社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

Strongly Asymmetric PKD Cryptographic Algorithms

an implementation using the matrix model

LuigiACCARDI† 入山 聖史†† MassimoREGOLI† 大矢 雅則††

† Università di Roma "Tor Vergata" Via Columbia, 2, 00133, Rome, Italy
†† 東京理科大学 〒 278-8510 千葉県野田市山崎 2641
E-mail: †accardi@volterra.mat.uniroma2.it, ††{iriyama,ohya}@is.noda.tus.ac.jp, †††regoli@uniroma2.it

あらまし This work follows the paper of the authors: *Strongly asymmetric PKD cryptographic algorithms*, which introduces a new family of algorithms for key exchange (the *double - parametrization* algorithm), and here we discuss possible configuration of a particular case (a *model*) of this family. The present paper describes the implementation of a model based on matrices described in the following.. The approach will be discussed in the following. Moreover we will show the problems encountered and solved in the final implementation and the technical issues.
キーワード asymmetric public key agreement, key conjugation, publick key distribution

Strongly Asymmetric PKD Cryptographic Algorithms

an implementation using the matrix model

Luigi ACCARDI†, Satoshi IRIYAMA††, Massimo REGOLI†, and Masanori OHYA††

† Università di Roma "Tor Vergata" Via Columbia, 2, 00133, Rome, Italy
†† Tokyo University of Science 2641 Noda City, Chiba, Japan
E-mail: †accardi@volterra.mat.uniroma2.it, ††{iriyama,ohya}@is.noda.tus.ac.jp, †††regoli@uniroma2.it

Abstract This work follows the paper of the authors: *Strongly asymmetric PKD cryptographic algorithms*, which introduces a new family of algorithms for key exchange (the *double - parametrization* algorithm), and here we discuss possible configuration of a particular case (a *model*) of this family. The present paper describes the implementation of a model based on matrices described in the following.. The approach will be discussed in the following. Moreover we will show the problems encountered and solved in the final implementation and the technical issues.
Key words asymmetric public key agreement, key conjugation, publick key distribution

1. Introduction

1.1 Description

Let us start with some important definition about the *Strongly Asymmetric PKA Algorithms*.

- Public information is split among **multiple public keys**
- B (**the receiver**) has one set of public keys
- The unique public key used by A (**the sender**) depends on those of the Receiver.
- security
- flexibility
- variety of concrete realizations which cannot be found in the standard PKA algorithms.

1.2 Implementation

The implementation is:

- easier to implement
- quicker.

So let us start with a general description (see [1])

- Public Ingredients
  - $\mathbb{N}$, the natural integers
  - $\mathcal{P}$, a semigroup (noted multiplicatively, with 1)
  - $\alpha \in \mathcal{P}$, an element of $\mathcal{P}$

$$\{\mathbb{N}\ ;\ \mathcal{P}\ ;\ \alpha\}$$

Notations:

$$\mathcal{P}_0(\alpha) := \{\alpha^n \ : \ n \in \mathbb{N}\} \subsetneqq \mathcal{P}$$

the (abelian) semigroup generated by α.

$$\mathrm{Mor}(\mathcal{P}_0(\alpha), \mathcal{P}) := \{\pi : \mathcal{P}_0(\alpha) \to \mathcal{P} \ :$$

$$\pi(1) = 1 \ ; \ \pi(xy) = \pi(x)\pi(y) \quad ; \quad \forall x, y \in \mathcal{P}_0(\alpha)\}$$

Typical example ($\mathcal{P}_0(\alpha)$ is commutative)

$$x \in \mathcal{P}_0(\alpha) \mapsto \pi(x) = Sx^M S^{-1}$$

S is any invertible element in $\mathcal{P}$.

1.3 Secret ingredients of Receiver

- $N_{B,1} : \mathcal{P} \to \mathcal{P}$ easily invertible map
- $N_{B,3} : \mathcal{P} \to \mathcal{P}$ easily invertible map (not morphism)
- $x_{B,1}, x_{B,2}, x_{B,3}, x_{B,4} : \mathcal{P} \to \mathcal{P}$

arbitrary functions satisfying the **compatibility conditions**:

- $x_{B,1}x_{B,2}|_{\mathcal{P}_0} = x_{B,3}x_{B,4}|_{\mathcal{P}_0}$
- $N_{B,1}x_{B,2}|_{\mathcal{P}_0}(\alpha) \in \mathrm{Mor}(\mathcal{P}_0(\alpha), \mathcal{P})$

Using these secret ingredients, B constructs his secret and public keys.

2. The protocol

Now using the above ingredients we can construct am entire protocol for key exchange in some steps (see figure (1))

3. The matrix version

In order to estimate the robustness of the algorithm, we begin to study one of its simplest realizations, i.e. most favorable for the attacker E.

This will give a lower limit for the breaking complexity of the algorithm.

3.1 The public ingredients

$$\mathcal{P} := M_d(\mathbb{F}) =: \ d \times d \text{ matrices} =: M_d \tag{1}$$

$$\alpha \in M_d \tag{2}$$

with a long period.

Explicit algorithms to construct such α are known.

3.2 Secret ingredients of Receiver

$$\pi_1(x) := ax + b \quad ; \quad a, b \in \mathbb{F} \tag{3}$$

$$\pi_2(x) := P_2 x^N P_2^{-1} \quad ; \quad x \in M_d \tag{4}$$

where P_2 is an invertible element in M_d and $N \in \mathbb{N}$,

$$\sigma := id \tag{5}$$

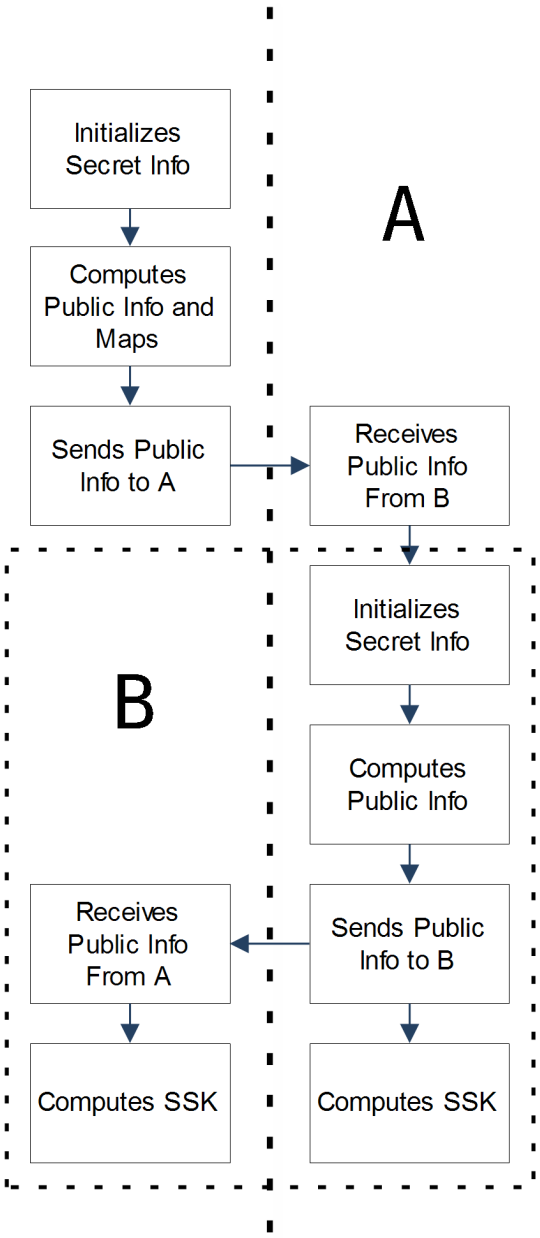

図 1 PKD diagram

$$\rho(x) := AxB^{-1} + C \tag{6}$$

where A, B are invertible elements in M_d.

$$\pi_q(y) := \sum_{H=0}^{M} a_H y^H b_H \tag{7}$$

where

$$a_H, b_H \in M_d \quad ; \quad M \in \mathbb{N} \tag{8}$$

3.3 Comments on the above choices

$$\rho(x) := AxB^{-1} + C$$

is invertible and its inverse is

$$\rho^{-1}(z) := A^{-1}(z-C)B \tag{9}$$

In the definition of

$$\pi_1(x) := ax + b \qquad ; \qquad a, b \in \mathbb{F}$$

a and b they can also be matrices. In fact, if we want π_1 not invertible and nontrivial, then a must be a matrix. If a is invertible, then π_1 is easily invertible and its inverse is

$$\pi_1^{-1}(y) := a^{-1}y - a^{-1}b \tag{10}$$

However, in order to further simplify the model, we choose a,b to be scalars.

With these choices the secret key of Receiver is the function:

$$\hat{x}_B := \pi_q \rho^{-1} = \pi_q(A^{-1}(\ \cdot\ - C)\ )B)$$

3. 4 Public Keys of Receiver

With the above choices, *Sender* should receive the public keys of *Receiver*, that is α and the functions:

$$\pi_q \pi_1 \sigma^{-1}(x) = \pi_q(ax+b) =$$

$$= \sum_{H=0}^{M} a_H (ax+b)^H b_H \tag{11}$$

$$\rho\pi_1\pi_2(x) = AaP_2 x^N P_2^{-1} B^{-1} + AbB^{-1} + C \tag{12}$$

$$\sigma\pi_2(\alpha) = \pi_2(\alpha) = P_2 \alpha^N P_2^{-1} \tag{13}$$

Notice that ρ is invertible and its inverse is

$$\rho^{-1}(z) := A^{-1}(z-C)B \tag{14}$$

Remark. In (3), a and b can also be matrices.
In fact if we want π_1 to be non invertible, they have to be matrices.
If a is invertible, then π_1 is easily invertible and its inverse is

$$\pi_1^{-1}(y) := a^{-1}y - a^{-1}b \tag{15}$$

3. 5 Structure of the public data of the Sender

Receiver sends $\sigma\pi_2(\alpha)$ to Sender. Knowing this Sender is able to compute:

$$\sigma\pi_2(\alpha)^{x_A} = (P_2\alpha^N P_2^{-1})^{x_A} = P_2 \alpha^{N x_A} P_2^{-1} \tag{16}$$

The public key of A is given by

$$y_A = \rho\pi_1\pi_2(\alpha^{x_A}) = (\rho\pi_1\sigma^{-1})(\sigma\pi_2(\alpha)^{x_A}) =$$

$$= A(aP_2(\alpha^{x_A})^N P_2^{-1} + b)B^{-1} + C =$$

$$= A(aP_2\alpha^{N x_A} P_2^{-1} + b)B^{-1} + C =$$

$$= AaP_2\alpha^{N x_A} P_2^{-1} B^{-1} + AbB^{-1} + C =$$

$$= Aa(P_2\alpha^N P_2^{-1})^{x_A} B^{-1} + AbB^{-1} + C =$$

$$= Aa(\sigma\pi_2(\alpha))^{x_A} B^{-1} + AbB^{-1} + C \tag{17}$$

3. 6 Structure of the public data of Receiver (first part)

In order to allow Sender to create her public key Receiver uses the identity

$$y_A = Aa(\pi_2(\alpha))^{x_A} B^{-1} + AbB^{-1} + C$$

and sends to Sender the matrix (d^2 numbers):

$$AbB^{-1} + C \in M_d \tag{18}$$

and the map

$$z \in M_d \mapsto AazB^{-1}$$

This means that Receiver sends to Sender the $d^4 + d^2$ numbers:

$$X_{ij;j_1j_2} := (Aa)_{ij_1}(B^{-1})_{j_2 j} \tag{19}$$

$$(AbB^{-1})_{ij} + C_{ij}$$

$$i, j, j_1, j_2 \in \{1, \ldots, d\}$$

(20)

3. 7 Computation of the SSK

$$\kappa = \pi_q(a(P_2\alpha^{N x_A} P_2^{-1}) + b) = \tag{21}$$

$$= \sum_{H=0}^{M} a_H (a(P_2\alpha^{N x_A} P_2^{-1}) + b)^H b_H$$

Since a, b are numbers, $\pi_q\pi_1\sigma^{-1}(x)$ is equal to

$$\sum_{H=0}^{M} a_H \sum_{J=0}^{H} \binom{H}{J} a^J b^{H-J} x^J b_H =$$

$$= \sum_{H=0}^{M}\sum_{J=0}^{H} \binom{H}{J} a^J b^{H-J} a_H x^J b_H$$

Therefore the SSK is

$$\sum_{H=0}^{M}\sum_{h=0}^{H} a_H a^h P_2 \alpha^{N x_A h} P_2^{-1} b^{H-h} \binom{H}{h} b_H = \tag{22}$$

$$= \sum_{H=0}^{M}\sum_{h=0}^{H} a_H a^h (P_2\alpha^N P_2^{-1})^{x_A h} b^{H-h} \binom{H}{h} b_H$$

Knowing x_A:

- in order to compute (16) and (17), Sender needs only to know

$$P_2\alpha^N P_2^{-1} \tag{23}$$

- in order to compute (17), Sender needs only to know

$$\alpha \quad ; \quad AbB^{-1} \quad ; \quad N \tag{24}$$

$$AaP_2X \quad ; \quad X^{-1}P_2^{-1}B^{-1} \tag{25}$$

where X is any invertible element in the commutant of the algebra generated by α.

- in order to compute (21), Sender needs only to know

$$a_H a^h P_2 \mu(H,h) Y \quad ; \quad Y^{-1} P_2^{-1} b^{H-h} \nu(H,h) b_H \tag{26}$$

where Y is any invertible element of the algebra generated by α and $\mu(H,h), \nu(H,h)$ are arbitrary natural integers such that

$$\mu(H,h)\nu(H,h) = \binom{H}{h} \tag{27}$$

4. The implementation

The present computer implementation of the matrix version of the algorithm is based on C++ language.

So, in the logic of object-oriented programming, we started to fix some abstract properties and methods of the base classes.

First, we realized that, within a *namespace* '*container*', we had to create two main classes:

- Sender class
- This class will contain derived abstract methods and 'proprietary' methods for Sender
- Receiver class
- This class will contain derived abstract methods and 'proprietary' methods for receiver

In both classes the idea was to fix a common interface for new implementations for others models in order to obtain a single communication protocol between applications and the classes libraries.

It is clear that some specific properties, methods and special classes should be included in the main namespace or classes depending on the circumstances (i.e. from the model).

```
namespace DP {
// Standard virtual classes
  class Receiver {
  ...
  }
  class Sender {
  ...
  }
 // Proprietary classes
};
```

the minimum number of properties are set to:

```
namespace DP {
  template <class SSK, class SecA>
     class Sender {
  public:
    virtual void Initialize(void) = 0;
    virtual SSK getPublicKey() = 0;
    virtual SSK getSecretKey(void) = 0;

    virtual void setSecret(SecA x) = 0;
    virtual void setSecret(string s) = 0;

    virtual void setPublic(string s) = 0;

    virtual string publicToString(void) = 0;
    virtual int stringToPublic(string s) = 0;

    virtual void reset() = 0;
  };
  template <class SSK, class SecA>
     class Receiver {
  public:
    virtual void Initialize(void) = 0;

    virtual void setSecret(string s) = 0;

    virtual SSK getSecretKey(SSK yA) = 0;
    virtual SSK getSecretKey(string s) = 0;

    virtual string publicToString(void) = 0;

    virtual void reset() = 0;
  };
 }
}
```

Following this abstract class we can write:

```
DP::Sender S;
DP::Receiver R;
string secS, secR, SSKS, SSKR;

S.Initialize();
R.Initialize()

///
/// Create secret info for Receiver
/// in some way...
///

R.setSecret(secR);

///
/// Create secret info for Sender
/// in some way...
///
```

```
S.setSecret(secS);

S.setPublic(R.publicToString());

/// SSKS == SSKR

SSKR = R.getSecretKey(S.publicToString());
SSKS = S.getSecretKey();

/// Reset ALL

S.reset(); R.reset();
```

4.1 Problems in key exchange

d	M	bytes
2	5	448
3	5	2088
4	5	6400
5	5	15400
6	5	31680
7	5	58408
8	5	99328
9	5	158760
10	5	241600

表 1 Size of public keys of Receiver

In the matrix model the first problem we encountered was about the two maps that Receiver must send to Sender (for example see (19)).

In fact, we can't send the individual ingredients (the elements) of the maps (which are, in our case, the secret matrices of Receiver), but, rather, the final coefficients obtained by a symbolic calculation of the relative formulas and that are directly connected with the arguments passed to the maps.

In this way, given the complexity of the calculation, you can not go back to the original matrices

So Receiver can sends to Sender all these information just to compute his public informations and the relative *shared secret key* without being familiar with the secret ingredients of Receiver.

Unfortunately, this strategy has a small problem in the matrix model implementation: the size of the public informations of Receiver.

In fact after some simple calculations we obtain that the cardinality of the sets of the numbers that Receiver has to send to Sender is strongly depending on the size of the matrices d and it is equal to:

$$|Pub_K| = 4d^2 + Md^4 + d^4$$

in terms of byte (assuming that a number is represented by 4 bytes):

$$|Pub_K|_b = 4(4d^2 + Md^4 + d^4)$$

the tables (1), shows this trend.

However the public key of Receiver must be transmitted only once. Notice that it is sufficient that Sender changes his secret key and sends its new public key to Receiver to obtain a new SSK.

In this case the size of the public key of Sender in terms of bytes is smaller:

$$4d^2$$

(see table (2)).

d	PK bytes	SK bytes
2	16	16
3	36	36
4	64	64
5	100	100
6	144	144
7	196	196
8	256	256
9	324	324
10	400	400

表 2 Size of public key of Sender and of the SSK

5. Data format

5.1 Receiver

The next step is to create a transmission protocol between Sender and Receiver.

Let us recall that Receiver must send a lot of numbers to Sender (see table (1)) and Sender must return only few numbers (see table (1)) to Receiver.

In addition, the Receiver also decides:

- the size of the key (i.e. the dimensions of the matrices)
- the prime number
- and the degree of the polynomial (7)

So the first packet should be as in figure (5.1) where:

- d is the dimension of the matrices
- M is the degree of polynomial (7)
- and p is the prime number

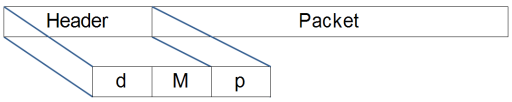

図 2 Matrix Model Header

But, in order to underline, here, the strength of the Strongly asymmetric PKA family, we will also put in the header the algorithm chosen.

In fact the Receiver can choose (if both, Sender and

Receiver, can agree on a common implementation of the Strongly asymmetric PKA algorithms) the method T that implements the general algorithm and then he will create a more complex header like the one in figure (5. 1) that, of course, depends on the method T.

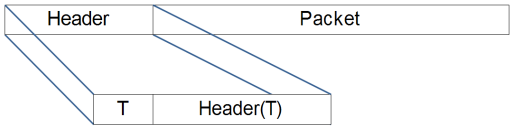

図 3　General Header

The dimension and format of the *Packet* depends on:

- The method T
- Some further specialization in Header(T)

In the Matrix Model the Packet depends on d (the dimension of the matrices) and from M (the degree of polynomial (7). With these choices the size of the packet will be exactly represented by table (1).

In our case we organize the packet in the following order:

- *alpha*
- First Public Matrices (i.e. matrix (18))
- Second Public Matrices (i.e. matrix (23))
- First Map (i.e. coefficients of map (19)) (this is a vector)
- Second Map (i.e. coefficients of map (22)) (this is a vector)
- Third Public Key (i.e. constant matrix in the map(22))

5. 2 Sender

From the Sender point of view, in the Matrix Model, all parameters were already set by the Receiver, so we would not need to use an header, but just to avoid mistake and error in programming we can create a header that contains a copy of the information of the Receiver.

But, as a consequence of the fact that:

- The Strongly asymmetric PKA algorithms family is very new
- We have no idea of the actual amount of possible implementations
- New implementations may require some choices even by Sender

then, we decided to provide also the Sender of a complex header that, in the case of Matrix Model is simple (see figure (5. 1) but in general case is the one represented in figure (5. 2).

6. Conclusion

In this work we introduced a particular class of the strongly asymmetric PKA algorithms.

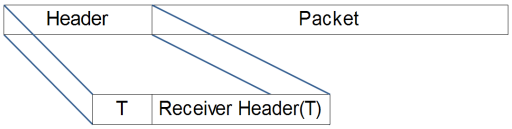

図 4　Sender General Header

This implementation, based on matrices has shown:

- a remarkable production speed of public and secret keys (see table (6.) or figure (6.)),
- ease of use (the secret ingredients Some are easy to fix),
- enormous versatility (possibility to change the dimension of the matrices, the degree of the polynomial , and the prime number)

Remark: As an explanation of table (6.) we have to say that the calculation includes the following activities:

（1） Production of Public info of Sender
（2） Production of Secret info of Receiver
（3） Production of Secret info of Sender

From a detailed analysis of the cycle of distribution an important result emerges. This is described in table (6.), where we can see that the most part of the activity is due by the Sender (ca 90% of the total activities) and only 10 % by the Receiver.

This is crucial, especially in those activities in which one of the interlocutors is a device with low computing power (i.e. cellular phone and other small devices).

in this case if the device with low computing power should play the role of Sender, the Sender itself will sends a message to Receiver to request it to start, in this way he reverses the roles in the communication.

As soon as the *true sender* will receive the key necessary for the production of the secret key, the normal communication starts.

Obviously, in order to implement this scenario, the partners must have already exchanged the basic public information.

d	seconds	PK / sec
4	1	1000
5	2.1	476.1904762
6	4.2	238.0952381
7	6.8	147.0588235
8	10.9	91.74311927
9	15.29	65.40222368
10	21.34	46.86035614

表 3　Production speed of 1000 secret keys (once established and exchanged the public keys of the Sender)

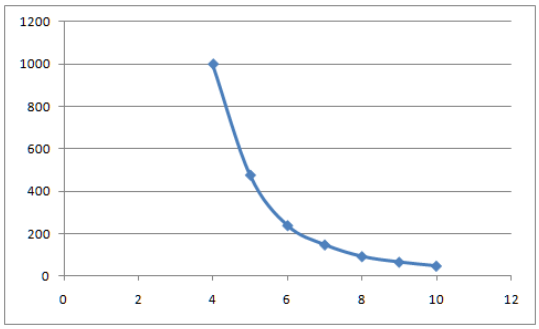

図 5　Number of secret keys generated in one second

% Pub. S.	% SSK S.	% SSK R.
15	75	10

表 4　Time of production key in percentage of Sender and Receiver Activities

7. Next Steps

Our intention for the future is:

- Complete the implementation of the Matrix Model by introducing new features
  - For example introducing the opportunity to create more than on secret matrices for each exchange using more than one Sender secret key
  - this doubles the size of the public information of the Sender but also doubles the size of the secret shared key
- Collecting new alternative methods based on mathematical tools
- Improving the transmission protocol
- Statistically analyze the keys produced to demonstrate the inviolability also statistical attacks

文　　献

[1] L. Accardi *New Features for Public Key Exchange Algorithms* 18-th International ICWG Meeting May 9-13, 2011 Krakow, Poland

[2] M. Regoli *From Cryptography to Biology and Vice Versa*, Open Systems & Information Dynamics (OSID), Volume: 18, Issue: 1(2011) pp. 87-105 DOI: 10.1142/S1230161211000066 World Scientific Press, Singapore