cryptography

MDPI

Letter

On Secret Sharing with Newton's Polynomial for Multi-Factor Authentication

Sergey Bezzateev [1], Vadim Davydov [2] and Aleksandr Ometov [3,*]

[1] Technologies of Information Security, Saint Petersburg State University of Aerospace Instrumentation, 190000 St. Petersburg, Russia; bsv@aanet.ru

[2] Faculty of Secure Information Technologies, ITMO University, 197101 St. Petersburg, Russia; vadimdavydov@outlook.com

[3] Tampere University, 33720 Tampere, Finland

* Correspondence: aleksandr.ometov@tuni.fi

Received: 1 September 2020; Accepted: 26 November 2020; Published: 1 December 2020

Abstract: Security and access control aspects are becoming more and more essential to consider during the design of various systems and the tremendous growth of digitization. One of the related key building blocks in this regard is, essentially, the authentication process. Conventional schemes based on one or two authenticating factors can no longer provide the required levels of flexibility and pro-activity of the access procedures, thus, the concept of threshold-based multi-factor authentication (MFA) was introduced, in which some of the factors may be missing, but the access can still be granted. In turn, secret sharing is a crucial component of the MFA systems, with Shamir's schema being the most widely known one historically and based on Lagrange interpolation polynomial. Interestingly, the older Newtonian approach to the same problem is almost left without attention. At the same time, it means that the coefficients of the existing secret polynomial do not need to be re-calculated while adding a new factor. Therefore, this paper investigates this known property of Newton's interpolation formula, illustrating that, in specific MFA cases, the whole system may become more flexible and scalable, which is essential for future authentication systems.

Keywords: authentication; interpolation; Netwon's polynomial; secret sharing

1. Introduction

Today, the digital evolution, along with information and communications technology (ITC) developments, already engross most areas of modern society. Nonetheless, to enable the secure and private operation of such various co-existing systems, we must develop different information security instruments. One of those corresponds to the system's authentication with its user being either a machine or a human being [1–3].

Standalone growth of authentication systems, from ownership factor (key, access card, etc.) towards more complicated knowledge (pin, password, etc.) and biometric-based authentication (fingerprint, facial recognition, gait, etc.), led to the appearance of multi-factor authentication (MFA) (also known as multimodal authentication in the biometry field) systems [4,5]. MFA utilizes an intelligent combination of factors of different types and has already become a strong use case for car-sharing and banking systems [6,7]. Here, each involved factor-provider (physical token, secret code, biometrics, etc.) in the system has some secret share derived from the access key. When the key is restored based on a threshold number of collected shares, the authentication is considered a success. The technology behind has its roots in the field of secret sharing.

One of the most famous secret sharing techniques is the well-known Shamir's secret sharing scheme [8]. In the classical version of the scheme, the Lagrange interpolation polynomial is used to recover the secret, but the addition of new key shares may be a complicated task.

Many modern systems require a proactive/continuous operation, which is of specific interest in biometry-related authentication [9]. In the example of car-sharing systems that generally are expected to utilize biometry-based factors if one of the authentication sensors breaks, it is necessary to quickly add a new one with an increase (or modification) in the scheme's threshold. That procedure may be challenging to execute using the Lagrange polynomial when dividing the secret due to the peculiarities of constructing the polynomial. Broadly speaking, the entire system should be reinitialized.

The paper provides an investigation of a known property of the Newton interpolation formula for MFA systems. Namely, using Newton's interpolation formula instead of the Lagrange interpolation formula with Shamir's secret sharing scheme is more beneficial. That is mainly since the entire system can be made more flexible and scalable since the coefficients of the existing secret polynomial do not need to be re-calculated [10,11]. Instead of the Lagrange polynomial, its use allows for quick changes of the threshold value, thereby greatly facilitating and simplifying the process of flexible system changes and prompt additions of new factors.

The paper is organized as follows. Section 2 provides a brief history of interpolation formulas, along with descriptions of Lagrange's and Newton's polynomials. Further, Section 3 describes the processes of obtaining the Lagrange formula from Newton's formula and provides the inverse transformation. Next, Section 4 is devoted directly to secret sharing schemes based on the interpolation formulas. Further, Section 5 elaborates on Newton's polynomial potential for secret sharing and an example of such use. Significantly, Section 6 provides some examples of polynomials' utilization for MFA systems and shows the proof of Newton's polynomial property for the sake of completeness. The last section concludes the paper.

2. Overview of Interpolation Polynomials

Generally, interpolation is a method of finding a new solution using a predefined set of results (values) of the function. The unknown value could be defined as a point using this formula. Considering a linear interpolation formula, it may be used to find a new value from two (or another number) of specified points. If we compare such a method with Lagrange's interpolation formula, a set of points would be required, and the Lagrange method will be used to find the new value [10].

As an example, consider a linear interpolation formula

$$y - y_1 = \frac{y_2 - y_1}{x_2 - x_1} x - x_1, \tag{1}$$

where points (x_1, y_1) and (x_2, y_2) are known.

Note, there is a significant difference between interpolation and unambiguous reconstruction of a polynomial from its values at a certain number of known points. Finding an interpolation function has many solutions since infinitely many curves can be drawn through given points; each of those will be a graph of a function for which all interpolation conditions are satisfied.

Next, we show the general interpolation scheme for a polynomial in one variable. Consider t points and the following polynomial of at most $(t-1)$ degree with $a_0, \ldots, a_{t-1} \in \mathbb{Z}$ as

$$a(x) = a_0 + \sum_{j=1}^{t-1} a_j x^j. \tag{2}$$

Thus, it could be represented as follows.

$$a(x) = a_0 + a_1 x + \cdots + a_{t-1} x^{t-1}, \tag{3}$$

where $a_0, a_1, ..., a_{t-1}$ are unknown.

Since t points are already known ($\{(x_1, y_1), (x_2, y_2), \ldots, (x_t, y_t)\}$), we arrive at t linear independent equations with t unknown variables:

$$\begin{aligned} y_1 &= a_0 + a_1 x_1 + a_2 x_1^2 + \cdots + a_{t-1} x_1^{t-1}, \\ y_2 &= a_0 + a_1 x_2 + a_2 x_2^2 + \cdots + a_{t-1} x_2^{t-1}, \\ &\ldots \\ y_t &= a_0 + a_1 x_t + a_2 x_t^2 + \cdots + a_{t-1} x_t^{t-1}. \end{aligned} \tag{4}$$

By rewriting in a matrix form, we arrive at

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_t \end{pmatrix} = \begin{pmatrix} 1 & x_1 & \ldots & x_1^{t-1} \\ 1 & x_2 & \ldots & x_2^{t-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_t & \ldots & x_t^{t-1} \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{t-1} \end{pmatrix}, \tag{5}$$

i.e., a Vandermonde matrix, and there are well-known techniques to find its determinant. Finally, it is not difficult to add a new point after finding values $(a_0, a_1, \ldots, a_{t-1})$.

2.1. Newton's Interpolation Formula

Scientists began to reevaluate the power of interpolation very long ago. The first mentions date back to about 300 BC when Babylonian astronomers used interpolation to fill in the gaps in the planetary celestial coordinate tables known at that time and recorded the data on special tablets. In this work, we will not go deep into the history of the emergence of interpolation formulas since there were no major discoveries before the era of the scientific revolution starting in 1611.

As early as in the 17th century, scientists used an interpolation formula equivalent to the Gregory–Newton formula for the approximation of the function $f(x)$ and finally gave the linear interpolation.

In 1675, Isaac Newton began his fundamental work on the classical theory of interpolation. In 1711, his "Methodus Differentialis" met the world [12]. Newton described his discoveries that became fundamental and are used to this day in many real systems.

Almost 80 years later, Edward Waring published an alternative representation of Newton's general formula for arbitrary interval data in 1779. The method did not require the computation of separated differences, although this formula is attributed to another scientist, Joseph-Louis Lagrange, who proposed the same formula 16 years later [12].

The Newton formula is an interpolation polynomial designed for a given set of data points. It is also called the Newton interpolation polynomial with separation of differences since the polynomial coefficients are calculated using the method of division of Newton differences [13].

For example, $(n+1)$ points are defined as $(x_0, y_0), \ldots (x_j, y_j), \ldots, (x_n, y_n)$. The values defined by $x_j (j = 0, \ldots, n)$ are interpolation points. The values defined by $y_j (j = 0, \ldots, n)$ are interpolation values. For the interpolation of function f, the interpolation values are determined as

$$y_j = f(x_j), \forall j = 0, \ldots, n. \tag{6}$$

The polynomial of Newton's basis is defined as follows

$$n_i(x) = \prod_{j=1}^{i-1} (x - x_j), \tag{7}$$

where $i = 1, \ldots, n$ and $n_0(x) = 1$.

The Newton interpolation polynomial is defined as

$$P_n(x) = \sum_{i=0}^{n} K_i n_i(x) = K_0 + K_1(x - x_0) + K_2(x - x_0)(x - x_1) + \ldots + K_n(x - x_0)\ldots(x - x_{n-1}), \quad (8)$$

where $P_n(x_j) = f(x_j), \forall_j = 0, \ldots, n$.

The Newton interpolation polynomial of degree n, $P_n(x)$, estimated by x_0, is

$$P_n(x_0) = \sum_{i=0}^{n} K_i n_i(x_0) = a_0 = f(x_0) = f[x_0], \quad (9)$$

$$P_n(x_j) = f(x_j), \forall j = 0, \ldots, n, \quad (10)$$

where $f[x_0]$ is a zero order divided difference.

The Newton interpolation polynomial of degree n, $P_n(x)$, estimated by x_1, is

$$P_n(x_1) = \sum_{i=0}^{n} K_i n_i(x_1) = K_0 + K_1(x_1 - x_0) = f[x_0] + K_1(x_1 - x_0) = f[x_1]. \quad (11)$$

Therefore,

$$K_1 = \frac{f[x_1] - f[x_0]}{x_1 - x_0} = f[x_0, x_1], \quad (12)$$

where $f[x_0, x_1]$ is the 1-st order divided difference.

Then, K_i in general form could be written as

$$K_i = \frac{f[x_1, \ldots, x_i] - f[x_0, \ldots, x_{(i-1)}]}{x_i - x_0}, \quad (13)$$

where $f[x_1, \ldots, x_i]$ is the n-th order divided difference.

2.2. Lagrange Interpolation Formula

Even though Waring published an alternative representation of Newton's general formula for arbitrary interval data first (which did not require the computation of separated differences), this finding was regrettably attributed to another scientist, Joseph-Louis Lagrange, who proposed the same formula 16 years later [12].

The Lagrange formula is traditionally represented as a polynomial, where for $(n + 1)$ pairs of numbers $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$, where $x_i \neq x_j$ for all $i \neq j$, there is only one a polynomial $L(x)$ of degree n, for which $L(x_j) = y_j$ for all $j = 0, \ldots, n$ as

$$L(x) = \sum_{i=0}^{n} y_i l_i(x), \quad (14)$$

where $l_i(x) = \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j}$ are basic polynomials with the following properties:

1. $\deg l_i(x) = n$;
2. $l_i(x_i) = 1$;
3. $l_i(x_j) = 0$ if $j \neq i$.

3. Interpolation Formulas Relation

3.1. Derivation of Lagrange's Formula from Newton's Formula

The Lagrange interpolation formula can be obtained directly from Newton's formula. Let us consider this transition in more detail.

From (8), we obtain

$$f(x_i) = P_n(x_i) = \sum_{j=0}^{n} K_j n_j(x_i). \tag{15}$$

Therefore, we can use the following interpolation formula by multiplying each $P_n(x_i)$ by coefficient $l_i(x)$ as

$$f(x) = \sum_{i=0}^{n} \sum_{j=0}^{i} K_j n_j(x_i) l_i(x), \tag{16}$$

$$l_i(x) = \frac{(x-x_0)(x-x_1)\dots(x-x_{i-1})(x-x_{i+1})\dots(x-x_n)}{(x_i-x_0)(x_i-x_1)\dots(x_i-x_{i-1})(x_i-x_{i+1})\dots(x_i-x_n)}. \tag{17}$$

3.2. Derivation of Newton's Formula from Lagrange's Formula

Newton's interpolation formula also can be obtained directly from the Lagrange formula.

We consider polynomials $P_0(x) = f(x_0), P_1(x), \dots P_{n-1}(x)$. In the classical form, Lagrange polynomial is written as

$$P_n(x) = \sum_{i=0}^{n} y_i \prod_{k=0, k\neq i}^{n} \frac{x-x_k}{x_i-x_k}. \tag{18}$$

Let us do a trick and represent the polynomial as the following sum

$$P_n(x) = P_0(x) + \sum_{i=1}^{n} (P_i(x) - P_{i-1}(x)). \tag{19}$$

From the interpolation definition, we get that

$$P_{j-1}(x_k) = P_j(x_k) = f(x_k) \tag{20}$$

for $k = 0, 1 \dots, j-1$ and $j = 1, 2, \dots, n$, which means that $P_j(x) - P_{j-1}(x)$ is an algebraic polynomial with the zeroes in $x_0, x_1, \dots, x_{j-1}$.

Then,

$$P_j(x) - P_{j-1}(x) = A_j(x_j - x_0)(x_j - x_1)\dots(x_j - x_{j-1}), \tag{21}$$

where A_j is some number. This number could be found knowing the fact $P_j(x_j) = f(x_j)$:

$$A_j = \frac{f(x_j) - P_{j-1}(x_j)}{(x_j - x_0)\dots(x_j - x_{j-1})}. \tag{22}$$

Substituting the value of P_{j-1} using Lagrange interpolation formula, we arrive at

$$A_j = \sum_{k=0}^{l} \frac{f(x_k)}{(x_k - x_0)\dots(x_k - x_{k-1})(x_k - x_{k+1})\dots(x_k - x_j)}. \tag{23}$$

Here, it can be observed that this value is the divided difference of the order j: $A_j = f(x_0, x_1, \dots x_j)$.

Expressing the rest of the polynomials recursively, we arrive at

$$\begin{aligned} P_n(x) =& K_0 + K_1(x - x_0) + K_2(x - x_0)(x - x_1) + \cdots + K_n(x - x_0) \ldots (x - x_{n-1}) \\ =& K_0 + \sum_{i=0}^{n} K_i \prod_{j=0}^{i-1} (x - x_j). \end{aligned} \tag{24}$$

Thus, we have shown the possibility of obtaining one interpolation formula from another using intermediate, auxiliary elements of the corresponding interpolation formulas.

3.3. Additional Notes on the Comparison of the Polynomials

Finally, while speaking about the Lagrange vs. Newton comparison in terms of the number of calculations required to compute the coefficients, it should be pointed out that the Lagrange coefficient's calculation requires n^2 additions and multiplications. For Newton, it is only $n^2/2$ [14], which indicates a benefit of the utilization of the latter one, especially for resource-constrained devices.

Interestingly, the authors of [15] highlight the possibility of using pre-calculated coefficients in the Lagrange formula in the case of a fixed, previously known group of users (devices, factors, parameters, etc.) with the corresponding identifiers (denoted x_i in this work and as u_i in [15]). Under this condition, one can pre-calculate the coefficients for the Lagrange interpolation function, significantly reducing computation when assembling a secret. However, when the values of the identifiers (x_i or u_i) of the participants of the secret assembly are not known in advance, or a new factor/parameter appears (as in the following example of the MFA operation,) such a simplification of calculations for the Lagrange formula, unfortunately, cannot be performed.

4. Shamir's Secret Sharing Scheme

The first (k, n) threshold secret sharing scheme was proposed in 1979 by Shamir [16] and Blackley [17], who worked independently of each other. The basic principle of these schemes is sharing the secret between the participants in the information exchange when everyone has only a part of the key. The secret can be recovered only if a specified number of key shares are combined.

Any schema of this type, as a rule, consists of secret sharing and secret recovery phases [18]. For example, let D be a secret data. The scheme's purpose would be to divide D into n shares $D_1 \ldots D_n$ with the following conditions met:

1. D is easily calculated if k or more of its shares are known;
2. D cannot be calculated if $k - 1$ or fewer of its parts are known.

Shamir's secret sharing scheme is based on the Lagrange interpolation formula and is still one of today's most popular schemes. Besides, there are schemes based on the Chinese remainder theorem (CRT) for integer rings proposed by Mignotte [19] and by Asmuth and Bloom [20], and on error-correcting codes proposed by McEliece and Sarwate using non-systematic Reed–Solomon codes [21].

The following subsection will describe the main phases of Shamir's secret sharing scheme in more detail.

4.1. Secret Sharing

Let us denote a prime number $p > D$. This number is known to all participants in the exchange. Let us construct a polynomial from $\mathbb{F}_p[x]$ of degree $l - 1$ with random coefficients and $a_0 = D$ as

$$L(x) = a_{l-1}x^{l-1} + a_{l-2}x^{l-2} + \cdots + a_1 x + D. \tag{25}$$

The secret shares are calculated using

$$\begin{aligned} D_1 &= L(1), \\ D_2 &= L(2), \\ &\dots \\ D_n &= L(n). \end{aligned} \tag{26}$$

As a result, each exchange participant receives his share of the secret and its number.

4.2. Recovering a Secret

To recover the secret, the Lagrange interpolation polynomial is used. Here, $(l+1)$ pairs of numbers $(x_0, y_0), (x_1, y_1), \dots, (x_l, y_l)$ with $x_i \neq x_j$ for all $i \neq j$ and there is one and only one polynomial $L(x)$ of the degree l, for which $L(x_j) = y_j$ for all $j = 0, \dots, l$.

The following equations are used for the calculation:

$$\begin{aligned} D = L(0) &= \sum_{i=0}^{l} y_i l_i(0) \mod p, \\ l_i(0) &= (-1)^l \prod_{j=0, j\neq i}^{l} \frac{x_j}{x_i - x_j} \mod p. \end{aligned} \tag{27}$$

The above formulas have the following properties:

1. Polynomial degree n;
2. $l_i(x_i) = 1$;
3. $l_i(x_i) = 0$ if $j \neq i$.

As shown earlier, Shamir's secret sharing scheme is based on the Lagrange interpolation polynomial. With an increase in the degree of a polynomial, and accordingly, an increase in the number of interpolation nodes required to restore it, it is necessary to rebuild the entire polynomial, which is inconvenient when using MFA. Let us demonstrate this statement with an example.

Suppose we have a car-sharing system based on the Lagrange polynomial, and currently, three factors are used to authenticate the driver. However, the owner wants to add a new factor (a new immobilizer or a pin code). At the current moment, the system has already calculated the values $l_0, l_1 \dots l_3$ and calculated the final polynomial $L(x)$. When adding a new factor, the product in the formulas for calculating l_i must be counted again since n has changed. Thus, it is necessary to recalculate all the values l_0, l_1, l_2, and l_3, and calculate the new value of l_4, which is quite costly computation-wise. In order to solve this problem, one can use another representation, namely, the Newton polynomial.

5. Newton's Polynomial in Secret Sharing Schemes

Newton's interpolation polynomial of degree n can be represented as

$$P_n(x) = \sum_{i=0}^{n} K_i n_i(x) = f[x_0] + \sum_{i=1}^{n} f[x_0, \dots, x_i] n_i(x). \tag{28}$$

Note that increasing the degree of the polynomial by one while the values of the polynomial at the previously known points remain unchanged will require adding the $(n+1)$-th interpolation node, and accordingly, calculating the coefficient

$$K_{n+1} = \frac{f[x_1, \dots, x_{n+1}] - f[x_0, \dots, x_n]}{x_{n+1} - x_0}, \tag{29}$$

as well as $K_{n+1}(x-x_0)\dots(x-x_n)$ values. Moreover, since the polynomial values at previously known points do not change, for the remaining degrees of the polynomial they do not change. Basically, there is no need to recalculate the previous coefficients; one only need add a new one.

The recalculated Newton interpolation polynomial of degree $n+1$ can be, thus, represented as

$$P_{n+1}(x) = \sum_{i=0}^{n+1} K_i n_i(x) = f[x_0] + \sum_{i=1}^{n+1} F[x_0,\dots,x_i]n_i(x) = P_n(x) + f[x_0,\dots,x_{n+1}]n_{n+1}(x). \quad (30)$$

In the Lagrange interpolation polynomial, if it is necessary to increase the degree of the polynomial we have to recalculate the values of the polynomial at the previous known points and the corresponding addition of one more $(n+1)$-th point. All interpolation coefficients must be recalculated since each term of the polynomial takes a single correct value, and when a new point is added, the values of all terms of the polynomial need to be aligned according to the new point.

6. Utilization Examples

6.1. Use Case Description

Let us assume to use an MFA system as a combination with biometric factors, a password, and a physical key. At first, let this system has two biometric factors: fingerprint and facial recognition.

These factors will be represented as two points with the coordinates $(x_1, y_1), (x_2, y_2)$. By using Lagrange and Newton interpolation formulas, we calculate a function (in this particular case, linear function) $f(x)$, which goes through two these points. Then, we choose password and key values as some points on this curve $(x_{key}, y_{key}), (x_{pass}, y_{pass})$. Finally, we have a curve $f(x), \deg f(x) = 1$ with four factors that are represented as points $(x_1, y_1), (x_2, y_2), (x_{key}, y_{key})$, and (x_{pass}, y_{pass}) on it.

It becomes straightforward to construct a threshold system $k = 2$ from $n = 4$ by using such a curve.

Assume, we wish to add to this MFA system one more biometric factor represented as the point with the coordinates (x_3, y_3). In this case, we have to calculate new function $\phi(x)$, which goes through three points $(x_1, y_1), (x_2, y_2)$, and (x_3, y_3) associated with the corresponding three biometric factors. Therefore, we should replace the function $f(x)$ with function $\phi(x)$. After that, in the same way as in the previous case, we choose password and key values as some points $(x_{key'}, y_{key'}), (x_{pass'}$, and $y_{pass'})$ on this curve.

Finally, we have a curve $\phi(x), \deg \phi(x) = 2$ with five factors which are represented as points $(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_{key'}, y_{key'})$, and $(x_{pass'}, y_{pass'})$ in it.

Using this curve allows us to construct a threshold system $k = 3$ from $n = 5$.

Now, we will resolve this case in terms of Lagrange and Newton interpolation formulas.

To construct a new curve associated with the function $\phi(x), \deg(\phi(x)) = 2$, we should use two previous known points $(x_1, y_1), (x_2, y_2)$ and one new point (x_3, y_3). Using Lagrange interpolation formula, it is necessary to change the polynomial, and thereby all basis polynomials must be recalculated. In Newton's polynomial form, there is an advantage that only the last term must be found to obtain the new polynomial. Next, we show some simple examples of adding one new biometric factor using two variants of polynomials.

6.2. Lagrange Interpolation Example

This section shows a simple example of how to add a new point using a Lagrange interpolation formula.

At first, let us assume that there are two known points $(x_1, y_1), (x_2, y_2)$ associated with two biometric factors, and it is required to find the function $f(x)$, which goes through the corresponding two points.

x	x_1	x_2
	-1	0
$f(x)$	y_1	y_2
	4	2

To find the function $f(x)$, we use the Lagrange interpolation formula

$$f(x) = \frac{(x - x_2)}{(x_1 - x_2)} \cdot y_1 + \frac{(x - x_1)}{(x_2 - x_1)} \cdot y_2. \tag{31}$$

Then, we simply calculate $f(x)$:

$$f(x) = \frac{(x - 0)}{(-1 - 0)} \cdot 4 + \frac{(x + 1)}{(0 + 1)} \cdot 2 = -4x + 2x + 2 = -2x + 2. \tag{32}$$

If we add a third biometric factor as a point (x_3, y_3) on the second step, we obtain the following.

x	x_1	x_2	x_3
	-1	0	1
$f(x)$	y_1	y_2	y_3
	4	2	-2

To find a new function $\phi(x)$ with degree 2, we should recalculate all coefficients in the following way.

$$\phi(x) = \frac{(x - x_2)(x - x_3)}{(x_1 - x_2)(x_1 - x_3)} \cdot y_1 + \frac{(x - x_1)(x - x_3)}{(x_2 - x_1)(x_2 - x_3)} \cdot y_2 + \frac{(x - x_1)(x - x_2)}{(x_3 - x_1)(x_3 - x_2)} \cdot y_3. \tag{33}$$

Therefore, we obtain

$$\begin{array}{ll} \phi(x) & = \frac{(x-0)(x-1)}{((-1)-0)((-1)-1)} \cdot 4 + \frac{(x-(-1))(x-1)}{(0-(-1))(0-1)} \cdot 2 + \frac{(x-(-1))(x-0)}{((1)-(-1))((1-0)} \cdot (-2) \\ & = -x^2 - 3x + 2. \end{array} \tag{34}$$

Here, we demonstrated how to find a new interpolation function using the dataset extended by one biometric factor. In this case, all basis polynomials must be recalculated.

6.3. Newton Interpolation Example

This subsection presents a simple example of how to add a point $(x_3, y_3) = (1, -2)$ using Newton interpolation formula. The same example as in the previous subsection is used.

x	x_1	x_2
	-1	0
$f(x)$	y_1	y_2
	4	2

To find the function $f(x)$, we use Newton interpolation formula

$$f(x) = K_0 + K_1(x - x_1). \tag{35}$$

Let us find divided differences:

$$K_0 = y_1 = 4, \tag{36}$$

$$K_1 = f[x_1, x_2] = \frac{y_2 - y_1}{x_2 - x_1} = \frac{-2}{1} = -2. \tag{37}$$

Next, we calculate the function $f(x)$ as

$$f(x) = 4 + (-2) \cdot (x + 1) = -2x + 2. \tag{38}$$

If we add a third biometric factor as a point (x_3, y_3) at the second step, we obtain the following.

x	x_1	x_2	x_3
	-1	0	1
$f(x)$	y_1	y_2	y_3
	4	2	-2

Then, we only have to find a new divided difference instead of full recalculation as

$$f[x_1, x_3] = \frac{y_3 - y_1}{x_3 - x_1} = \frac{-6}{2} = -3, \tag{39}$$

$$K_2 = \frac{f[x_1, x_3] - K_1}{x_3 - x_2} = \frac{(-3) + 2}{1} = -1, \tag{40}$$

and next, we just need to calculate the polynomial

$$\phi(x) = K_0 + K_1(x - x_1) + K_2(x - x_1)(x - x_2) = f(x) + K_2(x - x_1)(x - x_2), \tag{41}$$

$$\phi(x) = -2x + 2 - (x + 1)x = -2x + 2 - x^2 - x = -x^2 - 3x + 2. \tag{42}$$

This simple example clearly shows the features of two interpolation algorithms. The presented calculations demonstrate the fundamental difference between the Newton formula and the Lagrange formula: when the number of interpolation points increases in the Newton formula, it is unnecessary to recalculate all the previous components of the interpolation formula.

7. Conclusions

The evolution of the authentication schemes towards multi-factor ones leads to the need to redesign the access procedures to become more proactive with regard to potentially missing factors (due to sensor failure or the human factor). In this paper, we propose to use the Newton polynomial instead of the Lagrange polynomial in secret sharing schemes. The resulting flexible scheme can quickly change the number and composition of factors, which means that secret sharing schemes based on Newton's interpolation polynomial are potentially suitable for organizing MFA and can be recommended for use in various modern systems that require a timely reaction to the changes.

Comparing the Newton and Lagrange interpolation formulas, some significant advantages of using it should be highlighted. Firstly, Newton's polynomial application allows for faster execution time compared to Lagrange's interpolation, since the latter requires divisions, whereas Newton's polynomial does not use divisions at all. Secondly, the coefficients in the Newton form can be found somewhat faster than in the Lagrange form ($n^2/2$ versus n^2), since the use of Newton's polynomial does not require full recalculation of the polynomial. Finally, it was shown that one could substantially reduce complexity using Newton's polynomial instead of the Lagrange one when necessary to increase the number of interpolation points in secret sharing schemes.

Author Contributions: Conceptualization, S.B. and V.D.; methodology, S.B.; validation, A.O.; formal analysis, A.O.; investigation, S.B. and V.D.; writing—original draft preparation, V.D.; writing—review and editing, S.B. and A.O.; visualization, A.O.; supervision, S.B.; project administration, A.O. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Wang, D.; Wang, P.; Wang, C. Efficient Multi-Factor User Authentication Protocol with Forward Secrecy for Real-Time Data Access in WSNs. *ACM Trans. Cyber Phys. Syst.* **2020**, *4*, 1–26. [CrossRef]
2. Ometov, A.; Petrov, V.; Bezzateev, S.; Andreev, S.; Koucheryavy, Y.; Gerla, M. Challenges of Multi-Factor Authentication for Securing Advanced IoT Applications. *IEEE Netw.* **2019**, *33*, 82–88. [CrossRef]
3. Das, S.; Wang, B.; Tingle, Z.; Camp, L.J. Evaluating User Perception of Multi-Factor Authentication: A Systematic Review. *arXiv* **2019**, arXiv:1908.05901.
4. Ometov, A.; Bezzateev, S.; Mäkitalo, N.; Andreev, S.; Mikkonen, T.; Koucheryavy, Y. Multi-Factor Authentication: A Survey. *Cryptography* **2018**, *2*, 1. [CrossRef]
5. Kumar, K.; Farik, M. A Review of Multimodal Biometric Authentication Systems. *Int. J. Sci. Technol. Res.* **2016**, *5*, 5–9. [CrossRef]
6. Genovese, A.; Munoz, E.; Piuri, V.; Scotti, F. Advanced Biometric Technologies: Emerging Scenarios and Research Trends. In *From Database to Cyber Security*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 324–352.
7. Park, S.H.; Kim, J.H.; Jun, M.S. A Design of Secure Authentication Method with Bio-Information in the Car Sharing Environment. In *Advances in Computer Science and Ubiquitous Computing*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 205–210.
8. Shamir, A. How to Share a Secret. *Commun. ACM* **1979**, *22*, 612–613. [CrossRef]
9. Alotaibi, S.; Alruban, A.; Furnell, S.; Clarke, N.L. A Novel Behaviour Profiling Approach to Continuous Authentication for Mobile Applications. In proceedings of the International Conference on Information Systems Security and Privacy, Prague, Czech Republic, 23–25 February 2019; pp. 246–251.
10. Kogan, N.; Tassa, T. Improved Efficiency for Revocation Schemes via Newton Interpolation. *ACM Trans. Inf. Syst. Secur. (TISSEC)* **2006**, *9*, 461–486. [CrossRef]
11. Stavros, D.; Iraklis, S. Complexity Comparison of Lagrange and Newton Polynomial based Revocation Schemes. In Proceedings of the 2nd Conference on European Computing Conference, Athens, Greece, 25–27 September 2007; World Scientific and Engineering Academy and Society (WSEAS): Athens, Greece, 2008; pp. 44–53.
12. Meijering, E. A Chronology of Interpolation: From Ancient Astronomy to Modern Signal and Image Processing. *Proc. IEEE* **2002**, *90*, 319–342. [CrossRef]
13. Sebah, P.; Gourdon, X. Newton's Method and High Order Iterations. *Numbers Comput.* **2001**, *1*, 10.
14. Werner, W. Polynomial Interpolation: Lagrange versus Newton. *Math. Comput.* **1984**, *43*, 205–217. [CrossRef]
15. Naor, M.; Pinkas, B. Efficient Trace and Revoke Schemes. In Proceedings of the International Conference on Financial Cryptography, Anguilla, UK, 21–24 February 2000; Springer: Berlin/Heidelberg, Germany, 2000; pp. 1–20.
16. O'Gorman, L. Comparing Passwords, Tokens, and Biometrics for User Authentication. *Proc. IEEE* **2003**, *91*, 2021–2040. [CrossRef]
17. Blakley, G.R. Safeguarding Cryptographic Keys. In Proceedings of the International Workshop on Managing Requirements Knowledge (MARK), New York, NY, USA, 4–7 June 1979; IEEE: Piscataway, NJ, USA 1979; pp. 313–318.
18. Kaya, K.; Selçuk, A.A. Secret Sharing Extensions based on the Chinese Remainder Theorem. *IACR Cryptol. ePrint Arch.* **2010**, *2010*, 96.
19. Mignotte, M. How to Share a Secret. In *Workshop on Cryptography*; Springer: Berlin/Heidelberg, Germany, 1982; pp. 371–375.
20. Asmuth, C.; Bloom, J. A Modular Approach to Key Safeguarding. *IEEE Trans. Inf. Theory* **1983**, *29*, 208–210. [CrossRef]
21. McEliece, R.J.; Sarwate, D.V. On Sharing Secrets and Reed-Solomon Codes. *Commun. ACM* **1981**, *24*, 583–584. [CrossRef]