How to Simultaneously Exchange Secrets by General Assumptions

Tatsuaki Okamoto* Kazuo Ohta

NTT Laboratories
1-2356, Take, Yokosuka-shi, Kanagawa-ken, 238-03 Japan

Abstract

The simultaneous secret exchange protocol is the key tool for contract signing protocols and certified mail protocols. This paper proposes efficient simultaneous secret exchange protocols (or gradual secret releasing protocols) that are based on general assumptions such as the existence of one-way permutations and one-way functions, while the existing efficient simultaneous secret exchange protocols are based on more constrained assumptions such as specific number theoretic problems and the existence of oblivious transfer primitives (or trap-door one-way permutations). Moreover, while the existing simultaneous secret exchange protocols have an additional requirement that the underlying commit (encryption) function is "ideal", the above-mentioned "general assumptions" are provably sufficient for our schemes. Therefore, our protocols are provably secure under the general assumptions. In addition, our protocols are at least as efficient as the existing practical protocols, when efficient one-way permutations and one-way functions are used.

1 Introduction

1.1 Background

Let A and B be two users who exchange messages serially over a telecommunication network. To do business over such a network, they often need to exchange contracts "simultaneously", and to exchange a message and its receipt "simultaneously". Of course, such a simultaneous exchange of messages is physically impossible when using serial communication. Some protocols, however, achieve the virtually equivalent of simultaneous exchange of messages even with serial communication. A protocol for exchanging contracts simultaneously is called a *contract signing* protocol, and a protocol for exchanging a message and its receipt simultaneously is called a *certified mail* protocol.

Roughly speaking, there are three kinds of solutions to the problems of contract signing and certified mail: Types 1, 2 and 3. The first one (Type 1) is a trivial solution, which uses a reliable third party who takes an active part in the protocol. However, the assumption of the existence of a third party is considered too strong.

The second one (Type 2) assumes the existence of a "weak" third party, which serves as a reliable source of randomness and can play the role of a judge [BGMR, Rab]. Although Type 2 solutions have the advantage that they make no assumptions regarding the computing power of parties, requiring the use of a third party is still considered somewhat strong. Moreover, the Type 2 methodology can be applied only to contract signing and not to certified mail.

The third (Type 3) assumes no third party, and can be applied to both contract signing and certified mail [BCDG, Blu, Cle, Dam, EGL, GwL, LMR, Yao]. Therefore, the Type 3 approach seems to be superior to the other type approaches.

In Type 3 solutions, a "simultaneous secret exchange" (or "gradual secret releasing") protocol for encrypted (or committed) secret messages is essentially used in combination with a digital signature scheme [BCDG, Blu, Cle, Dam, EGL, GwL, LMR, Yao].

A simultaneous secret exchange protocol between parties A and B is usually executed as follows: First, A generates a (n bit string) secretly and likewise B generates b (n bit string) secretly, and they exchange $f(a)$ and $g(b)$, where A cannot get b from $g(b)$ (similarly B cannot get a from $f(a)$). Then, A and B open a and b bit by bit. A simultaneous secret exchange protocol should satisfy the following two conditions (informally described here):

CCS '94- 11/94 Fairfax Va., USA

- **Correctness:** The validity of each bit should be checked by each party at each stage, to ensure that a garbage has not been received. Even without the check at each stage, garbage bits can be detected finally after all n bits are sent, but it is too late, since an honest party gets garbage bits but the other dishonest party gets correct bits. Hence, the validity check at each stage is necessary.

- **Fairness:** When the running time of computing the remaining i bits of a from $f(a)$ after the first $(n-i)$ bits of a are released is $T(i)$, the difference between $T(i)$ and $T(i-1)$ $(1 \leq i \leq n)$ should be small. If the difference is non-negligible, then B has non-negligible advantage over A at the stage that A has released $(n-i+1)$ bits and B has released $(n-i)$ bits. Here, note that this condition should be satisfied even when $i = 1$. That is, $T(1)$ should be almost 0, since $T(0) = 0$. So, for example, the following naive protocol does not satisfy the fairness, since there exists a big gap between $T(1)$ and $T(0)$: $f(a) = (f_0(x_1), \ldots, f_0(x_n))$, where $a = (HC(x_1), \ldots, HC(x_n))$. $(HC(x)$ denotes the hard core bit [GrL] of $x.)$ x_i $(i = 1, \ldots, n)$ is revealed at A's (or B's) i-th step of the revealing phase. Then, $T(1)$ is the running time of inverting f_0, which is non-negligibly greater than $T(0) = 0$.

Intuitively, when these conditions are satisfied, for polynomial time machines A and B, at any stage of a simultaneous secret exchange protocol, if B gets a, then A gets b, and if A gets b, then B gets a.

Simultaneous secret exchange protocols can be constructed just by symmetrically using "gradual secret releasing protocols (bit by bit)", in which a party (e.g., A) commits a by $f(a)$, then releases a bit by bit. In other words, gradual secret releasing protocols are one-directional, while simultaneous secret exchange protocols are bi-directional.

A typical way to utilize a simultaneous secret exchange protocol for a contract signing protocol, in which A and B exchange signatures for a contract C, is as follows: First, A sends a message M_A and A's signature of M_A to B, where M_A is "I am committed to the contract C if B can present a solution (a) for problem $f(a)$". B acts in the same way (B's secret is b). Then, A and B exchange a and b bit by bit. Intuitively, at any stage of the protocol, if B gets a or A's signature for C, then A gets b or B's signature for C, and if A gets B's signature, then B gets A's signature.

Hereafter, this paper simply refers to the "simultaneous secret exchange protocol" as the "secret exchange protocol".

1.2 Problems of the existing secret exchange protocols

If a zero-knowledge protocol for any NP language [GMW] is used, a secret exchange protocol (gradual secret releasing protocol) can be constructed based on a one-way function (using the construction of a bit commitment scheme from a one-way function [ILL, Has, Nao]). That is, a party A commits a by $f(a)$, then releases a bit by bit along with the zero-knowledge proof which proves the correctness of the released bit to receiver B. However, this construction is far from being efficient (or practical) incurring a high communication and computation overhead. Here, "efficiency" means the communication and computation complexity is a low degree (e.g., less than 5) polynomial in the size of the secret string, n (or n^c, where $c < 5$). [1]

Many "efficient" secret exchange protocols have been presented [BCDG, Blu, Cle, Dam, EGL, LMR, Yao]. However, these secret exchange protocols are based on specific number theoretic problems or oblivious transfer primitives (or trap-door one-way permutations). For example, [Blu, Cle, LMR, Yao] are based on the factoring problem, [BCDG] is based on the discrete logarithm problem, [Dam] is based on both, and [EGL] is based on the oblivious transfer primitive (or trap-door one-way permutation). Note that in [Dam], a signature itself is committed and exchanged bit by bit, while in the other schemes, a random string (a) is committed and exchanged bit by bit, and the committed value $(f(a))$ is embedded in a signed message.

The other crucial problem of the existing schemes is that they need an additional assumption. That is, even if a (n bit string) is released bit by bit along with a proof of correctness for the released bit, the first i $(i < n)$ released bits of a might be enough to invert $f(a)$, when these i bits are in a fraction of i bit strings. For example, for a one-way function f, given $f(a)$ and the first $\lfloor n/2 \rfloor$ bits of a which is in a non-negligible fraction of $\lfloor n/2 \rfloor$ bit strings, it might be easy to compute a, although, given $f(a)$ and the first $(\lfloor n/2 \rfloor)$ bits of a which is not the fraction, it might be hard to compute a. Therefore, the existing schemes must assume that the underlying one-way function f is "ideal" (or "uniformly secure" [EGL]). Note that, in [Dam], the underlying signature verification function corresponds to the one-way function f above and that the verification function is similarly required to satisfy this condition.

To summarize the problems of the existing schemes are as follows:

- Any existing "efficient" simultaneous secret ex-

[1] Although the condition that $c < 5$ looks arbitrary, algorithms which do not satisfy this efficiency condition are often considered to be impractical. So, we set the condition as $c < 5$ in this paper, although this condition is not absolute.

change protocol is based on constrained assumptions such as specific number theoretic problems and the existence of oblivious transfer primitives (or trap-door one-way permutations).

- In addition to the above-mentioned assumptions, any existing simultaneous secret exchange protocol requires the other assumption that the underlying commit (encryption) function is "ideal" (or "uniformly secure" [EGL]).

Therefore, no secret exchange protocol which is efficient and provably secure under a general assumption such as the existence of one-way permutations and one-way functions has been proposed.

1.3 Results of this paper

This paper proposes the first secret exchange (gradual secret releasing) protocols that solve these problems. That is, the proposed protocols are efficient and provably secure under general assumptions such as the existence of one-way permutations and one-way functions[2].

We propose three protocols. The first one (Protocol 1) is based on one-way permutations, and is the most practical of the three. The second (Protocol 2) is based on one-to-one one-way functions, and is slightly less practical than the first one. Protocols 1 and 2 are applicable to both contract signing and certified mail. The third (Protocol 3) is based on regular one-way functions, and is almost as efficient as the first one but is applicable only to contract signing and not to certified mail (since Protocol 3 is a "weak" gradual secret releasing protocol, while Protocols 1 and 2 are gradual secret releasing protocols).

The key technique in our schemes is to utilize many one-way functions with series of security parameters which are selected from the same one-way function family. Informally, our schemes have the following feature: first, n bit secret a is committed to, and when $(n-i)$ bits of a have been released ($i = n, n-1, \ldots, 1$), the problem of revealing the remaining i-bits is the same as the problem of inverting a function F_i selected from a family of one-way permutations/functions $\mathcal{F}$, where i is the security parameter of this function.

1.4 Organization of this paper

This paper is organized as follows. Section 3 gives the definitions of one-way functions/permutations and "secure" gradual secret releasing protocols (and simultaneous secret exchange protocols). In section 4, a gradual secret releasing protocol based on a one-way permutation (Protocol 1) is presented, in section 5, a gradual secret releasing protocol based on a one-to-one one-way function (Protocol 2) is presented, and in section 6, a weak gradual secret releasing protocol based on a regular one-way function (Protocol 3) is presented. Section 7 compares the characteristics of Protocols 1, 2 and 3, and the applicability of these protocols to contract signing and certified mail protocols.

[2]Note that the efficiency of the underlying one-way function is trivially required for any protocol which is efficient.

2 Notations

- $x \in_U D$ denotes that x is selected from set D randomly and uniformly.
- $x \| y$ denotes the concatenation of string x and string y.
- $|x| = \lceil \log_2 x \rceil$. When A is a set, $\#A$ denotes the number of the elements of A.
- $[x]_a$ denotes the least a bits of x, $[x]^a$ denotes the most a bits of x.
- $x \odot y$ denotes the binary inner product of x and y. That is, when x and y are n bit strings such that $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$,

$$x \odot y = x_1 y_1 \oplus x_2 y_2 \oplus \cdots \oplus x_n y_n.$$

- Let function $f : D \rightarrow C$. D is called the *domain*, and denoted by $Dom(f)$, and C is called the *codomain*. $Im(f)$ denotes the *image* of function f of D ($Im(f) \subseteq C$). If $C = Im(f)$, f is called *onto*. For $y \in Im(f)$, $f^{-1}(y) = \{x \mid x \in D, f(x) = y\}$ is called the preimage of y. If for any $y \in Im(f)$, $\#f^{-1}(y) = 1$, then f is called *one-to-one*.
- $f \circ g(\cdot) = f(g(\cdot))$.

3 Definitions

3.1 One-Way Functions and One-Way Permutations

Definition 3.1 *A set of functions $\mathcal{F}$ is a family of one-way functions if*

$$\mathcal{F} = \{f_{\sigma_n} : D_{\sigma_n} \rightarrow C_{\sigma_n}\}_{\sigma_n \in I},$$

where I is an infinite set (of indices), $\forall \sigma_n$, D_{σ_n} is a finite set (the domain of f_{σ_n}), and $\forall \sigma_n$, C_{σ_n} is a finite set (the codomain of f_{σ_n}), and the following conditions are satisfied.

- *There exists probabilistic polynomial time algorithms S_1 and S_2 such that S_1, input 1^n, samples $\sigma_n \in I$, and S_2, input $\sigma_n \in I$, samples $x \in D_{\sigma_n}$. Here we call n the security parameter of f_{σ_n}.*

- *There exists a deterministic polynomial time algorithm, which, input $\sigma_n \in I$, and $x \in D_{\sigma_n}$, computes $f_{\sigma_n}(x)$.*
- *For any probabilistic polynomial time algorithm A, $\forall c > 0$, $\exists N_0$, $\forall n > N_0$,*

$$\Pr(A(\sigma_n, f_{\sigma_n}(x)) \in f_{\sigma_n}^{-1}(f_{\sigma_n}(x))) < 1/n^c.$$

The probability is taken over the coin flips of A and the sample distributions of σ_n and x.

Definition 3.2 *A family of one-way functions $\mathcal{F}$ is called regular if, for all $f \in \mathcal{F}$ and all $y_1, y_2 \in Im(f)$, $\#f^{-1}(y_1) = \#f^{-1}(y_2)$.*

Definition 3.3 *A family of one-way functions $\mathcal{F}$ is called with recognizable image if, for all $f \in \mathcal{F}$, $Im(f)$ (the image of f) is polynomial time recognizable (or $Im(f) \in \boldsymbol{P}$).*

Definition 3.4 *A family of one-way functions $\mathcal{F}$ is called a family of one-way permutations if, for all $f \in \mathcal{F}$, f is one-to-one, and its domain and image are equivalent ($D = Dom(f) = Im(f)$), where D is polynomial time recognizable (or $D \in \boldsymbol{P}$) and there exists a constant c such that, for all $f \in \mathcal{F}$, $c \leq \#D/2^M$. Here, $M = \min\{m \mid D \subseteq \{0,1\}^m\}$.*

Definition 3.5 *A family of one-way permutations $\mathcal{F}$ is called tight if, for all $f \in \mathcal{F}$, its domain and image are $\{0,1\}^n$.*

3.2 Simultaneous Secret Exchange and Gradual Secret Releasing Protocols

A simultaneous secret exchange protocol can be constructed using a gradual secret releasing protocol.

We now show the definition of a *secure* gradual secret releasing protocol. Generally, there are interactions between A and B, but in our protocol A only sends messages to B, who receives and checks the validity of messages. So, we define it based on our construction.

Definition 3.6 *A gradual secret releasing protocol (A, B) has the following properties. (A, B) is an interactive pair of Turing machines [Cle, FFS], and both A and B are polynomial time bounded in the input length, n. In our model, A is a probabilistic machine, and B is a deterministic machine. The protocol first runs for a commitment stage, in which A sends Σ_n and $X_n = F_{\Sigma_n}(s_n)$ to B, where $F_{\Sigma_n} \in_U \mathcal{F}^*$ (n: security parameter), and $|s_n| = n$. After the commitment stage (the first step), a series of steps numbered $2, \ldots, n+1$ is followed as the gradually releasing stage. At the $(n-i+1)$-th step (when i more steps remain unexecuted), A sends a string to B, which releases the $(n-i)$-th bit of s_n along with additional information (witness) w_i. After completion of the $(n-i+1)$-th step ($i = n-1, \ldots, 0$), B outputs β_i, the most $(n-i)$ bits of s_n (say $[s_n]^{n-i}$) along with $(w_{n-1}, \ldots, w_i)$.*

A gradual secret releasing protocol, (A, B), is secure if the following properties are satisfied:

- **Completeness:** *If both parties, A and B, follow the protocol, then B outputs s_n along with $w = (w_{n-1}, \ldots, w_0)$ with probability 1.*
- **Soundness:** *Assume that B follows the protocol. For any A^*, for any $i (0 \leq i \leq n-1)$, if (A^*, B) completes the first $(n-i+1)$ steps, then B outputs $[s_n]^{n-i}$ along with $(w_{n-1}, \ldots, w_i)$. Here, there exists a polynomial time predicate P such that $P(i, \Sigma_n, X_n, \beta_i) = 1$ if and only if $\beta_i = ([s_n]^{n-i}, (w_{n-1}, \ldots, w_i))$.*
- **Fairness:** *$\mathcal{F}^*$ is a family of "one-way" permutations/functions. Assume that A follows the protocol. At the $(n-i+1)$-th step ($i = n-1, \ldots, 0$), when i bits of s_n are unreleased, the information (distribution), H_i, which B is given from the beginning through the $(n-i+1)$-th step, includes $X_i = F_{\Sigma_i}(s_i)$, where s_i is the least i bits of s_n (or the unreleased bits of s_n at this step), and Σ_i (i: security parameter) is a part of Σ_n. Then, there exists a probabilistic polynomial time machine M (simulator) such that, for any B^*, for any $i \in \{0, \ldots, n-1\}$, for any Σ_i, for any X_i, $H_i \mid (i, \Sigma_i, X_i)$ is perfectly indistinguishable to (or equivalent to) $M^{B^*}(i, \Sigma_i, X_i)$. Here, $H_i \mid (i, \Sigma_i, X_i)$ denotes the conditional distribution under that H_i includes (i, Σ_i, X_i), where the probability is taken over A's coin flips.*

Notes:

1. Informally, the soundness condition means that B can check the validity of A's message, at each step, to ensure that a garbage has not been received.
2. Informally, the fairness condition means that, at the $(n-i+1)$-th step, the information of H_i except (i, Σ_i, X_i) does not help to compute the unreleased secret, s_i. In other words, the problem for B to compute s_i at this step is exactly the same problem to compute s_i only from (i, Σ_i, X_i) or to invert F_{Σ_i}. That is, at the $(n-i+1)$-th step, to compute the unreleased bits is just to invert a one-way function with security parameter i. Therefore, when the running time of computing the remaining i bits at the $(n-i+1)$-th step is $T(i)$, the difference between $T(i)$ and $T(i-1)$ $(1 \leq i \leq n)$ is just the difference of time between to invert F_{Σ_i} and to invert $F_{\Sigma_{i-1}}$. For a polynomial time machine

A, such a difference is negligible, since for sufficiently large i, if A can invert $F_{\Sigma_{i-1}}$, then A can also invert F_{Σ_i}. This is also satisfied when $i = 1$, since F_{Σ_1} can be easily inverted by A.

3. As mentioned above, using the gradual secret releasing protocol, a *secret exchange* protocol (A, B) can be realized as follows: first, they exchange the commitment to their secrets, then they repeatedly exchange one-bit release of their secrets through a gradual secret releasing protocol. Then, from the fairness of the gradual secret releasing protocol, at any step during the protocol, the difference between A and B of the amount of computing the other's unreleased secret is at most the difference of inverting F_{Σ_i} and $F_{\Sigma'_{i+1}}$, where they are selected from the same family of one-way permutations/functions and the difference of the security parameters between F_{Σ_i} and $F_{\Sigma'_{i+1}}$ is just 1 $(= (i+1) - i)$.

4. Generally, more than 1 bit of secret information can be released. That is, at the $(n-i+1)$-th step, A can release $(l(i+1) - l(i))$ bits to B, where $|s_n| = l(n)$ and $|s_i| = l(i)$.

Definition 3.7 *A weak gradual secret releasing protocol (A, B) is defined in the same manner as the gradual secret releasing protocol (Definition 3.6) except the following. In the secret exchange protocol, a value (s_n) which is committed to is uniquely revealed. In the "weak" gradual secret releasing protocol, in whose commitment stage A commits s_n to B, the value committed to is not always uniquely revealed. That is, after s_n is committed to in the commitment stage, s^*_n which is different from s_n can be revealed, if s^*_n is verified as the committed value. In other words, multiple values can be valid for one committed value in the "weak" gradual secret releasing protocol. Note that, however, an honest party releases the value which (s)he committed to.*

A weak gradual secret releasing protocol, (A, B), is secure if the following properties are satisfied:

- **Completeness:** *Same as Definition 3.6.*
- **Soundness:** *Assume that B follows the protocol. For any A^*, for any $i (1 \leq i \leq n)$, if (A^*, B) completes the first $(n - i + 1)$ steps, then B outputs $[s^*_n]^{n-i}$ along with $(w^*_{n-1}, \ldots, w^*_i)$. Here, s^*_n is one of the valid revealing values of A^*'s commitment to s_n, and $(w^*_{n-1}, \ldots, w^*_i)$ is the corresponding witness. There exists a polynomial time predicate P such that $P(i, \Sigma_n, X_n, \beta_i) = 1$ if and only if $\beta_i = ([s^*_n]^{n-i}, (w^*_{n-1}, \ldots, w^*_i))$.*
- **Fairness:** *Same as Definition 3.6.*

Note: Generally, more than 1 bit of secret information can be released (e.g., Protocol 3 in this paper).

4 Gradual Secret Releasing Protocol Based on One-Way Permutations

4.1 Gradual Secret Releasing Protocol Based on a Tight One-Way Permutation

Let $\mathcal{F}$ be a family of tight one-way permutations, and $f_{\sigma_i} \in \mathcal{F}$, where i denotes the security parameter, or $f_{\sigma_i} : \{0,1\}^i \to \{0,1\}^i$.

Protocol: (Gradual Secret Releasing Protocol 1)

Step 1 A randomly generates the parameters (indices) of a family of tight one-way permutations $\mathcal{F}$, $\sigma_1, \sigma_2, \ldots, \sigma_n$. Here

$$f_{\sigma_i} : \{0,1\}^i \to \{0,1\}^i, \ (i = 1, 2, \ldots, n).$$

A also selects an n-bit random string, $s_A = (x_n, \ldots, x_1)$, uniformly (i.e., $s_A \in_U \{0,1\}^n$). Then A calculates n-bit string X as follows:

$$x^*_1 = x_1,$$

$$x^*_i = x_i \| f_{\sigma_{i-1}}(x^*_{i-1}), \ (i = 2, \ldots, n),$$

$$X = f_{\sigma_n}(x^*_n).$$

Step 2 A sends $(\sigma_1, \sigma_2, \ldots, \sigma_n)$, and X to B to commit to A's secret x.

Step 3 When B receives them, B checks whether $(\sigma_1, \sigma_2, \ldots, \sigma_n)$ are valid for the parameters of $\mathcal{F}$, and whether $X \in \{0,1\}^n$. If they do not hold, B halts the protocol. Otherwise, B writes $(\sigma_1, \sigma_2, \ldots, \sigma_n)$, X on the output tape.

[End of commitment stage]

Step 4 For $i = n, \ldots, 1$, repeat the following procedures sequentially.
A sends x^*_i to B.

Step 5 When $i = n$, B checks whether $X = f_{\sigma_n}(x^*_n)$ holds or not.
When $i = n-1, \ldots, 1$, B checks whether

$$[x^*_{i+1}]_i = f_{\sigma_i}(x^*_i).$$

If it does not hold, B halts the protocol. Otherwise, B writes x_i and x^*_i on the output tape.

$$x_i = [x^*_i]^1.$$

[End of secret releasing stage]

Note:

1. f_{σ_i} can be commonly shared by many users. If so, the procedures for generating the parameters can be omitted from the above protocol.

2. s_A corresponds to s_n in Definition 3.6, and $[x_i^*]_{n-1}$ corresponds to w_i.

Theorem 4.1 *If $\mathcal{F}$ is a family of tight one-way permutations, then Gradual secret releasing protocol 1 is secure.*

Sketch of Proof:

Completeness: This is trivial.

Soundness: Assume that B follows the protocol, and that (A^*, B) completes the first $(n-i+1)$ steps. Then, for any strategy of A^*, the following are true (confirmed by B):

- The parameters of $\mathcal{F}$, σ_j $(j = 1, \ldots, n)$, are valid,
- $X \in \{0,1\}^n$,
- $$X = f_{\sigma_n}(x_n^*),$$
$$[x_{j+1}^*]_j = f_{\sigma_j}(x_j^*) \ (i \leq n-2, i+1 \leq j \leq n-1).$$

Let

$$F_i(x_i, x_{i-1}, \ldots, x_1) =$$
$$f_{\sigma_i}(x_i \| f_{\sigma_{i-1}}(x_{i-1} \| f_{\sigma_{i-2}}(\cdots \| f_{\sigma_1}(x_1) \cdots))).$$

Then, F_i is a permutation from $\{0,1\}^i$ to $\{0,1\}^i$, and $F_i(x_i, \ldots, x_1) = f_{\sigma_i}(x_i^*)$. Therefore, for any $X \in \{0,1\}^n$, there exists a unique n-bit string, $(x_n, \ldots, x_1)$, such that $X = F_n(x_n, \ldots, x_1)$. Therefore, a unique $s_A = (x_n, \ldots, x_1)$ is determined from the value X. Since $f_{\sigma_n}, \ldots, f_{\sigma_{i+1}}$ are also permutations over $\{0,1\}^n$, ..., $\{0,1\}^{i+1}$, respectively, there exists a unique vector of strings $(x_n^*, \ldots, x_{i+1}^*)$ satisfying the following equations.

$$X = f_{\sigma_n}(x_n^*),$$
$$[x_{j+1}^*]_j = f_{\sigma_j}(x_j^*) \ (i \leq n-2, i+1 \leq j \leq n-1).$$

Then, the most significant bit of x_j^* is equivalent to x_j for $j = n, \ldots, i+1$. Thus, for any A^*, for any i $(1 \leq i \leq n)$, if B follows the protocol, the probability that (A^*, B) completes the first $(n-i+1)$ steps, and that B's output $(x_n, \ldots, x_{i+1})$ is different from the most $(n-i)$ bits of x that is uniquely determined from X is zero. Here, predicate P is the verification procedure by B.

Fairness: First, we show that $\mathcal{F}^* = \{F_i\}$ is a family of one-way permutation, if $\mathcal{F} = \{f_{\sigma_i}\}$ is a family of one-way permutations. If there exists a probabilistic polynomial time algorithm $P1$ of inverting f_{σ_i} with nonnegligible probability, then a probabilistic polynomial time algorithm of inverting F_i with nonnegligible probability is clearly constructed. If there exists a probabilistic polynomial time algorithm $P2$ of inverting F_i with nonnegligible probability, then a probabilistic polynomial time algorithm of inverting f_{σ_i} with nonnegligible probability is also constructed. First, σ_i, $z = f_{\sigma_i}(y)$ is given. Then, generate σ_j $(j = 1, \ldots, i-1)$. Here, F_i is defined by $(\sigma_1, \ldots, \sigma_i)$. Then, run $P2$ for inputs, $(\sigma_1, \ldots, \sigma_i)$, and z, and obtain $(y_i, \ldots, y_1)$. So,

$$y = (y_i \| f_{\sigma_{i-1}}(y_{i-1} \| \cdots f_{\sigma_2}(y_2 \| f_{\sigma_1}(y_1)) \ldots))).$$

Thus, to invert F_i is at least as difficult as to invert f_{σ_i}. Since $\mathcal{F} = \{f_{\sigma_i}\}$ is a family of one-way permutations, $\mathcal{F}^*$ is also a family of one-way permutations.

Next, we show that the major part of the fairness condition. Let $H_i = (i, (\sigma_1, \ldots, \sigma_n), X, (x_n^*, \ldots, x_{i+1}^*))$ which B is given from the beginning through the $(n-i+1)$-th step. $(x_n, \ldots, x_{i+1})$ can be calculated from $(x_n^*, \ldots, x_{i+1}^*)$ (or H_i). Clearly, H_i includes

$$[x_{i+1}^*]_i = f_{\sigma_i}(x_i \| f_{\sigma_{i-1}}(x_{i-1} \| \cdots \| f_{\sigma_1}(x_1) \cdots))$$
$$= F_i(x_i, \ldots, x_1).$$

Simulator M is constructed as follows: Given $(i, (\sigma_1, \ldots, \sigma_i), Z = [x_{i+1}^*]_i = F_i(x_i, \ldots, x_1))$, M uniformly selects $(x'_{i+1}, \ldots, x'_n)$ and $(\sigma'_{i+1}, \ldots, \sigma'_n)$, and computes $W_m = x_m \| f_{\sigma_{m-1}}(x_{m-1} \| \cdots f_{\sigma_{i+1}}(x_{i+1} \| Z) \cdots)$, $(m = i+1, \ldots, n+1$, x_{n+1} is null string). Then, the conditional distribution of

$$H_i \mid (i, (\sigma_1, \ldots, \sigma_i), Z)$$
$$= ((\sigma_{i+1}, \ldots, \sigma_n), X, (x_n^*, \ldots, x_{i+1}^*)) \mid (i, (\sigma_1, \ldots, \sigma_i), Z)$$

is exactly equivalent to the distribution of

$$((\sigma'_{i+1}, \ldots, \sigma'_n), W_{n+1}, (W_n, \ldots, W_{i+1})) \mid (i, (\sigma_1, \ldots, \sigma_i), Z).$$

□

Proposition 4.2 *The computational complexity of Gradual secret releasing protocol 1 is $O(n\|\mathcal{F}\|)$, and the communication complexity is $O(n^2)$, where $\|\mathcal{F}\|$ denotes the complexity of computing f_{σ_n} in $\mathcal{F}$. If $\|\mathcal{F}\| = O(n^3)$, then $O(n\|\mathcal{F}\|) = O(n^4)$.*

4.2 Gradual Secret Releasing Protocol Based on a One-Way Permutation

In this section, we show that any one-way permutation can be converted to a tight one-way permutation with overwhelming probability. By combining this conversion and Protocol 1, we can construct a gradual secret releasing protocol based on any one-way permutation.

Lemma 4.3 *There exists a probabilistic polynomial time algorithm which constructs a tight one-way permutation f using any one-way permutation g with overwhelming probability.*

Proof:

From the definition of a one-way permutation, g has the same domain and image, D, and there exists a constant c $(0 < c \leq 1)$ such that for all $g \in \mathcal{G}$ $c \leq \#D/2^n$, where $n = \min\{m \mid D \subseteq \{0,1\}^m\}$. Let h_i $(i = 1, \ldots, 2dn)$ be $2dn$ independently selected random permutations (e.g., exclusive oring by a random string) from $\{0,1\}^n$ to $\{0,1\}^n$, where $d = 1/(\log(1/(1-c)))$, which is also a constant. $g^* : \{0,1\}^n \to \{0,1\}^n$ is defined as $g^*(x) = g(x)$ if $x \in D$ and $g^*(x) = x$ if $x \notin D$. Let $f = g^* \circ h_1 \circ g^* \circ h_2 \circ g^* \circ h_3 \circ \cdots \circ g^* \circ h_{2dn} \circ g^*$. f is clearly a tight permutation, since g^* and h_i are tight permutations. The probability that there exists $x \in \{0,1\}^n$ such that $f(x) = h_1 \circ h_2 \circ h_3 \circ \cdots \circ h_{2dn}(x)$ is negligible, since for a value x, the probability that $(h_1, h_2, \ldots, h_{2dn})$ are selected such that $f(x) = h_1 \circ h_2 \circ h_3 \circ \cdots \circ h_{2dn}(x)$ is at most $(1-c)^{2dn} = (1/2)^{2n}$. Hence, f is one-way with overwhelming probability, since for all $x \in \{0,1\}^n$, g^* is operated at least once to calaculate $f(x)$ with overwhelming probability. Thus, f is a tight one-way permutation with overwhelming probability. □

When a one-way permutation, g, is specific, the conversion from g to a tight one-way permutation, f, is more efficient. We show two typical examples: $g_1 : x \in Z_{p-1} \mapsto (a^x \bmod p) - 1 \in Z_{p-1}$, and $g_2 : x \in Z_N \mapsto x^e \bmod N \in Z_N$. Here, p is a prime and a is a generator of Z_p^*, and $\gcd(\phi(N), e) = 1$ ($\phi(N)$ is Euler's totient function). Now let $n = |p-1|$ for g_1 and $n = |N|$ for g_2. we define $h = \{0,1\}^n \to \{0,1\}^n$ as $h(x) = x + 2^{n-1} \bmod 2^n$. Then, $f_i = g_i \circ h \circ g_i$ $(i = 1, 2)$ is a tight one-way permutation.

Theorem 4.4 *A secure gradual secret releasing protocol, Protocol 1, can be constructed using any family of one-way permutations.*

5 Gradual Secret Releasing Protocol Based on a One-to-one One-Way Function

This section introduces a gradual secret releasing protocol based on a *one-to-one* one-way function *with recognizable image* (Definition 3.3), which is more general than a one-way permutation.

This protocol is still practical, but is a little bit less practical than Protocol 1, in the light of the communication complexity of the commitment stage.

Let $\mathcal{F}$ be a family of one-to-one one-way functions with recognizable images, and $f_{\sigma_i} \in \mathcal{F}$, where i denotes the security parameter. That is, $|\#Dom(f_{\sigma_i})| = |\#Im(f_{\sigma_i})| = i$. For simplicity of description, w.l.o.g., we assume that $Dom(f_{\sigma_i}) = \{0,1\}^i$.

Protocol: (Gradual Secret Releasing Protocol 2)

Step 1 A randomly generates the parameters (indices) of a family of one-to-one one-way functions with recognizable images $\mathcal{F}$, $\sigma_1, \sigma_2, \ldots, \sigma_n$. A also selects i-bit random strings x_i^* $(i = n, \ldots, 1)$, (or $x_n^*, x_{n-1}^*, \ldots, x_1^*$), and an n-bit random string p uniformly (or $x_i^* \in_U Dom(f_{\sigma_i}) = \{0,1\}^i$, $p \in_U \{0,1\}^n$). Then A calculates n strings $X_n, X_{n-1}, \ldots, X_1$ as follows:

$$X_i = f_{\sigma_i}(x_i^*)$$

A also calculates n-bit string $s_A = (a_n, \ldots, a_1)$ $(a_i \in \{0,1\})$ as follows:

$$a_n = x_n^* \odot p,$$

$$a_i = (a_n, \ldots, a_{i+1}, x_i^*) \odot p \quad (1 \leq i \leq n-1).$$

Step 2 A sends $(\sigma_1, \sigma_2, \ldots, \sigma_n)$, p, and $(X_n, X_{n-1}, \ldots, X_1)$ to B to commit to A's secret s_A.

Step 3 When B receives them, B checks whether $(\sigma_1, \sigma_2, \ldots, \sigma_n)$ are valid for the parameters of $\mathcal{F}$, and whether $p \in \{0,1\}^n$, and $X_i \in Im(f_{\sigma_i})(i = 1, \ldots, n)$. If they do not hold, B halts the protocol. Otherwise, B writes $(\sigma_1, \sigma_2, \ldots, \sigma_n)$, p, and $(X_n, X_{n-1}, \ldots, X_1)$ on the output tape.

[End of commitment stage]

Step 4 For $i = n, \ldots, 1$, repeat the following procedures sequentially.
A sends x_i^* to B.

Step 5 B checks whether $X_i = f_{\sigma_i}(x_i^*)$ holds. If it does not hold, B halts the protocol. Otherwise, B writes a_i and x_i^* on the output tape.

$$a_n = x_n^* \odot p \ (i = n),$$

$$a_i = (a_n, \ldots, a_{i+1}, x_i^*) \odot p \quad (1 \leq i \leq n-1).$$

[End of secret releasing stage]

Theorem 5.1 *If $\mathcal{F}$ is a family of one-to-one one-way permutations with recognizable image, then Gradual secret releasing protocol 2 is secure.*

The proof is similar to the proof of Theorem 4.1.

Proposition 5.2 *The computational complexity of Gradual secret releasing protocol 2 is $O(n\|\mathcal{F}\|)$, and the communication complexity is $O(n^2)$.*

6 Weak Gradual Secret Releasing Protocol Based on a Regular One-Way Function

This section introduces a "weak" gradual secret releasing protocol based on *regular* one-way functions.

This protocol is almost as efficient as Protocol 1.

Let $\mathcal{F}$ be a family of regular one-way functions, and $f_{\sigma_i} \in \mathcal{F}$, where i denotes the security parameter. For simplicity of description, we assume that $f_{\sigma_i} : \{0,1\}^{l(i)} \mapsto \{0,1\}^i$, where $l(i) \geq i$.

Protocol: (Weak Gradual Secret Releasing Protocol)

Step 1 A randomly generates the parameters (indices) of a family of regular one-way functions $\mathcal{F}$, $\sigma_1, \sigma_2, \ldots, \sigma_n$. Here

$$f_{\sigma_i} : \{0,1\}^{l(i)} \to \{0,1\}^i, \ (i = 1, 2, \ldots, n).$$

A also selects $(l(i) - i + 1)$-bit random strings $(i = n, \ldots, 1)$, $(x_n, x_{n-1}, \ldots, x_1)$, (or $x_i \in_U \{0,1\}^{l(i)-i+1}$). Then A calculates n-bit string X as follows:

$$x_1^* = x_1,$$

$$x_i^* = x_i \| f_{\sigma_{i-1}}(x_{i-1}^*), \quad (i = 2, \ldots, n),$$

$$X = f_{\sigma_n}(x_n^*).$$

A sets $s_A = (x_n, \ldots, x_1)$.

Step 2 A sends $(\sigma_1, \sigma_2, \ldots, \sigma_n)$, and X to B to commit to A's secret s_A.

Step 3 When B receives them, B checks whether $(\sigma_1, \sigma_2, \ldots, \sigma_n)$ are valid for the parameters of $\mathcal{F}$, and whether $X \in \{0,1\}^n$. If they do not hold, B halts the protocol. Otherwise, B writes $(\sigma_1, \sigma_2, \ldots, \sigma_n)$, and X on the output tape.

[End of commitment stage]

Step 4 For $i = n, \ldots, 1$, repeat the following procedures sequentially.
A sends x_i^* to B.

Step 5 When $i = n$, B checks whether $X = f_{\sigma_n}(x_n^*)$ holds or not.
When $i = n-1, \ldots, 1$, B checks whether

$$[x_{i+1}^*]_i = f_{\sigma_i}(x_i^*).$$

If it does not hold, B halts the protocol. Otherwise, B writes x_i on the output tape.

$$x_i = [x_i^*]^{l(i)-i+1}.$$

[End of secret releasing stage]

Theorem 6.1 *If $\mathcal{F}$ is a family of regular one-way functions, then the weak gradual secret releasing protocol (Protocol 3) is secure.*

Proposition 6.2 *The computational complexity of Protocol 3 is $O(n\|\mathcal{F}\|)$, and the communication complexity is $O(n^2)$, if $l(n) = cn$ (c: constant).*

7 Applicability to Contract Signing and Certified Mail Protocols

This section compares the characteristics of Protocols 1, 2 and 3, and the applicability of these protocols.

- The assumption for Protocol 1, one-way permutations, is more constrained than those for Protocols 2 and 3, i.e., one-to-one one-way functions and regular one-way functions respectively. It cannot be determined which assumption is more general between those for Protocols 2 and 3.
- Protocols 1 and 2 can be applied to both contract signing and certified mail protocols, since they are "usual" gradual secret releasing protocols, while Protocol 3 can be applied only to contract signing protocols, since it is a "weak" gradual secret releasing protocol. The contract signing and certified mail protocols using "usual" gradual secret releasing protocols (secret exchange protocols) are constructed in a manner similar to the protocols given in [EGL]. A protocol similar to [EGL]'s contract signing protocol can also be constructed by using a "weak" gradual secret releasing protocol, while it is impossible for a certified mail protocol, since the uniqueness of revealing secret is essential for a certified mail protocol, but not so for a contract signing protocol.
- Protocols 1 and 3 are more efficient than Protocol 2, but Protocol 2 is still practical. Protocol 1 is almost as efficient as Protocol 3.

8 Practical Implementation Examples

The proposed protocols can be implemented efficiently, under the condition that the underlying one-wayy function family is efficient. Here, we show two practical examples of implemeting one-way function families.

Example 1: One-way permutation (Protocol 1)

First, we show an example of one-way permutation family based on the discrete logarithm. This function family is one-way if the discrete logarithm problem is intractable.

In the first step, n primes, p_i $(i = 1, 2, \ldots, n)$, are selected such that $|p_i| = i$ ($p_1 = 2$ and $p_2 = 3$) and $p_i - 1$ has at least one big prime factor (e.g., q_i is a prime factor of $p_i - 1$ and $|q_i| > |p_i|/2$). Let a_i be a generator of $Z^*_{p_i}$, and $g_i : x \in Z_{p_i - 1} \mapsto (a_i{}^x \bmod p_i) - 1 \in Z_{p_i - 1}$, Then, one-way permutation family $\mathcal{F} = \{f_i \mid i = 1, 2, \ldots, \}$ is defined as follows: $f_i = g_i \circ h \circ g_i$, where $h : \{0,1\}^i \rightarrow \{0,1\}^i$ is defined as $h(x) = x + 2^{i-1} \bmod 2^i$.

Example 2: Regular one-way function (Protocol 3)

Next, we show an example of "pseudo" regular one-way function families, based on a practical hash function. This function family does not seem to be regular, but it seems to be sufficient for our practical purpose to construct Protocol 3. This is because the cardinality of the preimage of any element in the image seems to be almost equivalent to that of other element with high probability. (The definition of "pseudo" regular one-way function families and their sufficiency for our purpose will be shown in the final paper.)

Let H be a practical hash function (such as SHA and MD5): $\{0,1\}^* \rightarrow \{0,1\}^N$. Then, $f_{\sigma_i} : \{0,1\}^{l(i)} \mapsto \{0,1\}^i$ is defined as $f_{\sigma_i}(x) = [H(x)]_i$, where $l(i) = N + i + C$. (C is a constant factor. e.g., $C = 80$.)

9 Conclusions

This paper proposed the efficient simultaneous secret exchange protocols (or gradual secret releasing protocols) that solves the problems of the existing simultaneous secret exchange protocols. The proposed protocols are efficient and are proven to be secure under general assumptions such as the existence of one-way permutations and one-way functions.

References

[BCDG] E. F. Brickell, D. Chaum, I. Damgård, J. and van de Graaf: "Gradual and Veriable Release of a Secret," Proc. of CRYPTO'87, pp.156-166 (1987)

[BGMR] M. Ben-Or, O. Goldreich, S. Micali, and R. Rivest, "A Fair Protocol for Signing Contracts," IEEE Tran. on IT, Vol. 36, Number 1, pp.40-46 (1990)

[Blu] M. Blum, "How to Exchange (Secret) Keys," Proc. of STOC'83, pp.440-447 (1983)

[Cle] R. Cleve, "Controlled Gradual Disclosure Schemes for Random Bits and Their Applications," Proc. of CRYPTO'89, pp.573-588 (1989)

[Dam] I. Damgård, "Practical and Provably Secure Exchange of Digital Signatures (Extended Abstract)," to appear in Proc. of EUROCRYPT'93

[EGL] S. Even, O. Goldreich, and A.Lempel, "A Randomized Protocol for Signing Contracts," Communication of the ACM, Volume 28, Number 6, pp.637-647 (1985)

[FFS] U. Feige, A. Fiat, and A. Shamir, "Zero-Knowledge Proofs of Identity," Journal of CRYPTOLOGY, Vol. 1, Number 2 (1988)

[GrL] O. Goldreich, and L. Levin, "A Hard-Core Predicate for any One-way Function," Proc. of STOC'89, pp.25-32 (1989)

[GwL] S. Goldwasser, and L. Levin, "Fair Computation of General Functions in Presence of Immoral Majority", in Advances in Cryptology — CRYPTO '90, Lecture Notes in Computer Science 537, Springer–Verlag, Berlin, pp.77–93 (1990).

[GMW] O. Goldreich, S. Micali, and A. Wigderson, "Proof that Yield Nothing but their Validity and a Methodilogy of Cryptographic Protocol Design," Proc. of FOCS'86 (1986)

[Has] J.Håstad, "Pseudo-Random Generators under Uniform Assumptions," Proceedings of STOC (1990)

[ILL] R. Impagliazzo, L. Levin, M. Luby, " Pseudo-Random Number Generation from One-Way Functions," Proceedings of STOC, pp.12-24 (1989)

[LMR] M. Luby, S. Micali, and C. Rackoff, "How to Simultaneously Exchange Secret Bit by Flipping a Symmetrically-Biased Coin," Proc. of FOCS'83, pp.23-30 (1983)

[Nao] M.Naor, "Bit Commitment Using Pseudo-Randomness," in *Advances in Cryptology — Crypto '89, proceedings*, Lecture Notes in Computer Science 435, Springer–Verlag, Berlin, (1990).

[Rab] M. Rabin, "How to Exchange Secrets by Oblivious Transfer," Tech. Memo, TR-81, Aiken Comp. Lab., Harvard University (1981)

[Yao] A. Yao, "How to Generate and Exchange Secrets," Proc. of FOCS'86, pp.162-167 (1986)