Light Cryptography

Pascal Lafourcade[1][0000-0002-4459-511X], Takaaki Mizuki[2][0000-0002-8698-1043], Atsuki Nagao[3][0000-0002-1370-5240], and Kazumasa Shinagawa[4,5][0000-0002-5219-1975]

[1] LIMOS, University Clermont Auvergne, CNRS UMR 6158, Aubière, France
[2] Tohoku University, Japan
[3] Ochanomizu University, Japan
[4] Tokyo Institute of Technology, Japan
[5] National Institute of Advanced Industrial Science and Technology, Japan

Abstract. Physical cryptography provides cryptographic protocols using physical objects like cards and envelopes instead of using computers. In this paper, we introduce a new model for physical cryptography, called *light cryptography*. It uses transparent sheets and some properties of light and shadows. We design several secure light cryptographic protocols: one for set-intersection (which can solve the scheduling problem), one for maximum (which can solve the Yao's Millionaires' problem), one for computing the sum of integers. We believe that our protocols using light cryptography are a powerful tool for information security education because they are fairly simple and fun to use.

Keywords: Secure computation, physical cryptography, light cryptography, information security education.

1 Introduction

Suppose that a group of friends wishes to hold a party next month, but they have not yet decided the date of the party. They wish to choose a date that suits everyone because they want the presence of everyone. In the usual way to arrange a schedule, each participant must reveal his convenient dates and times to other participants. However, for privacy reasons, some participants do not want to reveal their agendas to other ones because their agendas contain personal information such as professional meetings, personal appointments or hobby schedules.

Secure computation is a cryptographic solution for such a problem [2, 9, 17]. It enables parties to compute some function of their inputs while only revealing the output value and no information about the input values. It is known that secure computation for any function is possible based on cryptographic assumptions (e.g., the hardness of the enhanced trapdoor permutation [9]) or setup assumptions (e.g., the existence of secure channel [2]). Various secure computation protocols have been proposed following these pioneer works [2,9,17]. Despite their usefulness, it is difficult for non-experts to understand why they are correct

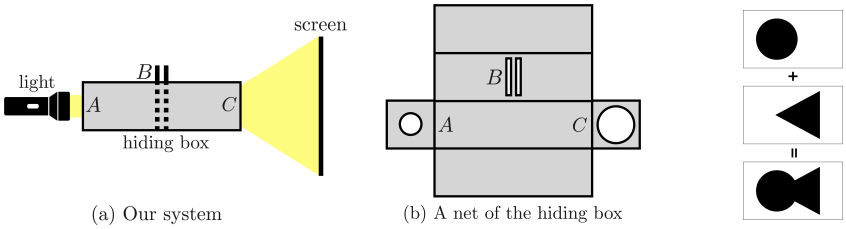
and secure because these protocols often rely on the knowledge of deep mathematics. Moreover, since we cannot check the inside of computers, a program implementing a protocol behaves like a black box from the user's point of view. It is clearly a difficulty to convince the user to trust such a computer-based system.

Physical cryptography is a suitable solution for secure computation to find a common date. The goal is to design a secure computation protocol using physical objects (e.g., cards, boxes, envelopes) instead of using computers. For example, *card-based cryptography* [5, 7] is a physical cryptographic model using a deck of cards; information is encoded in a sequence of face-down cards with for example the rule that ♣♡ represents a 0 and ♡♣ represents a 1, and a protocol consists of a list of operations like rearrangement, turning over, and shuffles. Because all computational flows are visible from all parties, it is easy to verify and understand the correctness and the security of protocols without the knowledge of deep mathematics.

Contributions: We introduce a new model for physical cryptography. All previous papers used physical objects like cards or envelopes and the users have to perform some computations. We change the paradigm and use *light* and its properties to compute the results of some functions securely; we naturally call this new approach *light cryptography*. Light cryptography allows us to design physical light cryptographic protocols. It is a secure computation protocol that uses transparent sheets and the properties of light and shadow. The idea comes from *ombromanie* (also known as *shadowgraphy*); this is the art of creating shadow images using hands (e.g., rabbits, birds, old men). The key observation is that it is sometimes difficult to guess the original shape of hands by watching only the printed shadow; thus, it has *one-wayness* in some sense like one-way functions in cryptography: a one-way function f is easy to compute but it is hard to find an input x from the output $y = f(x)$. In light cryptography like in ombromanie, it is hard to find the original shape of hands by giving only the shadow of the hands. We also believe that light and shadows are familiar physics concepts for everyone. Moreover, the correctness and the security of our protocols are easy to understand without the knowledge of deep mathematics.

We first define a model of secure computation based on light cryptography. Then we construct several protocols that use this concept for secure computations. We design a protocol to allow participants to determine a common date without revealing any information about their personal agendas. We also propose a maximum protocol that allows users to compute the maximum of their values in a secure way in order to demonstrate that light cryptography can easily solve the famous Yao's Millionaires' problem [16], where two millionaires want to know which has more money without revealing how much each has. We propose an extension that also gives the identity of the owner of the maximum. Finally, we propose a protocol to compute the sum of some integers.

We also believe that light cryptography can be a powerful tool for information security education which is important but somewhat difficult. Specifically, it is useful for teaching secure computation, which seems to attempt to achieve an



(a) Our system (b) A net of the hiding box

Fig. 1. Illustration of our system

Fig. 2. Shadow addition.

impossible (or unbelievable) task at the first glance. Light cryptography is a nice teaching material that gives an intuition that it is possible to do secure computation in some settings. The correctness and the security of protocols are also intuitive for students even who are not familiar with deep mathematics because its computation process is very simple; it just uses light. Moreover, it is suitable to perform our protocols in a classroom because result images can be displayed on a screen.

Related Work: There are many works of physical cryptography based on various physical objects. They are classified by type of input format. The first type of input format is by private actions like private rearrangements; for instance secure computations using a PEZ dispenser [1], a dial lock [11], marbles [8] and a 15 puzzle [12]. The other type of input format, which includes our model, is by submitting encoded objects like a sequence of face-down cards; such examples are secure computation protocols using a deck of cards [3–5,7,13], polarizing cards [15], and visual secret sharing schemes [6]. Polarizing cards [15] is based on the property of polarizing plates but the basic strategy to construct protocols is similar to other card-based cryptography [13]. D'Arco and De Prisco [6] proposed a physical cryptographic model, which combines the gate evaluation of secret sharing [10] and visual secret sharing scheme [14], which also uses transparent sheets for computation. While the paper presented a general protocol, it is somewhat complex compared to our protocols due to the gate evaluation from [10].

2 Model of Light Cryptography

In Section 2.1, we depict how the principles of light cryptography work. In Section 2.2, we define an important operation, *shadow addition*, which is an abstract property of light and shadow. In Section 2.3, we formally define *protocols* in our model of light cryptography.

2.1 Overview of Our System

Suppose that n players $P_1, P_2, \ldots, P_n$ having secret inputs $x_1, x_2, \ldots, x_n \in X$, respectively, wish to compute a joint function of their inputs $f(x_1, x_2, \ldots, x_n) \in Y$, where X and Y are a domain and a range of the function, respectively.

Before an execution of a protocol, we have to prepare the following:

- n transparent sheets (e.g., OHP (OverHead Projector) sheets),
- n black pens for transparent sheets,
- a light (e.g., projector),
- a screen,
- a non-transparent box called a *hiding box*.

Figure 1 (a) shows an illustration of our system viewing it directly above (when $n = 2$) and Figure 1 (b) shows a blueprint of the hiding box. The hiding box has a number of holes; the hole A is a hole for inputting light, the hole C is a hole for outputting light, and n holes B are used for inserting one transparent sheet per each player P_i.

At the beginning of a protocol, each player P_i has a transparent sheet (which is called an *input sheet*) and a black pen, and he/she writes a black image on the transparent sheet with the black pen according to his/her input x_i. Each player P_i covertly puts the sheet into the i-th hole of holes B of the hiding box. Finally, the hole A of the hiding box is illuminated by the light, and then, an output image corresponding to $f(x_1, x_2, \ldots, x_n)$ is displayed on the screen (see Figure 1 (a)).

2.2 Shadow Addition

Imagine that we have two transparent sheets (e.g., OHP sheets), on each of which black images are drawn. By superimposing them, we have a new image which is the union of two original images. We call it *shadow addition* and denote it by the symbol $+$.

We now formally define this operator. Let U be a set of all black-and-white images of fixed size. The shadow addition $+ : U \times U \to U$ is defined by $A + B = C$, where $A, B, C \in U$ and the black area of C is the union of the black areas of A and B. Figure 2 shows an example of shadow addition of two images: the input images are a circle and a triangle, and the resulting image is a keyhole. It is easy to observe that the shadow addition satisfies the *commutative law* and the *associative law*, i.e., it holds that $A + B = B + A$ and $A + (B + C) = (A + B) + C$ for any $A, B, C \in U$. We also have the *idempotence* property: $A + A = A$.

2.3 Defining Protocols

Formally, a *light cryptogrpahic protocol* is defined by a 7-tuple (n, U, I, X, Y, g, h), where n is the number of players, U is a set of all black-and-white images of the same size, $I \in U$ is an *initial image* which is drawn on every input sheet, X is the domain of the players' secret inputs, Y is the range of the output, $g : X \to U$ is an *input function*, and $h : U \to Y$ is an *output function*. The protocol proceeds as follows:

1. At the beginning of the protocol, each player has a black pen and an input sheet. On the input sheet, the initial image I has already been drawn.
2. According to his/her input $x_i \in X$, each player draws an image $g(x_i)$ on his/her input sheet; the new image is $I'_i = I + g(x_i)$.
3. Each player covertly puts his/her input sheet into a black box in turns.
4. By switching on light, every player knows the union image $I_{\mathsf{result}} \in U$ of their input sheets, where $I_{\mathsf{result}} = I'_1 + I'_2 + \cdots + I'_n$. The output of the protocol is $h(I_{\mathsf{result}}) \in Y$.

Correctness. We say that a protocol correctly computes a function f if for any $x_1, x_2, \ldots, x_n \in X$, the value $h(I_{\mathsf{result}})$ of the output function h for the union image $I_{\mathsf{result}} = I'_1 + I'_2 + \cdots + I'_n$ is equal to $f(x_1, x_2, \ldots, x_n)$, where $I'_i = I + g(x_i)$ for all $i \in \{1, 2, \ldots, n\}$.

Security. A protocol is secure if the output function h restricted on $U_X \subset U$ is injective, where U_X is the set of all images generated by the inputs.

We assume that all players are semi-honest, i.e., they follow the protocol specification, and they cannot see the input sheets in the hiding box even after the computation. In order to achieve such a hiding property, we can use a shredder to destroy the input sheets after the computation. Now let us explain the meaning of the security defined above. Suppose that a protocol Π correctly computes a function f. The security definition requires that for any pair of input-sequences $X = (x_1, x_2, \ldots, x_n), X' = (x'_1, x'_2, \ldots, x'_n) \in X^n$ such that $f(X) = f(X') = y$, the output images I_{result} and I'_{result} induced by X and X', respectively, are the same image, i.e., it holds that $I_{\mathsf{result}} = I'_{\mathsf{result}}$. Otherwise, the output function h is not injective because $h(I_{\mathsf{result}}) = h(I'_{\mathsf{result}}) = y$ holds from the correctness.

Enhancing Security. In our security model, all players are assumed to be semi-honest and they must not see the other players' input sheets. It can be accomplished by shredding the input sheets with a shredder just after the end of the protocol, but occasionally the input sheets may be seen by other players due to a mistake in operation. The idea for hiding the inputs even when such an accident occurs is applying a shuffle to the input sheets before putting them into the hiding box. Specifically, each player puts his own input sheet into an envelope and places it on a public table, and then all players together apply a shuffle on the envelopes in order to hide the order of the input sheets. Of course, when the input sheets are revealed after the computation, even if we applied the shuffle to them, some of input information is leaked. But in this case, an adversary cannot guess the correspondence between the input sheets and the players, while our original model leaks all inputs in the case of such an accident.

3 Set-Intersection (SI) Protocol

Suppose that n friends wish to decide the date of a party next month. They wish to make it a convenient schedule for everyone without revealing the convenient schedule of each participant.

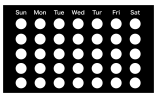
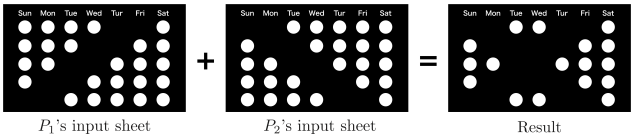


Fig. 3. Input sheet for set-intersection.



P_1's input sheet P_2's input sheet Result

Fig. 4. Example of set intersection.

Our SI Protocol

Let D be a set of possible dates with ℓ elements and assume that each participant P_i has a subset $x_i \subset D$ as input. A set-intersection function for n inputs over D takes n subsets $x_1, x_2, \ldots, x_n \subset D$ and outputs their intersection $x_1 \cap x_2 \cap \cdots \cap x_n$. Our set-intersection protocol is defined by the following tuple $(n, U_{\mathsf{SI}}, I_{\mathsf{SI}}, 2^D, 2^D, g_{\mathsf{SI}}, h_{\mathsf{SI}})$, where :

- The initial image I_{SI} is an image with $|D|$ holes (white circles). Figure 3 shows an example of I_{SI} with 35 holes, i.e, 5 weeks of 7 days, $\ell = 35 = 5 \times 7$.
- U_{SI} is the set of all images of the same size as the initial image I_{SI}.
- The domain and the range are 2^D, i.e., the set of all subsets of D.
- The input function g_{SI} takes an input $x \in 2^D$ and outputs a set of black circles corresponding to $D \setminus x$. For example, if it holds $D \setminus x = \{1, 2, 3\}$, $g_{\mathsf{SI}}(x)$ is the set of three black circles corresponding to $\{1, 2, 3\}$.
- The output function h_{SI} is the inverse function of g'_{SI}, where $g'_{\mathsf{SI}}(x) = g_{\mathsf{SI}}(x) + I_{\mathsf{SI}}$; it returns a set of elements corresponding to white circles.

SI protocol proceeds as follows:

1. Each player has a black pen and the input sheet.
2. Using a black pen, each player P_i fills a set of holes which corresponds to $D \setminus x_i$. Namely, he/she fills a hole corresponding to d with a black pen if and only if $d \notin x_i$. It means that he/she is not avaible at the date x_i.
3. Each player puts his/her input sheet into the hiding box.
4. The output is obtained by switching on the light. Namely, lighting holes are a set of holes corresponding to $x_1 \cap x_2 \cap \cdots \cap x_n$.

Figure 4 shows an example execution of our set-intersection protocol for two players. White circles in the input sheet of each of P_1 and P_2 are available dates for the party. The function g_{SI} returns a set of black circles corresponding to unavailable dates. The rightmost image is the resulting image I_{result}, which has a set of white circles corresponding to available dates for both parties.

The correctness holds from the fact that the resulting image I_{result} has a set of holes corresponding to the dates available for all players. The security holds from the fact that each result image is unique for each output value.

Fine-grained Scheduling Protocol

The situation is almost the same as that of set-intersection but now each participant has a yes/no/maybe schedule.

Let D be the set of possible dates with ℓ elements. Now each P_i has an input $x_i = (x_i^{(1)}, \ldots, x_i^{(\ell)})$ with $x_i^{(j)} \in \{\text{“yes”}, \text{“no”}, \text{“maybe”}\}$. The output function that they wish to compute is $y = (y^{(1)}, \ldots, y^{(\ell)})$ defined as follows:

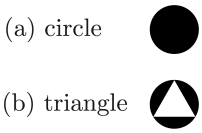


Fig. 5. Input circles.

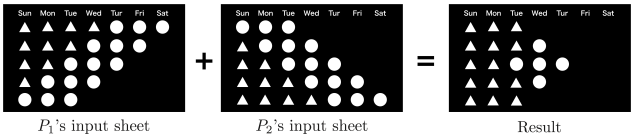


Fig. 6. Execution of fine-grained scheduling protocol.

- $y^{(j)} =$ "yes" if $x_i^{(j)} =$ "yes" for all $i \in \{1, 2, \ldots, n\}$,
- $y^{(j)} =$ "maybe" else if $x_i^{(j)} \in \{$"yes", "maybe"$\}$ for all $i \in \{1, 2, \ldots, n\}$,
- $y^{(j)} =$ "no" otherwise.

Our fine-grained scheduling protocol is defined by $(n, U_{\mathsf{FS}}, I_{\mathsf{FS}}, X, X, g_{\mathsf{FS}}, h_{\mathsf{FS}})$:

- U_{FS} and I_{FS} are the same as U_{SI} and I_{SI} of our set-intersection protocol, respectively.
- The domain and the range are $X = \{$"yes", "no", "maybe"$\}^{\ell}$, i.e., the list of ℓ elements of "yes"/"no"/"maybe".
- The input function g_{FS} takes an input $x_i = (x_i^{(1)}, x_i^{(2)}, \ldots, x_i^{(\ell)})$ and outputs a set of (at most ℓ) black circles, where at the j-th position, a circle is put if $x_i^{(j)} =$ "no", a circle with triangle is put if $x_i^{(j)} =$ "maybe" (see Figure 5).
- The output function h_{FS} is the inverse function of g'_{FS}, where $g'_{\mathsf{FS}}(x) = g_{\mathsf{FS}}(x) + I_{\mathsf{FS}}$.

Figure 6 shows an example execution of our fine-grained scheduling protocol for two players when $\ell = 35$. White circles and triangles in the input sheet of P_1 and P_2 are "yes" dates and "maybe" dates for the party. The rightmost image is the resulting image I_{result}.

The correctness and the security hold from a similar observation to the case of our set-intersection protocol.

4 Min/Max Protocol

Suppose that n students wish to know the highest score among all the students in an examination without revealing each score to other students.

Our Max Protocol

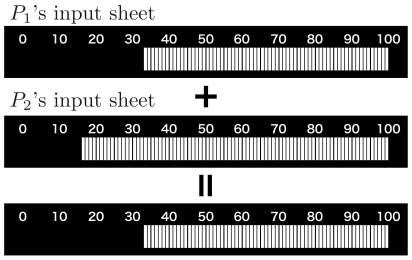
Let $\ell \in \mathbb{N}$ be the highest score (e.g., $\ell = 100$) and let $X = \{0, 1, 2, \ldots, \ell\}$ be the input domain. Assume that each student P_i has a score $x_i \in X$ as input. The function they wish to compute is the max (resp. min) function that takes n integers $x_1, x_2, \ldots, x_n \in X$ as input and outputs the maximum number $\max(x_1, x_2, \ldots, x_n) \in X$ (resp. the minimum number $\min(x_1, x_2, \ldots, x_n) \in X$).

Our max protocol is defined by $(n, U_{\max}, I_{\max}, X, X, g_{\max}, h_{\max})$, where:

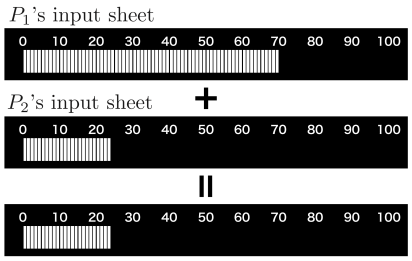
- $U_{\max}$ is the set of all black-and-white images of the same size as the one in Figure 7.
- The initial image $I_{\max}$ is given in Figure 7.
- The input function $g_{\max}$ takes an input $x \in X$ and outputs a black rectangle from 0 to x.

Fig. 7. Input sheet for min/max.

Fig. 8. Input sheet for max with name.



(a) Max protocol when $x_1 = 33$ and $x_2 = 16$



(b) Min protocol when $x_1 = 70$ and $x_2 = 24$

Fig. 9. Executions of max/min protocol.

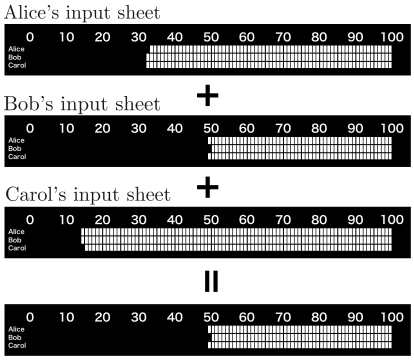


Fig. 10. Execution of max with name protocol when $(x_1, x_2, x_3) = (33, 50, 15)$.

– The output function $h_{\max}$ is the inverse function of $g'_{\max}$, where $g'_{\max}(x) = g_{\max}(x) + I_{\max}$; it outputs the value of the scale at the boundary between black and white.

Figure 9 (a) shows an example execution of our max protocol when $x_1 = 33$ and $x_2 = 16$. The function $g_{\max}$ with the input $x_1 = 33$ (resp. $x_2 = 16$) returns a black rectangle from 0 to 33 (resp. 16). Because the output image has a rectangle from 0 to 33, the output value is 33; it is correct because $33 = \max(33, 16)$.

The correctness holds from the above observation. The security holds from the fact that each resulting image is unique for each output value.

Our Min Protocol Our min protocol is obtained by the analogy with our max protocol. Figure 9 (b) shows an example execution of our min protocol when $x_1 = 70$ and $x_2 = 24$.

Max with Name Protocol

The situation is almost the same as that of max but now there are three students Alice, Bob, and Carol having $x_1, x_2, x_3 \in \{0, 1, \ldots, 100\}$, respectively, and they want to know the owner of the maximum together with the value of maximum.

Let $N = \{A, B, C\}$ be a set of names and suppose that Alice, Bob, and Carol have $(A, x_1), (B, x_2)$, and (C, x_3). The function that they want to compute is $\mathsf{MN} : (N \times X)^n \to 2^N \times X$ that takes the inputs as above and outputs $(S, \max(x_1, x_2, x_3))$, where $S \subset 2^N$ is a set of all names who have the maximum value. For instance, if $x_1 = 10$ and $x_2 = x_3 = 20$, then $S = \{B, C\}$.

Our protocol is defined by $(n, U_{\mathsf{MN}}, I_{\mathsf{MN}}, N \times X, 2^N \times X, g_{\mathsf{MN}}, h_{\mathsf{MN}})$, where:

- U_{MN} is the set of all black-and-white images of the same size as the one in Figure 8.
- The initial image I_{MN} is given in Figure 8. There are three lines, each of which corresponds to Alice, Bob, and Carol, respectively.
- The input function g_{MN} is defined as follows: for an input $(\mathsf{name}, x) \in N \times X$, the output $g_{\mathsf{MN}}(\mathsf{name}, x)$ is three black rectangles from 0 to x in the name's line and 0 to $x - 1$ in other two lines.
- The output function h_{MN} is the inverse function of g'_{MN}, where $g'_{\mathsf{MN}} : 2^N \times X \to U_{\mathsf{MN}}$ takes a set $S = \{\mathsf{n}_1, \ldots, \mathsf{n}_k\}$ and $x \in X$ for $k \leq 3$ as input and outputs the union of the images $g_{\mathsf{MN}}(\mathsf{n}_1, x) + \cdots + g_{\mathsf{MN}}(\mathsf{n}_k, x) + I_{\mathsf{MN}}$.

Figure 10 shows an example execution of our max with name protocol when $x_1 = 33$, $x_2 = 50$, and $x_3 = 15$. The function g_{MN} with the input (A, x_1) returns a black rectangle from 0 to 33 in Alice's line and a black rectangle from 0 to 32 in others' lines. Because the output image has a rectangle from 0 to 50 in Bob's line and a rectangle from 0 to 49 in others' lines, the output value is $(\{B\}, 33)$; it is correct because $(\{B\}, 33) = \mathsf{MN}((A, 33), (B, 50), (C, 15))$.

The correctness and the security hold from a similar observation to the case of our max protocol.

5 Extension to Randomized Input

Our model defined in Section 2.3 does not allow to use randomness in computation, thus all of our protocols are deterministic. Of course, deterministic is a nice property in order to make protocols simple, but randomness enables us to construct protocols for a larger class of functions. In this section, we extend our model to the randomized input setting.

5.1 Defining Protocols in the Randomized Input Setting

In the randomized input setting, a protocol is defined as in the same in Section 2.3 except that the deterministic input function $g(x_i)$ is replaced by a randomized input function $g(x_i; r_i)$ that takes a random coin $r_i \in \{0, 1\}^*$ together with an input $x_i \in X$. In the following, we define a relaxed variant of correctness and a randomized version of the security.

p-Correctness. A protocol is p-correct if for any $x_1, x_2, \ldots, x_n \in X$, the value $h(I_{\mathsf{result}})$ of the output function h for the union image $I_{\mathsf{result}} = I'_1 + I'_2 + \cdots + I'_n$ is equal to $f(x_1, x_2, \ldots, x_n)$ with a probability of at least p, where $I'_i = I + g(x_i)$ for all $i \in \{1, 2, \ldots, n\}$.

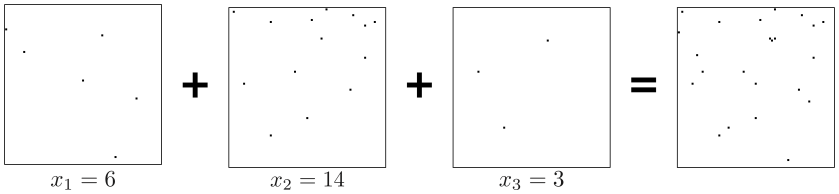


Fig. 11. Example of addition.

Simulation Security. A protocol is secure if there exists a probabilistic polynomial-time simulator $\mathcal{S}$ such that the following two random variables are equal:

- A random variable I_{result} over U: it is generated honestly when all parties' random tapes are chosen uniformly and independently at random.
- A random variable I_{sim} over U: it is generated by $\mathcal{S}$ given only the output value $f(x_1, x_2, \ldots, x_n)$.

5.2 Addition Protocol

Suppose that n students wish to know the average score among them without revealing each score to other students. Assume that each student P_i has a score $x_i \in \{0, 1, 2 \ldots, \ell\}$ as input. The function they wish to compute is the summation among them $x_1, x_2, \ldots, x_n$. (Note that the average can be obtained from the summation just by dividing by n.)

Our addition protocol is defined by $(n, U_{\mathsf{AD}}, I_{\mathsf{AD}}, X, X, g_{\mathsf{AD}}, h_{\mathsf{AD}})$, where:

- U_{AD} is the set of all square images of a fixed size with $M \times M$ pixels for some integer M.
- The initial image I_{AD} is just a white image.
- The input domain is $X = \{0, 1, \ldots, \ell\}$.
- The input function $g_{\mathsf{AD}}(x; r)$ takes a value $x \in X$ with a random coin r and outputs a set of x pixels chosen uniformly random by the use of r.
- The output function h_{AD} is the inverse function of $g_{\mathsf{AD}}(\cdot, r)$, where r is a fixed value; it just counts the number of pixels.

Figure 11 shows an example execution of our addition protocol when $x_1 = 6$, $x_2 = 14$, and $x_3 = 3$. The function g_{AD} with the input $x_1 = 6$ (resp. $x_2 = 14$ or $x_3 = 3$) returns a set of 6 (resp. 14 or 3) black pixels randomly. The total number of points in the resulting image is 23.

Our addition protocol has $(1 - \delta)$-correctness if it holds $\frac{(n\ell)^2}{2M^2} < \delta$.

Proof. Let coll_i be the event that at least one collision exists among $\{P_1, P_2, \cdots, P_i\}$'s input sheets ($i \in \{1, 2, \ldots, n\}$). From a simple observation, we have

$$\Pr[\mathsf{coll}_n] = \Pr[\mathsf{coll}_2] + \Pr[\mathsf{coll}_3 \mid \neg\mathsf{coll}_2] + \Pr[\mathsf{coll}_4 \mid \neg\mathsf{coll}_3] + \cdots + \Pr[\mathsf{coll}_n \mid \neg\mathsf{coll}_{n-1}].$$

For a fixed ℓ points in P_1's sheet, the probability that a P_2's single point colludes one of the P_1's points is exact $\frac{\ell}{M^2}$. By union bound, we have

$$\Pr[\mathsf{coll}_2] \leq \frac{\ell^2}{M^2} \quad \text{and} \quad \Pr[\mathsf{coll}_{k+1} \mid \neg\mathsf{coll}_k] \leq \frac{k\ell^2}{M^2}.$$

Thus, combining them and the assumption, we have

$$\Pr[\mathsf{coll}_n] \leq \frac{\ell^2}{M^2} + \frac{2\ell^2}{M^2} + \cdots + \frac{(n-1)\ell^2}{M^2} \leq \frac{(n\ell)^2}{2M^2} < \delta.$$

This completes the proof. □

In order to prove the security, we have to construct a simulator $\mathcal{S}$ that can generate a simulated resulting image I_{sim}. Given an output value $y = x_1 + x_2 + \cdots + x_n$, $\mathcal{S}$ draws a set of random y pixels on the input sheet and sets it as simulated resulting image I_{sim}. Because it is the same distribution as the original resulting image I_{result}, we can conclude that our addition protocol is secure.

Idea for Enhancing Correctness

Using a circle with line "⊘" instead of a pixel "·" improves the probability of the correctness. Now a random coin is used to determine a position of a circle and an angle of the line. If there are L possible angles, then the probability p of the p-correctness is roughly $1 - \frac{\delta}{L}$. (We can not place the center of the circle ⊘ at the edge of input sheets, so the number of possible "positions" is reduced a little. However, this is not a problem because the "angle" effect is much greater.)

6 Conclusion

In this paper, we introduce a new physical cryptography, light cryptography, which uses the property of light. We also give some simple protocols in order to solve some problems that are complex to solve and to understand using classical cryptography. These examples can be explained to non-experts and clearly show the power of our light and shadow model. The next future step is to design more protocols and also to provide practical lecture material for information security education and use it in our cryptography courses.

Acknowledgments: This work was supported in part by JSPS KAKENHI Grant Numbers 17J01169 and 17K00001.

References

1. J. Balogh, J. A. Csirik, Y. Ishai, and E. Kushilevitz. Private computation using a PEZ dispenser. *Theor. Comput. Sci.*, 306(1-3):69–84, 2003.
2. M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pp. 1–10, 1988.

3. X. Bultel, J. Dreier, J. Dumas, and P. Lafourcade. Physical zero-knowledge proofs for akari, takuzu, kakuro and kenken. In *8th International Conference on Fun with Algorithms, FUN 2016*, Vol. 49 of *LIPIcs*, pp. 8:1–8:20, 2016.
4. X. Bultel, J. Dreier, J. Dumas, P. Lafourcade, D. Miyahara, T. Mizuki, A. Nagao, T. Sasaki, K. Shinagawa, and H. Sone. Physical zero-knowledge proof for makaro. In *Stabilization, Safety, and Security of Distributed Systems - 20th International Symposium, SSS 2018*, Vol. 11201, pp. 111–125. Springer, 2018.
5. C. Crépeau and J. Kilian. Discreet solitary games. Vol. 773 of *Lecture Notes in Computer Science*, pp. 319–330. Springer, 1994.
6. P. D'Arco and R. D. Prisco. Secure computation without computers. *Theor. Comput. Sci.*, 651:11–36, 2016.
7. B. den Boer. More efficient match-making and satisfiability: *The Five Card Trick*. In J. Quisquater and J. Vandewalle eds., *Advances in Cryptology - EUROCRYPT '89, Workshop on the Theory and Application of Cryptographic Techniques, Houthalen, Belgium*, Vol. 434, pp. 208–217. Springer, Apr. 1989.
8. J. Dreier, H. Jonker, and P. Lafourcade. Secure auctions without cryptography. In *Fun with Algorithms - 7th International Conference, FUN 2014, Lipari Island, Sicily, Italy, July 1-3, 2014. Proceedings*, Vol. 8496, pp. 158–170. Springer, 2014.
9. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987*, pp. 218–229, 1987.
10. V. Kolesnikov. Gate evaluation secret sharing and secure one-round two-party computation. In B. K. Roy ed., *Advances in Cryptology - ASIACRYPT 2005, 11th International Conference on the Theory and Application of Cryptology and Information Security, Chennai, India, December 4-8, 2005, Proceedings*, Vol. 3788 of *Lecture Notes in Computer Science*, pp. 136–155. Springer, 2005.
11. T. Mizuki, Y. Kugimoto, and H. Sone. Secure multiparty computations using a dial lock. In J. Cai, S. B. Cooper, and H. Zhu eds., *Theory and Applications of Models of Computation, 4th International Conference, TAMC 2007, Shanghai, China*, Vol. 4484 of *Lecture Notes in Computer Science*, pp. 499–510. Springer, May 2007.
12. T. Mizuki, Y. Kugimoto, and H. Sone. Secure multiparty computations using the 15 puzzle. In A. W. M. Dress, Y. Xu, and B. Zhu eds., *Combinatorial Optimization and Applications, First International Conference, COCOA 2007, Xi'an, China*, Vol. 4616 of *Lecture Notes in Computer Science*, pp. 255–266. Springer, Aug. 2007.
13. T. Mizuki and H. Sone. Six-card secure AND and four-card secure XOR. In X. Deng, J. E. Hopcroft, and J. Xue eds., *Frontiers in Algorithmics, Third International Workshop, FAW 2009, Hefei, China, June 20-23, 2009. Proceedings*, Vol. 5598 of *Lecture Notes in Computer Science*, pp. 358–369. Springer, 2009.
14. M. Naor and A. Shamir. Visual cryptography. In A. D. Santis ed., *Advances in Cryptology - EUROCRYPT '94, Workshop on the Theory and Application of Cryptographic Techniques, Perugia, Italy, May 9-12, 1994, Proceedings*, Vol. 950 of *Lecture Notes in Computer Science*, pp. 1–12. Springer, 1994.
15. K. Shinagawa, T. Mizuki, J. C. N. Schuldt, K. Nuida, N. Kanayama, T. Nishide, G. Hanaoka, and E. Okamoto. Secure computation protocols using polarizing cards. *IEICE Transactions*, 99-A(6):1122–1131, 2016.
16. A. C. Yao. Protocols for secure computations. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, SFCS '82, pp. 160–164, Washington, DC, USA, 1982. IEEE Computer Society.
17. A. C. Yao. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*, pp. 162–167, 1986.