New Subclass Framework and Concrete Examples of Strongly Asymmetric Public Key Agreement

Satoshi Iriyama 1,†, Koki Jimbo 1,†,* and Massimo Regoli 2,†

Abstract: Strongly asymmetric public key agreement (SAPKA) is a class of key exchange between Alice and Bob that was introduced in 2011. The greatest difference from the standard PKA algorithms is that Bob constructs multiple public keys and Alice uses one of these to calculate her public key and her secret shared key. Therefore, the number of public keys and calculation rules for each key differ for each user. Although algorithms with high security and computational efficiency exist in this class, the relation between the parameters of SAPKA and its security and computational efficiency has not yet been fully clarified. Therefore, our main objective in this study was to classify the SAPKA algorithms according to their properties. By attempting algorithm attacks, we found that certain parameters are more strongly related to the security. On this basis, we constructed concrete algorithms and a new subclass of SAPKA, in which the responsibility of maintaining security is significantly more associated with the secret parameters of Bob than those of Alice. Moreover, we demonstrate 1. insufficient but necessary conditions for this subclass, 2. inclusion relations between the subclasses of SAPKA, and 3. concrete examples of this sub-class with reports of implementational experiments.

Keywords: public key exchange; security; asymmetric; asymmetric algorithm; cryptography; framework; limited computational power; computationally biased

1. Introduction

Since Shannon proposed the concept of “perfect secrecy” in crypto-systems [1], in which he introduced a theoretically unbreakable system even against computational power, the distribution of secret keys between the sender (Alice) and receiver (Bob) via an insecure channel has been one of the greatest problems in cryptography.

The Diffie–Hellman (DH) public key agreement (PKA) protocol that was proposed in 1976 [2] and the Rivest–Shamir–Adleman (RSA) crypto-system that was presented in 1978 [3] represented the most significant works in the area of cryptography, and it was previously believed that the problem of key distribution had been resolved.

However, recent considerable developments in the computational power of eavesdroppers have introduced several potential (even if not immediate) threats against standard PKA algorithms and public key cryptographies, particularly for small key lengths [4]. To maintain security, users have been forced to select longer keys, and the increased key length has led to higher computational costs. Thus, the preparation of secure communication infrastructure, particularly for devices with limited memory and computational power, has become challenging. Furthermore, the threat of quantum computers that are currently under development and Shor’s algorithm [5] cannot be underestimated.

Considering the demand for algorithms that are resilient against any type of theoretical attack, including quantum algorithm-based attacks, the development and study of new PKA algorithms and public key cryptographies, namely post-quantum cryptography...
Appl. Sci. 2021, 11, 5540

2 of 32

(PQC), has become widespread. PKAs and public key cryptographies based on lattice problems such as the shortest vector problem (SVP), closest vector problem, and learning with errors (LWE) are among the most well known methods. Among these, SVP-based PKA and public key cryptographies, including NTRU prime [6], NTRU-HRSS-KEM [7], and NTRU Encrypt [8], module LWE-based PKA such as CRYSTALS–Kyber [9], and ring LWE-based PKA including NewHope [10] are leading approaches in this research area and have been considered as candidates for the NIST (National Institute of Standards and Technology) standardization of PQC systems [11,12]. When the parameters are properly selected, the above algorithms are considered to be resilient against attacks that use quantum computers and sufficiently computationally efficient to be used in practice.

However, there has been substantial discussion regarding the security of such algorithms. For example, in certain LWE-based algorithms, even if sufficiently large parameters are selected, a possible weakness has been observed [13–15]. Moreover, the notion that the difficulty in solving ring LWE is equivalent to that of solving the LWE (the difficulty of LWE is discussed in [14,16]) has not yet been proven; thus, other cases of weakened security [9,10] may arise. Weak parameters for NTRU-type PKA algorithms and public key cryptography are also reported in [17]. Owing to these uncertainties in the parameter settings to maintain security even in an ideal situation (i.e., without assuming limited memory and computational power), the preparation of secure communication infrastructure with these new-generation algorithms for less capable devices has resulted in greater difficulty and insecurity. Although there is no doubt that these algorithms will offer significant benefits even after the post-quantum computer era, security analysis of these algorithms should continue until users can be provided with a “guide” that explains how to set parameters to ensure secure PKA and public key cryptography according to the needs and environments of users.

1.1. Research Concept and Goals

We define a function $C : \mathbb{N} \times \mathbb{R}^+ \to \mathbb{R}^+$, which shows the computational costs for $T$ calculation steps by a device with efficiency $E_D$ (calculation steps per time) as follows:

$$C(T, E_D) := T \times \frac{1}{E_D}. \tag{1}$$

Thus, it is given by time (s, ms, or another unit). Let $PKA$ be a set of PKA algorithms and let $T_A : PKA \times \mathbb{N} \to \mathbb{N}$ be a function that shows the calculation steps required for Alice to calculate the $N \in \mathbb{N}$ bit length of the secret shared key (SSK) of an algorithm $Alg \in PKA$, which increases monotonically for $N$. $T_B$ denotes the calculation steps for Bob in a similar manner.

Next, we consider $Alg \in PKA$, which has the following relation:

$$T_A(Alg, N) = T_B(Alg, N) \tag{2}$$

for all $N \in \mathbb{N}$. In this case, we suppose that the maximal computational cost that Bob is allowed to incur for the SSK calculation of $Alg \in PKA$, denoted by $C^B_{max}$, is $C^B_{max} = T_B(Alg, N_0) \times \frac{1}{E_B}$, which is achieved when the bit length of the SSK is some $N_0 \in \mathbb{N}$, and Bob can compute their SSK for all $N \in \mathbb{N}$ to satisfy:

$$C_B(T_B(Alg, N), E_B) := T_B(Alg, N) \times \frac{1}{E_B} \leq C^B_{max} := T_B(Alg, N_0) \times \frac{1}{E_B}, \tag{3}$$

where $E_B$ denotes the device efficiency of Bob. As $T_B$ is a monotonically increasing function for the bit size of the SSK, condition (2) is reduced to $N \leq N_0$. Furthermore, if $E_B = E_A$, where $E_A$ denotes the device efficiency of Alice, the SSK can be computed for all $N \leq N_0$.

Thus, Alice and Bob can calculate the SSK of a bit size that is equal to or less than $N_0$ within time $C^B_{max}$.
For the same $Alg \in PKA$, we assume that $E_B > E_A$ and the maximal computational cost that Bob is allowed to incur $C_{max}^B$ is the same as (2), where $N_0$ is the smallest bit size of SSK to maintain security. Let Alice’s computational cost for calculating her SSK of $N \in \mathbb{N}$ bits be $C_A(T_A(Alg, N), E_A) := T_A(Alg, N) \times \frac{1}{E_A}$, and if Alice needs to calculate her SSK within the cost $C_{max}^B$ as well as Bob, the following relation must be satisfied:

$$C_A(T_A(Alg, N), E_A) = T_A(Alg, N) \times \frac{1}{E_A} \leq C_{max}^B = T_B(Alg, N_0) \times \frac{1}{E_B},$$

where the equality holds for some $N_1 \in \mathbb{N}$, but in this case, $N_1 < N_0$ must be satisfied because $T_A$ is a monotonically increasing function for the bit size of the SSK and $\frac{1}{E_A} < \frac{1}{E_B}$. This observation indicates that they must either use an $N_1$-bit SSK, which is obviously less secure than when using an $N_0$-bit SSK, or let Alice incur a cost of $T_A(Alg, N_0) \times \frac{1}{E_A}$, which is larger than $C_{max}^B$. Most PKA algorithms, including the DH algorithm, satisfy (1), and there are many cases in which $E_B > E_A$ in modern society where IoT techniques are continually being developed; thus, this situation is inevitable in the near future, if not immediate.

We consider determining an algorithm denoted by $Alg_{A,B} \in PKA$, where

$$T_A(Alg_{A,B}, N) < T_B(Alg_{A,B}, N)$$

being satisfied for all $N \in \mathbb{N}$ is one solution to the above undesirable situation. We denote the maximal computational cost that Bob is allowed to incur as $C_{max,A,B}^B := T_B(Alg_{A,B}, N_0) \times \frac{1}{E_B}$, where $N_0 \in \mathbb{N}$ and it is the smallest bit size of SSK to maintain security. In addition to the above $T_A = T_B$ case, we suppose that both Bob and Alice must calculate her SSK within the maximal computational cost that Bob is allowed to incur $C_{max,A,B}^B$. In this case, Alice can calculate all $N$ bits of the SSK to satisfy

$$C_A(T_A(Alg_{A,B}, N), E_A) = T_A(Alg_{A,B}, N) \times \frac{1}{E_A} \leq C_{max,A,B}^B = T_B(Alg_{A,B}, N_0) \times \frac{1}{E_B}.$$

The equality holds when $T_A(Alg_{A,B}, N_1) \times \frac{1}{E_A} = T_B(Alg_{A,B}, N_0) \times \frac{1}{E_B}$ holds for some $N_1 \in \mathbb{N}$. In this case, it should be noted that $N_1 = N_0$ is achieved; that is, Alice calculating the SSK of $N_0$ bits within time $C_{max,A,B}^B$ is possible, provided that

$$\frac{T_A(Alg_{A,B}, N_0)}{T_B(Alg_{A,B}, N_0)} \leq \frac{E_A}{E_B}$$

holds, which is impossible when (1), because in this case, the left-hand side is equal to 1, but the right-hand side is less than 1. As $\frac{E_A}{E_B}$ is given (we may say that $\frac{E_A}{E_B}$ is a communication environment in which the algorithm is used), (4) is not always achieved for some $Alg_{A,B} \in PKA$ and $N_0 \in \mathbb{N}$. Conversely, we can determine the minimal environment $\frac{E_A}{E_B}$, where Alice and Bob can calculate $N_0$ bits SSK within $C_{max,A,B}^B$ time using $Alg_{A,B} \in PKA$ by simply calculating the left-hand side of (4).

Based on the above considerations, our research goals are as follows:

1. Constructions of $Alg_{A,B} \in PKA$.
2. For $Alg_{A,B} \in PKA$, the determination of $T_A(Alg_{A,B}, N_0)$ and $T_B(Alg_{A,B}, N_0)$ for any $N_0 \in \mathbb{N}$.
3. The construction of the PKA class to which $Alg_{A,B} \in PKA$ belongs and the introduction of conditions for PKA algorithms to be members of this class.

As mentioned above, goal 2 provides a lower bound of $\frac{E_A}{E_B}$ to calculate the SSK of $N_0$ bits within time $C_{max,A,B}^B$. Goal 3 provides instructions on how to construct PKA algorithms to possess the relation (3). Thus, improving algorithms such as those of [9,10] to possess this property may be possible by attempting to fix their parameters according to
the class conditions. We do not attempt to improve these algorithms in this study, but this subject is worthy of consideration and will be one of our most important future works.

We consider that these goals are achievable by fully utilizing the characteristics of the PKA framework known as strongly asymmetric public key agreement (SAPKA) [18]. The characteristics, high level of generality, and asymmetry of the key agreement process of SAPKA are explained in Section 1.2, along with its definition, and concrete methods that are derived from the characteristics are explained in Section 2.

Note that this study is not focused on how to construct secure PKA algorithms against any types of theoretical attacks; rather, it investigates how to reduce Alice’s computational complexity while maintaining the security of one given PKA algorithm. Our main theorems (in Section 5) do not provide any instructions on how to enhance the security of PKA algorithms, and resilience against attack such as man-in-the-middle (MITM) attack is not discussed in this paper (we consider that these topics should be discussed after existence of $A|g_{A < B}$ is proven and mentioned in Section 7.1).

1.2. SAPKA Framework

We provide a brief definition of SAPKA (the explicit definition is presented in Section 2.4) and its characteristics in this section.

First, Bob prepares a multiplicative semi-group $S$ with 1. Subsequently, he selects five maps:

$$x_1, x_2, x_3, x_4, N_1 : S \rightarrow S,$$

where $N_1$ must be an easily invertible map. In this case, “easily” means that the calculation of $N_1^{-1} \circ N_1(y)$ for all $y \in S$ can be performed in polynomial time. Furthermore, $x_1, x_2, x_3,$ and $x_4$ must satisfy the following equation, which is known as the compatibility condition:

$$x_1 \circ x_2(y) = x_3 \circ x_4(y) \quad (5)$$

for all $y \in S$, where $\circ$ denotes the map composition. Equation (5) is a condition for Alice and Bob to calculate the same SSK (see the key agreement process in Figure 1). The key agreement process of SAPKA can be described as in Figure 1, and every secret/public key is displayed in Table 1.

![Figure 1. Key agreement process of SAPKA.](image-url)
Table 1. Complete list of key of SAPKA.

| Key   | Parameter                      |
|-------|--------------------------------|
| Bob   | Secret keys \( x_1, x_2, x_3, x_4, N_1 : S \rightarrow S \) |
|       | Public keys \( y_{B,1} = N_1^{-1} \circ x_4 : S \rightarrow S \) |
|       | \( y_{B,2} = x_1 \circ x_2 : S \rightarrow S \) |
| SSK   | \( \kappa_B = x_3 \circ x_4(x_A) \in S \) |
| Alice | Secret key \( x_A \in S \) |
|       | Public key \( y_A = N_1^{-1} \circ x_4(x_A) \in S \) |
| SSK   | \( \kappa_A = x_1 \circ x_2(x_A) \in S \) |

As can be observed from Figure 1 and Table 1, Bob’s public keys are described by the map compositions and not by the element of \( S \). Sending a map means sending the calculation rule of the map in combination with a set of parameters, which is the domain of the map. Thus, Alice simply follows the rules of \( y_{B,1} \) and \( y_{B,2} \) to calculate \( y_A \) and \( \kappa_A \), and to calculate these, she must first receive \( y_{B,1} \) and \( y_{B,2} \) from Bob. Regardless of the \( x_A \) that Alice selects (provided that \( x_A \in S \)), the equality of \( \kappa_A \) and \( \kappa_B \) holds, because the compatibility condition (5) holds for all elements of \( S \). The generality mentioned above arises from the fact that there are only several restrictions for the secret keys of Bob, namely \( x_1, x_2, x_3, x_4, N_1 \), and semi-group \( S \). As the restrictions are only those in (5) and invertible regarding \( N_1 \), Bob has substantial freedom in terms of the choices of these maps and the algebraic structure. By fixing these maps and \( S \) concretely, various PKA algorithms can be described, including the most well known of these, namely the DH algorithm (presented in [18]). In this study, we do not attempt to describe new-generation algorithms such as [7,9,10] in the form of SAPKA. However, we are optimistic that these can be described because \( S \) can be selected as not only scalars but also matrices, for example, with numerous options for \( x_1, x_2, x_3, x_4, N_1 \).

Another notable characteristic of SAPKA is the asymmetry of the key agreement process. In this case, the asymmetry means that the number of public keys calculated by Alice and Bob differ, and thus, the two perform essentially different operations. Owing to this characteristic, an eavesdropper (Eve) must attempt attacks against a maximum of two public keys to obtain the secret information of either Bob or Alice. This may allow Alice to select her secret key from a set of small bit sizes and to reduce her computational complexity in certain cases. In Section 2, we explain Eve’s strategies for recovering the SSK from public keys, an observation from her strategies, and the research method derived from this observation.

2. Methods and Abstract of This Study

We explain the strategies for Eve to recover the SSK and the method for our research goal, which can be derived from the observation of her strategies.

2.1. Eve’s Strategies for Recovering SSK

Eve wishes to calculate \( \kappa_A \) or \( \kappa_B \) by determining the secret key of Alice or Bob from the public keys. She knows the two composed maps of Bob and one element of Alice, as follows:

\[
y_{B,1}(y) = N_1^{-1} \circ x_4(y) \quad (y \in S),
\]

\[
y_{B,2}(y) = x_1 \circ x_2(y) \quad (y \in S),
\]

\[
y_A = N_1^{-1} \circ x_4(x_A).
\]
1. Introduce algorithms from SAPKA (Section 3).

This means that Bob may take substantially more responsibility for maintaining security. Requirement 2. It is difficult to determine the map from (6) and (8):

\[ x_A = y_{B,1}^{-1}(y_A) = x_4^{-1} \circ N_1 \circ N_1^{-1} \circ x_4(x_A). \]  

(9)

Subsequently, she can calculate

\[ \kappa_A = x_1 \circ x_2(x_A). \]

If \( x_4 \) is not an invertible map, she obtains a set \( \{ x_E \in S : y_A = N_1^{-1} \circ x_4(x_E) \} \) instead of an element from (9). However, in this case, she can calculate the SSK as follows:

\[ x_1 \circ x_2(x_E) = x_3 \circ x_4(x_E) = x_3 \circ N_1 \circ N_1^{-1} \circ x_4(x_E) = x_3 \circ N_1(y_A) = \kappa_B. \]  

(10)

The second equation of (10) is obtained from the compatibility condition (5), and the final equation is obtained from the definition of \( \kappa_B \) (see Figure 1 or Step 5 of Section 2.4).

Eve’s Strategy 2

First, Eve attempts to obtain a map \( N_1 \) from (6) and then attempts to obtain a map \( x_{3,E} : S \to S \) that satisfies

\[ x_{3,E} \circ N_1 \circ N_1^{-1} \circ x_4(y) = x_1 \circ x_2(y) \]  

(11)

for all \( y \in S \). Finally, she can calculate the SSK as follows:

\[ x_{3,E} \circ N_1(y_A) = x_{3,E} \circ N_1 \circ N_1^{-1} \circ x_4(x_A) = x_1 \circ x_2(x_A) = \kappa_A. \]  

(12)

Suppose that Bob constructs \( y_{B,1} \) and \( y_{B,2} \) to satisfy the following two requirements:

Requirement 1. It is difficult to calculate \( x_4^{-1} \circ N_1 \circ N_1^{-1} \circ x_4(y) = y \) for all \( y \in S \) in real time, or it is difficult to determine the invertible map of \( y_{B,1}, x_4^{-1} \circ N_1 \) in real time.

Requirement 2. It is difficult to determine the map \( N_1 \) from \( y_{B,1} = N_1^{-1} \circ x_4 \) in real time, or it is difficult to obtain a map \( x_{3,E} \) to satisfy (11) in real time.

Then, it is difficult for Eve to obtain \( x_A \) from (9) and to proceed to (12) in real time; that is, Eve cannot obtain the SSK in real time. It should be noted that the secret key \( x_A \) that Alice selects is not strongly related to Eve’s breaking complexity owing to Requirement 1. This means that Bob may take substantially more responsibility for maintaining security than Alice. In this case, Alice can select her secret key space as a small one in terms of the bit size, provided that an exhaustive search for \( x_A \) is difficult in real time, and we expect that this will reduce Alice’s computational complexity for \( y_A \) and \( \kappa_A \).

2.2. Methods

According to this observation, the methods that can be established for the research goal can be derived as follows:

1. Introduce algorithms from SAPKA (Section 3).

2. Estimate the breaking complexity of these algorithms under the assumption that the secret key space of Alice is smaller than that of Bob and verify whether or not Alice’s small key space reduces the breaking complexity of the algorithms (Sections 4.2 and 4.3).

3. For algorithms for which Alice’s small key space does not reduce the breaking complexity, estimate their computational complexity to calculate the keys of both Alice and Bob (Sections 4.4 and 4.5).

4. Construct the SAPKA subclass for which the algorithms possess the relation that the complexity of Alice for her keys is smaller than that of Bob (we call this subclass the “main SAPKA subclass” until we give definition of it) and introduce the necessary conditions for the subclass by generalizing the results of 3 (Sections 5.1 and 5.2).
In Section 6, we implement some of the algorithms in Section 3 and report the experimental results. We can concretely observe what can be offered by the algorithms of the subclass constructed in Section 5.1.

At the beginning of Section 5, we add restrictions to the SAPKA framework, particularly for the public keys of Bob, before discussing our main themes. We explain these restrictions briefly below.

2.3. Restrictions on SAPKA Framework

We have already explained the high level of generality of SAPKA. However, owing to this generality, algorithms that do not ensure secure PKA are also included in this class. We present one example as follows:

Let \( S := \mathbb{Z}_p \) for some prime number \( p \). Bob selects the numbers \( x_B, n_B \in S \) and keeps them secret. Let the maps \( x_1, x_2, x_3, x_4, N_1 : S \to S \) be defined, for \( y \in S \), as:

- \( x_1(y) := x_B y \)
- \( x_2 := id \)
- \( x_3(y) := x_B y \)
- \( x_4(y) := id \)
- \( N_1(y) := n_B y. \)

In this case, all keys are described as in Table 2.

**Table 2. Complete list of key of example algorithm of SAPKA.**

| Key      | Parameter                                                                 |
|----------|---------------------------------------------------------------------------|
| Bob      | maps \( x_1, x_2, x_3, x_4, N_1 : S \to S \) and elements \( x_B, n_B \in S \) |
| Public keys | \( y_{B,1}(y) = N_1^{-1} \circ x_4 \circ x_2(y) = n_B^{-1} y \) (\( y \in S \)) |
| Public keys | \( y_{B,2}(y) = x_1 \circ x_2(y) = x_B y \) (\( y \in S \)) |
| SSK      | \( \kappa_B = x_3 \circ x_4(x_A) = x_B x_A \)                            |
| Alice    | \( x_A \in S \)                                                          |
| Public key | \( y_A = N_1^{-1} \circ x_4(x_A) = n_B^{-1} x_A \)                       |
| SSK      | \( \kappa_A = x_1 \circ x_2(x_A) = x_B x_A \)                            |

The SSK calculations are performed as follows:

\[
\kappa_A = x_1 \circ x_2(x_A) = x_B x_A
\]

\[
\kappa_B = x_3 \circ N_1(y_A) = x_3 \circ N_1 \circ N_1^{-1} \circ x_4(x_A) = x_B x_A.
\]

In this case, Bob sending \( y_{B,1} \) means that \( n_B^{-1} \) must be a public key, although this element is supposed to be a secret; otherwise, Alice cannot calculate \( y_A \). Eve can easily calculate Alice’s secret key \( x_A \) with \( n_B n_B^{-1} x_A = x_A \); thus, Eve can recover the SSK by calculating \( y_{B,2}(x_A) = x_1 \circ x_2(x_A) \). Without limiting this type of algorithm to be included in the SAPKA class, the main subclass that we attempt to construct in Section 5 can include weak algorithms. In this case, the subclass cannot be the one that we aim to construct.

To limit algorithms such as the above example, we add restrictions on the construction of Bob’s public keys using a map referred to as a “non-easily invertible map” with the following definition.

**Definition 1.** The map \( f_S : S \to S \) is called as a non-easily invertible map if the calculation \( f_S^{-1} \circ f_S(y) \) for all \( y \in S \) is difficult to compute in the mean of the computational complexity, and the order of complexity for computing \( f_S^{-1} \circ f_S(y) \) (\( y \in S \)) is equal to or greater than \( O(2^{\theta(n)}) \), where \( n \) is the bit size of \( y \) and \( t : \mathbb{N} \to \mathbb{R}^+ \) is a monotonically increasing function:

\[
t(n) = an
\]
for some $a \in \mathbb{R}^+ \setminus \{0\}$.

Of this definition is Landau’s big-$O$ notation (Definition 5 in Section 4.1). We limit Bob’s public keys $y_{B1}$ and $y_{B2}$ to be constructed as

$$y_{B1}(y) = N_1^{-1} \circ x_4(y) = f_{\hat{x}_1} \circ N_1^{-1} \circ x_4(y)$$

$$y_{B2}(y) = x_1 \circ x_2(y) = f_{\hat{x}_1} \circ x_1 \circ x_2(y)$$

for certain non-easily mapped $f_{\hat{x}} : S \rightarrow S$ and certain maps $x'_1, x'_2, x'_4, N'_1 : S \rightarrow S$. By doing so, attacks from Eve cannot be performed within a polynomial time. Furthermore, with several additional restrictions in Section 5, Requirement 2 of Section 2.1 can be achieved.

We attempt 4. of the method mentioned in Section 2.2 within this restricted SAPKA class with a non-easily invertible map (the explicit definition of this restricted SAPKA by a non-easily invertible map is also provided Section 5).

2.4. Explicit Definition of SAPKA

We present the definition of SAPKA [18]. The SAPKA algorithms have the following common ingredients:

- a multiplicative semi-group $S$ with 1;
- a set $\hat{M}_S$ of easily invertible maps $: S \rightarrow S$, known as noise space; and
- a set $M_S$ of maps $: S \rightarrow S$.

In the above, $S$ is public, and $\hat{M}_S$ and $M_S$ belong to Bob’s secret. From the key space,

$$K_B := M_S \times M_S \times M_S \times M_S \times \hat{M}_S.$$ 

Bob prepares the quintuple $(x_1, x_2, x_3, x_4, N_1)$ as their secret key, and $x_1, x_2, x_3, x_4$ must satisfy the following condition, which allows Alice and Bob to obtain the SSK.

**Definition 2.** Let $S$ be a multiplicative semi-group with 1. If the functions $x_1, x_2, x_3, x_4 : S \rightarrow S$ satisfy the following condition for all $y \in S$:

$$x_1 \circ x_2(y) = x_3 \circ x_4(y),$$

the maps $x_1, x_2, x_3, x_4$ are said to be compatible, where $\circ$ denotes a map composition.

**Definition 3.** For a multiplicative semi-group $S$ with 1 and the quintuple $C := (x_1, x_2, x_3, x_4, N_1) \in K_B$, if the maps $x_1, x_2, x_3, x_4$ satisfy (13), we state that $C$ is a member of the SAPKA class. We express this relation as:

$$C \in SAPKA.$$ 

The key agreement process of SAPKA is described as follows:

Step 1B Bob prepares the maps $(x_1, x_2, x_3, x_4, N_1) \in K_B$, of which $x_1, x_2, x_3, x_4$ satisfy (13).

In this case, each of $x_1, x_2, x_3, x_4, N_1$ is their secret key.

Step 2B Bob constructs their public keys $y_{B1}, y_{B2}$ as a map for each:

$$y_{B1} := N_1^{-1} \circ x_4$$

$$y_{B2} := x_1 \circ x_2$$

and sends $(y_{B1}, y_{B2})$ to Alice.

Step 1A Alice selects her secret key $x_A$ from $S$.

Step 2A Alice calculates her public key $y_A$ as follows:

$$y_A := y_{B1}(x_A) = N_1^{-1} \circ x_4(x_A)$$
and sends it to Bob.

Step 3A Alice calculates the SSK denoted by $\kappa_A$ as follows:

$$\kappa_A := y_{B,2}(x_A) = x_1 \circ x_2(x_A).$$

Step 3B Bob calculates the SSK denoted by $\kappa_B$ as follows:

$$\kappa_B := x_3 \circ N_1(y_A) = x_3 \circ N_1 \circ N^{-1}_1 \circ x_4(x_A) = x_3 \circ x_4(x_A).$$

2.5. Abstract of This Study

Here, we give an outline of this study, including research goal, short abstract of each section.

Research Goal

Construction of algorithms that possess the property that Alice’s complexity for SSK is smaller than that of Bob’s and introduction of SAPKA subclass that includes algorithms of the same property.

Section 3: Concrete Examples of SAPKA

We show concrete examples that how Alice and Bob calculate the SSK, respectively, when the key agreement process is asymmetric. Three examples that are normal DH of SAPKA description, noise element included type DH, and matrix type DH are shown.

Section 4: Breaking Complexity of SAPKA Algorithms

In this section, we demonstrate what problems that the algorithms of Section 3 are reduced to. Estimations of breaking complexities are done under the assumption that Alice’s secret key space is smaller than Bob’s secret key space. Here, we can see how problems algorithms reduced to are varied according to algebraic structure and SAPKA parameters. Finally, using some of SAPKA algorithms, we discuss how small Alice’s secret key space can be while maintaining security.

Section 5: Generalization

Here, we try to construct the SAPKA subclass of property that Alice’s computational complexity for SSK is smaller than that of Bob. At first, we exclude algorithms of weak security from the subclass framework as we mentioned in Section 2.3. Without this procedure, the subclass can contain algorithms of above property but sufficiently weak security so that Eve can obtain the SSK within polynomial time. After this limitation is done, we define the subclass and introducing some conditions (only necessarily conditions for algorithms into the subclass) by expanding the results of Section 4 so that it can be applied to the SAPKA framework.

3. Concrete Examples of SAPKA

3.1. DH

As demonstrated in [18], the SAPKA class includes the DH algorithm. However, the process is symmetric because, in this case, $S := \mathbb{Z}_p$, where $p$ is a prime number (thus, $S$ forms a finite field), $g \in S$ is the public parameter, $N^{-1}_1$ is the identity map, and Alice can calculate the public key $y_A = x_4(x_A) = g^{x_A}$, even before receiving $y_{B,1}$. The process is asymmetric in the sense that Alice and Bob use different secret information, but it is symmetric in the sense that the two perform the same operations independently.

3.2. Noised DH (NDH)

The following example is a variant of the DH that does not improve its security substantially but is useful to illustrate the concept of SAPKA algorithms simply.
The key agreement process is as follows:

Step 1B Bob prepares the quintuple $C_{NDH} := (x_1, x_2, x_3, x_4, N_1) \in K_B$ as follows:

- $x_1(y) := (g^{x_B})^y$
- $x_2 := id$
- $x_3(y) = y^{x_B}$
- $x_4(y) = g^y$
- $N_1(y) := y^{x_B^{-1}}$

where $y$ is selected arbitrarily from $S$.

Step 2B Bob constructs their public keys $y_{B,1}, y_{B,2}$ as a map for each

$$y_{B,1}(y) := N_1^{-1} \circ x_4(y) = N_1^{-1}(g^y) = (g^y)^{y_{B,2}} = S^{y_{B,2}}y$$

$$y_{B,2}(y) := x_1 \circ x_2(y) = (g^{x_B})^y$$

and sends $(y_{B,1}, y_{B,2})$ to Alice. This is equivalent to sending $g^{y_{B,1}}, S^{y_{B,2}} \in S$.

Step 1A Alice selects her secret key $x_A$ from $S$.

Step 2A Alice calculates her public key $y_A$ as follows,

$$y_A := y_{B,1}(x_A) = N_1^{-1} \circ x_4(x_A) = S^{x_B x_A}$$

and sends it to Bob.

Step 3A Alice calculates the SSK denoted by $\kappa_A$ as in the DH case:

$$\kappa_A := y_{B,2}(x_A) = x_1 \circ x_2(x_A) = (S^{x_B})^{x_A} = S^{x_B x_A}.$$ 

Step 3B Bob calculates the SSK denoted by $\kappa_B$ as follows:

$$\kappa_B := x_3 \circ N_1(y_A)$$

$$= x_3 \circ N_1 \circ N_1^{-1} \circ x_4(x_A)$$

$$= x_3 \circ N_1(S^{x_B x_A})$$

$$= x_3(S^{x_B^{-1} x_B x_A})$$

$$= (S^{x_A} b)$$

$$= S^{x_B x_A}.$$ 

### 3.3. Schur Exponentiation-Based DH (SEDH)

Let $S$ be the multiplicative semi-group $S := M(d, \mathbb{Z}_p)$ of $d \times d$ matrices with entries in $\mathbb{Z}_p$. The algorithm introduced in this section uses Schur exponentiation; that is, element-wise matrix exponentiation. The symbol $c^M$ is defined as follows:

$$(c^M)_{ij} := \begin{cases} 
M^i_j & \text{if } M \text{ is a scalar} \\
M^i_{M^i_{ij}} & \text{if } M \text{ is a matrix}
\end{cases} ; \quad i, j \in \{1, \ldots, d\}.$$ 

The key agreement process is as follows:

Step 1B Bob selects a matrix $x_B \in S$, an invertible matrix $N_B \in S$, and a primitive element $g$ of $\mathbb{Z}_p$. Subsequently, he constructs the quintuple $C_{SE} := (x_1, x_2, x_3, x_4, N_1) \in K_B$ for all $a, b \in \{1, \ldots, d\}$ as follows:

- $x_1(y)_{a,b} := \prod_{i \in \{1, \ldots, d\}} (g^{x_B})^i_{a,1}^{y_{b,2}}$
• $x_2 := id$
• $x_3(y)_{a,b} := \prod_{l \in \{1, \ldots, d\}} (y)_{l,b}^{(x_B)_{a,l}}$
• $x_4(y) := g^{o y}$
• $N_1(y)_{a,b} := \prod_{l \in \{1, \ldots, d\}} (y)_{l,b}^{(N_B^{-1})_{a,l}}$

where $y \in S$. The compatibility condition holds for all $y \in S$ and $a, b \in \{1, \ldots, d\}$:

$$x_1 \circ x_2(y)_{a,b} = \prod_{l \in \{1, \ldots, d\}} (g^{o x_b})_{a,l}^{(y)_{l,b}} = \prod_{l \in \{1, \ldots, d\}} g^{(x_B)_{a,l}(y)_{l,b}}$$

$$= g^{\sum_{l \in \{1, \ldots, d\}} (x_B)_{a,l}(y)_{l,b}} = g^{(x_B y)_{a,b}} = (g^{x_B y})_{a,b}$$

$$x_3 \circ x_4(y)_{a,b} = \prod_{l \in \{1, \ldots, d\}} (g^{o y})_{l,b} = \prod_{l \in \{1, \ldots, d\}} g^{(y)_{l,b}(x_B)_{a,l}}$$

$$= g^{\sum_{l \in \{1, \ldots, d\}} (x_B)_{a,l}(y)_{l,b}} = g^{(x_B y)_{a,b}} = (g^{x_B y})_{a,b}. $$

Step 2B Bob constructs their public keys $y_{B,1}, y_{B,2}$ as a map for each

$$y_{B,1}(y)_{a,b} := N_1^{-1} \circ x_4(y)_{a,b} = \prod_{l \in \{1, \ldots, d\}} (g^{o y})_{l,b}^{(N_B)_{a,l}}$$

$$y_{B,2}(y)_{a,b} := x_1 \circ x_2(y)_{a,b} = \prod_{l \in \{1, \ldots, d\}} (g^{o x_b})_{a,l}^{(y)_{l,b}}$$

for all $a, b \in \{1, \ldots, d\}$. In this case, $y_{B,1}(y)_{a,b}$ can also be expressed as follows:

$$y_{B,1}(y)_{a,b} = \prod_{l \in \{1, \ldots, d\}} (g^{o y})_{l,b}^{(N_B)_{a,l}} = \prod_{l \in \{1, \ldots, d\}} (g^{o N_B})_{a,l}^{(y)_{l,b}}.$$

Bob sends the maps $(y_{B,1}, y_{B,2})$ to Alice. This is equivalent to sending the matrices $g^{N_B}, g^{o N_B} \in S$.

Step 1A Alice selects her secret key $x_A \in S$.

Step 2A Alice calculates her public key $y_A$ for all $a, b \in \{1, \ldots, d\}$ as follows:

$$y_{A,a,b} := y_{B,1}(x_A)_{a,b} = N_1^{-1} \circ x_4(x_A)_{a,b} = \prod_{l \in \{1, \ldots, d\}} (g^{o N_B(x_A)})_{a,l}^{(x_A)_{l,b}}$$

$$= (g^{o N_B x_A})_{a,b}$$

and sends it to Bob.

Step 3A Alice calculates the SSK denoted by $\kappa_A$ for all $a, b \in \{1, \ldots, d\}$ as follows:

$$\kappa_{A,a,b} := y_{B,2}(x_A)_{a,b} := x_1 \circ x_2(x_A)_{a,b} = \prod_{l \in \{1, \ldots, d\}} (g^{o x_b})_{a,l}^{(x_A)_{l,b}} = (g^{o x_B x_A})_{a,b},$$

Step 3B Bob calculates the SSK denoted by $\kappa_B$ for all $a, b \in \{1, \ldots, d\}$ as follows:

$$\kappa_{B,a,b} := x_3 \circ N_1(y_A)_{a,b} = x_3 \circ N_1^{-1} \circ x_4(x_A)_{a,b} = x_3 \circ N_1(g^{o N_B x_A})_{a,b}$$

$$= \prod_{l \in \{1, \ldots, d\}} (g^{o N_B x_A})_{a,l}^{(x_B N_B^{-1})_{a,l}} = g^{\sum_{l \in \{1, \ldots, d\}} (N_B x_A)_{l,b}(x_B N_B^{-1})_{a,l}} = g^{(x_B N_B^{-1})N_B x_A}_{a,b}$$

$$= g^{(x_B x_A)_{a,b}} = (g^{o x_B x_A})_{a,b}. $$

This process is illustrated in Figure 2.
Step 1B
Secrets of Bob
\[
\begin{align*}
N_B & \in GL(d, \mathbb{Z}_p) \\
X_B & \in M(d, \mathbb{Z}_p)
\end{align*}
\]
Step 2B
\[
y_{B,1} := g^{N_B} \\
y_{B,2} := g^{X_B}
\]
Step 3B
\[
\kappa_B := g^{X_B X_A}
\]
Step 1A
Secret of Alice
\[
X_A \in M(d, \mathbb{Z}_p)
\]
Step 2A
\[
y_A := g^{N_B X_A}
\]
Step 3A
\[
\kappa_A := g^{X_B X_A}
\]

Figure 2. Key agreement process of Schur exponentiation-based DH.

Remark 1. Let \( V \) be a vector space of \( V := \mathbb{Z}_p^d \). Each of the above maps \( x_1, x_2, x_3, x_4, N_1 \) can also be considered as the map \( V \rightarrow V \) and the condition
\[
x_1 \circ x_2(y)_a = \prod_{l \in \{1, \ldots, d\}} (g^{x_B})^y_{a,d} = (g^{x_B y})_a = \prod_{l \in \{1, \ldots, d\}} (g^{y x})^{x_B}_a = x_3 \circ x_4(y)_a. \tag{17}
\]
is satisfied for all \( y \in V \) and \( a \in \{1, \ldots, d\} \), where \( x_B, N_B \) are the same as above. Thus, Alice can select her secret key \( x_A \) from \( V \) and the computational complexity for calculating \( y_A, \kappa_A, \) and \( x_B \) is obviously reduced compared with the case when \( x_A \in S = \mathcal{M}(d, \mathbb{Z}_p) \). The precise breaking and computational complexities for the \( C_{SE} \) of both the \( x_A \in V \) and \( x_A \in S \) cases are investigated in the following section.

4. Breaking Complexity of SAPKA Algorithms

As mentioned in Section 2.2, we estimate the breaking complexity (Sections 4.2 and 4.3) and computational complexity (Section 4.4) of the algorithms in Sections 3.2 and 3.3 under the assumption in Section 4.1.

4.1. Assumptions

Let \( n \) and \( m \) be numbers in \( \mathbb{N} \), and \( n > m \). We define two functions \( | \cdot | : \mathbb{N} \rightarrow \mathbb{N} \) and \( | \cdot |_S : S \rightarrow \mathbb{N} \), which represent the bit lengths of the input. The difference between the two functions is the domains. The input of the first one is from \( \mathbb{N} \), and the second one is from \( S \). The semi-groups of the above two algorithms are constructed based on a prime number \( p \), where \(|p| = n \). In this case, we can express the bit size of the element in \( S \) as \(|y|_S \leq n \) when \( S = \mathbb{Z}_p \) and \(|y|_S \leq d^2n \) when \( S = \mathcal{M}(d, \mathbb{Z}_p) \). We construct subsets denoted by \( \mathcal{S} \) of \( S \) for both the \( S = \mathbb{Z}_p \) and \( S = \mathcal{M}(d, \mathbb{Z}_p) \) cases, respectively, as follows:
\[
\mathcal{S} := \{ y \in S : |y|_S \leq m \} \quad (\text{when } S = \mathbb{Z}_p \text{ case})
\]
\[
\mathcal{S} := \{ y \in S : |y|_S \leq d^2m \} \quad (\text{when } S = \mathcal{M}(d, \mathbb{Z}_p) \text{ case}),
\]
with the aim of determining whether the complexity of the algorithms in Sections 3.2 and 3.3 are reduced if we suppose that Alice’s secret key is selected from the small bit length set \( \mathcal{S} \).
Prior to investigating the breaking complexity of the algorithms in Sections 3.2 and 3.3 under the assumption and a certain condition for \( n \) and \( m \), we demonstrate that not only the complexity, but also the order of complexity (a definition of order is provided in Definition 5) to compute one discrete logarithm problem (DLP) differ between the cases when an exponent is selected from a larger bit length set and from a smaller bit length set. The required settings are as follows:

- a prime number \( p \), where \(|p| = n\);
- a semi-group \( S := \mathbb{Z}_p^*\);
- a subset \( \hat{S} := \{y \in S : |y|_S \leq m\} \subset S\); and
- a map \( x_g : S \to S\), \( x_g(y) := g^y \), where \( g \) is a primitive element of \( S\).

We prove that if \( n = s(m) > m \) for some monotonically increasing function \( s : \mathbb{N} \to \mathbb{N} \) of not a linear and monic polynomial, the calculation of the DLP, namely

\[
x_g^{-1} \circ x_g(y) = \log_x g^y = y,
\]

is easier when \( y \in \hat{S} \) than when \( y \in S \) in terms of the complexity and order (Proposition 1). First, we name this input \( y \) of \( x_g \) according to the size of set \( y \) as per the following definition:

**Definition 4.** Let \( n \) and \( m \) be numbers in \( \mathbb{N} \) \((n > m)\), and let \( n \) be the bit size of each element of \( \mathbb{Z}_p\). For a given map \( x_g : \mathbb{Z}_p \to \mathbb{Z}_p\), defined as

\[
x_g(y) := g^y,
\]

where \( g \) is an element in \( S\), if the set \( y \) belongs to \( \hat{S}\), we call this \( y \) an \( m \)-bit logarithm. If the set is \( S\), \( y \) is known as an \( n \)-bit logarithm.

The complexity of obtaining the \( n \)-bit logarithm refers to the complexity of calculating \( x_g^{-1} \circ x_g(y) \) when \( y \in S \) (for the \( m \)-bit logarithm, \( y \in \hat{S}\)). We represent the time complexity using the function \( T : \mathbb{N} \to \mathbb{R}^+\), where the input is the bit size and the output is the multiplication steps. The complexity of calculating the DLP of the \( m \)-bit logarithm is given by \( T(m) = 2^{\frac{m}{s(m)}} \) \([19,20]\). In this case, the complexity of calculating the \( n \)-bit logarithm is \( T(n) = T(s(m)) = 2^{\frac{s(m)}{m}} \). Obviously, \( T(n) > T(m) \) when \( n = s(m) > m \); however, we should also consider how the complexity increases as \( m \) increases. We can compare the growth rate of \( T(m) \) and \( T(s(m)) \) using Landau’s big-\( O \) notation with the following definition:

**Definition 5.** For functions \( T, g : \mathbb{N} \to \mathbb{R}^+\), if there exist \( c, m_0 \in \mathbb{N} \) such that

\[
T(m) \leq cg(m)
\]

is satisfied for all \( m \geq m_0 \), we state that \( T(m) \) has an order of \( g(m) \) time complexity, and we describe it as

\[
T(m) \in O(g(m)).
\]

Relation (19) indicates that \( O(g) \) is a set defined by:

\[
O(g) := \{T : \text{there exists } c, m_0 \in \mathbb{N} \text{ to satisfy (18) for all } m \geq m_0\}.
\]

Therefore, for two complexity functions \( T, T' : \mathbb{N} \to \mathbb{R}^+\), where \( T(m) \leq T'(m) \) for all \( m \in \mathbb{N} \) and \( t(m) \in O(g(m))\), \( T'(m) \in O(g'(m))\), if we wish to state that the growth rate of \( T' \) is higher than that of \( T \) (the order of \( T' \) is higher than \( T \)), we can simply describe it as follows:

\[
O(g'(m)) \subset O(g(m))
\]

or

\[
g'(m) > g(m)
\]
for all \( m \). In this case,

\[
T'(n) \not\in O(g(n)) \quad T(n) \in O(g(n)), T'(n) \in O(g'(n))
\]

hold. We can prove that the growth rate of \( t(s(m)) = 2^{\frac{\sigma(m)}{\sigma(n)}} \) is higher than \( T(m) = 2^{\frac{m}{n}} \) as follows:

**Proposition 1.** If and only if the monotonically increasing function \( s : \mathbb{N} \to \mathbb{N} \) for satisfying \( n = s(m) > m \) is a not linear and monic polynomial, the growth rate of \( T((s(m)) = 2^{\frac{\sigma(m)}{\sigma(n)}} \) is higher than \( T(m) = 2^{\frac{m}{n}} \); that is,

\[
O(2^{\frac{\sigma(m)}{\sigma(n)}}) \subset O(2^{\frac{m}{n}})
\]

holds.

**Proof.** Note that \( T(s(m)) \in O(2^{\frac{\sigma(m)}{\sigma(n)}}) \) and \( T(m) \in O(2^{\frac{m}{n}}) \). Suppose that \( s \) is a one-degree but non-monic polynomial defined as:

\[
n = s(m) = hm + b,
\]

where \( h, b \in \mathbb{N} \), \( h \neq 1 \). If \( O(2^{\frac{m}{n}}) = O(2^{\frac{\sigma(m)}{\sigma(n)}}) \) holds in this case, there must exist the constants \( c, m_0 \in \mathbb{N} \) such that for all \( m \geq m_0 \),

\[
2^{\frac{\sigma(m)}{\sigma(n)}} = 2^{\frac{hm+b}{\sigma(n)}} \leq c \cdot 2^{\frac{m}{n}} \quad (20)
\]

is satisfied. However, in this case, \( 2^{\frac{\sigma(m)}{\sigma(n)}} \leq c \) holds, so that \( c \) cannot be a constant. Thus, in this case,

\[
O(2^{\frac{\sigma(m)}{\sigma(n)}}) \subset O(2^{\frac{m}{n}})
\]

holds. When the case \( s \) is over a one-degree polynomial, it is obvious that there is no \( c, m_0 \in \mathbb{N} \) to satisfy (20) for all \( m \in m_0 \).

If \( s \) is a linear and monic polynomial described as

\[
n = s(m) = m + b,
\]

where \( b \in \mathbb{N} \), we have

\[
O(2^{\frac{\sigma(m)}{\sigma(n)}}) = O(2^{\frac{m+b}{n}}) = O(2^{\frac{m}{n}}) = O(2^{\frac{m+n}{n}}).
\]

Therefore, if Alice and Bob attempt a key exchange using the DH algorithm, where Bob’s secret key \( x_B \) is

\[
x_B \in S
\]

and Alice’s secret key \( x_A \) is

\[
x_A \in S.
\]

Eve should attempt to obtain \( x_A \), which is easier for her to obtain than \( x_B \). The security of the algorithms in Sections 3.2 and 3.3 also depends on the difficulty of the DLP, but these algorithms may not be the same as DH owing to one of Bob’s public keys \( y_{B,1} \) and especially the map \( N_1 \). In Section 2.1, we discussed the requirement for \( y_{B,1} \) to allow Alice to select her secret key from a small bit length set without reducing the breaking complexity. For both algorithms, Requirement 2 is satisfied if Bob selects \( n_B \), as it is an \( n \)-bit logarithm for \( x_B \) with the algorithm of Section 3.2, and similarly, \( N_B \) is selected with the algorithm of Section 3.3. Of course, in this case, the number \( n \) must be sufficiently large to ensure that the calculation of the \( n \)-bit logarithm is not achieved in real time.
At this point, we focus on verifying whether the other algorithms in Sections 3.2 and 3.3 satisfy Requirement 1 of Section 2.1. Subsequently, we attempt to determine the polynomial $s$; that is, how small $m$ can be and the computational complexity of these algorithms for each key of Alice and Bob when $m$ is selected to be as small as possible (Section 4.4).

4.2. Eve Attempts an Attack against NDH

We assume the following:

- a prime number $p$, where $|p| = n$;
- a semi-group $S := \mathbb{Z}_p$;
- a subset $\mathcal{S} := \{y \in S : |y|_S \leq m\} \subset S$;
- $x_A, n_B, y_B, x_B, \in \mathcal{S}$;
- a map $x_S : S \to S, x_S(y) := g^y$, where $g$ is a primitive element of $S$;
- a non-linear and monic polynomial $s$, where $n = s(m) > m$ for all $m \in \mathbb{N}$.

Eve's goal is to obtain $x_A$ from the following public keys

- $y_{B,1}(1) = y_{B,2}(y) \equiv N_1^{-1} \cdot x_4(y) = N_1^{-1} (g^y) = (g^y)^{nB} = g^{nBy}$
- $y_{B,2}(y) := x_1 \circ x_2(y) = (g^{x_B})^y$,

and public elements

- $g^{x_B}$
- $g^{nB}$
- $y_A = N_1^{-1} \cdot x_4(x_A) = g^{nBx_A}$.

She must calculate

$$y_{B,1}^{-1}(y_A) = x_4^{-1} \circ N_1(y_A) = \log_{g}(y_A)^{nB} = \log_{g} g^{nBx_A}x_A^{-1} = \log_{g} g^{x_A}x_A = x_A. \quad (21)$$

As she does not know $n_B$, it appears that she must obtain the $n$-bit logarithm $n_B$ by calculating $\log_{g} g^{nB} = n_B$ before obtaining the $m$-bit logarithm $x_A$. However, (21) can be described as

$$y_{B,1}^{-1}(y_A) = x_4^{-1} \circ N_1(y_A) = n_B^{-1} \log_{g} y_A = n_B^{-1} \frac{\log_{g} g^{nBx_A}}{\log_{g} g} = \log_{g} g^{nBx_A} = x_A.$$

Thus, when we define a map $x_{g'} : S \to S$ such that

$$x_{g'}(y) := x_{g^{nB}}(y) = (g^{nB})^y,$$

$y_{B,1}^{-1} = x_{g'}^{-1},$ we can construct $x_{g'}$ using only $g^{nB}$, which is public. For this attack, Eve only needs to calculate $x_{g}^{-1}(y_A) = x_{g}^{-1} \circ x_{g'}(x_A)$ to obtain the $m$-bit logarithm $x_A$. Therefore, it can be said that Alice selecting a small bit length set $\mathcal{S}$ reduces the breaking complexity for Eve; that is, Requirement 1 of Section 2.1 is not satisfied in this case.

4.3. Eve Attempts an Attack against SEDH

We assume the following:

- a prime number $p$, where $|p| = n$;
- a semi-group $S := \mathcal{M}(d, \mathbb{Z}_p)$;
- a subset $\mathcal{S} := \{y \in S : |y|_S \leq d^2m\} \subset S$;
- $x_A, N_B, x_B, \in \mathcal{S}$;
- a map $x_S : S \to S, x_S(y) := g^y$, where $g$ is a primitive element of $S$; and
- a not linear monic polynomial $s$, where $n = s(m) > m$ for all $m \in \mathbb{N}$.

Eve’s goal is to obtain $x_A$ from the following public keys:

- $y_{B,1}(y)_{a,b} := N_1^{-1} \circ x_4(y)_{a,b} = \prod_{i \in \{1, \ldots, d\}} (g^{x_B})_{1,b}.$
where Eve avoids calculating a key of Alice to calculate \( \log N \)
\( x \) of the attack with an exhaustive search for \( x \). We now attempt to find other descriptions for map \( y \) logarithms by calculating \( \log d \) equivalent to the complexity of calculating \( d \) of (22), and the complexity of calculating \( \log a \) for all \( y \).

In this case, \( X \) is described as
\[ y_{A} = N_{1}^{-1} \circ x_{4}(x_{A}) = \log_{b}(g^{oN_{B}x_{A}})_{a,b} = \log_{b}(g^{oN_{B}x_{A}})_{a,b} = (x_{A})_{a,b} \]
for all \( a, b \in \{1, \ldots, d\} \). It appears that she first needs to obtain an element \( N_{B} \) by calculating \( \log_{b}(g^{oN_{B}})_{a,b} \) for all \( a, b \in \{1, \ldots, d\} \), the complexity of which is equivalent to the complexity for calculating \( d^{2} \) \( n \)-bit logarithms for the map \( x_{A} \), because she does not know \( N_{B} \), so as to obtain \( N_{B}^{-1} \). After obtaining \( N_{B} \), she can proceed to the final two equalities of (22), and the complexity of calculating \( \log_{g}(g^{x_{A}})_{a,b} \) for all \( a, b \in \{1, \ldots, d\} \) is equivalent to the complexity of calculating \( d^{2} \) \( m \)-bit logarithms for the map \( x_{A} \). We now verify that Eve really needs to calculate at least the same amount as to obtain \( d^{2} \) \( m \)-bit logarithms by calculating \( \log_{g}(g^{oN_{B}})_{a,b} \) for all \( a, b \in \{1, \ldots, d\} \). In Attack 1, we attempt to find other descriptions for map \( y_{B,1} \), as in Section 4.2. In Attack 2, we attempt an attack in which Eve avoids calculating \( d^{2} \) DLPs for \( N_{B} \), and we compare the breaking complexity of the attack with an exhaustive search for \( x_{A} \in \mathcal{S} \). In Attack 3, we attempt to determine the least complexity that Eve requires to obtain \( x_{A} \). Hereafter, we refer to the logarithm of matrix \( M \in \mathcal{M}(d, \mathbb{Z}_{p}) \) as follows:

\[
\log_{g} M = (\log_{g} M_{a,b}) \quad (g \in \mathbb{Z}_{p})
\]

\[
\log_{G} M = (\log_{G} M_{a,b}) \quad (G \in \mathcal{M}(d, \mathbb{Z}_{p})).
\]

Attack 1

Using the known element \( g^{oN_{B}} \), Eve attempts the following calculation for all \( a, b \in \{1, \ldots, d\} \):

\[
\log_{g} g^{oN_{B}}(y_{A}) = \log_{g} g^{oN_{B}} g^{oN_{B}x_{A}}.
\]

However, unlike in Section 4.2, the relation \( \log_{g} g^{oN_{B}} g^{oN_{B}x_{A}} = x_{A} \) does not hold. She obtains \( X \in \mathcal{S} \) instead of \( x_{A} \in \mathcal{S} \) from (23):

\[
\log_{g} g^{oN_{B}} g^{oN_{B}x_{A}} = \log_{g} g^{oN_{B}} g^{oX} = X.
\]

In this case, \( X \) holds for the relation

\[ N_{B} \bullet X = N_{B}x_{A}, \]

where \( \bullet \) denotes element-wise matrix multiplication. Obviously, \( X = x_{A} \) does not always hold, so \( N_{1}^{-1} \circ x_{4}(y_{A}) = \log_{g} g^{oN_{B}}(y_{A}) \) also does not hold. Moreover, to obtain \( x_{A} \), Eve needs to calculate \( \log_{g} g^{oN_{B}} = N_{B} \).

Attack 2

We introduce an attack in which Eve does not calculate \( \log_{g} g^{oN_{B}} = N_{B} \). The public key of Alice \( y_{A} \) is described as

\[
y_{A} = \begin{pmatrix}
\Pi_{l \in \{1, \ldots, d\}} (g^{oN_{B}})^{(x_{A})_{1,l}} & \cdots & \Pi_{l \in \{1, \ldots, d\}} (g^{oN_{B}})^{(x_{A})_{l,d}} \\
\vdots & \ddots & \vdots \\
\Pi_{l \in \{1, \ldots, d\}} (g^{oN_{B}})^{(x_{A})_{d,l}} & \cdots & \Pi_{l \in \{1, \ldots, d\}} (g^{oN_{B}})^{(x_{A})_{l,d}}
\end{pmatrix}
\]
Step 1 For row \( b \in \{1, \ldots, d\} \) of \( y_A \), Eve creates a matrix \( W_b = (w_{a,l}^{(b)}) \in \mathcal{M}(d, \mathbb{Z}_p) \) to satisfy
\[
\begin{pmatrix}
(g^{aN_b})_{1,1} x_{A,1}^{(b)} & (g^{aN_b})_{1,2} x_{A,2}^{(b)} & \cdots & (g^{aN_b})_{1,d} x_{A,d}^{(b)} \\
(g^{aN_b})_{2,1} x_{A,1}^{(b)} & (g^{aN_b})_{2,2} x_{A,2}^{(b)} & \cdots & (g^{aN_b})_{2,d} x_{A,d}^{(b)} \\
\vdots & \vdots & \ddots & \vdots \\
(g^{aN_b})_{d,1} x_{A,1}^{(b)} & (g^{aN_b})_{d,2} x_{A,2}^{(b)} & \cdots & (g^{aN_b})_{d,d} x_{A,d}^{(b)} 
\end{pmatrix}
\]
\begin{equation}
\label{eq:24}
= \begin{pmatrix}
(1, \ldots, 1) \\
\vdots \\
(1, \ldots, 1)
\end{pmatrix}.
\end{equation}

Step 2 For all \( a, l \in \{1, \ldots, d\} \), Eve calculates
\[
v_{a,l} := \log_{(g^{aN_b})_{a,l}} w_{a,l}^{(b)}.
\]
\begin{equation}
\label{eq:25}
(26)
\end{equation}

Step 3 For all \( l \in \{1, \ldots, d\} \), Eve verifies whether or not
\[
v_{1,l} = v_{2,l} = \cdots = v_{d,l}
\]
is true.

Step 4 If (26) is true for all \( l \in \{1, \ldots, d\} \), Eve can recover row \( b \in \{1, \ldots, d\} \) of \( x_A \):
\[
\begin{align*}
(x_A)_{1,b} &= v_{1,1} = \cdots = v_{d,1} \\
(x_A)_{2,b} &= v_{1,2} = \cdots = v_{d,2} \\
\vdots \\
(x_A)_{d,b} &= v_{1,d} = \cdots = v_{d,d}.
\end{align*}
\]

As \( N_b \) is an invertible matrix, there exists exactly one \((X_{1,b}, X_{2,b}, \ldots, X_{d,b})^t \in \mathbb{Z}_p^d\) to satisfy system (24) for a given \( y_A, g^{aN_b} \) of row \( b \in \{1, \ldots, d\} \); thus, it must be \((x_{A,1,b}, x_{A,2,b}, \ldots, x_{A,d,b})^t \). If (26) is false in some \( l \in \{1, \ldots, d\} \), Eve returns to Step 1.

In Step 2, Eve needs to calculate at least two DLPs of \( m \)-bit logarithms; thus, she requires \( 2 \cdot 2^m \) multiplications. As \( \mathbb{Z}_p \) forms a group under multiplication, for any arbitrary element \( e \) in \( \mathbb{Z}_p \), there exists a unique \( w_{a,d}^{(b)} \in \mathbb{Z}_p \) such that:
\[
\epsilon w_{a,d}^{(b)} = (y_A)_{a,b}
\]
for each \( a \in \{1, \ldots, d\} \). Thus, Eve has \(2^m(d-1)\) choices for \((w_{a,d}^{(b)}, \epsilon w_{a,d}^{(b)}, \ldots, \epsilon^{m-1} w_{a,d}^{(b)}) \) and each \( \epsilon w_{a,d}^{(b)} \), and therefore, Eve must repeat Steps 2 to 4 at most \(2^m(d-1) \) times. The total multiplications required for this attack is denoted as:
\[
2^{nd(d-1)+\frac{m}{2}+1} = 2^{s(m)d(d-1)+\frac{m}{2}+1},
\]
which is much larger than the total multiplications required for an exhaustive search on each row of \( x_A \in \mathbb{S} \), namely:
\[
m2^d m
\]

Attack 3

Suppose that Eve attempts to determine \( x_A \) directly from the formula of Alice’s public key \( y_A \):
\[
(y_A)_{a,b} = \prod_{l \in \{1, \ldots, d\}} (g^{a N_B})_{a,l}^{(x_A)_{l,b}}.
\]
\begin{equation}
\label{eq:27}
(27)
\end{equation}
As \( N_B \) is an invertible matrix, this is equivalent to solving the following system for \( X \in S \):

\[
(y_A)_{a,b} = \prod_{l \in \{1, \ldots, d\}} (g^{o_N B_l})^{X_{a,l}}_{a,l}
\]

(28)

for all \( a, b \in \{1, \ldots, d\} \) and given \( y_A, g^{o_N B} \). The least complexity for finding \( X \) from (28) is expressed by the following theorem:

**Theorem 1.** The complexity of solving (28) for \( X \in S \) is equal to or greater than the complexity of obtaining \( N_B \) from \( \log_S g^{o_N B} \).

**Proof.** Suppose that Eve can determine \( X = x_A \in S \) from (28) for a given \( g^{o_N B} \) and \( y_A \) with a complexity denoted by the function \( T : N \to \mathbb{R}^+ \) defined as \( T(m) := p(m) \), where \( p \) is a polynomial. Under this assumption, Eve can solve system (28) even when \( x_A \in S \) (thus, \( X \in S \)) because, in this case, the complexity is \( T(s(m)) = p(s(m)) \), which is also a polynomial. It should be noted that solving (28) is equivalent to solving the following system for all \( a, b \in \{1, \ldots, d\} \):

\[
(y_A)^t_{a,b} = \prod_{l \in \{1, \ldots, d\}} (g^{o_N B_l})^{(X)^t_{a,l}}_{l,b}
\]

(29)

for given matrices \( (g^{o_N B_l})^t \), \( (y_A)^t \), where \( t \) denotes matrix transposition, because

\[
(y_A)^t_{a,b} = g^{o_N (x_A)^t(N_B)^t} = g^{o_N (x_A)^t(N_B)^t} = g^{\sum_{l \in \{1, \ldots, d\}} (x_A)^t_{a,l}(N_B)^t_{l,b}},
\]

\[
= \prod_{l \in \{1, \ldots, d\}} (g^{o_N B_l})^{(x_A)^t_{a,l}}_{l,b}
\]

for all \( a, b \in \{1, \ldots, d\} \). This emphasizes that Eve can solve the system of both variable matrices by left multiplication and right multiplication on the Schur exponent of a given matrix. Therefore, once she has obtained \( X \in S \) and calculates \( g^{o_N X} \) within polynomial time, the following system can also be solved for an unknown \( N_B \in S \):

\[
(y_A)_{a,b} = \prod_{l \in \{1, \ldots, d\}} (g^{o_N X})^{(N_B)^t_{a,l}}_{l,b}
\]

(30)

for all \( a, b \in \{1, \ldots, d\} \) within polynomial time. As \( N_B \) is the matrix logarithm of \( g^{o_N B} \), we can state that, if Eve can solve system (28) for \( X \) within polynomial time, she can obtain \( d^2 \) \( n \)-bit logarithms within polynomial time. \( \square \)

This theorem claims that the problem of calculating \( \log_S g^{o_N B} = N_B \) is polynomial-time-reducible to the problem of solving system (28) for \( X \in S \).

### 4.4. Determination of \( n \) and \( m \)

We investigate how large the bit sizes \( n \) and \( m \) must be, as well as the computational complexity of calculating the public keys and SSK for Alice and Bob according to \( n \) and \( m \). If the attacks on SEDH are only those in Section 4.3, Eve must execute an exhaustive search for each row of \( x_A \) (as the attack is performed by each row, Eve requires \( d \) times exhaustive searches within a vector space of size \( 2^{dm} \)) or must obtain \( N_B \), the difficulty of which is equivalent to obtaining \( d^2 \) \( n \)-bit logarithms, as in the above attacks. We assume that attacks against SEDH are only those described in Section 4.3, and we determine the sizes of \( n \) and \( m \). Considering that the difficulty for Eve to obtain \( N_B \) depends on the difficulty of the DLP and that of obtaining \( x_A \) directly from (24) by an exhaustive search depends on the vector space of size \( 2^{dm} \), \( n \) and \( m \) must satisfy the requirements in Table 3.
Table 3. Requirements for $n$ and $m$.

| Requirements                        |
|-------------------------------------|
| $n$                                 |
| Sufficiently large for one DLP      |
| $m$                                 |
| $2^{dm}$ must be sufficiently large for an exhaustive search |

As mentioned in Section 4, the complexity of one $n$-bit logarithm denoted by the function $T : \mathbb{N} \rightarrow \mathbb{R}^{+}$ is

$$T(n) = T(s(m)) = 2^{\frac{s(m)}{2}},$$

and the complexity of an exhaustive search for $x_A$ of each row denoted by the function $T' : \mathbb{N} \rightarrow \mathbb{R}^{+}$ is $T'(m) = 2^{dm}$. (In fact, Eve must solve $d^2$ DLPs or $d$ exhaustive searches for a vector space $\mathbb{Z}_{2^m}^d$, but $d^2$ and $d$ are so trivial for exponential functions that we consider these complexities for only one number.) If Alice and Bob would like the breaking complexity for one DLP and that for an exhaustive search for one row of $x_A$ to be equalized, the polynomial $s : \mathbb{N} \rightarrow \mathbb{N}$ mentioned at the beginning of Section 4.3; that is, the relation between $n$ and $m$, is obtained by the following theorem:

**Theorem 2.** If the time complexity for obtaining one $n$-bit logarithm is equal to the complexity of an exhaustive search within a vector space of size $2^{dm}$, the polynomial $s : \mathbb{N} \rightarrow \mathbb{N}$ for $m \in \mathbb{N}$ is obtained by

$$s(m) = 2dm.$$

**Proof.** We obtain

$$T(n) = T(s(m)) = T'(m) \iff \frac{s(m)}{2} = dm.$$  \hspace{1cm} (31)

With this relation, what does the computational complexity for Alice and Bob become?

### 4.5. Computational Complexity of SEDH for Alice and Bob

For a given $d \in \mathbb{N}$, approximately $dm$ multiplications are required if the bit size of the exponent is $n$ to calculate one scalar exponentiation. Furthermore, approximately $m$ multiplications are required if the bit size of the exponent is $m$. Thus, the calculation $g^{X}$ (32)

requires approximately $d^2s(m) = 2d^3m$ multiplications if $X \in \mathcal{S}$. If $X \in \overline{\mathcal{S}}$ it requires approximately $d^2m$ multiplications. Moreover, the calculation

$$\prod_{l \in \{1, \ldots, d\}} (Y_{a,l})^{(X)_{l,b}}$$  \hspace{1cm} (33)

requires $ds(m) = 2d^2m$ multiplications if $X \in \mathcal{S}$ and $dm$ multiplications if $X \in \overline{\mathcal{S}}$ for each $a, b \in \{1, \ldots, d\}$. Thus, according to (14) and (15), the computational complexities required by Alice for $y_A$ and her SSK denoted by the functions $T_{y_A}, T_{\kappa_A} : \mathbb{N} \rightarrow \mathbb{R}^{+}$ of $m$ are

$$T_{y_A}(m) := d^3m$$  \hspace{1cm} (34)

$$T_{\kappa_A}(m) := d^3m.$$  \hspace{1cm} (35)

From the calculations of $g^{y_A}, g^{N_B}$, and (16), Bob’s computational complexities for $y_{B,1}, y_{B,2}$, and their SSK, denoted by the functions $T_{y_{B,1}}, T_{y_{B,2}}, T_{\kappa_B} : \mathbb{N} \rightarrow \mathbb{R}^{+}$ of $n$, become

$$T_{y_{B,1}}(n) := d^2n$$  \hspace{1cm} (36)
\[
T_{yB_2}(n) := d^2 n \quad (37)
\]
\[
T_{xB}(n) := d^3 n \quad (38)
\]
As \( n = s(m) = 2dm \), \( T_{yB_1}, T_{yB_2}, \) and \( T_{xB} \) can be described as functions of \( m \) as follows:

\[
T_{yB_1}(n) = T_{yB,1}(s(m)) = d^2 s(m) = 2d^3 m
\]
\[
T_{yB_2}(n) = T_{yB,2}(s(m)) = d^2 s(m) = 2d^3 m
\]
\[
T_{xB}(n) = T_{xB}(s(m)) = d^3 s(m) = 2d^3 m.
\]

Thus, the total complexity for Alice and Bob can be compared:

\[
T_{yA}(m) + T_{xB}(m) = 2d^3 m < 4d^3 m + 2d^4 m = T_{yB,1}(s(m)) + T_{yB,2}(s(m)) + T_{xB}(s(m)). \quad (39)
\]

When \( d \) is given, \( T_{yA}(m) + T_{xB}(m) \) and \( T_{yB,1}(s(m)) + T_{yB,2}(s(m)) + T_{xB}(s(m)) \) have the same order of complexity. \( 2d^4 \) and \( 2d^3 \) are considered as coefficients.

Of course, we can consider the total complexity for Alice and Bob by the functions \( T'_{A}, T'_B : \mathbb{N} \rightarrow \mathbb{R}^+ \) of \( d \), which are defined as

\[
T'_{A}(d) := 2md^3
\]
\[
T'_{B}(d) := 4md^3 + 2md^4
\]
for a given \( m \). In this case, \( T'_{A}(d) \in O(d^3), T'_{B}(d) \in O(d^4) \), and \( O(d^4) \subset O(d^3) \) are obvious. Which values of \( m \) and \( d \) should be given and which should be the input of the functions can be determined the environment in which the algorithm is used, such as the computational resources and communication infrastructure.

**Remark 2.** As mentioned in Remark 1, Alice can select her secret key \( x_A \) not only as a matrix, but also as a vector, whereas \( x_B \) and \( N_B \) are selected as matrices. Let \( \mathcal{V} \) be a vector space of \( \mathbb{V} := \mathbb{Z}_p^d \), where \( |y| \leq d \) (\( y \in \mathcal{V} \)) and a subset \( \mathcal{V} \) of \( \mathcal{V} := \{ y \in \mathcal{V} : |y| \leq dm \} \subset \mathcal{V} \), and suppose that \( x_A \) is selected from \( \mathcal{V} \). In this case, the breaking strategy against this algorithm for Eve is as follows:

- solving \( d^2 \) DLPs for \( N_B \) and \( d \) DLPs for \( x_A \), and
- an exhaustive attack for \( x_A \) within \( \mathcal{V} \), the size of which is \( 2^{sm} \).

That is, the breaking complexity is equivalent to the case when \( x_A \in \mathcal{S} \) in terms of the order of complexity. Thus, there is also no problem in selecting \( n \) and \( m \) from the relation of Theorem 2 in this case. The computational complexities for \( y_A, k_A, \) and \( k_B \), which are denoted by the functions \( T_{yA,vec}, T_{kA,vec}, \) and \( T_{kB,vec} : \mathbb{N} \rightarrow \mathbb{R}^+ \), are

\[
T_{yA,vec}(m) := d^2 m
\]
\[
T_{kA,vec}(m) := d^2 m
\]
\[
T_{kB,vec}(n) := d^2 n
\]

for a given \( d \). It can be observed that the complexities are reduced from (34), (35), and (38). Moreover, the calculation of (33) is independent for each element \( a \in \{1, \ldots, d\} \), and thus, it can be computed in parallel. The calculation is also parallelized for each \( l \in \{1, \ldots, d\} \). In Section 6, we report the results of the implementational experiments of SEDH in the case of \( x_A \in \mathcal{V} \). To determine how rapidly the calculation is performed, particularly for Alice, we compare the calculation speed with that of the usual DH algorithm.

5. Generalization

In this section, all investigations are conducted under the following assumptions:

- the numbers \( n, m \in \mathbb{N} \) and a not linear and monic polynomial \( s \), where \( n = s(m) > m \) for all \( m \in \mathbb{N} \).

\[
\text{Appl. Sci. 2021, 11, 5540}
\]
Table 4. Public keys and SSK of Alice and Bob for SAPKA\(^k\) class.

| Key  | Parameter |
|------|-----------|
| Bob  | Public keys |
|      | \(y_{B,1}(y) = N_1^{-1} \circ x_4(y) = f_g \circ N_1^{-1} \circ x_4(y) \quad (y \in S)\), |
|      | \(y_{B,2}(y) = x_1 \circ x_2(y) = f_g \circ x_1 \circ x_2(y) \quad (y \in S)\) |
|      | SSK |
|      | \(\kappa_B = x_3 \circ N_1(y_A) = f_g \circ x_3 \circ N_1^{-1} \circ x_4(x_A)\) \quad (44) |
| Alice | Public key |
|      | \(y_A = N_1^{-1} \circ x_4(x_A) = f_g \circ N_1^{-1} \circ x_4(x_A)\) |
|      | SSK |
|      | \(\kappa_A = x_1 \circ x_2(x_A) = f_g \circ x_1 \circ x_2(x_A)\) |
Moreover, (41) and (42) ensure that the maps \( x'_1 \circ x'_2 \) and \( N^{-1}_1 \circ x'_4 \) and thus the element \( x_A \), cannot be disclosed by Eve within polynomial time. If \( x'_1 \circ x'_2(e) \in \mathcal{S} \) and \( N^{-1}_1 \circ x'_4(e) \in \mathcal{S} \), the complexities for calculating

\[
f^{-1}_g \circ f \circ x'_1 \circ x'_2(e) = x'_4 \circ x'_2(e)
\]

\[
f^{-1}_g \circ f \circ N^{-1}_1 \circ x'_4(e) = N^{-1}_1 \circ x'_4(e)
\]

are reduced compared to the case when (43) holds (see Remark 3 in Section 5.1). If \( x'_1, x'_2, x'_3, x'_4, N'_1 \) are linear maps, for example, Eve can obtain a linear map \( x'_3, E \circ N^{-1}_1 : \mathcal{S} \to \mathcal{S} \) to satisfy:

\[
x'_3, E \circ N^{-1}_1 \circ x'_4(e) = x'_1 \circ x'_2(e)
\]

by matrix multiplication, because \( x'_1 \circ x'_2(e) \) is described as a matrix. Moreover, as \( \mathcal{S} \) is a multiplicative semi-group, \( x'_1 \circ x'_2(y) \) for any \( y \in \mathcal{S} \) can be expressed as \( x'_1 \circ x'_2(y) \) and either of the equations

\[
x'_1 \circ x'_2(yc) = yx'_1 \circ x'_2(e) = yx'_3, E \circ N^{-1}_1 \circ x'_4(e) = x'_3 \circ N^{-1}_1 \circ x'_4(yc) = x'_1 \circ x'_2(yc)
\]

or

\[
x'_1 \circ x'_2(yc) = x'_1 \circ x'_2(y) = x'_3, E \circ N^{-1}_1 \circ x'_4(y) = x'_3 \circ N^{-1}_1 \circ x'_4(yc) = x'_1 \circ x'_2(yc)
\]

holds for all \( y \in \mathcal{S} \). Thus, only if Eve knows the calculation rule of map \( x_3 \circ N_1 \) can she obtain a map \( x'_3, E \circ N^{-1}_1 \) to satisfy

\[
x'_3, E \circ N^{-1}_1 \circ x'_4(yc) = x'_3 \circ N^{-1}_1 \circ x'_4(yc) = x'_3 \circ N^{-1}_1 \circ x'_4(yc) = x'_3 \circ x'_2(yc) = \kappa_A.
\]

Of course, if the maps \( x'_1, x'_2, x'_3, x'_4, N'_1 \) are not linear, this attack may be impossible, but if (43) holds and \( n \) is sufficiently large to make the calculation of \( f^{-1}_g \circ f(y) \) \( (y \notin \mathcal{S}) \) difficult in real time, Eve cannot proceed to (46). Thus, Requirement 2 of Section 2.1 is achieved for all algorithms in the SAPKA/\( f_\mathcal{S} \) class.

Algorithms of SAPKA/\( f_\mathcal{S} \) Class

It can be proven that \( \mathcal{C}_{\text{NDH}} \) in Section 3.2 and \( \mathcal{C}_{\text{SE}} \) in Section 3.3 belong to the SAPKA/\( f_\mathcal{S} \) class. For \( \mathcal{C}_{\text{NDH}} \), if \( \mathcal{S} := \mathbb{Z}_p \) and Bob defines \( x'_1, x'_2, x'_3, x'_4, N'_1, f_\mathcal{S} \) as

- \( x'_1(y) := x_B y \)
- \( x'_2 := \text{id} \)
- \( x'_3(y) := x_B y \)
- \( x'_4(y) := \text{id} \)
- \( N'_1 := n_B^{-1} y \)
- \( f_\mathcal{S}(y) := g^{th} \)

then (40) holds, and (41) and (42) are satisfied according to the descriptions of the public keys of Bob in Table 5.

Table 5. Public keys of Bob for \( \mathcal{C}_{\text{NDH}} \)

| Public keys of Bob | \( y_{B,1}(y) := N^{-1}_1 \circ x_4(y) = g^{n_B y} = f_\mathcal{S} \circ N^{-1}_1 \circ x'_4(y) \) \( (y \in \mathcal{S}) \), \( y_{B,2}(y) := x_1 \circ x_2(y) = g^{x_2 y} = f_\mathcal{S} \circ x'_1 \circ x'_2(y) \) \( (y \in \mathcal{S}) \) |
|---|---|

For \( \mathcal{C}_{\text{SE}} \), if \( \mathcal{S} := \mathcal{M}(d, \mathbb{Z}_p) \) and Bob defines \( x'_1, x'_2, x'_3, x'_4, N'_1, f_\mathcal{S} \) as

- \( y_{B,1}(y) := N^{-1}_1 \circ x_4(y) = g^{n_B y} = f_\mathcal{S} \circ N^{-1}_1 \circ x'_4(y) \) \( (y \in \mathcal{S}) \), \( y_{B,2}(y) := x_1 \circ x_2(y) = g^{x_2 y} = f_\mathcal{S} \circ x'_1 \circ x'_2(y) \) \( (y \in \mathcal{S}) \)
\[ x_1'(y) := x_B y \]
\[ x_2' := id \]
\[ x_3'(y) := x_B y \]
\[ x_4'(y) := id \]
\[ N_1' := N_B^{-1} y \]
\[ f_g(y) := g^{x_4} \]

(40) holds, and (41) and (42) are satisfied according to Table 6.

Table 6. Public keys of Bob for \( C_{SE} \).

| Public keys of Bob | for \( C_{SE} \) |
|-------------------|------------------|
| \( y_{B,1}(y) = N_1^{-1} \circ x_4(y) = g^{N_B y} = f_g \circ N_1^{-1} \circ x_4(y) (y \in S) \), | |
| \( y_{B,2}(y) = x_1 \circ x_2(y) = g^{x_4(y)} = f_g \circ x_4 \circ x_2(y) (y \in S) \) | |

At this point, we construct a \( SAPKA_{f,g} \) subclass, the algorithms of which possess the same property as the SEDH that was investigated in Sections 4.3 and 4.4, namely biased computational complexity to Bob (see Section 5.1). We mainly focus on this class in this study.

5.1. Notations and Definitions

Prior to constructing the main subclass (Definition 9), we present several definitions and notations below. Definition 7 provides one \( SAPKA_{f,g} \) subclass in which the algorithms are symmetric and include DH. In Section 5.2, we investigate how this subclass relates to the main subclass of Definition 9. Definition 8 provides the name for the input \( y \) of a given map \( f_g' : S \rightarrow S \) according to the set of \( y \). We use this definition to emphasize the difference in complexity for calculating \( f_g' \circ f_g(y) \) of a given map \( f_g' \) when \( y \in S \) and when \( y \notin S \). The notations directly below Definition 8 are functions that demonstrate the complexity for calculating each key. As these notations and Definition 8 allow us to express the breaking and computational complexity for each key quantitatively, the main subclass can be defined by equalities and inequalities.

Definition 7. For the quintuple \( C \) such that \( C \in SAPKA_{f,g} \), if \( N_1' = x_4' = id \), we state that \( C \) is a member of the symmetric \( SAPKA_{f,g} \) class, and we express this relation as

\[ C \in SAPKA_{f,g,\text{symmetry}}. \]

If \( C \in SAPKA_{f,g,\text{symmetry}} \), one of the public keys of Bob \( y_{B,1} \) is described as

\[ y_{B,1}(y) = N_1^{-1} \circ x_4(y) = f_g \circ N_1^{-1} \circ x_4(y) = f_g(y), \]

where \( y \in S \). This means that Bob sending \( y_{B,1} \) is equivalent to simply informing Alice of the construction rule of \( y_A \). Alice can calculate \( y_A \) without any secret/public information from Bob. Symmetry means that the number of public keys that both Alice and Bob construct is essentially one, and they can construct their public keys without any public/secret information from the other.

For \( C_{NDH} \) in Section 3.2, if the parameter \( n_B \) is equal to 1, \( C_{NDH} \) describes the usual DH. We denote this case of \( C_{NDH} \) as \( C_{DH} \), and we obtain

\[ C_{DH} \in SAPKA_{f,g,\text{symmetry}}. \]

Moreover, \( C_{SE} \) in the case of \( N_B = E \), denoted by \( C_{SE,(N_B=E)} \), is

\[ C_{SE,(N_B=E)} \in SAPKA_{f,g,\text{symmetry}}. \]
Definition 8. For a given non-easily invertible map $f'_S : S \rightarrow S$, if the set to which the input $y$ belongs is $S$, we refer to the input $y$ as a kn-bit $f'_S$ element. If the set to which the input $y$ belongs is $S$, we refer to the input $y$ as a kn-bit $f'_S$ element.

The complexity of obtaining the kn-bit $f'_S$ element means the complexity of calculating $f'_S \circ f_S(y)$ when $y \in S$ (for the $km$-bit $f'_S$ element, $y \in S$).

We define the following functions of bit size that demonstrate the complexity of calculating each public key and SSK as well as the complexity for Eve to obtain a certain key:

- $T_{y_{B,1}} : N \rightarrow \mathbb{R}^+ :$ the complexity required for Bob to construct $y_{B,1}$;
- $T_{y_{B,2}} : N \rightarrow \mathbb{R}^+ :$ the complexity required for Bob to construct $y_{B,2}$;
- $T_{y_A} : N \rightarrow \mathbb{R}^+ :$ the complexity required for Alice to calculate $y_A$;
- $T_{k_A} : N \rightarrow \mathbb{R}^+ :$ the complexity required for Alice to calculate her SSK $k_A$;
- $T_{y_B} : N \rightarrow \mathbb{R}^+ :$ the complexity required for Bob to calculate their SSK $k_B$;
- $T_{Eve \cdot x_A} : N \rightarrow \mathbb{R}^+ :$ the least complexity required for Eve to obtain Alice’s secret key $x_A$ from public informations, where the input of $T_{Eve \cdot x_A}$ is $m$; and
- $T_{f'_S}$ : $N \rightarrow \mathbb{R}^+ :$ the complexity required to compute $f'_S \circ f_S(y)$ for map $f'_S$, where the input of $T_{f'_S}$ is $m$ if $y \in S$ and $n$ if $y \in \mathbb{S}$.

Furthermore, for $T_{Eve \cdot x_A}$, $T_{f'_S}$, we define the following functions to describe their order:

- $h_{Eve \cdot x_A} : N \rightarrow \mathbb{R}^+ :$ to obtain the relation $T_{Eve \cdot x_A} \in O(h_{Eve \cdot x_A})$, and
- $h_{f'_S} : N \rightarrow \mathbb{R}^+ :$ to obtain the relation $T_{f'_S} \in O(h_{f'_S})$.

Example 1.

- If $T_{y_{B,1}}, T_{y_{B,2}},$ and $T_{k_B}$ are defined as functions of $n$, $T_{y_A}$ and $T_{k_A}$ are defined as functions of $m$, and the polynomial $s$ is concretely determined, we can compare the total computational complexity for Alice and Bob as per Section 4.5.
- The least complexity for Eve to obtain $x_A$ in Section 3.3 is determined as

$$T_{Eve \cdot x_A}(m) \geq d^2 2^\frac{s(n)}{m}.$$  

- The complexity of obtaining the $d^2n$-bit $f'_S$ element, where $f'_S$ is the non-easily invertible map of $C_{SE}$ as defined in Table 6, is expressed as

$$T_{f'_S}(n) = T_{f'_S}(s(m)) = d^2 2^{\frac{s(m)}{m}}.$$  

- The complexity of obtaining the $d^2m$-bit $f'_S$ element is determined as

$$T_{f'_S}(m) = d^2 2^\frac{s}{m}.$$  

In general, the complexity of obtaining the kn-bit $f'_S$ element for a given non-easily invertible map $f'_S : S \rightarrow S$ is described by $T_{f'_S}(n) = T_{f'_S}(s(m))$, and that of the km-bit $f'_S$ element is $T_{f'_S}(m)$. We can compare not only the complexity but also the order of complexity of the two using the map $h_{f'_S}$ defined above, as per the following Remark 3.

Remark 3. When the order of complexity for computing the kn-bit $f'_S$ element for map $f'_S$ is equal to $O(2^{am})$, where $a \in \mathbb{R}^+ \setminus \{0\}$:

$$T_{f'_S}(s(m)) \in O(h_{f'_S}(s(m))) = O(2^{as(m)}) \subseteq O(2^{am}) = O(h_{f'_S}(m)) \equiv T_{f'_S}(m)$$ (47)
holds, and the equality holds if and only if $s$ is a linear monic polynomial for some $t \in \mathbb{N}$:

$$n = s(m) = m + 1.$$  

This can be proven in a similar manner to Proposition 1, and in this case, $s$ is restricted as a non-linear and monic polynomial, so the equality never holds.

Furthermore, needless to say, $T$ in Theorem 3.

When the order of complexity for computing the km-bit $f^{-1}_g$ element is greater than $O(2^m)$, although the sufficient and necessary condition of $s$ for the equality of (47) may not be the same,

$$O(h_{f^{-1}_g}(s(m))) = O(2^{as(m)}) \subset O(2^{am}) = O(h_{f^{-1}_g}(m))$$

holds because, for all $m$, the ratio $\frac{h_{f^{-1}_g}(s(m))}{h_{f^{-1}_g}(m)}$ is much larger and cannot be a constant in this case. This means that when $s$ is a not linear and monic polynomial, the problem of obtaining the kn-bit $f^{-1}_g$ element by computing $f^{-1}_g \circ f_g(y)$ ($y \in S$) and the problem of obtaining the km-bit $f^{-1}_g$ element by computing $f^{-1}_g \circ f_g(y)$ ($y \in S$) belong to different orders of complexity classes. Furthermore, needless to say, $T_{f^{-1}_g}(s(m)) > T_{f^{-1}_g}(m)$ holds.

**Definition 9.** For the quintuple $C$ of $C \in \text{SAPKA}_{f_g}$, if the following relations hold:

$$T_{Eve \, x_A}(m) \geq T_{f^{-1}_g}(s(m))$$

$$T_{y_{B,1}}(s(m)) + T_{y_{B,2}}(s(m)) + T_{y_B}(m) > T_{x_A}(m) + T_{x_B}(m),$$

we state that $C$ is a member of the class that is computationally biased to Bob $\text{SAPKA}_{f_g}$, and we write this relation as

$$C \in \text{SAPKA}_{f_g} A < B.$$  

If the relation

$$T_{Eve \, x_A}(m) \leq T_{f^{-1}_g}(m)$$

holds, $C$ is a member of the computationally unbiased $\text{SAPKA}_{f_g}$ class, and we write this relation as

$$C \in \text{SAPKA}_{f_g} A = B.$$  

Relation (49) indicates that the breaking complexity of an algorithm that is constructed from $C \in \text{SAPKA}_{f_g} A < B$ is equal to or greater than the complexity that is required to obtain the km-bit $f^{-1}_g$ element, even if $x_A$ is selected from $\mathcal{S}$. Furthermore, (51) indicates that when $C \in \text{SAPKA}_{f_g} A = B$, the breaking complexity is equal to or less than that of obtaining the km-bit $f^{-1}_g$ element. As noted in Remark 3, not only does $T_{f^{-1}_g}(s(m)) > T_{f^{-1}_g}(m)$ hold, but the orders of complexity are also different, so there cannot exist a $C'$ such that $C' \in \text{SAPKA}_{f_g} A < B \cap \text{SAPKA}_{f_g} A = B$. Thus, we obtain Lemma 1 in Section 5.2.

### 5.2. Inclusion Relations between Subclasses of SAPKA and Necessity Condition

**Lemma 1.** For $\text{SAPKA}_{f_g} A < B \subset \text{SAPKA}_{f_g}$ and $\text{SAPKA}_{f_g} A = B \subset \text{SAPKA}_{f_g}$, the following relation holds:

$$\text{SAPKA}_{f_g} A < B \cap \text{SAPKA}_{f_g} A = B = \emptyset.$$  

According to the attack introduced in Section 4.3, we obtain the following results.

**Theorem 3.** For $C_{SE}$ in Section 3.3, if Attack 2 of Section 4.3 and the exhaustive attack for $x_A \in \mathcal{S}$ are only attacks without calculating $\log_B S^N_B = N_B$ or there exist other attacks but the order of complexity is equal to or greater than $O(2^{dm})$,

$$C_{SE} \in \text{SAPKA}_{f_g} A < B.$$
Theorem 5 (Necessary Condition 2). If the quintuple $C := (x_1, x_2, x_3, x_4, N_1) \in \text{SAPKA}_{f_g:A,B}$, then there does not exist a map $f_g$ such that to satisfy

$$T_{\text{Eve}} x_A(m) = f_g^{-1} (s(m)).$$

Under this assumption, $n = s(m) = 2dm$ holds, and thus, we obtain

$$T_{y_{B,1}}(s(m)) + T_{y_{B,2}}(s(m)) + T_{x_B}(s(m)) = 4md^3 + 2md^4 > T_{y_A}(m) + T_{x_A}(m) = 2md^3$$

from Section 4.5. □

We introduce two necessary conditions for the $\text{SAPKA}_{f_g:A,B}$ class.

Theorem 4 (Necessary Condition 1). If the quintuple $C := (x_1, x_2, x_3, x_4, N_1) \in \text{SAPKA}_{f_g:A,B}$, the relation

$$C \notin \text{SAPKA}_{f_g:\text{symmetry}}$$

holds.

Proof. We prove the contrapositive. If $C \in \text{SAPKA}_{f_g:\text{symmetry}}$, Alice’s public key $y_A$ is described as:

$$y_A = N_1^{-1} \circ x_A = f_g \circ N_1^{-1} \circ x_A'$$

according to (40), but $N_1'$ and $x_A'$ are identities; thus, $y_A$ becomes

$$y_A = f_g(x_A).$$

Because $x_A \in S$, $x_A$ is a $km$-bit $f_g^{-1}$ element for map $f_g$, and we obtain

$$T_{\text{Eve}} x_A(m) = f_g^{-1}(m).$$

Thus, $C \in \text{SAPKA}_{f_g:A,B}$. Moreover, Lemma 1 indicates that $C \notin \text{SAPKA}_{f_g:A,B}$. □

Corollary 1. The quintuple $C_{\text{DH}}$ of the usual DH is

$$C_{\text{DH}} \in \text{SAPKA}_{f_g:A=B},$$

and $C_{\text{SE}(N_B=E)}$ defined in Section 5.1 is

$$C_{\text{SE}(N_B=E)} \in \text{SAPKA}_{f_g:A=B}.$$

Theorem 5 (Necessary Condition 2). If the quintuple $C := (x_1, x_2, x_3, x_4, N_1) \in \text{SAPKA}_{f_g:A,B}$, then there does not exist a map $f_g' : S \rightarrow S$ such that to satisfy

$$f_g' \circ N_1' \circ x_A'(y) = f_g'(y) \wedge T_{f_g'^{-1}}(m) \leq T_{f_g^{-1}}(m)$$

for all $y \in S, m \in \mathbb{N}$.

Proof. We prove the contrapositive. We assume that for $C \in \text{SAPKA}_{f_g:A,B}$, there exists a map $f_g' : S \rightarrow S$ to satisfy

$$f_g' \circ N_1' \circ x_A'(y) = f_g'(y) \quad (54)$$

$$T_{f_g'^{-1}} \leq T_{f_g^{-1}}$$

for all $y \in S$. According to (54), $y_A$ can be described as

$$y_A = f_g \circ N_1^{-1} \circ x_A'(x_A) = f_g'(x_A).$$
This indicates that $x_A$ is a $km$-bit $f_g^{-1}$ element for a map $f'_g$. In combination with (55), the relation

$$T_{Eve} x_A(m) = T_{f_g^{-1}}(m) \leq T_{f_g^{-1}}(m)$$

holds. Thus, $C \in SAPKA_{f_g,A=B}$; that is, $C \notin SAPKA_{f_g,A<B}$ is proven. □

**Corollary 2.**

$$C_{NDH} \in SAPKA_{f_g,A=B}$$

**Proof.** The attack of Section 4.2 indicates that

$$y_A = f_g \circ N_1^{-1} \circ x'_A(x_A) = g^{nB}x_A = (g^{nB})x_A = f'_g(x_A),$$

where $f'_g(y) := f_g(N_1^{-1} \circ x'_A(1))(y) = f_g yB(y)$, and (53) is satisfied. □

**Remark 4.** Theorem 1 of Section 4.3 implies that there does not exist a map $f'_g$ to satisfy $f'_g(y) = f_g \circ N_1^{-1} \circ x'_A(y)$ with $T_{f_g^{-1}} \leq T_{f_g^{-1}}$ as for algorithm of CSE.

According to the proof of Theorem 4 and Corollary 2, we obtain the following inclusion relation:

**Corollary 3.**

$$SAPKA_{f_g,\text{symmetry}} \subset SAPKA_{f_g,A=B}.$$ 

**Proof.** From the proof of Theorem 4,

$$C \in SAPKA_{f_g,\text{symmetry}} \Rightarrow C \in SAPKA_{f_g,A=B}$$

is obtained. Thus, we obtain

$$SAPKA_{f_g,\text{symmetry}} \subset SAPKA_{f_g,A=B}.$$ 

Moreover, $C_{NDH}$ is related to $C_{NDH} \notin SAPKA_{f_g,\text{symmetry}}$, but $C_{NDH} \in SAPKA_{f_g,A=B}$ according to Corollary 2. Thus, $SAPKA_{f_g,\text{symmetry}} \subset SAPKA_{f_g,A=B}$ is obtained. □

Figure 3 presents the inclusion relations between the subclasses of SAPKA.

![Figure 3. Inclusion relations between subclasses of SAPKA.](image)
Remark 5. The proposition

\[
\text{SAPKA}_{f,g,A < B} \cup \text{SAPKA}_{f,g,A = B} = \text{SAPKA}_{f,g}
\]

as well as the necessary conditions (52) and (53) as sufficient conditions for the \(\text{SAPKA}_{f,g,A < B}\) class have not yet been confirmed. These will be investigated in our subsequent paper.

6. Implementational Experiments

In this section, we report the simple experimental results. All experiments were performed in the following environment:

- OS: Windows 10 Pro
- Processor: 2.90 GHz Intel Core i7-10700
- RAM: 8 GB
- Language: JAVA

Tables 7–9 indicate the computational complexity for Alice \((y_A, \kappa_A)\) and Bob \((y_B, y_B^1, \kappa_B)\) according to \(d\) and \(m\) while \(n\) was fixed as 1024, 3072, and 5120 bits. As can be observed from Tables 7–9, the calculation time for Bob increased as \(d\) increased. This is because the calculation complexities of (32) and (33) were dependent only on \(d\), whereas \(n\) was fixed. For Alice’s computation, there were cases in which the calculation time decreased as \(d\) increased (Tables 8 and 9). This can be explained by (32) and (33) for each \(a \in \{1, \ldots, d\}\), being independent, and an increase in \(d\) resulted in a decrease in \(m\). Therefore, these values can be computed in parallel with a smaller \(m\) when \(d\) increases. When \(d > 4\), the calculation time will increase despite the decrease in the size of \(m\) owing to a shortage in processors for efficient parallel computing.

Moreover, the difference in the order of complexity for Alice and Bob can be observed from Figure 4, where \(m\) was fixed as 128 bits and \(d\) increased from 2 to 16.

Table 7. Calculation time (ms) for Alice and Bob when \(n = 1024\).

| \(d\) | \(m\) | Bob (ms) | Alice (ms) |
|------|------|---------|------------|
| 2    | 256  | 7.69307551 | 1.89631633 |
| 4    | 128  | 16.7975061 | 3.39046122 |
| 6    | 85   | 25.310163  | 5.37513889 |

Table 8. Calculation times (ms) for Alice and Bob when \(n = 3072\).

| \(d\) | \(m\) | Bob (ms) | Alice (ms) |
|------|------|---------|------------|
| 2    | 768  | 131.792466 | 17.612742 |
| 4    | 384  | 272.971624 | 16.480546 |
| 6    | 256  | 472.95086  | 28.988962 |

Table 9. Calculation times (ms) for Alice and Bob when \(n = 5120\).

| \(d\) | \(m\) | Bob (ms) | Alice (ms) |
|------|------|---------|------------|
| 2    | 1280 | 528.017186 | 65.886198 |
| 4    | 640  | 1164.85195 | 57.475974 |
| 6    | 426  | 1942.06485 | 108.263714 |
Figure 4. Comparison of calculation times (ms) for Alice and Bob according to $d$ ($m = 128$).

We verified how rapidly the calculation of Alice could be achieved by comparing the speed with that of the usual DH. Table 10 and Figure 5 compare the calculation times of SEDH and DH for $y_A$ and $\kappa_A$ for the same security level ($n$) from 1024 to 7168 bits. The calculation speeds in the following table and figure are the averages of 50 calculations.

Table 10. Comparison of calculation times (ms) for Alice with DH.

| Security ($n$) | ($m$) | ($d$) | DH (ms) | SEDH (ms) |
|---------------|-------|-------|---------|-----------|
| 1024          | 256   | 2     | 2.727202| 1.896316327|
| 2048          | 512   | 2     | 15.514816| 6.451664815|
| 3072          | 384   | 4     | 51.776748| 16.480546 |
| 4096          | 1024  | 2     | 111.332034| 32.10356 |
| 5120          | 640   | 4     | 202.806142| 57.459794 |
| 6144          | 768   | 4     | 335.208752| 94.12185 |
| 7168          | 896   | 4     | 535.273036| 144.93926 |

Figure 5. Comparison of calculation times (ms) for Alice with DH.

7. Conclusions

In this study, we have estimated the breaking complexity and computational complexity of two SAPKA class algorithms. One of these potentially possesses the property
whereby the security is less dependent on the secret key space of Alice than that of Bob. This property allows Alice to calculate her SSK with less complexity than Bob in the algorithm. Moreover, we generalized these algorithms and constructed SAPKA subclasses according to the security/efficiency properties. We constructed several subclass algorithms in which the above property holds. We expect that algorithms in this class will aid in constructing secure communication infrastructures, even with a small capability of devices for one side (Alice’s computational asymmetry). The necessary conditions for this class and inclusion relations with other SAPKA subclasses were also investigated.

In the following Future Work section, we provide several unclear points, which include new problems obtained from this study (indicated by •) and problems not discussed in this paper but that must be investigated for practical use of presented algorithms and frameworks (indicated by o).

**Future Works**

- Is it true that $SAPKA_{fg,A<B} \cup SAPKA_{fg,A=B} = SAPKA_{fg}$? (Remark 5)
- Is it true that if the relations (52) and (53) hold for $C \in SAPKA_{fg}$, C is a member of the $SAPKA_{fg,A<B}$ class? That is, are (52) and (53) also sufficient conditions for the $SAPKA_{fg,A<B}$ class? As mentioned in Section 2.2, this paper does not provide how to construct secure PKA algorithms resilient to any types of theoretical attacks. However, if sufficient conditions for the $SAPKA_{fg,A<B}$ class are found, algorithms that cannot be broken at least within polynomial time can be easily constructed because those are protected thanks to non-easily mappable $f_g$ (page 23). Of course, conditions might differ depending on required security level, but considering the sufficient conditions surely helps us provide how to construct secure PKA algorithms.

- Does S need to be selected as a multiplicative semi-group/group? For example, the algorithms in Section 3.3 are effective even in vector space, which forms an additive group (Remark 1). We should investigate how we can construct PKA algorithms on algebraic structures other than multiplicative semi-groups and then compare the breaking and computational complexities with the original one.
  - We should attempt to describe algorithms such as LWE-based or NTRU-type algorithms with the SAPKA parameters and verify whether these algorithms belong to the $SAPKA_{fg,A<B}$ class. Otherwise, would it be possible to modify these to belong to the $SAPKA_{fg,A<B}$ class?
  - It is well known that key agreement algorithms are vulnerable against man-in-the-middle (MITM) attack. Obviously, also algorithms in SAPKA are also not exceptional without any measures. Moreover, the number of public keys especially for Bob is larger than symmetric type algorithms, so they way to manage public keys for SAPKA algorithms must be discussed more. Adaptability with techniques such as digital signature [21,22] and public key certifications [23] must be well discussed.
  - As mentioned in [1], the SSKs that Alice and Bob calculate should be ephemeral, and each key must have enough randomness. To practically use algorithms introduced in this paper, if public keys and SSK are sufficiently random to protect Eve from deducing SSK must be investigated.

**Author Contributions:** Conceptualization, K.J., S.I. and M.R.; Formal analysis, K.J. and S.I.; Implementation, K.J.; Writing—original draft, K.J.; Writing—review and editing, K.J., S.I. and M.R. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

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

The following abbreviations are used in this manuscript:

- PKA: public key agreement
- SAPKA: strongly asymmetric public key agreement
- SSK: secret shared key
- DH: Diffie-Hellman
- NDH: Noised Diffie-Hellman
- SEDH: Schur Exponentiation-Based Diffie-Hellman
- DLP: discrete logarithm problem
- MITM: man-in-the-middle

References

1. Shannon, C.E. Communication Theory of Secrecy Systems. *Bell Syst. Tech. J.* 1949, 28, 656–715. [CrossRef]
2. Diffie, W.; Hellman, M. New directions in cryptography. *IEEE Trans. Inf. Theory* 1976, 22, 644–654. [CrossRef]
3. Rivest, R.L.; Shamir, A.; Adleman, L. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM* 1978, 21, 120–126. [CrossRef]
4. Adrian, D.; Bhargavan, K.; Durumeric, Z.; Gaudry, P.; Green, M.; Halderman, J.A.; Heninger, N.; Springall, D.; Thomé, E.; Valenta, L.; et al. Imperfect Forward Secrecy: How Diffie-Hellman Fails in Practice. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, 12–16 October 2015; pp. 5–17.
5. Shor, P.W. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM J. Comput.* 1997, 26, 1484–1509. [CrossRef]
6. Bernstein, D.J.; Chen, D.; Lange, T.; van Vredendaal, C. NTRU Prime: Reducing Attack Surface at Low Cost. In *Selected Areas in Cryptography—SAC 2017*; Lecture Notes in Computer Science; Adams, C., Camenisch, J., Eds.; Springer: Cham, Switzerland, 2017; Volume 10719. [CrossRef]
7. Hülsing, A.; Rijneveld, J.; Schanck, J.; Schwabe, P. High-Speed Key Encapsulation from NTRU. In *Cryptographic Hardware and Embedded Systems—CHES 2017*; Lecture Notes in Computer Science; Fischer, W., Homma, N., Eds.; Springer: Cham, Switzerland, 2017; Volume 10529. [CrossRef]
8. Hoffstein, J.; Pipher, J.; Silverman, J.H. NTRU: A ring-based public key cryptosystem. In *Algorithmic Number Theory—ANTS 1998*; Lecture Notes in Computer Science; Buhler, J.P., Ed.; Springer: Berlin/Heidelberg, Germany, 1998; Volume 1423. [CrossRef]
9. Bos, J.; Ducas, L.; Kiltz, E.; Lepoint, T.; Lyubashevsky, V.; Schanck, J.M.; Schwabe, P.; Seiler, G.; Stehle, D. CRYSTALS—Kyber: A CCA-Secure module-lattice-based KEM. In Proceedings of the 2018 IEEE European Symposium on Security and Privacy (EuroS and P), London, UK, 24–26 April 2018; pp. 353–367. [CrossRef]
10. Alkim, E.; Ducas, L.; Poppelmann, T.; Schwabe, P. Post-quantum key exchange—A new hope. In Proceedings of the 25th USENIX Security Symposium (USENIX Security 16), Austin, TX, USA, 10–12 August 2016; pp. 327–343.
11. Post-Quantum Cryptography Competition Round 2 Submissions. Available online: https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions (accessed on 16 March 2021).
12. PQC Standardization Process: Third Round Candidate Announcement. Available online: https://csrc.nist.gov/News/2020/pqc-third-round-candidate-announcement (accessed on 16 March 2021).
13. Micciancio, D.; Regev, O. Worst-case to average-case reductions based on Gaussian measures. *J. Comput.* 2007, 37, 267–302. [CrossRef]
14. Regev, O. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM* 2009, 56, 1–40. [CrossRef]
15. Laine, K.; Lauter, K. Key Recovery for LWE in Polynomial Time. IACR Cryptology ePrint Archive 2015, no. 176. Available online: https://eprint.iacr.org/2015/176.pdf (accessed on 14 June 2021).
16. Peikert, C. Public-key cryptosystems from the worst-case shortest vector problem. In Proceedings of the forty-first annual ACM symposium on Theory of computing, Bethesda, MD, USA, 31 May–2 June 2009; pp. 333–342.
17. Coppersmith, D.; Shamir, A. Lattice Attacks on NTRU. In *Advances in Cryptology—EUROCRYPT ’97*; Lecture Notes in Computer Science; Fumy, W., Ed.; Springer: Berlin, Germany, 1997; Volume 1233. [CrossRef]
18. Accardi, L.; Iriyama, S.; Regoli, M.; Ohyama, M. Strongly Asymmetric Public Key Agreement Algorithms; Technical Report ISEC2011-20; IEICE: Tokyo, Japan, 2011; pp. 115–121.
19. Pollard, J. Monte Carlo Methods for Index Computation (modp). *Math. Comput.* 1978, 32, 918–924.
20. Pohlig, S.; Hellman, M. An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance (Corresp.). *IEEE Trans. Inf. Theory* 1978, 24, 106–110. [CrossRef]
21. Lamport, L. *Constructing Digital Signatures from a One-Way Function*; Technical Report CSL-98; SRI International: Menlo Park, CA, USA, 1979; Volume 238.
22. Merkle, R.C. *A Certified Digital Signature, Conference on the Theory and Application of Cryptology*, Springer: New York, NY, USA, 1989.

23. Cooper, D.; Santesson, S.; Farrell, S.; Boeyen, S.; Housley, R.; Polk, W. *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*, RFC 5280. 2008; pp. 1–151. Available online: https://datatracker.ietf.org/doc/html/rfc5280 (accessed on 14 June 2021).