that in the case where η is equal to S or $\perp_{\mathsf{A}}$, we do not need to compute the Matrix Profile. If η is equal to S, the algorithm is in searchs if there exists indices $i \in \mathcal{I}, j \in \mathcal{J}_i$ s.t. $d_{i,j} < \delta$ and in the case where η is equal to $\perp_{\mathsf{A}}$, the algorithm finds if for all indices $i \in I$, there exists an index $j \in \mathcal{J}_i$ s.t. $d_{i,j} \leq \epsilon$.

If η is equal to A, the algorithm is in search if there exists an index $i \in \mathcal{I}$ s.t. MPD_i is greater than ϵ, and, if η is equal to $\perp_{\mathsf{S}}$, the algorithm finds if for all indices $i \in \mathcal{I}$, MPD_i is greater than or equal to δ. It then produces a non-interactive proof of knowledge to prove the comparison.

The bit length ℓ is determined as the number of bits required to encode the threshold value ϵ (or δ in the case of similarity) as well as the subsequence distances, specifically. Precisely

$$\ell = max\left(\log_2\left(m \cdot \left(\max_{i \in [\![n]\!]}(x_i)\right)^2\right), \log_2(\epsilon)\right) \tag{5}$$

assuming that $\max_{i \in [\![n]\!]}(x_i)$ is publicly known, and replacing ϵ to δ when using similarity threshold.

<u>Prove$(K, C, \eta, \mathsf{pp})$:</u> parses K as $((x_i)_{i\in[\![n]\!]}, (k_i)_{i\in[\![n]\!]})$ and C as $(c_i)_{i\in[\![n]\!]}$. If $\eta \in \{\mathsf{A}, \perp_{\mathsf{A}}\}$, parses pp as $\left(\mathsf{Eucl}, m, \epsilon, \max_{i\in[\![n]\!]}(x_i)\right)$, else parses pp as $\left(\mathsf{Eucl}, m, \delta, \max_{i\in[\![n]\!]}(x_i)\right)$.

For each $i \in [\![n]\!], j \in [\![n]\!]$ s.t. $j < i$, computes $c_{i,j} = \frac{c_i}{c_j}$, $x_{i,j} = x_i - x_j$ and $k_{i,j} = k_i - k_j$, picks $\tilde{k}_{i,j} \xleftarrow{\$} \mathbb{Z}_p^*$, computes $\tilde{c}_{i,j} = c_{i,j}^{x_{i,j}} h^{\tilde{k}_{i,j}}$, and generates the following proof:

$$\pi_{\mathsf{sq},i,j} \leftarrow \mathsf{NIP}\left\{ x_{i,j}, k_{i,j}, \tilde{k}_{i,j} \quad : \quad \begin{array}{c} c_{i,j} = g^{x_{i,j}} h^{k_{i,j}} \\ \wedge \tilde{c}_{i,j} = c_{i,j}^{x_{i,j}} h^{\tilde{k}_{i,j}} \end{array} \right\}.$$

Sets $\mathcal{I} = [\![n-m+1]\!]$ and $\mathfrak{J} = (\mathcal{J}_i)_{i\in\mathcal{I}} = ([\![1; i-m/2-1]\!])_{i\in\mathcal{I}}$.For each $i \in \mathcal{I}$ and $j \in \mathcal{J}_i$, computes $d_{i,j} = \sum_{r=0}^{m-1} x_{i+r,j+r}^2$, $w_{i,j} = \sum_{r=0}^{m-1} x_{i+r,j+r} k_{i+r,j+r} + \tilde{k}_{i+r,j+r}$ and $D_{i,j} = \prod_{r=0}^{m-1} \tilde{c}_{i+r,j+r}$.

Computes the binary decomposition $(d_{i,j,u})_{u\in[\![\ell]\!]} \in \{0,1\}^\ell$ of $d_{i,j}$ s.t. $d_{i,j} = \sum_{u=1}^{\ell} d_{i,j,u} \cdot 2^u$, using ℓ calculating with equation 5.

For each $u \in [\![\ell - 1]\!]$, picks $w_{i,j,u} \xleftarrow{\$} \mathbb{Z}_p^*$ and sets $w_{i,j,\ell} = \frac{w_{i,j} - \sum_{u=1}^{\ell-1} w_{i,j,u} \cdot 2^u}{2^\ell}$.

For each $i \in \mathcal{I}, j \in \mathcal{J}_i, u \in [\![\ell]\!]$, sets $D_{i,j,u} = g^{d_{i,j,u}} h^{w_{i,j,u}}$ and generates the following proofs:

$$\pi_{d_{\mathsf{bin}},i,j,u} \leftarrow \mathsf{NIP}\left\{ w_{i,j,u} : \begin{array}{c} \mathsf{Ope}(w_{i,j,u}, D_{i,j,u}) = 0 \\ \vee \quad \mathsf{Ope}(w_{i,j,u}, D_{i,j,u}) = 1 \end{array} \right\}.$$

- If $\eta \in \{\mathsf{A}, \perp_{\mathsf{S}}\}$, for each $i \in \mathcal{I}$, sets $MPD_i = \min_{j\in\mathcal{J}_i} d_{i,j}$, and computes its binary decomposition $(MPD_{i,u})_{u\in]\!]\ell[\![} \in \{0,1\}^\ell$ s.t. $MPD_i = \sum_{u=1}^{\ell} MPD_{i,u} \cdot 2^u$. For each $i \in \mathcal{I}$ and $u \in [\![\ell]\!]$, commits $MPD_{i,u}$ by picking $z_{i,u} \xleftarrow{\$} \mathbb{Z}_p^*$ and computes $M_{i,u} = g^{MPD_{i,u}} h^{z_{i,u}}$, and generates the following proof:

$$\pi_{MPD_{\mathsf{bin}},i,u} \leftarrow \mathsf{NIP}\left\{ z_{i,u} : \begin{array}{c} \mathsf{Ope}(z_{i,u}, M_{i,u}) = 0 \\ \vee \quad \mathsf{Ope}(z_{i,u}, M_{i,u}) = 1 \end{array} \right\}.$$

For each $i \in \mathcal{I}$, generates the proof:

$$\pi_{MPD_{\mathsf{exist}},i} \leftarrow \mathsf{NIP}\left\{ (z_{i,u} - w_{i,j,u})_{j\in\mathcal{J}_i, u\in[\![\ell]\!]} : \bigvee_{j\in\mathcal{J}_i} \left(\mathsf{Ope}((z_{i,u})_{u\in[\![\ell]\!]}, (M_{i,u})_{u\in[\![\ell]\!]}) = \mathsf{Ope}\left((w_{i,j,u})_{\substack{j\in\mathcal{J}_i \\ u\in[\![\ell]\!]}}, (D_{i,j,u})_{\substack{j\in\mathcal{J}_i \\ u\in[\![\ell]\!]}} \right) \right) \right\}.$$

For each $i \in \mathcal{I}$ and $j \in \mathcal{J}_i$, generates the following proof:

$$\pi_{MPD_{\min},i,j} \leftarrow \mathsf{NIP}\left\{ (z_{i,u} - w_{i,j,u})_{j\in\mathcal{J}_i, u\in[\![\ell]\!]} : \mathsf{Ope}((z_{i,u})_{u\in[\![\ell]\!]}, (M_{i,u})_{u\in[\![\ell]\!]}) \leq \mathsf{Ope}\left((w_{i,j,u})_{\substack{j\in\mathcal{J}_i \\ u\in[\![\ell]\!]}}, (D_{i,j,u})_{\substack{j\in\mathcal{J}_i \\ u\in[\![\ell]\!]}} \right) \right\}.$$

- If $\eta = \mathsf{A}$, it generates the proof:

$$\pi_{\mathsf{th}} \leftarrow \mathsf{NIP}\left\{ (z_{i,u})_{i\in\mathcal{I}, u\in[\![\ell]\!]} : \bigvee_{i\in\mathcal{I}} \mathsf{Ope}((z_{i,u})_{i\in\mathcal{I}, u\in[\![\ell]\!]}, (M_{i,u})_{i\in\mathcal{I}, u\in[\![\ell]\!]}) > \epsilon \right\}.$$

- If $\eta = \perp_{\mathsf{S}}$, it generates the proofs:

$$\pi_{\mathsf{th}} \leftarrow \mathsf{NIP}\left\{ (z_{i,u})_{i\in\mathcal{I}, u\in[\![\ell]\!]} : \bigwedge_{i\in\mathcal{I}} \mathsf{Ope}((z_{i,u})_{i\in\mathcal{I}, u\in[\![\ell]\!]}, (M_{i,u})_{i\in\mathcal{I}, u\in[\![\ell]\!]}) \geq \delta \right\},$$

It sets $\pi_{\mathsf{cmp}} = ((M_{i,u})_{i\in\mathcal{I}, u\in[\![\ell]\!]}, (\pi_{MPD_{\mathsf{exist}},i})_{i\in\mathcal{I}}, (\pi_{MPD_{\min},i,j})_{i\in\mathcal{I}, j\in\mathcal{J}_i}, \pi_{\mathsf{th}})$.

- If $\eta = \perp_{\mathsf{A}}$, it produces the following proofs:

$$\pi_{\mathsf{cmp}} \leftarrow \mathsf{NIP}\left\{ (w_{i,j,u})_{i\in\mathcal{I}, j\in\mathcal{J}_i, u\in[\![\ell]\!]} : \bigwedge_{i\in\mathcal{I}} \left(\bigvee_{j\in\mathcal{J}_i} \mathsf{Open}((w_{i,j,u})_{u\in[\![\ell]\!]}, (D_{i,j,u})_{u\in[\![\ell]\!]}) \leq \epsilon \right) \right\},$$

- If $\eta = \mathsf{S}$, it generates the following proof:

$$\pi_{\mathsf{cmp}} \leftarrow \mathsf{NIP}\left\{ (w_{i,j,u})_{i\in\mathcal{I}, j\in\mathcal{J}_i, u\in[\![\ell]\!]} : \bigvee_{i\in\mathcal{I}} \left(\bigvee_{j\in\mathcal{J}_i} \mathsf{Open}((w_{i,j,u})_{u\in[\![\ell]\!]}, (D_{i,j,u})_{u\in[\![\ell]\!]}) < \delta \right) \right\}.$$

It sets $\pi = ((\tilde{c}_{i,j})_{i\in[\![n]\!], j\in[\![n]\!], j<i}, (D_{i,j,u})_{i\in\mathcal{I}, j\in\mathcal{J}_i, u\in[\![\ell]\!]}, (\pi_{\mathsf{sq},i,j})_{i\in[\![n]\!], j\in[\![n]\!], j<i}, (\pi_{d_{\mathsf{bin}},i,j,u})_{i\in\mathcal{I}, j\in\mathcal{J}_i, u\in[\![\ell]\!]}, \pi_{\mathsf{cmp}})$.

<u>Verify$(C, \pi, \eta, \mathsf{pp})$:</u> parses C as $(c_i)_{i\in[\![n]\!]}$, π as $((\tilde{c}_{i,j})_{i\in[\![n]\!], j\in[\![n]\!], j<i}, (D_{i,j,u})_{i\in\mathcal{I}, j\in\mathcal{J}_i, u\in[\![\ell]\!]}, (\pi_{\mathsf{sq},i,j})_{i\in[\![n]\!], j\in[\![n]\!], j<i}, (\pi_{d_{\mathsf{bin}},i,j,u})_{i\in\mathcal{I}, j\in\mathcal{J}_i, u\in[\![\ell]\!]}, \pi_{\mathsf{cmp}})$, and if $\eta \in \{\mathsf{A}, \perp_{\mathsf{A}}\}$, parses pp as $\left(\mathsf{Eucl}, m, \epsilon, \max_{i\in[\![n]\!]}(x_i)\right)$, else parses pp as $\left(\mathsf{Eucl}, m, \delta, \max_{i\in[\![n]\!]}(x_i)\right)$

Computes $\ell' = \max\left(\log_2\left(m \cdot \left(\max_{i\in[\![n]\!]}(x_i)\right)^2\right), \log_2(\epsilon)\right)$, and verifies that $\ell = \ell'$. For each $i \in [\![n]\!], j \in [\![n]\!]$ s.t. $j < i$, sets $c_{i,j} = \frac{c_i}{c_j}$, and for each $i \in \mathcal{I}$, $j \in \mathcal{J}_i$, computes $D_{i,j} = \prod_{r=0}^{m-1} \tilde{c}_{i+r,j+r}$. Verifies the proofs, and returns 1 if the proofs are valid and if for all $i \in \mathcal{I}$ and all $j \in \mathcal{J}_i, D_{i,j} = \prod_{u=1}^{\ell} (D_{i,j,u})^{2^u}$. otherwise, it returns 0.

In appendix A, we propose a possible alternative of our scheme, with a minor change, that allows to prove anomaly or similarity in a private time series relative to a public time series.

5.1 Security Analysis

Theorem 1. Since the NIP used in $\mathsf{VASC}_{\mathsf{Ped}}$ are correct, the proof π in $\mathsf{VASC}_{\mathsf{Ped}}$ is correct.

Theorem 2. Since Pedersen commitments used in $\mathsf{VASC}_{\mathsf{Ped}}$ is perfectly hiding and the NIP used in $\mathsf{VASC}_{\mathsf{Ped}}$ are zero-knowledge, then π in $\mathsf{VASC}_{\mathsf{Ped}}$ is zero-knowledge.

Theorem 3. Since Pedersen commitments used in $\mathsf{VASC}_{\mathsf{Ped}}$ is computationnally binding in a group s.t. the DL assumption holds and the NIP used in $\mathsf{VASC}_{\mathsf{Ped}}$ are extractable, then π in $\mathsf{VASC}_{\mathsf{Ped}}$ is extractable.

Full details of the proofs of the theorems are given in Appendix C.3.

6 Instantiation

In this section, we describe how we instantiate the signature and the NIP used in $\mathsf{VASC}_{\mathsf{Ped}}$. All of these are derived from Schnorr-based sigma protocol, which are rendered non-interactive by applying the Fiat-Shamir transformation [9].

6.1 Instantiation of the signature

In the scenarios we have outlined that the values of the time series must be authenticated by the device. To this end, we instantiated the signature of the commitment generated by the device using a Schnorr's signature [16]. A signature scheme is a tuple of three algorithms (Gen, Sign, Ver) $s.t.$ the algorithm Gen on input the security parameter generates a public/private keys pair (pk, sk), Sign generates a signature σ given a message m, and sk, and Ver on input m, pk and σ returns an acceptance bit. A signature requires to be EUF-CMA meaning that given a public key and an access to an oracle that produces valid signatures on messages of its choice, a PT adversary cannot generate a valid signature that has not already been generated by the oracle.

6.2 NIP building blocks

Let $\mathbb{G} = \langle g \rangle$ be a group of prime order p, and $h \in \mathbb{G}$ $s.t.$ nobody know $\log_g(h)$. The Schnorr sigma protocol [16] allows a prover to prove to a verifier that they know the discrete logarithm x of an element $y \in \mathbb{G}$. It proceeds in three phases: first, the prover sends to the verifier a commitment $R = g^r$ where r is chosen randomly in $\mathbb{Z}_p^*$, next, the verifier generates a challenge $c \in \mathbb{Z}_p^*$ and sends it to the prover, finally the prover returns a response $z = r + c.x$ to the verifier. The verifier accepts the proof if $g^z = R \cdot y^{-c}$.

The protocol defined in Figure 4 in Appendix B.1, describes a sigma protocol to combine multiple proofs of knowledge of discrete logarithm into a single proof (an AND-proof) by using the same challenge. The paper [6] presents a method to prove the knowledge of one witness among several without revealing which one (an OR-proof). This protocol is detailed in Figure 5 in Appendix B.1.

In our paper, the proofs $(\pi_{d_{\text{bin}},i,j,u})_{i\in\mathcal{I},j\in\mathcal{J}_i,u\in[\![\ell]\!]}$ (resp. $(\pi_{MPD_{\text{bin}},i,u})_{i\in\mathcal{I},u\in[\![\ell]\!]}$) are instantiated using several OR-proofs. Our aim is to prove that the committed values in $(D_{i,j,u})_{i\in\mathcal{I},j\in\mathcal{J}_i,u\in[\![\ell]\!]}$ (resp. $(M_{i,u})_{i\in\mathcal{I},u\in[\![\ell]\!]}$) are committed in binary, therefore we consider the proof:

$$\mathsf{NIP}\left\{w_{i,j,u} \;:\; D_{i,j,u} = h^{w_{i,j,u}} \vee \frac{D_{i,j,u}}{g} = h^{w_{i,j,u}}\right\}$$
$$(\text{resp. } \mathsf{NIP}\left\{z_{i,u} \;:\; M_{i,u} = h^{z_{i,u}} \vee \frac{M_{i,u}}{g} = h^{z_{i,u}}\right\}).$$

Furthermore, the proof $(\pi_{MPD_{\text{exist}},i})_{i\in\mathcal{I}}$ result of a combination of OR-proof and AND-proof, and allow to prove that for each $i \in \mathcal{I}$, the committed values in $(M_{i,u})_{u\in[\![\ell]\!]}$ match with one committed in $(D_{i,j,u})_{j\in\mathcal{J}_i,u\in[\![\ell]\!]}$, the proof is defined as follows:

$$\mathsf{NIP}\left\{(z_{i,u} - w_{i,j,u})_{\substack{j\in\mathcal{J}_i\\ u\in[\![\ell]\!]}} \;:\; \bigvee_{j\in\mathcal{J}_i}\left(\bigwedge_{u=1}^{\ell}\frac{M_{i,u}}{D_{i,j,u}} = h^{z_{i,u}-w_{i,j,u}}\right)\right\}.$$

In [4], Chaum and Pedersen proposed an interactive protocol to prove the knowledge of the same discrete logarithm of two elements y_1 and y_2 in $\mathbb{G}$. The protocol for the proofs $(\pi_{\text{sq},i,j})_{i,j\in[\![n]\!],j<i}$ derives from this protocol and AND-proof, and is detailed in Figure 3 in Appendix B.1.

In what follows, we consider a, a positive private integer (resp. ϵ, a positive public integer), and we denote by $(a_u)_{u\in[\![\ell]\!]}$ (resp. $(\epsilon_u)_{u\in[\![\ell]\!]}$) where the rightmost bit represents the significant bit. We consider $(\widetilde{y}_u)_{u\in[\![\ell]\!]}$ the Pedersen commitments of the values $(a_u)_{u\in[\![\ell]\!]}$ using the generators g and h and the witnesses $(\omega_u)_{u\in[\![\ell]\!]}$.

6.2.1 Proving $a < \epsilon$ where a is private. In order to prove that a is strictly less than ϵ, we use bit comparison. A naive solution could be to prove that there exists an index $v \in [\![\ell]\!]$ $s.t.$ $a_v - \epsilon_v = -1$ and for all $u \in [\![v+1;\ell]\!]$, $a_u - \epsilon_u = 0$.

The proof corresponding to the aforementioned constraint is defined as follows:

$$\mathsf{NIP}\left\{(\omega_u)_{u\in[\![\ell]\!]} : \bigvee_{v=1}^{\ell}\left(\frac{y_v}{g^{\epsilon_v}} = h^{\omega_v} \wedge \bigwedge_{u=v+1}^{\ell}\frac{y_u}{g^{\epsilon_u}} = h^{\omega_u}\right)\right\}.$$

We observe that this proof is quadratic in ℓ.

In [3], Bultel *et al.* proposed a more efficient zero-knowledge proof to prove that $a < \epsilon$. They consider a constraint where $(u_k)_{k\in[\![\alpha]\!]}$ represents the indices of the ones in $(\epsilon_u)_{u\in[\![\ell]\!]}$ (please notice that as ϵ is public, the verifier as the knowledge of these indices). They show that proving that a is strictly less than ϵ involves proving that:

$$\begin{aligned}&\bigwedge_{u=u_\alpha+1}^{\ell} a_u = 0 \wedge \Bigg(a_{u_\alpha} = 0\\ &\vee \Bigg(\bigwedge_{u=u_{\alpha-1}+1}^{u_\alpha-1} a_u = 0 \wedge \Bigg(a_{u_{\alpha-1}} = 0 \qquad (6)\\ &\vee \Bigg(\ldots \vee \Bigg(\bigwedge_{u=u_1+1}^{u_2-1} a_u = 0 \wedge a_{u_1} = 0\Bigg)\ldots\Bigg)\Bigg)\Bigg)\Bigg).\end{aligned}$$

The logic behind is that when ϵ_u is equal to zero, then a_u must be equal to zero and when ϵ_u is equal to one, then a_u can be equal to zero or one. If there exist an index u_k $s.t.$ ϵ_{u_k} is equal to one and a_{u_k} is equal to zero, then the constraint $a < \epsilon$ is reached, the recurrence is stopped, else the recurrence continue until that for the last index of one in ϵ (the index u_1), we have $a_{u_1} = 0$.

For instance, if the binary decomposition of ϵ is 0101100, we have $\alpha = 3$ and $(u_k)_{k\in[\![3]\!]} = (2, 4, 5)$. The constraint 6 is expressed as follows:

$a_7 = 0 \wedge a_6 = 0 \wedge (a_5 = 0 \vee (a_4 = 0 \vee (a_3 = 0 \wedge a_2 = 0)))$.

In order to prove that a is strictly lesser than ϵ, by assigning

the value $(y_u)_{u\in[\![\ell]\!]} = (\tilde{y}_u)_{u\in[\![\ell]\!]}$, Bultel et al. [3] proposed the following proof that reflects the constraint 6:

$$\pi_0 \leftarrow \mathsf{NIP}\left\{(\omega_u)_{u\in[\![\ell]\!]} : \begin{array}{l} \bigwedge_{u=u_\alpha+1}^{\ell} y_u = h^{\omega_u} \\ \wedge\Bigg(y_{u_\alpha} = h^{\omega_{u_\alpha}} \\ \vee\Bigg(\bigwedge_{u=u_{\alpha-1}+1}^{u_\alpha-1} y_u = h^{\omega_u} \\ \wedge\Bigg(y_{u_{\alpha-1}} = h^{\omega_{i,j,u_{\alpha-1}}} \\ \vee(\ldots \\ \vee\Bigg(\bigwedge_{u=u_1+1}^{u_2-1} y_u = h^{\omega_u} \\ \wedge y_{u_1} = h^{\omega_{u_1}}\Bigg)\ldots\Bigg)\Bigg)\Bigg)\Bigg) \end{array}\right\}.$$

This proof is linear in the number of terms in the relation.

6.2.2 Proving $a \leq \epsilon$ where a is private. To prove that a is less than or equal to ϵ, we follow the same approach than in 6.2.1. Let $(u_k)_{k\in[\![\alpha]\!]}$ be the indices of the zeros in $(\epsilon_u)_{u\in[\![\ell]\!]}$, proving that $a \leq \epsilon$ involves proving that:

$$\begin{array}{l} \bigvee_{u=u_\alpha+1}^{\ell} a_u = 0 \vee \Bigg(a_{u_\alpha} = 0 \\ \wedge\Bigg(\bigvee_{u=u_{\alpha-1}+1}^{u_\alpha-1} a_u = 0 \vee \Bigg(a_{u_{\alpha-1}} = 0 \\ \wedge\Bigg(\ldots\wedge\Bigg(\bigvee_{u=u_1+1}^{u_2-1} a_u = 0 \vee a_{u_1} = 0\Bigg)\ldots\Bigg)\Bigg)\Bigg)\Bigg). \end{array} \tag{7}$$

The logic behind is that when ϵ_u is equal to zero, then a_u must be equal to zero and when ϵ_u is equal to one, then a_u can be equal to zero or one. If there exist an index u_k $s.t.$ ϵ_{u_k} is equal to one and a_{u_k} is equal to zero, then the constraint $a < \epsilon$ is reached, the recurrence is stopped, else the recurrence continue until that for the last index of zero in ϵ (the index u_1), we have $a_{u_1} = 0$.

For instance, if the binary decomposition of ϵ is 110011001 $s.t.$ the rightmost bit is the significant bit, we have $\alpha = 4$ and $(u_k)_{k\in[\![4]\!]} = (3, 4, 7, 8)$. The constraint 7 is expressed as follows: $a_9 = 0 \vee (a_8 = 0 \wedge (a_7 = 0 \wedge (a_6 = 0 \vee a_5 = 0 \vee (a_4 = 0 \wedge a_3 = 0))))$.

By assigning the value $(y_u)_{u\in[\![\ell]\!]} = (\tilde{y}_u)_{u\in[\![\ell]\!]}$, to prove that a is less than or equal to ϵ, we build the proof that use the boolean constraint 7 as follows:

$$\pi_1 \leftarrow \mathsf{NIP}\left\{(\omega_u)_{u\in[\![u_1;\ell]\!]} : \begin{array}{l} \bigvee_{u=u_\alpha+1}^{\ell} y_u = h^{\omega_u} \\ \vee\Bigg(y_{u_\alpha} = h^{\omega_{u_\alpha}} \\ \wedge\Bigg(\bigvee_{u=u_{\alpha-1}+1}^{u_\alpha-1} y_u = h^{\omega_u} \\ \vee\Bigg(y_{u_{\alpha-1}} = h^{\omega_{i,j,u_{\alpha-1}}} \\ \wedge(\ldots \\ \wedge\Bigg(\bigvee_{u=u_1+1}^{u_2-1} y_u = h^{\omega_u} \\ \vee y_{u_1} = h^{\omega_{u_1}}\Bigg)\ldots\Bigg)\Bigg)\Bigg)\Bigg) \end{array}\right\}.$$

6.2.3 Proving $a > \epsilon$ where a is private. To prove that a is strictly greater than ϵ, we consider the negation of the constraint 7, we obtain the following constraint:

$$\begin{array}{l} \bigwedge_{u=u_\alpha+1}^{\ell} a_u = 1 \wedge \Bigg(a_{u_\alpha} = 1 \\ \vee\Bigg(\bigwedge_{u=u_{\alpha-1}+1}^{u_\alpha-1} a_u = 1 \wedge \Bigg(a_{u_{\alpha-1}} = 1 \\ \vee\Bigg(\ldots\vee\Bigg(\bigwedge_{u=u_1+1}^{u_2-1} a_u = 1 \wedge a_{u_1} = 1\Bigg)\ldots\Bigg)\Bigg)\Bigg)\Bigg), \end{array} \tag{8}$$

where the indices $(u_k)_{k\in[\![\alpha]\!]}$ represents the indices of the zeros in the binary decomposition of ϵ. We can derive a proof on the same structure as π_0, excepts that we attribute the values $(y_u)_{u\in[\![u_1;\ell]\!]} = \left(\frac{\tilde{y}_u}{g}\right)_{u\in[\![u_1;\ell]\!]}$.

6.2.4 Proving $a \geq \epsilon$ where a is private. Proving that a is strictly greater than ϵ, we consider the negation of the constraint 6, we obtain the following constraint:

$$\begin{array}{l} \bigvee_{u=u_\alpha+1}^{\ell} a_u = 1 \vee \Bigg(a_{u_\alpha} = 1 \\ \wedge\Bigg(\bigvee_{u=u_{\alpha-1}+1}^{u_\alpha-1} a_u = 1 \vee \Bigg(a_{u_{\alpha-1}} = 1 \\ \wedge\Bigg(\ldots\wedge\Bigg(\bigvee_{u=u_1+1}^{u_2-1} a_u = 1 \vee a_{u_1} = 1\Bigg)\ldots\Bigg)\Bigg)\Bigg)\Bigg), \end{array} \tag{9}$$

where the indices $(u_k)_{k\in[\![\alpha]\!]}$ represents the indices of the ones in the binary decomposition of ϵ. We can derive a proof on the same structure as π_1, excepts that we assign the values $(y_u)_{u\in[\![\ell]\!]} = \left(\frac{\tilde{y}_u}{g}\right)_{u\in[\![\ell]\!]}$.

In the following, we consider two positive private integer d and d', and we note $(d_u)_{u\in[\![\ell]\!]}$ and $(d'_u)_{u\in[\![\ell]\!]}$ their corresponding bit decompositions where the rightmost bit is the significant bit.

6.2.5 Proving $d \leq d'$ where d and d' are private. In order to prove that d is less than or equal to d', we can use the

same approach as 6.2.2 but with a slight variant. As d and d' are both private, we cannot exploit the knowledge of the indices of the bits equal to zero. We can simply assume that if d is less than or equal to d' then there will exists an index $u \in [\![\ell]\!]$ $s.t.$ $d_u - d'_u = -1$ and the values $(d_u - d'_u)_{v \in [\![u+1;\ell]\!]}$ are all equal to 0. If such index in find, the constraint is reached. We use the following boolean constraint:

$$\begin{aligned} & d_\ell - d'_\ell = -1 \vee \big(d_\ell - d'_\ell = 0 \\ & \wedge \big(d_{\ell-1} - d'_{\ell-1} = -1 \vee \big(d_{\ell-1} - d'_{\ell-1} = 0 \\ & \wedge \big(\ldots \wedge \big(d_1 - d'_1 = -1 \vee d_1 - d'_1 = 0\big)\big)\big) \ldots\big)\big)\big), \end{aligned} \tag{10}$$

and we the following proof:

$$\pi_2 \leftarrow \mathsf{NIP}\left\{ (\gamma_u - \gamma_{u'})_{u \in [\![\ell]\!]} : \begin{array}{l} \frac{gD_\ell}{D'_\ell} = h^{\gamma_\ell - \gamma'_\ell} \\ \vee \left(\frac{D_\ell}{D'_\ell} = h^{\gamma_\ell - \gamma'_\ell}\right. \\ \wedge \left(\frac{gD_{\ell-1}}{D'_{\ell-1}} = h^{\gamma_{\ell-1} - \gamma'_{\ell-1}}\right. \\ \vee \left(\frac{D_{\ell-1}}{D'_{\ell-1}} = h^{\gamma_{\ell-1} - \gamma'_{\ell-1}}\right. \\ \wedge (\ldots \\ \wedge \left(\frac{gD_1}{D'_1} = h^{\gamma_1 - \gamma'_1}\right. \\ \left.\left.\left.\vee \frac{D_1}{D'_1} = h^{\gamma_1 - \gamma'_1}\right)\right) \ldots\right)\right)\right) \end{array} \right\}.$$

All our NIP building blocks are correct, zero-knowledge and extractable, the proof of this claim can be found in Appendix C.1

6.3 Instantiation of the proof $\pi_{MPD_{\min}}$

Remember that in this proof, we want to prove that MPD_i (which is committed in $(M_{i,u})_{u \in [\![\ell]\!]}$) is smaller or equal to the value $d_{i,j}$ (which is committed in $(D_{i,j,u})_{j \in \mathcal{J}_i, u \in [\![\ell]\!]}$). To achieve this, we instantiated the proofs $(\pi_{MPD_{\text{bin}},i,j})_{i \in \mathcal{I}, j \in \mathcal{J}_i}$ using the proof π_2. We executed the proof π_2 for each $i \in \mathcal{I}$ and $j \in \mathcal{J}_i$, by allocating the values $M_{i,u}$ to D_u, $D_{i,j,u}$ to D'_u, $z_{i,u}$ to γ_u and $w_{i,j,u}$ to γ'_u.

6.4 Instantiation of the proof π_{th}

6.4.1 Case $\eta = \mathsf{A}$. The proof π_{th} consists of the following proof:

$$\pi_{\mathsf{th}} \leftarrow \mathsf{NIP}\left\{ (z_{i,u})_{\substack{i \in \mathcal{I}, \\ u \in [\![\ell]\!]}} : \bigvee_{i \in \mathcal{I}} \mathsf{Open}((z_{i,u})_{u \in [\![\ell]\!]}, (M_{i,u})_{u \in [\![\ell]\!]}) > \epsilon \right\}.$$

We recall that in this case, the prover must prove the knowledge of witnesses $(z_i)_{i \in \mathcal{I}}$ $s.t.$ there exist an index $i \in \mathcal{I}$ $s.t.$ $MPD_i > \epsilon$. This proof use boolean relations and can be instantiated using combination of an OR-proof (Figure 5 in Appendix B.1) and a proof that prove that a positive private integer a is strictly greater than a positive public integer ϵ (6.2.3).

6.4.2 Case $\eta = \perp_{\mathsf{S}}$. The proof π_{th} consists of the following proof:

$$\pi_{\mathsf{th}} \leftarrow \mathsf{NIP}\left\{ (z_{i,u})_{\substack{i \in \mathcal{I}, \\ u \in [\![\ell]\!]}} : \bigwedge_{i \in \mathcal{I}} \mathsf{Open}((z_{i,u})_{u \in [\![\ell]\!]}, (M_{i,u})_{u \in [\![\ell]\!]}) \geq \delta \right\}.$$

In this case, the prover must prove the knowledge of witnesses $(z_i)_{i \in \mathcal{I}}$ $s.t.$ for all $i \in \mathcal{I}$, $MPD_i \geq \delta$. We instantiated the proof using a combination of an AND-proof (Figure 4 in Appendix B.1) and a proof that prove that a positive private integer a is greater than or equal to a positive public integer ϵ (6.2.4). We claim that π_{th} is correct, zero-knowledge and extractable, the proof of this claim can be found in Appendix C.2.4.

6.5 Instantiation of the proof π_{cmp}

6.5.1 Case $\eta = \perp_{\mathsf{A}}$. The proof π_{cmp} consists of the following proof:

$$\pi_{\mathsf{cmp}} \leftarrow \mathsf{NIP}\left\{ (w_{i,j,u})_{i \in \mathcal{I}, j \in \mathcal{J}_i, u \in [\![\ell]\!]} : \bigwedge_{i \in \mathcal{I}} \left(\bigvee_{j \in \mathcal{J}_i} \mathsf{Open}((w_{i,j,u})_{u \in [\![\ell]\!]}, (D_{i,j,u})_{u \in [\![\ell]\!]}) \leq \epsilon \right) \right\}.$$

For this proof, the prover has to prove has to prove the knowledge of witnesses $(w_{i,j})_{i \in \mathcal{I}, j \in \mathcal{J}_i}$ $s.t.$ for any index $i \in \mathcal{I}$, there exist an index $j \in \mathcal{J}_i$ $s.t.$ $d_{i,j} \leq \delta$. This proof use composition of boolean relation and can be instantiated using combination of OR-proofs (Figure 5 in Appendix B.1) and AND-proof (B.1) and a proof that allows to prove that a positive private integer a is less than or equal to a positive public integer ϵ (6.2.2). We claim that π_{cmp} is correct, zero-knowledge and extractable, we demonstrate this claim in Appendix C.2.5.

6.5.2 Case $\eta = \mathsf{S}$. The proof π_{cmp} consists of the following proof:

$$\pi_{\mathsf{cmp}} \leftarrow \mathsf{NIP}\left\{ (w_{i,j,u})_{i \in \mathcal{I}, j \in \mathcal{J}_i, u \in [\![\ell]\!]} : \bigvee_{i \in \mathcal{I}} \left(\bigvee_{j \in \mathcal{J}_i} \mathsf{Open}((w_{i,j,u})_{u \in [\![\ell]\!]}, (D_{i,j,u})_{u \in [\![\ell]\!]}) < \delta \right) \right\}.$$

We recall that in this case, the prover must prove the knowledge of witnesses $(w_{i,j})_{i \in \mathcal{I}, j \in \mathcal{J}_i}$ $s.t.$ there exists indices $i \in \mathcal{I}$ and $j \in \mathcal{J}_i$ $s.t.$ $d_{i,j} < \delta$. This proof use composition of boolean relation and can be instantiated using combination of OR-proofs (Figure 5 in Appendix B.1) and a proof that allows to prove that a positive private integer a is less than a public integer ϵ (6.2.1). We claim that π_{cmp} is correct, zero-knowledge and extractable, the proof of this claim is given in Appendix C.2.5.

7 Experiments

In this section, we describe how we instantiate the parameters in the proofs of the four scenarios, and the experiment settings to evaluate the performances.

7.1 Parameters instantiation

We set n, the length of the private time series, to 10, 100, 200 and 1000, to observe how the runtime scales. $n = 1000$ is a realistic time series length using in ECG analysis [21, 24], with around 10 complete heart-beat cycles. We initiate time series values x_i with positive integers in $[\![45]\!]$, a sufficient precision for raw ECG data. We use $m = 3$ for $n = 10$, $m = 10$ for $n = 100$ and $n = 200$, and $m = 100$ for $n = 1000$, noting that one QRS transition in a heart beat with the default sampling frequency is around 10 points, and a whole heart beat is around 100 points, justifying the choices of these parameters. For the bit length used to encode d_{ij} and MPD_i values, we use $\ell = 13$, $\ell = 15$ and $\ell = 20$ for $m = 3$, $m = 10$ and $m = 100$, calculated with the formula described in Section 5.

7.2 Results

We implemented all proofs introduced in Section 5 in Rust [1]. Experiments were run on a laptop equipped with a 12-core 13th Gen Intel Core i7-1365U processor and 32 GB of RAM.

The device first generates the commitment and then signs the message. The verifier subsequently performs signature verification. The runtime measurements for these steps are presented in Table 2. Then, we report the runtimes for all proofs in Table 3, and the corresponding total runtimes for no similarity and anomaly ($\perp_S$ and A since the time is very close), for no anomaly ($\perp_A$), and for similarity are then reported in Tables 4, 5 and 6 respectively. Note that the scenario $\perp_S$ (together with the scenario A) represents the worst theoretical complexity usecase, because the phases of MP calculating, proving and verifying are needed. While the scenario $\perp_A$ (together with S) showcases the usecase with a better performance. Precisely, the scenario $\perp_S$ (and A) consists of unit proofs π_{sq}, $\pi_{d_{bin}}$, $\pi_{MPD_{bin}}$, $\pi_{MPD_{exist}}$, $\pi_{MPD_{min}}$, and π_{th}. For π_{sq}, $\pi_{d_{bin}}$, and $\pi_{MPD_{bin}}$, the prover must generate new commitments that are required by the verifier during the verification phase. We therefore report the commitment time in the parentheses; the values outside the parentheses correspond then to the sum of the commit time and the proof-generation time. For $\pi_{MPD_{bin}}$, $\pi_{MPD_{exist}}$, and π_{th}, the commitments produced in the preceding proofs are reused, so no additional commitment time is included for these proofs. For the scenario $\perp_A$ (and S), it consists π_{sq}, $\pi_{d_{bin}}$ and π_{cmp}.

As described in the introduction, the scenarios we consider do not require strict time constraints, as both the proving and verification procedures can be executed in an offline setting. The best performance comes from S scenario, with a total proving and verification time of 27.5 seconds for a time series of 100 points, 134.13 seconds for 200 points, and approximately 1.1 hours for 1,000 points. Among all the unit proofs, the main performance bottlenecks arise in $\pi_{MPD_{min}}$ and $\pi_{d_{bin}}$ proofs (in line with their theoretical complexity). Scenarios that require these two components include $\perp_S$ and A. Nevertheless, even for $n = 1000$, the overall runtime remains practical, totaling approximately 2.4 hours (about 1.3 hours for the prover and 1.1 hours for the verifier). Finally, for the $\perp_A$ scenario, the applied optimizations further reduce the runtime to approximately 1.4 hours for $n = 1000$.

A complete analysis of the asymptotic complexity of our different algorithms is given in Appendix B.2.

[1] https://anonymous.4open.science/r/ZKMP-7F81/

n	Commit	Sign	Verify
10	501.06 μs	61 μs	87 μs
100	4.94 ms	307 μs	332 μs
200	9.89 ms	585 μs	603 μs
1000	52.24 ms	2.9 ms	3.0 ms

Table 2: Runtime of commit, sign and verify for C

Proof	n	Prove	Verify
π_{sq}	10	11.0 ms (5.93 ms)	6.14 ms
(where commit)	100	1.15 s (524 ms)	832 ms
	200	4.55 s (2.06 s)	3.41 s
	1000	114 s (51.12 s)	83.67 s
$\pi_{d_{bin}}$	10	73.5 ms (41.5 ms)	31 ms
(where commit)	100	13.5 s (6.63 s)	6.86 s
	200	64.9 s (29.88 s)	36.14 s
	1000	1940 s (981.28 s)	953.81 s
$\pi_{MPD_{bin}}$	10	19 ms (6.55 ms)	11.7 ms
(where commit)	100	226 ms (70.7 ms)	155 ms
	200	479 ms (154 ms)	325 ms
	1000	3.05 s (938 ms)	2.114 s
$\pi_{MPD_{exist}}$	10	50.5 ms	46.6 ms
	100	7.74 s	7.51 s
	200	36.99 s	36.92 s
	1000	1087.62 s	1064.97 s
$\pi_{MPD_{min}}$	10	56 ms	60.8 ms
	100	13.74 s	13.68 s
	200	64.54 s	66.06 s
	1000	1645.58 s	1824.85 s
π_{th}	10	5.51 ms	5.35 ms
($\perp_S$)	100	73.5 ms	66.8 ms
	200	146 ms	148 ms
	1000	1.11 s	1.03 s
π_{th}	10	5.18 ms	5 ms
(A)	100	71.4 ms	65.5 ms
	200	140 ms	141 ms
	1000	1.01 s	981 ms
π_{cmp}	10	52.3 ms	49.5 ms
($\perp_A$)	100	7.84 s	7.75 s
	200	32.77 s	34.92 s
	1000	909.85 s	902.32 s
π_{cmp}	10	16 ms	16.1 ms
(S)	100	3.23 s	3.21 s
	200	15.88 s	15.41 s
	1000	448.84 s	433.77 s

Table 3: Detailed runtime for all proofs

n	Proof Time	Verify Time	Total Time
10	215.49 ms	161.61 ms	377.1 ms
100	36.4 s	29.11 s	65.5 s
200	171.64 s	143.01 s	314.64 s
1000	4789.93 s	3930.43 s	8720.37 s (2.4h)

Table 4: Total runtime for no similarity and anomaly

n	Prove Time	Verify Time	Total Time
10	136.78 ms	86.7 ms	223.48 ms
100	22.46 s	15.44 s	37.9 s
200	102.25 s	74.47 s	176.73 s
1000	2962.43 s	1939.8 s	4902.23 s (1.4h)

Table 5: Total runtime for no anomaly

n	Prove Time	Verify Time	Total Time
10	96.36 ms	59.87 ms	156.23 ms
100	17.27 s	10.23 s	27.5 s
200	81.91 s	52.22 s	134.13 s
1000	2501.42 s	1471.25 s	3972.67 s (1.1h)

Table 6: Total runtime for similarity

8 Related Works

A line of work considers privacy-preserving similarity on time series. Zhu *et al.* [26] proposed a two-party protocol that privately evaluates Dynamic Time Warping (DTW) and discrete Fréchet distance between two time series using partially homomorphic encryption (HE). Liu and Yi [14] proposed a collaborative medical analytics system based on DTW that combines multiparty computation (MPC) to enable privacy-preserving DTW queries over distributed electrocardiogram (ECG) and photoplethysmogram (PPG) data. Zheng *et al.* [25] introduced a solution supporting similarity queries over time series of different lengths by using time-warp edit distance (TWED) as the similarity metric and symmetric HE.

However, none of these methods employ the MP, which supports subsequence-level similarity and anomaly detection, with or without a reference time series. Second, some of these applications [14, 26] consider a different setting in which several independent data providers want to determine a consensus without revealing their raw time series. Other works around similarity query on private data [25] are mostly based on MPC which does not allow for a non-interactive protocol, or the prover only sends an element that can be verified later. Finally, generic cryptographic tools could in practice be used to construct a protocol similar to ours, such as using generic Zero-Knowledge (ZK) proofs or MPC. However, these techniques typically require expressing the computation as Boolean or arithmetic circuits, whose construction is non-trivial and often incurs significant overhead, tending to be less efficient than protocols tailored to a specific structure. Some generic ZK proofs (*e.g.,* ZK-snarks [12]) are non-interactive and allow the generation of proofs of constant size, but their proving and verification times scale with the size of the proven circuit. In principle, such proofs could serve as an alternative to our work, allowing less data to be exchanged. However, these proofs achieve a lower level of security (computational ZK only), require less standard tools (*e.g.,* pairings), and rely on strong assumptions and heuristic security models (*e.g.,* common reference string model, algebraic group model, and random oracle). In comparison, as our protocol is a combination of a Pedersen commitment and Schnorr proofs, it is perfectly ZK (an adversary with unlimited resources cannot deduce information about the values of the time series). Our construction therefore relies only on the discrete logarithm assumption (which ensures that the commitment is binding) in its non-interactive version and on the random oracle (which is the most well-understood and established heuristic model compared to AGM/GGM).

Another line of work examines the privacy issues related to the Matrix Profile (MP). Introduced broadly to the community in 2016 by Keogh and colleagues [21], the MP has become a standard tool for time-series mining, particularly for detecting pairwise anomalies and similarities. Recently, some studies have claimed that the Matrix Profile provides certain privacy-preserving properties, such as robustness against sensitive-attribute inference [8] or reconstruction attacks [23], often arguing that the non-linear *min* operation involved in its computation is not reversible. However, Zhang *et al.* [22] recently demonstrated that the MP is actually vulnerable to singling-out, linkability, attribute inference, and reconstruction attacks. By solving the MP constraints using a powerful optimizer, they show that an attacker can reconstruct the original time series with up to 0.99 Pearson correlation, revealing that MP does not inherently guarantee privacy. Given that MP is frequently applied to sensitive domains such as healthcare data [11, 18] and geo-location time series [20], we therefore treat the values of the MP as sensitive and protected outputs in our threat model.

9 Conclusion and future works

This paper presents ZKPMP, a secure protocol for proving claims about anomalies and similarities in private time series. The protocol relies on Pedersen commitments and Schnorr-based non-interactive zero-knowledge proofs (NIZKs). We consider a strong adversarial model in which the private time series itself, the pairwise subsequence distances, the corresponding Matrix Profile values, as well as the indices of potential anomalies and similarities are all treated as sensitive information and must not be disclosed. Under these stringent constraints, we carefully design a set of zero-knowledge proofs and evaluate their performance on real-world time series with lengths of up to 1,000 data points, demonstrating the practicality of our approach for real-world applications. In particular, we leverage the Matrix Profile to prove the absence of similarity (anomaly

detection) at the subsequence level, a challenging and often overlooked setting in the literature. Finally, We believe that our protocol can also be adapted to other related tasks, such as authentication, by reusing the similarity proof and verification components.

References

[1] Nivedita Bijlani, Oscar Mendez Maldonado, and Samaneh Kouchaki. 2022. G-CMP: Graph-enhanced Contextual Matrix Profile for unsupervised anomaly detection in sensor-based remote health monitoring. *arXiv preprint arXiv:2211.16122* (2022).

[2] Manuel Blum, Paul Feldman, and Silvio Micali. 1988. Non-Interactive Zero-Knowledge and Its Applications. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing* (Chicago, Illinois, USA) *(STOC '88)*. Association for Computing Machinery, New York, NY, USA, 103–112. doi:10.1145/62212.62222

[3] Xavier Bultel and Charles Olivier-Anclin. 2025. Taming Delegations in Anonymous Signatures: k-Times Anonymity for Proxy and Sanitizable Signature. In *Cryptology and Network Security*, Markulf Kohlweiss, Roberto Di Pietro, and Alastair Beresford (Eds.). Springer Nature Singapore, Singapore, 165–186.

[4] David Chaum and Torben Pryds Pedersen. 1993. Wallet Databases with Observers. In *Advances in Cryptology — CRYPTO' 92*, Ernest F. Brickell (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 89–105.

[5] Ivaylo I Christov. 2004. Real time electrocardiogram QRS detection using combined adaptive threshold. *Biomedical engineering online* 3, 1 (2004), 28.

[6] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. 1994. Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols. In *Advances in Cryptology — CRYPTO '94*, Yvo G. Desmedt (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 174–187.

[7] Hoang Anh Dau, Anthony Bagnall, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, and Eamonn Keogh. 2019. The UCR time series archive. *IEEE/CAA Journal of Automatica Sinica* 6, 6 (2019), 1293–1305.

[8] Audrey Der, Chin-Chia Michael Yeh, Yan Zheng, Junpeng Wang, Huiyuan Chen, Zhongfang Zhuang, Liang Wang, Wei Zhang, and Eamonn Keogh. 2023. Time series synthesis using the matrix profile for anonymization. In *2023 IEEE International Conference on Big Data (BigData)*. IEEE, 1908–1911.

[9] Amos Fiat and Adi Shamir. 1987. How To Prove Yourself: Practical Solutions to Identification and Signature Problems. In *Advances in Cryptology — CRYPTO' 86*, Andrew M. Odlyzko (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 186–194.

[10] Marc Fischlin and Arno Mittelbach. 2021. An Overview of the Hybrid Argument. Cryptology ePrint Archive, Paper 2021/088. https://eprint.iacr.org/2021/088 https://eprint.iacr.org/2021/088.

[11] Ary L Goldberger, Luis AN Amaral, Leon Glass, Jeffrey M Hausdorff, Plamen Ch Ivanov, Roger G Mark, Joseph E Mietus, George B Moody, Chung-Kang Peng, and H Eugene Stanley. 2000. PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals. *circulation* 101, 23 (2000), e215–e220.

[12] Jens Groth. 2016. On the size of pairing-based non-interactive arguments. In *Annual international conference on the theory and applications of cryptographic techniques*. Springer, 305–326.

[13] Eamonn Keogh, Kaushik Chakrabarti, Michael Pazzani, and Sharad Mehrotra. 2001. Locally adaptive dimensionality reduction for indexing large time series databases. In *Proceedings of the 2001 ACM SIGMOD international conference on Management of data*. 151–162.

[14] Xiaoning Liu and Xun Yi. 2019. Privacy-preserving collaborative medical time series analysis based on dynamic time warping. In *European Symposium on Research in Computer Security*. Springer, 439–460.

[15] Torben Pryds Pedersen. 2001. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology—CRYPTO'91: Proceedings*. Springer, 129–140.

[16] C. P. Schnorr. 1990. Efficient Identification and Signatures for Smart Cards. In *Advances in Cryptology — CRYPTO' 89 Proceedings*, Gilles Brassard (Ed.). Springer New York, New York, NY, 239–252.

[17] Nader Shakibay Senobari, Zachary Zimmerman, Gareth Funning, Peter M. Shearer, Philip Brisk, and Eamonn & Keogh. 2019. Using the Matrix Profile to detect seismic events – from the lab experiment scale to local and global scales. In *SCEC Annual Meeting*.

[18] Rutuja Wankhedkar and Sanjay Kumar Jain. 2021. Motif discovery and anomaly detection in an ECG using matrix profile. In *Progress in Advanced Computing and Intelligent Engineering: Proceedings of ICACIE 2019, Volume 1*. Springer, 88–95.

[19] James J Yang and Anne Buu. 2024. Efficient matrix profile computation with Euclidean distance using Eigen transformation: Performance evaluation based on beat-to-beat interval (BBI) data. *Statistics in medicine* 43, 16 (2024), 3051–3061.

[20] Chin-Chia Michael Yeh, Audrey Der, Uday Singh Saini, Vivian Lai, Yan Zheng, Junpeng Wang, Xin Dai, Zhongfang Zhuang, Yujie Fan, Huiyuan Chen, Prince Osei Aboagye, Liang Wang, Wei Zhang, and Eamonn Keogh. 2024. Matrix Profile for Anomaly Detection on Multidimensional Time Series. In *2024 IEEE International Conference on Data Mining (ICDM)*. 911–916. doi:10.1109/ICDM59182.2024.00114

[21] Chin-Chia Michael Yeh, Yan Zhu, Liudmila Ulanova, Nurjahan Begum, Yifei Ding, Hoang Anh Dau, Diego Furtado Silva, Abdullah Mueen, and Eamonn Keogh. 2016. Matrix profile I: all pairs similarity joins for time series: a unifying view that includes motifs, discords and shapelets. In *2016 IEEE 16th international conference on data mining (ICDM)*. Ieee, 1317–1322.

[22] Haoying Zhang, Nicolas Anciaux, Benjamin Nguyen, Fabien Girard, Jose Maria de Fuentes, and Adrien Boiret. 2026. Privacy Attacks on Matrix Profiles via Reconstruction Techniques (to appear PETS 2026). In *Privacy Enhancing Technologies Symposium*.

[23] Li Zhang, Jiahao Ding, Yifeng Gao, and Jessica Lin. 2023. PMP: Privacy-aware matrix profile against sensitive pattern inference for time series. In *Proceedings of the 2023 SIAM International Conference on Data Mining (SDM)*. SIAM, 891–899.

[24] Puyang Zhao, James J Yang, and Anne Buu. 2025. Applied statistical methods for identifying features of heart rate that are associated with nicotine vaping. *The American journal of drug and alcohol abuse* 51, 2 (2025), 165–172.

[25] Yandong Zheng, Rongxing Lu, Yunguo Guan, Jun Shao, and Hui Zhu. 2021. Efficient and privacy-preserving similarity range query over encrypted time series data. *IEEE Transactions on Dependable and Secure Computing* 19, 4 (2021), 2501–2516.

[26] Haohan Zhu, Xianrui Meng, and George Kollios. 2014. Privacy preserving similarity evaluation of time series data.. In *EDBT*, Vol. 2014. 499–510.

[27] Yan Zhu, Chin-Chia Michael Yeh, Zachary Zimmerman, Kaveh Kamgar, and Eamonn Keogh. 2018. Matrix profile XI: SCRIMP++: time series motif discovery at interactive speeds. In *2018 IEEE international conference on data mining (ICDM)*. IEEE, 837–846.

A Alternative of the proof π

In the following paragraph, we propose a variation of our scheme that considers the computation of anomaly or similarity in a private time series relative to a public time series rather than considering the comparison of a time series with itself.

In the algorithm Prove parses pp as $(m, \mathrm{Eucl}, \epsilon, \hat{x})$ where $\hat{x}$ is a time series of length $\hat{n}$ s.t. $\hat{n}$ is not necessarily equal to n. For each $i \in [\![n]\!]$, picks $\tilde{k}_i \xleftarrow{\$} \mathbb{Z}_p^*$, computes $\tilde{c}_i = c_i^{x_i} h^{\tilde{k}_i}$ and generates the following proof:

$$\pi_{\mathrm{sq},i} \leftarrow \left\{ x_i, k_i, \tilde{k}_i \ : \ c_i = g^{x_i} h^{k_i} \wedge \tilde{c}_i = c_i^{x_i} h^{\tilde{k}_i} \right\}.$$

For each $i \in [\![n]\!]$, $j \in [\![\hat{n}]\!]$, computes $\tilde{c}_{i,j} = \frac{\tilde{c}_i g^{\hat{x}_j^2}}{(c_i)^{2\hat{x}_j}}$. Sets $\mathcal{I} = [\![n-m+1]\!]$ and $\mathcal{J} = [\![\hat{n}-m+1]\!]$, for each $i \in \mathcal{I}$, $j \in \mathcal{J}$, computes $d_{i,j} = \sum_{r=0}^{m-1} x_{i+r}^2 - 2x_{i+r} \cdot \hat{x}_{j+r} + x_{j+r}^2$, $w_{i,j} = \sum_{r=0}^{m-1} (\hat{x}_{i+r} - 2x_{j+r}) k_{i+r} + \tilde{k}_{i+r}$ and $D_{i,j} = \prod_{r=0}^{m-1} \tilde{c}_{i+r,j+r}$.

We can remark that,

$$
\begin{aligned}
\prod_{r=0}^{m-1} \tilde{c}_{i+r,j+r} &= \prod_{r=0}^{m-1} \frac{\tilde{c}_{i+r} g^{\hat{x}_{j+r}^2}}{(c_{i+r})^{2\hat{x}_{j+r}}} \\
&= \prod_{r=0}^{m-1} \frac{(g^{x_{i+r}} h^{k_{i+r}})^{x_{i+r}} h^{\tilde{k}_{i+r}} g^{\hat{x}_{j+r}^2}}{(g^{x_{i+r}} h^{k_{i+r}})^{2\hat{x}_{j+r}}} \\
&= \prod_{r=0}^{m-1} \frac{g^{x_{i+r}^2} h^{k_{i+r} \cdot x_{i+r} + \tilde{k}_{i+r}} g^{\hat{x}_{j+r}^2}}{g^{2x_{i+r} \cdot \hat{x}_{j+r}} h^{2k_{i+r} \cdot \hat{x}_{j+r}}} \\
&= \prod_{r=0}^{m-1} g^{x_{i+r}^2 - 2x_{i+r} \cdot \hat{x}_{j+r} + \hat{x}_{j+r}^2} \cdot \\
& h^{k_{i+r}(x_{i+r} - 2\hat{x}_{j+r}) + \tilde{k}_{i+r}} \\
&= g^{\sum_{r=0}^{m-1} (x_{i+r} - x_{j+r})^2} h^{\sum_{r=0}^{m-1} k_{i+r}(x_{i+r} - 2\hat{x}_{j+r}) + \tilde{k}_{i+r}} \\
&= g^{d_{i,j}} h^{w_{i,j}}.
\end{aligned}
$$

The rest of the scheme remains unchanged, excepts that the set $\mathcal{J}_i$ is replaced by the set $\mathcal{J}$.

B Asymptotic complexity

B.1 Complexity of NIP building blocks

In this paper, we use several NIPs as building blocks to construct all the NIP used in the $\mathsf{VASC}_{\mathsf{Ped}}$ scheme. Figure 5 describes the sigma protocol for the OR-proof, which allows one to prove knowledge of the discrete logarithm (in basis g_i) of one of several elements y_i in $\mathbb{G}$. Figure 4 describes the sigma protocol for proving the knowledge of a set of discrete logarithms (in basis g_i) of elements y_i in the group $\mathbb{G}$. Furthermore, Figure 3 shows the sigma protocol (that we named EQ-proof) allowing to prove the knowledge of the value in which two Pedersen commitment y_1 and y_2 can be opened (knowing the opening key) and that these values are the same. The table 7 presents the asymptotic complexity of the NIP building blocks.

Prover $(\alpha, \beta_1, \beta_2)$		**Verifier** (y_1, y_2, g_1, g_2, h)
$r, s_1, s_2 \xleftarrow{\$} \mathbb{Z}_p^*$ $R_1 = g_1^r, R_2 = g_2^r$ $S_1 = h^{s_1}, S_2 = h^{s_2}$	$\xrightarrow{R_1, R_2, S_1, S_2}$	$c \xleftarrow{\$} \mathbb{Z}_p^*$
$u = r + \alpha c$ $v_1 = s_1 + \beta_1 c$ $v_2 = s_2 + \beta_2 c$	$\xleftarrow{c}$	
	$\xrightarrow{u, v_1, v_2}$	If $g_1^u h^{v_1} = R_1 S_1 y_1^c$ and $g_2^u h^{v_2} = R_2 S_2 y_2^c$ then accept, else reject

Figure 3: EQ − Open-**proof for the relation** $y_1 = g_1^\alpha h^{\beta_1} \wedge y_2 = g_2^\alpha h^{\beta_2}$.

Simulator as input a statement (y, g), generates a response z and a challenge c, and returns $(g^u \cdot y^{-c}, c, z)$.

B.2 Complexity of $\mathsf{VASC}_{\mathsf{Ped}}$ scheme

In table 8, we summarize the asymptotic complexity of the scheme, we compare the complexity of the generation of the proof π in Table 8, and its verification in Table 9 varying on the parameter η. The complexity is evaluated in terms of the number of exponentiations in the group $\mathbb{G}$ required. A more detailed complexity of π_{cmp} is given for

Prover $(\omega_i)_{i \in [\![n]\!]}$		**Verifier** $(y_i, g_i)_{i \in [\![n]\!]}$
$\forall i \in [\![n]\!], r_i \xleftarrow{\$} \mathbb{Z}_p^*, R_i = g_i^{r_i}$	$\xrightarrow{(R_i)_{i \in [\![n]\!]}}$	$c \xleftarrow{\$} \mathbb{Z}_p^*$
$\forall i \in [\![n]\!], z_i = r_i + \omega_i \cdot c$	$\xleftarrow{c}$	
	$\xrightarrow{(z_i)_{i \in [\![n]\!]}}$	If $g_i^{z_i} = R_i \cdot y_i^c$ then accept, else reject

Figure 4: AND-proof for the relation $\bigwedge_{i=1}^{n} y_i = g_i^{\omega_i}$.

Prover ω		**Verifier** $(y_i, g_i)_{i \in [\![n]\!]}$
$r \xleftarrow{\$} \mathbb{Z}_p^*, R_{i^*} = g_{i^*}^r$ $\forall i \in [\![n]\!] \setminus \{i^*\}$, $(R_i, c_i, z_i) \leftarrow \mathsf{Simulator}(y_i, g_i)$	$\xrightarrow{(R_i)_{i \in [\![n]\!]}}$	$c \xleftarrow{\$} \mathbb{Z}_p^*$
$c_{i^*} = c \oplus \bigoplus_{i \in [\![n]\!] \setminus \{i^*\}} c_i$ $z_{i^*} = r + \omega_i \cdot c_{i^*}$	$\xleftarrow{c}$	
	$\xrightarrow{(c_i, u_i)_{i \in [\![n]\!]}}$	If $c = \bigoplus_{i \in [\![n]\!]} c_i$ and $\forall i \in [\![n]\!]$, $g^{z_i} = R_i y_i^{c_i}$ then accept, else reject

Figure 5: OR-proof for the relation $\bigvee_{i=1}^{n} y_i = g_i^\omega$ **(s.t.** $\exists i^* \in [\![n]\!]; y_{i^*} = g_{i^*}^\omega$**).**

NIP	Prover time	Verifier time
Fig.3(EQ-Open)	$O(1)$	$O(1)$
Fig.4 (AND-proof)	$O(n)$	$O(n)$
Fig.5 (OR-proof)	$O(n)$	$O(n)$

Table 7: Asymptotic complexity of the NIP **building blocks**

$\eta \in \{\mathsf{A}, \perp_{\mathsf{S}}\}$ (resp. $\{\mathsf{S}, \perp_{\mathsf{A}}\}$) in Table 10 (resp. 11). We notice that the complexity of the proof π_{cmp} depends on the binary decomposition of ϵ, then for this proof, we express the complexity in the worst case (when $u_1 = 1$). Our complexity is not determined solely by the number of values in the time series n, but also by the subsequence window m and the number of bit ℓ in the binary decomposition. Some proofs are executed for each pair $i \in \mathcal{I}$ and $j \in \mathcal{J}_i$ where the number of elements in $\mathcal{J}_i$ depends on i. This result in a complexity that depends on the sum of the cardinality of $\mathcal{J}_i$ which the following calculation give us a quadratic complexity in the number n.

$$
\begin{aligned}
\mathrm{card}(J_i) &= \mathrm{card}\left(\left[\!\left[1; i - \frac{m}{2} - 1 \right]\!\right] \right) \\
&= \begin{cases} i - \frac{m}{2} - 1 & \text{if } i > \frac{m}{2} + 1, \\ 0 & \text{else.} \end{cases}
\end{aligned}
$$

$$
\begin{aligned}
\sum_{i\in\mathcal{I}} \mathrm{card}(\mathcal{J}_i) &= \sum_{i=\frac{m}{2}+2}^{n-m+1} i - \frac{m}{2} - 1 \\
&= -\left(\frac{m}{2}+1\right)\cdot\left(n-m+1-\left(\frac{m}{2}+2\right)+1\right) \\
&+ \sum_{i=1}^{n-m+1} i - \sum_{i=1}^{m/2+1} i \\
&= -\left(\frac{m}{2}+1\right)\cdot\left(n-\frac{3m}{2}\right) \\
&+ \frac{(n-m+1)\cdot(n-m+2)}{2} \\
&- \frac{(m/2+1)\cdot(m/2+2)}{2} \\
&= -\frac{mn}{2} + \frac{3m^2}{4} - n + \frac{3m}{2} \\
&+ \frac{n^2}{2} - nm + \frac{3n}{2} + \frac{m^2}{2} - \frac{3m}{2} \\
&+ 1 - \frac{m^2}{8} - \frac{3m}{4} - 1 \\
&= \frac{n^2}{2} - \frac{3nm}{2} + \frac{n}{2} + \frac{9m^2}{8} - \frac{3m}{4}.
\end{aligned}
$$

Element	Prover time	
$(\widetilde{c}_{i,j})_{i,j\in[\![n]\!];j<i}$	$O(n\cdot\frac{n-1}{2})$	
π_{sq}	$O(n\cdot\frac{n-1}{2})$	
$(D_{i,j,u})_{i\in\mathcal{I},j\in\mathcal{J}_i,u\in[\![\ell]\!]}$	$O\left(\ell\cdot\sum_{i\in\mathcal{I}}\mathrm{card}(\mathcal{J}_i)\right)$	
$\pi_{d_{\mathsf{bin}}}$	$O\left(2\cdot\ell\cdot\sum_{i\in\mathcal{I}}\mathrm{card}(\mathcal{J}_i)\right)$	
η	A or $\perp_{\mathsf{S}}$	$\perp_{\mathsf{A}}$ or S
π_{cmp}	$O\left(4\cdot\ell\cdot\mathrm{card}(\mathcal{I})\right.$ $\left.+3\cdot\ell\cdot\sum_{i\in\mathcal{I}}\mathrm{card}(\mathcal{J}_i)\right)$	$O\left(\ell\cdot\sum_{i\in\mathcal{I}}\mathrm{card}(\mathcal{J}_i)\right)$
π	$O\left(n\cdot(n-1)\right.$ $+4\cdot\ell\cdot\mathrm{card}(\mathcal{I})$ $\left.+6\cdot\ell\cdot\sum_{i\in\mathcal{I}}\mathrm{card}(\mathcal{J}_i)\right)$	$O\left(n\cdot(n-1)\right.$ $\left.+4\cdot\ell\cdot\sum_{i\in\mathcal{I}}\mathrm{card}(\mathcal{J}_i)\right)$

Table 8: Asymptotic complexity the generation of π in the algorithm Prove **for our** $\mathsf{VASC}_{\mathsf{Ped}}$ **scheme.**

NIP	Verifier time	
π_{sq}	$O(n\cdot\frac{n-1}{2})$	
$\pi_{d_{\mathsf{bin}}}$	$O\left(2\cdot\ell\cdot\sum_{i\in\mathcal{I}}\mathrm{card}(\mathcal{J}_i)\right)$	
η	A or $\perp_{\mathsf{S}}$	$\perp_{\mathsf{A}}$ or S
π_{cmp}	$O\left(2\cdot\ell\cdot\mathrm{card}(\mathcal{I})\right.$ $\left.+3\cdot\ell\cdot\sum_{i\in\mathcal{I}}\mathrm{card}(\mathcal{J}_i)\right)$	$O\left(\ell\cdot\sum_{i\in\mathcal{I}}\mathrm{card}(\mathcal{J}_i)\right)$
π	$O\left(n\cdot\frac{n-1}{2}\right.$ $+2\cdot\ell\cdot\mathrm{card}(\mathcal{I})$ $\left.+5\cdot\ell\cdot\sum_{i\in\mathcal{I}}\mathrm{card}(\mathcal{J}_i)\right)$	$O\left(n\cdot\frac{n-1}{2}\right.$ $\left.+3\cdot\ell\cdot\sum_{i\in\mathcal{I}}\mathrm{card}(\mathcal{J}_i)\right)$

Table 9: Asymptotic complexity of the verification of the proof π for our $\mathsf{VASC}_{\mathsf{Ped}}$ **scheme.**

Element	Prover time	Verifier time
$(M_{i,u})_{i\in\mathcal{I},u\in[\![\ell]\!]}$	$O(\ell\cdot\mathrm{card}(\mathcal{I}))$	-
$\pi_{MPD_{\mathsf{bin}}}$	$O(2\cdot\ell\cdot\mathrm{card}(\mathcal{I}))$	$O(2\cdot\ell\cdot\mathrm{card}(\mathcal{I}))$
$\pi_{MPD_{\mathsf{exist}}}$	$O\left(\ell\cdot\sum_{i\in\mathcal{I}}\mathrm{card}(\mathcal{J}_i)\right)$	$O\left(\ell\cdot\sum_{i\in\mathcal{I}}\mathrm{card}(\mathcal{J}_i)\right)$
$\pi_{MPD_{\mathsf{min}}}$	$O\left(2\cdot\ell\cdot\sum_{i\in\mathcal{I}}\mathrm{card}(\mathcal{J}_i)\right)$	$O\left(2\cdot\ell\cdot\sum_{i\in\mathcal{I}}\mathrm{card}(\mathcal{J}_i)\right)$
π_{th}	max: $O(\ell\cdot\mathrm{card}(\mathcal{I}))$	max: $O(\ell\cdot\mathrm{card}(\mathcal{I}))$
π_{cmp}	max: $O\left(4\cdot\ell\cdot\mathrm{card}(\mathcal{I})\right.$ $\left.+3\cdot\ell\cdot\sum_{i\in\mathcal{I}}\mathrm{card}(\mathcal{J}_i)\right)$	max: $O\left(2\cdot\ell\cdot\mathrm{card}(\mathcal{I})\right.$ $\left.+3\cdot\ell\cdot\sum_{i\in\mathcal{I}}\mathrm{card}(\mathcal{J}_i)\right)$

Table 10: Asymptotic complexity of the proof π_{cmp} where $\eta \in \{\mathsf{A},\perp_{\mathsf{S}}\}$ where $\mathrm{card}(\mathcal{I}) = n-m+1$ and $\sum_{i\in\mathcal{I}}\mathrm{card}(\mathcal{J}_i) = \frac{n^2}{2} - \frac{3nm}{2} + \frac{n}{2} + \frac{9m^2}{8} - \frac{3m}{4}$.

NIP	Prover time	Verifier time
π_{cmp}	max: $O\left(\ell\cdot\sum_{i\in\mathcal{I}}\mathrm{card}(\mathcal{J}_i)\right)$	max: $O\left(\ell\cdot\sum_{i\in\mathcal{I}}\mathrm{card}(\mathcal{J}_i)\right)$

Table 11: Asymptotic complexity of the proof π_{cmp} where $\eta \in \{\perp_{\mathsf{A}},\mathsf{S}\}$ where $\sum_{i\in\mathcal{I}}\mathrm{card}(\mathcal{J}_i) = \frac{n^2}{2} - \frac{3nm}{2} + \frac{n}{2} + \frac{9m^2}{8} - \frac{3m}{4}$.

C Security Proofs

C.1 Security Proofs of our NIP buildings blocks

C.1.1 Proof that π_0 is correct, zero-knowledge and extractable. We recall that π_0 corresponds to the following proof:

$$\pi_0 \leftarrow \mathsf{NIP}\left\{(\omega_u)_{u\in[\![u_1;\ell]\!]} : \begin{array}{l} \bigwedge_{u=u_\alpha+1}^{\ell} y_u = h^{\omega_u} \\ \wedge\Bigg(y_{u_\alpha} = h^{\omega_{u_\alpha}} \\ \vee\Bigg(\bigwedge_{u=u_{\alpha-1}+1}^{u_\alpha-1} y_u = h^{\omega_u} \\ \wedge\Bigg(y_{u_{\alpha-1}} = h^{\omega_{i,j,u_{\alpha-1}}} \\ \vee(\ldots \\ \vee\Bigg(\bigwedge_{u=u_1+1}^{u_2-1} y_u = h^{\omega_u} \\ \wedge y_{u_1} = h^{\omega_{u_1}}\Bigg)\ldots\Bigg)\Bigg)\Bigg)\Bigg) \end{array}\right\}.$$

Through this paragraph, $\mathcal{R}_{0,0}$ represents the relation:

$$\bigwedge_{u=u_\alpha+1}^{\ell} y_u = h^{\omega_u} \wedge \Bigg(y_{u_\alpha} = h^{\omega_{u_\alpha}} \vee \Bigg(\bigwedge_{u=u_{\alpha-1}+1}^{u_\alpha-1} y_u = h^{\omega_u} \wedge \Bigg(y_{u_{\alpha-1}} = h^{\omega_{u_{\alpha-1}}} \vee \Bigg(\ldots \vee \Bigg(\bigwedge_{u=u_1+1}^{u_2-1} y_u = h^{\omega_u} \wedge y_{u_1} = h^{\omega_{u_1}}\Bigg)\ldots\Bigg)\Bigg)\Bigg)\Bigg).$$

and we denote by (R, c, z) the transcript for this relation. We denote by $\mathcal{R}_x$ (resp. $\mathcal{R}_{(x,y)}$) the relation: $y_x = h^{\omega_x}$ (resp. $\bigwedge_{u=x}^{y} y_u = h^{\omega_u}$).

(R_x, c_x, z_x) (resp. $(R_{(x,y)}, c_{(x,y)}, z_{(x,y)})$) denotes the transcript for the relation $\mathcal{R}_x$ (resp. $\mathcal{R}_{(x,y)}$). The challenges for the relation $\mathcal{R}_{0,0}$ verify the following equations:

$$\begin{aligned} ch &= c_{(u_\alpha+1,\ell)} \\ &= c_{u_\alpha} \oplus c_{(u_{\alpha-1}+1,u_\alpha-1)}. \end{aligned} \tag{11}$$

for all $k \in [\![2;\alpha-1]\!]$,

$$c_{(u_k+1,u_{k+1}-1)} = c_{u_k} \oplus c_{(u_{k-1}+1,u_k-1)}. \tag{12}$$

$$c_{(u_1+1,u_2-1)} = c_{u_1}. \tag{13}$$

π_0 is correct, zero-knowledge and extractable.

Correctness. Let $((y_u)_{u\in[\![u_1;\ell]\!]}, h)$ be a statement and $(\omega_u)_{u\in[\![u_1;\ell]\!]}$ be a witness $s.t.$ $(((y_u)_{u\in[\![u_1;\ell]\!]}, h), (\omega_u)_{u\in[\![u_1;\ell]\!]}) \in \mathcal{R}_{0,0}$. Let $(R, c, z) \leftarrow \mathsf{NIP}\{(\omega_u)_{u\in[\![u_1;\ell]\!]} : ((\omega_u)_{u\in[\![u_1;\ell]\!]}, (y_{u\in[\![u_1;\ell]\!]}, h)) \in \mathcal{R}_{0,0}\}$ be a transcript. Parses R as $((R_{u_k}, R_{(u_k+1,u_{k+1}-1)})_{k\in[\![1;\alpha-1]\!]}, R_{u_\alpha}, R_{(u_\alpha+1,\ell)})$, c as $((c_{u_k}, c_{(u_k+1,u_{k+1}-1)})_{k\in[\![1;\alpha-1]\!]}, c_{u_\alpha}, c_{(u_\alpha+1,\ell)})$ and z as $((z_{u_k}, z_{(u_k+1,u_{k+1}-1)})_{k\in[\![1;\alpha-1]\!]}, z_{u_\alpha}, z_{(u_\alpha+1,\ell)})$.

- For each $k \in [\![1;\alpha-1]\!]$, for each $u \in [\![u_k+1;u_{k+1}-1]\!]$, we have: $R_u = h^{z_u} \cdot y_u^{-c_{(u_k+1,u_{k+1}-1)}}$.
- For each $k \in [\![1;\alpha]\!]$, we have $R_{u_k} = h^{z_{u_k}} \cdot y_{u_k}{}^{-c_{u_k}}$.
- for each $u \in [\![u_\alpha+1;\ell]\!]$, we have: $R_u = h^{z_u} \cdot y_u^{-c_{(u_\alpha+1,\ell)}}$.

Moreover, the equations 11,12 and 13 hold.

Hence $\mathsf{NIPVerify}((y_u)_{u\in[\![u_1;\ell]\!]}, h), (R, c, z))$ returns 1.

Zero-Knowledge. Let $\mathsf{Simulator}_0$ be a simulator described as follows:

$\mathsf{Simulator}_0((y_u)_{u\in[\![u_1;\ell]\!]}, h)$:

- Simulates the responses for all the relations:
  - picks $z_{u_\alpha} \xleftarrow{\$} \mathbb{Z}_p^*$ and for each $u \in [\![u_\alpha+1;\ell]\!]$, picks $z_u \xleftarrow{\$} \mathbb{Z}_p^*$, and sets $z_{(u_\alpha+1,\ell)} = (z_u)_{u\in[\![u_\alpha+1;\ell]\!]}$.
  - for each $k \in [\![1;\alpha-1]\!]$, picks $z_{u_k} \xleftarrow{\$} \mathbb{Z}_p^*$ and for each $u \in [\![u_k+1;u_{k+1}-1]\!]$, picks $z_u \xleftarrow{\$} \mathbb{Z}_p^*$, and sets $z_{(u_k+1,u_{k+1}-1)} = (z_u)_{u\in[\![u_k+1;u_{k+1}-1]\!]}$.
- Simulates the challenges for all the relations according to the equations 11,12, and 13:
  - picks $c_{u_1} \xleftarrow{\$} \mathbb{Z}_p^*$ and sets $c_{(u_1+1,u_2-1)} = c_{u_1}$.
  - for each $k \in [\![2;\alpha-1]\!]$, picks $c_{u_k} \xleftarrow{\$} \mathbb{Z}_p^*$ and sets $c_{(u_k+1,u_{k+1}-1)} = c_{u_k} \oplus c_{(u_{k-1}+1,u_k-1)}$.
  - picks $c_{u_\alpha} \xleftarrow{\$} \mathbb{Z}_p^*$, and sets $ch = c_{(u_\alpha+1,\ell)} = c_{u_\alpha} \oplus c_{(u_{\alpha-1}+1,u_\alpha-1)}$.
- Simulates the commitments for all relations:
  - for each $k \in [\![1;\alpha-1]\!]$, computes $R_{u_k} = h^{z_{u_k}} \cdot y_{u_k}^{-c_{u_k}}$ and for each $u \in [\![u_k+1,u_{k+1}-1]\!]$, computes $h^{z_u} \cdot y_u^{-c_{(u_k+1,u_{k+1}-1)}}$, and sets $R_{(u_k+1,u_{k+1}-1)} = (R_u)_{u\in[\![u_k+1;u_{k+1}-1]\!]}$.
  - computes $R_{u_\alpha} = h^{z_{u_\alpha}} \cdot y_{u_\alpha}^{-c_{u_\alpha}}$, and for each $u \in [\![u_\alpha+1;\ell]\!]$, computes $R_u = h^{z_u} \cdot y_u^{-c_{(u_\alpha+1,\ell)}}$, and sets $R_{(u_\alpha+1,\ell)} = (R_u)_{u\in[\![u_\alpha+1;\ell]\!]}$.

Sets $R = ((R_{u_k}, R_{(u_k+1,u_{k+1}-1)})_{k\in[\![1;\alpha-1]\!]}, R_{u_\alpha}, R_{(u_\alpha+1,\ell)})$, $c = ((c_{u_k}, c_{(u_k+1,c_{k+1}-1)})_{k\in[\![1;\alpha-1]\!]}, c_{u_\alpha}, c_{(u_\alpha+1,\ell)})$ and $z = ((z_{u_k}, z_{(u_k+1,u_{k+1}-1)})_{k\in[\![1;\alpha-1]\!]}, z_{u_\alpha}, z_{(u_\alpha+1,\ell)})$, and returns (R, c, z).

Finally, $\mathsf{NIP}\{(\omega_u)_{u\in[\![\ell]\!]} : ((\omega_u)_{u\in[\![u_1;\ell]\!]}, ((y_u)_{u\in[\![u_1;\ell]\!]}, h)) \in \mathcal{R}_{0,0}\}$ and $\mathsf{Simulator}_0((y_u)_{u\in[\![u_1;\ell]\!]}, h)$ are identically distributed.

Extractability. Let (R, c, z) and (R, c', z') be two valid transcripts for the statement $((y_u)_{u\in[\![u_1;\ell]\!]}, h)$. Parses R as $((R_{u_k}, R_{(u_k+1,u_{k+1}-1)})_{k\in[\![1;\alpha-1]\!]}, R_{u_\alpha}, R_{(u_\alpha+1,\ell)})$, c as $((c_{u_k}, c_{(u_k+1,u_{k+1}-1)})_{k\in[\![1;\alpha-1]\!]}, c_{u_\alpha}, c_{(u_\alpha+1,\ell)})$, c' as$((c'_{u_k}, c'_{(u_k+1,c_{k+1}-1)})_{k\in[\![1;\alpha-1]\!]}, c'_{u_\alpha}, c'_{(u_\alpha+1,\ell)})$, z as $((z_{u_k}, z_{(u_k+1,u_{k+1}-1)})_{k\in[\![1;\alpha-1]\!]}, z_{u_\alpha}, z_{(u_\alpha+1,\ell)})$ and z' as $((z'_{u_k}, z'_{(u_k+1,u_{k+1}-1)})_{k\in[\![1;\alpha-1]\!]}, z'_{u_\alpha}, z'_{(u_\alpha+1,\ell)})$.

- from the verified relation $\mathcal{R}_{(u_{\alpha+1,\ell})}$, we have $c_{(u_\alpha+1,\ell)} \neq c'_{(u_\alpha+1,\ell)}$, then for each $u \in [\![u_\alpha+1;\ell]\!]$, $y_u^{c_{(u_\alpha+1,\ell)}-c'_{(u_\alpha+1,\ell)}} = h^{z_u-z'_u}$ implies $y_u = h^{\frac{z_u-z'_u}{c_{(u_\alpha+1,\ell)}-c'_{(u_\alpha+1,\ell)}}}$, then $\omega_u = \frac{z_u-z'_u}{c_{(u_\alpha+1,\ell)}-c'_{(u_\alpha+1,\ell)}}$.
- from the verified relations $\mathcal{R}_{(u_k+1;u_{k+1}-1)}$, for some $k \in [\![1;\alpha-1]\!]$, we have $c_{(u_k+1,u_{k+1}-1)} \neq c'_{(u_k+1,u_{k+1}-1)}$, then for each $u \in [\![u_k+1;u_{k+1}-1]\!]$, $y_u^{c_{(u_k+1,u_{k+1}-1)}-c'_{(u_k+1,u_{k+1}-1)}} = h^{z_u-z'_u}$ implies $y_u = h^{\frac{z_u-z'_u}{c_{(u_k+1,u_{k+1}-1)}-c'_{(u_k+1,u_{k+1}-1)}}}$, then $\omega_u = \frac{z_u-z'_u}{c_{(u_k+1,u_{k+1}-1)}-c'_{(u_k+1,u_{k+1}-1)}}$.

- from the verified relations $\mathcal{R}_k$, for some $k \in [\![1;\alpha]\!]$, we have $c_{u_k} \neq c'_{u_k}$, then $y_{u_k}^{c_{u_k}-c'_{u_k}} = h^{z_{u_k}-z'_{u_k}}$ implies $y_{u_k} = h^{\frac{z_{u_k}-z'_{u_k}}{c_{u_k}-c'_{u_k}}}$, then $\omega_{u_k} = \frac{z_{u_k}-z'_{u_k}}{c_{u_k}-c'_{u_k}}$.

C.1.2 Proof that π_1 is correct, zero-knowledge and extractable. We recall that the proof π_1 corresponds to the following proof:

$$\pi_1 \leftarrow \mathsf{NIP}\left\{(\omega_u)_{u\in[\![u_1;\ell]\!]} : \begin{array}{l} \bigvee_{u=u_\alpha+1}^{\ell} y_u = h^{\omega_u} \\ \vee\Bigg(y_{u_\alpha} = h^{\omega_{u_\alpha}} \\ \wedge\Bigg(\bigvee_{u=u_{\alpha-1}+1}^{u_\alpha-1} y_u = h^{\omega_u} \\ \vee\Bigg(y_{u_{\alpha-1}} = h^{\omega_{i,j,u_{\alpha-1}}} \\ \wedge(\ldots \\ \wedge\Bigg(\bigvee_{u=u_1+1}^{u_2-1} y_u = h^{\omega_u} \\ \vee y_{u_1} = h^{\omega_{u_1}}\Bigg)\ldots\Bigg)\Bigg)\Bigg)\Bigg) \end{array}\right\}.$$

Through this paragraph, $\mathcal{R}_{0,1}$ represents the relation:

$$\bigvee_{u=u_\alpha+1}^{\ell} y_u = h^{\omega_u} \vee \Bigg(y_{u_\alpha} = h^{\omega_{u_\alpha}} \wedge \Bigg(\bigvee_{u=u_{\alpha-1}+1}^{u_\alpha-1} y_u = h^{\omega_u} \vee \Bigg(y_{u_{\alpha-1}} = h^{\omega_{u_{\alpha-1}}} \wedge \Bigg(\ldots \wedge \Bigg(\bigvee_{u=u_1+1}^{u_2-1} y_u = h^{\omega_u} \vee y_{u_1} = h^{\omega_{u_1}}\Bigg)\ldots\Bigg)\Bigg)\Bigg)\Bigg).$$

and we denote by (R, c, z) the transcript of the proof for this relation. We denote by $\mathcal{R}_x$ (resp. $\mathcal{R}_{(x,y)}$) the relation $y_x = h^{\omega_x}$ (resp. $\bigvee_{u=x}^{y} y_u = h^{\omega_u}$).

(R_x, c_x, z_x) (resp. $(R_{(x,y)}, c_{(x,y)}, z_{(x,y)})$) denotes the transcript of the proof for the relation $\mathcal{R}_x$ (resp. (R_x, c_x, z_x)). The challenges for the relation $\mathcal{R}_{0,1}$ verify the following equations:

$$ch = c_{(u_\alpha+1,\ell)} \oplus c_{u_\alpha}. \tag{14}$$

for all $k \in [\![2;\alpha]\!]$,

$$c_{u_k} = c_{(u_{k-1}+1,u_k-1)} \oplus c_{u_{k-1}}. \tag{15}$$

$$c_{(u_{k-1}+1,u_k-1)} = \bigoplus_{u=u_{k-1}+1}^{u_k-1} c_u. \tag{16}$$

π_1 is correct, zero-knowledge and extractable.

Correctness. Let $((y_u)_{u\in[\![u_1;\ell]\!]}, h)$ be a statement and $(\omega_u)_{u\in[\![u_1;\ell]\!]}$ be witnesses $s.t.$ $(((y_u)_{u\in[\![u_1;\ell]\!]}, h), (\omega_u)_{u\in[\![u_1;\ell]\!]}) \in \mathcal{R}_{0,1}$.

Let $(R, c, z) \leftarrow \mathsf{NIP}\{(\omega_u)_{u\in[\![u_1;\ell]\!]} : ((\omega_u)_{u\in[\![u_1;\ell]\!]}, ((y_u)_{u\in[\![u_1;\ell]\!]}, h)) \in \mathcal{R}_{0,1}\}$ be a transcript. Parses R as $(R_{u_1}, (R_{(u_{k-1}+1,u_k-1)}, R_{u_k})_{k\in[\![2;\alpha]\!]}, R_{(u_\alpha+1,\ell)})$, c as $(c_{u_1}, (c_{(u_{k-1}+1,u_k-1)}, c_{u_k})_{k\in[\![2;\alpha]\!]}, c_{(u_\alpha+1,\ell)}, ch)$ and z as $(z_{u_1}, (z_{(u_{k-1}+1,u_k-1)}, z_{u_k})_{k\in[\![2;\alpha]\!]}, z_{(u_\alpha+1,\ell)})$.

- For each $k \in [\![2;\alpha]\!]$, for each $u \in [\![u_{k-1}+1; u_k-1]\!]$, we have: $R_u = h^{z_u} \cdot y_u^{-c_u}$.
- For each $k \in [\![1;\alpha]\!]$, we have: $R_{u_k} = h^{z_{u_k}} \cdot y_{u_k}{}^{-c_{u_k}}$.
- For each $u \in [\![u_\alpha+1;\ell]\!]$, we have: $R_u = h^{z_u} \cdot y_u^{-c_u}$.

Moreover, the equations 14, 15 and 16 hold.

Hence, $\mathsf{NIPVerify}(((y_u)_{u\in[\![u_1;\ell]\!]}, h), (R, c, z))$ returns 1.

Zero-Knowledge. Let $\mathsf{Simulator}_1$ be a simulator described as follows:

$\mathsf{Simulator}_1((y_u)_{u\in[\![u_1;\ell]\!]}, h)$:

- Simulates the responses for all the relations:
  - for each $u \in [\![u_\alpha+1;\ell]\!]$, picks $z_u \xleftarrow{\$} \mathbb{Z}_p^*$, and sets $z_{(u_\alpha+1,\ell)} = (z_u)_{u\in[\![u_\alpha+1;\ell]\!]}$.
  - for each $k \in [\![2;\alpha]\!]$, picks $z_{u_k} \xleftarrow{\$} \mathbb{Z}_p^*$, and for each $u \in [\![u_{k-1}+1; u_k-1]\!]$, picks $z_u \xleftarrow{\$} \mathbb{Z}_p^*$, and sets $z_{(u_{k-1}+1,u_k-1)} = (z_u)_{u\in[\![u_{k-1}+1;u_k-1]\!]}$.
  - picks $z_{u_1} \xleftarrow{\$} \mathbb{Z}_p^*$.
- Simulates the challenges for all the relations according to the equations 14, 15 and 16:
  - picks $c_{u_1} \xleftarrow{\$} \mathbb{Z}_p^*$.
  - for each $k \in [\![2;\alpha]\!]$, for each $u \in [\![u_{k-1}+1; u_k-1]\!]$, picks $c_u \xleftarrow{\$} \mathbb{Z}_p^*$. For each $k \in [\![2;\alpha]\!]$, sets $c_{(u_{k-1}+1,u_k-1)} = \bigoplus_{u=u_{k-1}+1}^{u_k-1} c_u$ and sets $c_{u_k} = c_{(u_{k-1}+1,u_k-1)} \oplus c_{u_{k-1}}$.
  - for each $u \in [\![u_\alpha+1;\ell]\!]$, picks $c_u \xleftarrow{\$} \mathbb{Z}_p^*$, sets $ch = c_{(u_\alpha+1,\ell)} = \bigoplus_{u=u_\alpha+1}^{\ell} c_u$.
- Simulates the commitments for all relations:
  - for each $u \in [\![u_\alpha+1;\ell]\!]$ computes $R_u = h^{z_u} \cdot y_u^{-c_u}$, and sets $R_{(u_\alpha+1,\ell)} = (R_u)_{u\in[\![u_\alpha+1;\ell]\!]}$.
  - for each $k \in [\![2;\alpha]\!]$, computes $R_{u_k} = h^{z_{u_k}} \cdot y_{u_k}^{-c_{u_k}}$ and for each $u \in [\![u_{k-1}+1; u_k-1]\!]$, sets $R_u = h^{z_u} \cdot y_u^{-c_u}$, and computes $R_{(u_{k-1}+1,u_k-1)} = (R_u)_{u\in[\![u_{k-1}+1;u_k-1]\!]}$.
  - computes $R_{u_1} = h^{z_{u_1}} \cdot y_{u_1}^{-c_{u_1}}$.

Sets $R = (R_{u_1}, (R_{(u_{k-1}+1,u_k-1)}, R_{u_k})_{k\in[\![2;\alpha]\!]}, R_{(u_\alpha+1,\ell)})$, $c = (c_{u_1}, (c_{(u_{k-1}+1,u_k-1)}, c_{u_k})_{k\in[\![2,\alpha]\!]}, c_{(u_\alpha+1,\ell)}, ch)$ and $z = (z_{u_1}, (z_{(u_{k-1}+1,u_k-1)}, z_{u_k})_{k\in[\![2;\alpha]\!]}, z_{(u_\alpha+1,\ell)})$, and returns (R, c, z).

Finally, $\mathsf{NIP}\{(\omega_u)_{u\in[\![u_1;\ell]\!]} : ((\omega_u)_{u\in[\![u_1;\ell]\!]}, ((y_u)_{u\in[\![u_1;\ell]\!]}, h) \in \mathcal{R}_{0,1}\}$ and $\mathsf{Simulator}_1((y_u)_{u\in[\![u_1;\ell]\!]}, h)\}$ are identically distributed.

Extractability. Let (R, c, z) and (R, c', z') be two valid transcripts for the statement $((y_u)_{u\in[\![u_1;\ell]\!]}, h)$, $s.t.$ $R = (R_{u_1}, (R_{(u_{k-1}+1,u_k-1)}, R_{u_k})_{k\in[\![2;\alpha]\!]}, R_{(u_\alpha+1,\ell)})$, $c = (c_{u_1}, (c_{(u_{k-1}+1,u_k-1)}, c_{u_k})_{k\in[\![2;\alpha]\!]}, c_{(u_\alpha+1,\ell)}, ch)$, $c' = (c'_{u_1}, (c'_{(u_{k-1}+1,u_k-1)}, c'_{u_k})_{k\in[\![2;\alpha]\!]}, c'_{(u_\alpha+1,\ell)}, ch')$, $z = (z_{u_1}, (z_{(u_{k-1}+1,u_k-1)}, z_{u_k})_{k\in[\![2;\alpha]\!]}, z_{(u_\alpha+1,\ell)})$ and $z' = (z'_{u_1}, (z'_{(u_{k-1}+1,u_k-1)}, z'_{u_k})_{k\in[\![2;\alpha]\!]}, z'_{(u_\alpha+1,\ell)})$.

- from the verified relations $\mathcal{R}_{(u_{k-1}+1,u_k-1)}$ for some $k \in [\![2;\alpha]\!]$, we have: $c_{(u_{k-1}+1,u_k-1)} \neq c'_{(u_{k-1}+1,u_k-1)}$, then there exists a verified relation $\mathcal{R}_u$ $s.t.$ $u \in [\![u_{k-1}+1, u_k-1]\!]$ and $c_u \neq c'_u$, we have: $y_u^{c_u-c'_u} = h^{z_u-z'_u}$ implies $y_u = h^{\frac{z_u-z'_u}{c_u-c'_u}}$, then $\omega_u = \frac{z_u-z'_u}{c_u-c'_u}$.

- from the verified relations $\mathcal{R}_k$ for some $k \in [\![1;\alpha]\!]$, we have: $c_{u_k} \neq c'_{u_k}$, then $y_{u_k}^{c_{u_k}-c'_{u_k}} = h^{z_{u_k}-z'_{u_k}}$ implies $y_{u_k} = h^{\frac{z_{u_k}-z'_{u_k}}{c_{u_k}-c'_{u_k}}}$, then $\omega_{u_k} = \frac{z_{u_k}-z'_{u_k}}{c_{u_k}-c'_{u_k}}$.
- from the verified relation $\mathcal{R}_{(u_\alpha+1,\ell)}$ we have: $c_{(u_\alpha+1,\ell)} \neq c'_{(u_\alpha+1,\ell)}$, then there exists a verified relation $\mathcal{R}_u$ $s.t.$ $u \in [\![u_\alpha+1;\ell]\!]$ and $c_u \neq c'_u$, we have: $y_u^{c_u-c'_u} = h^{z_u-z'_u}$ implies $y_u = h^{\frac{z_u-z'_u}{c_u-c'_u}}$, then $\omega_u = \frac{z_u-z'_u}{c_u-c'_u}$.

C.1.3 Proof that π_2 is correct, zero-knowledge and extractable. The proof π_2 consists of the following proof:

$$\pi_2 \leftarrow \mathsf{NIP}\left\{(\gamma_u-\gamma_{u'})_{u\in[\![\ell]\!]} : \begin{array}{l} \frac{gD_\ell}{D'_\ell} = h^{\gamma_\ell-\gamma'_\ell} \\ \vee\left(\frac{D_\ell}{D'_\ell} = h^{\gamma_\ell-\gamma'_\ell}\right. \\ \wedge\left(\frac{gD_{\ell-1}}{D'_{\ell-1}} = h^{\gamma_{\ell-1}-\gamma'_{\ell-1}}\right. \\ \vee\left(\frac{D_{\ell-1}}{D'_{\ell-1}} = h^{\gamma_{\ell-1}-\gamma'_{\ell-1}}\right. \\ \wedge(\ldots \\ \wedge\left(\frac{gD_1}{D'_1} = h^{\gamma_1-\gamma'_1}\right. \\ \left.\left.\left.\vee\frac{D_1}{D'_1} = h^{\gamma_1-\gamma'_1}\right)\right)\ldots\right)\right)\right) \end{array}\right\}.$$

For each $u \in [\![\ell]\!]$, we allocate the value $y_{2u} = \frac{gD_u}{D'_u}$, $y_{2u-1} = \frac{D_u}{D'_u}$ and $\omega_{2u} = \omega_{2u-1} = \gamma_u - \gamma'_u$. We denote by $\mathcal{R}_{0,2}$ the relation:

$$\begin{array}{l} y_{2\ell} = h^{\omega_{2\ell}} \vee (y_{2\ell-1} = h^{\omega_{2\ell-1}} \\ \wedge (y_{2\ell-2} = h^{\omega_{2\ell-2}} \vee (y_{2\ell-3} = h^{\omega_{2\ell-3}} \\ \quad \wedge (\ldots \wedge (y_2 = h^{\omega_2} \vee y_1 = h^{\omega_1}) \ldots)))) . \end{array}$$

We denote by (R, c, z) the transcript for the relation $\mathcal{R}_{0,2}$. We denote by $\mathcal{R}_x$ the relation $y_x = h^{\omega_x}$, and (R_x, c_x, z_x) its corresponding transcript. This proof is correct, extractable and zero-knowledge. The challenges for the relation $\mathcal{R}_{0,2}$ verify the following equations:

$$ch = c_{2\ell} \oplus c_{2\ell-1}. \tag{17}$$

for all $u \in [\![1;\ell-1]\!]$,

$$c_{2u+1} = c_{2u} \oplus c_{2u-1}. \tag{18}$$

Correctness. Let $((y_{2u})_{u\in[\![\ell]\!]}, h)$ be a statement and $(\omega_{2u})_{u\in[\![\ell]\!]}$ be the witnesses $s.t.$ $(((y_{2u})_{u\in[\![\ell]\!]}, h), (\omega_{2u})_{u\in[\![\ell]\!]}) \in \mathcal{R}_{0,2}$. Let $(R, c, z) \leftarrow \mathsf{NIP}\{(\omega_{2u})_{u\in[\![\ell]\!]} : (((y_{2u})_{u\in[\![\ell]\!]}), (\omega_{2u})_{u\in[\![\ell]\!]}) \in \mathcal{R}_{0,2}\}$ be a transcript. Parses R as $(R_{2u-1}, R_{2u})_{u\in[\![\ell]\!]}$, c as $((c_{2u}, c_{2u-1})_{u\in[\![\ell]\!]}, ch)$ and z as $(z_{2u}, z_{2u-1})_{u\in[\![\ell]\!]}$.

We have $R_{2u} = h^{z_{2u}} \cdot y_{2u}^{-c_{2u}}$ (resp. $R_{2u-1} = h^{z_{2u-1}} \cdot y_{2u-1}^{-c_{2u-1}}$). Then $\mathsf{NIPVerify}(((y_{2u})_{u\in[\![\ell]\!]}, h), (R, c, z))$ returns 1.

Zero-knowledge. We denote $\mathsf{Simulator}_2$ the simulator for the relation $\mathcal{R}_{0,2}$ defined as follows:

$\underline{\mathsf{Simulator}_2((y_{2u-1}, y_{2u})_{u\in[\![\ell]\!]}, h) :}$

- simulates the responses for all the relations $\mathcal{R}_{2u}$ and $\mathcal{R}_{2u-1}$: for each $u \in [\![\ell]\!]$, picks $z_{2u} \xleftarrow{\$} \mathbb{Z}_p^*$, $z_{2u-1} \xleftarrow{\$} \mathbb{Z}_p^*$.
- simulates the challenges according to the equations 17 and 18 as follows: picks $c_1 \xleftarrow{\$} \mathbb{Z}_p^*$ and $c_2 \xleftarrow{\$} \mathbb{Z}_p^*$, for each $u \in [\![\ell-1]\!]$, sets $c_{2u+1} = c_{2u} \oplus c_{2u-1}$. Picks $c_{2\ell} \xleftarrow{\$} \mathbb{Z}_p^*$ and sets $ch = c_{2\ell} \oplus c_{2\ell-1}$.
- simulates the commitments for all the relations as follows: for each $u \in [\![\ell]\!]$, sets $R_{2u} = h^{z_{2u}} \cdot y_{2u}^{-c_{2u}}$ and $R_{2u-1} = h^{z_{2u-1}} \cdot y_{2u-1}^{-c_{2u-1}}$ It sets $R = (R_{2u-1}, R_{2u})_{u\in[\![\ell]\!]}$, $c = (c_{2u}, c_{2u-1})_{u\in[\![\ell]\!]}, ch)$, $z = (z_{2u}, z_{2u-1})_{u\in[\![\ell]\!]}$ and returns (R, c, z).

Extractability. Let (R, c, z) and (R', c', z') two valid transcript for the statement $((y_{2u-1}, y_{2u})_{u\in[\![\ell]\!]}, h)$. Parses R as $(R_{2u-1}, R_{2u})_{u\in[\![\ell]\!]}$, c as $(c_{2u}, c_{2u-1})_{u\in[\![\ell]\!]}, ch)$ and z as $(z_{2u}, z_{2u-1})_{u\in[\![\ell]\!]}$. From the verified relations R_{2u} (resp. R_{2u-1}) for some $u \in [\![\ell]\!]$, we have $c_{2u} \neq c'_{2u}$ (resp. $c_{2u-1} \neq c'_{2u-1}$), then we can extract the witnes as follows: $\omega_{2u} = \frac{z_{2u}-z'_{2u}}{c_{2u}-c'_{2u}}$ (resp. $\omega_{2u-1} = \frac{z_{2u-1}-z'_{2u-1}}{c_{2u-1}-c'_{2u-1}}$).

C.2 Security Proofs of our NIP

C.2.1 Proof that EQ − Open*-proof (Fig 3 in Appendix B.1) is correct, zero-knowledge and extractable.* Though this paragraph, we denote by $\mathcal{R}_{\mathrm{EQ-Open}}$ the relation $y_1 = g_1^\alpha h^{\beta_1} \wedge y_2 = g_2^\alpha h^{\beta_2}$ and we denote by (R, c, z) the transcript for this relation.

Correctness. Let (y_1, y_2, g_1, g_2, h) be a statement and $(\alpha, \beta_1, \beta_2)$ be a witness $s.t.$ $((y_1, y_2, g_1, g_2, h), (\alpha, \beta_1, \beta_2)) \in \mathcal{R}$. Let $(R, c, z) \leftarrow \mathsf{NIP}\{(\alpha, \beta_1, \beta_2) : y_1 = g_1^\alpha h^{\beta_1} \wedge y_2 = g_2^\alpha h^{\beta_2}\}$ be a transcript. Parses R as (R_1, R_2, S_1, S_2) and z as (u, v_1, v_2). We denote by r, s_1 and s_2 the random picked by the prover during the commitment phase, we have: $g_1^u h^{v_1} = g^{r+\alpha\cdot c} h^{s_1+\beta_1\cdot c} = R_1 g_1^{\alpha\cdot c} S_1 h^{\beta_1\cdot c} = R_1 S_1 (g_1^\alpha h^{\beta_1})^c = R_1 S_1 y_1^c$, and $g_2^u h^{v_2} = g^{r+\alpha\cdot c} h^{s_2+\beta_2\cdot c} = R_2 g_2^{\alpha\cdot c} S_2 h^{\beta_2\cdot c} = R_2 S_2 (g_2^\alpha h^{\beta_2})^c = R_2 S_2 y_2^c$. Finally, we obtain that: $\mathsf{NIPVerify}((y_1, y_2, g_1, g_2, h), (R, c, z))$ returns 1.

Zero-Knowledge. Let $\mathsf{Simulator}_{\mathrm{EQ-Open}}$ be a simulator for the relation $\mathcal{R}$ defined as follows:

$\underline{\mathsf{Simulator}\mathrm{EQ-Open}(y_1, y_2, g_1, g_2, h) :}$ picks $c \xleftarrow{\$} \mathbb{Z}_p^*$, $u \xleftarrow{\$} \mathbb{Z}_p^*$, $v_1 \xleftarrow{\$} \mathbb{Z}_p^*$, $v_2 \xleftarrow{\$} \mathbb{Z}_p^*$, $S_1 \xleftarrow{\$} \mathbb{G}$ and $S_2 \xleftarrow{\$} \mathbb{G}$. Computes $R_1 = \frac{g_1^u h^{v_1}}{S_1 y_1^c}$ and $R_2 = \frac{g_2^u h^{v_2}}{S_2 y_2^c}$, and returns $((R_1, R_2, S_1, S_2), c, (u, v_1, v_2))$.

Finally, $\mathsf{NIP}\{(\alpha, \beta_1, \beta_2) : y_1 = g_1^\alpha h^{\beta_1} \wedge y_2 = g_2^\alpha h^{\beta_2}\}$ and $\mathsf{Simulator}_{\mathrm{EQ-Open}}(y_1, y_2, g_1, g_2, h)$ are identically distributed.

Extractability. Let (R, c, z) and (R, c', z') be two valid transcripts for the statement (y_1, y_2, g_1, g_2, h). Parses R as (R_1, R_2, S_1, S_2), z as (u, v_1, v_2) and z' as (u', v'_1, v'_2). Since $c \neq c'$, we can extract the witnesses as follows:

- $y_1^{c-c'} = g_1^{u-u'} h^{v_1-v'_1}$ implies that $y_1 = g_1^{\frac{u-u'}{c-c'}} h^{\frac{v_1-v'_1}{c-c'}}$.
- $y_2^{c-c'} = g_2^{u-u'} h^{v_2-v'_2}$ implies that $y_2 = g_2^{\frac{u-u'}{c-c'}} h^{\frac{v_2-v'_2}{c-c'}}$.

Thus $\alpha = \frac{u-u'}{c-c'}$, $\beta_1 = \frac{v_1-v'_1}{c-c'}$ and $\beta_2 = \frac{v_2-v'_2}{c-c'}$.

C.2.2 Proof that $\pi_{MPD_{\mathrm{exist}}}$ is correct, zero-knowledge and extractable. This proof is a combination of OR-proof and

AND-proof which are correct, zero-knowledge and extractable, then $\pi_{MPD_{\text{exist}}}$ is correct, zero-knowledge and extractable and we denote by $\text{Simulator}_{MPD_{\text{exist}}}$ the simulator of $\pi_{MPD_{\text{exist}}}$.

C.2.3 Proof that $\pi_{MPD_{\min}}$ is correct, zero-knowledge and extractable. The proof $\pi_{MPD_{\min}}$ consists of executing the proof π_2 for each $i \in \mathcal{I}$, $j \in \mathcal{J}_i$ by allocating the value $M_{i,u}$ to D_u and $D_{i,j,u}$ to D'_u. The proof π_2 have been shown to be correct, extractable and zero-knowledge in C.1.3, then the proofs $(\pi_{MPD_{\min},i,j,u})_{i\in\mathcal{I},j\in\mathcal{J}_i,u\in[\![\ell]\!]}$ are correct, extractable and zero-knowledge.

C.2.4 Proof that π_{th} is correct, zero-knowledge and extractable. **Case** $\eta = \mathsf{A}$

By assigning for each $i \in \mathcal{I}$ and each $u \in [\![\ell]\!]$, the value $y_{i,u} = \frac{M_{i,u}}{g}$ and $\omega_{i,u} = z_{i,u}$. The proof

$$\pi_{\text{th}} \leftarrow \mathsf{NIP}\left\{(z_{i,u})_{i\in\mathcal{I},u\in[\![\ell]\!]} : \bigvee_{i\in\mathcal{I}} \mathsf{Ope}((z_{i,u})_{i\in\mathcal{I},u\in[\![\ell]\!]}, (M_{i,u})_{i\in\mathcal{I},u\in[\![\ell]\!]}) > \epsilon\right\}$$

consists of the following proof:

$$\mathsf{NIP}\left\{(\omega_{i,u})_{\substack{i\in\mathcal{I},\\ u\in[\![u_1;\ell]\!]}} : \bigvee_{i\in\mathcal{I}} \left(\begin{array}{l} \bigwedge_{u=u_\alpha+1}^{\ell} y_{i,u} = h^{\omega_{i,u}} \\ \wedge\left(y_{i,u_\alpha} = h^{\omega_{i,u_\alpha}}\right. \\ \vee\left(\bigwedge_{u=u_{\alpha-1}+1}^{u_\alpha-1} y_{i,u} = h^{\omega_{i,u}}\right. \\ \wedge\left(y_{i,u_{\alpha-1}} = h^{\omega_{i,u_{\alpha-1}}}\right. \\ \vee(\ldots \\ \vee\left(\bigwedge_{u=u_1+1}^{u_2-1} y_{i,u} = h^{\omega_{i,u}}\right. \\ \left.\left.\left.\left.\wedge y_{i,u_1} = h^{\omega_{i,u_1}}\right)\ldots\right)\right)\right)\right) \end{array}\right)\right\},$$

where $(\epsilon_u)_{u\in[\![\ell]\!]}$ represents the binary decomposition of ϵ and $(u_k)_{k\in[\![\alpha]\!]}$ are the indices s.t. $\epsilon_{u_k} = 0$.

Through this paragraph, we denote by $\mathcal{R}_{\text{th},0}$ the relation $\bigvee_{i\in\mathcal{I}} \mathcal{R}_i$, where $\mathcal{R}_i$ represents the relation $\mathcal{R}_{0,0}$ defined in C.1.1. By composition with the proof π_0 and an OR-proof, this proof is correct and extractable. Moreover, this proof is zero-knowledge, and we have the following simulator:

$\text{Simulator}_{\text{th},0}\left(\left(y_{i,u}\right)_{i\in\mathcal{I},u\in[\![u_1;\ell]\!]}, h\right)$: for each $i \in \mathcal{I}$, runs $(R_i, c_i, z_i) \leftarrow \text{Simulator}_0\left(\left(y_{i,u}\right)_{u\in[\![u_1;\ell]\!]}, h\right)$, parses c_i as $(c_{i,u_1}, c_{i,(u_1+1,u_2-1)}, (c_{i,u_k}, c_{i,(u_k+1,u_{k+1}-1)})_{k\in[\![2;\alpha-1]\!]}, c_{i,u_\alpha}, c_{i,(u_\alpha+1,\ell)})$, and sets $ch_i = c_{i,(u_\alpha+1,\ell)}$.

Computes $ch = \bigoplus_{i\in\mathcal{I}} ch_i$.

Sets $R = (R_i)_{i\in\mathcal{I}}$, $c = ((c_i)_{i\in\mathcal{I}}, ch)$ and $z = (z_i)_{i\in\mathcal{I}}$, and returns (R, c, z).

Case $\eta = \perp_{\mathsf{S}}$

We express the proof

$$\pi_{\text{th}} \leftarrow \mathsf{NIP}\left\{(z_{i,u})_{i\in\mathcal{I},u\in[\![\ell]\!]} : \bigwedge_{i\in\mathcal{I}} \mathsf{Ope}((z_{i,u})_{i\in\mathcal{I},u\in[\![\ell]\!]}, (M_{i,u})_{i\in\mathcal{I},u\in[\![\ell]\!]}) \geq \epsilon\right\},$$

as $(\pi_{\text{th},i})_{i\in\mathcal{I}}$.

By allocating for each $u \in [\![\ell]\!]$, $y_{i,u} = \frac{M_{i,u}}{g}$ and $\omega_{i,u} = z_{i,u}$, $\pi_{\text{th},i}$ is defined as follows:

$$\mathsf{NIP}\left\{(\omega_{i,u})_{u\in[\![\ell]\!]} : \left(\begin{array}{l} \bigvee_{u=u_\alpha+1}^{\ell} y_{i,u} = h^{\omega_{i,u}} \\ \vee\left(y_{i,u_\alpha} = h^{\omega_{u_{i,\alpha}}}\right. \\ \wedge\left(\bigvee_{u=u_{\alpha-1}+1}^{u_\alpha-1} y_{i,u} = h^{\omega_{i,u}}\right. \\ \vee\left(y_{i,u_{\alpha-1}} = h^{\omega_{i,u_{\alpha-1}}}\right. \\ \wedge(\ldots \\ \wedge\left(\bigvee_{u=u_1+1}^{u_2-1} y_{i,u} = h^{\omega_{i,u}}\right. \\ \left.\left.\left.\left.\vee y_{i,u_1} = h^{\omega_{i,u_1}}\right)\ldots\right)\right)\right)\right) \end{array}\right)\right\}.$$

Each proof $\pi_{\text{th},i}$ are the same as the proof 6.2.4 which is correct, zero-knowledge and extractable, thus the proofs $(\pi_{\text{th},i})_{i\in\mathcal{I}}$ are correct, zero-knowledge and we can extract the witnesses as in C.1.2. We define the simulator $\text{Simulator}_{\text{th},1}$ as follows:

$\text{Simulator}_{\text{th},1}\left(\left(y_{i,u}\right)_{i\in\mathcal{I},u\in[\![u_1;\ell]\!]}, h\right)$: For each $i \in \mathcal{I}$, runs $(R_i, c_i, z_i) \leftarrow \text{Simulator}_1\left(\left(y_{i,u}\right)_{u\in[\![u_1;\ell]\!]}, h\right)$, and returns $((R_i)_{i\in\mathcal{I}}, (c_i)_{i\in\mathcal{I}}, (z_i)_{i\in\mathcal{I}})$.

C.2.5 Proof that π_{cmp} is correct, zero-knowledge and extractable. **Case** $\eta = \perp_{\mathsf{A}}$

We express the proof

$$\pi_{\text{cmp}} \leftarrow \mathsf{NIP}\left\{(w_{i,j,u})_{i\in\mathcal{I},j\in\mathcal{J}_i,u\in[\![\ell]\!]} : \bigwedge_{i\in\mathcal{I}}\left(\bigvee_{j\in\mathcal{J}_i} \mathsf{Open}((w_{i,j,u})_{u\in[\![\ell]\!]}, (D_{i,j,u})_{u\in[\![\ell]\!]}) \leq \epsilon\right)\right\}$$

as $(\pi_{\text{cmp},i})_{i\in\mathcal{I}}$. By allocating for each $j \in \mathcal{J}_i$ and $u \in [\![\ell]\!]$, $y_{i,j,u} = D_{i,j,u}$ and $\omega_{i,j,u} = w_{j,u}$, we define $\pi_{\text{cmp},i}$ as follows:

$$\mathsf{NIP}\left\{(\omega_{i,j,u})_{\substack{j\in\mathcal{J}_i,\\ u\in[\![u_1;\ell]\!]}} : \bigvee_{j\in\mathcal{J}_i}\left(\begin{array}{l} \bigvee_{u=u_\alpha+1}^{\ell} y_{i,j,u} = h^{\omega_{i,j,u}} \\ \vee\Bigg(y_{i,j,u_\alpha} = h^{\omega_{i,j,u_\alpha}} \\ \wedge\Bigg(\bigvee_{u=u_{\alpha-1}+1}^{u_\alpha-1} y_{i,j,u} = h^{\omega_{i,j,u}} \\ \vee\Bigg(y_{i,j,u_{\alpha-1}} = h^{\omega_{i,j,u_{\alpha-1}}} \\ \wedge(\ldots \\ \wedge\Bigg(\bigvee_{u=u_1+1}^{u_2-1} y_{i,j,u} = h^{\omega_{i,j,u}} \\ \vee y_{i,j,u_1} = h^{\omega_{i,j,u_1}}\Bigg)\ldots\Bigg)\Bigg)\Bigg)\Bigg)\end{array}\right)\right\}.$$

Each proof $\pi_{\mathsf{cmp},i}$ is a composition of an OR-proof and the proof π_1 defined in C.1.2, since these proofs are correct, extractable and zero-knowledge, each proof $\pi_{\mathsf{cmp},i}$ is correct, zero-knowledge and extractable. Moreover we have the following simulator:

$\underline{\mathsf{Simulator}_{\mathsf{cmp},1}((y_{i,j,u})_{j\in\mathcal{J}_i,u\in u_1;[\![\ell]\!]}, h)}$: for each $j \in \mathcal{J}_i$, runs $(R_j, c_j, z_j) \leftarrow \mathsf{Simulator}_1((y_{i,j,u})_{u\in[\![u_1;\ell]\!]}, h)$, and parses R_j as $(R_{j,u})_{u\in[\![u_1;\ell]\!]}$, c_j as $((c_{j,u})_{u\in[\![u_1;\ell]\!]}, ch_j)$ and z_j as $(z_{j,u})_{u\in[\![u_1;\ell]\!]}$. Computes $ch = \bigoplus_{j\in\mathcal{J}_i} ch_j$. Sets $R = (R_{j,u})_{j\in\mathcal{J}_i,u\in[\![u_1;\ell]\!]}$, $c = (((c_{j,u})_{u\in[\![u_1;\ell]\!]}, ch_j)_{j\in\mathcal{J}_i}, ch)$ and $z = (z_{j,u})_{j\in\mathcal{J}_i,u\in[\![u_1;\ell]\!]}$, and returns (R, c, z).

Case η = S We assign the values $y_{i,j,u} = D_{i,j,u}$ and $\omega_{i,j,u} = w_{i,j,u}$ for each $i \in \mathcal{I}$, $j \in \mathcal{J}_i$ and $u \in [\![\ell]\!]$. The proof

$$\pi_{\mathsf{cmp}} \leftarrow \mathsf{NIP}\left\{(w_{i,j,u})_{i\in\mathcal{I},j\in\mathcal{J}_i,u\in[\![\ell]\!]} : \bigvee_{i\in\mathcal{I}}\left(\bigvee_{j\in\mathcal{J}_i}\mathsf{Open}((w_{i,j,u})_{u\in[\![\ell]\!]}, (D_{i,j,u})_{u\in[\![\ell]\!]}) < \epsilon\right)\right\}$$

consists of proving that $((y_{i,j,u})_{i\in\mathcal{I},j\in\mathcal{J}_i,u\in[\![\ell]\!]}, (\omega_{i,j,u})_{i\in\mathcal{I},j\in\mathcal{J}_i,u\in[\![\ell]\!]})) \in \mathcal{R}_{\mathsf{cmp},0}$, where $\mathcal{R}_{\mathsf{cmp},0}$ is defined as follows: $\bigvee_{i\in\mathcal{I}}\left(\bigvee_{j\in\mathcal{J}_i}\mathcal{R}_{i,j}\right)$. $\mathcal{R}_{i,j}$ denotes the relation:

$$\begin{aligned}&\bigwedge_{u=u_\alpha+1}^{\ell} y_{i,j,u} = h^{\omega_{i,j,u}} \wedge \Bigg(y_{i,j,u_\alpha} = h^{\omega_{i,j,u_\alpha}} \\ &\vee\Bigg(\bigwedge_{u=u_{\alpha-1}+1}^{u_\alpha-1} y_{i,j,u} = h^{\omega_{i,j,u}} \wedge \Bigg(y_{i,j,u_{\alpha-1}} = h^{\omega_{i,j,u_{\alpha-1}}} \\ &\vee(\ldots \\ &\vee\Bigg(\bigwedge_{u=u_1+1}^{u_2-1} y_{i,j,u} = h^{\omega_{i,j,u}} \wedge y_{i,j,u_1} = h^{\omega_{i,j,u_1}}\Bigg)\ldots\Bigg)\Bigg)\Bigg)\Bigg).\end{aligned}$$

Since this proof is the composition of an OR-proof and the proof π_0 given in C.1.1 which are correct, zero-knowledge and extractable, this proof is correct, zero-knowledge and extractable. We have the following simulator:

$\underline{\mathsf{Simulator}_{\mathsf{cmp},0}((y_{i,j,u})_{i\in\mathcal{I},j\in\mathcal{J}_i,u\in[\![u_1;\ell]\!]}, h)}$: for each $i \in \mathcal{I}$, $j \in \mathcal{J}_i$, run $(R_{i,j}, c_{i,j}, z_{i,j}) \leftarrow \mathsf{Simulator}_0((y_{i,j,u})_{u\in[\![u_1;\ell]\!]}, h)$, and parses $R_{i,j}$ as $(R_{i,j,u})_{u\in[\![u_1;\ell]\!]}$, $c_{i,j}$ as $((c_{i,j,u})_{u\in[\![u_1;\ell]\!]}, ch_{i,j})$ and $z_{i,j}$ as $(z_{i,j,u})_{u\in[\![u_1;\ell]\!]}$. Computes $ch_i = \bigoplus_{j\in\mathcal{J}_i} ch_{i,j}$ and $ch = \bigoplus_{i\in\mathcal{I}} ch_i$. Sets $R = (R_{i,j,u})_{i\in\mathcal{I},j\in\mathcal{J}_i,u\in[\![u_1;\ell]\!]}$, $c = ((((c_{i,j,u})_{u\in[\![u_1;\ell]\!]}, ch_j)_{j\in\mathcal{J}_i}, ch_i)_{i\in\mathcal{I}}, ch)$ and $z = (z_{i,j,u})_{i\in\mathcal{I},j\in\mathcal{J}_i,u\in[\![u_1;\ell]\!]}$, and returns (R, c, z).

C.3 Proofs of the theorem 1, 2 and 3

C.3.1 Proof that π in $\mathsf{VASC}_{\mathsf{Ped}}$ is correct. Since all the NIP used in $\mathsf{VASC}_{\mathsf{Ped}}$ are correct, by construction the proof π outputted by the algorithm Prove in $\mathsf{VASC}_{\mathsf{Ped}}$ is correct.

C.3.2 Proof that π in $\mathsf{VASC}_{\mathsf{Ped}}$ is zero-knowledge. We begin by formalizing the following remark:

As the following equation holds:

$$\prod_{u=1}^{\ell}\left(D_{i,j,u}\right)^{2^u} = \prod_{r=0}^{m-1}\tilde{c}_{i+r,j+r}, \tag{19}$$

and as the adversay can verify this equation, in the simulation the commitments $(\tilde{c}_{i,j})_{i,j\in[\![n]\!];j<i}$ must depend on the commitments $(D_{i,j,u})_{i\in\mathcal{I},j\in\mathcal{J}_i,u\in[\![\ell]\!]}$.

For each $i \in \mathcal{I}$ and each $j \in \mathcal{J}_i$, we have:

$$\begin{aligned}D_{i+1,j+1} &= \prod_{r=0}^{m-1}\tilde{c}_{i+r+1,j+r+1} = \prod_{r=1}^{m}\tilde{c}_{i+r,j+r} \\ &= \prod_{r=1}^{m-1}\tilde{c}_{i+r,j+r}\cdot\tilde{c}_{i+m,j+m} = \frac{D_{i,j}}{\tilde{c}_{i,j}}\cdot\tilde{c}_{i+m,j+m}\end{aligned} \tag{20}$$

The algorithm Prove returns a proof π. Since π is assumed to be zero-knowledge, there exists a simulator that simulates the proof π. We build the following simulator:

$\underline{\mathsf{Simulator}_{\mathsf{Prove}}(C, \eta, \mathsf{pp})}$**:** Parses C as $(c_i)_{i\in[\![n]\!]}$, and θ as $\left(\mathsf{Eucl}, m, \epsilon, \max_{i\in[\![n]\!]}(x_i)\right)$. For each $i \in [\![n]\!]$, $j \in [\![n]\!]$ *s.t.* $j < i$, sets $c_{i,j} = \frac{c_i}{c_j}$. Sets $\mathcal{I} = [\![n - m + 1]\!]$ and $\mathfrak{J} = (\mathcal{J}_i)_{i\in\mathcal{I}} = ([\![1; i - m/2 - 1]\!])_{i\in\mathcal{I}}$. For each $i \in \mathcal{I}$, $j \in \mathcal{J}_i$, $u \in [\![\ell]\!]$, picks $D_{i,j,u} \xleftarrow{\$} \mathbb{G}$ and sets $D_{i,j} = \prod_{u=1}^{\ell}\left(D_{i,j,u}\right)^{2^u}$.

For each $i \in \mathcal{I}$, $r \in [\![0; m-2]\!]$, picks $\tilde{c}_{i+r,1+r} \xleftarrow{\$} \mathbb{G}$ and sets $\tilde{c}_{i+m-1,1+m-1} = \frac{D_{i,1}}{\prod_{r=0}^{m-2}\tilde{c}_{i+r,1+r}}$. For each $i \in \mathcal{I}$, $j \in \mathcal{J}_i$, sets $\tilde{c}_{i+m-1,j+m-1} = \frac{D_{i-1,j-1}}{D_{i,j}}\cdot\tilde{c}_{i-1,j-1}$. Simulates $\pi_{\mathsf{sq},i,j}$ by running $\mathsf{Simulator}_{\mathsf{EQ-Open}}((c_{i,j}, \tilde{c}_{i,j}, h), (g, c_{i,j}, h))$. For each $i \in \mathcal{I}$, $j \in \mathcal{J}_i$, $u \in [\![\ell]\!]$, simulates the proof $\pi_{d_{\mathsf{bin}},i,j,u}$ by running $\mathsf{Simulator}_{\mathsf{OR}}\left(D_{i,j,u}, \frac{D_{i,j,u}}{g}, h\right)$.

- If $\eta \in \{\mathsf{A}, \perp_{\mathsf{S}}\}$, for each $i \in \mathcal{I}$, $u \in [\![\ell]\!]$, picks $M_{i,u} \xleftarrow{\$} \mathbb{G}$, simulates the proof $\pi_{MPD_{\mathsf{bin}},i,u}$ by running $\mathsf{Simulator}_{\mathsf{OR}}\left(M_{i,u}, \frac{M_{i,u}}{g}, h\right)$. For each $i \in \mathcal{I}$, simulates the proof $\pi_{MPD_{\mathsf{exist}},i}$ by running $\mathsf{Simulator}_{\mathsf{MPD}_{\mathsf{exist}}}\left(\left(\frac{M_{i,u}}{D_{i,j,u}}\right)_{j\in\mathcal{J}_i,u\in[\![\ell]\!]}, h\right)$. For each

$i \in \mathcal{I}$ and $j \in \mathcal{J}_i$, simulates the proof $\pi_{MPD_{\min},i,j}$ by running $\mathsf{Simulator}_2\left(\left(\frac{M_{i,u}}{D_{i,j,u}}\right)_{u\in[\![\ell]\!]}, h\right)$.

- If $\eta = \mathsf{A}$, computes $(\epsilon_u)_{u\in[\![\ell]\!]}$ the binary decomposition of ϵ and denotes by $(u_j)_{j\in[\![\alpha]\!]}$ the indices s.t. $\epsilon_{u_j} = 0$. Simulates the proof π_{th} by running $\mathsf{Simulator}_{\mathsf{th},0}\left(\left(\frac{M_{i,u}}{g}\right)_{i\in\mathcal{I},u\in[\![u_1;\ell]\!]}, h\right)$, and sets $\pi_{\mathsf{cmp}} = \Big((M_{i,u})_{i\in\mathcal{I},u\in[\![u_1;\ell]\!]}, (\pi_{MPD_{\mathsf{exist}},i})_{i\in\mathcal{I}}, (\pi_{MPD_{\min},i,j})_{i\in\mathcal{I},j\in\mathcal{J}_i}, \pi_{\mathsf{th}}\Big)$.
- If $\eta = \perp_{\mathsf{S}}$, computes $(\epsilon_u)_{u\in[\![\ell]\!]}$ the binary decomposition of ϵ and denotes by $(u_j)_{j\in[\![\alpha]\!]}$ the indices s.t. $\epsilon_{u_j} = 1$. For each $i \in \mathcal{I}$, simulates the proof $\pi_{\mathsf{th},i}$ by running $\mathsf{Simulator}_{\mathsf{th},1}$ $\left(\left(\frac{M_{i,u}}{g}\right)_{u\in[\![u_1;\ell]\!]}, h\right)$, and sets $\pi_{\mathsf{cmp}} = \Big((M_{i,u})_{i\in\mathcal{I},u\in[\![\ell]\!]}, (\pi_{MPD_{\mathsf{exist}},i})_{i\in\mathcal{I}}, (\pi_{MPD_{\min},i,j})_{i\in\mathcal{I},j\in\mathcal{J}_i}, (\pi_{\mathsf{th},i})_{i\in\mathcal{I}}\Big)$.

- If $\eta = \mathsf{S}$, computes $(\epsilon_u)_{u\in[\![\ell]\!]}$ the binary decomposition of ϵ and denotes by $(u_j)_{j\in[\![\alpha]\!]}$ the indices s.t. $\epsilon_{u_j} = 1$. Simulates the proof π_{cmp} by running $\mathsf{Simulator}_{\mathsf{cmp},0}\Big((D_{i,j,u})_{i\in\mathcal{I},j\in\mathcal{J}_i,u\in[\![\ell]\!]}, h\Big)$.
- If $\eta = \perp_{\mathsf{A}}$, computes $(\epsilon_u)_{u\in[\![\ell]\!]}$ the binary decomposition of ϵ and denotes by $(u_j)_{j\in[\![\alpha]\!]}$ the indices s.t. $\epsilon_{u_j} = 0$. For each $i \in \mathcal{I}$, simulates the proof $\pi_{\mathsf{cmp},i}$ by running $\mathsf{Simulator}_{\mathsf{cmp},1}\Big((D_{i,j,u})_{j\in\mathcal{J}_i,u\in[\![u_1;\ell]\!]}, h\Big)$.

Sets $\pi = ((\tilde{c}_{i,j})_{i,j\in[\![n]\!]}, (D_{i,j,u})_{i\in\mathcal{I},j\in\mathcal{J}_i,u\in[\![\ell]\!]}, (\pi_{\mathsf{sq},i,j})_{i,j\in[\![n]\!]}, (\pi_{d_{\mathsf{bin}},i,j,u})_{i\in\mathcal{I},j\in\mathcal{J}_i,u\in[\![\ell]\!]}, \pi_{\mathsf{cmp}})$, and returns π.

We notice that $\mathsf{Simulator}_{\mathsf{Prove}}$ replaces each commitment by a random element and simulates each NIP according to the equations 19 and 20. In the following, we consider a sequence of games in which the first game corresponds to the case where the adversary receives the proof generated by the algorithm Prove and the last game corresponds to the case where the adversary receives a simulated proof from $\mathsf{Simulator}_{\mathsf{Prove}}$.

We claim that:

$$\begin{aligned}&\Pr[\pi \leftarrow \mathsf{Prove}(K, C, \eta, \mathsf{pp}), b \leftarrow \mathcal{A}(\pi) : b = 1]\\ &= \Pr[\pi \leftarrow \mathsf{Simulator}_{\mathsf{Prove}}(C, \eta, \mathsf{pp}), b \leftarrow \mathcal{A}(\pi) : b = 1]\end{aligned}$$

Game G_0: This game is the case where the adversary receives a proof π generated by the algorithm Prove.

Game G_1: From the game G_0, the challenger parses pp as $\left(\mathsf{Eucl}, m, \epsilon, \max_{i\in[\![n]\!]}(x_i)\right)$, $\mathcal{I}$ as $[\![n-m+1]\!]$, $\mathcal{J}_i$ as $[\![1; i-m/2-1]\!]$, C as $(c_i)_{i\in[\![n]\!]}$ and $\pi_{(0)}$ as $\pi = ((\tilde{c}_{(0),i,j})_{i,j\in[\![n]\!]}, (D_{(0),i,j,u})_{i\in\mathcal{I},j\in\mathcal{J}_i,u\in[\![\ell]\!]}, (\pi_{(0),\mathsf{sq},i,j})_{i,j\in[\![n]\!]}, (\pi_{(0),d_{\mathsf{bin}},i,j,u})_{i\in\mathcal{I},j\in\mathcal{J}_i,u\in[\![\ell]\!]}, \pi_{(0),\mathsf{cmp}})$.

- If $\eta \in \{\mathsf{A}, \perp_{\mathsf{S}}\}$, parses $\pi_{(0),\mathsf{cmp}}$ as $\Big((M_{(0),i,u})_{i\in\mathcal{I},u\in[\![\ell]\!]}, \left(\pi_{MPD_{(0),\mathsf{exist}},i}\right)_{i\in\mathcal{I}}, (\pi_{(0),MPD_{\min},i,j})_{i\in\mathcal{I},j\in\mathcal{J}_i}, \pi_{(0),\mathsf{th}}\Big)$.

This game is the same as G_0 excepts that the challenger replaces each NIP by a simulated proof, *i.e.*, for $i \in \mathcal{I}$, $j \in \mathcal{J}_i$, the challenger replaces $\pi_{(0),\mathsf{sq},i,j}$ by a simulated proof using $\mathsf{Simulator}_{\mathsf{EQ-Open}}$, for each $i \in \mathcal{I}$, $j \in \mathcal{J}_i$, $u \in [\![\ell]\!]$, replaces $\pi_{(0),d_{\mathsf{bin}},i,j,u}$ by a simulated proof using $\mathsf{Simulator}_{\mathsf{OR}}$, and

- if $\eta \in \{\mathsf{A}, \perp_{\mathsf{S}}\}$,
  - for each $i \in \mathcal{I}$, replaces $\pi_{(0),MPD_{\mathsf{exist}},i}$ by a simulated proof using $\mathsf{Simulator}_{\mathsf{exist}}$,
  - for each $i \in \mathcal{I}$, $j \in \mathcal{J}_i$, replaces $\pi_{(0),MPD_{\min},i,j}$ by a simulated proof using $\mathsf{Simulator}_{\min}$,
    - if $\eta = \mathsf{A}$, replaces $\pi_{(0),\mathsf{th}}$ by a simulated proof using $\mathsf{Simulator}_{\mathsf{th},0}$.
- if $\eta = \perp_{\mathsf{S}}$, replaces $\pi_{(0),\mathsf{th}}$ by a simulated proof using $\mathsf{Simulator}_{\mathsf{th},1}$.
- if $\eta = \mathsf{S}$, replaces $\pi_{(0),\mathsf{cmp}}$ by a simulated proof using $\mathsf{Simulator}_{\mathsf{cmp},0}$.
- if $\eta = \perp_{\mathsf{A}}$, replaces for each $i \in \mathcal{I}$, $\pi_{(0),\mathsf{cmp},i}$ by a simulated proof using $\mathsf{Simulator}_{\mathsf{cmp},1}$.

Since all NIP are zero-knowledge, the following holds:

$$\Pr[\mathcal{A} \text{ returns 1 in } G_1] = \Pr[\mathcal{A} \text{ returns 1 in } G_0].$$

Game G_2: This game is the same as G_1 excepts that the challenger replaces $(\tilde{c}_{(0),i,j})_{i,j\in[\![n]\!];j<i}$ by random elements in $\mathbb{G}$ denoted $(\tilde{c}_{(1),i,j})_{i,j\in[\![n]\!];j<i}$.

In the following, we consider a sequence of games $(G_{1,i})_{i\in\mathcal{I}}$ in which we use the hybrid argument [10] and in which the challenger replaces $(\tilde{c}_{(0),i,j})_{j\in[\![n]\!];j<i}$ by random elements. We claim that:

$$\Pr[\mathcal{A} \text{ returns 1 in } G_2] = \Pr[\mathcal{A} \text{ returns 1 in } G_1].$$

We define $G_{1,0}$ as G_1 and $G_{1,n-m+1}$ as G_2.

Game $G_{1,i}$, $(i \in \mathcal{I}, j \in \mathcal{J}_i)$: This game is the same as $G_{1,i-1}$ excepts that the challenger replaces $(\tilde{c}_{(0),i,j})_{j\in\mathcal{J}_i}$ by random elements $(\tilde{c}_{(1),i,j})_{j\in\mathcal{J}_i}$.

We notice that as the equations 19 and 20 and hold, the commitments $(\tilde{c}_{i,j})_{i,j\in[\![n]\!];j<i}$ depend on the commitments $(D_{i,j,u})_{i\in\mathcal{I},j\in\mathcal{J}_i,u\in[\![\ell]\!]}$, some values of $(\tilde{c}_{i,j})_{i,j\in[\![n]\!];j<i}$ must be fixed.

Game $G_{1,i,1}$, $(i \in \mathcal{I})$: This game is the same as $G_{1,i,0}$ excepts that the challenger replaces $(\tilde{c}_{(0),i+r,j+r})_{r\in[\![0;m-2}$ by $(\tilde{c}_{(1),i+r,j+r})_{r\in[\![0;m-2}$ as follows:

For each $r \in [\![0; m-2]\!]$, picks $\tilde{c}_{(1),i+r,1+r} \xleftarrow{\$} \mathbb{G}$ and sets $\tilde{c}_{(1),i+m-1,1+m-1} = \frac{D_{(0),i,1}}{\prod_{r=0}^{m-2} \tilde{c}_{(1),i+r,1+r}}$.

Since in our scheme, we use Pedersen commitment which is perfectly hiding, the two games $G_{1,i,1,r}$ and $G_{1,i,1,r-1}$ are indistinguishable for the adversary, we have:

$$\Pr\left[\mathcal{A} \text{ returns 1 in } G_{1,i,1}\right] = \Pr\left[\mathcal{A} \text{ returns 1 in } G_{1,i,0}\right].$$

We can notice that $(\tilde{c}_{(1),i+r,1+r})_{r\in[\![0;m-1]\!]}$ are now fixed, the challenger do not have to replace them in the following games.

Game $G_{1,i,1,r}$, $(i \in \mathcal{I}, r \in [\![0; m-2]\!])$**:** This game is the same as $G_{1,i,1,r-1}$ excepts that the challenger replaces $\tilde{c}_{(0),i+r,1+r}$ by $\tilde{c}_{(1),i+r,1+r} \xleftarrow{\$} \mathbb{G}$ and sets $\tilde{c}_{(1),i+m-1,1+m-1} = \frac{D_{(0),i,1}}{\prod_{k=0}^{r} \tilde{c}_{(1),i+k,1+k} \cdot \prod_{k=r+1}^{m-2} \tilde{c}_{(0),i+k,1+k}}$.

The two games $G_{1,i,1,r}$ and $G_{1,i,1,r-1}$ are indistinguishable for the adversary, we have:

$$\Pr\left[\mathcal{A} \text{ returns 1 in } G_{1,i,1,r}\right] = \Pr\left[\mathcal{A} \text{ returns 1 in } G_{1,i,1,r-1}\right].$$

Game $G_{1,i,j}$, $(i \in \mathcal{I}, j \in \mathcal{J}_i; j > 1)$**:** For simplicity, we denote the predecessor of j belonging to $\mathcal{J}_i$ by $j-1$. This game is the same as $G_{1,i,j-1}$ excepts that the challenger replaces $(\tilde{c}_{(0),i+m-1,j+m-1})_{i \in \mathcal{J}_i}$ by $(\tilde{c}_{(1),i+m-1,j+m-1})_{j \in \mathcal{J}_i}$ as follows:
For each $i \in \mathcal{I}$, $j \in \mathcal{J}_i$, sets $\tilde{c}_{i+m-1,j+m-1} = \frac{D_{i-1,j-1}}{D_{i,j}} \cdot \tilde{c}_{i-1,j-1}$.

We can notice that in this paper, by simplicity, the algorithm Prove computes additional commitments $(\tilde{c}_{(0),i,j})_{i,j \in [\![n]\!]; i > n-m+1; j < i-m/2-1}$, that are not linked to some $D_{(0),i,j}$. For these commitments, the challenger must replace them by random.

Game G_3**:** This game is the same as G_2 excepts that the challenger replaces the commitments $(\tilde{c}_{(0),i,j})_{i,j \in [\![n]\!]; i > n-m+1; j < i-m/2-1}$ by random elements $(\tilde{c}_{(1),i,j})_{i,j \in [\![n]\!]; i > n-m+1; j < i-m/2-1}$ in $\mathbb{G}$.

We claim that:

$$\Pr\left[\mathcal{A} \text{ returns 1 in } G_3\right] = \Pr\left[\mathcal{A} \text{ returns 1 in } G_2\right].$$

In the following, we consider a sequence of games $(G_{3,i})_{i \in [\![n]\!]; i > n-m+1}$ in which we use the hybrid argument [10] and in which the challenger replaces $(\tilde{c}_{(0),i,j})_{j \in [\![n]\!]; j < i-m/2-1}$ by random elements in $\mathbb{G}$.

We define $G_{2,0}$ as G_2 and $G_{2,n}$ as G_3.

Game $G_{2,i}$, $(i \in [\![n]\!]; i > n-m+1)$**:** This game is the same as $G_{2,i-1}$ excepts that the challenger replaces the commitments $(\tilde{c}_{(0),i,j})_{j \in [\![n]\!]; j < i-m/2-1 \in [\![}$ by random elements in $\mathbb{G}$.
We claim that:

$$\Pr\left[\mathcal{A} \text{ returns 1 in } G_{2,i}\right] = \Pr\left[\mathcal{A} \text{ returns 1 in } G_{2,i-1}\right].$$

We define $G_{2,i,0}$ as $G_{3,i-1}$ and $G_{2,i,i-m/2-2}$ as $G_{3,i+1}$.

Game $G_{2,i,j}$, $(i, j \in [\![n]\!]; i > n-m+1; j < i-m/2-1$**:** This game is the same as $G_{2,i,j-1}$ excepts that the challenger replaces $\tilde{c}_{(0),i,j}$ by $\tilde{c}_{(1),i,j} \xleftarrow{\$} \mathbb{G}$. As we use Pedersen commitment which is perfectly hiding, the following hold:

$$\Pr\left[\mathcal{A} \text{ returns 1 in } G_{2,i,j}\right] = \Pr\left[\mathcal{A} \text{ returns 1 in } G_{2,i,j-1}\right].$$

Game G_4**:** This game is the same as G_3 excepts that the challenger replaces $(D_{(0),i,j,u})_{i \in \mathcal{I}, j \in \mathcal{J}_i, u \in [\![\ell]\!]}$ by random elements $(D_{(1),i,j,u})_{i \in \mathcal{I}, j \in \mathcal{J}_i, u \in [\![\ell]\!]}$, each picked in $\mathbb{G}$. Since the equations 19 and 20, the commitments $(\tilde{c}_{(0),i,j})_{i,j \in [\![n]\!]; j < i}$ depend on the commitments $(D_{(0),i,j,u})_{i,j \in [\![n]\!]; j < i}$.

In the following, we consider a sequence of games $(G_{3,i})_{i \in \mathcal{I}}$ in which we use the hybrid argument [10] and in which the challenger replaces for each $i \in \mathcal{I}$, $(D_{(0),i,j,u})_{j \in \mathcal{J}_i, u \in [\![\ell]\!]}$ by random elements $(D_{(1),i,j,u})_{i \in \mathcal{I}, j \in \mathcal{J}_i, u \in [\![\ell]\!]}$ and the commitments $(\tilde{c}_{(1),i,j})_{i,j \in [\![n]\!]; j < i}$ were computed as in the previous game G_3. We claim that:

$$\Pr\left[\mathcal{A} \text{ returns 1 in } G_4\right] = \Pr\left[\mathcal{A} \text{ returns 1 in } G_3\right].$$

We define $G_{3,0}$ as G_3 and $G_{3,n-m+1}$ as G_4.

Game $G_{3,i}$, $(i \in \mathcal{I})$**:** This game is the same as $G_{3,i-1}$ except that the challenger replaces for each $j \in \mathcal{J}_i$, $(D_{(0),i,j,u})_{j \in \mathcal{J}_i, u \in [\![\ell]\!]}$ by random elements $(D_{(1),i,j,u})_{j \in \mathcal{J}_i, u \in [\![\ell]\!]}$ and $(\tilde{c}_{(0),i,j})_{i,j \in [\![n]\!]; j < i}$ are replaced by $(\tilde{c}_{(1),i,j})_{i,j \in [\![n]\!]; j < i}$.

We claim that:

$$\Pr\left[\mathcal{A} \text{ returns 1 in } G_{3,i}\right] = \Pr\left[\mathcal{A} \text{ returns 1 in } G_{3,i-1}\right].$$

We define $G_{1,i,0}$ as $G_{1,i-1}$ and $G_{1,i,i-m/2-1}$ as $G_{1,i+1}$.

Game $G_{3,i,1}$, $(i \in \mathcal{I})$**:** This game is the same as $G_{3,i,0}$ excepts that the challenger replaces $(D_{(0),i,1,u})_{u \in [\![\ell]\!]}$ by random elements $(D_{(1),i,1,u})_{u \in [\![\ell]\!]}$ and $(\tilde{c}_{(0),i,j})_{i,j \in [\![n]\!]; j < i}$ are replaced by $(\tilde{c}_{(1),i,j})_{i,j \in [\![n]\!]; j < i}$.

We claim that:

$$\Pr\left[\mathcal{A} \text{ returns 1 in } G_{3,i,1}\right] = \Pr\left[\mathcal{A} \text{ returns 1 in } G_{3,i,0}\right].$$

We define $G_{3,i,1,0}$ as $G_{3,i-1,1}$ and $G_{3,i,1,\ell}$ as $G_{3,i,2}$.

Game $G_{3,i,1,u}$, $(i \in \mathcal{I}, u \in [\![\ell]\!])$**:** This game is the same as $G_{3,i,0,u-1}$ excepts that the challenger replaces $D_{(0),i,1,u}$ by a random element $D_{(1),i,1,u} \xleftarrow{\$} \mathbb{G}$. The challenger computes $D_{(1),i,1} = \prod_{k=1}^{u} \left(D_{(1),i,1,k}\right)^{2^k} \cdot \prod_{k=u+1}^{\ell} \left(D_{(0),i,1,k}\right)^{2^k}$ and $(\tilde{c}_{(0),i+r,1+r})_{r \in [\![0;m-1]\!]}$ are replaced by $(\tilde{c}_{(1),i+r,1+r})_{r \in [\![0;m-1]\!]}$ as follows:
For each $r \in [\![0; m-2]\!]$, $\tilde{c}_{(1),i+r,1+r} \xleftarrow{\$} \mathbb{G}$ and sets $\tilde{c}_{(1),i+m-1,1+m-1} = \frac{D_{(1),i,1}}{\prod_{r=0}^{m-2} \tilde{c}_{(1),i+r,1+r}}$.
Since the Pedersen commitment, is perfectly hiding, we have:

$$\Pr\left[\mathcal{A} \text{ returns 1 in } G_{3,i,1,u}\right] = \Pr\left[\mathcal{A} \text{ returns 1 in } G_{3,i,1,u-1}\right].$$

Game $G_{3,i,j}$, $(i \in \mathcal{I}, j \in \mathcal{J}_i; j > 1)$**:** This game is the same as $G_{3,i,j-1}$ excepts that the challenger replaces $(D_{(0),i,j,u})_{u \in [\![\ell]\!]}$ by random elements $(D_{(1),i,j,u})_{u \in [\![\ell]\!]}$ and $\tilde{c}_{(0),i+m-1,j+m-1}$ is replaced by $\tilde{c}_{(1),i+m-1,j+m-1}$.

We claim that:

$$\Pr\left[\mathcal{A} \text{ returns 1 in } G_{3,i,j}\right] = \Pr\left[\mathcal{A} \text{ returns 1 in } G_{3,i,j-1}\right].$$

We define $G_{3,i,j,0}$ as $G_{3,i,j-1}$ and $G_{3,i,j,\ell}$ as $G_{3,i,j+1}$.

Game $G_{3,i,j,u}$, $(i \in \mathcal{I}, j \in \mathcal{J}_i, u \in [\![\ell]\!]; j > 1)$**:** This game is the same as $G_{3,i,j,u-1}$ excepts that the challenger replaces $D_{(0),i,j,u}$ by a random element $D_{(1),i,j,u} \xleftarrow{\$} \mathbb{G}$. The challenger computes $D_{i,j} = \prod_{k=1}^{u} \left(D_{(1),i,j,k}\right)^{2^k} \cdot$

$\prod_{k=u+1}^{\ell} \left(D_{(0),i,j,k}\right)^{2^k}$ and replaces $\tilde{c}_{(0),i+m-1,j+m-1}$ by $\tilde{c}_{(1),i+m-1,j+m-1} = \frac{D_{(1),i-1,j-1}}{D_{i,j}} \cdot \tilde{c}_{(1),i-1,j-1}$.
Since the Pedersen commitment, is perfectly hiding, we have:

$$\Pr\left[\mathcal{A} \text{ returns 1 in } G_{3,i,j,u}\right] = \Pr\left[\mathcal{A} \text{ returns 1 in } G_{3,i,j-1,u-1}\right].$$

This concludes the proof in the case where $\eta \in \{\perp_{\mathsf{A}}, \mathsf{S}\}$.

The following considers an additional sequence of games for the case where $\eta \in \{\mathsf{A}, \perp_{\mathsf{S}}\}$ exclusively.

Game G_5 : This game is the same as G_4 excepts that the challenger replaces $(M_{(0),i,u})_{i\in\mathcal{I},u\in[\![\ell]\!]}$ by random elements in $\mathbb{G}$.

We claim that:

$$\Pr\left[\mathcal{A} \text{ returns 1 in } G_5\right] = \Pr\left[\mathcal{A} \text{ returns 1 in } G_4\right].$$

In the following, we consider a sequence of games $(G_{4,i})_{i\in\mathcal{I}}$ in which we use the hybrid argument [10] and in which for each $i \in \mathcal{I}$, the challenger replaces $(M_{(0),i,u})_{u\in[\![\ell]\!]}$ by random elements.

We define $G_{4,i,0}$ as $G_{4,i-1}$ and $G_{4,i,\ell}$ as $G_{4,i+1}$.

Game $G_{4,i,u}$: This game is the same as $G_{4,i,u-1}$ excepts that the challenger replaces $M_{(0),i,u}$ by $M_{(1),i,u} \xleftarrow{\$} \mathbb{G}$. Since we use Pedersen commitments which is perfectly hiding, we have:

$$\Pr\left[\mathcal{A} \text{ returns 1 in } G_{4,i,u}\right] = \Pr\left[\mathcal{A} \text{ returns 1 in } G_{4,i,u-1}\right].$$

This concludes the proof in the case where $\eta \in \{\mathsf{A}, \perp_{\mathsf{S}}\}$.

Finally, $\mathsf{Prove}(K, C, \eta, \mathsf{pp})$ and $\mathsf{Simulator}_{\mathsf{Prove}}(C, \eta, \mathsf{pp})$ follows the same distribution, this concludes the proof of the theorem.

C.3.3 *Proof that π in $\mathsf{VASC}_{\mathsf{Ped}}$ is extractable.*

- $\mathcal{R}_{\mathsf{sq},i,j}$ defines the relation:
  $c_{i,j} = g^{x_{i,j}} h^{k_{i,j}} \wedge \tilde{c}_{i,j} = c_{i,j}^{x_{i,j}} h^{\tilde{k}_{i,j}}$.
- $\mathcal{R}_{d_{\mathsf{bin}},i,j,u}$ denotes the relation:
  $D_{i,j,u} = h^{w_{i,j,u}} \vee \frac{D_{i,j,u}}{g} = h^{w_{i,j,u}}$.
- $\mathcal{R}_{MPD_{\mathsf{bin}},i,u}$ denotes the relation:
  $M_{i,u} = h^{z_{i,u}} \vee \frac{M_{i,u}}{g} = h^{z_{i,u}}$.
- $\mathcal{R}_{MPD_{\mathsf{exist}},i,u}$ denotes the relation:
  $\bigvee_{j\in\mathcal{J}_i} \left(\bigwedge_{u=1}^{\ell} \frac{M_{i,u}}{D_{i,j,u}} = h^{z_{i,u}-w_{i,j,u}}\right)$.
- $\mathcal{R}_{MPD_{\mathsf{min}},i,j}$ denotes the relation:
  $\frac{gM_{i,\ell}}{D_{i,j,\ell}} = h^{z_{i,\ell}-w_{i,j,\ell}} \vee \left(\frac{M_{i,\ell}}{D_{i,j,\ell}} = h^{z_{i,\ell}-w_{i,j,\ell}}\right.$
  $\wedge \left(\frac{gM_{i,\ell-1}}{D_{i,j,\ell-1}} = h^{z_{i,\ell-1}-w_{i,j,\ell-1}} \vee \left(\frac{M_{i,\ell-1}}{D_{i,j,\ell-1}} = h^{z_{i,\ell-1}-w_{i,j,\ell-1}}\right.\right.$
  $\wedge \left(\ldots \wedge \left(\frac{gM_{i,1}}{D_{i,j,1}} = h^{z_{i,1}-w_{i,j,1}} \vee \left(\frac{M_{i,1}}{D_{i,j,1}} = h^{z_{i,1}-w_{i,j,1}}\right)\right)\ldots\right)\right)\right)\right)$.
- $\mathcal{R}_{\mathsf{th},0}$ denotes the relation:
  $\bigvee_{i\in\mathcal{I}} \left(\bigwedge_{u=u_\alpha+1}^{\ell} \frac{M_{i,u}}{g} = h^{z_{i,u}} \wedge \left(\frac{M_{i,u_\alpha}}{g} = h^{z_{i,u_\alpha}}\right.\right.$
  $\vee \left(\bigwedge_{u=u_{\alpha-1}+1}^{u_\alpha-1} \frac{M_{i,u}}{g} = h^{z_{i,u}} \wedge \left(\frac{M_{i,u_{\alpha-1}}}{g} = h^{z_{i,u_{\alpha-1}}}\right.\right.$
  $\vee \left(\ldots \vee \left(\bigwedge_{u=u_1+1}^{u_2-1} \frac{M_{i,u}}{g} = h^{z_{i,u}} \wedge \left(\frac{M_{i,u_1}}{g} = h^{z_{i,u_1}}\right)\right)\ldots\right)\right)\right)\right)\right)$.
- $\mathcal{R}_{\mathsf{th},1}$ denotes the relation:
  $\bigwedge_{i\in\mathcal{I}} \left(\bigvee_{u=u_\alpha+1}^{\ell} \frac{M_{i,u}}{g} = h^{z_{i,u}} \vee \left(\frac{M_{i,u_\alpha}}{g} = h^{z_{i,u_\alpha}}\right.\right.$
  $\wedge \left(\bigvee_{u=u_{\alpha-1}+1}^{u_\alpha-1} \frac{M_{i,u}}{g} = h^{z_{i,u}} \vee \left(\frac{M_{i,u_{\alpha-1}}}{g} = h^{z_{i,u_{\alpha-1}}}\right.\right.$
  $\wedge \left(\ldots \wedge \left(\bigvee_{u=u_1+1}^{u_2-1} \frac{M_{i,u}}{g} = h^{z_{i,u}} \vee \frac{M_{i,u_1}}{g} = h^{z_{i,u_1}}\right)\ldots\right)\right)\right)\right)\right)$.
- $\mathcal{R}_{\mathsf{cmp},0}$ denotes the relation:
  $\bigvee_{i\in\mathcal{I}} \left(\bigvee_{j\in\mathcal{J}_i} \left(\bigwedge_{u=u_\alpha+1}^{\ell} D_{i,j,u} = h^{w_{i,j,u}} \wedge \left(D_{i,j,u_\alpha} = h^{w_{i,j,u_\alpha}}\right.\right.\right.$
  $\vee \left(\bigwedge_{u=u_{\alpha-1}+1}^{u_\alpha-1} D_{i,j,u} = h^{w_{i,j,u}} \wedge \left(D_{i,j,u_{\alpha-1}} = h^{w_{i,j,u_{\alpha-1}}} \vee \left(\ldots\right.\right.\right.$
  $\left.\left.\left.\left.\left.\vee \left(\bigwedge_{u=u_1+1}^{u_2-1} D_{i,j,u} = h^{w_{i,j,u}} \wedge D_{i,j,u_1} = h^{w_{i,j,u_1}}\right)\ldots\right)\right)\right)\right)\right)\right)$.
- $\mathcal{R}_{\mathsf{cmp},1}$ denotes the relation:
  $\bigwedge_{i\in\mathcal{I}} \left(\bigvee_{j\in\mathcal{J}_i} \left(\bigvee_{u=u_\alpha+1}^{\ell} D_{i,j,u} = h^{w_{i,j,u}} \vee \left(D_{i,j,u_\alpha} = h^{w_{i,j,u_\alpha}}\right.\right.\right.$
  $\wedge \left(\bigvee_{u=u_{\alpha-1}+1}^{u_\alpha-1} D_{i,j,u} = h^{w_{i,j,u}} \vee \left(D_{i,j,u_{\alpha-1}} = h^{w_{i,j,u_{\alpha-1}}} \wedge \left(\ldots\right.\right.\right.$
  $\left.\left.\left.\left.\left.\wedge \left(\bigvee_{u=u_1+1}^{u_2-1} D_{i,j,u} = h^{w_{i,j,u}} \vee \left(D_{i,j,u_1} = h^{w_{i,j,u_1}}\right)\right)\ldots\right)\right)\right)\right)\right)\right)$.
- if $\eta = \mathsf{A}$, we define $\mathcal{R}$ as follows:
  $\left(\bigwedge_{i,j\in[\![n]\!];j<i} \mathcal{R}_{\mathsf{sq},i,j}\right) \wedge \left(\bigwedge_{i\in\mathcal{I},j\in\mathcal{J}_i,u\in[\![\ell]\!]} \mathcal{R}_{d_{\mathsf{bin}},i,j,u}\right)$
  $\wedge \left(\bigwedge_{i\in\mathcal{I},u\in[\![\ell]\!]} \mathcal{R}_{MPD_{\mathsf{bin}},i,u}\right) \wedge \left(\bigwedge_{i\in\mathcal{I},u\in[\![\ell]\!]} \mathcal{R}_{MPD_{\mathsf{exist}},i,u}\right)$
  $\wedge \left(\bigwedge_{i\in\mathcal{I},j\in\mathcal{J}_i} \mathcal{R}_{MPD_{\mathsf{min}},i,j}\right) \wedge \mathcal{R}_{\mathsf{th},0}$.
- if $\eta = \perp_{\mathsf{S}}$, we define $\mathcal{R}$ as follows:
  $\left(\bigwedge_{i,j\in[\![n]\!];j<i} \mathcal{R}_{\mathsf{sq},i,j}\right) \wedge \left(\bigwedge_{i\in\mathcal{I},j\in\mathcal{J}_i,u\in[\![\ell]\!]} \mathcal{R}_{d_{\mathsf{bin}},i,j,u}\right)$
  $\wedge \left(\bigwedge_{i\in\mathcal{I},u\in[\![\ell]\!]} \mathcal{R}_{MPD_{\mathsf{bin}},i,u}\right) \wedge \left(\bigwedge_{i\in\mathcal{I},u\in[\![\ell]\!]} \mathcal{R}_{MPD_{\mathsf{exist}},i,u}\right)$
  $\wedge \left(\bigwedge_{i\in\mathcal{I},j\in\mathcal{J}_i} \mathcal{R}_{MPD_{\mathsf{min}},i,j}\right) \wedge \mathcal{R}_{\mathsf{th},1}$.
- if $\eta = \mathsf{S}$, we define $\mathcal{R}$ as follows:
  $\left(\bigwedge_{i,j\in[\![n]\!];j<i} \mathcal{R}_{\mathsf{sq},i,j}\right) \wedge \left(\bigwedge_{i\in\mathcal{I},j\in\mathcal{J}_i,u\in[\![\ell]\!]} \mathcal{R}_{d_{\mathsf{bin}},i,j,u}\right)$
  $\wedge \left(\bigwedge_{i\in\mathcal{I},j\in\mathcal{J}_i} \mathcal{R}_{MPD_{\mathsf{min}},i,j}\right) \wedge \mathcal{R}_{\mathsf{cmp},0}$.

- if $\eta = \perp_{\mathsf{A}}$, we define $\mathcal{R}$ as follows:

$$\left(\bigwedge_{i,j\in[\![n]\!];j<i} \mathcal{R}_{\mathsf{sq},i,j}\right) \wedge \left(\bigwedge_{i\in\mathcal{I},j\in\mathcal{J}_i,u\in[\![\ell]\!]} \mathcal{R}_{d_{\mathsf{bin}},i,j,u}\right)$$
$$\wedge \left(\bigwedge_{i\in\mathcal{I},j\in\mathcal{J}_i} \mathcal{R}_{MPD_{\mathsf{min}},i,j}\right) \wedge \mathcal{R}_{\mathsf{cmp},1}.$$

Let ω be the witness and s be the statement for the relation $\mathcal{R}$.

Game G_0: This game is the same as the extractability experiment, we have:

$$\Pr[\mathcal{A} \text{ wins } G_0] = \Pr\left[\begin{array}{l} (\pi,\pi') \leftarrow \mathcal{A}^{\mathsf{Simulator}(.)}(s), \\ \mathsf{NIPVerify}(s,\pi)=1, \\ \mathsf{NIPVerify}(s,\pi')=1, \\ \omega \leftarrow \mathsf{Ext}(s,\pi,\pi') \end{array} : (s,\omega)\notin\mathcal{R}\right].$$

Game G_1: This game is the same as G_0 excepts that $\mathcal{C}$ uses the extractors on the proofs outputted by $\mathcal{A}$ and aborts and returns 0 on the event F_1 = "An extractor fails on at least one proof". $\mathcal{C}$ extracts the witnesses:

- $(x_{i,j})_{i,j\in[\![n]\!];j<i}$, $(k_{i,j})_{i,j\in[\![n]\!];j<i}$ and $(\tilde{k}_{i,j})_{i,j\in[\![n]\!];j<i}$ from $(\pi_{\mathsf{sq},i,j})_{i,j\in[\![n]\!];j<i}$.
- $(w'_{i,j,u})_{i\in\mathcal{I},j\in\mathcal{J}_i,u\in[\![\ell]\!]}$ from $(\pi_{d_{\mathsf{bin}},i,j,u})_{i\in\mathcal{I},j\in\mathcal{J}_i,u\in[\![\ell]\!]}$.
- If $\eta \in \{\mathsf{A},\perp_{\mathsf{S}}\}$, $\mathcal{C}$ extracts the following witnesses:
  - $(z'_{i,u})_{i\in\mathcal{I},u\in[\![\ell]\!]}$ from $(\pi_{MPD_{\mathsf{bin}},i,u})_{i\in\mathcal{I},u\in[\![\ell]\!]}$.
  - $(z_{i,u}-w_{i,j,u})_{i\in\mathcal{I},j\in\mathcal{J}_i,u\in[\![\ell]\!]}$ from $(\pi_{MPD_{\mathsf{exist}},i})_{i\in\mathcal{I}}$.
  - $(\overline{z_{i,u}-w_{i,j,u}})_{i\in\mathcal{I},j\in\mathcal{J}_i,u\in[\![\ell]\!]}$ derived from $(\pi_{MPD_{\mathsf{min}},i,j})_{i\in\mathcal{I},j\in\mathcal{J}_i}$.
  - $(\hat{z_{i,u}})_{i\in\mathcal{I},u\in[\![u_1;\ell]\!]}$ from π_{th}.
- If $\eta \in \{\perp_{\mathsf{A}},\mathsf{S}\}$, $\mathcal{C}$ extracts the witnesses $(w_{i,j,u})_{i\in\mathcal{I},j\in\mathcal{J}_i,u\in[\![u_1;\ell]\!]}$ from π_{cmp}.

If $\mathcal{A}$ wins its game, the witnesses are correctly extracted.

- If $\eta \in \{\mathsf{A},\perp_{\mathsf{S}}\}$, we emphasize that for each $u \in [\![u_1;\ell]\!]$, $z'_{i,u} = \hat{z_{i,u}}$ because $(\pi_{MPD_{\mathsf{bin}},i,u})_{i\in\mathcal{I},u\in[\![\ell]\!]}$ and π_{th} implicitly use the same $M_{i,u}$. Similarly, for each $i \in \mathcal{I}, j \in \mathcal{J}_i, u \in [\![\ell]\!]$, $z_{i,u} - w_{i,j,u} = \overline{z_{i,u}-w_{i,j,u}}$ and $\overline{z_{i,u}-w_{i,j,u}} = z'_{i,u} - w'_{i,j,u}$.
- If $\eta \in \{\perp_{\mathsf{A,S}}\}$, we have: for each $i \in \mathcal{I}, j \in \mathcal{J}_i, u \in [\![u_1;\ell]\!]$, $w_{i,j,u} = w'_{i,j,u}$.

Let $\epsilon_{\mathsf{ext}}(\lambda)$ be the maximum probability of failure of the extractors. Since all NIP are extractables, we have:

$$|\Pr[\mathcal{A} \text{ wins } G_1] - \Pr[\mathcal{A} \text{ wins } G_0]| \le \epsilon_{\mathsf{ext}}(\lambda).$$

Game G_2: This game is the same as G_1 excepts that aborts and returns 0 on the event F_2 ="The extractor extracts $(w_{i,j,u})_{i\in\mathcal{I},j\in\mathcal{J}_i,u\in[\![\ell]\!]}$, $(x_{i,j})_{i,j\in[\![n]\!];j<i}$, $(k_{i,j})_{i,j\in[\![n]\!];j<i}$ and $(\tilde{k}_{i,j})_{i,j\in[\![n]\!];j<i}$ $s.t.$ there exists $i \in \mathcal{I}$, $j \in \mathcal{J}_i$ $s.t.$ $\sum_{u=1}^{\ell} d_{i,j,u}\cdot 2^u \neq \sum_{r=0}^{m-1} x^2_{i+r,j+r}$ and $\sum_{u=1}^{\ell} w_{i,j,u}\cdot 2^u \neq \sum_{r=0}^{m-1} x_{i+r,j+r}\cdot k_{i+r,j+r} + \tilde{k}_{i+r,j+r}$ and $\prod_{u=1}^{\ell}(D_{i,j,u})^{2^u} = \prod_{r=0}^{m-1}\tilde{c}_{i+r,j+r}$".

We claim that:

$$|\Pr[\mathcal{A} \text{ wins } G_2] = \Pr[\mathcal{A} \text{ wins } G_1]| \le \Pr[F_2].$$

We prove this claim by reduction, we suppose that the event F_2 occurs with a non negligible probability, we build a PPT algorithm $\mathcal{B}$ whose purpose is to break the binding property of Pedersen commitment.

$\underline{\mathcal{B}(\tilde{g},\tilde{h}):}$

simulates the game G_1 to the adversary excepts that during the setup generation, the algorithm replaces g by $\tilde{g}$ and h by $\tilde{h}$. $\mathcal{B}$ generates $x \xleftarrow{\$} (\mathbb{Z}_p^*)^n$ and runs the algorithm $(K,C) \leftarrow \mathsf{Com}(x)$. Runs $(\pi,\eta,\mathsf{pp}) \leftarrow \mathcal{A}(C)$. $\mathcal{B}$ extracts the witnesses as in G_1. We note $(x_{i,j})_{i,j\in[\![n]\!];j<i}$, $(k_{i,j})_{i,j\in[\![n]\!];j<i}$ and $(\tilde{k}_{i,j})_{i,j\in[\![n]\!];j<i}$ the witnesses extracted from $(\pi_{\mathsf{sq},i,j})_{i,j\in[\![n]\!];j<i}$, and $(w'_{i,j,u})_{i\in\mathcal{I},j\in\mathcal{J}_i,u\in[\![\ell]\!]}$ the witnesses extracted from $(\pi_{d_{\mathsf{bin}},i,j,u})_{i\in\mathcal{I},j\in\mathcal{J}_i,u\in[\![\ell]\!]}$. Parses pp as $(\mathsf{Eucl}, m, \epsilon, \max_{i\in[\![n]\!]}(x_i))$, and π as $((\tilde{c}_{i,j})_{i,j\in[\![n]\!]}, (D_{i,j,u})_{i\in\mathcal{I},j\in\mathcal{J}_i,u\in[\![\ell]\!]}, (\pi_{\mathsf{sq},i,j})_{i,j\in[\![n]\!]}, (\pi_{d_{\mathsf{bin}},i,j,u})_{i\in\mathcal{I},j\in\mathcal{J}_i,u\in[\![\ell]\!]}, \pi_{\mathsf{cmp}})$. If the event F_2 occurs, $\mathcal{B}$ finds $i \in \mathcal{I}$, $j \in \mathcal{J}_i$ $s.t.$ $\sum_{u=1}^{\ell} d_{i,j,u}\cdot 2^u \neq \sum_{r=0}^{m-1} x^2_{i+r,j+r}$ and $\sum_{u=1}^{\ell} w_{i,j,u}\cdot 2^u \neq \sum_{r=0}^{m-1} x_{i+r,j+r}\cdot k_{i+r,j+r} + \tilde{k}_{i+r,j+r}$ and $\prod_{u=1}^{\ell}(D_{i,j,u})^{2^u} = \prod_{r=0}^{m-1}\tilde{c}_{i+r,j+r}$, $\mathcal{B}$ returns $\left(\left(\sum_{r=0}^{m-1} x^2_{i+r,j+r}, \sum_{r=0}^{m-1} x_{i+r,j+r}k_{i+r,j+r} + \tilde{k}_{i+r,j+r}\right), \left(\sum_{u=1}^{\ell} d_{i,j,u}\cdot 2^u, \sum_{u=1}^{\ell} w_{i,j,u}\cdot 2^u\right), \prod_{r=0}^{m-1}\tilde{c}_{i+r,j+r}\right)$, else $\mathcal{B}$ aborts and returns nothing.

Finally, $\mathcal{B}$ returns two different way to open the Pedersen commitment $\prod_{r=0}^{m-1}\tilde{c}_{i+r,j+r}$, since the Pedersen commitment is computationally binding this happens with a negligible probability $\epsilon_{\mathsf{binding}}(\lambda)$, we have:

$$|\Pr[\mathcal{A} \text{ wins } G_2] = \Pr[\mathcal{A} \text{ wins } G_1]| \le \Pr[F_2] \le \epsilon_{\mathsf{binding}}(\lambda).$$

Finally, we have:

$$\Pr\left[\begin{array}{l} (\pi,\pi') \leftarrow \mathcal{A}^{\mathsf{Simulator}(.)}(s), \\ \mathsf{NIPVerify}(s,\pi)=1, \\ \mathsf{NIPVerify}(s,\pi')=1, \\ \omega \leftarrow \mathsf{Ext}(s,\pi,\pi') \end{array} : (s,\omega)\notin\mathcal{R}\right].$$

$\le \epsilon_{\mathsf{ext}}(\lambda) + \epsilon_{\mathsf{binding}}(\lambda)$.

At this point, the adversary has no other possibility to win the game, this concludes the proof of the theorem.