Secret sharing a key in a distributed way, Lagrange vs Newton

Anastassios Voudouris
University of Piraeus
Piraeus, Greece
anastassis.voudouris@ssl-unipi.gr

Ilias Politis
InQbit Innovations SRL
Bucharest, Romania
ilias.politis@inqbit.io

Christos Xenakis
University of Piraeus
Piraeus, Greece
xenakis@unipi.gr

ABSTRACT

In secret sharing, a dealer knows the secret it shares. In a distributed key generation (DKG) protocol, a shared secret is collectively generated in a group in a completely distributed way such that any subset of size greater than a threshold can reveal or use the shared secret, while smaller subsets do not have any knowledge about it. The most important aspect is that there is no dealer or trusted party. The core idea of secret sharing schemes is Shamir's secret sharing method, which uses Lagrange's interpolation to reconstruct the shared secret key. This paper investigates an alternative method, called Newton's interpolation and it cites the probability of implementing it on current DKG protocols.

CCS CONCEPTS

• **Security and privacy** → **Cryptography**; • **Mathematics of computing**; • **Applied computing**; • **Networks** → Network reliability;

KEYWORDS

Secret sharing, Distributed key generation , threshold, Lagrange interpolation, Newton.

ACM Reference Format:
Anastassios Voudouris, Ilias Politis, and Christos Xenakis. 2022. Secret sharing a key in a distributed way, Lagrange vs Newton. In *The 17th International Conference on Availability, Reliability and Security (ARES 2022), August 23–26, 2022, Vienna, Austria.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3538969.3544424

1 INTRODUCTION

Generally, most cryptographic schemes, and especially asymmetric ones, manage the public-key encryption through a public key infrastructure (PKI). PKI provides secure communication on an insecure public network and uses a digital signature to verify the identity of the entities. The PKI relies on the security of a central control point, named the Certification Authority (CA), and is considered trusted by all and is considered trusted by all, however it also acts as a single point of failure. If this point gets compromised, the security of the entire network will fall down. Recent advancements in technology [11] showed that the research community needs to level up current cryptographic protocols in a decentralized manner. New ideas are driven by a well-studied area of cryptography called threshold cryptography. A (n, k) threshold cryptography scheme allows n entities to share the ability to perform a cryptographic operation so that any k entities suffices to perform this operation jointly, whereas it is infeasible for at most $k - 1$ entities to do so, even by collusion. The cryptographic key, which will be used to encrypt the message from one person to another, is the fundamental tool for both symmetric and asymmetric cryptography and must be protected at all costs, from the generation part to the secret-sharing of it among the entities of the group.

In secret sharing, a dealer knows the secret it shares. Therefore, it has to be trusted as in the case of CA for the PKI and remain intact during key sharing or at least ensure the secrecy of the key before its reconstruction. Many cryptosystems require the complete distribution of trust and no party (not even a dealer) should be in sole possession of the secret. This is the only way to eliminate the single point of failure. So the concept of distributed key generation (DKG) comes in hand. In a DKG protocol, a shared secret is collectively generated in a group in a fully distributed way such that any subset of size greater than a threshold can reveal or use the shared secret, while smaller subsets do not have any knowledge about it. The most important aspect is that there is no dealer or trusted party; hence, each node in the group runs an instance of a secret sharing scheme and computes its final share by adding the shares it has received from the other members of the group. The secret-shared private keys can later be used in a threshold cryptosystem, e.g., to produce threshold signatures, decrypt ciphertexts of threshold encryption or generate common coins, but before they can be used, shares have to be gathered and through interpolation methods, the original keys will be reconstructed.

DKG protocol is more than a secret sharing protocol. In a secret sharing protocol the secret shares can be used to recover the group secret, but this can be done only once. After everyone has learned the group secret the shares cannot be reused. In a DKG the shares can be used repeatedly for an unlimited number of group signatures without ever recovering the group secret key explicitly. Most DKG protocols are discrete-log based cryptosystems and typically utilize multiple instances of a verifiable secret sharing protocol (VSS).

Most of the existing DKG protocols assume the synchronous model and asynchronous DKG received attention only recently, with poor efficiency though. DKG, secret sharing, and threshold cryptography are all branches of the same tree, called in general terms multiparty computation (MPC). One security application of MPC is to protect cryptographic keys from theft and misuse, by never having them exposed at any single point at any single time,

ARES 2022, August 23–26, 2022, Vienna, Austria
© 2022 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-9670-7/22/08…$15.00
https://doi.org/10.1145/3538969.3544424

and by enforcing usage policies at multiple entities. This provides the ability to protect cryptographic keys in software, providing an alternative to legacy hardware solutions, such as TPM's root of trust, that introduces many operational challenges in today's hybrid and multi-cloud environments.

In this paper, a brief overview of basics schemes is provided on how the keys are generated and later distributed among the members of a group that wants to communicate. A comparison is also made between the two methods of interpolation, Lagrange and Newton, with the former being the core of the Shamir Secret Sharing scheme and the latter is proposed as an improvement. There is a theoretical proposal regarding multi-factor authentication in the work of Bezzateev [4]. The main objective is to perform a comparative analysis of the Lagrange and Newton interpolation methods to identify the potential of replacing the former method on the existing DKG schemes, considering also a trusted setup for new nodes joining a DKG protocol. Shamir's secret sharing is theoretically information secure, meaning that its security cannot be improved, so the focus is on performance. Newton can accommodate vector or matrix interpolation from scalar data which is very useful on threshold biometric authentication and also supports incremental interpolation, e.g., adding new nodes without the need for calculating the polynomial from scratch, which the present paper considers will have an impact on performance.

The rest of the paper is structured as follows. Section 2 provides an overview of the related work on DKG schemes and secret sharing. Section 3 analyses Shamir's secret sharing scheme with reference also to Feldman's verifiable secret sharing scheme, which is a slight improvement of Shamir's on the verification of the shares. Section 4 analyses Pedersen's Distributed key generation scheme and in the fifth section lies the core idea of the paper, where Lagrange and Newton's interpolations are analyzed and a small comparison is made between the two methods. The paper concludes with proposed future work and open problems.

2 RELATED WORK

2.1 Related Work on DKG

Numerous works have studied the problem of DKG with various cryptographic assumptions and network conditions. Most of the works have been in the synchronous network model. Pedersen proposed the first DKG protocol [28] using a non-interactive information-theoretic verifiable secret sharing. Gennaro showed that Pedersen's protocol allows an attacker to bias the public-key distribution by selectively denouncing one or more of the parties it controls, and as a result influence the final outcome. Gennaro's approach adds complexity as it requires an additional secret sharing step using Pedersen VSS protocol [15]. Neji proposed a simple mechanism to mitigate the bias-attack illustrated by Gennaro [25]. Neji's paper was not very specific about the simulator that was used, so in ADKG paper [10] authors presented a new proof of secrecy. Canetti presented an extension of Gennaro in order to be secure against an adaptive adversary [7]. Fouque and Stern presented a one-round, publicly-verifiable DKG that uses only public channels to make the protocol non-interactive [12]. Their final transcript size is $O(n^2)$ and their security proof does not allow for rushing adversaries and make use of use of Paillier encryption which is likely to make the DKG slow and have high communication costs. Their advantage is that the outputting secrets are field, rather than group, elements, so they can be used with current threshold schemes. Gurkan designed a PVSS-based DKG protocol with a linear size public-verification transcript [17]. However, the secret key in their DKG is a group element instead of a field element, so their scheme is incompatible with off-the-shelf threshold signature or encryption schemes. Later, Groth designed a non-interactive DKG protocol based on a new PVSS scheme [16], where the secret key in the protocol is a field element.

Regarding the asynchronous network assumption, works are more limited. Kate and Goldberg were the first to study DKG in an asynchronous communication model. They extended Pedersen's DKG to a partially synchronous network [20]. They require a network of $n \geq 3t+2f+1$ participants, out of which t are controlled by the adversary and thus considered Byzantine and f parties may fail in the crash-stop model. They reduced the broadcast overhead per DKG party from $O(n)$ to $O(1)$ using their constant-sized polynomial commitment scheme. In his work, Tomescu showed how to reduce the size of the final DKG transcript to $O(n)$ by making the parties' contributions aggregatable. They achieved this by making their transcripts publicly-verifiable so that anybody receiving and aggregating transcripts can verify their correctness [32]. They reduced the cost of verifying their transcripts to $O(n/logn)$ whereas prior approaches were $O(n^2)$ [20]. They also showed that Pedersen's DKG is security-preserving with respect to any rekeyable encryption scheme, signature scheme, or VUF scheme where the sharing algorithm is the same as encryption or signing. Kokoris designed an asynchronous DKG (ADKG) scheme with a total communication cost of $O(\kappa n^4)$ and an expected round complexity of $O(n)$ [22]. Abraham et al. proposed an ADKG protocol with a communication cost of $O(\kappa n^3 logn)$ [1]. Gao [14] and Das [9] gave two methods to lower the communication cost of [1] to $O(\kappa n^3)$. Since Abraham et al. uses the PVSS scheme of Gurkan , all three constructions inherit the limitation that the secret key is a group element. The increasing popularity of threshold signatures has led to many DKG implementations [18, 26, 29, 30].

2.2 Related Work on secret sharing

The problem of secret sharing was firstly introduced by A. Shamir in [31], along with the so-called Shamir threshold secret sharing schemes, based on the Lagrangian interpolation. Since the publication of Shamir's work, several other methods have been proposed, such as M. Mignotte [23] and C. Asmuth and J. Bloom [2], which are both based on the Chinese remainder theorem for coprime modules over the ring of integers $\mathbb{Z}$. Chinese remainder theorem is based on a well known property of integers, the so-called Euclidean division, which is not exclusive to $\mathbb{Z}$. Other rings that admit it are called Euclidean Domains. Two of them are highlighted , the ring of univariate polynomials over a field $\mathbb{K}[X]$, and the ring of Gaussian Integers $\mathbb{Z}[i]$. Mignotte secret sharing scheme can be extended to these rings. In the literature we can find successive generalizations of Mignotte's scheme, firstly by S. Iftene to not necessarily coprime integers [19]. Later T. Galibus and G. Matveev [13] provided a version over polynomial rings of one variable and showed the remarkable property that any access structure can be realized by a Mignotte polynomial SSS. Finally, in [27] an extension of Mignotte threshold scheme to the ring of Gaussian Integers is proposed. Gaussian Integers play a role in digital signal processing, cryptography, coding, and many other fields of science and technology [5]. Mignotte's threshold scheme has already been proposed for image sharing, [33], which will be very useful for future work, if biometrics are considered as images and try to merge biometric authentication with threshold secret sharing. Authors in [24], improved the extension of Mignotte SSS to Gaussian Integers proposed in [27], because in the Euclidean division over $\mathbb{Z}[i]$, quotient and remainder are not unique, so they proposed a new version of Mignotte's SSS over $\mathbb{Z}[i]$ that works properly unlike [27].

2.3 Motivation

In this paper, considering the aforementioned related work on the field, the subject of distributed key generation is tackled as well as secret sharing from a different point of view. Most of the research has been made on the security of the schemes, depending on different kind of adversaries and the interactiveness that will take place between the users of a group in order to gather their shares of the cryptographic keys, and use them in a aggregatable way for a threshold digital signature or other cryptographic actions.

When a new member wants to join the communication protocol, or an old member leaves the group or maybe get compromised, various solutions have been proposed also, such as resharing of the keys to all members, refreshing etc. Depending on the network and adversary assumptions these are costly steps that should be taken into consideration. On a synchronous, trusted setup, Newton's interpolation instead of Lagrange's should be considered. The advantage of this , in Shamir's secret sharing scheme, is that there is no need to evaluate the initial polynomial when a new member wishes to join the communication protocol and recalculate the coefficients of a higher degree polynomial, by making use of the incremental interpolation that Newton's method has to offer.

3 SECRET SHARING - VERIFIABLE SECRET SHARING

3.1 Shamir Secret sharing scheme

Shamir's Secret Sharing scheme is an algorithm that was first proposed in 1979 by the renowned Israeli cryptographer Adi Shamir. Shamir's (t, n)-threshold scheme makes use of the property that a polynomial of degree d can be uniquely reconstructed by any set of $d + 1$ points using the Lagrange interpolation. Assume there exist n parties $P_i, i \in [1, n]$ and the threshold is set at $t(1 \leq t \leq n)$:

(1) A 'dealer' generates a secret $s \in \mathbb{F}_p$ and constructs a random polynomial $P(x) = f(x) = \sum_{i=0}^{t} a_i x^i$ in $\mathbb{F}_p$ of degree t and with its constant term set to the secret value $f(0) = f_0 = s$.
(2) The dealer then calculates n shares $s_i = f(i), \forall i = 1, 2.., n$.
(3) The shares are privately 'dealt' to the parties (sharing phase).
(4) Any $t + 1$ parties can use their shares to reconstruct the polynomial, through Lagrange interpolation, and thus reveal the secret (reconstruction phase).

The security of the above scheme relies on the fact that one must trust the dealer to be honest. If the dealer is dishonest, then one has to use a share validation mechanism.

A Verifiable Secret Sharing (VSS) scheme enables parties to verify the shares' consistency, thus ensuring the correct reconstruction of the shared secret. Most of the current secret sharing schemes use Feldman's VSS, which extends the aforementioned Shamir Secret sharing scheme to a verifiable one, by making the dealer broadcast homomorphic commitments to each of the polynomial's coefficients $F_i = g^{a_i}$ [20]. In more depth, as one polynomial

$$F(x) = \prod_{i=0}^{t} F_i^{x^i} = g^{\sum_{i=0}^{t} a_i x^i} = g^{f(x)} \tag{1}$$

4 DISTRIBUTED KEY GENERATION

A DKG enables a set of parties to generate a keypair such that any sufficiently large subset can perform an action that requires the secret key while any smaller subset cannot. To achieve this, a DKG essentially turns each party into a dealer for a verifiable secret sharing (VSS) scheme. This process yields a single collective public key, generated in a distributed manner, with each party keeping a share of the secret key for themselves. No one-round DKG can achieve secrecy because a rushing adversary (an adversary that plays last) can always influence the final distribution [32]. The core idea of the Pedersen (and the Joint-Feldman) protocol is that each party executes Feldman's verifiable secret sharing (VSS) protocol, acting as a dealer to share a randomly chosen secret among all parties.

The protocol's steps work as follows:

(1) Each party P_i generates a secret s_i, a random polynomial $f_i(x) = \sum_{i=0}^{t} a_i x^i$ with degree t and a Feldman commitment to it $F_i(x) = \prod_i g^{a_i x^i}$.
(2) It then calculates its public key share $p_i = g^{s_i}$ and broadcasts a commitment to it $C_i = C(p_i, r_i)$, where r_i is a random string.
(3) When all P_i finished broadcasting, each P_i opens C_i and the shared public key is set to $pk = \prod_{i=1}^{n} p_i$ with the corresponding secret being $s = \sum_{i=1}^{n} s_i$.
(4) P_i broadcasts the commitment to the polynomial and distributes shares s_{ij} of its secret to all other parties P_j (sharing phase).

Pedersen's DKG features a complaint round inside the sharing phase to remove invalid shares but is out of the scope of this paper. The state of the system after the sharing phase is completed as soon as each participating party holds $n - 1$ shares and their secret. For reconstructing the shared secret, all secrets s_i need to be obtained by reconstructing each polynomial $f_j(x)$ for which $t + 1$ points are required by each polynomial. Since each party has a point of each polynomial, any $t + 1$ parties suffice to reconstruct all n secrets, thus revealing the common secret.

5 INTERPOLATION METHODS

In general, polynomial interpolation is used to approximate complicated curves such as the evaluation of natural logarithm or the evaluation of trigonometric functions. It is also used to perform sub-quadratic multiplication and squaring such as Karatsuba multiplication and Toom-Cook multiplication because it has a faster computation time. But in this paper, it will be used to reconstruct a polynomial for secret sharing schemes.

The question of how to reconstruct a smooth function $f(x)$ that agrees with a number of collected sample values does not have a straight answer, particularly since there are more than one ways to accomplish this task. We denote as $x_1, x_2, \ldots, x_N$ for the x-locations where f is being sampled and denote the known value of $f(x)$ at $x = x_1$ as $y_1 = f(x_1)$, at $x = x_2$ as $y_2 = f(x_2)$, etc. The aim is to reconstruct a function $f(x); x \in [a; b]$ such that the plot of f passes through the following points:

$$(x_1, y_1), (x_2, y_2), \cdots, (x_N, y_N) \tag{2}$$

A commonly used approach is to use a properly crafted polynomial function

$$f(x) = P_n(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_{n-1} x^{n-1} + a_n x^n \tag{3}$$

to interpolate the points $(x_0, y_0), \ldots, (x_n, y_n)$.

Two ways of evaluating the polynomial $P_n(x)$, in Lagrange's and Newton's interpolations will be explored.

5.1 Lagrange Interpolation

For a given set of $n + 1$ points $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$ define the Lagrange polynomials of degree-n $L_0(x), L_1(x), \ldots, L_n(x)$ as:

$$L_i(x_j) = \delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \tag{4}$$

Since $L_i(x)$ is a degree-n polynomial, with the n-roots $x_0, x_1, x_2, \ldots, x_{i-1}, x_{i+1}, x_{i+2}, \ldots, x_n$ it must have the form

$$L_i(x) = \alpha_i \prod_{j=0, j\neq 1}^{n} (x - x_j) \tag{5}$$

If the invariant a_i is evaluated from the relation $L_i(x_i) = 1$, then

$$L_i(x) = \prod_{j=0, j\neq i} \frac{(x - x_j)}{(x_i - x_j)}, \quad i = 0, \ldots n \tag{6}$$

Thus, the polynomial $P_n(x)$ is finally calculated

$$P_n(x) = \sum_{i=0}^{n} f(x_i) L_i(x) \tag{7}$$

which is called the Lagrangian form of the interpolated polynomial.

5.2 Newton Interpolation

For a given set of data points $(x_0, y_0), \ldots, (x_n, y_n)$ the polynomial $P_n(x)$ for which $P(x_i) = y_i, \ i = 0, \ldots n$ can be written in the following form

$$P_n(x) = a_0 + a_1 (x - x_0) + a_2 (x - x_0)(x - x_1) + \ldots + a_{n-1}(x - x_0)(x - x_1) \ldots (x - x_{n-1}) \tag{8}$$

where the coefficients $a_0, \ldots, a_n$ can easily be calculated consecutively as

$$\begin{aligned} P(x_0) = y_0 \quad &\rightarrow \quad a_0 = y_0 \\ P(x_1) = y_1 \quad &\rightarrow \quad a_1 = \frac{y_1 - y_0}{x_1 - x_0} \\ &\vdots \end{aligned} \tag{9}$$

The advantage of this representation is that for the interpolation of a new point, only a new coefficient needs to be calculated. Coefficients $a_0, \ldots, a_n$ of the Newton's method are usually calculated through a method called *divided differences.*

Divided differences

A divided difference is a function defined over a set of sequentially indexed centers, e.g.,

$$x_i, x_{i+1}, \ldots, x_{i+j-1}, x_{i+j} \tag{10}$$

The divided difference of these values is denoted by:

$$f\left[x_i, x_{i+1}, \ldots, x_{i+j-1}, x_{i+j}\right] \tag{11}$$

The value of this symbol is defined recursively as follows. For divided differences with one argument

$$f[x_i] \equiv f(x_i) = y_i \tag{12}$$

With two arguments:

$$f[x_i, x_{i+1}] = \frac{f[x_{i+1}] - f[x_i]}{x_{i+1} - x_i} \tag{13}$$

With three:

$$f[x_i, x_{i+1}, x_{i+2}] = \frac{f[x_{i+1}, x_{i+2}] - f[x_i, x_{i+1}]}{x_{i+2} - x_i} \tag{14}$$

With j + 1 arguments:

$$f\left[x_i, x_{i+1}, \ldots, x_{i+j-1}, x_{i+j}\right] = \frac{f\left[x_{i+1}, \ldots, x_{i+j}\right] - f\left[x_i, \ldots, x_{i+j-1}\right]}{x_{i+j} - x_i} \tag{15}$$

The fact that makes divided differences so useful is that $f\left[x_i, \ldots, x_{i+j}\right]$ can be shown to be the coefficient of the highest power of x in a polynomial that interpolates through

$$(x_i, y_i), (x_{i+1}, y_{i+1}), \ldots, (x_{i+j-1}, y_{i+j-1}), (x_{i+j}, y_{i+j}) \tag{16}$$

Remember, the polynomial that interpolates $(x_0, y_0), \ldots, (x_n, y_n)$, which is

$$P_n(x) = \underbrace{P_{n-1}(x)}_{\text{highest power } = x^{n-1}} \quad \underbrace{+ a_n (x - x_0) \ldots (x - x_{n-1})}_{= a_n x^n + \text{ lower powers}} \tag{17}$$

Thus, $a_n = f[x_0, x_1, x_2, \ldots, x_n]$. Or, in other words

$$\begin{aligned} P_n(x) = {} & f[x_0] \\ & + f[x_0, x_1](x - x_0) \\ & + f[x_0, x_1, x_2](x - x_0)(x - x_1) \\ & \vdots \quad f[x_0, x_1, \ldots, x_n](x - x_0) \ldots (x - x_{n-1}) \end{aligned} \tag{18}$$

So, if the divided differences, can be quickly evaluated then $P_n(x)$ has been determined. The resulting polynomials of the two methods are the same. For more information about different kinds of interpolation, readers are referred to [6, 8]

5.3 A Toy example

Alice selects a random polynomial $p(x) = 3x^2 + 5x + 1$, and wants to share the constant term of the polynomial which is $P(0) = 1$. She calculates some values of the polynomial $P(3) = 43,\ P(4) = 69,\ P(5) = 101$, which will be the shares. At first two receivers will be considered, which will take the two shares and reconstruct a polynomial, and later a third person will want to join and take his share, and then all together will reconstruct Alice's initial polynomial, once with Lagrange and then with Newton's method.

5.3.1 Lagrange's Method. Two people take their shares $P(3) = 43,\ P(4) = 69$, and try to create a polynomial, as below

$$\begin{aligned}
P(x) &= \frac{x-4}{x_1-x_2}\cdot y_1 + \frac{x-x_1}{x_2-x_1}\cdot y_2\\
&= \frac{x-4}{3-4}\cdot 43 + \frac{x-3}{4-3}\cdot 69\\
&= \frac{x-4}{-1}\cdot 43 + \frac{x-3}{1}\cdot 69\\
&= -43\cdot(x-4) + (x-3)\cdot 69\\
&= -43x + 172 + 69x - 207\\
&= 26x - 35
\end{aligned} \tag{19}$$

As it can be seen, two shares are not enough to reconstruct Alice's initial polynomial, which has degree 2, as they need $t+1$ shares, where t is the threshold, which in our situation is $t = 2$ and in applications is the same as the degree of the initial polynomial. Now a third person joins the group and takes its share which is $P(5) = 101$, so with three shares now they try to estimate the intitial polynomial. First, the Lagrange polynomials :

$$\begin{aligned}
L_3 &= \frac{(x-4)(x-5)}{(3-4)(3-5)} = \frac{(x-4)(x-5)}{2} = \frac{1}{2}\cdot(x^2 - 5x - 4x + 20)\\
&= \frac{1}{2}(x^2 - 9x + 20)\\
L_4 &= \frac{(x-3)(x-5)}{(4-3)(4-5)} = \frac{(x-3)(x-5)}{-1} = -(x-3)\cdot(x-5)\\
&= (-x^2 + 8x - 15)\\
L_5 &= \frac{(x-3)(x-4)}{(5-3)(5-4)} = \frac{(x-3)(x-4)}{2} = \frac{1}{2}(x-3)\cdot(x-4)\\
&= \frac{1}{2}(x^2 - 7x + 12)
\end{aligned} \tag{20}$$

And the reconstruction, after the evaluation of the L_i's is

$$\begin{aligned}
P(x) &= \sum_{i=3}^{5} f(x_i)L_i(x) =\\
&= \frac{43}{2}\cdot(x^2 - 9x + 20) + 69\cdot(-x^2 + 8x - 15) + \frac{101}{2}\cdot(x^2 - 7x + 12)\\
&= \vdots\\
&= 3x^2 + 5x + 1
\end{aligned} \tag{21}$$

We observe that three shares are enough to estimate Alice's initial polynomial of degree 2, as expected.

5.3.2 Newton's Method. The same two people, will try to reconstruct a polynomial with their shares $P(3) = 43,\ P(4) = 69$, but with Newton's method now, as below

$$\begin{aligned}
f[x_0] &= 43\\
f[x_0, x_1] &= \frac{y_1 - y_0}{x_1 - x_0} = \frac{69-43}{4-3} = 26
\end{aligned} \tag{22}$$

Thus:

$$\begin{aligned}
P(x) = f[x_0] + f[x_0, x_1]\cdot(x - x_0) &= 43 + 26\cdot(x-3)\\
&= 43 + 26x - 78\\
&= 26x - 35
\end{aligned} \tag{23}$$

Same result as before, e.g., the same polynomial from the two shares, and again they are not able to reconstruct the second degree polynomial because they need an extra share. A third person joins the group again with the share $P(5) = 101$. In a similar way, they try to reconstruct the initial polynomial. The difference is that they do not need to recalculate the whole L_i's, but only

$$f[x_1, x_2] = \frac{y_2 - y_1}{x_2 - x_1} = \frac{101-69}{5-4} = 32 \tag{24}$$

$$f[x_0, x_1, x_2] = \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0} = \frac{32-26}{5-3} = 3 \tag{25}$$

So the polynomial will be

$$\begin{aligned}
P(x) &= f[x_0] + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1)\\
&= 26x - 35 + 3(x - x_0)\cdot(x - x_1)\\
&= 26x - 35 + 3(x^2 - 7x + 12)\\
&= 3x^2 + 5x + 1
\end{aligned} \tag{26}$$

5.4 Comparison of polynomial interpolation methods

Starting with an examination of Lagrange's method, it is apparent that determining $P_n(x)$ is very easy. The method of writing a formula for $P_n(x)$ without solving any systems has been shown. However, in order to write $P_n(x) = a_0 + a_1x + \ldots + a_nx^n$ the cost of evaluating the a_i's would be high. Each $L_i(x)$ would need to be expanded, leading to $O(n^2)$ operations for each $L_i(x)$ implying $O(n^3)$ operations for $P_n(x)$. The cost of evaluating $P_n(x)$ for an arbitrary x is significant. The a_i's do not need to be computed beforehand, as long as the evaluation of $P_n(x)$ is carried out at a selected few locations. For each $L_i(x)$ the evaluation requires n subtractions and n multiplications, implying a total of $O(n^2)$ operations, which is better than $O(n^3)$ for computing the a_i's. Lastly, it does not accommodate incremental interpolation, which is a crucial disadvantage on today's asynchronous networks.

Passing on to the evaluation of Newton's method, the cost of computing $P_n(x)$ is $O(n^2)$ and the cost of evaluating $P_n(x)$ for an arbitrary x is $O(n)$. This can be accelerated (similar to Horner's method) using the recursive scheme defined above. Its most important aspect is that it allows for incremental interpolation without recalculating the initial polynomial and its coefficients.

Shamir's secret sharing scheme is based on the Lagrangian interpolated polynomial. An advantage of Lagrange over Newton interpolation, is that the quantities that have to be computed in $O(n^2)$ operations do not depend on the data f_j, whereas Newton interpolation requires the recomputation of the divided difference for each new function. In the Newton formula, the divided differences do have such a dependence [3]. Another advantage is that it does not depend on the order in which the nodes are arranged.

However, besides the advantages, when the degree of the polynomial needs to be increased, a recalculation of the values of the polynomial at the previous known points and the corresponding addition of one more $(n + 1)$-th point is required. 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. So, 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 on multiparty computation schemes, and especially during authentication procedures, where people can enter or leave from the group at any time, go offline, or even get compromised and need to be discarded from the protocol.

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

$$P_n(x) = f[x_0] + \sum_{i=1}^{n} f[x_0, \ldots, x_i] \prod_{i=0}^{n} (x - x_i) \tag{27}$$

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 only adding the $(n + 1)$-th interpolation node, and accordingly, calculating the corresponding coefficient. The polynomial values at previously known points do not change, so there is no need to recalculate the previous coefficients; one only needs to add a new one. The recalculated Newton interpolation polynomial of degree $n + 1$ can be, thus, represented as

$$P_{n+1}(x) = P_n(x) + f[x_0, \ldots, x_{n+1}] \prod_{i=0}^{n} (x - x_i) \tag{28}$$

and this is the main reason Newton's interpolation is proposed in this paper. Furthermore, Newton's interpolation has the ability to accommodate vector or matrix interpolation from scalar data, better than Lagrange in terms of efficiency, which is very useful on threshold biometric authentication, and incremental interpolation on distributed key generation may have a favorable impact on performance level [21].

6 CONCLUSION-FUTURE WORK

It has been shown how Shamir's secret sharing scheme works and a brief explanation of Feldman's VSS has been provided. After analyzing the two methods of interpolating a polynomial, Langrange's and Newton's, our future work will be to implement Newton's interpolation method to existing DKG protocols and test the performance.

ACKNOWLEDGMENTS

This work was funded by the European Commission as part of the SealedGRID Project (H2020-MSCA-RISE-2017, Project ID: 777996).

REFERENCES

[1] Ittai Abraham, Philipp Jovanovic, Mary Maller, Sarah Meiklejohn, Gilad Stern, and Alin Tomescu. 2021. Reaching consensus for asynchronous distributed key generation. In *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing*. 363–373.

[2] Charles Asmuth and John Bloom. 1983. A modular approach to key safeguarding. *IEEE transactions on information theory* 29, 2 (1983), 208–210.

[3] Jean-Paul Berrut and Lloyd N Trefethen. 2004. Barycentric lagrange interpolation. *SIAM review* 46, 3 (2004), 501–517.

[4] Sergey Bezzateev, Vadim Davydov, and Aleksandr Ometov. 2020. On Secret Sharing with Newton's Polynomial for Multi-Factor Authentication. *Cryptography* 4, 4 (2020), 34.

[5] Richard E Blahut. 2012. *Algebraic methods for signal processing and communications coding*. Springer Science & Business Media.

[6] Roland Bulirsch, Josef Stoer, and J Stoer. 2002. *Introduction to numerical analysis*. Vol. 3. Springer.

[7] Ran Canetti, Rosario Gennaro, Stanisław Jarecki, Hugo Krawczyk, and Tal Rabin. 1999. Adaptive security for threshold cryptosystems. In *Annual International Cryptology Conference*. Springer, 98–116.

[8] Samuel Daniel Conte and Carl De Boor. 2017. *Elementary numerical analysis: an algorithmic approach*. SIAM.

[9] Sourav Das, Zhuolun Xiang, and Ling Ren. 2021. Asynchronous data dissemination and its applications. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. 2705–2721.

[10] Sourav Das, Tom Yurek, Zhuolun Xiang, Andrew Miller, Lefteris Kokoris-Kogias, and Ling Ren. 2021. Practical asynchronous distributed key generation. *Cryptology ePrint Archive* (2021).

[11] Information Technology Laboratory Computer Security Division. 2010 (accessed May 2, 2022). Multi-party threshold cryptography: CSRC. https://csrc.nist.gov/Projects/threshold-cryptography.

[12] Pierre-Alain Fouque and Jacques Stern. 2001. One round threshold discrete-log key generation without private channels. In *International Workshop on Public Key Cryptography*. Springer, 300–316.

[13] Tatyana Galibus and Genadii Matveev. 2007. Generalized mignotte's sequences over polynomial rings. *Electronic Notes in Theoretical Computer Science* 186 (2007), 43–48.

[14] Yingzi Gao, Yuan Lu, Zhenliang Lu, Qiang Tang, Jing Xu, and Zhenfeng Zhang. 2021. Efficient asynchronous byzantine agreement without private setups. *arXiv preprint arXiv:2106.07831* (2021).

[15] Rosario Gennaro, Stanisław Jarecki, Hugo Krawczyk, and Tal Rabin. 1999. Secure distributed key generation for discrete-log based cryptosystems. In *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 295–310.

[16] Jens Groth. 2021. Non-interactive distributed key generation and key resharing. *Cryptology ePrint Archive* (2021).

[17] Kobi Gurkan, Philipp Jovanovic, Mary Maller, Sarah Meiklejohn, Gilad Stern, and Alin Tomescu. 2021. Aggregatable distributed key generation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 147–176.

[18] Timo Hanke, Mahnush Movahedi, and Dominic Williams. 2018. Dfinity technology overview series, consensus system. *arXiv preprint arXiv:1805.04548* (2018).

[19] Sorin Iftene. 2007. General secret sharing based on the chinese remainder theorem with applications in e-voting. *Electronic Notes in Theoretical Computer Science* 186 (2007), 67–84.

[20] Aniket Kate, Yizhou Huang, and Ian Goldberg. 2012. Distributed key generation in the wild. *Cryptology ePrint Archive* (2012).

[21] Noam Kogan and Tamir Tassa. 2006. Improved efficiency for revocation schemes via Newton interpolation. *ACM Transactions on Information and System Security (TISSEC)* 9, 4 (2006), 461–486.

[22] Eleftherios Kokoris Kogias, Dahlia Malkhi, and Alexander Spiegelman. 2020. Asynchronous Distributed Key Generation for Computationally-Secure Randomness, Consensus, and Threshold Signatures.. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. 1751–1767.

[23] Maurice Mignotte. 1982. How to share a secret. In *Workshop on cryptography*. Springer, 371–375.

[24] Diego Munuera-Merayo. 2021. On Mignotte secret sharing schemes over Gaussian integers. *arXiv preprint arXiv:2104.06361* (2021).

[25] Wafa Neji, Kaouther Blibech, and Narjes Ben Rajeb. 2016. Distributed key generation protocol with a new complaint management strategy. *Security and communication networks* 9, 17 (2016), 4585–4595.

[26] Orbs Network. 2018. DKG for BLS threshold signature scheme on the EVM using solidity.

[27] Ibrahim Ozbek, Fatih Temiz, and SİAP İrfan. 2019. A generalization of the Mignotte's scheme over Euclidean domains and applications to secret image sharing. *Journal of Algebra Combinatorics Discrete Structures and Applications* 6, 3 (2019), 147–161.

[28] Torben Pryds Pedersen. 1991. A threshold cryptosystem without a trusted party. In *Workshop on the Theory and Application of of Cryptographic Techniques*. Springer, 522–526.

[29] Philipp Schindler. 2020. Ethereum-based Distributed Key Generation Protocol.

[30] Philipp Schindler, Aljosha Judmayer, Nicholas Stifter, and Edgar Weippl. 2019. Ethdkg: Distributed key generation with ethereum smart contracts. *Cryptology ePrint Archive* (2019).

[31] Adi Shamir. 1979. How to share a secret. *Commun. ACM* 22, 11 (1979), 612–613.

[32] Alin Tomescu, Robert Chen, Yiming Zheng, Ittai Abraham, Benny Pinkas, Guy Golan Gueta, and Srinivas Devadas. 2020. Towards scalable threshold cryptosystems. In *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 877–893.

[33] Güzin Ulutaş, Mustafa Ulutaş, and Vasif Nabiyev. 2011. Secret sharing scheme based on mignotte's scheme. In *2011 IEEE 19th Signal Processing and Communications Applications Conference (SIU)*. IEEE, 291–294.