A tour of Verifiable Secret Sharing schemes and Distributed Key Generation protocols.

Ignacio Manzur Tomasini

Follow

Nethermind.eth

--

1

Listen

Share

by Ignacio Manzur, Marc Graczyk and Albert Garreta. Special thanks to Ahmet Ramazan Agirtas for advice and valuable comments throughout the writing of this post, and to Michal Zajac for reviewing.

This article introduces Distributed Key Generation (DKG) protocols and one of its fundamental building blocks: Verifiable Secret Sharing (VSS) schemes. We begin our exploration with the cryptographic primitives that make Distributed Validator Technology (DVT) possible. While an expansion of our previous article on DVT, almost the entirety of this post can be read without any previous knowledge of DVT.

The present article can also be read as a continuation of Secret Sharing For Sharing Mnemonics,

[Part I

](/nethermind-eth/using-shamirs-secret-sharing-to-share-mnemonics-c40429835117)and

[Part 2

](/nethermind-eth/secret-sharing-for-sharing-mnemonics-part-ii-d76dc1581681).

Table of contents:

Introduction

What do Distributed Key Generation (DKG) protocols enable? At a high level, they allow for the distribution of power (in the form of a key) among a group of independent parties that do not trust each other. Only when a quorum has been achieved, the power, i.e. the key, can be used.

More concretely, DKG protocols allow for the generation and distribution of secret key fragments, or shares, among several parties. A crucial point is that the key is created during the protocol execution and that no single party is privy to the key before, during, or after such execution, only their individual key shares. Moreover, nobody apart from the involved parties knows the key either, and the protocol requires no intervention from trusted third parties.

Consider the following example: a company is run by a group of directors who do not trust each other and hence are uncomfortable with another director having access to the secret signing key of the company. To solve this issue, the group of directors can run a DKG protocol to generate a key, giving each director access to only part of it. This means that to successfully sign something, a quorum of directors must be established.

Notice the importance of the key being generated during the protocol execution: it would be undesirable if the secret key was known to some party and distributed among the directors using, for example, an out-of-the-box version of Shamir's Secret

Sharing scheme.

As described in our previous post, another use-case of DKG protocols can be found in DVT solutions, where a group of operators can create and distribute shares of an Ethereum (or of another blockchain) secret validator signing key. This, in combination with consensus protocols and Threshold Signature Schemes (TSS), allows the group of operators to run an Ethereum validator without the need to trust each other.

Adding some formality

Let us set some notations, definitions, and terminology. We let n

be the number of parties involved in the DKG protocol, and we let t

be the number of Byzantine

parties

among them. A party is called Byzantine

if it is malfunctioning (e.g., its computer is off) or actively trying to sabotage the protocol.

Throughout this article, we assume t

$< n$

$/2$, i.e. we assume that less than half of the parties are Byzantine.

In a DKG protocol, a group of n

parties $P_1, \ldots, P_n$

interacts with each other, through secure and authenticated communication channels, to generate

a key pair $(sk, pk$

) — a secret key and a public key, respectively. They create n

secret key shares

$sk_1, \ldots, sk_n$

, so that the party P_i

owns the share sk_i

at the end of the protocol, for each $i=1, \ldots, n$.

In fact, only the non-Byzantine parties are guaranteed to end up with a secret key share.

A DKG protocol is usually designed with a key recovery

mechanism that allows any $t+1$

parties to discover the secret key sk

. This procedure uses the secret shares of the $t+1$

parties involved in it, hence the name “key shares” given to the keys $sk_1, \ldots, sk_n$

.

We remark that many applications do not use the key recovery mechanism. Indeed, running it reveals the secret key

$sk$to a set of parties, which may defeat the purpose of using a DKG protocol in the first place. For example, it is common for Threshold Signature Schemes to use a DKG protocol to create key shares

sk_i, which are later combined to sign data with the secret key

skwithout the need of

skever being revealed to any party. We will discuss this in detail in an upcoming post of this series.

The following properties are required of a DKG protocol.

- Correctness

: If no more than t

parties are Byzantine, then, roughly speaking, the protocol works as intended (in cryptography, the term “correctness” always refers, essentially, to a protocol working as expected).

- Secrecy

: If again, no more than t

parties are Byzantine, then no information on sk

can be learned during the key generation phase. This is slightly vague, but a formal treatment of this topic is out of the scope of the article. In short, it means that no party and no observer learn sk

during or after the execution of the protocol. It also means that no “partial information” can be learned about sk

. For example, it must be infeasible to deduce, say, the last bit of sk

. More than that, it must be infeasible to conclude that the last bit of sk

is, say, “twice more likely to be 1

than 0

” (more generally, we want sk

and pk

to be uniformly distributed among the set of all possible keys).

We have omitted technical details and definitions. We refer toGennaro et al. (2006) for more formal treatment.

Overview of protocols

The two DKG protocols that are most widely used nowadays are the so-called [Joint Feldman DKG

](https://link.springer.com/chapter/10.1007/3-540-46416-6_47) (JF DKG) protocol (1991) — one of the first such protocols to be described — and [Gennaro et al.’s DKG

](https://link.springer.com/article/10.1007/s00145-006-0347-3) protocol (2006). Many DKG protocols are variations of these two.

The two protocols are still used concurrently today. One reason is that, in choosing one or the other, there is a tradeoff between efficiency and security. On the one hand, JF DKG is more efficient in computation, storage, and communication complexity. On the other hand, Gennaro et al.’s DKG protocol is more secure and resolves a known attack on the JF DKG protocol, which prevents the public key pk

from being uniformly distributed. We will discuss this in more detail later.

The two protocols rely heavily on Verifiable Secret Sharing (VSS) schemes,

which are extensions of Secret Sharing Schemes (SSS). More precisely, VSS schemes are SSS schemes with the extra functionality of allowing the parties to verify that the secret key shares they received are consistent with the secret being shared. Why is such a mechanism important? Recall that in Shamir’s Secret Sharing scheme, a corrupted dealer (the party sharing its secret) may send some key shares that do not correspond to the ones an honest dealer following the protocol would send. Even worse, during key reconstruction dishonest parties may contribute incorrect shares to gain information from the shares submitted by honest parties.

VSS schemes provide protection against such attacks by leveraging cryptographic commitments

, a type of binding procedure in which a party publishes information that ties it to a certain value v

while keeping it hidden from others, with the ability to reveal v

later. The idea is that a party that commits to a specific value cannot alter it after that.

Next, we will go through both the Joint Feldman protocol and Gennaro et al.’s protocol, as well as the VSS schemes that underly them. Let’s begin by reviewing some mathematical concepts and establishing notation.

Math preliminaries and notation

Here we introduce basic math notations and concepts that are used when describing the DKG protocols.

Finite fields and discrete logarithms

We let p

and q

be two large primes for the rest of the article. We assume q

divides p-1

, i.e. we assume that p-1

is a multiple of q

. By $\mathbb{F}_p$

we denote the finite field

with p

elements. As a set, this is

Addition and multiplication in $\mathbb{F}_p$

are performed modulo p

.

We fix an element g

$\in \mathbb{F}_p$

of “order q

”, meaning that q

is the smallest positive integer x

such that g^x

$\equiv 1 (\mathrm{mod}\ p$

).

Technical point (not essential for understanding this article)

Hence, g

generates a cyclic group $\mathbb{G}$ of order q. We assume that the [discrete logarithm problem

](https://en.wikipedia.org/wiki/Discrete_logarithm) is hard in $\mathbb{G}$. Roughly, this means that given $h \in \mathbb{F}_p^*$such that $g^k=h$ for some $k \in \{$

0,…, q-1}it is infeasible to find k.

⚠ In practice, $\mathbb{G}$ is taken to be the group generated by a point of an elliptic curve over $\mathbb{F}_p$. For simplicity, we will stick to working directly on $\mathbb{F}_p$.

Degree of polynomials and their defining points

Observe the following phenomenon: given two points (x,y)

and (x’, y’)

in a plane, there is a unique line that passes through these two points. In particular, if two lines pass through the same two distinct points, then they are the same line.

Now, notice that a line is defined by a polynomial of degree 1

, that is a function f

of the form $f(X)=aX+b$

for some constants a, b

.

Now, what happens when we turn our attention to degree 2

polynomials, i.e. functions of the form $f(X) = aX^2 + bX+ c$

? Well, in this case, for any three points (x_0, y_0), (x_1, y_1), (x_2, y_2)

there exists a unique degree 2

polynomial f(X)

that traverses these points, i.e. $f(x_i) =y_i$

for i=0,1,2

.

Again, it can be shown that any two degree-2

polynomials agreeing when evaluated at three distinct elements x_0, x_1, and x_2

must necessarily be the same polynomial.

This trend generalizes to polynomials of arbitrary degree and to any field:

Given

r+1pairs of elements

(x_0, y_0), …, $(x_r, y_r)$with

$x_0,…, x_r, y_0, …, y_r \in \mathbb{F}_p$(or in an arbitrary field $\mathbb{K}$), there exists a unique polynomial

f(X)with coefficients in $\mathbb{F}_p$of degree

r such that

$f(x_i)=y_i$for all

i=0,…, r. The process by which such

f(X)is found is called

[polynomial interpolation

](https://en.wikipedia.org/wiki/Polynomial_interpolation).

In particular, given two polynomials

f(X),g(X) of degree

r with coefficients in

$\mathbb{F}_p$, if there are

r+1 elements $x_0, …, x_r \in \mathbb{F}_p$ such that

$f(x_i) = g(x_i)$ for all

i=0,…,r then

f and

g are the same polynomial.

This property has far-reaching consequences in cryptography. Indeed, it is a fundamental building block of secret sharing schemes, Reed-Solomon codes, and STARKs, to give a few examples.

As usual, we use $\mathbb{F}_p$

[X

] to denote the set of polynomials on a variable X

, with coefficients in $\mathbb{F}_p$

.

Feldman’s VSS scheme and Joint Feldman DKG protocol

We are now ready to start diving into the Joint Feldman DKG protocol. As already mentioned, it relies on a particular Verifiable Secret Sharing (VSS) scheme called Feldman’s VSS scheme.

Feldman’s Verifiable Secret Sharing scheme

A dealer wants to share a secret among n

parties $P_1,\ldots, P_n$

in a way that any subset of at least t+1

parties can reconstruct it. Recall that t < n/2

is the maximum number of Byzantine parties, and we want to allow the parties to check the information sent by the dealer and other parties for consistency.

VSS schemes provide a solution satisfying these requirements in three steps, namely:

- The broadcast step.
- The verification step.
- The reconstruction step.

Assume that the secret σ

can be represented as an element of $\mathbb{F}q$

. The dealer chooses, and keeps secret, a random polynomial of degree t > 0

from $\mathbb{F}q[X]$,

say

such that $f(0)=a_0=\sigma$

is the shared secret.

Broadcast step

The dealer now computes elements $s_1, \ldots, s_n$

of the form

We call the elements$s_1, \ldots, s_n$

shares.

Next, the dealer sends s_i

privately to P_i

for each i=1,…,n

. Note how we start at i=1

and not at i=0

since f(0)=σ

is our secret.

The protocol is the same as Shamir's Secret Sharing scheme up to this point. We mentioned before that such a scheme is vulnerable to attacks from Byzantine parties (including the dealer itself) that do not send shares corresponding to an honest execution of the protocol. So how can a party P_i

be sure that the share received is the result of the dealer following the protocol honestly?

The idea is to make the shares s_i

verifiable

. To this end, we make the dealer commit

the coefficients of the polynomial f

. That is, the dealer publishes the values:

If the values published are the claimed ones, we now have:

The values $v_0, \ldots, v_t$

are called commitments

for the coefficients of f

. The figure below shows the broadcast step for the case n=5

and t=2

.

Verification step

To verify its share s_i

, the party P_i

computes (g

to the power s_i

), then (v_0

- (v_1

to the power i) … * (v_t

to the power i

^{t})), and finally it checks that these two values are equal modulo p

.

If P_i

finds that the two values disagree, then P_i

publishes a complaint

against the dealer. The dealer then publishes s_i

, and each party verifies whether

An honest party P_j

considers the dealer disqualified

if either more than t

complaints are published against the dealer, or the previous check fails. Since any set of more than t

parties contains an honest party, P_j

disqualifies the dealer if, and only if, the dealer has been dishonest.

Key reconstruction step

At the end of the protocol, either the dealer has been disqualified, or the honest parties hold at least t+1

verified shares. Hence, the honest parties can reconstruct the original secret using polynomial interpolation (we will revisit this argument with further details while describing the JF DKG protocol below).

The same commitments that the dealer publishes are used to detect incorrect shares at reconstruction time. The dishonest (i.e., Byzantine) parties cannot hold more than t

shares (because t

is the number of corrupted parties). It can be argued that, in this situation, these parties cannot learn anything about the secret during the broadcast phase.

This completes our description of Feldman's VSS scheme. Notice how this scheme is an extension of Shamir's Secret Sharing method. Indeed, the scheme is the same as Shamir's, with the added functionality of allowing the parties to verify their shares' validity, and filter out any incorrect shares submitted by malicious parties during the reconstruction step.

The Joint-Feldman DKG protocol

T.P. Pedersen proposed this protocol in 1991. The protocol consists mainly of n

parallel applications of Feldman's VSS scheme

where each party P_i

is the dealer in one of the protocols and a receiver in the rest. This duality is characteristic of a DKG protocol.

To generate the public and secret keys sk, pk

, as well as the shares of sk

, each party P_i

chooses a random polynomial

over $\mathbb{F}q$

and takes$f_i(0)$

as its secret (notice thus that P_i

's secret $f_i(0)$

is randomly chosen), and then performs Feldman's VSS scheme with all other parties, P_i

being the dealer. As a result, there are n

Feldman's VSS schemes running in parallel, one per party.

In the figure below, we illustrate which pieces of information are sent from one party to another during this first step, taking n=3

. We also indicate what information is known to each party by the end of this step. The figure does not take into account the values committed by each party during the execution of the schemes, which are all public. All values are reduced modulo q

but we omit the "mod q

" appendices in the figure for readability purposes.

Note that some parties may get disqualified as a dealer. For simplicity, let us assume that the non-disqualified parties are P_1, …, P_r and that the disqualified parties are the ones indexed from r+1

to n

. The public key pk

is then computed as

while the secret key is defined as

Note that pk

can be computed by any party since the elements in the product in (3)

are made public (as commitments) during the execution of the n

parallel Feldman's VSS schemes. On the other hand, observe that $f_i(0)$

is known only to party P_i

(for eachi=1, …, r

), and so individual parties have no obvious way of computing sk

(one must prove that, under appropriate assumptions, this is the case). Next, each honest party P_j

computes its secret key $sharesk_j$

as

P_j

knows all the information needed to compute sk_j

: it

received the element $f_1(j)$

during Feldman's VSS scheme with P_1

as the dealer, it received$f_2(j)$

during Feldman's VSS with P_2

as the dealer, and so on. See the figure above. Finally, we define

Note that $A_0=pk$

. We call A_k

the k

-th public verification value.

As its name suggests, it is a public value, since each one of the factors in the product is made public during an execution of Feldman's VSS.

How to reconstruct sk

from the secret key shares

Next, we describe how any set of at least t+1

parties can reconstruct the secret key created during the JF DKG protocol. Recall, however, that such a mechanism is not used in the context of DVT, or if sk

is to be used in a Threshold Signature Scheme (TSS) involving the parties.

The method follows standard interpolation arguments with an extra verification step. Define $h(X):= f_1(X) + … +f_r(X)$

. Then for each j=1, …, r

, we have that $sk_j=h(j)$

.

With overwhelming probability, h

is of degree exactly t

and we omit the proof. Note that $h(0) = f_1(0) + \ldots + f_r(0)$

is the secret key sk

, modulo p

.

Since h

has degree t

, any t+1

correct shares among $sk_1, \ldots, sk_r$

will suffice to reconstruct the secret sk

by using polynomial interpolation. Say that these t+1

shares are $sk_1, \ldots, sk_{t+1}$

. We have$h(1)= sk_1$

, … , $h(t+1)= sk_{t+1}$

. As mentioned earlier, a unique degree t

polynomial satisfying such equalities does exist, and using a degree t

polynomial interpolation on the points $(1, sk_1), \ldots , (t+1, sk_{t+1})$

will compute the polynomial h

. The secret key sk

can be obtained by computing h(0)

modulo q,

as usual. It can also be argued that any collection of strictly less than t+1

shares leaks no information about sk

.

The public verification values allow honest parties to check that the shares submitted by the other parties at the reconstruction time are correct. Say that P_i

wants to verify P_j

’s share sk_j

. Observe that:

Hence, P_i

computes the quantity on the left-hand side and verifies that modulo p

, this value equals g

raised to the power of the share sent by P_j

. If the two values agree, then the share sent by P_j

is correct. Otherwise, the share is not the result of an honest execution of the protocol, and the reconstruction stops. Then P_j

is eliminated from this reconstruction. This ensures that the collection of at least t+1

shares used to reconstruct the secret sk

will, in fact, successfully reconstruct it.

A vulnerability in the Joint Feldman DKG protocol

Gennaro et al. (2006) showed that an adversary could influence the distribution of the public key, making it non-uniform, by using the complaint system against itself. Just by corrupting two parties, the adversary can control the last bit of pk

. This, in turn, leaks information about the secret key sk

. For example, it is a fact that about half of the powers of g

(when working in $\mathbb{F}_p$

*) end with the bit 0

, and the other half end with the bit 1

. In this case, because the relation $pk \equiv g^{sk}$

(mod p

) holds (check this!), knowing the last bit of the public key allows for discarding half of all possible secret keys.

This implies that the JF DKG protocol cannot be used as a generic secure DKG protocol. As mentioned, DKG protocols are executed in many use-cases to distribute key shares for later use in a Threshold Signature Scheme, such as BLS. Notably, the key reconstruction step is never supposed to be performed.

It was shown recently by Gurkan et al. (2021) that the JF DKG protocol is security-preserving

when used to distribute key shares for the BLS signature scheme. This means that an adversary that can break the signature scheme by forging a signature after participating in the DKG protocol, can also break the original signature scheme. In other words, using the JF DKG protocol to create and distribute key shares for the BLS scheme is as secure as the BLS scheme itself

(and the same is true for any rekeyable

encryption scheme).

Gennaro et al. proposed a protocol that does not present the above vulnerability.

Pedersen’s VSS scheme and Gennaro et al.’s DKG protocol

Gennaro et al.’s DKG protocol (2006) presents an alternative to the JF DKG protocol by resolving the vulnerability pointed out above. It relies on a VSS scheme called Pedersen’s VSS, which, similarly to Feldman’s VSS scheme, is also an extension of Shamir’s Secret Sharing scheme. Its advantage over Feldman’s VSS scheme is the so-called perfect secrecy

of the shared secret. Roughly speaking, the information the adversary learns of the secret is independent of the secret being shared. Such a VSS scheme is called information-theoretically secure

. Alternatively, information-theoretic security is also characterized as resistance against computationally unbounded (i.e., having “infinite” computing power) attackers. The price paid for this improvement is an increased overall computational complexity of the protocol

.

Gennaro et al.’s DKG protocol uses an additional

parameter h

from the subgroup generated by g

.

The core structure of the two protocols (Gennaro et al.’s and Joint Feldman’s) are the same, therefore, we will only emphasize the relevant differences.

Pedersen's VSS scheme

Broadcast step

This time the dealer generates two random degree t

polynomials $f_1(X)$

and $f_2(X)$

with$f_1(0) = \sigma$

being the dealer's secret. Explicitly : $f_1(X) = a_t X^t + \ldots + a_1 X + a_0$

and $f_2(X) = b_t X^t + \ldots + b_1 X + b_0$

. The dealer then privately sends to each party P_i

(for i=1, …, n

) the elements

Such elements are called the secret key shares of P_i

. The commitments are now:

Verification step

Each party P_i

performs the following verification check, which will hold if the dealer has been honest:

Besides that, the verification process is the same: the issuing of complaints, the dealer publishing the conflicting shares, and the eventual disqualifying of the dealer.

Key recovery step

This step is almost identical to the reconstruction step in Feldman's VSS scheme: if the dealer is not disqualified, any t +1

honest parties can use their shares to reconstruct the secret. One may also use the commitments to verify the revealed shares s_i

, in which case each party must also reveal $\check{s}_i$

.

Gennaro et al.'s DKG protocol

For the most part, Gennaro et al.'s DKG protocol proceeds in the same way as the JF DKG protocol, using Pedersen's VSS scheme instead of Feldman's. Next, we outline some of the main differences. As in the JF DKG protocol, the public key is defined as

However, if one just follows the JF DKG protocol using Pedersen's VSS scheme instead of Feldman's, one sees that pk

cannot be computed by all parties, since the values

are never committed. Indeed, as we have seen, Pedersen's VSS scheme commits values of the form

For this reason, Gennaro et al.'s DKG protocol adds an extra step where each party P_i

also commits the values

At this point, we need to look out for parties that have been honest throughout the protocol, except at the point of committing the values appearing in (4).

To this end, for each party P_i

, every other party P_j

verifies whether or not the following holds:

where s_{ij} is the share of P_i

's secret that P_i

sent to P_j

. We will leave you with this as an exercise: if P_i

has been honest, then the equation holds :)

Now, for each party P_i

that fails this check, but that passed all previous checks, the rest of the parties P_j

uses polynomial interpolation to recover P_i

's secret, namely the value a_{i0}, and then computes v_{i0}. This can be done because there are at least t+1

honest parties, and each such party P_j

knows the value$f_i(j)$

(mod p

).

The increased commitment size, as well as the additional verification and reconstruction steps, make this protocol more expensive (in terms of storage, computation, and communication) than its JF DKG counterpart.

Other directions to explore

A line of DKG protocols we have not explored uses so-called bilinear pairings on cyclic elliptic curve groups. The intricacies of these are beyond the scope of the post, and we refer to this protocol due to Kate and Goldberg (2007) for further details.

In the VSS schemes we have seen, notice how the dealer needs to compute as many commitments as the degree of the polynomials chosen plus one. This means that the number of commitments is roughly n/2

per party, or, more technically, $\mathcal{O}$(n)

. This is because we operate under the assumption of t < n/2

Byzantine parties, and so in theory choosing polynomials of degree t

should suffice. But in general, one cannot know in advance the exact number t.

For maximum security and optimal efficiency, one should therefore choose polynomials of degree ⌊ n/2

⌋ (the “floor” of n/2)

. The eVSS (2010) and AMT (2020) DKG protocols address this efficiency issue, having a constant number of commitments per party, no matter the value of n

(i.e., the number of commitments is reduced from $\mathcal{O}$(n)

to $\mathcal{O}$(1)

). This is done by combining bilinear pairings on finite groups with the so-called KZG polynomial commitment scheme.

Recent protocols, such as this protocol due to Groth (2021), remove the interaction between the parties. Each dealer creates and publishes dealings (which contain shares and other commitment information) without interacting with other participants. Each receiver can verify whether a dealing is correct without interaction with other participants. To this end, non-interactive zero-knowledge proofs (NIZKs) are used to ensure that each dealing is correct. Since anybody can verify the NIZK proof, this means the dealings are publicly verifiable, and everybody will agree on whether a dealing is valid or not.

Conclusion

This is the second post in our series explaining Distributed Validator Technology. We have introduced the two most common DKG protocols, which are a central cryptographic primitive of DVT and Threshold Signature Schemes, and we have shown how they are built on top of the concept of Verifiable Secret Sharing. Furthermore, we have briefly referenced more sophisticated DKG protocols.

Stay tuned for the following posts in the series!

- Post 3

(the next post): Consensus protocols

- Post 4

: Threshold Signature Schemes

References

- Villalobos, I., Garreta, A., Sorting out Distributed Validator Technology

(first post in this series):https://medium.com/nethermind-eth/sorting-out-distributed-validator-technology-a6f8ca1bbce3

- Zajac, M., Secret sharing for Sharing Mnemonics, Part I

: https://medium.com/nethermind-eth/using-shamirs-secret-sharing-to-share-mnemonics-c40429835117

- Zajac, M., Sen, U., Secret sharing for Sharing Mnemonics, Part II

: https://medium.com/nethermind-eth/secret-sharing-for-sharing-mnemonics-part-ii-d76dc1581681

- Desmedt, Y., Frankel, Y. (1992). Shared generation of authenticators and signatures. In: Feigenbaum, J. (eds) Advances in Cryptology — CRYPTO '91. CRYPTO 1991. Lecture Notes in Computer Science, vol 576. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-46766-1_37

- Pedersen, T.P., A Threshold Cryptosystem without a Trusted Party

. Advances in Cryptology — EUROCRYPT 1991. Lecture Notes in Computer Science. https://link.springer.com/chapter/10.1007/3-540-46416-6_47

- Gennaro, R., Jarecki, S., Krawczyk, H. et al. Secure Distributed Key Generation for Discrete-Log Based Cryptosystems

. J Cryptology, 2007. https://link.springer.com/article/10.1007/s00145-006-0347-3

- Feldman, P., A practical scheme for non-interactive verifiable secret sharing

. 28th Annual Symposium on Foundations of Computer Science, 1987.https://ieeexplore.ieee.org/document/4568297

- Pedersen, T.P., Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing

. Advances in Cryptology — CRYPTO 1991. Lecture Notes in Computer Science. https://link.springer.com/chapter/10.1007/3-540-46766-1_9

- Kate, A., Goldberg, I., Asynchronous Distributed Private-Key Generators for Identity-Based Cryptography

. Cryptology e-Print archive, 2009. https://eprint.iacr.org/2009/355

- Boneh, D., Lynn, B. & Shacham, H., Short Signatures from the Weil Pairing

. J Cryptology, 2004. https://link.springer.com/article/10.1007/s00145-004-0314-9

- Gurkan, K. et al. Aggregatable Distributed Key Generation

. Cryptology e-Print archive, 2021. https://eprint.iacr.org/2021/005

- Joux, A., Nguyen, K. Separating Decision Diffie-Hellman from Computational Diffie-Hellman in Cryptographic Groups

. J

ournal of Cryptology, 2003. https://link.springer.com/article/10.1007/s00145-003-0052-4

- Kate, A. Distributed Key Generation and Its Applications

. https://uwspace.uwaterloo.ca/bitstream/handle/10012/5285/Thesis.pdf?sequence=1

- Tomescu, A. et al. Towards Scalable Threshold Cryptosystems

. IEEE Symposium on Security and Privacy (SP), 2020.https://people.csail.mit.edu/devadas/pubs/scalable_thresh.pdf

- Groth, J. Non-interactive distributed key generation and key resharing.