Golden Sequence for the PPSS Broadcast Encryption Scheme with an Asymmetric Pairing

Renaud Dubois[1], Margaux Dugardin[1], Aurore Guillevic[1, 2]

July 2013

[1] Laboratoire Chiffre (LCH), Thales Communications and Security
4, avenue des Louvresses – 92622 Gennevilliers Cedex – France
renaud.dubois@thalesgroup.com
margaux.dugardin@thalesgroup.com
aurore.guillevic@thalesgroup.com
[2]Crypto Team, DI, ENS – 45 rue d'Ulm – 75230 Paris Cedex 05 – France

Abstract

Broadcast encryption is conventionally formalized as broadcast encapsulation in which, instead of a ciphertext, a session key is produced, which is required to be indistinguishable from random. Such a scheme can provide public encryption functionality in combination with a symmetric encryption through the hybrid encryption paradigm. The Boneh-Gentry-Waters scheme of 2005 proposed a broadcast scheme with constant-size ciphertext. It is one of the most efficient broadcast encryption schemes regarding overhead size. In this work we consider the improved scheme of Phan-Pointcheval-Shahandashi-Stefler [PPSS12] which provides an adaptive CCA broadcast encryption scheme. These two schemes may be tweaked to use bilinear pairings[DGS12].This document details our choices for the implementation of the PPSS scheme. We provide a complete golden sequence of the protocol with efficient pairings (Tate, Ate and Optimal Ate). We target a 128-bit security level, hence we use a BN-curve [BN06]. The aim of this work is to contribute to the use and the standardization of PPSS scheme and pairings in concrete systems.

Keywords: Broadcast Encryption Implementation.

1 Introduction

1.1 Overview

A broadcast encryption is a cryptographic scheme that enables encryption of broadcast content such that only a set of target users, selected at the time of encryption, can decrypt the content. Apparent applications include group communication, pay TV, content protection, file access control, and geolocalization. In [PPSS12], the authors propose an efficient dynamic broadcast encryption scheme with constant-size ciphertexts. This scheme is an improvement of [BGW05] from selective CPA to adaptive CCA security. We study the BGW scheme implementation proposed in [DGS12] and adapt the modifications to the PPSS scheme. We use a more efficient asymmetric pairing and provide more details about the sum computation.

This document presents detailed example vectors for the broadcast encryption scheme specified in [PPSS12] with an asymmetric pairing. For each function and each step of the scheme we give an example vector using elliptic curve domain parameters over $\mathbb{F}_p$. The BGW scheme introduced an efficient broadcast encryption scheme with constant-size ciphertexts (a description of the authorized users must be added to this ciphertext). The intersting properties of BGW are achieved thanks to a bilinear pairing. The broadcaster owns a master secret key and each receiver owns a single secret key. In [DGS12] the authors showed that this scheme is practical even with a large set of users. They provided efficient timings for encryption on a standard PC and decryption on a smartphone. In this work we detail each step and function of the PPSS scheme implemented on a Barreto-Naehrig curve. This work will be usefull for engineers whishing to promote this scheme and develop a demonstrator. More generally this work will be usefull to anyone who wants do discover in practice the new generations of broadcast encryption schemes using pairings.

1.2 Organization

This document is organized as follows.

- Section 2 describes the mathematical preliminaries and notations.
- Section 3 details the scheme [PPSS12] used for the broadcast encryption.
- Section 4 gives the parameters of the finite field and the curves.
- Section 5 gives the golden sequence, with two examples of encryption and decryption with the Tate Pairing.
- Appendix A gives the notations used in this document.
- Appendix B gives the PPSS scheme designed with the users sorted in several groups.
- Appendix C gives the golden sequence with the Ate pairing.
- Appendix D gives the golden sequence with the Optimal Ate pairing.

2 Mathematical Preliminaries and notations

2.1 Elliptic curves over $\mathbb{F}_q$

An elliptic curve over $\mathbb{F}_q$ is defined in terms of solutions to an equation in $\mathbb{F}_q$. The reduced form of the equation defining an elliptic curve over $\mathbb{F}_q$ differs depending on whether the field has characteristic 2, 3 or is a prime finite field. In this document, we work only with the large characteristic.

Elliptic curves over $\mathbb{F}_p$:
Let $\mathbb{F}_p$ be a prime finite field so that $p \geqslant 5$ is an odd prime number, and let a_E,$b_E \in \mathbb{F}_p$ satisfying $4a_E^3 + 27b_E \neq 0 \mod p$. We explain in Sec. 2.5 our choice for the size of p. Then an elliptic curve $E(\mathbb{F}_p)$ defined by the parameters a_E,$b_E \in \mathbb{F}_p$ consists of the set of solutions or points $P = (x, y)$ for x,$y \in \mathbb{F}_p$ to the reduced Weierstrass equation:

$$y^2 = x^3 + a_E x + b_E \mod p$$

together with an extra point O called the point at infinity. The equation $y^2 = x^3 + a_E x + b_E \mod p$ is called the defining equation of $E(\mathbb{F}_p)$. For a given point $P = (x_P, y_P)$, x_P is called the x-coordinate of P, and y_P is called the y-coordinate of P.

The number of points on $E(\mathbb{F}_p)$ is denoted by $\#E(\mathbb{F}_p)$. The Hasse Theorem states that:

$$p + 1 - 2\sqrt{p} \leq \#E(\mathbb{F}_p) \leq p + 1 + 2\sqrt{p}$$

It is possible to define an addition law to add points on E. The addition law is specified as follows:

1. Law to add the point at infinity to itself:
$$O + O = O.$$
2. Law to add the point at infinity to any other point: For all $(x, y) \in E(\mathbb{F}_p)$,
$$O + (x, y) = (x, y) + O = (x, y).$$
3. Law to add two points with the same x-coordinate: when the points are either distinct or have both y-coordinate 0: For all $(x, y) \in E(\mathbb{F}_p)$,
$$(x, y) + (x, -y) = O$$
-i.e. the negative of the point (x, y) is $-(x, y) = (x, -y)$.

4. Law to add two points with different x-coordinates: let $(x_1, y_1) \in E(\mathbb{F}_p)$ and $(x_2, y_2) \in E(\mathbb{F}_p)$ be two points such that $x_1 \neq x_2$. Then $(x_1, y_1) + (x_2, y_2) = (x_3, y_3)$, where:

$$x_3 = \lambda^2 - x_1 - x_2 \mod p \text{ , } y_3 = \lambda(x_1 - x_3) - y_1 \mod p, \text{ and } \lambda = \frac{y_2 - y_1}{x_2 - x_1} \mod p \text{ .}$$

5. Law to add a point to itself (double a point): Let $(x_1, y_1) \in E(\mathbb{F}_p)$ be a point with $y_1 \neq 0$. Then $(x_1, y_1) + (x_1, y_1) = 2 \cdot (x_1, y_1) = (x_3, y_3)$, where:

$$x_3 = \lambda^2 - 2x_1 \mod p \text{ , } y_3 = \lambda(x_1 - x_3) - y_1 \mod p \text{ and } \lambda = \frac{3x_1^2 + a_E}{2y_1} \mod p \text{ .}$$

The set of points on $E(\mathbb{F}_p)$ forms a group under this addition law. Furthermore the group is abelian - meaning that $P_1 + P_2 = P_2 + P_1$ for all points $P_1, P_2 \in E(\mathbb{F}_p)$. Note that the addition law can always be computed efficiently using simple field arithmetic.

Cryptographic schemes based on ECC rely on scalar multiplication of elliptic curve points. Given an integer t and a point $P \in E(\mathbb{F}_p)$, scalar multiplication is the process of adding P to itself t times. The result of this scalar multiplication is denoted $t \cdot P$. Scalar multiplication of elliptic curve point can be computed efficiently using the addition law together with the double-and-add algorithm or one of its variants.

2.2 Pairing

In cryptography, we define a pairing by the map:

$$e : (G_1, +) \times (G_2, +) \rightarrow (G_3, \times)$$

The pairing e satisfies:

- Bilinearity: let $P \in G_1$ and $Q \in G_2$, $\forall (u, v) \in \mathbb{F}_p^* : e(u \cdot P, v \cdot Q) = e(P, Q)^{uv}$.
- Non-degeneracy: for any $P \in G_1 \setminus \{0\}$ $\exists Q \in G_2$ such that $e(P, Q) \neq 1$
- For practical purpose, e has to be efficiently computable.

In this document, we use the Tate pairing and two other variants: the Ate pairing and the Optimal Ate Pairing, defined in [HSV06, Ver10].

Tate pairing:

Let $\mathbb{F}_p$ be a prime finite field and E an elliptic curve over $\mathbb{F}_p$ with a subgroup of prime order m. Let k be the embedding degree i.e. the smallest integer k such that $m | p^k - 1$.

$$\begin{array}{rccc} e_T : & E(\mathbb{F}_p)[m] \times E(\mathbb{F}_{p^k})/mE(\mathbb{F}_{p^k}) & \rightarrow & \mathbb{F}_{p^k}^* / (F_{p^k}^*)^m \\ & (P, Q) & \mapsto & f_{m,P}(D_Q)^{\frac{p^k - 1}{m}} \end{array}$$

with:

- For every $P \in E(\mathbb{F}_p)$, let $f_{m,P}$ be the $\mathbb{F}_p$-rational function with divisor:

  $$(f_{m,P}) = m(P) - (m \cdot P) - (m - 1)O.$$

- The divisor $D_Q = (Q + R) - (R)$ with R a random point in $E(\mathbb{F}_{p^k})$, such as D_Q is co-prime with $(P) - (O)$.
- The final exponentiation is used to have a unique representative. This Tate pairing may be denoted by reduced Tate pairing in a cryptographic context. This means we perform the final exponentiation.

Ate pairing:
Let $E(\mathbb{F}_p)$ be an elliptic curve, m a large prime with $m \mid \#E(\mathbb{F}_q)$ and denote by t the trace of the Frobenius endomorphism, $\#E(\mathbb{F}_p) = p + 1 - t$. Let k be the embedding degree with respect to p and m. For $T = t - 1$, $Q \in \mathbb{G}_2 = E[m] \cap \mathrm{Ker}(\pi_q - [q])$ and $P \in \mathbb{G}_1 = E[m] \cap \mathrm{Ker}(\pi_q - [1])$, the Ate pairing is defined as

$$\begin{array}{rcl} e_A : \mathbb{G}_2 \times \mathbb{G}_1 & \to & \mathbb{F}_{p^k}^* / (F_{p^k}^*)^m \\ (Q, P) & \mapsto & f_{T,Q}(D_P)^{\frac{p^k-1}{m}} \end{array}$$

with:

- For every $Q \in \mathbb{G}_2$, let $f_{T,Q}$ be the $\mathbb{F}_{p^k}$-rational function with divisor:
$$(f_{T,Q}) = T(Q) - (T \cdot Q) - (T - 1)O.$$
- The divisor $D_P = (P) - (O)$, D_P is co-prime with $(Q) - (O)$ since $\mathbb{G}_2 \cap \mathbb{G}_1 = O$ by construction.
- The final exponentiation is used to have a unique representative.
- The Frobenius is $\pi_p : E \to E : (x, y) \to (x^p, y^p)$. We use the same notation π_p for the Frobenius in $\mathbb{F}_{p^k}$.

We know that:
$$\pi_p(e_T(Q, P))^{\frac{(t-1)^k-1}{m}} = e_A(Q, P)^k \ .$$

Optimal Ate pairing:
In [Ver10], the author explain how to compute a pairing in $O\left(\frac{\log_2(m)}{\varphi(k)}\right)$.

Here is the Magma[BCP97] code to compute the Tate pairing and the Ate Pairing[1]:

```
x_E := 4611686018427944831;
p := 36*x_E^4+36*x_E^3+24*x_E^2+6*x_E+1;
m := 36*x_E^4+36*x_E^3+18*x_E^2+6*x_E+1;
k := 12;
t := p+1-m;
Fp := FiniteField(p);
lambda := Fp ! -1;
Fp2<X> := ExtensionField<Fp , x | x^2 - lambda>;
beta := Fp2! 1+X;
Fpk<U> := ExtensionField<Fp2 , u| u^6-beta>;
b_E_Fp := 12;
a_E_Fp := 0;
E_Fp := EllipticCurve([Fp ! a_E_Fp, Fp ! b_E_Fp]);
E_Fpk := E_Fp(Fpk);
P := E_Fpk ! [ 1, 10208195048256637760526282262283388199581052229439012341787449317362490730242];
bt := Fp2! b_E_Fp/beta;
Et_Fp2 := EllipticCurve([Fp2 ! a_E_Fp, Fp2! bt]);
Q :=Et_Fp2 ![ 4180895785587028667826786850619781135848051703205812940997073315544780465195
          +X*2198361849197333770042321426456007583724775794524124257318292856528840823424,
             10278790021048961159171385485866198250182016309472954570413203392144239750957
          +X*12031699434177040182637280953199138587350591234273202953866202774531978144509];

n := 101;
alpha := 4626059160041950428763316192902226066119825950263450353576299783137533861908;
```

---

[1] The Magma Algebra System has not implemented the function Optimal Ate Pairing.

```
Pn := alpha^n*P;
Q1 := alpha * Q;
Q1 := E_Fpk ! [Q1[1]*U^2, Q1[2]*U^3];

e:=ReducedTatePairing(Pn,Q1,m);
e;
(2127812259550993495072584731037300935704889386208043623713155084905253792095*X +
 12040400999163887538212377340089328065927347824189722347923869229369290080663)*U^5 +
(16188971139605893944883252994981336385916587324616518086389293519646187783112*X +
 9291245618952431096054272516026793305152077614557855985349622539553855200880)*U^4 +
(145420087133900650406556921688926195370190973630123616890122328239612895717*X +
 15241427258801996725372979128513686955119404666059012223506760614915576032858)*U^3 +
(3992586121504756341504873227277175827562094454075522566015033329571147714786*X +
 1521139847720600344495516242465234524670070815494222964997130067772954527114)*U^2 +
(5038203151404631215916969864982186096569109661221215159734364820370518154378*X +
 2034617478678334330114287231256676256529198327912042265372082307973047752582)*U +
 3736224127587849951207374140465494525783296894277339955308027604152345576535*X +
 9777006672766793355569318537895760242288532693135997144224010579608079171503

e1:=ReducedAteTPairing(Q1,Pn,m,p);
e1;
(12793504745823214202486921592248938326228519139335780636658737471402397474051*X +
 2695784522523109676632842347919192364083954576423543791362742613381652436319)*U^5 +
(8690600451356069447465221007428368339463640640321549811654028691982172060957*X +
 14298575420554289594792509115830063424429790989698922278036285003200195357441)*U^4 +
(3076137051186605784990224803334726556784795497521097092187992591079716946744*X +
 5181302670099845548837284208093312981855513208249475704966936823262062834217)*U^3 +
(13215537932989410804981533038265425736838035447246391225124785211393905317674*X +
 1153302214757886741403939359571516898489672022545254368854466846594116076258)*U^2 +
(12552263301499855327232446196944428108745030852315680584657136375473421919685*X +
 14867456709199068373298987401160283843298847977670925602408199365402343882038)*U +
 11131443985348788290486407838823010152214468606170871745863301904649070776125*X +
 15557552376765027901259096680778299977167019284665517908046027812000922253665

L:=((t-1)^k-1 )div m ;
Frobenius(ReducedTatePairing(Q1,Pn,m),Fp)^L eq ReducedAteTPairing(Q1,Pn,m,p)^k;
true
```

2.3 Barreto-Naehrig Curve and Optimal Pairing

The family of BN-curves [BN06] has embedding degree $k = 12$ and is given by the following parameterization:

$$p(x) = 36x^4 + 36x^3 + 24x^2 + 6x + 1$$

$$m(x) = 36x^4 + 36x^3 + 18x^2 + 6x + 1$$

In [Ver10], the author obtained:

$$W(x) = [6x + 2, 1, -1, 1]$$

The Optimal Ate Pairing can be computed as :

$$e_{Opt} = (f_{6x+2,Q}(P) \cdot M)^{\frac{p^k-1}{m}}$$

where $M = l_{Q_3,-Q_2}(P) \cdot l_{-Q_2+Q_3,Q_1}(P) \cdot l_{Q_1-Q_2+Q_3,(6x+2)\cdot Q}(P)$, $Q_i = p^i \cdot Q$ for $i = 1, 2, 3$ and l_{Q_i,Q_j} is the equation of the line through Q_i and Q_j (or the tangent line when $Q_i = Q_j$). Moreover the line $l_{Q_1-Q_2+Q_3,(6x+2)\cdot Q}(P)$ can be removed from the computation since $Q_1 - Q_2 + Q_3 = -(6x + 2) \cdot Q$ by construction.

We know that:

$$e_{Opt}(Q,P) = \frac{(e_T(Q,P))^{6x^2-6x+1}}{(e_A(Q,P))^{1-2(t-1)+3(t-1)^2}}$$

2.4 Conversion between Decimal Basis and Hexadecimal Basis

This document uses integer notation in decimal basis and in hexadecimal basis. Let $Base$ be the base (10 or 16). Let (z_i) a sequence of integers ($\forall i,\ 0 \leq z_i \leq Base - 1$).

Let X an integer such that: $X = a_n \times Base^n + a_{n-1} \times Base^{n-1} + \ldots + a_1 \times Base + a_0$

The notation of X is: $z_n z_{n-1} \ldots z_1 z_0$. The numbers X are in decimal basis and the numbers $\overline{X}$ are in hexadecimal basis.

For example: $X = 123 = 1 \times 10^2 + 2 \times 10 + 3 = 7 \times 16 + 11$ so $\overline{X} = 7B$

For the legibility of this document, we write the hexadecimal number by 4 bytes long blocks.

2.5 Security Level, Recommended Size

The elliptic curve $E(\mathbb{F}_p)$ is a group. The generic attacks on the discrete logarithm (Pollard-ρ, Baby Step Giant Step combined with Polhlig-Hellman) are in $O(\sqrt{l})$, where l is the largest prime factor of $\#E(\mathbb{F}_q)$. The Lenstra-Verheul, NIST and NESSIE recommendations for ECC (in [Ecr07]) are:

Security level	Recommended Size ($log_2(l) \simeq log_2(p)$)	Embedding Degree k
56	112	
64	128	
80	160	$6-8$
96	192	
112	224	$10-16$
128	256	$12-20$
160	320	
192	384	$20-26$
256	512	$28-36$

In this document, we use an elliptic curve with $m = \#E_1(\mathbb{F}_p)$ (defined in section 4). m is a prime number and $log_2(m) \simeq 256$, so the security level is: 128 bits. We choose a BN curve.

3 PPSS Scheme

This section specifies the broadcast encryption scheme explained in [PPSS12], using elliptic curve domain parameters over $\mathbb{F}_p$ and $\mathbb{F}_{p^2}$. In [DGS12], the authors propose to adapt the scheme [BGW05] to an asymmetric paring in order to have a group E_1 with smaller coefficients and use precomputation to compute more quickly the sum. This adaptation can be extended easily to PPSS.

The PPSS scheme needs a bilinear pairing hence a pairing-friendly curve. We have chosen to use a Barreto-Naehrig curve. This gives us the best performances at the moment for pairings at the 128-bit security level. The PPSS scheme is fully collision-secure. This means any collusion of revoked users cannot recover the secret key of an authorized user. The PPSS scheme needs a one-way universal hash function H_κ. The assumptions used for the security proof are the BDHE and GBDHE assumptions. The Bilinear Diffie-Hellman Exponent problem ($\ell-$BDHE) with a symmetric pairing is given a vector of $2\ell+1$ elements $\left(h, g, g^{\lambda}, g^{\lambda^2}, \ldots, g^{\lambda^\ell}, g^{\lambda^{\ell+2}}, \ldots, g^{\lambda^{2\ell}}\right)$ of a prime order bilinear group G, compute the element $e(g,h)^{\lambda^{\ell+1}} = e(g^{\lambda^{\ell+1}}, h)$ with $g^{\lambda^{\ell+1}}$ missing in the input sequence. The generalized version stands for asymmetric bilinear pairings. The input sequence is $\left(g^{\lambda^i}, h^{\lambda^j}\right)_{1 \leqslant i \neq \ell+1 \leqslant 2\ell, 1 \leqslant j \leqslant \ell-1}$

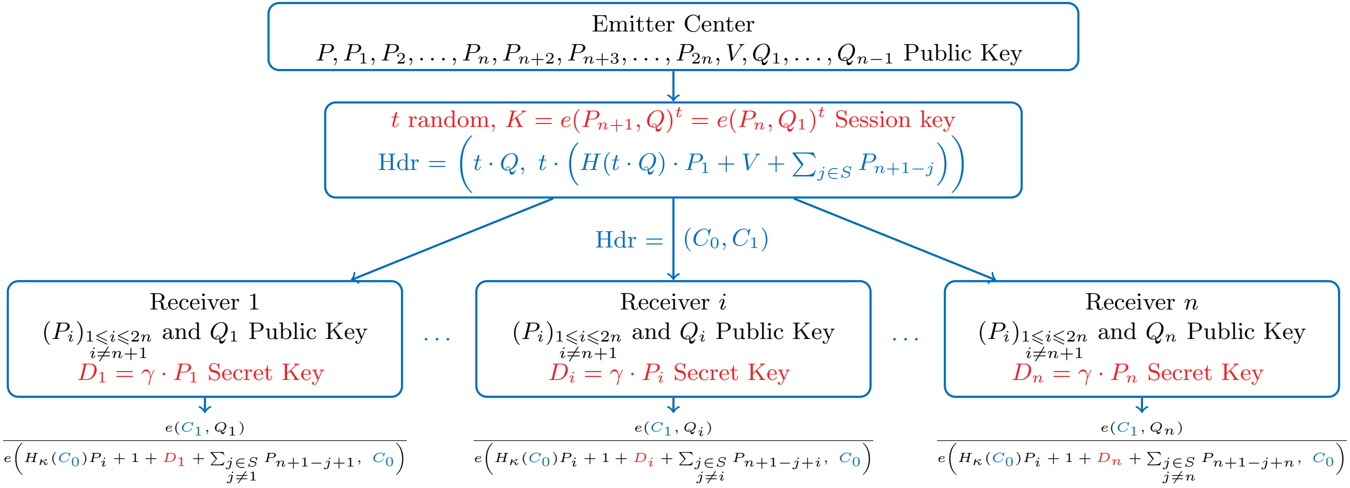
with $g \in G_1, h \in G_2$. The challenge is to output $e(g,h)^{\lambda^{\ell+1}} = e(g^{\lambda^{\ell+1}}, h)$.

The scheme used an asymmetric pairing $e : E_1(\mathbb{F}_p) \times E_2(\mathbb{F}_{p^2}) \rightarrow (\mathbb{F}_{p^k})^*$. Now, we will use the additive notation for both E_1 and E_2 and the multiplicative notation for $(\mathbb{F}_{p^k})^*$.

To respect the notation in the article [PPSS12], we work with $n-1$ users.
E (Emitter) and R (Receiver) use the broadcast scheme as follows.
A Broadcast scheme is composed by 4 functions:

1. Set Up explained in 3.1.
2. Join explained in 3.2.
3. Encrypt explained in 3.3.
4. Decrypt explained in 3.4.

This figure represents the PPSS scheme for $n-1$ users:



3.1 Set Up $(n-1)$:

E generates the master secret key and the public key for the scheme (MSK, PK_s).

Input: The elliptic curve domain parameters as specified in Section 4 and $n-1$ is the number of users

Action: E selects the keys.

1. Compute n with the number of users.
2. Generate an random integer α in $[2, \ldots, m-1]$
3. Generate an random integer γ in $[2, \ldots, m-1]$.
4. Compute the sequence P_i of E_1 for $i = 1, \ldots, n, n+2, \ldots, 2n$, such that $P_i = \alpha^i \cdot P$.
5. Compute the point V of E_1, such that $V = \gamma \cdot P$.
6. Compute the sequence Q_i of E_2 for $i = 1, \ldots, n-1$, such as $Q_i = \alpha^i \cdot Q$.
7. Generate a random index κ to choose the hash function H_κ.
8. Store $PK_s = (P, P_1, P_2, \ldots, P_n, P_{n+2}, \ldots, P_{2n}, V, Q, Q_1, \ldots, Q_{n-1}, \kappa)$.

9. Store the $MSK = (\alpha, \gamma)$.

Output: $PK_s = (P, P_1, P_2, \dots, P_n, P_{n+2}, \dots, P_{2n}, V, Q, Q_1, \dots, Q_{n-1}, \kappa)$, $MSK = (\alpha, \gamma)$ and the hash function H_κ.

3.2 Join (MSK, i):

E generates a secret key for R.

Input: The master secret key MSK and the index of the user $i \in [1, n-1]$.

Action: E generates a i-th secret key for R and select the public key from PK_s.

1. Compute the point $D_i = \gamma \cdot \alpha^i \cdot P \in E_1$.

Output: The elliptic curve point D_i for the secret key and the public key: $PK_i = (P, P_1, \dots, P_n, P_{n+2}, \dots, P_{2n}, Q_i)$

E gives D_i and PK_i to R.

3.3 Encrypt(S, Pk_s, H_κ):

E generates a session key to encrypt a message and the header, such that R can compute the session key, iff R is authorized.

Input: S the set of the authorized users, the public key PK_s and the hash function H_κ.

Action: E generates a session key K and a header key `Hdr`

1. Generate an integer t.
2. Compute the session key K.
   2.1 Compute the pairing $e(P_{n+1}, Q)$
   2.2. Compute the exponentiation in $\mathbb{F}_{p^k}$: $K = (e(P_{n+1}, Q))^t$
3. Compute the header $\texttt{Hdr} = (C_0, C_1)$
   3.1. Compute $C_0 = t \cdot Q$ in E_2.
   3.2. Compute $h = H_\kappa(t \cdot Q)$.
   3.3. Compute $h \cdot P_1 = (x_{hP_1}, y_{hP_1})$ in E_1.
   3.4. Compute $Sum = \sum_{j \in S} P_{n+1-j} = (x_{Sum}, y_{Sum})$ in E_1.
   3.5. Compute $h \cdot P_1 + V + Sum$ in E_1.
   3.6. Compute $C_1 = t \cdot (h \cdot P_1 + V + Sum)$ in E_1.

Output: The pair $(K, \texttt{Hdr})$.

E encrypts a message with K, adds the `Hdr` to the message and broadcasts all.

3.4 Decrypt$(i, D_i, PK_i, S, \texttt{Hdr}, H_\kappa)$:

R can compute the session key if he is authorized.

Input: The user i, the secret key D_i, the public key PK_i, a description of the set S (of authorized users), the header `Hdr` and the hash function H_κ.

Action: Compute the session key K if the i-th user is authorized.

1. Compute the pairing $K_1 = e(C_1, Q_i)$.
2. Compute $h = H_\kappa(C_0)$.
3. Compute $Sum = \sum_{j \in S \setminus \{i\}} P_{n+1-j+i}$ in E_1.
4. Compute $h \cdot P_{i+1} + D_i + Sum$ in E_1.
5. Compute the pairing $K_2 = e(h \cdot P_{i+1} + D_i + Sum, C_0)$.
6. Compute the inversion in $\mathbb{F}_{p^k}$ of $e(h \cdot P_{i+1} + D_i + Sum, C_0)$.
7. Compute the session key $K = K_1 \times K_2^{-1}$.

Output: The key K.

R can decrypt the ciphertext with K.

4 Parameter Initialization

The elliptic curve domain parameters over $\mathbb{F}_p$ are specified by $(x, p, a_{E_1}, b_{E_1}, G, m, t_{E_1})$, where the finite field $\mathbb{F}_p$ is defined by:

$$\begin{aligned} x &= 4611686018427944831 \\ p &= 16283262549005455731706454238259997169449030509273276621164013331956021995283 \end{aligned}$$

As an octet string, we have:

$$\begin{aligned} \overline{x} &= \mathtt{40000000\ \ 00087F7F} \\ \overline{p} &= \mathtt{24000000\ \ 00131EDE\ \ 500003CE\ \ EC974A28\ \ 964D2C8B\ \ EE1F7C51\ \ 1355420E\ \ 690A2713} \end{aligned}$$

The curve $E_1 : y^2 = x^3 + a_{E_1} x + b_{E_1}$ over $\mathbb{F}_p$ is defined by:

$$\begin{aligned} a_{E_1} &= 0 \\ b_{E_1} &= 12 \end{aligned}$$

As an octet string, we have:

$$\begin{aligned} \overline{a_{E_1}} &= \mathtt{0} \\ \overline{b_{E_1}} &= \mathtt{C} \end{aligned}$$

The generator $P = (x_P, y_P)$ of $E_1(\mathbb{F}_p)$ is defined by:

$$\begin{aligned} x_P &= 1 \\ y_P &= 10208195048256637760526282262283388199581052229439012341787449317362490730242 \end{aligned}$$

As an octet string, we have:

$$\begin{aligned} \overline{x_P} &= \mathtt{1} \\ \overline{y_P} &= \mathtt{1691A236\ \ 9AA68F26\ \ AF4FC3D1\ \ 7DBE8F1E\ \ 3D86AB88\ \ F68D170A\ \ 554FEF98\ \ 7E38E702} \end{aligned}$$

The order m and the trace t_{E_1} of the group $E_1(\mathbb{F}_p)$ is defined by:

$$\begin{aligned} m &= 16283262549005455731706454238259997169321424621677893876895737635789744283917 \\ t_{E_1} &= 127605887595382744268275696166277711367 \end{aligned}$$

As an octet string, we have:

$$\begin{aligned} \overline{m} &= \mathtt{24000000\ \ 00131EDE\ \ 500003CE\ \ EC974A28\ \ 364D2C8B\ \ EE05FDD4\ \ 1355405D\ \ 1C6EA10D} \\ \overline{t_{E_1}} &= \mathtt{60000000\ \ 00000954\ \ 00000000\ \ 003A0261} \end{aligned}$$

We define the quadratic extension by $\mathbb{F}_{p^2} \simeq \mathbb{F}_p[X]/(X^2 - \lambda)$, with $\lambda = p - 1$.

λ = 16283262549005455731706454238259997169449030509273276621164013331956021995282

As an octet string, we have:

$\overline{\lambda}$ = `24000000 00131EDE 500003CE EC974A28 964D2C8B EE1F7C51 1355420E 690A2712`

We define the extension $\mathbb{F}_{p^{12}}$ by $\mathbb{F}_{{p^2}^6} \simeq \mathbb{F}_{p^2}[U]/(U^6 - \beta)$ with $\beta \in \mathbb{F}_{p^2}$ $(\beta = \beta_0 + \lambda\beta_1)$.

β_0 = 1
β_1 = 1

As an octet string, we have:

$\overline{\beta_0}$ = `1`
$\overline{\beta_1}$ = `1`

The parameter of the curves over $E_2(\mathbb{F}_{p^2})$ is defined by (a_{E_2}, b_{E_2}) and its generators $Q = (Qx, Qy) = (Qx = Qx_0 + \lambda Qx_1$ et $Qy = Qy_0 + \lambda Qy_1)$.

$$a_{E_2} = 0$$

$$b_{E_2} = 12/\beta$$

Qx_0 = 4180895785587028667826786850619781135848051703205812940997073315544780465195
Qx_1 = 2198361849197333770042321426456007583724775794524124257318292856528840823424
Qy_0 = 10278790021048961159171385485866198250182016309472954570413203392144239750957
Qy_1 = 12031699434177040182637280953199138587350591234273202953866202774531978144509

As an octet string, we have:

$\overline{Qx_0}$ = `093E4D9B A200D5F4 67F8DE35 1FA89F68 796C7E0D 99B00CBF 43443696 0B1B642B`
$\overline{Qx_1}$ = `04DC3A8C ECBF3FEE 72B7C0B0 FA79756A 6BBB1B0C F1BDE9A0 EE7B2C99 E48E1280`
$\overline{Qy_0}$ = `16B996C7 AD4F692A 1A14B760 53C4FA1A 5C9C0596 86564D72 CEE1630F 9217EF2D`
$\overline{Qy_1}$ = `1A99B357 71D91184 58B5E67E 0C995A6F 25F77C75 DEC3F1E2 0AA487BB BFF5AAFD`

Here, $\log_2(m) = 254$, so we have a 127 bit security level with this curve.

We use the Tate pairing $e_T : E_1 \times E_2 \to (\mathbb{F}_{p^{12}})^*$, explained in 2.2.

For the family hash function, we use the HMAC_SHA256 in [HMA08]. $H_\kappa(x) =$ HMAC_SHA256(x, κ)(where κ is the key).

5 Golden Sequence

We provide this golden sequence to anyone who wants to verify his own implementation of the scheme in [PPSS12], tweaked to use asymmetric pairing as explained in [DGS12] for [BGW05], easily adaptable for PPSS.
The Security level is 127 bits because $\log_2(m) = 254$. (see section 2.5)

5.1 Set Up

E generates the master secret key and the public key for the scheme (MSK, PK_s).

Input: The elliptic curve domain parameters as specified in Section 4 and the number of users $n - 1$

Action: Selects the keys.

1. Compute n in function of the number of users.

   $n - 1 = 100$

   $n = 101$

2. Generate an integer α

   2.1. Randomly or pseudorandomly select an integer α in the interval $[1, m-1]$.

   α = 4626059160041950428763316192902226066119825950263450353576299783137533861908

   2.2. Convert α to the octet string $\overline{\alpha}$.

   $\overline{\alpha}$ = A3A41B6 E6122DCB D07A777B 248D2AE1 FFAD2B0F E5C21D6F A1204178 53FBA014

3. Generate an integer γ.

   3.1. Randomly or pseudorandomly select an integer γ in the interval $[1, m-1]$.

   γ = 7084151545225827683048027685504717764765820549485153500437942755079924941816

   3.2. Convert γ to the octet string $\overline{\gamma}$.

   $\overline{\gamma}$ = FA97CD8 D6EBBA0B 5009E0A9 B8BB56CE 73DEAA56 68B05AB5 3F85EB7F A05303F8

4. Compute the suite $P_i = (x_{P_i}, y_{P_i})$ of E_1 for $i = 1, \ldots, n, n+2, \ldots, 2n$, such as $P_i = \alpha^i \cdot P$.

   $\overline{P}$ = (1 ,
   1691A236 9AA68F26 AF4FC3D1 7DBE8F1E 3D86AB88 F68D170A 554FEF98 7E38E702)

   x_{P_1} = 8756548003751963313740253182383096452130833802089262714269164634677292157681

   y_{P_1} = 1759428182473644884533561713628609073597949874079275581965268974884096704504

   and

   x_{P_6} = 10677727583815662828631663720667260096824918613174008559333515067698971939683

   y_{P_6} = 12102989905481467395619092214245312862737116532853841126909425074766368031297

   and

   x_{P_n} = 14103341190032026297502531464841754115697825634327393479077911885421510355939

   y_{P_n} = 3240328156121332973762206011534720641361342678849179159352331734220842482124

   As an octet string, we have:

   $\overline{x_{P_1}}$ = 135C07D1 249FDEF8 BFF2DFD0 90A2F79B 8FDA147B 86A7F99C A7D8CEE9 9E09CAF1

   $\overline{y_{P_1}}$ = 3E3CD12 5C7A3F46 870613F2 A7924EAB C0DFE58D 548944D9 29349F33 AC29FFF8

   and

   $\overline{x_{P_6}}$ = 179B6130 4AE3C46D EFCB42AC 241C8304 4F9ED0FA 06108A2E E5B09233 B4F41363

   $\overline{y_{P_6}}$ = 1AC20CAD FDBF0AB4 055D8C8B 17498528 ADA81086 0C006C21 40DDCD88 562D4241

   and

   $\overline{x_{P_n}}$ = 1F2E354E DF838381 FB886C67 ECFDE49E D1E04EC5 BB201D0B 9F8DDA3F 886F6BE3

   $\overline{y_{P_n}}$ = 729F5F3 44F16BC9 C3833794 28F13899 94B218BC D9710596 104FE566 130839CC

5. Compute the point $V = (x_V, y_V)$ of E_1, such as $V = \gamma \cdot P$.

   x_V = 6854284133316136958068950795498250209547235928318577865338442429849499842285

   y_V = 6156029464667595409844675784591674601879322950347255832901970794708288250434

   As an octet string, we have:

   $\overline{x_V}$ = F276328 A897017D 73ED95AD D017D4CD 3B8766FB C519765F 3200CEA8 EF1FB6ED

   $\overline{y_V}$ = D9C306F 8AA5032B 212FFDC9 00EB27E1 72EA27B9 85591D4A 1D7CF993 CB4DA242

6. Compute the suite $Q_i = (x_0(Q_1) + \lambda x_1(Q_1), y_0(Q_1) + \lambda y_1(Q_1))$ of E_2 for $i = 1, \ldots, n-1$, such as $Q_i = \alpha^i \cdot Q$.

   $\overline{Q}$ = ((93E4D9B A200D5F4 67F8DE35 1FA89F68 796C7E0D 99B00CBF 43443696 0B1B642B ,
   4DC3A8C ECBF3FEE 72B7C0B0 FA79756A 6BBB1B0C F1BDE9A0 EE7B2C99 E48E1280),
   (16B996C7 AD4F692A 1A14B760 53C4FA1A 5C9C0596 86564D72 CEE1630F 9217EF2D ,
   1A99B357 71D91184 58B5E67E 0C995A6F 25F77C75 DEC3F1E2 0AA487BB BFF5AAFD))

$x_0(Q_1) = 5678198471493198473634910809486919214558721286741989740213944933498823752559$

$x_1(Q_1) = 3520589294613129300061109693307720615894843176090027995695500565750245079547$

$y_0(Q_1) = 13049912931196200256556971713599186545608623895422717141549252740077537081984$

$y_1(Q_1) = 1129297166433691338858098890044206360782403434710483092065192743377218765876$

and

$x_0(Q_5) = 7518310856601574668361122578033701599940212928139060231437212648867999356690$

$x_1(Q_5) = 2744482615241759171669590900629304974262453235426263447016387869187718121582$

$y_0(Q_5) = 15088075053734061890700421823724207898057951768675241712213684473205255388577$

$y_1(Q_5) = 7741107513544302770223088801533137043273566079936670950037428178626215023045$

As an octet string, we have:

$\overline{x_0(Q_1)}$ = 1415FE8 B1FE4B90 F7E6C5B3 5D716171 73CAC5DB 94517E7D 6F93AC73 A7F22C5B

$\overline{x_1(Q_1)}$ = 7C8953A A7FEE45F 168397CA 14117A15 421F071F 9756EA2F A3F43C85 1571B9FB

$\overline{y_0(Q_1)}$ = 1CD9FD2D 1A70BAD6 8AD193FE F734073B 6B589AFE 272CFD09 D66A10C2 A2AE3A80

$\overline{y_1(Q_1)}$ = 27F28D7 F4F737C5 181D8188 C6F2F8AC D4F3965A 7AE4F427 4F580DCE C8D30434

and

$\overline{x_0(Q_5)}$ = 109F3690 B88AA8A6 A3BFBA59 B748160F 336CF286 74EF398C 0FAA0FB1 23119712

$\overline{x_1(Q_5)}$ = 6115275 F81690E1 BEC455A9 CAE8260B BEA83A7D 64DD72BE 3AB40648 07E2086E

$\overline{y_0(Q_5)}$ = 215B8C3F E9D71760 0281CDF1 E830D8F2 C50D8914 F87ECB3F 97D1BF50 ADB93DA1

$\overline{y_1(Q_5)}$ = 111D4FC4 0611F31C 06F40938 A3AE6913 CA7273D9 465F46A7 9939AA01 D92A11C5

7. Choose the hash function H_κ

   7.1 Generate a pseudorandom index κ for hash function H.
   $\kappa = 90635379122041306652578531471302610019127743211974938460623100826577748195803$

   7.2. Convert α to the octet string $\overline{\alpha}$.
   $\overline{\kappa}$ = 1409C7D9 B598BE09 CD4519A6 9042BC42 F99EA0FE 295663CC A942C0E4 DE0F19DB

   7.3 Define H by $H(x) = \text{HMAC\_SHA256}(x, \kappa)$. The reference of this function is in [HMA08].

8. Store the public key $PK_s = (P, P_1, P_2, \ldots, P_n, P_{n+2}, \ldots, P_{2n}, V, Q, Q_1, Q_2, \ldots, Q_{n-1}, \kappa)$.

9. Store the master secret key $MSK = (\alpha, \gamma)$.

Output: $PK_s = (P, P_1, P_2, \ldots, P_n, P_{n+2}, \ldots, P_{2n}, V, Q, Q_1, \ldots, Q_{n-1}, \kappa)$, $MSK = (\alpha, \gamma)$ and the hash function H.

5.2 Join

E generates a secret key for R.

Input: The master secret key MSK and the index of the user $i \in [1, n-1]$

Action: Generate a i-th secret key for a user.

- The number of the user is: $i = 5$. We can be take all the integer in $[1, 100]$.

1. Compute the point $D_i = \gamma \cdot \alpha^5 \cdot P \in E_1$.

   $\overline{\alpha}$ = A3A41B6 E6122DCB D07A777B 248D2AE1 FFAD2B0F E5C21D6F A1204178 53FBA014

   $\overline{\gamma}$ = FA97CD8 D6EBBA0B 5009E0A9 B8BB56CE 73DEAA56 68B05AB5 3F85EB7F A05303F8

   $\overline{P}$ = (1 ,
   1691A236 9AA68F26 AF4FC3D1 7DBE8F1E 3D86AB88 F68D170A 554FEF98 7E38E702)

   $x_{D_5} = 1111685591128591012397073286157702586793019966854922136086609421936616443687$

   $y_{D_5} = 14944299665814395389306984237128660478256910672283797942227650283648125229404$

As an octet string, we have:

$$\begin{array}{lcl} \overline{x_{D_5}} & = & \texttt{2753116 528BEBA3 0E64CEAB 673FAAA1 B7AA4C0A 6A0B8DC3 26FD3BBD 8CF28F27} \\ \overline{y_{D_5}} & = & \texttt{210A2C82 61AD15B0 23D51C34 20E58108 973D4B6A F6D6BCC5 934CFF85 AD2BD95C} \end{array}$$

Output: The elliptic curve point $\overline{D_5}$ with:

$$\begin{array}{lll} \overline{D_5} & = (& \texttt{2753116 528BEBA3 0E64CEAB 673FAAA1 B7AA4C0A 6A0B8DC3 26FD3BBD 8CF28F27} \quad , \\ & & \texttt{210A2C82 61AD15B0 23D51C34 20E58108 973D4B6A F6D6BCC5 934CFF85 AD2BD95C} \quad) \end{array}$$

E gives D_i and $PK_i = (P, P_1, \ldots, P_n, P_{n+2}, \ldots, P_{2n}, Q_i)$ to R.

5.3 Example 1: Test with 100/100 authorized users

In this example, all the users are authorized.

5.3.1 Encrypt

E generates a session key to encrypt a message and the header, R can compute the session key, iff R is authorized.

Input: S the set of the users who are authorized, the public key PK_s, and the hash function H.

Action: Generate a session key K and a header key `Hdr`

1. Generate an integer t

   1.1. Randomly or pseudorandomly select an integer t in the interval $[1, m-1]$.

   $t \; = \; 5710657168379116176003428016857198065066034451013474592937188291088088041169$

   1.2. Convert t to the octet string $\bar{t}$.

   $\bar{t} \; = \; \texttt{CA01E0E EF2718A2 A90C3636 FCC04963 F9B9CFA1 22F216B3 D300A198 5CE006D1}$

2. Compute the session key K.

   2.1 Compute the pairing $e_T(P_{n+1}, Q)$ We can use P_n and Q_1.

$$\begin{array}{lll} \overline{P_n} & = (& \texttt{1F2E354E DF838381 FB886C67 ECFDE49E D1E04EC5 BB201D0B 9F8DDA3F 886F6BE3} \quad , \\ & & \texttt{729F5F3 44F16BC9 C3833794 28F13899 94B218BC D9710596 104FE566 130839CC} \quad) \\ \overline{Q_1} & = ((& \texttt{1415FE8 B1FE4B90 F7E6C5B3 5D716171 73CAC5DB 94517E7D 6F93AC73 A7F22C5B} \quad , \\ & & \texttt{7C8953A A7FEE45F 168397CA 14117A15 421F071F 9756EA2F A3F43C85 1571B9FB} \quad), \\ & (& \texttt{1CD9FD2D 1A70BAD6 8AD193FE F734073B 6B589AFE 272CFD09 D66A10C2 A2AE3A80} \quad , \\ & & \texttt{27F28D7 F4F737C5 181D8188 C6F2F8AC D4F3965A 7AE4F427 4F580DCE C8D30434} \quad)) \end{array}$$

$$\begin{array}{lll} e_T(P_n, Q_1) & = ((& 97770066727667933555693185378957602422885326931359971442240105796080791715503 \quad , \\ & & 37362241275878499512073741404654945257832968942773399553080276041523455765535 \quad), \\ & (& 20346174786783343301142872312566762565291983279120422653720823079730477525582 \quad , \\ & & 50382031514046312159169698649821860965691096612212151597343648203705181543378 \quad), \\ & (& 15211398477206003444955162424652345246700708154942229649971300677729545271114 \quad , \\ & & 39925861215047563415048732272771758275620944540755225660150333295711477147786 \quad), \\ & (& 152414272588019967253729791285136869551194046660590122235067606149155760322858 \quad , \\ & & 1454200871339006504065569216889261953701909736301236168901223282396128957717 \quad), \\ & (& 92912456189524310960542725160267933051520776145578559853496225395538552008880 \quad , \\ & & 161889711396058939448832529949813363859165873246165180863892935196461877831112 \quad), \\ & (& 120404009991638875382123773400893280659273478241897223479238692293692900080663 \quad , \\ & & 21278122595509934950725847310373009357048893862080436237131550849052537920095 \quad)) \end{array}$$

As an octet string, we have:

$$\begin{array}{rllllllllll}
\overline{e_T(P_n, Q_1)} = ((& \texttt{159D96F4} & \texttt{DC00B050} & \texttt{0CD063A5} & \texttt{0BA4BEB9} & \texttt{3050FDDB} & \texttt{6F41C859} & \texttt{497A1F12} & \texttt{43EBFFAF} & , \\
 & \texttt{842A0BF} & \texttt{24DA4C95} & \texttt{3BE3EB7B} & \texttt{561CA417} & \texttt{4CFDC272} & \texttt{8636B6B0} & \texttt{8529DDBE} & \texttt{2D124457} &), \\
(& \texttt{47F8D7C} & \texttt{A97F4307} & \texttt{04ECC764} & \texttt{071B1564} & \texttt{D6575BE3} & \texttt{4CF31E68} & \texttt{1EE8D187} & \texttt{B2A19786} & , \\
 & \texttt{B23859D} & \texttt{2D102D17} & \texttt{C15BA066} & \texttt{37EE8B49} & \texttt{B18EF3FB} & \texttt{2BE3EDB4} & \texttt{F5482300} & \texttt{A4A79C8A} &), \\
(& \texttt{35CEF44} & \texttt{CACE42A7} & \texttt{56C5F090} & \texttt{06C7DBA7} & \texttt{AD2D4ECA} & \texttt{31D7B934} & \texttt{9D6DDDAF} & \texttt{94A2458A} & , \\
 & \texttt{8D3B941} & \texttt{FD875DEF} & \texttt{DED652E7} & \texttt{FB8E7CC7} & \texttt{7712AB32} & \texttt{A437C4FD} & \texttt{2E3EB72E} & \texttt{476290E2} &), \\
(& \texttt{21B25795} & \texttt{560640B3} & \texttt{F1A07FD1} & \texttt{5788AC20} & \texttt{9959ECDF} & \texttt{3D553C07} & \texttt{2AA7BE01} & \texttt{AACDEE5A} & , \\
 & \texttt{524E0A} & \texttt{D0F96838} & \texttt{2BB2A793} & \texttt{C0B30252} & \texttt{8F860FDD} & \texttt{4B01CE3F} & \texttt{A2884B06} & \texttt{F1C821E5} &), \\
(& \texttt{148AA89D} & \texttt{F941B9DB} & \texttt{2849F3E3} & \texttt{71C5F5D5} & \texttt{0841FAC6} & \texttt{B5DA47E6} & \texttt{AB08DDAE} & \texttt{CE1F5E70} & , \\
 & \texttt{23CAA209} & \texttt{3E47299A} & \texttt{2BDA7210} & \texttt{3C631C3B} & \texttt{A12F80F5} & \texttt{A262D022} & \texttt{56F17D8F} & \texttt{505303C8} &), \\
(& \texttt{1A9EA01E} & \texttt{6DACB6C8} & \texttt{E9783621} & \texttt{798C5B53} & \texttt{07EF8513} & \texttt{E252745B} & \texttt{3A03E996} & \texttt{2B4B4197} & , \\
 & \texttt{4B44C8F} & \texttt{34D31EF4} & \texttt{C9C28E17} & \texttt{2DA58CB1} & \texttt{4E80397E} & \texttt{ABA28E67} & \texttt{2201DFA1} & \texttt{5791695F} &))
\end{array}$$

2.2. Compute the exponentiation in $\mathbb{F}_{p^k}$: $K = (e_T(P_{n+1}, Q))^t$

$$\begin{array}{rll}
K = ((& 1585174373219231923941953934221543074612861392737095144364452424513086305519 7 & , \\
 & 889648731136069855099597526711355850043796693279402623203917554160776343128 &), \\
(& 1203784208883178977961619147501486197992977307821834677417892037262867697262 3 & , \\
 & 1416076275059458210547757006955314521447875819454055757025020632179509948975 3 &), \\
(& 477431406323079528308302208933002986172354080285934059946452844508148369481 8 & , \\
 & 753365799964501208859913189062913498705220685677269817392222034955423725191 0 &), \\
(& 421464048407054439098744285492051629810745876348006059874073181588714879044 8 & , \\
 & 460923459276581275945277063221343605596295938573145860570632538075656947790 7 &), \\
(& 1181283229025953527747075555523093510756645745500107857706718177268219594561 2 & , \\
 & 583688107394375416134917056982953489947288330361197187213959117686923946798 0 &), \\
(& 466817568309317449698041936367471512318150842046728871711023926103050046618 4 & , \\
 & 262702326734080491605971848903376930071302688904536279896245759081867862507 7 &))
\end{array}$$

As an octet string, we have:

$$\begin{array}{rllllllllll}
\overline{K} = ((& \texttt{230BC4DD} & \texttt{817497B2} & \texttt{E2B74F91} & \texttt{B46F9B09} & \texttt{212D7E83} & \texttt{7C399629} & \texttt{0FCBE17C} & \texttt{BDC2115D} & , \\
 & \texttt{1F785F9} & \texttt{746E8A30} & \texttt{AB22682B} & \texttt{20CA4F66} & \texttt{FFF05944} & \texttt{0FDC7EB1} & \texttt{9147BA13} & \texttt{44ED6E58} &), \\
(& \texttt{1A9D2D5B} & \texttt{2B42CAAD} & \texttt{461D99F6} & \texttt{5EB00205} & \texttt{76E13DFD} & \texttt{661706EC} & \texttt{5AB0675B} & \texttt{AE90904F} & , \\
 & \texttt{1F4EB52A} & \texttt{EDE718DE} & \texttt{58286EED} & \texttt{CCDE6C20} & \texttt{8B54B778} & \texttt{40F9F9DC} & \texttt{11B2A618} & \texttt{AF3B75D9} &), \\
(& \texttt{A8E2A7E} & \texttt{E2F11666} & \texttt{0CDCCEB0} & \texttt{B40EAA0D} & \texttt{99AD9341} & \texttt{12FBF99D} & \texttt{D3FA3D38} & \texttt{8EC352E2} & , \\
 & \texttt{10A7E639} & \texttt{E503B3CA} & \texttt{A01C4664} & \texttt{E346826F} & \texttt{FC8AFFE6} & \texttt{5FD9A191} & \texttt{357ED027} & \texttt{75022146} &), \\
(& \texttt{95166E8} & \texttt{1083838F} & \texttt{2984628C} & \texttt{A69541F6} & \texttt{E663BCF8} & \texttt{36C8FA8D} & \texttt{B30129D7} & \texttt{6D4DB2B0} & , \\
 & \texttt{A30BBFD} & \texttt{03B51510} & \texttt{28E188D6} & \texttt{6782EA7C} & \texttt{623F8944} & \texttt{E140A8A5} & \texttt{9182EA92} & \texttt{7FFA5313} &), \\
(& \texttt{1A1DD37D} & \texttt{E17552E3} & \texttt{9DC59495} & \texttt{B22BDAFD} & \texttt{C76680CD} & \texttt{A6438B3A} & \texttt{3ED2E6FF} & \texttt{E3F4C88C} & , \\
 & \texttt{CE78EBF} & \texttt{CD937D0D} & \texttt{3DA8DDEA} & \texttt{2997A5EC} & \texttt{C74AF144} & \texttt{8281C19E} & \texttt{323E1D34} & \texttt{51BE2FCC} &), \\
(& \texttt{A521803} & \texttt{8FEB75A0} & \texttt{ECBCADDB} & \texttt{101D5845} & \texttt{57D02D17} & \texttt{DEFD860F} & \texttt{1F07C36F} & \texttt{86AAFE08} & , \\
 & \texttt{5CED7AC} & \texttt{9CB8CC58} & \texttt{8E9D849C} & \texttt{4C2579F2} & \texttt{D502B80A} & \texttt{BFCC89C8} & \texttt{BB2E981A} & \texttt{8659D335} &))
\end{array}$$

3. Compute the header $\texttt{Hdr} = (C_0, C_1, \ldots, C_A)$

3.1. Compute $C_0 = t \cdot Q = (x_0(tQ) + \lambda x_1(tQ), y_0(tQ) + \lambda y_1(tQ))$ in E_2.

$$\begin{array}{rllllllllll}
\overline{Q} = ((& \texttt{93E4D9B} & \texttt{A200D5F4} & \texttt{67F8DE35} & \texttt{1FA89F68} & \texttt{796C7E0D} & \texttt{99B00CBF} & \texttt{43443696} & \texttt{0B1B642B} & , \\
 & \texttt{4DC3A8C} & \texttt{ECBF3FEE} & \texttt{72B7C0B0} & \texttt{FA79756A} & \texttt{6BBB1B0C} & \texttt{F1BDE9A0} & \texttt{EE7B2C99} & \texttt{E48E1280} &), \\
(& \texttt{16B996C7} & \texttt{AD4F692A} & \texttt{1A14B760} & \texttt{53C4FA1A} & \texttt{5C9C0596} & \texttt{86564D72} & \texttt{CEE1630F} & \texttt{9217EF2D} & , \\
 & \texttt{1A99B357} & \texttt{71D91184} & \texttt{58B5E67E} & \texttt{0C995A6F} & \texttt{25F77C75} & \texttt{DEC3F1E2} & \texttt{0AA487BB} & \texttt{BFF5AAFD} &))
\end{array}$$

$$\begin{array}{rcl}
x_0(tQ) & = & 14683718403430397383651068670799253120392864380488596340794839425547650743668 \\
x_1(tQ) & = & 5876108720253712425975055069753186417552636673121682208728768528428762357 19 \\
y_0(tQ) & = & 12345884364359289892839126196511324727723362040834026907725768603669504953886 \\
y_1(tQ) & = & 2868016710552886558321378170385074207758067915475594883772648099880875096561
\end{array}$$

As an octet string, we have:

$$\begin{array}{rclllllllll}
\overline{x_0(tQ)} & = & \texttt{2076B0AA} & \texttt{2B6EE972} & \texttt{1E999B4F} & \texttt{19BE8C88} & \texttt{B312D24A} & \texttt{E6B65E50} & \texttt{4AB77863} & \texttt{7BC43D74} \\
\overline{x_1(tQ)} & = & \texttt{14C9372} & \texttt{9B763F57} & \texttt{22D614A3} & \texttt{A1EE9CCD} & \texttt{0E844EDF} & \texttt{27CDDF7E} & \texttt{78B85878} & \texttt{720DEFC7} \\
\overline{y_0(tQ)} & = & \texttt{1B4B85DE} & \texttt{33171D20} & \texttt{32B4C549} & \texttt{14A65949} & \texttt{182AD020} & \texttt{BCD4FDB1} & \texttt{747E06C8} & \texttt{0EFB8E1E} \\
\overline{y_1(tQ)} & = & \texttt{6573D6C} & \texttt{3B06AEE5} & \texttt{C3613FBD} & \texttt{F4734AA2} & \texttt{6FCC69BE} & \texttt{BC50D3F2} & \texttt{2E68EDCE} & \texttt{DFF879F1}
\end{array}$$

3.2. Compute $h = H(t \cdot Q)$.

$h = 98199306462589948048616479283485563760012384702830991890937488386064463556 65$

As an octet string, we have:

$\overline{h}$ = `15B5E23F 86370FF0 E5DB7CE2 A81638BB 54953BD4 005A5C67 3879E0BF 21C964D1`

3.3. Compute $h \cdot P_1 = (x_{hP_1}, y_{hP_1})$ in E_1.

$\overline{P_1}$ = (`135C07D1 249FDEF8 BFF2DFD0 90A2F79B 8FDA147B 86A7F99C A7D8CEE9 9E09CAF1` ,
`3E3CD12 5C7A3F46 870613F2 A7924EAB C0DFE58D 548944D9 29349F33 AC29FFF8`)

$x_{hP_1} = 3170247396668770457812063486796320621599007255368600206444700761645009615 92$

$y_{hP_1} = 1628728484284127282062764608111777591393080299299693254784768675382085203716$

As an octet string, we have:

$\overline{x_{hP_1}}$ = `B36DFD 2496B4B2 7DAB81CD 09B9AF15 5BE545A9 3CF56D16 5951FE7C 4A853938`

$\overline{y_{hP_1}}$ = `399D3E1 1DF7F673 83F02CF5 83EF5E28 D7315263 8E238697 F7EBF1E4 0AC6B704`

3.4. Compute $Sum = \sum_{j \in S} P_{n+1-j} = (x_{Sum}, y_{Sum})$ in E_1.

$x_{Sum} = 2988370176706151363890319061919387014040865976603272276272656967731513889 90$

$y_{Sum} = 7810082091837573323364271346764246124426463725710398563915414934598710908927$

As an octet string, we have:

$\overline{x_{Sum}}$ = `A922C1 0A3B6DCB F6705E96 1C38A4F9 EB5002F5 34E495D0 72A3F538 1E73F53E`

$\overline{y_{Sum}}$ = `1144598D 52FD3452 6AAF50AC EB4E7CAF E2C9FCF7 DBC8EB20 A02D3392 937AD7FF`

3.5. Compute $h \cdot P_1 + V + Sum$ in E_1.

$\overline{V}$ = (`F276328 A897017D 73ED95AD D017D4CD 3B8766FB C519765F 3200CEA8 EF1FB6ED` ,
`D9C306F 8AA5032B 212FFDC9 00EB27E1 72EA27B9 85591D4A 1D7CF993 CB4DA242`)

$x_{hP_1+V+Sum} = 7295369918326400462191947385659321018089109673933347887775371090039752403128$

$y_{hP_1+V+Sum} = 1867730271618052702336957365710878374291114517780513562538803047159559102598$

As an octet string, we have:

$\overline{x_{hP_1+V+Sum}}$ = `10210875 B9D8B1F7 26C89576 81CFBBD1 B5FB6B65 2E9C58AB 2AA11499 EE3CA8B8`

$\overline{y_{hP_1+V+Sum}}$ = `421190E 1CF4A030 E6D11439 55D09750 306D2522 94205908 FB5DBD8D 4FD02086`

3.6. Compute $C_1 = t \cdot (h \cdot P_1 + V + Sum) = (x_{C_1}, y_{C_1})$ in E_1.

$x_{C_1} = 1348073546068760520081780411529350401116644894990236135688956149145445436601$

$y_{C_1} = 2128129062043655935337628690307024329709655975436456314198354002296617476 05$

As an octet string, we have:

$\overline{x_{C_1}}$ = `2FAFB8A E2F66A3C 8302FAA0 CB086586 C1A36F4E E01D1C25 092D304C 55B18CB9`

$\overline{y_{C_1}}$ = `7872A5 68091BB9 65340A75 3088ECF6 DB6B8D5B 6067778E 7BE14157 E050A995`

Output: The pair $(K,$ `Hdr`$)$.

E encrypts a message with K, adds the `Hdr` to the message and broadcasts all.

5.3.2 Decrypt

R can compute the session key, iff he is authorized.

Input: The user i, the secret key D_i, the public key PK_i, the set S (set of the authorized users), the header `Hdr` and the hash function H.

Action: Find the session key K iff the i-th user is authorized.

Here, $i = 5$.

1. Compute the pairing $e_T(C_1, Q_i)$. We denote $K_1 = e_T(C_1, Q_5)$.

$$
\begin{array}{llll}
\overline{C_1} & = (& \texttt{2FAFB8A E2F66A3C 8302FAA0 CB086586 C1A36F4E E01D1C25 092D304C 55B18CB9} & , \\
 & & \texttt{7872A5 68091BB9 65340A75 3088ECF6 DB6B8D5B 6067778E 7BE14157 E050A995} &) \\
\overline{Q_5} & = ((& \texttt{109F3690 B88AA8A6 A3BFBA59 B748160F 336CF286 74EF398C 0FAA0FB1 23119712} & , \\
 & & \texttt{6115275 F81690E1 BEC455A9 CAE8260B BEA83A7D 64DD72BE 3AB40648 07E2086E} &), \\
 & (& \texttt{215B8C3F E9D71760 0281CDF1 E830D8F2 C50D8914 F87ECB3F 97D1BF50 ADB93DA1} & , \\
 & & \texttt{111D4FC4 0611F31C 06F40938 A3AE6913 CA7273D9 465F46A7 9939AA01 D92A11C5} &)) \\
K_1 & = ((& 15993188120222354521396488415082849825782315805084947729938197632810617101241 & , \\
 & & 5817505896253371077764186803721541389516915219966618322203910490540579957947 &), \\
 & (& 1228834733156814923897398066293602706403685110399773654790411358291566514996 & , \\
 & & 12875303878690364970873307994829737006079907601890663600811891215964836989745 &), \\
 & (& 10544220102252797106913839050689481643476899362367304575496419781546375132811 & , \\
 & & 6503295853601703075864383007727928845860011398389442367616167393294140360081 &), \\
 & (& 928432584113618347055200091637563715524648630472036654081976753010588413041 & , \\
 & & 8220381208938361721537712403711675446777344882607345528950489278022947227136 &), \\
 & (& 2335491823599471511008315678577350052524834900329364960220508068641535764919 & , \\
 & & 10622281577123752545452785294580224010381909084284658902780806438186869132023 &), \\
 & (& 9677648059316594087622930760478245196155350425446317436400950989677155361800 & , \\
 & & 7132407142921609835694857762583830750525887711537583606734140481389450417415 &))
\end{array}
$$

As an octet string, we have:

$$
\begin{array}{llll}
\overline{K_1} & = ((& \texttt{235BD2DD 72408E9D 49DFCDF3 0C1C4B68 01AE883C 41D0211F D057C008 BDD35BB9} & , \\
 & & \texttt{CDC9776 853B2039 4E366471 7D5DAC37 65B50FB7 5B1B637B 7A763207 362B2CBB} &), \\
 & (& \texttt{2B77EED 81087FDA D410DB79 C9B15328 69C5636B 72B86449 D3A3E623 5493D334} & , \\
 & & \texttt{1C7729EB 635927B6 7DA0B10A E927362B 5698F0BE 9A122A96 C6EC9690 92100B31} &), \\
 & (& \texttt{174FD12C 9324CC9F 04C11C26 E4380F17 D72D2B1B 86652784 4C0D04B7 4614368B} & , \\
 & & \texttt{E60BC29 CC98014C FFA04050 9AF49BE9 B4B65B76 CCBE3399 17BAAAF7 5D04C591} &), \\
 & (& \texttt{20D7966 83D25AA6 58CB123E 412494F7 D5E9FA2A 0F4AF0A9 28D8C7C6 BCCE6C71} & , \\
 & & \texttt{122C9225 6971EEB4 88C85936 AF3B03B6 D6D506A9 929C5B98 3CFAF447 EA44C600} &), \\
 & (& \texttt{529D76E 5EA828F9 CEDB8FC0 DF498BC1 C11636ED D74E2C60 DD625F27 9A3129B7} & , \\
 & & \texttt{177BFF91 9A1D658C D36AA49E D097EFE0 19D5B943 415B105A 00E86B5A BC150AF7} &), \\
 & (& \texttt{15655ACD 0EDC8F5B 148F5B4D 6A101CFF EF954D7C 3F91D756 F26E2064 25A8A408} & , \\
 & & \texttt{FC4CCA4 6FF4075E 4FE2BD9A C8391485 EB3FD4EF C59E9457 4E93658D 5236D507} &))
\end{array}
$$

2. Compute $h = H(C_0)$.

$$
\begin{array}{llll}
\overline{C_0} & = ((& \texttt{2076B0AA 2B6EE972 1E999B4F 19BE8C88 B312D24A E6B65E50 4AB77863 7BC43D74} & , \\
 & & \texttt{14C9372 9B763F57 22D614A3 A1EE9CCD 0E844EDF 27CDDF7E 78B85878 720DEFC7} &), \\
 & (& \texttt{1B4B85DE 33171D20 32B4C549 14A65949 182AD020 BCD4FDB1 747E06C8 0EFB8E1E} & , \\
 & & \texttt{6573D6C 3B06AEE5 C3613FBD F4734AA2 6FCC69BE BC50D3F2 2E68EDCE DFF879F1} &)) \\
h & = & 9819930646258994804861647928348556376001238470283099189093748838606446355665 &
\end{array}
$$

As an octet string, we have:

$$
\overline{h} = \texttt{15B5E23F 86370FF0 E5DB7CE2 A81638BB 54953BD4 005A5C67 3879E0BF 21C964D1}
$$

3. Compute $Sum = \sum_{j \in S \setminus \{i\}} P_{n+1-j+i}$ in E_1.

$$
\begin{array}{lll}
x_{Sum} & = & 60085549313211170317419206389277779364329890976918063666823969053462792375 35 \\
y_{Sum} & = & 31323780289077838237811418326123006558481238004552290258660388961214417644 44
\end{array}
$$

As an octet string, we have:

$$
\begin{array}{lll}
\overline{x_{Sum}} & = & \texttt{D48B8B9 147A916E E7E8E469 8A174831 13CD85FC 91924975 BA8289C2 D595679F} \\
\overline{y_{Sum}} & = & \texttt{6ECDCF6 82D2CAC5 B463ABCC BBA731ED 8A601A44 34B5DE3A 59729107 5895445C}
\end{array}
$$

4. Compute $h \cdot P_{i+1} + D_i + Sum$ in E_1.

$$
\begin{array}{llll}
\overline{P_6} & = (& \texttt{179B6130 4AE3C46D EFCB42AC 241C8304 4F9ED0FA 06108A2E E5B09233 B4F41363} & , \\
 & & \texttt{1AC20CAD FDBF0AB4 055D8C8B 17498528 ADA81086 0C006C21 40DDCD88 562D4241} &) \\
\overline{D_5} & = (& \texttt{2753116 528BEBA3 0E64CEAB 673FAAA1 B7AA4C0A 6A0B8DC3 26FD3BBD 8CF28F27} & , \\
 & & \texttt{210A2C82 61AD15B0 23D51C34 20E58108 973D4B6A F6D6BCC5 934CFF85 AD2BD95C} &)
\end{array}
$$

$$
\begin{array}{lcl}
x_{hP+D+Sum} & = & 1573459351300283877351921381576294013204570923559479124140493086941171061408\,9 \\
y_{hP+D+Sum} & = & 6666363255519675415617439368147875457061512425653508816034245666952157918304
\end{array}
$$

As an octet string, we have:

$$
\begin{array}{lcl}
\overline{x_{hP+D+Sum}} & = & \texttt{22C976DE}\quad\texttt{5EB60BDA}\quad\texttt{E7B9337F}\quad\texttt{A7ED7C5F}\quad\texttt{3F0622DC}\quad\texttt{7078C57D}\quad\texttt{5A684FB0}\quad\texttt{8EB9BA49} \\
\overline{y_{hP+D+Sum}} & = & \texttt{EBD0723}\quad\texttt{E6A183CF}\quad\texttt{DDB8617C}\quad\texttt{2E6124FA}\quad\texttt{967CE6EF}\quad\texttt{2E572CD7}\quad\texttt{B23A4177}\quad\texttt{F5B4F460}
\end{array}
$$

5. Compute the pairing $K_2 = e_T(h \cdot P_{i+1} + D_i + Sum, C_0)$.

$$
\begin{array}{rlll}
K_2 & = ((& 5176509948500844553144938550397200380279129144765769192109842573345216207 12 & , \\
 & & 8366774127230815174821100597277916170519470463325525960219132593649261053 71 &), \\
 & (& 2071767981992078205207414535850595984907830772548964741292378537128014592119 & , \\
 & & 12845944116646451795037337769375809249004986454952660502462514659209083939229 &), \\
 & (& 1283614883569997407493096514238523193160611897954903244390478818046133844414 & , \\
 & & 7048230179206804001713793542999497331473173258808952259369166647843572175189 &), \\
 & (& 862665294298729294593816719671665076358963610567486729577189230800818992723 & , \\
 & & 6924174079089927991874329970154190339383022217576781570118555155771981869973 &), \\
 & (& 868748554497401984935023422769027414025981412591579797154163164856838581965 & , \\
 & & 4038808909697898677952795934316351218766386911272184070009902575083740270261 &), \\
 & (& 3312777779692060868622966284280828652184071781422035634218572933428060470488 & , \\
 & & 1469995520611360615980548356302226895955305285234710271396047244506466582179 &))
\end{array}
$$

As an octet string, we have:

$$
\begin{array}{rlll}
\overline{K_2} & = ((& \texttt{124FAE6}\quad\texttt{8A424979}\quad\texttt{A3E78E2D}\quad\texttt{C65C1212}\quad\texttt{55AFBE03}\quad\texttt{C1B21CAB}\quad\texttt{F1C20118}\quad\texttt{5823E8E8} & , \\
 & & \texttt{1D98AEA}\quad\texttt{0DDBEAD3}\quad\texttt{B1DF74A1}\quad\texttt{9E3F2034}\quad\texttt{A39D3BEF}\quad\texttt{02D56BC4}\quad\texttt{E347D61A}\quad\texttt{51EC271B} &), \\
 & (& \texttt{4949441}\quad\texttt{6A8078F0}\quad\texttt{2608BEC5}\quad\texttt{8F7F444B}\quad\texttt{F2534F50}\quad\texttt{79133090}\quad\texttt{4F823FE3}\quad\texttt{60CB3877} & , \\
 & & \texttt{1C668BF5}\quad\texttt{49DC63C7}\quad\texttt{093DC01A}\quad\texttt{41E4E59F}\quad\texttt{AABC3A4C}\quad\texttt{1AE0B46A}\quad\texttt{04B735E0}\quad\texttt{E20CE99D} &), \\
 & (& \texttt{2D68012}\quad\texttt{36678C72}\quad\texttt{A1ED0754}\quad\texttt{7B118B49}\quad\texttt{7F3F77F1}\quad\texttt{1A8DF0B9}\quad\texttt{4F9AE516}\quad\texttt{C9B2B9BE} & , \\
 & & \texttt{F95282B}\quad\texttt{525DEF66}\quad\texttt{C156AF55}\quad\texttt{632BFF53}\quad\texttt{936444B3}\quad\texttt{6140ED56}\quad\texttt{F04B154A}\quad\texttt{A9B85155} &), \\
 & (& \texttt{1E84052}\quad\texttt{27EDED0B}\quad\texttt{EF195282}\quad\texttt{77795276}\quad\texttt{7BBC8609}\quad\texttt{DCFE8FCA}\quad\texttt{04B11EFC}\quad\texttt{6364F653} & , \\
 & & \texttt{F4EF192}\quad\texttt{D36EB052}\quad\texttt{DFEE2DDD}\quad\texttt{8DCD42C2}\quad\texttt{FB0E9B6E}\quad\texttt{C6C319AA}\quad\texttt{9CEAA576}\quad\texttt{FD54B395} &), \\
 & (& \texttt{1EBB1BA}\quad\texttt{D142CF85}\quad\texttt{B1418348}\quad\texttt{0AD485B3}\quad\texttt{05DA3289}\quad\texttt{CE1BAB13}\quad\texttt{F085FA9B}\quad\texttt{28F8EECD} & , \\
 & & \texttt{8EDE284}\quad\texttt{7E36F45F}\quad\texttt{696D1963}\quad\texttt{60E94029}\quad\texttt{E476C2E9}\quad\texttt{9B2B9EAD}\quad\texttt{F410B3BD}\quad\texttt{6BB90EB5} &), \\
 & (& \texttt{752F73D}\quad\texttt{29B6B4A0}\quad\texttt{17E793E6}\quad\texttt{0BAA404F}\quad\texttt{73482035}\quad\texttt{063F9873}\quad\texttt{CA35F6BF}\quad\texttt{69E5BCD8} & , \\
 & & \texttt{33FFCEC}\quad\texttt{43EDB9FE}\quad\texttt{9850BA60}\quad\texttt{969371A8}\quad\texttt{A7B84CC8}\quad\texttt{4D3924F0}\quad\texttt{5109249A}\quad\texttt{49977EA3} &))
\end{array}
$$

6. Compute the inversion in $\mathbb{F}_{p^k}$ of $e_T(h \cdot P_{i+1} + D_i + Sum, C_0)$.

$$
\begin{array}{rlll}
K_2^{-1} & = ((& 517650994850084455314493855039720038027912914476576919210984257334521620712 & , \\
 & & 836677412723081517482110059727791617051947046332552596021913259364926105371 &), \\
 & (& 14211494567013377526499039702409401184541199736724311879871634794828007403164 & , \\
 & & 3437318432359003936669116468884187920444044054320616118701498672746938056054 &), \\
 & (& 1283614883569997407493096514238523193160611897954903244390478818046133844414 & , \\
 & & 7048230179206804001713793542999497331473173258808952259369166647843572175189 &), \\
 & (& 15420597254706726437112637518588332093090066898705789891586824101155203002560 & , \\
 & & 9359088469915527739832124268105806830066008291696495051045458176184040125310 &), \\
 & (& 868748554497401984935023422769027414025981412591579797154163164856838581965 & , \\
 & & 4038808909697898677952795934316351218766386911272184070009902575083740270261 &), \\
 & (& 12970484769313394863083487953979168517264958727851240986945440398527961524795 & , \\
 & & 14813267028394095115725905881957770273493725224038566349767966087449555413104 &))
\end{array}
$$

As an octet string, we have:

$$\begin{array}{llr}
\overline{K_2^{-1}} = ((& \texttt{124FAE6 8A424979 A3E78E2D C65C1212 55AFBE03 C1B21CAB F1C20118 5823E8E8} & , \\
 & \texttt{1D98AEA 0DDBEAD3 B1DF74A1 9E3F2034 A39D3BEF 02D56BC4 E347D61A 51EC271B} &), \\
(& \texttt{1F6B6BBE 9592A5EE 29F74509 5D1805DC A3F9DD3B 750C4BC0 C3D3022B 083EEE9C} & , \\
 & \texttt{799740A B636BB17 46C243B4 AAB26488 EB90F23F D33EC7E7 0E9E0C2D 86FD3D76} &), \\
(& \texttt{2D68012 36678C72 A1ED0754 7B118B49 7F3F77F1 1A8DF0B9 4F9AE516 C9B2B9BE} & , \\
 & \texttt{F95282B 525DEF66 C156AF55 632BFF53 936444B3 6140ED56 F04B154A A9B85155} &), \\
(& \texttt{2217BFAD D82531D2 60E6B14C 751DF7B2 1A90A682 1120EC87 0EA42312 05A530C0} & , \\
 & \texttt{14B10E6D 2CA46E8B 7011D5F1 5ECA0765 9B3E911D 275C62A6 766A9C97 6BB5737E} &), \\
(& \texttt{1EBB1BA D142CF85 B1418348 0AD485B3 05DA3289 CE1BAB13 F085FA9B 28F8EECD} & , \\
 & \texttt{8EDE284 7E36F45F 696D1963 60E94029 E476C2E9 9B2B9EAD F410B3BD 6BB90EB5} &), \\
(& \texttt{1CAD08C2 D65C6A3E 38186FE8 E0ED09D9 23050C56 E7DFE3DD 491F4B4E FF246A3B} & , \\
 & \texttt{20C00313 BC2564DF B7AF496E 5603D87F EE94DFC3 A0E65760 C24C1D74 1F72A870} &))
\end{array}$$

7. Compute the session key $K = K_1 \times K_2^{-1}$.

$$\begin{array}{llr}
K = ((& 1585174373219231923941953934221543074612861392737095144364452424513086305519 7 & , \\
 & 88964873113606985509959752671135585004379669327940262320391755416077634312 8 &), \\
(& 12037842088831789779616191475014861979929773078218346774178920372628676972623 & , \\
 & 14160762750594582105477570069553145214478758194540557570250206321795099489753 &), \\
(& 4774314063230795283083022089330029861723540802859340599464528445081483694818 & , \\
 & 7533657999645012088599131890629134987052206856772698173922220349554237251910 &), \\
(& 4214640484070544390987442854920516298107458763480060598740731815887148790448 & , \\
 & 4609234592765812759452770632213436055962959385731458605706325380756569477907 &), \\
(& 11812832290259535277470755555230935107566457455001078577067181772682195945612 & , \\
 & 5836881073943754161349170569829534899472883303611971872139591176869239467980 &), \\
(& 4668175683093174496980419363674715123181508420467288717110239261030500466184 & , \\
 & 2627023267340804916059718489033769300713026889045362798962457590818678625077 &))
\end{array}$$

As an octet string, we have:

$$\begin{array}{llr}
\overline{K} = ((& \texttt{230BC4DD 817497B2 E2B74F91 B46F9B09 212D7E83 7C399629 0FCBE17C BDC2115D} & , \\
 & \texttt{1F785F9 746E8A30 AB22682B 20CA4F66 FFF05944 0FDC7EB1 9147BA13 44ED6E58} &), \\
(& \texttt{1A9D2D5B 2B42CAAD 461D99F6 5EB00205 76E13DFD 661706EC 5AB0675B AE90904F} & , \\
 & \texttt{1F4EB52A EDE718DE 58286EED CCDE6C20 8B54B778 40F9F9DC 11B2A618 AF3B75D9} &), \\
(& \texttt{A8E2A7E E2F11666 0CDCCEB0 B40EAA0D 99AD9341 12FBF99D D3FA3D38 8EC352E2} & , \\
 & \texttt{10A7E639 E503B3CA A01C4664 E346826F FC8AFFE6 5FD9A191 357ED027 75022146} &), \\
(& \texttt{95166E8 1083838F 2984628C A69541F6 E663BCF8 36C8FA8D B30129D7 6D4DB2B0} & , \\
 & \texttt{A30BBFD 03B51510 28E188D6 6782EA7C 623F8944 E140A8A5 9182EA92 7FFA5313} &), \\
(& \texttt{1A1DD37D E17552E3 9DC59495 B22BDAFD C76680CD A6438B3A 3ED2E6FF E3F4C88C} & , \\
 & \texttt{CE78EBF CD937D0D 3DA8DDEA 2997A5EC C74AF144 8281C19E 323E1D34 51BE2FCC} &), \\
(& \texttt{A521803 8FEB75A0 ECBCADDB 101D5845 57D02D17 DEFD860F 1F07C36F 86AAFE08} & , \\
 & \texttt{5CED7AC 9CB8CC58 8E9D849C 4C2579F2 D502B80A BFCC89C8 BB2E981A 8659D335} &))
\end{array}$$

Output: The key K.

R can decrypt the ciphertext with K.

5.4 Example 2: Test with 50/100 authorized users

For this example, only the i-th users, where i is odd, are authorized.
For example: 5 is authorized and 8 is revoked.

5.4.1 Encrypt

E generates a session key to encrypt a message and the header, R can compute the session key, iff R is authorized.

Input: S set of authorized users, public key PK_s and hash function H.

Action: Generate a session key K and a header key `Hdr`

1. Generate an integer t

   1.1. Randomly or pseudorandomly select an integer t in the interval $[1, m-1]$.

   t = 5710657168379116176003428016857198065066034451013474592937188291088088041169

   1.2. Convert t to the octet string $\bar{t}$.

   $\bar{t}$ = `CA01E0E EF2718A2 A90C3636 FCC04963 F9B9CFA1 22F216B3 D300A198 5CE006D1`

2. Compute the session key K.

   2.1 Compute the pairing $e_T(P_{n+1}, Q)$ We can use P_n and Q_1.

$\overline{P_n}$ = (`1F2E354E DF838381 FB886C67 ECFDE49E D1E04EC5 BB201D0B 9F8DDA3F 886F6BE3` ,
`729F5F3 44F16BC9 C3833794 28F13899 94B218BC D9710596 104FE566 130839CC`)

$\overline{Q_1}$ = ((`1415FE8 B1FE4B90 F7E6C5B3 5D716171 73CAC5DB 94517E7D 6F93AC73 A7F22C5B` ,
`7C8953A A7FEE45F 168397CA 14117A15 421F071F 9756EA2F A3F43C85 1571B9FB`),
(`1CD9FD2D 1A70BAD6 8AD193FE F734073B 6B589AFE 272CFD09 D66A10C2 A2AE3A80` ,
`27F28D7 F4F737C5 181D8188 C6F2F8AC D4F3965A 7AE4F427 4F580DCE C8D30434`))

$e_T(P_n, Q_1)$ = ((9777006672766793355569318537895760242288532693135997144224010579608079171503 ,
3736224127587849951207374140465494525783296894277339955308027604152345576535),
(2034617478678334330114287231256676256529198327912042265372082307973047752582 ,
5038203151404631215916969864982186096569109661221215159734364820370518154378),
(1521139847720600344495516242465234524670070815494222964997130067772954527114 ,
3992586121504756341504873227277175827562094454075522566015033329571147714786),
(15241427258801996725372979128513686955119404666059012223506760614915576032858 ,
145420087133900650406556921688926195370190973630123616890122328239612895717),
(9291245618952431096054272516026793305152077614557855985349622539553855200880 ,
16188971139605893944883252994981336385916587324616518086389293519646187783112),
(12040400999163887538212377340089328065927347824189722347923869229369290080663 ,
2127812259550993495072584731037300935704889386208043623713155084905253792095))

As an octet string, we have:

$\overline{e_T(P_n, Q_1)}$ = ((`159D96F4 DC00B050 0CD063A5 0BA4BEB9 3050FDDB 6F41C859 497A1F12 43EBFFAF` ,
`842A0BF 24DA4C95 3BE3EB7B 561CA417 4CFDC272 8636B6B0 8529DDBE 2D124457`),
(`47F8D7C A97F4307 04ECC764 071B1564 D6575BE3 4CF31E68 1EE8D187 B2A19786` ,
`B23859D 2D102D17 C15BA066 37EE8B49 B18EF3FB 2BE3EDB4 F5482300 A4A79C8A`),
(`35CEF44 CACE42A7 56C5F090 06C7DBA7 AD2D4ECA 31D7B934 9D6DDDAF 94A2458A` ,
`8D3B941 FD875DEF DED652E7 FB8E7CC7 7712AB32 A437C4FD 2E3EB72E 476290E2`),
(`21B25795 560640B3 F1A07FD1 5788AC20 9959ECDF 3D553C07 2AA7BE01 AACDEE5A` ,
`524E0A D0F96838 2BB2A793 C0B30252 8F860FDD 4B01CE3F A2884B06 F1C821E5`),
(`148AA89D F941B9DB 2849F3E3 71C5F5D5 0841FAC6 B5DA47E6 AB08DDAE CE1F5E70` ,
`23CAA209 3E47299A 2BDA7210 3C631C3B A12F80F5 A262D022 56F17D8F 505303C8`),
(`1A9EA01E 6DACB6C8 E9783621 798C5B53 07EF8513 E252745B 3A03E996 2B4B4197` ,
`4B44C8F 34D31EF4 C9C28E17 2DA58CB1 4E80397E ABA28E67 2201DFA1 5791695F`))

   2.2. Compute the exponentiation in $\mathbb{F}_{p^k}$: $K = (e_T(P_{n+1}, Q))^t$

K = ((15851743732192319239419539342215430746128613927370951443644524245130863055197 ,
889648731136069855099597526711355850043796693279402623203917554160776343128),
(12037842088831789779616191475014861979929773078218346774178920372628676972623 ,
14160762750594582105477570069553145214478758194540557570250206321795099489753),
(4774314063230795283083022089330029861723540802859340599464528445081483694818 ,
7533657999645012088599131890629134987052206856772698173922220349554237251910),
(4214640484070544390987442854920516298107458763480060598740731815887148790448 ,
4609234592765812759452770632213436055962959385731458605706325380756569477907),
(11812832290259535277470755555230935107566457455001078577067181772682195945612 ,
5836881073943754161349170569829534899472883303611971872139591176869239467980),
(4668175683093174496980419363674715123181508420467288717110239261030500466184 ,
2627023267340804916059718489033769300713026889045362798962457590818678625077))

As an octet string, we have:

$$\overline{K} = \begin{array}{llllllllll} ((& \texttt{230BC4DD} & \texttt{817497B2} & \texttt{E2B74F91} & \texttt{B46F9B09} & \texttt{212D7E83} & \texttt{7C399629} & \texttt{0FCBE17C} & \texttt{BDC2115D} & , \\ & \texttt{1F785F9} & \texttt{746E8A30} & \texttt{AB22682B} & \texttt{20CA4F66} & \texttt{FFF05944} & \texttt{0FDC7EB1} & \texttt{9147BA13} & \texttt{44ED6E58} &), \\ (& \texttt{1A9D2D5B} & \texttt{2B42CAAD} & \texttt{461D99F6} & \texttt{5EB00205} & \texttt{76E13DFD} & \texttt{661706EC} & \texttt{5AB0675B} & \texttt{AE90904F} & , \\ & \texttt{1F4EB52A} & \texttt{EDE718DE} & \texttt{58286EED} & \texttt{CCDE6C20} & \texttt{8B54B778} & \texttt{40F9F9DC} & \texttt{11B2A618} & \texttt{AF3B75D9} &), \\ (& \texttt{A8E2A7E} & \texttt{E2F11666} & \texttt{0CDCCEB0} & \texttt{B40EAA0D} & \texttt{99AD9341} & \texttt{12FBF99D} & \texttt{D3FA3D38} & \texttt{8EC352E2} & , \\ & \texttt{10A7E639} & \texttt{E503B3CA} & \texttt{A01C4664} & \texttt{E346826F} & \texttt{FC8AFFE6} & \texttt{5FD9A191} & \texttt{357ED027} & \texttt{75022146} &), \\ (& \texttt{95166E8} & \texttt{1083838F} & \texttt{2984628C} & \texttt{A69541F6} & \texttt{E663BCF8} & \texttt{36C8FA8D} & \texttt{B30129D7} & \texttt{6D4DB2B0} & , \\ & \texttt{A30BBFD} & \texttt{03B51510} & \texttt{28E188D6} & \texttt{6782EA7C} & \texttt{623F8944} & \texttt{E140A8A5} & \texttt{9182EA92} & \texttt{7FFA5313} &), \\ (& \texttt{1A1DD37D} & \texttt{E17552E3} & \texttt{9DC59495} & \texttt{B22BDAFD} & \texttt{C76680CD} & \texttt{A6438B3A} & \texttt{3ED2E6FF} & \texttt{E3F4C88C} & , \\ & \texttt{CE78EBF} & \texttt{CD937D0D} & \texttt{3DA8DDEA} & \texttt{2997A5EC} & \texttt{C74AF144} & \texttt{8281C19E} & \texttt{323E1D34} & \texttt{51BE2FCC} &), \\ (& \texttt{A521803} & \texttt{8FEB75A0} & \texttt{ECBCADDB} & \texttt{101D5845} & \texttt{57D02D17} & \texttt{DEFD860F} & \texttt{1F07C36F} & \texttt{86AAFE08} & , \\ & \texttt{5CED7AC} & \texttt{9CB8CC58} & \texttt{8E9D849C} & \texttt{4C2579F2} & \texttt{D502B80A} & \texttt{BFCC89C8} & \texttt{BB2E981A} & \texttt{8659D335} &)) \end{array}$$

3. Compute the header $\texttt{Hdr} = (C_0, C_1, \ldots, C_A)$

3.1. Compute $C_0 = t \cdot Q = (x_0(tQ) + \lambda x_1(tQ), y_0(tQ) + \lambda y_1(tQ))$ in E_2.

$$\overline{Q} = \begin{array}{llllllllll} ((& \texttt{93E4D9B} & \texttt{A200D5F4} & \texttt{67F8DE35} & \texttt{1FA89F68} & \texttt{796C7E0D} & \texttt{99B00CBF} & \texttt{43443696} & \texttt{0B1B642B} & , \\ & \texttt{4DC3A8C} & \texttt{ECBF3FEE} & \texttt{72B7C0B0} & \texttt{FA79756A} & \texttt{6BBB1B0C} & \texttt{F1BDE9A0} & \texttt{EE7B2C99} & \texttt{E48E1280} &), \\ (& \texttt{16B996C7} & \texttt{AD4F692A} & \texttt{1A14B760} & \texttt{53C4FA1A} & \texttt{5C9C0596} & \texttt{86564D72} & \texttt{CEE1630F} & \texttt{9217EF2D} & , \\ & \texttt{1A99B357} & \texttt{71D91184} & \texttt{58B5E67E} & \texttt{0C995A6F} & \texttt{25F77C75} & \texttt{DEC3F1E2} & \texttt{0AA487BB} & \texttt{BFF5AAFD} &)) \end{array}$$

$$\begin{array}{lcl} x_0(tQ) & = & 14683718403430397383651068670799253120392864380488596340794839425547650743668 \\ x_1(tQ) & = & 587610872025371242597505506975318641755263667312168220872876852842876235719 \\ y_0(tQ) & = & 12345884364359289892839126196511324727723362040834026907725768603669504953886 \\ y_1(tQ) & = & 2868016710552886558321378170385074207758067915475594883772648099880875096561 \end{array}$$

As an octet string, we have:

$$\begin{array}{lclllllllll} \overline{x_0(tQ)} & = & \texttt{2076B0AA} & \texttt{2B6EE972} & \texttt{1E999B4F} & \texttt{19BE8C88} & \texttt{B312D24A} & \texttt{E6B65E50} & \texttt{4AB77863} & \texttt{7BC43D74} \\ \overline{x_1(tQ)} & = & \texttt{14C9372} & \texttt{9B763F57} & \texttt{22D614A3} & \texttt{A1EE9CCD} & \texttt{0E844EDF} & \texttt{27CDDF7E} & \texttt{78B85878} & \texttt{720DEFC7} \\ \overline{y_0(tQ)} & = & \texttt{1B4B85DE} & \texttt{33171D20} & \texttt{32B4C549} & \texttt{14A65949} & \texttt{182AD020} & \texttt{BCD4FDB1} & \texttt{747E06C8} & \texttt{0EFB8E1E} \\ \overline{y_1(tQ)} & = & \texttt{6573D6C} & \texttt{3B06AEE5} & \texttt{C3613FBD} & \texttt{F4734AA2} & \texttt{6FCC69BE} & \texttt{BC50D3F2} & \texttt{2E68EDCE} & \texttt{DFF879F1} \end{array}$$

3.2. Compute $h = H(t \cdot Q)$.

$$h = 98199306462589948048616479283485563760012384702830991890937488386064463556 65$$

As an octet string, we have:

$$\overline{h} = \texttt{15B5E23F} \quad \texttt{86370FF0} \quad \texttt{E5DB7CE2} \quad \texttt{A81638BB} \quad \texttt{54953BD4} \quad \texttt{005A5C67} \quad \texttt{3879E0BF} \quad \texttt{21C964D1}$$

3.3. Compute $h \cdot P_1 = (x_{hP_1}, y_{hP_1})$ in E_1.

$$\overline{P_1} = \begin{array}{llllllllll} (& \texttt{135C07D1} & \texttt{249FDEF8} & \texttt{BFF2DFD0} & \texttt{90A2F79B} & \texttt{8FDA147B} & \texttt{86A7F99C} & \texttt{A7D8CEE9} & \texttt{9E09CAF1} & , \\ & \texttt{3E3CD12} & \texttt{5C7A3F46} & \texttt{870613F2} & \texttt{A7924EAB} & \texttt{C0DFE58D} & \texttt{548944D9} & \texttt{29349F33} & \texttt{AC29FFF8} &) \end{array}$$

$$\begin{array}{lcl} x_{hP_1} & = & 317024739666877045781206348679632062159900725536860020644470076164500961592 \\ y_{hP_1} & = & 1628728484284127282062764608111777591393080299299693254784768675382085203716 \end{array}$$

As an octet string, we have:

$$\begin{array}{lclllllllll} \overline{x_{hP_1}} & = & \texttt{B36DFD} & \texttt{2496B4B2} & \texttt{7DAB81CD} & \texttt{09B9AF15} & \texttt{5BE545A9} & \texttt{3CF56D16} & \texttt{5951FE7C} & \texttt{4A853938} \\ \overline{y_{hP_1}} & = & \texttt{399D3E1} & \texttt{1DF7F673} & \texttt{83F02CF5} & \texttt{83EF5E28} & \texttt{D7315263} & \texttt{8E238697} & \texttt{F7EBF1E4} & \texttt{0AC6B704} \end{array}$$

3.4. Compute $Sum = \sum_{j \in S} P_{n+1-j} = (x_{Sum}, y_{Sum})$ in E_1.

$$\begin{array}{lcl} x_{Sum} & = & 853317625250902777763144281929356879661479227011041364652582756071880110627 \\ y_{Sum} & = & 13521723848236722074874649781731344934704182588988790232516080163579481757917 \end{array}$$

As an octet string, we have:

$$\begin{array}{lclllllllll} \overline{x_{Sum}} & = & \texttt{1E2F5ED} & \texttt{D73887A2} & \texttt{51CC6BE9} & \texttt{D91AAD6B} & \texttt{CAF8FC81} & \texttt{6B0AF79B} & \texttt{7B2DCD5F} & \texttt{AE4DB223} \\ \overline{y_{Sum}} & = & \texttt{1DE50644} & \texttt{A8633F22} & \texttt{5E87E376} & \texttt{5706746A} & \texttt{580913A5} & \texttt{0F862132} & \texttt{29044C4E} & \texttt{2B7F18DD} \end{array}$$

3.5. Compute $h \cdot P_1 + V + Sum$ in E_1.

$$\overline{V} = \begin{array}{llllllllll} (& \texttt{F276328} & \texttt{A897017D} & \texttt{73ED95AD} & \texttt{D017D4CD} & \texttt{3B8766FB} & \texttt{C519765F} & \texttt{3200CEA8} & \texttt{EF1FB6ED} & , \\ & \texttt{D9C306F} & \texttt{8AA5032B} & \texttt{212FFDC9} & \texttt{00EB27E1} & \texttt{72EA27B9} & \texttt{85591D4A} & \texttt{1D7CF993} & \texttt{CB4DA242} &) \end{array}$$

$x_{hP_1+V+Sum}$ = 13596869651920697016390769392163103872484805924947510646082429151285487492592

$y_{hP_1+V+Sum}$ = 11552084512297115856253386062932806884965694650717445844301121656701077311110

As an octet string, we have:

$\overline{x_{hP_1+V+Sum}}$ = `1E0F8E35 6E141425 9D964BD4 19C59B84 6051AEA5 F93452A3 25FE4A93 622A5DF0`

$\overline{y_{hP_1+V+Sum}}$ = `198A3F85 434282CF BB28DFC5 DD73FC04 1C7CFFCB B393D0FA 8D2FD68E F1564E86`

3.6. Compute $C_1 = t \cdot (h \cdot P_1 + V + Sum) = (x_{C_1}, y_{C_1})$ in E_1.

x_{C_1} = 5785890678566517660256972785601277500814701582546427671542193685402664181085

y_{C_1} = 4513383763853343306747436220861402806790428805276481117403379083343843276903

As an octet string, we have:

$\overline{x_{C_1}}$ = `CCAB2B4 EBE374C3 8043C17A 4CC1CCE8 8B989BB2 FF864503 CA9CEFE9 BE121D5D`

$\overline{y_{C_1}}$ = `9FA7C14 2C4821E0 E42B7DDB CC92B048 A27F3232 C79BC8E5 A43FCB63 1497B867`

Output: The pair $(K, \texttt{Hdr})$.

E encrypts a message with K, adds the `Hdr` to the message and broadcasts all.

5.4.2 Decrypt

R can compute the session key, iff he is authorized.

Input: The user i, the secret key D_i, the public key PK_i, the set S (set of the authorized users), the header `Hdr` and the hash function H.

Action: Find the session key K iff the i-th user is authorized.

Here, $i = 5$.

1. Compute the pairing $e_T(C_1, Q_i)$. We denote $K_1 = e_T(C_1, Q_5)$.

$\overline{C_1}$ = (`2FAFB8A E2F66A3C 8302FAA0 CB086586 C1A36F4E E01D1C25 092D304C 55B18CB9` ,
`7872A5 68091BB9 65340A75 3088ECF6 DB6B8D5B 6067778E 7BE14157 E050A995`)

$\overline{Q_5}$ = ((`109F3690 B88AA8A6 A3BFBA59 B748160F 336CF286 74EF398C 0FAA0FB1 23119712` ,
`6115275 F81690E1 BEC455A9 CAE8260B BEA83A7D 64DD72BE 3AB40648 07E2086E`),
(`215B8C3F E9D71760 0281CDF1 E830D8F2 C50D8914 F87ECB3F 97D1BF50 ADB93DA1` ,
`111D4FC4 0611F31C 06F40938 A3AE6913 CA7273D9 465F46A7 9939AA01 D92A11C5`))

K_1 = ((255547694901569998840311918059188787828925913304628383250746056292449136487 ,
14986018521133729266200730666699827878837395403191809004295891997631563659453),
(14756912283190495590466832622512979891254129963524210664306543891687804668481 ,
3965109996919648175107018191962948391617880248582945052907319259436721549640),
(15061055373167802386394966760881094012934973729906953824558400922087214755993 ,
14748100868261716468099063314275711879559808585064247993014623300547487885222),
(14941378612798835683558430298394764186074671094229932929972206569250899255480 ,
8344760354468622632288357689993005186094374422601840486816090515336685740487),
(9567588099413523710372143961824378281085460147742467096114350908103713803648 ,
13376922627353575243586120111317132336204757069686424833443123842630458462621),
(12747267670850038714559490269073448419411553231116713133468359928423556233842 ,
6315532508551802957345067466666207676328665842007472896905808948594517141223))

As an octet string, we have:

$$
\begin{array}{rlllllllllll}
\overline{K_1} & = ((& \texttt{90A286} & \texttt{D135EACA} & \texttt{0B0EFD8A} & \texttt{EC526257} & \texttt{B80DDA8F} & \texttt{E26EBDA8} & \texttt{D92512CB} & \texttt{02E82F67} & , \\
 & & \texttt{2121C930} & \texttt{BAA2EB03} & \texttt{FF38079F} & \texttt{B6D1F42F} & \texttt{C7CBB76A} & \texttt{240168AD} & \texttt{53DFB21F} & \texttt{B19378BD} &), \\
 & (& \texttt{20A01DCA} & \texttt{0F452FB3} & \texttt{1B9839A8} & \texttt{C7A93A4C} & \texttt{3D8F5698} & \texttt{5E564D5F} & \texttt{93A2A3C4} & \texttt{F66A8A41} & , \\
 & & \texttt{8C42C37} & \texttt{E323D383} & \texttt{3FC44638} & \texttt{A3039A0D} & \texttt{A8D21AB1} & \texttt{FE29F7A4} & \texttt{1378537C} & \texttt{2DC6B148} &), \\
 & (& \texttt{214C4158} & \texttt{428DF34E} & \texttt{9AC9FF66} & \texttt{B78663D6} & \texttt{5E55A0A6} & \texttt{52391B7D} & \texttt{B4DDE49A} & \texttt{B7033099} & , \\
 & & \texttt{209B2118} & \texttt{83E0C713} & \texttt{D0D1950C} & \texttt{EEA12863} & \texttt{6BC6CCBA} & \texttt{A21F942C} & \texttt{025EB0AF} & \texttt{F26A67A6} &), \\
 & (& \texttt{21088546} & \texttt{80E0BEA5} & \texttt{B0EB58D0} & \texttt{24426536} & \texttt{B4E298EC} & \texttt{6C975101} & \texttt{50FA7543} & \texttt{43AE60B8} & , \\
 & & \texttt{1272F78C} & \texttt{53FCC380} & \texttt{E6E833C4} & \texttt{761C88B3} & \texttt{72C3690F} & \texttt{68498524} & \texttt{851ED5D8} & \texttt{BC60E1C7} &), \\
 & (& \texttt{1527101E} & \texttt{4181A031} & \texttt{A4EC506B} & \texttt{20D99390} & \texttt{B9580C36} & \texttt{D244C83B} & \texttt{A5D75793} & \texttt{9032CD80} & , \\
 & & \texttt{1D9311E4} & \texttt{DDD2B247} & \texttt{E9728544} & \texttt{49A4F1A3} & \texttt{15309133} & \texttt{027B7B7F} & \texttt{6D40EE31} & \texttt{0532D19D} &), \\
 & (& \texttt{1C2EB2A4} & \texttt{7DE3E83B} & \texttt{E904B770} & \texttt{8195A1B1} & \texttt{B5E2E446} & \texttt{3446B7AE} & \texttt{0AC7C7D1} & \texttt{7B92BA72} & , \\
 & & \texttt{DF676F8} & \texttt{41808123} & \texttt{B8D4643E} & \texttt{160BA152} & \texttt{4A8A47DD} & \texttt{A7DE389B} & \texttt{FB6337D7} & \texttt{9242E6E7} &))
\end{array}
$$

2. Compute $h = H(C_0)$.

$$
\begin{array}{rlllllllllll}
\overline{C_0} & = ((& \texttt{2076B0AA} & \texttt{2B6EE972} & \texttt{1E999B4F} & \texttt{19BE8C88} & \texttt{B312D24A} & \texttt{E6B65E50} & \texttt{4AB77863} & \texttt{7BC43D74} & , \\
 & & \texttt{14C9372} & \texttt{9B763F57} & \texttt{22D614A3} & \texttt{A1EE9CCD} & \texttt{0E844EDF} & \texttt{27CDDF7E} & \texttt{78B85878} & \texttt{720DEFC7} &), \\
 & (& \texttt{1B4B85DE} & \texttt{33171D20} & \texttt{32B4C549} & \texttt{14A65949} & \texttt{182AD020} & \texttt{BCD4FDB1} & \texttt{747E06C8} & \texttt{0EFB8E1E} & , \\
 & & \texttt{6573D6C} & \texttt{3B06AEE5} & \texttt{C3613FBD} & \texttt{F4734AA2} & \texttt{6FCC69BE} & \texttt{BC50D3F2} & \texttt{2E68EDCE} & \texttt{DFF879F1} &))
\end{array}
$$

$$h = 9819930646258994804861647928348556376001238470283099189093748838606446355665$$

As an octet string, we have:

$$\overline{h} = \texttt{15B5E23F}\ \ \texttt{86370FF0}\ \ \texttt{E5DB7CE2}\ \ \texttt{A81638BB}\ \ \texttt{54953BD4}\ \ \texttt{005A5C67}\ \ \texttt{3879E0BF}\ \ \texttt{21C964D1}$$

3. Compute $Sum = \sum_{j \in S \setminus \{i\}} P_{n+1-j+i}$ in E_1.

$$
\begin{array}{rcl}
x_{Sum} & = & 10432544163617539274381073465547698407593613728859338926191215566465231525656 \\
y_{Sum} & = & 45308993453510403251887152802461974804435856145663032994465399596133950800006
\end{array}
$$

As an octet string, we have:

$$
\begin{array}{rcllllllll}
\overline{x_{Sum}} & = & \texttt{17109C59} & \texttt{CA42E756} & \texttt{8AAB85E1} & \texttt{91337EAB} & \texttt{5A5038D5} & \texttt{2E7EEB78} & \texttt{2129AB19} & \texttt{0DF89718} \\
\overline{y_{Sum}} & = & \texttt{A0465ED} & \texttt{32FFE0ED} & \texttt{46FF7F23} & \texttt{49F48B9E} & \texttt{323A8CE5} & \texttt{43A4A6A3} & \texttt{9976F7CA} & \texttt{F9909346}
\end{array}
$$

4. Compute $h \cdot P_{i+1} + D_i + Sum$ in E_1.

$$
\begin{array}{rlllllllllll}
\overline{P_6} & = (& \texttt{179B6130} & \texttt{4AE3C46D} & \texttt{EFCB42AC} & \texttt{241C8304} & \texttt{4F9ED0FA} & \texttt{06108A2E} & \texttt{E5B09233} & \texttt{B4F41363} & , \\
 & & \texttt{1AC20CAD} & \texttt{FDBF0AB4} & \texttt{055D8C8B} & \texttt{17498528} & \texttt{ADA81086} & \texttt{0C006C21} & \texttt{40DDCD88} & \texttt{562D4241} &) \\
\overline{D_5} & = (& \texttt{2753116} & \texttt{528BEBA3} & \texttt{0E64CEAB} & \texttt{673FAAA1} & \texttt{B7AA4C0A} & \texttt{6A0B8DC3} & \texttt{26FD3BBD} & \texttt{8CF28F27} & , \\
 & & \texttt{210A2C82} & \texttt{61AD15B0} & \texttt{23D51C34} & \texttt{20E58108} & \texttt{973D4B6A} & \texttt{F6D6BCC5} & \texttt{934CFF85} & \texttt{AD2BD95C} &)
\end{array}
$$

$$
\begin{array}{rcl}
x_{hP+D+Sum} & = & 11905509215636884338601590263203909922552689390743074919200654366156331153144 \\
y_{hP+D+Sum} & = & 11390392740075150787765972305305097103864781844984529978147182834497274647121
\end{array}
$$

As an octet string, we have:

$$
\begin{array}{rcllllllll}
\overline{x_{hP+D+Sum}} & = & \texttt{1A524788} & \texttt{18D8FEE7} & \texttt{C96D0F43} & \texttt{611EEE13} & \texttt{B7B978C6} & \texttt{E9829ACB} & \texttt{41F931F7} & \texttt{B5A7F6F8} \\
\overline{y_{hP+D+Sum}} & = & \texttt{192EBBDC} & \texttt{1A31B592} & \texttt{AD593EAF} & \texttt{E1E05F27} & \texttt{F79153A7} & \texttt{F358C741} & \texttt{B1917C89} & \texttt{BD6A1251}
\end{array}
$$

5. Compute the pairing $K_2 = e_T(h \cdot P_{i+1} + D_i + Sum, C_0)$.

$$
\begin{array}{rlll}
K_2 & = ((& 44729219482915953926103855368945458342211363009097859556689432046420280532 33 & , \\
 & & 102030267878316497403294489750903644570374824528975914603473475954882681424 30 &), \\
 & (& 27937465274190493597086703265305930393969635337060198055786957561364486945 77 & , \\
 & & 72428388253533452312945800381633496000491888879042087029176405521511865393 29 &), \\
 & (& 153886841591835561568378458295749519034311866473690795819322095617372825937 26 & , \\
 & & 75757428320983206312677216332690253633622102888765196486591849780332315212 5 &), \\
 & (& 54314870574731215504531375615645595753721078031053015304440518509016056525 16 & , \\
 & & 100989899908513765739532858252323468185804246622105115222741828801150662367 21 &), \\
 & (& 102378655579003397980992652111355973845153923812324915868754326075976523320 42 & , \\
 & & 12778497065639674065998337070658588276699592069066807036774870876159676498 57 &), \\
 & (& 76795681097231400471852770852097303913238061565484358965355034223349747962 61 & , \\
 & & 75746037795745796671862735784043970410951060741677943056930997226728726287 9 &))
\end{array}
$$

As an octet string, we have:

$$\begin{array}{lllllllllll}
\overline{K_2} & =((& \mathtt{9E39588} & \mathtt{057FAE2E} & \mathtt{A0AB641F} & \mathtt{E2E2018F} & \mathtt{237C703F} & \mathtt{1750A5C4} & \mathtt{83BD181F} & \mathtt{A764E2F1} & , \\
 & & \mathtt{168EB561} & \mathtt{2BA84C98} & \mathtt{2C851C38} & \mathtt{6E84FC0D} & \mathtt{8426A474} & \mathtt{D282E3E2} & \mathtt{EDED8D8D} & \mathtt{C6CC475E} &), \\
 & (& \mathtt{62D345A} & \mathtt{0796B985} & \mathtt{20402D2A} & \mathtt{4FC2FE48} & \mathtt{9C4B51DE} & \mathtt{DA3C7CFF} & \mathtt{2C240E8A} & \mathtt{133A9931} & , \\
 & & \mathtt{10034D2F} & \mathtt{31C31349} & \mathtt{EF074CF6} & \mathtt{79D39725} & \mathtt{51B478A2} & \mathtt{65E2B570} & \mathtt{AA25151A} & \mathtt{8B8A0341} &), \\
 & (& \mathtt{2205AFC3} & \mathtt{57A0CCC1} & \mathtt{80E3B19F} & \mathtt{BEEF346B} & \mathtt{2DFFDB02} & \mathtt{8E18DB30} & \mathtt{C26F3644} & \mathtt{AD6E23BE} & , \\
 & & \mathtt{1ACC597} & \mathtt{E8AA14D5} & \mathtt{6D318120} & \mathtt{DA2D3EB6} & \mathtt{2512AE70} & \mathtt{43915CA9} & \mathtt{F941E496} & \mathtt{EC7196FD} &), \\
 & (& \mathtt{C021CDC} & \mathtt{02741D3F} & \mathtt{09439E27} & \mathtt{9D2C659B} & \mathtt{54F73D47} & \mathtt{B7C0DAF8} & \mathtt{F3AC8172} & \mathtt{392D1824} & , \\
 & & \mathtt{1653D365} & \mathtt{E5E11ED0} & \mathtt{50E570AB} & \mathtt{A9EC2E66} & \mathtt{C399219A} & \mathtt{F126DCA7} & \mathtt{7ABECC1C} & \mathtt{31496F31} &), \\
 & (& \mathtt{16A26D32} & \mathtt{F9060B90} & \mathtt{2801B6D2} & \mathtt{3CAD1509} & \mathtt{966BD839} & \mathtt{CB43A699} & \mathtt{416ABE65} & \mathtt{713FCE0A} & , \\
 & & \mathtt{2D33CBF} & \mathtt{E993D29A} & \mathtt{49232885} & \mathtt{D6317718} & \mathtt{24CCC13D} & \mathtt{A6A0D702} & \mathtt{1FB46719} & \mathtt{61126041} &), \\
 & (& \mathtt{10FA7B44} & \mathtt{AC0D76EA} & \mathtt{2468B692} & \mathtt{F8C858D2} & \mathtt{C152DEBC} & \mathtt{6AAFED7A} & \mathtt{FDDF7704} & \mathtt{7D27A1E5} & , \\
 & & \mathtt{1ACB516} & \mathtt{ED9FB592} & \mathtt{1278E881} & \mathtt{4524EA33} & \mathtt{65C52344} & \mathtt{D380D35B} & \mathtt{71934209} & \mathtt{2AE5369F} &))
\end{array}$$

6. Compute the inversion in $\mathbb{F}_{p^k}$ of $e_T(h \cdot P_{i+1} + D_i + Sum, C_0)$.

$$\begin{array}{llll}
K_2^{-1} & =((& 4472921948291595392610385536894545834221136300909785955668943204642028053233 & , \\
 & & 1020302678783164974032944897509036445703748245289759146034734759548826814243 0 &), \\
 & (& 1348951602158640637199778391172940413005206697556725681558531757581957330070 6 & , \\
 & & 904042372365211050041187420009664756939984162136906791824637277980483545595 4 &), \\
 & (& 1538868415918355615683784582957495190343118664736907958193220956173728259372 6 & , \\
 & & 757574283209832063126772163326902536336221028887651964865918497803323152125 &), \\
 & (& 1085177549153233418125331667669543759407692270616797509071996148105441634276 7 & , \\
 & & 618427255815407915775316841302765035086860584706276509888983045184095575856 2 &), \\
 & (& 1023786555790033979809926521113559738451539238123249158687543260759765233204 2 & , \\
 & & 127784970656396740659983370706585882766995920690668070367748708761596764985 7 &), \\
 & (& 860369443928231568452117715305026677812522435272484072462850990962104719902 2 & , \\
 & & 1552580217104799776498782688041955746533951990185649719059470335968873473240 4 &))
\end{array}$$

As an octet string, we have:

$$\begin{array}{lllllllllll}
\overline{K_2^{-1}} & =((& \mathtt{9E39588} & \mathtt{057FAE2E} & \mathtt{A0AB641F} & \mathtt{E2E2018F} & \mathtt{237C703F} & \mathtt{1750A5C4} & \mathtt{83BD181F} & \mathtt{A764E2F1} & , \\
 & & \mathtt{168EB561} & \mathtt{2BA84C98} & \mathtt{2C851C38} & \mathtt{6E84FC0D} & \mathtt{8426A474} & \mathtt{D282E3E2} & \mathtt{EDED8D8D} & \mathtt{C6CC475E} &), \\
 & (& \mathtt{1DD2CBA5} & \mathtt{F87C6559} & \mathtt{2FBFD6A4} & \mathtt{9CD44BDF} & \mathtt{FA01DAAD} & \mathtt{13E2FF51} & \mathtt{E7313384} & \mathtt{55CF8DE2} & , \\
 & & \mathtt{13FCB2D0} & \mathtt{CE500B94} & \mathtt{60F8B6D8} & \mathtt{72C3B303} & \mathtt{4498B3E9} & \mathtt{883CC6E0} & \mathtt{69302CF3} & \mathtt{DD8023D2} &), \\
 & (& \mathtt{2205AFC3} & \mathtt{57A0CCC1} & \mathtt{80E3B19F} & \mathtt{BEEF346B} & \mathtt{2DFFDB02} & \mathtt{8E18DB30} & \mathtt{C26F3644} & \mathtt{AD6E23BE} & , \\
 & & \mathtt{1ACC597} & \mathtt{E8AA14D5} & \mathtt{6D318120} & \mathtt{DA2D3EB6} & \mathtt{2512AE70} & \mathtt{43915CA9} & \mathtt{F941E496} & \mathtt{EC7196FD} &), \\
 & (& \mathtt{17FDE323} & \mathtt{FD9F019F} & \mathtt{46BC65A7} & \mathtt{4F6AE48D} & \mathtt{4155EF44} & \mathtt{365EA158} & \mathtt{1FA8C09C} & \mathtt{2FDD0EEF} & , \\
 & & \mathtt{DAC2C9A} & \mathtt{1A32000D} & \mathtt{FF1A9323} & \mathtt{42AB1BC1} & \mathtt{D2B40AF0} & \mathtt{FCF89FA9} & \mathtt{989675F2} & \mathtt{37C0B7E2} &), \\
 & (& \mathtt{16A26D32} & \mathtt{F9060B90} & \mathtt{2801B6D2} & \mathtt{3CAD1509} & \mathtt{966BD839} & \mathtt{CB43A699} & \mathtt{416ABE65} & \mathtt{713FCE0A} & , \\
 & & \mathtt{2D33CBF} & \mathtt{E993D29A} & \mathtt{49232885} & \mathtt{D6317718} & \mathtt{24CCC13D} & \mathtt{A6A0D702} & \mathtt{1FB46719} & \mathtt{61126041} &), \\
 & (& \mathtt{130584BB} & \mathtt{5405A7F4} & \mathtt{2B974D3B} & \mathtt{F3CEF155} & \mathtt{D4FA4DCF} & \mathtt{836F8ED6} & \mathtt{1575CB09} & \mathtt{EBE2852E} & , \\
 & & \mathtt{22534AE9} & \mathtt{1273694C} & \mathtt{3D871B4D} & \mathtt{A7725FF5} & \mathtt{30880947} & \mathtt{1A9EA8F5} & \mathtt{A1C20005} & \mathtt{3E24F074} &))
\end{array}$$

7. Compute the session key $K = K_1 \times K_2^{-1}$.

$$\begin{array}{llll}
K & =((& 1585174373219231923941953934221543074612861392737095144364452424513086305519 7 & , \\
 & & 889648731136069855099597526711355850043796693279402623203917554160776343128 &), \\
 & (& 1203784208883178977961619147501486197992977307821834677417892037262867697262 3 & , \\
 & & 1416076275059458210547757006955314521447875819454055757025020632179509948975 3 &), \\
 & (& 477431406323079528308302208933002986172354080285934059946452844508148369481 8 & , \\
 & & 753365799964501208859913189062913498705220685677269817392222034955423725191 0 &), \\
 & (& 421464048407054439098744285492051629810745876348006059874073181588714879044 8 & , \\
 & & 460923459276581275945277063221343605596295938573145860570632538075656947790 7 &), \\
 & (& 1181283229025953527747075555523093510756645745500107857706718177268219594561 2 & , \\
 & & 583688107394375416134917056982953489947288330361197187213959117686923946798 0 &), \\
 & (& 466817568309317449698041936367471512318150842046728871711023926103050046618 4 & , \\
 & & 262702326734080491605971848903376930071302688904536279896245759081867862507 7 &))
\end{array}$$

As an octet string, we have:

$$\begin{array}{lllllllllll}
\overline{K} & = ((& \texttt{230BC4DD} & \texttt{817497B2} & \texttt{E2B74F91} & \texttt{B46F9B09} & \texttt{212D7E83} & \texttt{7C399629} & \texttt{0FCBE17C} & \texttt{BDC2115D} & , \\
 & & \texttt{1F785F9} & \texttt{746E8A30} & \texttt{AB22682B} & \texttt{20CA4F66} & \texttt{FFF05944} & \texttt{0FDC7EB1} & \texttt{9147BA13} & \texttt{44ED6E58} &), \\
 & (& \texttt{1A9D2D5B} & \texttt{2B42CAAD} & \texttt{461D99F6} & \texttt{5EB00205} & \texttt{76E13DFD} & \texttt{661706EC} & \texttt{5AB0675B} & \texttt{AE90904F} & , \\
 & & \texttt{1F4EB52A} & \texttt{EDE718DE} & \texttt{58286EED} & \texttt{CCDE6C20} & \texttt{8B54B778} & \texttt{40F9F9DC} & \texttt{11B2A618} & \texttt{AF3B75D9} &), \\
 & (& \texttt{A8E2A7E} & \texttt{E2F11666} & \texttt{0CDCCEB0} & \texttt{B40EAA0D} & \texttt{99AD9341} & \texttt{12FBF99D} & \texttt{D3FA3D38} & \texttt{8EC352E2} & , \\
 & & \texttt{10A7E639} & \texttt{E503B3CA} & \texttt{A01C4664} & \texttt{E346826F} & \texttt{FC8AFFE6} & \texttt{5FD9A191} & \texttt{357ED027} & \texttt{75022146} &), \\
 & (& \texttt{95166E8} & \texttt{1083838F} & \texttt{2984628C} & \texttt{A69541F6} & \texttt{E663BCF8} & \texttt{36C8FA8D} & \texttt{B30129D7} & \texttt{6D4DB2B0} & , \\
 & & \texttt{A30BBFD} & \texttt{03B51510} & \texttt{28E188D6} & \texttt{6782EA7C} & \texttt{623F8944} & \texttt{E140A8A5} & \texttt{9182EA92} & \texttt{7FFA5313} &), \\
 & (& \texttt{1A1DD37D} & \texttt{E17552E3} & \texttt{9DC59495} & \texttt{B22BDAFD} & \texttt{C76680CD} & \texttt{A6438B3A} & \texttt{3ED2E6FF} & \texttt{E3F4C88C} & , \\
 & & \texttt{CE78EBF} & \texttt{CD937D0D} & \texttt{3DA8DDEA} & \texttt{2997A5EC} & \texttt{C74AF144} & \texttt{8281C19E} & \texttt{323E1D34} & \texttt{51BE2FCC} &), \\
 & (& \texttt{A521803} & \texttt{8FEB75A0} & \texttt{ECBCADDB} & \texttt{101D5845} & \texttt{57D02D17} & \texttt{DEFD860F} & \texttt{1F07C36F} & \texttt{86AAFE08} & , \\
 & & \texttt{5CED7AC} & \texttt{9CB8CC58} & \texttt{8E9D849C} & \texttt{4C2579F2} & \texttt{D502B80A} & \texttt{BFCC89C8} & \texttt{BB2E981A} & \texttt{8659D335} &))
\end{array}$$

Output: The key K.

R can decrypt the ciphertext with K.

6 Conclusion

We presented a detailed golden sequence for the PPSS scheme to encourage the diffusion and implementation of this scheme. Further work in this area is still possible to increase the computation efficiency. The setup needs consequent computing resources. It can be performed in reasonable time (no more than few minutes) on a standard PC. The decryption step can be performed on a smartphone if some optimizations are implemented (such as those described in [DGS12]). We recommend to use an optimal ate pairing on a BN curve and mostly to use precomputations for the sum (over the authorized users). Indeed the pairing and the sum computations are the bottleneck of this scheme. These example vectors were computed with the LibCryptoLCH, a proprietary library developed at Laboratoire Chiffre, Thales. For research development, we can suggest these two other libraries. The RELIC library [AG] has good performances for pairing computations. The recent work of Sanchez and Rodriguez [SRH13] provide also a library optimized for ARM smartphones.

References

[AG] D. F. Aranha and C. P. L. Gouvêa. RELIC is an Efficient LIbrary for Cryptography. `http://code.google.com/p/relic-toolkit/`.

[BCP97] Wieb Bosma, John Cannon, and Catherine Playoust. The Magma algebra system. I. The user language, 1997. Computational algebra and number theory (London, 1993) Version 2.19.6,`http://magma.maths.usyd.edu.au/calc/`.

[BGW05] Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In Victor Shoup, editor, *CRYPTO*, volume 3621 of *Lecture Notes in Computer Science*, pages 258–275. Springer, 2005.

[BN06] Paulo S.L.M. Barreto and Michael Naehrig. Pairing friendly elliptic curves of prime order. In *SAC 2005*, volume 3897 of *LNCS*, pages 319–331, 2006.

[DGS12] Renaud Dubois, Aurore Guillevic, and Marine Sengelin. Improved broadcast encryption scheme with constant-size ciphertext. In Michel Abdalla and Tanja Lange, editors, *Pairing*, volume 7708 of *Lecture Notes in Computer Science*, pages 196–202. Springer, 2012.

[Ecr07] EcryptII. Yearly report on algorithms and keysizes (2010-2011). European Network of Excellence in Cryptology II, 2007. ICT -2007-216676.

[HMA08] Federal information processing standard (FIPS) publication 198-1, the keyed-hash message authentification code (HMAC). *Cryptologia*, July 2008.

[HSV06] Florian Hess, Nigel P. Smart, and Frederik Vercauteren. The eta pairing revisited. *IEEE Transactions on Information Theory*, 52(10):4595–4602, 2006.

[PPSS12] Duong Hieu Phan, David Pointcheval, Siamak Fayyaz Shahandashti, and Mario Strefler. Adaptive cca broadcast encryption with constant-size secret keys and ciphertexts. In Willy Susilo, Yi Mu, and Jennifer Seberry, editors, *ACISP*, volume 7372 of *Lecture Notes in Computer Science*, pages 308–321. Springer, 2012.

[SRH13] Ana Helena Sánchez and Francisco Rodríguez-Henríquez. Neon implementation of an attribute-based encryption scheme. In Michael J. Jacobson Jr., Michael E. Locasto, Payman Mohassel, and Reihaneh Safavi-Naini, editors, *ACNS*, volume 7954 of *Lecture Notes in Computer Science*, pages 322–338. Springer, 2013.

[Ver10] Frederik Vercauteren. Optimal pairings. *IEEE Transactions on Information Theory*, 56(1):455–461, 2010.

A Notations

A.1 Mathematical Notations

The notations adopted in this document are listed in the following.

$\lceil x \rceil$	Ceiling: the smallest integer $\geq x$. For example, $\lceil 5 \rceil = 5$ and $\lceil 5.3 \rceil = 6$.
$\lfloor x \rfloor$	Floor: the largest integer $\leq x$. For example, $\lfloor 5 \rfloor = 5$ and $\lfloor 5.3 \rfloor = 5$.
$[x, y]$	The interval of integers between and including x and y.
mod	Modulo.
$\log_2$	The logarithm in basis 2. For example $\log_2(8) = 3$.
ECC	Elliptic Curve Cryptography.
E	An elliptic curve over the field $\mathbb{F}_q$ defined by a_E and b_E ($y^2 = x^3 + a_E x + b_E$).
$E(\mathbb{F}_q)$	The set of all points (with coordinates in $\mathbb{F}_q$) on an elliptic curve E defined over $\mathbb{F}_q$ and including the point at infinity O.
O	The point at infinity of an elliptic curve. This is the neutral element of the elliptic curve group.
$\#E(\mathbb{F}_q)$	If E is defined over $\mathbb{F}_q$, then $\#E(\mathbb{F}_q)$ denotes the number of points on the curve (including the point at infinity O). $\#E(\mathbb{F}_q)$ is the order of the curve E.
$\mathbb{F}_p$	The finite field of p elements, where p is prime.
$\mathbb{F}_q$	The finite field of q elements; in this document $q = p^2$ or $q = p^{12}$.
k	The embedding degree, here $k = 12$.
λ	An element of $\mathbb{F}_p$ such as $\sqrt{\lambda} \notin \mathbb{F}_p$ (λ is not a square in $\mathbb{F}_p$).
$\mathbb{F}_{p^2}$	The quadratic extension of $\mathbb{F}_p$, such that $\mathbb{F}_{p^2} \simeq \mathbb{F}_p[X]/(X^2 - \lambda)$.
β	An element of $\mathbb{F}_{p^2}$, such that $\sqrt{\beta} \notin \mathbb{F}_{p^2}$ and $\sqrt[3]{\beta} \notin \mathbb{F}_{p^2}$.
$\mathbb{F}_{p^{12}}$	The extension field degree 12 of $\mathbb{F}_p$ such that $\mathbb{F}_{p^{12}} \simeq \mathbb{F}_{p^2}[U]/(U^6 - \beta)$.
G_1	A subgroup of an elliptic curve.
G_2	Another subgroup of an elliptic curve.
π_p	The Frobenius map $\pi_p : E \to E : (x, y) \mapsto (x^p, y^p)$ for an elliptic curve defined over $\mathbb{F}_p$.
$\mathbb{G}_1$	$\mathbb{G}_1 = E[m] \cap \mathrm{Ker}(\pi_p - [1])$ used in the Ate Pairing definition.
$\mathbb{G}_2$	$\mathbb{G}_2 = E[m] \cap \mathrm{Ker}(\pi_p - [p])$ used in the Ate Pairing definition.
p	A odd prime number greater than 5.
$PPSS$	Phan-Pointcheval-Shahandashi-Stefler scheme, a broadcast scheme, [PPSS12].
E_1	A Barreto-Naehrig curve, defined over $\mathbb{F}_p$.
(a_{E_1}, b_{E_1})	The two coefficients which define the elliptic curve $E_1 : y^2 = x^3 + a_{E_1} x + b_{E_1}$.
m	The order of the curve E_1 over $\mathbb{F}_p$.
x	The integer unsed to compute E_1 parameters.

t_{E_1}	The trace of Frobenius of $E_1(\mathbb{F}_p)$.
P	A distinguished point on the elliptic curve $E_1(\mathbb{F}_p)$ named the base point or generator of the curve.
E_2	The twisted elliptic curve of degree 6 of $E_1(\mathbb{F}_{p^2})$, defined over $\mathbb{F}_{p^2}$ and of order a multiple of m ($\#E_2(\mathbb{F}_{p^2}) = m \cdot (p + t_{E_1} - 1)$).
(a_{E_2}, b_{E_2})	The pair which define the elliptic curve $E_2 : y^2 = x^3 + a_{E_2}x + b_{E_2}$.
Q	A distinguished point on the elliptic curve $E_2(\mathbb{F}_{p^2})$ of order m named the base point or generator (of the subgroup of order m).
$+$	This symbol corresponds to the group law on the elliptic curves.
$\cdot$	This symbol corresponds to a scalar multiplication of an elliptic curve point.
$\times$	This symbol corresponds to multiplication in $\mathbb{F}_q$.
α	A pseudo-random integer in $[1, m-1]$ used for the MSK.
γ	A pseudo-random integer in $[1, m-1]$ used for the MSK.
t	A pseudo-random integer in $[1, m-1]$ used for the session key.
P_i	A point on E_1 such that $P_i = \alpha^i \cdot P$.
V	A point on E_1 such that $V = \gamma \cdot P$.
Q_i	A point on E_2 of order m such that $Q_i = \alpha^i \cdot Q$.
x_P	The x-coordinate of a point P in decimal basis.
y_P	The y-coordinate of a point P in decimal basis.
$\overline{x_P}$	The x-coordinate of a point P in hexadecimal basis.
$\overline{y_P}$	The y-coordinate of a point P in hexadecimal basis.
X	The notation of X in decimal basis.
$\overline{X}$	The notation of X in hexadecimal basis.
E	The emitter center of the broadcast encryption.
R	The receiver or a user.
$n-1$	The maximal number of users in the scheme.
$B-1$	The number of users in one group.
A	The number of groups in the scheme.
MSK	The master secret key of the scheme, $MSK = (\alpha, \gamma)$.
PK_s	The public key of the scheme.
D_i	The secret key for the i-th user.
K	The session key.
`Hdr`	The header associated with a session key K.
C_0	The first part of the `Hdr`.
C_1	The second part of the `Hdr`.
$(H_\kappa)_\kappa$	The family of hash functions indexed by κ, here we use the key κ with the function HMAC_SHA256.
κ	A random integer defining the index of the hash function family (the key in HMAC_SHA256).
$H = H_\kappa$	The hash function, here we use HMAC_SHA256 in [HMA08] with κ as key.
h	The result of the hash function of $t \cdot Q$, ($h = H(t \cdot Q)$).
i	The index of the user.
a	The group number to which the user belongs ($1 \leq a \leq A$).
b	The position number in the group a ($0 \leq b \leq B-1$).
e	The pairing used in the PPSS scheme.
e_T	The Tate pairing explained in 2.2.
e_A	The ate pairing explained in 2.2.
e_{Opt}	The optimal ate pairing explained in 2.2.
S	The set of authorized users.
S_a	The set of authorized users in the a-th group of users.

γ_a	A pseudo-random integer in $[1, m-1]$, used for the MSK.
V_a	A point on $E_1(\mathbb{F}_p)$ such that $V_a = \gamma_a \cdot P$.

A.2 Notations of Elements in Finite Fields and Elliptic Curves

Let $x \in \mathbb{F}_p$, the notation is: x.
Let $x \in \mathbb{F}_{p^2}$, the notation is: $x_0 + \lambda x_1 = (x_0, x_1)$.
Let $x \in \mathbb{F}_{p^{12}}$, the notation is: $(x_{00} + x_{01} \times X) + (x_{10} + x_{11} \times X) \times U + (x_{20} + x_{21} \times X) \times U^2 + (x_{30} + x_{31} \times X) \times U^3 + (x_{40} + x_{41} \times X) \times U^4 + (x_{50} + x_{51} \times X) \times U^5 = ((x_{00}, x_{01}), (x_{10}, x_{11}), (x_{20}, x_{21}), (x_{30}, x_{31}), (x_{40}, x_{41}), (x_{50}, x_{51}))$.
Let $G \in E_1$, the notation is: (x, y).
Let $G \in E_2$, the notation is: $(x, y) = ((x_0, x_1), (y_0, y_1))$.

B Adaptation with group

In [BGW05], the authors propose to split the group of receivers in A groups of $B-1$ users. A user i is referenced by its group number (say a) and its range in that group (say b). Hence $i = \{a, b\}$ with $1 \leq a \leq A$ and $1 \leq b \leq B-1$. Let $n-1$ be the number of users. They propose to choose $B = \lfloor \sqrt{n-1} \rfloor + 1$ and $A = \lceil \frac{n-1}{B-1} \rceil$.

B.1 Set Up $_B(n-1)$:

E generates the master secret key and the public key for the scheme (MSK, PK_s).

Input: The elliptic curve domain parameters as specified in Section 4 and $n-1$ the number of users

Action: E selects the keys.

1. Choose the parameters B and A.
2. Generate an random integer α in $[1, m-1]$
3. Generate A random integers $(\gamma_1, \gamma_2, \ldots, \gamma_A)$ in $[1, m-1]$.
4. Compute the sequence P_i of E_1 for $i = 1, \ldots, B, B+2, \ldots, 2B$, such as $P_i = \alpha^i \cdot P$.
5. Compute the sequence V_i of E_1 for $i = 1, \ldots, A$, such as $V_i = \gamma_i \cdot P$.
6. Compute the sequence Q_i of E_2 for $i = 1, \ldots, B$ such as $Q_i = \alpha^i \cdot Q$.
7. Generate an random index κ to choose the H_κ function.
8. Store $PK_s = (P, P_1, P_2, \ldots, P_B, P_{B+2}, \ldots, P_{2B}, V_1, \ldots, V_A, Q, Q_1, \kappa)$.
9. Store the $MSK = (\alpha, \gamma_1, \ldots, \gamma_A)$.

Output: $PK_s = (P, P_1, P_2, \ldots, P_B, P_{B+2}, \ldots, P_{2B}, V_1, \ldots, V_A, Q, Q_1, \kappa)$, $MSK = (\alpha, \gamma_1, \ldots, \gamma_A)$ and the hash function H_κ.

B.2 Join (MSK, i):

E generates a secret key for R.

Input: The master secret key MSK and the number of the user $i \in [1, n-1]$.

Action: E generates a i-th secret key for R and the public key.

1. Compute $b = i \ \texttt{mod} \ B-1$ et $a = \lceil i/B-1 \rceil$ ($b \in [1, B-1]$ and $a \in [1, A]$

2. Compute the point $D_{a,b} = \gamma_a \cdot \alpha^b \cdot P \in E_1$.

Output: The elliptic curve point $D_{a,b}$ for the secret key and the public key: $PK_{a,b} = (P, P_1, \ldots, P_B, P_{B+2}, \ldots, P_{2B}, Q_b, \kappa)$

E gives $D_{a,b}$ and $PK_{a,b}$to R.

B.3 Encrypt(S, Pk_s, H_κ):

E generates a session key to encrypt a message and the header,such that R can compute the session key, iff R is authorized.

Input: S the set of the users who are authorized, the public key PK_s and the hash function H_κ.

Action: E generates a session key K and a header key `Hdr`

1. Generate an integer t.
2. Compute the session key K.
   - 2.1 Compute the pairing $e(P_{B+1}, Q)$
   - 2.2. Compute the exponentiation in $\mathbb{F}_{p^k}$: $K = (e(P_{B+1}, Q))^t$
3. Compute the header `Hdr` $= (C_0, C_1, \ldots, C_A)$
   - 3.1. Compute $C_0 = t \cdot Q$ in E_2.
   - 3.2. Compute $h = H_\kappa(t \cdot Q)$.
   - 3.3. Compute $h \cdot P_1 = (x_{hP_1}, y_{hP_1})$ in E_1.
   - 3.4. For each group of B users index by a:
     - 3.4.1 Compute $Sum_a = \sum_{j \in S_a} P_{B+1-j} = (x_{Sum_a}, y_{Sum_a})$ in E_1.
     - 3.4.2. Compute $h \cdot P_1 + V_a + Sum_a$ in E_1.
     - 3.4.3. Compute $C_a = t \cdot (h \cdot P_1 + V_a + Sum_a)$ in E_1.

Output: The pair $(K,$ `Hdr`$)$.

E encrypts a message with K, adds the `Hdr` to the message and broadcasts all.

B.4 Decrypt$(i = \{a, b\}, D_{a,b}, PK_{a,b}, S_a, \texttt{Hdr}, H_\kappa)$:

R can find the session key, if he is authorized.

Input: The user i, the secret key $D_{a,b}$,the public key $PK_{a,b}$, the set S_a (set of the authorized users in the group a) , the header `Hdr` and the hash function H_κ.

Action: Find the session key K if the i-th user is authorized.

1. Compute the pairing $K_1 = e(C_a, Q_b)$.
2. Compute $h = H_\kappa(C_0)$.
3. Compute $Sum = \sum_{j \in S_a \setminus \{i\}} P_{B+1-j+i}$ in E_1.
4. Compute $h \cdot P_{b+1} + D_{a,b} + Sum$ in E_1.
5. Compute the pairing $K_2 = e(h \cdot P_{b+1} + D_{a,b} + Sum, C_0)$.
6. Compute the inversion in $\mathbb{F}_{p^k}$ of $e(h \cdot P_{b+1} + D_{a,b} + Sum, C_0)$.
7. Compute the session key $K = K_1 \times K_2^{-1}$.

Output: The key K.

R can decrypt the ciphertext with K.

C Golden Sequence with Ate Pairing

We generate the test vectors with using the Ate pairing (explained in 2.2) as the asymmetric pairing. We have not rewritten the vector tests, who are the same than in section 5. The Set Up step and Join step are the same.

We choose $e(P, Q) = e_A(Q, P)$.

C.1 Example 1: Test with $100/100$ authorized users

In this example, all the users are authorized.

C.1.1 Encrypt

E generates a session key to encrypt a message and the header, R can compute the session key, iff R is authorized.

Input: S the set of the users who are authorized, the public key PK_s, and the hash function H.

Action: Generate a session key K and a header key `Hdr`

1. Generate an integer t
2. Compute the session key K.

   2.1 Compute the pairing $e_A(Q, P_{n+1})$ We can use P_n and Q_1.

$$\begin{array}{rlllllllllll}
\overline{Q_1} & = ((& \texttt{1415FE8} & \texttt{B1FE4B90} & \texttt{F7E6C5B3} & \texttt{5D716171} & \texttt{73CAC5DB} & \texttt{94517E7D} & \texttt{6F93AC73} & \texttt{A7F22C5B} & , \\
 & & \texttt{7C8953A} & \texttt{A7FEE45F} & \texttt{168397CA} & \texttt{14117A15} & \texttt{421F071F} & \texttt{9756EA2F} & \texttt{A3F43C85} & \texttt{1571B9FB} &), \\
 & (& \texttt{1CD9FD2D} & \texttt{1A70BAD6} & \texttt{8AD193FE} & \texttt{F734073B} & \texttt{6B589AFE} & \texttt{272CFD09} & \texttt{D66A10C2} & \texttt{A2AE3A80} & , \\
 & & \texttt{27F28D7} & \texttt{F4F737C5} & \texttt{181D8188} & \texttt{C6F2F8AC} & \texttt{D4F3965A} & \texttt{7AE4F427} & \texttt{4F580DCE} & \texttt{C8D30434} &)) \\
\overline{P_n} & = (& \texttt{1F2E354E} & \texttt{DF838381} & \texttt{FB886C67} & \texttt{ECFDE49E} & \texttt{D1E04EC5} & \texttt{BB201D0B} & \texttt{9F8DDA3F} & \texttt{886F6BE3} & , \\
 & & \texttt{729F5F3} & \texttt{44F16BC9} & \texttt{C3833794} & \texttt{28F13899} & \texttt{94B218BC} & \texttt{D9710596} & \texttt{104FE566} & \texttt{130839CC} &)
\end{array}$$

$$\begin{array}{rlll}
e_A(Q_1, P_n) & = ((& 155575523767650279012590966807782999771670192846655179080460278120009222253665 & , \\
 & & 111314439853487882904864078388230101522144686061708717458633019046490707776125 &), \\
 & (& 148674567091990683732989874011602838432988479776709256024081993654023438882038 & , \\
 & & 125522633014998553272324461969444281087450308523156805846571363754734219196 85 &), \\
 & (& 11533022147578867414039393595715168984896720225452543688544668465941160762 58 & , \\
 & & 132155379329894108049815330382654257368380354472463912251247852113939053176 74 &), \\
 & (& 51813026700998455488372842080933129818555132082494757049669368232620628342 17 & , \\
 & & 30761370511866057849902248033347265567847954975210970921879925910797169467 44 &), \\
 & (& 142985754205542895947925091158300634244297909896989222780362850032001953574 41 & , \\
 & & 86906004513560694474652210074283683394636406403215498116540286919821720609 57 &), \\
 & (& 26957845225231096766328423479191923640839545764235437913627426133816524363 19 & , \\
 & & 127935047458232142024869215922489383262228519139335780636658737471402397474051 &))
\end{array}$$

As an octet string, we have:

$$\begin{array}{rlllllllllll}
\overline{e_A(Q_1, P_n)} & = ((& \texttt{22654339} & \texttt{A7C75891} & \texttt{72C8D8C2} & \texttt{CFB20187} & \texttt{8DEF5F74} & \texttt{5CE9E779} & \texttt{069E16A1} & \texttt{93EEB961} & , \\
 & & \texttt{189C2C8C} & \texttt{F7028670} & \texttt{2BA1477D} & \texttt{727410D7} & \texttt{F85F270E} & \texttt{4E24350A} & \texttt{704CD040} & \texttt{44A30F3D} &), \\
 & (& \texttt{20DEAEAA} & \texttt{B559CD98} & \texttt{E66825E5} & \texttt{B3672BEE} & \texttt{2CD99403} & \texttt{6278CA5A} & \texttt{92176EA1} & \texttt{6BD5BD36} & , \\
 & & \texttt{1BC0544A} & \texttt{6E01A56A} & \texttt{903AD2D5} & \texttt{05857AC3} & \texttt{C7EE4D14} & \texttt{F99B522B} & \texttt{9819FFB7} & \texttt{D607A5C5} &), \\
 & (& \texttt{28CBEF4} & \texttt{AF917E11} & \texttt{902D3418} & \texttt{8520D25D} & \texttt{0CB81A74} & \texttt{18C774CE} & \texttt{D0E74EA2} & \texttt{927BC2E2} & , \\
 & & \texttt{1D37BAB9} & \texttt{DB5BFFA9} & \texttt{5886CE28} & \texttt{C86D9BFD} & \texttt{097C38C4} & \texttt{330A36D2} & \texttt{0D3575C3} & \texttt{9560D72A} &), \\
 & (& \texttt{B74836D} & \texttt{53381E94} & \texttt{488390E1} & \texttt{8997D33C} & \texttt{7AC4930B} & \texttt{DC4E2789} & \texttt{0E01FA4E} & \texttt{56EBF229} & , \\
 & & \texttt{6CD0828} & \texttt{B4C8C0DE} & \texttt{731AFA2B} & \texttt{7C102088} & \texttt{58DA9529} & \texttt{92E45D5A} & \texttt{29B9E17D} & \texttt{AA4C0338} &), \\
 & (& \texttt{1F9CB4F6} & \texttt{F5BAAA5C} & \texttt{40F28DD1} & \texttt{BE56A1D6} & \texttt{90E16D51} & \texttt{8C41352E} & \texttt{71346595} & \texttt{814E6B01} & , \\
 & & \texttt{1336B49E} & \texttt{792EA767} & \texttt{9AA8FC58} & \texttt{374C0A1E} & \texttt{7B30C61F} & \texttt{3318E764} & \texttt{3967520B} & \texttt{C09C7D1D} &), \\
 & (& \texttt{5F5C28D} & \texttt{5196C375} & \texttt{7ACDA855} & \texttt{85F0BACB} & \texttt{87B41247} & \texttt{BA964F7F} & \texttt{E383A68D} & \texttt{0DA0555F} & , \\
 & & \texttt{1C48DDF8} & \texttt{EBA9F071} & \texttt{AD37A543} & \texttt{513CDEF3} & \texttt{0866183C} & \texttt{A571ECD0} & \texttt{D410841F} & \texttt{84711D03} &))
\end{array}$$

2.2. Compute the exponentiation in $\mathbb{F}_{p^k}$: $K = (e_A(Q, P_{n+1}))^t$

$$
\begin{array}{lllllllll}
\bar{t} \; = & \texttt{CA01E0E} & \texttt{EF2718A2} & \texttt{A90C3636} & \texttt{FCC04963} & \texttt{F9B9CFA1} & \texttt{22F216B3} & \texttt{D300A198} & \texttt{5CE006D1}
\end{array}
$$

$$
\begin{array}{llll}
K & = ((& 71069444066117429272820887801344745423777079781326495725901232190340847806 98 & , \\
 & & 1258759866955075411811449670435810675651738979516537289436234784861429125200 8 &), \\
 & (& 651907020137135794752749757678117454698675617610300278462301261500315911475 5 & , \\
 & & 1780261461183952226375357174335853042706416535421374488198350559095822453 10 &), \\
 & (& 1370953268223170446865738240492666601489029567491766881922163633314560759235 0 & , \\
 & & 652914899957185406521761775668705984569563074073626898615024491668058939148 7 &), \\
 & (& 64452449714192324238525500808353444465637170643879840254415386172657031673 6 & , \\
 & & 1584800849352092542157221749214396511443721062034095458722766721209962018799 2 &), \\
 & (& 868523473195348656683815249544405801212097018773670241090840455620365447372 2 & , \\
 & & 1188671089306447270035398022307409227685673717704001080565289332176461006001 3 &), \\
 & (& 697944529495114173392394041984991818152006657067466842572364803097236689870 5 & , \\
 & & 359703227618451566898922179143642843559414554208499010429653022407221816764 2 &))
\end{array}
$$

As an octet string, we have:

$$
\begin{array}{llllllllllll}
\overline{K} & = ((& \texttt{FB66353} & \texttt{0E549A3A} & \texttt{719C8846} & \texttt{B16425A5} & \texttt{530E1EA7} & \texttt{4F62FED5} & \texttt{667C6023} & \texttt{8328AA9A} & , \\
 & & \texttt{1BD45410} & \texttt{1327B461} & \texttt{90058B7B} & \texttt{C4690B82} & \texttt{C8C922C8} & \texttt{DE54F362} & \texttt{EA157409} & \texttt{24992328} &), \\
 & (& \texttt{E69A9B8} & \texttt{E07C5702} & \texttt{8EB43CCE} & \texttt{0ED80CF4} & \texttt{953462D2} & \texttt{7A0DC543} & \texttt{E71B7406} & \texttt{0E7FB403} & , \\
 & & \texttt{64C25C} & \texttt{C2FBD758} & \texttt{9744F597} & \texttt{B4132367} & \texttt{88DCBF34} & \texttt{36784DCE} & \texttt{3EAA6F80} & \texttt{A0298DBE} &), \\
 & (& \texttt{1E4F520D} & \texttt{7AD6D55F} & \texttt{C09DC2C7} & \texttt{099E1010} & \texttt{9E843A79} & \texttt{A528E378} & \texttt{BDBD60DC} & \texttt{3DF3599E} & , \\
 & & \texttt{E6F5E0C} & \texttt{479827AE} & \texttt{A9982B8D} & \texttt{9F11E983} & \texttt{DFA0718F} & \texttt{0EF405FC} & \texttt{BCE7B858} & \texttt{76F1367F} &), \\
 & (& \texttt{16CC9B6} & \texttt{4A69B3CF} & \texttt{5E0F21C5} & \texttt{D5A96A6C} & \texttt{6A7A5E4F} & \texttt{89A8CD9C} & \texttt{CF0C1A5B} & \texttt{6E8083C0} & , \\
 & & \texttt{2309A7A9} & \texttt{CD9AC98C} & \texttt{F7D46749} & \texttt{CC404832} & \texttt{DB3CF7F6} & \texttt{6FCA91D8} & \texttt{93CF9F26} & \texttt{9C402F58} &), \\
 & (& \texttt{1333AB2C} & \texttt{E1171C7B} & \texttt{77283F03} & \texttt{54D82022} & \texttt{57AB9996} & \texttt{1B7AA7F5} & \texttt{DDC5C010} & \texttt{CB7167FA} & , \\
 & & \texttt{1A47A3D3} & \texttt{8F5ACAF2} & \texttt{755DE32E} & \texttt{F94721D5} & \texttt{4BA7E034} & \texttt{11CC302E} & \texttt{627A6939} & \texttt{9B852EED} &), \\
 & (& \texttt{F6E39DE} & \texttt{2A0875BD} & \texttt{D308C969} & \texttt{D0E2AA9B} & \texttt{374BFD97} & \texttt{24805F2B} & \texttt{B2A7C92E} & \texttt{2A6D4611} & , \\
 & & \texttt{7F3D91E} & \texttt{6E31D3E1} & \texttt{1FE90D6B} & \texttt{8E354C65} & \texttt{D071AAAD} & \texttt{63CD898F} & \texttt{A2A22C46} & \texttt{3E8CC95A} &))
\end{array}
$$

3. Compute the header `Hdr` $= (C_0, C_1, \ldots, C_A)$

3.2. Compute $h = H(t \cdot Q)$.

3.3. Compute $h \cdot P_1 = (x_{hP_1}, y_{hP_1})$ in E_1.

3.4. Compute $Sum = \sum_{j \in S} P_{n+1-j} = (x_{Sum}, y_{Sum})$ in E_1.

3.5. Compute $h \cdot P_1 + V + Sum$ in E_1.

3.6. Compute $C_1 = t \cdot (h \cdot P_1 + V + Sum) = (x_{C_1}, y_{C_1})$ in E_1.

Output: The pair $(K, \texttt{Hdr})$.

E encrypts a message with K, adds the `Hdr` to the message and broadcasts all.

D Golden Sequence with Optimal Ate Pairing

We generate the test vectors with using the Optimal Ate pairing (explained in 2.2) as the asymmetric pairing. We have not rewritten the vector tests, who are the same than in section 5. The Set Up step and Join step are the same.

We choose $e(P, Q) = e_{Opt}(Q, P)$.

D.1 Example 1: Test with 100/100 authorized users

In this example, all the users are authorized.

D.1.1 Encrypt

E generates a session key to encrypt a message and the header, R can compute the session key, iff R is authorized.

Input: S the set of the users who are authorized, the public key PK_s, and the hash function H.

Action: Generate a session key K and a header key `Hdr`

1. Generate an integer t
2. Compute the session key K.

2.1 Compute the pairing $e_{Opt}(Q, P_{n+1})$ We can use P_n and Q_1.

$$\begin{array}{rlll}
\overline{Q_1} & = ((& \texttt{1415FE8 B1FE4B90 F7E6C5B3 5D716171 73CAC5DB 94517E7D 6F93AC73 A7F22C5B} & , \\
 & & \texttt{7C8953A A7FEE45F 168397CA 14117A15 421F071F 9756EA2F A3F43C85 1571B9FB} &), \\
 & (& \texttt{1CD9FD2D 1A70BAD6 8AD193FE F734073B 6B589AFE 272CFD09 D66A10C2 A2AE3A80} & , \\
 & & \texttt{27F28D7 F4F737C5 181D8188 C6F2F8AC D4F3965A 7AE4F427 4F580DCE C8D30434} &)) \\
\overline{P_n} & = (& \texttt{1F2E354E DF838381 FB886C67 ECFDE49E D1E04EC5 BB201D0B 9F8DDA3F 886F6BE3} & , \\
 & & \texttt{729F5F3 44F16BC9 C3833794 28F13899 94B218BC D9710596 104FE566 130839CC} &)
\end{array}$$

$$\begin{array}{rlll}
e_{Opt}(Q_1, P_n) & = ((& 5976870746792449574645602442921421724689746093918425713510092949256742180686 & , \\
 & & 12170486461298907872159868262391669141449039589411353021615354033069907316169 &), \\
 & (& 4782273022530600146730720919135556002844281056270482237410672817205193290597 & , \\
 & & 12599834058084952817074711059668910856751548047382213510969689286745759207746 &), \\
 & (& 10642185189419736889269754379847596896623005541414199820376251537512117284858 & , \\
 & & 16146279360393791025932186772286591892550556056152638386557760495667139069457 &), \\
 & (& 14839525527398706610454470040660976793454765502859821135130298374682242718455 & , \\
 & & 4040354256441007304293237807053995502142828012593819893325045870610671667752 &), \\
 & (& 8700857019271774661779697225907327248102475566084473512622625579512516633964 & , \\
 & & 14452873313259167235441213740775866729157685926912542689217889031394692237563 &), \\
 & (& 4794706782548299538766722548962108446692463226595731087011473219092667903060 & , \\
 & & 5350481077890923028663758822184109175069416643242720098321474056052111295414 &))
\end{array}$$

As an octet string, we have:

$$\begin{array}{rlll}
\overline{e_{Opt}(Q_1, P_n)} & = ((& \texttt{D36C9F9 5BD15283 50FAC594 FB7B1324 382B2673 C40B7EDA 40A64911 4419174E} & , \\
 & & \texttt{1AE84050 641DC0A4 DCAE0F59 E33CEE29 C3991A93 3B9EDAFE BF527D84 A7ED0DC9} &), \\
 & (& \texttt{A92ABAD 1828A677 7E775DBC 8218780C 1B4423F4 24AFDAF6 73F68635 F4993365} & , \\
 & & \texttt{1BDB40DB DBE094EF 7CDEFD11 19A6F2CD 1262F280 5DC4153C 52E0D4CD AEA0D142} &), \\
 & (& \texttt{1787436B 9D0E8318 4B68694B 34C12E09 8DEFC8D3 345231E7 59B4E016 63CFAFFA} & , \\
 & & \texttt{23B27863 24951A7C 7C8EBD18 56A12766 E2CCEBAF 803F3643 C9213BBB 8CE04611} &), \\
 & (& \texttt{20CEDFB1 9858A8C3 E9335D88 A538DBAB E880841D E0CD7733 681F5654 05341AF7} & , \\
 & & \texttt{E4AD0A DBE3F4D1 10763C4B 6394D5A6 878B642B 9FEB2F7A 750D2649 EFFAB420} &), \\
 & (& \texttt{133C82B3 B8AA30CC DC3137F6 093287F2 FFE27DDA AE807C49 4BA7E460 9C48E56C} & , \\
 & & \texttt{1FF40951 E2EFB1A2 EED0DE69 0578AFC9 D4B31841 9CF79238 FC9977A7 C93108FB} &), \\
 & (& \texttt{A99B536 E2BDCAB3 4E0B8A1B 0C0006E9 6F378104 2C15146A 6AC22075 3C132C54} & , \\
 & & \texttt{BD443D5 3B84E855 ECE9F0B5 D5FC9434 622A066A 4D8EB461 D0E3FAD2 BB5BC3B6} &))
\end{array}$$

2.2. Compute the exponentiation in $\mathbb{F}_{p^k}$: $K = (e_{Opt}(Q, P_{n+1}))^t$

$$\overline{t} = \texttt{CA01E0E EF2718A2 A90C3636 FCC04963 F9B9CFA1 22F216B3 D300A198 5CE006D1}$$

$$\begin{array}{lll}
K = ((& 1556718930424607051067080384204832605184446128981897050289002402739492244753 9 & , \\
 & 1615294331809360540709012010801602756947973368681154791909752064022876554880 5 &), \\
(& 852556839564094667954562366958254550724229352750439385389736076468055251060 6 & , \\
 & 1475477291869542612230422091862030789161380529998729993588501856903157620912 9 &), \\
(& 226002953330533847096651464475579159628057250235302406694265234701188901362 3 & , \\
 & 803483710724333410901830161287776743286025936701950577125350684938939226095 8 &), \\
(& 1470564359916705438674433227376351985705233611710052241769566367213670208005 8 & , \\
 & 1128802624809533049937494267642213031944359082554022241808603544561460267778 0 &), \\
(& 481691699913896627307123982263448980371424091425898642139053304124367003625 1 & , \\
 & 1222100919194346494182594874394792470906723803107656162783394231822501137457 9 &), \\
(& 698986421301845412384191212183242144777384254114076299004854038406029965587 1 & , \\
 & 766672092098511524756369373109881484706951220943830454782006921464247058255 7 &))
\end{array}$$

As an octet string, we have:

$$\begin{array}{llllllllll}
\overline{K} = ((& \texttt{226AB787} & \texttt{2989315F} & \texttt{6FEE71BD} & \texttt{04726093} & \texttt{35839CF8} & \texttt{366799FF} & \texttt{2B840D15} & \texttt{ECA302B3} & , \\
 & \texttt{23B63DEF} & \texttt{12603B03} & \texttt{CE5D05D7} & \texttt{63B91DAE} & \texttt{4F496C0E} & \texttt{CF6D8A22} & \texttt{3547FE67} & \texttt{AB9D4105} &), \\
(& \texttt{12D94CFB} & \texttt{4FEAD364} & \texttt{40E263B4} & \texttt{77EC9729} & \texttt{90751675} & \texttt{8BDD66BF} & \texttt{5CD3BA5D} & \texttt{C15AF48E} & , \\
 & \texttt{209EE7D0} & \texttt{9EC64299} & \texttt{D39D7F80} & \texttt{E7EFB3C3} & \texttt{187DD969} & \texttt{A2AD91A8} & \texttt{DA160969} & \texttt{9D0046E9} &), \\
(& \texttt{4FF21A2} & \texttt{74EB32BA} & \texttt{E7BF98B2} & \texttt{304ED6B7} & \texttt{0AF9921E} & \texttt{3C6BC3D9} & \texttt{AFF765B0} & \texttt{8126A377} & , \\
 & \texttt{11C38E80} & \texttt{2DF948E2} & \texttt{D0D81362} & \texttt{339ED5EF} & \texttt{D64704D0} & \texttt{A8494F56} & \texttt{61581C47} & \texttt{0C0D6F5E} &), \\
(& \texttt{2083196C} & \texttt{E0E3C4B2} & \texttt{1C4541EE} & \texttt{FCFFD177} & \texttt{07F648FD} & \texttt{D65EC492} & \texttt{E5DE129A} & \texttt{008FB43A} & , \\
 & \texttt{18F4CBE3} & \texttt{E2B43A67} & \texttt{83C8AC70} & \texttt{55B712AD} & \texttt{46B77FE6} & \texttt{02891E2A} & \texttt{F0905CB8} & \texttt{3E14EE14} &), \\
(& \texttt{AA64745} & \texttt{9B06AB1B} & \texttt{EEC3600D} & \texttt{02BD3C77} & \texttt{E442B8B4} & \texttt{0FFA8005} & \texttt{5092E4C0} & \texttt{650DBF1B} & , \\
 & \texttt{1B04D898} & \texttt{9FA702D6} & \texttt{51E39DF7} & \texttt{BE181753} & \texttt{4DDA7E0B} & \texttt{ECAD4EDC} & \texttt{9F3E7ECF} & \texttt{3A71F9F3} &), \\
(& \texttt{F741F79} & \texttt{5060F496} & \texttt{F495E125} & \texttt{FCE44BED} & \texttt{E555E6CA} & \texttt{2DF8DC53} & \texttt{8DD2B5B0} & \texttt{3FF1BABF} & , \\
 & \texttt{10F335D3} & \texttt{F50C5185} & \texttt{9006019C} & \texttt{79DA61CE} & \texttt{B63A1AA5} & \texttt{7F54C2F3} & \texttt{794AC70A} & \texttt{FDE6F91D} &))
\end{array}$$

3. Compute the header `Hdr` $= (C_0, C_1, \ldots, C_A)$

3.2. Compute $h = H(t \cdot Q)$.

3.3. Compute $h \cdot P_1 = (x_{hP_1}, y_{hP_1})$ in E_1.

3.4. Compute $Sum = \sum_{j \in S} P_{n+1-j} = (x_{Sum}, y_{Sum})$ in E_1.

3.5. Compute $h \cdot P_1 + V + Sum$ in E_1.

3.6. Compute $C_1 = t \cdot (h \cdot P_1 + V + Sum) = (x_{C_1}, y_{C_1})$ in E_1.

Output: The pair $(K,$`Hdr`$)$.

E encrypts a message with K, adds the `Hdr` to the message and broadcasts all.