Secure two-party computation: a visual way

Paolo D'Arco and Roberto De Prisco
Dipartimento di Informatica
University of Salerno
Italy.

Abstract

In this paper we propose a novel method for performing secure two-party computation. By merging together in a suitable way two beautiful ideas of the 80's and the 90's, Yao's garbled circuit construction and Naor and Shamir's visual cryptography, respectively, we enable Alice and Bob to securely evaluate a function $f(\cdot, \cdot)$ of their inputs, x and y, through a *pure physical* process. Indeed, once Alice has prepared a set of properly constructed transparencies, Bob computes the function value $f(x, y)$ by applying a sequence of simple steps which require the use of a pair of scissors, superposing transparencies, and the human visual system. A crypto-device for the function evaluation process is not needed any more.

Keywords: Yao's Construction, Visual Cryptography, Secure Computation.

1 Introduction

Yao's Construction. Latins said: *Verba volant, scripta manent.* Yao's construction disproves the saying. Indeed, [22] and [23], the papers which usually are cited when the construction is used or referred to, do not contain any description of it. It has never been written down by the author, but only provided to the community during an oral presentation (FOCS 1986). Fortunately, *verba* were captured by other researchers, who used the construction in subsequent papers, first of all [13]. Later on, it has been widely exploited in protocol design, but, apart some notable exceptions, it has more or less been considered as a powerful tool for establishing existential results. However, in the last years, since it has been shown that fine-tuned implementations, for reasonable input sizes, are becoming practical in many settings, new attention has been devoted to it. A version of the construction has been clearly described and proved secure according to precise definitions and assumptions in [19]. In a few other new recently introduced cryptographic primitives and protocols, e.g., *functional encryption* [5] or *non-interactive verifiable computing* [14], the construction plays a key role, and in [3] it has been even proposed to move from a view of Yao's construction as a cryptographic tool to a view of the construction as a *cryptographic goal*, which can be achieved with several security properties and privacy degrees[1]. From a certain point of view, Yao's idea is living nowadays a sort of *second life*.

[1]The introduction of [3] offers a brief history of the construction and a nice accounting of the research efforts which followed.

Roughly speaking, Yao's construction, enables two parties, Alice and Bob, to privately evaluate a boolean function $f(\cdot, \cdot)$ on their inputs, x and y, in such a way that each party gets the result and, at the same time, *preserves* the privacy of its own input, apart from what can be inferred about it by the other party from its input and the function value $f(x, y)$, e.g., if the function $f(\cdot, \cdot)$ is the `xor` function, given x `xor` y there is no way to preserve the other input.

In a nutshell, the construction works as follows: the boolean function $f(\cdot, \cdot)$ is represented through a boolean circuit $C(\cdot, \cdot)$ for which, for each x, y, it holds that $C(x, y) = f(x, y)$. Yao's idea is to use the circuit as a *conceptual guide* for the computation which, instead of a sequence of `and`, `or` and `not` operations on strings of bits x and y, becomes a *sequence of decryptions* on sequences of ciphertexts. More precisely, one of the party, say Alice, given $C(\cdot, \cdot)$, computes a new object $\tilde{C}$, which is usually referred to as the *garbled circuit* [1], where:

- to each wire w of $C(\cdot, \cdot)$, are associated in $\tilde{C}$ two random keys, k_w^0 and k_w^1, which (secretly, the correspondence is not public) represent 0 and 1, and,

- to each gate $G(\cdot, \cdot)$ of $C(\cdot, \cdot)$, corresponds in $\tilde{C}$ a *gate table* $\tilde{G}$ with four rows, each of which is a *double encryption*, obtained by using two different keys $k_{w_1}^a$ and $k_{w_2}^b$, for $a, b \in \{0, 1\}$, of a message which is itself a random key $k_{w_3}^c$, for $c \in \{0, 1\}$. In details, each double encryption $E_{ab}{=}E_{k_{w_2}^b}(E_{k_{w_1}^a}(k_{w_3}^c))$ uses *one of the four* possible pairs of keys $(k_{w_1}^a, k_{w_2}^b)$, associated to the input wires (w_1, w_2) of gate $G(\cdot, \cdot)$, and the message which is encrypted is the random key $k_{w_3}^c$, associated to the wire w_3 of output of the gate $G(\cdot, \cdot)$ *if and only if* $G(a, b) = c$. The four double encryptions E_{00}, E_{01}, E_{10} and E_{11} are stored in the gate table rows in *random* order.

Once $\tilde{C}$ has been computed, Alice sends to Bob all the gate tables $\tilde{G}$ associated to the circuit gates $G(\cdot, \cdot)$, and *reveals* the random keys k_w^0 and k_w^1, associated to all the *output* wires w, and their correspondences with the values 0 and 1. Moreover, for the input wires of the circuit, she sends to Bob the random keys $k_{w_1}^{x_1}, k_{w_2}^{x_2}, \ldots, k_{w_n}^{x_n}$ corresponding to the bit-values of her own input $x = x_1 x_2 \ldots x_n$. To perform the computation represented by $\tilde{C}$, then Bob needs only the keys associated to the input wires corresponding to *his own* input. This issue can be solved by means of *executions* of 1-out-of-2 *oblivious transfer* protocols [11], through which Bob receives the random keys $k_{w_{n+1}}^{y_1}, k_{w_{n+2}}^{y_2}, \ldots, k_{w_{2n}}^{y_{2n}}$ corresponding to the bit-values of his own input $y = y_1 y_2 \ldots y_n$ and nothing else, while Alice from the transfer does not know which specific keys Bob has recovered.

Finally Bob, according to the topology of the original circuit $C(\cdot, \cdot)$, level after level, decrypts *one and only one* entry from each gate table $\tilde{G}$ in $\tilde{C}$, until he computes *one and only one* random key associated to each output wire. The binary string which corresponds to the sequence of computed random keys, associated to the output wires, is the value $C(x, y)$. Bob sends the result of the computation to Alice[2].

It is easy to check that the computation is correct and, intuitively, that the privacy of the inputs is preserved. The random keys held by Bob, the rows of each $\tilde{G}$, and the random keys obtained decrypting a row in each $\tilde{G}$, do not leak any information about the actual bits of the input values.

Visual Cryptography. Visual cryptography is a special type of secret sharing in which the secret is an image and the shares are random-looking images printed on transparencies. It was introduced

[2]Appendix B provides a detailed description of Yao's protocol.

by Naor and Shamir [18]. The captivating peculiarity of this type of secret sharing is that the reconstruction of the secret is performed without any computational machinery: it is enough to superpose the shares (transparencies) in order to reconstruct the secret. Roughly speaking, for black-and-white images, the bit value 0 is encoded as a transparent pixel, the bit value 1 is encoded as a black pixel, and the reconstruction operation is an `or` and is performed by the human visual system when the shares are superposed. Visual cryptography has been extensively studied; we refer the interested reader to [9] for a collection of surveys on several aspects of visual cryptography. For the goal of this paper we will be using a particular type of visual cryptography: probabilistic visual cryptography [21, 10].

Our Contribution. In this paper we merge together Yao's construction and properly defined visual cryptography schemes, in order to propose a method through which Alice and Bob can securely evaluate a function $f(\cdot,\cdot)$ of their inputs, x and y, through a *pure physical* process.

Our efforts were inspired and driven by the work of Kolesnikov [17], who showed that a different approach to the function evaluation process in Yao's construction can be pursued. Roughly speaking, instead of constructing the garbled circuit $\tilde{C}$ by using for each gate $G(\cdot,\cdot)$ a gate table $\tilde{G}$, containing a double encryption for each possible input pair of keys, it is possible to use *secret sharing schemes* designed to realize the functionalities implemented by the logical gates. Such schemes were referred to as *gate equivalent secret sharing schemes* (GESS, for short) [17]. Using a GESS, any time that two shares, say $sh^a_{w_1}$ and $sh^b_{w_2}$, associated to the input wires w_1 and w_2 of gate $G(\cdot,\cdot)$, are combined through the reconstruction function of the GESS, the secret s_{w_3}, associated to the output wire w_3 of gate $G(\cdot,\cdot)$ is recovered. It follows that an *explicit representation* $\tilde{G}$ of $G(\cdot,\cdot)$ is *not* needed any more, because all the information required to reconstruct the secret value associated to w_3, depending on the functionality of the target gate $G(\cdot,\cdot)$, is coded and, hence, *implicitly represented*, into the shares $sh^a_{w_1}$ and $sh^b_{w_2}$. Therefore, given the circuit $C(\cdot,\cdot)$, and by applying a bottom-up process, which starts from the circuit output wires and ends when the circuit input wires are reached, Alice can construct shares associated to the circuit input wires which encode *all the information* needed to evaluate $C(\cdot,\cdot)$ on every pair of inputs (x, y). Then, as in Yao's construction, Alice sends directly to Bob the shares corresponding to the bit-values of her own input x, while Bob, by means of *executions* of 1-out-of-2 *oblivious transfer* protocols, receives the shares corresponding to the bit-values of his own input y. Finally, Bob applies iteratively the GESS reconstruction functions, until the secrets associated to the output wires, which correspond to the value $C(x, y)$, are obtained. We provide a *generalization* of the above approach and a *visual implementation.*

Notice that, the technique used by Kolesnikov [17], does not immediately extend to visual secret sharing. In order to exploit visual secret sharing, some technical details and issues need to be addressed. The most important ones are two: (*i*) we need to define and construct a *visual counterpart* of a GESS scheme, and (*ii*) propose a physical method to perform the oblivious transfer. Both of them are goals of independent interests. We show that the GESS construction provided in [17] is a *special case* of a general construction which uses multi-secret sharing schemes. Therefore, it can be instantiated by using a visual multi-secret sharing scheme. We also provide a construction. Regarding the oblivious transfer, even if physical metaphors have often been used for describing cryptographic primitives and protocols, only few papers have dealt with physical implementations. To our knowledge, the state of the art is summarized in [20], which is the first paper that rigorously addresses the issue of realizing cryptographic protocols by using tamper-evident seals (sealed envelopes and locked boxes). We could use an oblivious transfer protocol of [20], but since we discuss a simpler scenario, we propose an easier construction which uses *indistinguishable envelopes*. The main result we achieve can be (informally) stated as follows:

Theorem 1.1 *Every two-party computation representable by means of a boolean function $f(\cdot,\cdot)$ can be performed preserving the privacy of the inputs x and y through a pure physical visual evaluation process.*

2 Definitions and Tools

Let us start by setting up the notation and stating basic definitions. We follow essentially the treatment of [19, 12] (i.e., see Section 2 of [19] or Chapter 7 of [12]).

2.1 Notation

Efficient Algorithms. An efficient algorithm is a probabilistic algorithm running in $poly(k)$ time, where k is a security parameter. Efficient algorithms are referred to as *PPT algorithms.*
Negligible functions. A function $f(\cdot)$ is negligible if it vanishes faster than the inverse of any fixed positive polynomial. That is, for any positive integer c, there exists an integer k_0 such that $f(k) \leq \frac{1}{k^c}$, for any $k \geq k_0$. We denote by $negl(k)$ a negligible function.
Algorithms and random variables. If $A(\cdot)$ is a probabilistic algorithm, then, for any x, the notation $A(x)$ refers to the random variable that assigns to the string σ the probability that A, on input x, outputs σ.
Distribution Ensembles. If S is an infinite set, and $X = \{X_s\}_{s\in S}$ and $Y = \{Y_s\}_{s\in S}$ are distribution ensembles[3], then we say the X and Y are *identically distributed*, $X \overset{p}{\equiv} Y$ for short, if, for every distinguisher D and for every $s \in S$, it holds that $|Pr[D(X_s) = 1] - Pr[D(Y_s) = 1]|$ is equal to 0. Similarly, if the Ds are PPT algorithms, and for all sufficiently large (in the length of the parameter) $s \in S$ it holds that $|Pr[D(X_s) = 1] - Pr[D(Y_s) = 1]|$ is a negligible function $negl(s)$ in s, we say that X and Y are *computationally indistinguishable*, $X \overset{c}{\equiv} Y$ for short.

2.2 Secure Two-party Computation

We consider two-party computation in presence of a *static semi-honest* adversary. The adversary controls one of the parties and, although it follows the protocol specification, it might try to learn extra information from the transcript of the messages received during the execution.

A two-party computation is a random process that maps pairs of inputs to pairs of outputs, one for each party. We refer to such a process as a *functionality* and denote it $f : \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}^* \times \{0,1\}^*$, where $f(x, y) = (f_1(x, y), f_2(x, y))$.

Let π be a two-party protocol for computing f. Intuitively, a protocol is secure if whatever a party can compute participating in the protocol can also be computed by himself by using *only* his

[3]A random variable is sufficient to represent the input, the output or any intermediate computation of a randomized entity in a *single* protocol execution. However, since it is of interest analyzing the behavior of protocol executions, according to input sizes depending on the security parameter k, *collections* of random variables are needed: an ensemble is exactly a family of random variables, where each of them, say X_s, is uniquely identified by an index s, related to the security parameter k.

own input and his own function value. More precisely, for $i \in \{1,2\}$, denoting with the random variables $view_i^\pi(x,y)$, the view (i.e., input, random coins, messages received...) that party i has during the execution of $\pi(x,y)$, by $output_i^\pi(x,y)$ the output of party i, and by $output^\pi(x,y)$ the output of both parties, we state the following[4]:

Definition 2.1 *Let f be a functionality. A protocol π computes f in a perfectly (computationally) secure way, in presence of a static semi-honest adversary, if*

$$\{output^\pi(x,y)\}_{(x,y)\in\{0,1\}^*} = \{f(x,y)\}_{(x,y)\in\{0,1\}^*}$$

and there exists (PPT) algorithms Sim_1 and Sim_2 such that:

$$\{Sim_1(x, f_1(x,y))\}_{(x,y)\in\{0,1\}^*} \overset{p/c}{\equiv} \{view_1^\pi(x,y)\}_{(x,y)\in\{0,1\}^*},$$

$$\{Sim_2(y, f_2(x,y))\}_{(x,y)\in\{0,1\}^*} \overset{p/c}{\equiv} \{view_2^\pi(x,y)\}_{(x,y)\in\{0,1\}^*}.$$

3 Visual Gate Evaluation Secret Sharing

In this section, building on the definitions and the constructions provided in [17], we introduce the notion of *visual* gate evaluation secret sharing (VGESS, for short), and we show how to construct a VGESS scheme. We proceed as follows: (i) we recall some notions on visual secret and multi-secret sharing schemes, (ii) we recall the definition of GESS schemes [17], (iii) we define a *general* construction for GESS schemes, GenGESS for short, in terms of multi-secret sharing schemes[5]. The construction in [17] ends up to be a special instance of it. Finally, in order to take benefits from the general form, we define VGESS and, by using a visual multi-secret sharing scheme, we realize an implementation.

3.1 Visual Cryptography

Given two images I_1 and I_2, with the same size, printed on transparencies, we denote with $\mathtt{Sup}(I_1, I_2)$ the image that results from the superposition of the two images. Interpreting white as 0 and black as 1, for each pixel position (i,j), we have that $\mathtt{Sup}(I_1, I_2) = I_1(i,j)$ $\mathtt{or}$ $I_2(i,j)$.

Let us start with the definition[6] of a particular type of visual secret sharing scheme for a set $\mathcal{P} = \{1,2\}$ of two parties, with access structure defined by $\mathcal{A} = \{\{1,2\}\}$ and $\mathcal{F} = \{\{1\},\{2\}\}$ and no pixel expansion. Such schemes are referred to as probabilistic $(2,2)$-VCS with no pixel expansion.

[4]We deal in the following with a deterministic functionality. Hence, we state the simplified versions of the definitions in [19, 12]. Moreover, we also state the definition for the unconditionally secure case. As we will show later, by using an unconditionally secure *physical* implementation of the oblivious transfer, known to be possible [20], the definition in the *physical* world is achieved by our protocol.

[5]In Appendix C the reader can find general definitions and notions for secret sharing and multi-secret sharing schemes, if he is not familiar with the subject.

[6]In this abstract, to simplify the presentation of our approach, instead of providing general definitions, we concentrate on specific definitions of VCS for the tools we need in our construction.

Definition 3.1 Probabilistic $(2,2)$-VCS with no pixel expansion. *Let S be a set of secret images, such that $|S| \geq 2$. A probabilistic $(2,2)$-VCS with no pixel expansion is a secret sharing scheme realizing the access structure defined by $\mathcal{A} = \{\{1,2\}\}$ and $\mathcal{F} = \{\{1\},\{2\}\}$ where Shr and Rec are such that*

- *Shr is a probabilistic algorithm which takes in input a secret $I \in S$ and outputs a pair of visual shares (sh_1, sh_2)*
- *Rec is the deterministic algorithm* $\texttt{Sup}(\cdot,\cdot)$ *which superposes sh_1 to sh_2*

satisfying the following properties:

- **Correctness***: For each pixel position (i,j), if $I(i,j) = \bullet$ then $\texttt{Sup}(sh_1, sh_2)(i,j) = \bullet$, and if $I(i,j) = \circ$ then $pr[\texttt{Sup}(sh_1, sh_2)(i,j) = \circ] > 0$.*
- **Privacy***: For each pixel position (i,j), regardless of the values of $I(i,j)$, $pr[sh_1(i,j) = \circ] = pr[sh_2(i,j) = \circ]$, and, consequently, $pr[sh_1(i,j) = \bullet] = pr[sh_2(i,j) = \bullet]$.*

Notice that, in the above definition we require that black pixels are reconstructed perfectly. In general, VCSs can be implemented by means of *distribution matrices*. Precisely, let n and m be two integers, where n represents the number of parties and m the pixel expansion, i.e., the parameter that specifies how many sub-pixels are needed in each share to encode a single pixel of the secret image. A scheme is usually defined by two collections $\mathcal{C}_\circ$ and $\mathcal{C}_\bullet$ of $n \times m$ matrices with elements in $\{\circ, \bullet\}$. The *Shr* algorithm, for each secret pixel, chooses a distribution matrix M at random from $\mathcal{C}_\circ$, if the secret pixel is white, or from $\mathcal{C}_\bullet$, if the secret pixel is black, and uses row i of M to construct the pixel on the i-th share. For example, the following collections of distribution matrices can be used to realize a probabilistic $(2,2)$-VCS with no pixel expansion ($m = 1$):

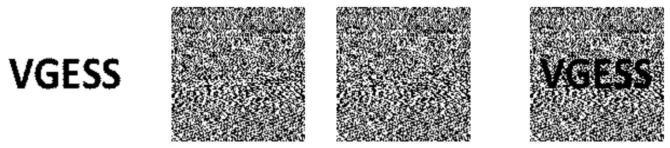
$$\mathcal{C}_\circ = \left\{ \begin{bmatrix} \circ \\ \circ \end{bmatrix}, \begin{bmatrix} \bullet \\ \bullet \end{bmatrix} \right\} \qquad \mathcal{C}_\bullet = \left\{ \begin{bmatrix} \circ \\ \bullet \end{bmatrix}, \begin{bmatrix} \bullet \\ \circ \end{bmatrix} \right\}$$

More precisely, assuming that the set S of secret images contains all black-and-white square images I of $n \times n$ pixels, and that $R = \{0,1\}$, denoting the distribution matrices in $\mathcal{C}_\circ$ as $\mathcal{C}_{\circ,0}, \mathcal{C}_{\circ,1}$, and in $\mathcal{C}_\bullet$ as $\mathcal{C}_{\bullet,0}, \mathcal{C}_{\bullet,1}$, a probabilistic $(2,2)$-VCS with no pixel expansion, can be realized as follows:

```
Probabilistic $(2,2)$-VCS
---
*Shr*$(I)$
For every $i,j = 1,\ldots,n$,
  Choose uniformly at random $r_{i,j} \in R = \{0,1\}$
  Use $\mathcal{C}_{I(i,j),r_{i,j}}$ as distribution matrix for $sh_1(i,j)$ and $sh_2(i,j)$.
Output $(sh_1, sh_2)$
---
*Rec*$(sh_1, sh_2)$
Return $I = \texttt{Sup}(sh_1, sh_2)$.
```

An example of application of the scheme is given in Figure 1.

It is easy to see that the Probabilistic $(2,2)$-VCS satisfies Definition 3.1. More precisely:



Secret image Share 1 Share 2 Superposition of shares 1 and 2

Figure 1: Example of shares and superposition for a probabilistic $(2,2)$-scheme with $m = 1$.

Theorem 3.2 *The* Probabilistic $(2,2)$-VCS *construction realizes a probabilistic* $(2,2)$-*VCS with no pixel expansion.*

We also remark that scheme Probabilistic $(2,2)$-VCS is the same as the random grid scheme of Kafri and Keren [15].

Let us now define a 2-MVCS with no pixel expansion, i.e., a visual multi-secret sharing scheme for a set $\mathcal{P} = \{1,2,3\}$ of three parties, with access structures defined by $\mathcal{A}_1 = \{\{1,2\}\}, \mathcal{F}_1 = \{\{1\},\{2\},\{3\}\}$ and $\mathcal{A}_2 = \{\{1,3\}\}, \mathcal{F}_2 = \{\{1\},\{2\},\{3\}\}$. The scheme will be used to share 2 secret images I_0 and I_1 which will be reconstructed, respectively, by $\mathcal{A}_1$ and $\mathcal{A}_2$.

Definition 3.3 Probabilistic 2-MVCS with no pixel expansion. *Let S be a set of secret images, such that $|S| \geq 2$. A probabilistic 2-MVCS is a multi-secret sharing scheme with domains $S_1 = S_2 = S$ realizing the access structure defined by $\mathcal{A}_1 = \{\{1,2\}\}, \mathcal{F}_1 = \{\{1\},\{2\},\{3\}\}$ and $\mathcal{A}_2 = \{\{1,3\}\}, \mathcal{F}_2 = \{\{1\},\{2\},\{3\}\}$, where Shr and Rec are such that*

- *Shr is a probabilistic algorithm which takes in input two secret images $I_0 \in S$ and $I_1 \in S$ and outputs three visual shares (sh_1, sh_2, sh_3).*
- *Rec is the deterministic algorithm* $\mathtt{Sup}(\cdot,\cdot)$ *which superposes a pair of shares.*

satisfying the following properties:

- **Correctness***: For $h = 0,1$, for each pixel position (i,j), if $I_h(i,j) = \bullet$, then $\mathtt{Sup}(sh_1, sh_{2+h})(i,j) = \bullet$, and if $I_h(i,j) = \circ$, then $pr[\mathtt{Sup}(sh_1, sh_{2+h})(i,j) = \circ] > 0$.*
- **Privacy***: For each pixel position (i,j), $pr[sh_1(i,j) = \circ] = pr[sh_2(i,j) = \circ] = pr[sh_3(i,j) = \circ]$, and, consequently, $pr[sh_1(i,j) = \bullet] = pr[sh_2(i,j) = \bullet] = pr[sh_3(i,j) = \bullet]$.*

Notice that the definition does not state any requirement for the superposition of sh_2 and sh_3, that is we neither require a reconstruction nor an assurance of no information leakage for the combination of the two shares: we simply don't care as in our application they will never appear at the same time.

By using as building block the Probabilistic $(2,2)$-VCS, and thus the same collections of distribution matrices $\mathcal{C}_\circ, \mathcal{C}_\bullet$, a Probabilistic 2-MVCS can be realized as follows:

```
Probabilistic 2-MVCS

Shr(I_0, I_1)
For every $i, j = 1, \ldots, n$,
    Choose uniformly at random $r_{i,j} \in R = \{0,1\}$
    Use $\mathcal{C}_{I_0(i,j), r_{i,j}}$ as distribution matrix for $sh_1(i,j)$ and $sh_2(i,j)$
    If $I_1(i,j) = \bullet$ then
        if $sh_1(i,j) = \circ$ then $sh_3(i,j) = \bullet$
        if $sh_1(i,j) = \bullet$ then $sh_3(i,j) = \circ$
    If $I_1(i,j) = \circ$ then
        if $sh_1(i,j) = \bullet$ then $sh_3(i,j) = \bullet$
        if $sh_1(i,j) = \circ$ then $sh_3(i,j) = \circ$
Output $(sh_1, sh_2, sh_3)$

Rec(sh_i, sh_j)
Return $I = \texttt{Sup}(sh_i, sh_j)$.
```

It is easy to see that the Probabilistic 2-MVCS satisfies Definition 3.3. More precisely:

Theorem 3.4 *The* Probabilistic 2-MVCS *construction realizes a probabilistic 2-MVCS with no pixel expansion.*

3.2 GESS: Definition

At this point, we recall the definition of a GESS scheme given in [17]. Let us define a *selector* v as a pair of bits, that is $v \in V^2 = \{0,1\} \times \{0,1\}$. A *selection function Sel* takes in input a pair of pairs and a selector, and selects one element from each of the two pairs, according to the selector. More precisely, *Sel* is defined as $Sel : (((a_0, a_1), (b_0, b_1)), (v_1, v_2)) \to (a_{v_1}, b_{v_2})$.

Given a gate G and a selector $v = (v_1, v_2)$, we denote with $G(v)$ the output of gate G on input (v_1, v_2).

Definition 3.5 *A gate evaluation secret sharing scheme for gate G is a pair of algorithms (Shr,Rec) such that*

- *Shr is a probabilistic algorithm which takes in input two secrets $s_0 \in S$ and $s_1 \in S$ and outputs a tuple (t_1, t_2) where each t_i, for $i = 1, 2$, consists of two shares, i.e., $t_1 = (sh_{1,0}, sh_{1,1})$ and $t_2 = (sh_{2,0}, sh_{2,1})$*

- *Rec is a deterministic algorithm which takes in input two shares and outputs $s \in S$ or $\perp$*

satisfying the following conditions:

- **Correctness***: For each $s_0 \in S$ and $s_1 \in S$, and for any selector $v \in V^2$, it holds that $Rec(Sel(Shr(s_0, s_1), v)) = s_{G(v)}$.*

- **Privacy***: There exists a PPT algorithm Sim such that, for each $s_0 \in S$ and $s_1 \in S$, and for any selector $v \in V^2$, it holds that $Sim(s_{G(v)}) \overset{p}{\equiv} Sel(Shr(s_0, s_1), v)$.*

3.3 A General Construction for $GESS$

A $GESS$ for a gate G ($GESS_G$, for short) can be implemented by using a *2-MSSS* (see Appendix C) $\Sigma = (Shr_\Sigma, Rec_\Sigma)$. More precisely, we use two instances of Σ for a set of parties $\mathcal{P} = \{1, 2, 3\}$, denoted with the letters A and B to simplify the presentation [7]. Instance $A = (Shr_A, Rec_A)$ and instance $B = (Shr_B, Rec_B)$, with $Shr_A = Shr_B = Shr_\Sigma$ and $Rec_A = Rec_B = Rec_\Sigma$, have secret domains $S_1 = S_2 = \{s_0, s_1\}$, and both of them realize the pair of access structures defined by $\mathcal{A}_1 = \{\{1,2\}\}, \mathcal{F}_1 = \{\{1\},\{2\},\{3\}\}$ and $\mathcal{A}_2 = \{\{1,3\}\}, \mathcal{F}_2 = \{\{1\},\{2\},\{3\}\}$.

The construction is given in Table 1. In step 1, the two instances of Σ provide shares which reconstruct $s_{G(0,0)}$ and $s_{G(0,1)}$ (instance A) and $s_{G(1,0)}$ and $s_{G(1,1)}$ (instance B). Then, in step 2 the shares of A and B are viewed as sub-shares, and are rearranged and concatenated in order to construct shares which reproduce the functionality implemented by G. The random permutation bit b is used to hide the correspondence first-part/second-part of the share associated to the right wire and the secret which is reconstructed. Finally, in step 3, the shares for the wires of G are given in output.

Notice that the construction generalizes the construction given in [17]. Indeed, Kolesnikov's construction is a special case, where, assuming that the secrets s_0, s_1 are n-bit strings and R_0 and R_1 are also n-bit strings, chosen uniformly at random, the shares produced by the two instances of the 2-MSSS are $sh_1^A = R_0, sh_2^A = s_{G(0,0)}R_0, sh_3^A = s_{G(0,1)}R_0$, and $sh_1^B = R_1, sh_2^B = s_{G(1,0)}R_1, sh_3^B = s_{G(1,1)}R_1$, where R_0 and R_1 is the fresh randomness used by A and B, respectively, and the $Rec(\cdot,\cdot)$ function is the `xor` function.

We show now that the general construction for $GESS_G$ satisfies Definition 3.5. More precisely:

Theorem 3.6 *The* GenGESS *construction realizes a* $GESS_G$.

3.4 Visual GESS

Visual gate evaluation secret sharing schemes ($VGESS$, for short) are a visual realization of a GESS scheme. More precisely:

Definition 3.7 *A visual gate evaluation secret sharing scheme for gate G ($VGESS_G$, for short) is a pair of algorithms (Shr,Rec) such that*

- *Shr is a probabilistic algorithm which takes in input two secret images $I_0 \in S$ and $I_1 \in S$ and outputs a tuple (t_1, t_2) where each t_i, for $i = 1, 2$, consists of two visual shares, i.e., $t_1 = (sh_{1,0}, sh_{1,1})$ and $t_2 = (sh_{2,0}, sh_{2,1})$*
- *Rec is the deterministic algorithm* $\texttt{Sup}(\cdot,\cdot)$ *which superposes a pair of shares.*

satisfying the following conditions:

- **Correctness***: For each $I_0 \in S$ and $I_1 \in S$, and for any selector $v \in V^2$, it holds that, for each pixel position (i, j), if $I_{G(v)}(i, j) = \bullet$, then $\texttt{Sup}(Sel((Shr(I_0, I_1), v))(i, j) = \bullet$, and if $I_{G(v)}(i, j) = \circ$, then $pr[\texttt{Sup}(Sel((Shr(I_0, I_1), v))(i, j) = \circ] > 0$.*

[7] We stress that the scheme is the same, and it is used twice with independent and fresh randomness.

GenGESS

Shr(s_0, s_1)

1. Run $Shr_A(s_0, s_1)$ and $Shr_B(s_0, s_1)$. Let the shares and the possible reconstructed secrets be denoted as follows:

$MSSS\,scheme$	$Shr_\Sigma(s_0, s_1)$	$Rec_\Sigma(sh_1^X, sh_2^X)$	$Rec_\Sigma(sh_1^X, sh_3^X)$
A	sh_1^A, sh_2^A, sh_3^A	$s_{G(0,0)}$	$s_{G(0,1)}$
B	sh_1^B, sh_2^B, sh_3^B	$s_{G(1,0)}$	$s_{G(1,1)}$

2. Choose uniformly at random a *permutation bit* $b \in \{0, 1\}$ and, denoting with $||$ the *concatenation* operator, constructs shares $sh_{1,0}$ and $sh_{1,1}$ for the left wire of G, and $sh_{2,0}$ and $sh_{2,1}$ for the right wire, as follows:

left wire	right wire (if $b = 0$)	right wire (if $b = 1$)
$sh_{1,0} = b \| \| sh_1^A$	$sh_{2,0} = sh_2^A \| \| sh_2^B$	$sh_{2,0} = sh_2^B \| \| sh_2^A$
$sh_{1,1} = \bar{b} \| \| sh_1^B$	$sh_{2,1} = sh_3^A \| \| sh_3^B$	$sh_{2,1} = sh_3^B \| \| sh_3^A$

3. Output $((sh_{1,0}, sh_{1,1}), (sh_{2,0}, sh_{2,1}))$

Rec$(c||sh_\alpha, sh_\beta||sh_\gamma)$

- If $c = 0$ then output $Rec_\Sigma(sh_\alpha, sh_\beta)$; else output $Rec_\Sigma(sh_\alpha, sh_\gamma)$.

Table 1: General construction for a GESS scheme with a multi-secret sharing scheme.

- **Privacy**: *There exists a PPT algorithm Sim such that, for each $I_0 \in S$ and $I_1 \in S$, and for any selector $v \in V^2$, it holds that $Sim(s_{G(v)}) \overset{p}{\equiv} Sel(Shr(s_0, s_1), v)$.*

It is easy to check that the general construction for $GESS_G$, based on a multi-secret sharing scheme, realizes a $VGESS_G$ if the multi-secret sharing scheme therein used is substituted with a visual multi-secret sharing scheme. Indeed, the following result holds:

Corollary 3.8 *The* GenGESS *construction for a gate G realizes a $VGESS_G$ if the 2-MSSS is instanced with the* Probabilistic 2-MVCS.

4 A Visual Two-party Protocol

In this section we describe our visual two-party protocol. We start by showing how to realize a physical oblivious transfer and then we provide a full specification of the protocol.

4.1 Physical Oblivious Transfer

The 1-out-of-2 oblivious transfer (1-out-of-2-OT, for short) functionality [11] is an extensively studied cryptographic primitive, which plays a key-role in secure computation. Several implementations under general assumptions (e.g., enhanced trapdoor permutations) and specific assumptions (e.g., factoring, discrete-log assumption) are available, secure w.r.t. semi-honest and malicious adversaries, respectively. It is well known that the oblivious transfer is sufficient for secure multi-party function evaluation. Actually, the protocol we are going to propose is an *unconditionally secure reduction* of secure two-party function evaluation to 1-out-of-2-OT.

Let Alice's secrets be n-bit strings z_0 and z_1, let σ be Bob's bit-choice, and let $\perp$ denote no output. The 1-out-of-2-OT functionality is specified by $((z_0, z_1, \sigma) \rightarrow (\perp, z_\sigma))$. The construction we propose is partially inspired to the approach pursued in [8], when the voter comes out from the booth.

A physical 1-out-of-2 OT protocol. Let us assume that the two secrets z_0 and z_1 are represented in form of transparencies, and Alice has two *indistinguishable envelopes* which *perfectly hide* the transparency inside. Alice and Bob proceed as follows:

1. Alice puts the two secrets in the two envelopes, one in the first and one in the second, and closes both of them. She also adds to each envelope a paper post-it with number 0 and number 1, depending on the secret which is inside. Then, she hands the two envelopes to Bob.
2. Bob turns his shoulders to Alice[8], checks that the envelopes are identical, takes the envelopes with the post-it corresponding to the secret he is interested in, removes the post-it from both envelopes, turns again in front of Alice, and inserts under Alice surveillance the remaining envelope in a paper-shredder which reduces the envelop and its content in dust[9].

Theorem 4.1 *Assuming that indistinguishable envelopes which perfectly hide the transparency inside can be used, then the* Physical 1-out-of-2 OT *protocol realizes a physical perfectly secure* 1-*out-of*-2-*OT.*

4.2 Our Visual Two-Party Protocol

The protocol is the same reduction of secure function evaluation to 1-out-of-2 OT given via Construction 1 in [17], but with *VGESSs* instead of *GESSs*.

V2PC Protocol. Let $f : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^m$ be the target functionality and let $C(\cdot,\cdot)$ be a boolean circuit that computes $f(\cdot,\cdot)$, i.e., $C(\cdot,\cdot)$ is such that, for all inputs $x, y \in \{0,1\}^n$, it outputs $C(x,y) = f(x,y)$. Let us also assume that the circuit is composed of q *wires*, labeled uniquely with $\omega_1, \ldots, \omega_q$, $2n$ of which are *input wires*, say $\omega_1, \ldots, \omega_{2n}$, and m of which are *output wires*, and ℓ *gates*, represented for $h = 1, \ldots, \ell$ by functions $G_h : \{0,1\} \times \{0,1\} \rightarrow \{0,1\}$. No circuit-output wire is also a gate-input wire. Along the same line of the original Yao's protocol,

[8] If Alice thinks that Bob has had a career as illusionist, in order to be sure that Bob does not substitute the envelope that will be destroyed with an identical but fake one, might requests the Bob shows up in swimsuit.

[9] An alternative could be that the envelope is burned in front of Alice. The key-property that need to be satisfied is that the physical process should be irreversible, the secret cannot be even partially recovered.

V2PC Protocol

Shares construction phase (performed by Alice)

1. Let I_0 and I_1 be two images that encode the values 0 and 1. Associate them to the output wire of the output gates.
2. For each gate G_h whose output wire ω_k has been associated to images s_0 and s_1
   (a) Let ω_i and ω_j be the input wires, and let $VGESS_{G_h}$ be a visual GESS realizing gate G_h
   (b) Run $Shr(s_0, s_1)$, where s_0 encodes 0 and s_1 encodes 1, to obtain the shares $sh^{G_h}_{1,0}, sh^{G_h}_{1,1}$, and $sh^{G_h}_{2,0}, sh^{G_h}_{2,1}$. Let $sh^{G_h}_{1,0}, sh^{G_h}_{1,1}$ be the images s_0 and s_1 associated to 0 and 1 for the wire ω_i, and let $sh^{G_h}_{2,0}, sh^{G_h}_{2,1}$ be the images s_0 and s_1 associated to 0 and 1 for the wire ω_j.
3. Output the shares associated to wires $\omega_1, \ldots, \omega_n$ (Alice's input) and to $\omega_{n+1}, \ldots, \omega_{2n}$ (Bob's input).

Computation phase (performed by Alice and Bob)

1. Alice hands to Bob the shares $sh^{G_1}_{1,x_1}, sh^{G_2}_{1,x_2}, \ldots, sh^{G_n}_{1,x_n}$, corresponding to her input $x = x_1, \ldots, x_n$, associated to wires $\omega_1, \ldots, \omega_n$.
2. For every $j = 1, \ldots, n$, Alice and Bob execute the 1-out-of-2 OT protocol described before in which Alice's inputs are the shares $sh^{G_j}_{2,0}, sh^{G_j}_{2,1}$, associated to wire ω_{n+j}, while Bob's input is the bit y_j of his own input $y = y_1, \ldots, y_n$.
3. Bob, for $h = 1, \ldots, \ell$, applies the *Rec* algorithm of the $VGESS_{G_h}$, and computes the circuit output value $C(x, y) = f(x, y)$.
4. Finally Bob shows the result to Alice.

Table 2: V2PC Protocol

the description can be split in two phases: (i) shares construction phase, and (ii) interactive computation phase, described in Table 2.

At this point, we have all the elements needed to state and prove the following result:

Theorem 4.2 *Let $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^m$ be a boolean function, and let $C(\cdot,\cdot)$ be a boolean circuit that computes $f(\cdot,\cdot)$, i.e., $C(\cdot,\cdot)$ is such that, for all inputs $x, y \in \{0,1\}^n$, it holds that $C(x,y) = f(x,y)$. Then, assuming* indistinguishable envelopes *can be used, the* V2PC protocol *computes f in a perfectly secure way, in presence of a static semi-honest adversary.*

Remark. Notice that in the V2PC Protocol the size of the shares associated to the right wire input gate, doubles at each level of the circuit. However, as shown in [17], it is the best that can be done in a perfectly secure reduction of secure function evaluation to OT which uses GESS schemes.

5 Conclusions

Chapter 7 of [9] describes several applications of visual cryptography. In this paper we have shown that every two-party computation representable by means of a boolean function $f(\cdot,\cdot)$ can be performed preserving the privacy of the inputs x and y through a pure physical visual evaluation process. Several extensions are possible: e.g., study non-trivial extensions to the multi-party case, optimizations, use of different visual cryptography schemes in order to achieve different properties, just to name a few.

Cryptography is a wonderful world created by humans for serious and noble reasons, populated by many inanimate actors: *goals*, which are conceptualized and formalized, often in terms of *randomized functionalities*, in order to treat them properly; *basic tools and techniques*, which can be used to cope with the different issues an adversarial environment gives rise to when a functionality need to be realized, and *protocols*, which implement a well-defined communication and computational multi-party strategy to achieve the target goal, i.e., compute a certain functionality. Other actors are *definitions* and *proofs*, which are the usual linguistic, logical and mathematical constructs through which functionalities and tools are precisely stated and presented, and protocols are shown to achieve the functionalities for which they are designed in an adversarial model. Research papers often use all of them but, sometimes, some actors are missing, or do not play the roles they deserve. Depending on the contribution the paper provides to the field, papers could be categorized in different ways: in some papers *creativeness, intuition* and *new ideas* are the main components; in others, *generalization, abstraction*, or *sound and rigorous formalization* of ideas introduced in a rough or partial form before take the greatest part, while in others, *optimization, refinement*, and *efficiency improvements* of protocols, tools and techniques, are the main features. Papers might also be categorized according to the impact on the real world, i.e., if they provide a valuable *technological transfer*, or to the *theoretical advancement* to the field they bring with them. Nevertheless, some papers might just use some of the above actors for *intellectual* or *aesthetic* pleasure: neither technological transfer nor theoretical advancement (at least immediately) are provided. However, they might be a useful tool in divulgation activities, and in preparing simple and intriguing presentations of more complex ideas for a general audience. The current paper, perhaps, is (partially) a representative of the last class of papers.

References

[1] D. Beaver, S. Micali, and P. Rogaway. *The round complexity of secure protocols.* In Proc. 22nd ACM Symp. on Theory of Computing, pages 503-513, 1990.

[2] A. Beimel. *Secret Sharing: A survey*, Proc. of IWCC 2011, 2011.

[3] M. Bellare, V. T. Hoang, and P. Rogaway. *Garbling schemes.* Cryptology ePrint Archive, Report 2012/265, 2012.

[4] M. Bellare and P. Rogaway. *Robust computational secret sharing and a unified account of classical secret-sharing goals.* Proc. of the 14th ACM Conference on Computer and Communications Security (ACM CCS), ACM Press, 2007.

[5] D. Boneh, A. Sahai, and B. Waters. *Functional encryption: Definitions and challenges.* In TCC, pages 253-273, 2011.

[6] R. Capocelli, A. De Santis, L. Gargano, and U. Vaccaro. *On the size of shares for secret sharing schemes.* J. of Cryptology, vol. 6, pp. 157167, 1993.

[7] R. Canetti, *Security and Composition of Multiparty Cryptographic Protocols*, J. of Cryptology, vol 13, pp. 143-202, 2000.

[8] D. Chaum, *Secret-Ballot Receipts and Transparent Integrity*, available at http://www.vreceipt.com/article.pdf

[9] S. Cimato and C.-N. Yang editors, *Visual Cryptography and Secret Image Sharing*, CRC Press, Boca Raton, Florida, USA, ISBN 978-1-4398-3721-4, 2012.

[10] S. Cimato, R. De Prisco and A. De Santis *Probabilistic Visual Cryptography Schemes.* Comput. J. 49(1): 97-107, 2006

[11] S. Even, 0. Goldreich, and A. Lempel. *A Randomized Protocol for Signing Contracts.* CACM, vol. 28, No. 6, pp. 637-647, 1985.

[12] O. Goldreich. *Foundations of Cryptography*, Vol. II, MIT Press, 2004.

[13] O. Goldreich, S. Micali, and A. Wigderson. *How to play any mental game.* In STOC, pages 218-229, 1987.

[14] R. Gennaro, C. Gentry, and B. Parno. *Non-interactive verifiable computing: Outsourcing computation to untrusted workers.* In CRYPTO, pages 465-482, 2010.

[15] O. Kafri and E. Keren, *Encryption of pictures and shapes by random grids*, Optics Letters, Vol.12(6) pp. 377-379, 1987.

[16] E. Karnin, J. Greene, and M. Hellman. *On secret sharing systems.* IEEE Transactions on Information Theory, vol. 29, no. 1, pp. 3551, 1983

[17] V. Kolesnikov. *Gate Evaluation Secret Sharing and Secure One-Round Two-Party Computation.* In ASIACRYPT 2005, LNCS, Vol. 3788, pp. 136-155, 2005.

[18] M. Naor and A. Shamir. *Visual cryptography*, in Advances in Cryptology—Eurocrypt '94, Lecture Notes in Comput. Sci., Vol. 950, pp. 1-12, 1995.

[19] Y. Lindell and B. Pinkas. *A proof of security of Yaos protocol for two-party computation.* Journal of Cryptology, Vol. 22, pp. 161-188, April 2009.

[20] T. Moran and M. Naor, *Basing cryptographic protocols on tamper-evident seals*, Theoretical Computer Science, Vol. 411, pp. 1283 – 1310, 2010.

[21] C-N. Yang. *New Visual Secret Sharing Schemes using Probabilistic Method*, Pattern Recognition Letters, 25, 481-494, 2004.

[22] A. C. Yao. *Protocols for secure computations.* In Proc. 23tr IEEE Symp. on Foundations of Comp. Science, pages 160-164, 1982.

[23] A. C. Yao. *How to generate and exchange secrets (extended abstract).* In Proc. 27th IEEE Symp. on Foundations of Comp. Science, pages 162-167, 1986.

A A simple example

In this section we provide a simple example of application of the proposed method. The secret function is

$$f((x_1, x_2), (y_1, y_2)) = (x_1 \texttt{ and } y_1) \texttt{ or } (x_2 \texttt{ and } y_2)$$

where (x_1, x_2) is the private input of Alice and (y_1, y_2) is the private input of Bob, with x_1, x_2, y_1 and y_2 being bits.

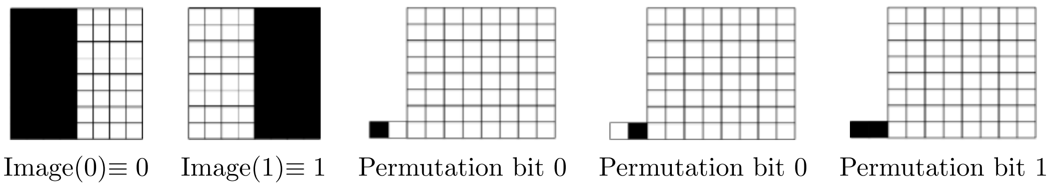


Figure 2: Bit representations: bit value, Image(0) and Image(1), and permutation bit, prepended to a blank image.

Binary values are represented as two images consisting of 8×8 pixels, more specifically we will use Image(0) and Image(1) shown in Figure 2 to encode 0 and 1.

The correctness property of the VGESS definition 3.7 requires that the black area of the secret image will be reconstructed (deterministically) with black pixels while the white area will be reconstructed, with some probability, with at least on white pixel. The rationale behind the definition is that in the reconstruction phase we will have to be able to visual distinguish the final output value of the function. The "quality" of the reconstructed image heavily depends on the depth of the circuit. Indeed, the more levels are in the circuit the more image superpositions, e.g., intermediate image reconstructions, have to be performed. For each intermediate image reconstruction, the number of black pixels in the output can only increase. Thus, the size of the image that we use to encode the values of the output (0 and 1), must be sufficiently large in order to guarantee that the reconstruction of the output will have, with some probability, at least one white pixel in the white area of the original secret image. More specifically, denoting with d the depth of the circuit, we have that the probability that a specific pixel in the reconstructed white area is white is equal to $(\frac{1}{2})^d$. Since our representation encodes the bit values (0 and 1) as images containing a secret white area of $m^2/2$ pixels then, the a-priori probability that one of these pixels is white is equal to $\frac{2}{m^2}$. Thus, to make sure (with any wanted probability) that we will be able to recognize the encoding of the bit values, we have to choose a sufficiently large m, i.e., $m >> \sqrt{2^{d+1}}$. In our example $d = 2$ and we have chosen $m = 8$.

In the share construction phase of the V2PC protocol, Alice has to construct a VGESS for each gate. Alice starts from gate G_3. Gate G_3 gives the output value of f, which can be either 0 or 1. Alice constructs the $VGESS_{G_3}$ which uses the two *2-MVCS* A and B. The shares of scheme

A reconstruct the secrets $s_{G_3(0,0)}, s_{G_3(0,1)}$, and those of scheme B reconstruct $s_{G_3(1,0)}, s_{G_3(1,1)}$. Since G_3 is an `or` gate we have that scheme A reconstructs $Image(0), Image(1)$, and scheme B $Image(1), Image(1)$.

To finish up the construction of the shares for $VGESS_{G_3}$ Alice has to choose, at random, the permutation bit b. In the example we are constructing we assume that the share of scheme A are placed on the left, so that $b = 0$ for Sh_1^A and clearly $\overline{b} = 1$ for Sh_1^B. The random bit will be visually represented as a 2-pixel image which encodes 0 as one black pixel and one white pixel and 1 as two black pixels[10].

The 2-pixel image will be prepended to the share image (and will become part of the share). Figure 2 shows the permutation bit prepended to a blank share.

Figure 3 (left) shows the shares for G_3, including the permutation bit.

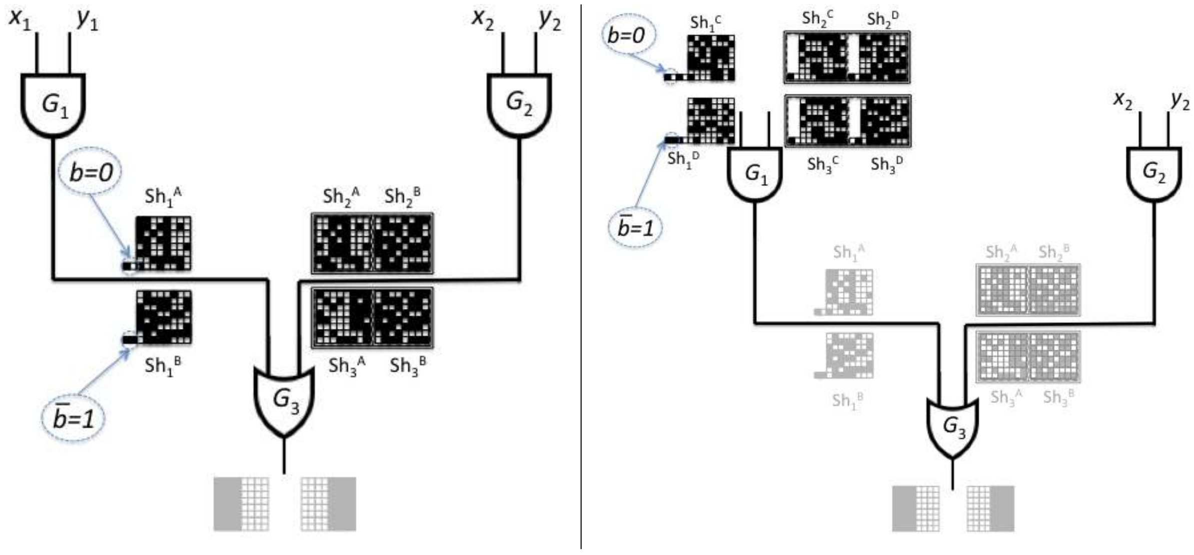


Figure 3: Shares construction for gate G_3 (left) and gate G_1 (right).

Now Alice can go on and consider gate G_1. The output of G_1 can be either $0||Sh_1^A$ or $1||Sh_1^B$, where the first element is the permutation bit. Hence the secrets that we need to share are $\{0||Sh_1^A, 1||Sh_1^B\}$. Share Sh_1^A corresponds to the wire value 0, while share Sh_1^B to the wire value 1. Since G_1 is an `and` gate, Alice will need to use two *2-MVCS* schemes C and D such that scheme C reconstructs $s_{G_1(0,0)} = 0||Sh_1^A$ and $s_{G_1(0,1)} = 0||Sh_1^A$, and scheme B reconstructs $s_{G_1(1,0)} = 0||Sh_1^A$ and $s_{G_1(1,1)} = 1||Sh_1^B$.

Also for gate G_1 Alice has to choose the permutation bit that will allow the correct reconstruction. Also in this case we decided to use $b = 0$. Figure 3 (right) shows the shares for G_1.

Finally Alice constructs the shares for G_2. The output wire of G_2 has to be able to reconstruct either $Sh_2^A||Sh_2^B$ (when the wire value is 0) or $Sh_3^A||Sh_3^B$ (when the wire value 1). Gate G_2 is an `and`

[10]Notice that, for the permutation bit, we are using a deterministic $(2,2)$-VCS with pixel expansion $m = 2$. We have used this solution for the permutation bit because, first of all it is possible to use a scheme with pixel expansion since each permutation bit propagates only from one level of the circuit to the subsequent one, and secondly because a scheme with pixel expansion allows a deterministic reconstruction.

gate, hence Alice will need to use two *2-MVCS* schemes E and F such that scheme E reconstructs $s_{G_2(0,0)} = Sh_2^A||Sh_2^B$ and $s_{G_2(0,1)} = Sh_2^A||Sh_2^B$, and scheme F reconstructs $s_{G_2(1,0)} = sh_2^A||Sh_2^B$ and $s_{G_2(1,1)} = Sh_3^A||Sh_3^B$.

Also for gate G_2 Alice has to choose a permutation bit that will allow the correct reconstruction. In this case we decided to use $b = 1$. Figure 4 shows the shares for G_2.

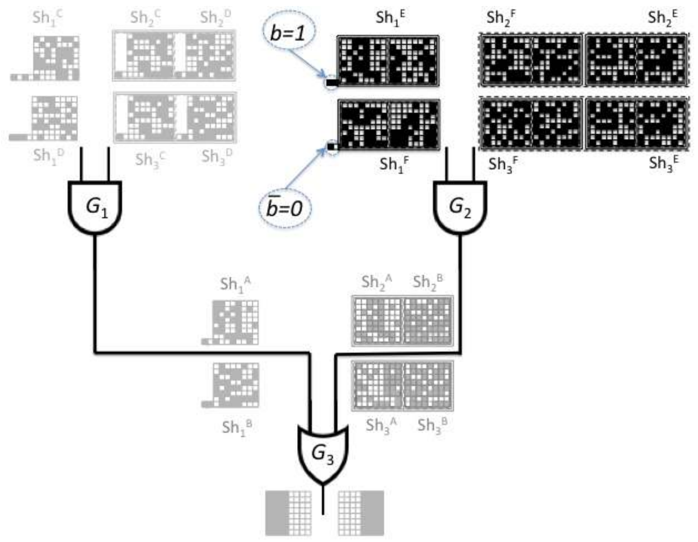


Figure 4: Shares construction for gate G_2

Alice has now completed the construction phase and all the shares that she needs for the computation are the ones shown in Figure 5. The figure shows for each input wire the shares that correspond to the values 0 and 1. For example for the left input wire of G_1 the value 0 corresponds to share Sh_1^C while the value 1 corresponds to the share Sh_1^D.

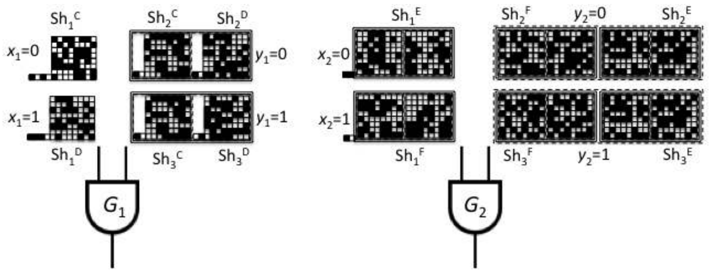
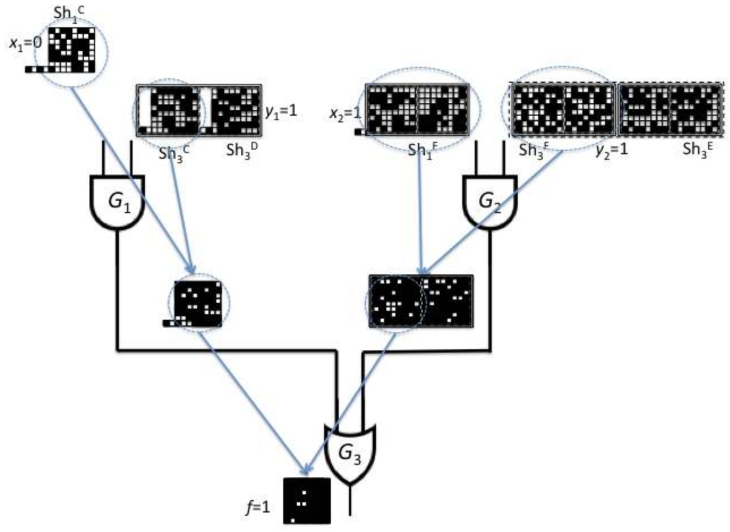


Figure 5: Visual circuit for the computation of f

Notice that all the shares shown in the figure are known only to Alice so far. At this point Alice chooses the shares that represent the values of her input. As an example, assume that Alice's input values are $x_1 = 0$ and $x_2 = 1$. Alice can throw away Sh_1^D and Sh_1^E and keep Sh_1^C, that represents $x_1 = 0$, and Sh_1^F, that represents $x_2 = 1$. Alice passes both shares, Sh_1^C and Sh_1^F to Bob. Then Alice and Bob run two executions of the 1-out-of-2 physical OT protocol so that Alice will pass to Bob only the shares that correspond to Bob's input. As an example assume that Bob's input values are $y_1 = 1$ and $y_2 = 1$. After the execution of the two 1-out-of-2 OT protocols, Bob has all the shares that correspond to his input values and can perform the visual computation of $f((1,0),(1,1))$, as depicted in Figure 6. Bob starts by evaluating gate G_1. He first looks at the permutation bit for G_1, which is 0. Then it takes the share on the left wire and superposes it on the left part of the share on the right wire, obtaining an intermediate share that will go on the left input wire of G_3. In a similar way Bob will evaluate G_2. At this point Bob has the input shares for G_3 and thus can evaluate also this gate. The permutation bit for gate G_3 is 0 and thus Bob superposes the share on the left input wire to the left part of the share on the right input wire. The result is the final output of the function and corresponds to 1.

Figure 7 shows another example of computation for the input $x_1 = 0, x_2 = 1, y_1 = 1$ and $y_2 = 0$.



c

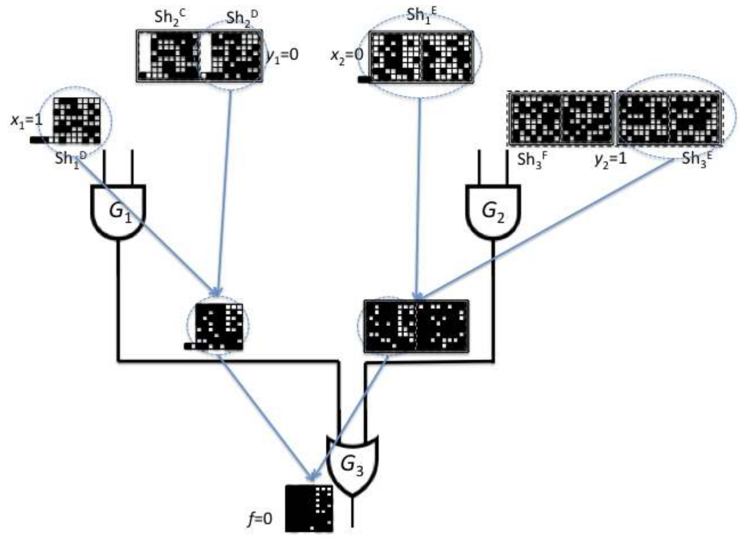
Figure 6: An example of visual evaluation of the circuit for the computation of f for the input $((1,0),(1,1))$

B Yao's Protocol

Let (Gen, E, D) be a symmetric encryption scheme, and let $f : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n$ be the target functionality[11]. Moreover, let $C(\cdot,\cdot)$ be a boolean circuit that computes $f(\cdot,\cdot)$, i.e., $C(\cdot,\cdot)$ is such that, for all inputs $x, y \in \{0,1\}^n$, it outputs $C(x,y) = f(x,y)$. Let us also assume that the circuit is composed of m *wires*, $2n$ of which are *input wires* and n are *output wires*, labeled uniquely with $\omega_1, \ldots, \omega_m$, and ℓ *gates*, represented for $h = 1, \ldots, \ell$ by functions $G_h : \{0,1\} \times \{0,1\} \rightarrow \{0,1\}$. No circuit-output wire is also a gate-input wire.

The protocol description can be split in two phases:

[11]Compared to the general model, and w.l.o.g. [19], Alice and Bob compute a deterministic same output functionality, i.e., $f(x,y) = f_1(x,y) = f_2(x,y)$. Moreover we assume that n is polynomial in the security parameter k.



c

Figure 7: An example of visual evaluation of the circuit for the computation of f for the input $((0,1),(1,0))$

- the garbled circuit construction phase (off-line)
- the interactive two-party phase (on-line)

In the first phase, Alice constructs a *garbled version* $\tilde{C}$ of $C(\cdot,\cdot)$ as follows: for $i = 1,\ldots,m$ and $j = 0,1$, using Gen, generates independently encryption keys $k^j_{\omega_i} \leftarrow Gen(1^n)$, and associates to each wire ω_i the encryption keys $k^0_{\omega_i}$ and $k^1_{\omega_i}$. Then, she constructs and associates to each gate $G_h(\cdot,\cdot)$ a gate table containing four double encryptions: assuming for example that the input wires and the output wire for a certain gate $G_h(\cdot,\cdot)$ are labeled ω_i, ω_j and ω_k, respectively, the table contains, in a randomly permuted order:

$E_{k^0_{\omega_j}}(E_{k^0_{\omega_i}}(k^{G_h(0,0)}_{\omega_k}))$
$E_{k^0_{\omega_j}}(E_{k^1_{\omega_i}}(k^{G_h(0,1)}_{\omega_k}))$
$E_{k^1_{\omega_j}}(E_{k^0_{\omega_i}}(k^{G_h(1,0)}_{\omega_k}))$
$E_{k^1_{\omega_j}}(E_{k^1_{\omega_i}}(k^{G_h(1,1)}_{\omega_k}))$

Notice that to *not* gates do not correspond tables. Indeed, they are easy to deal: it is just needed in the garbled circuit construction to associate to the output wire of a *not* gate the keys associated to the input wire in switched order, i.e., a *not* gate switches the semantics 0 and 1 secretly associated to the input keys.

The garbled circuit $\tilde{C}$ is the collection of *all* the gate tables associated to the gates $G_1,\ldots,G_\ell$, along with the keys associated to the circuit-output wires, with an *explicit mapping* of the bits 0 or 1 to these keys.

In the second phase, Alice and Bob proceed as follows

1. Alice sends the garbled circuit $\tilde{C}$ to Bob.

2. Assuming that $\omega_1, \ldots, \omega_n$ are the input wires corresponding to x and $\omega_{n+1}, \ldots, \omega_{2n}$ are the input wires corresponding to y, then
   (a) Alice sends Bob the encryption keys $k_{\omega_1}^{x_1}, k_{\omega_2}^{x_2}, \ldots, k_{\omega_n}^{x_n}$.
   (b) For ever $j = 1, \ldots, n$, Alice and Bob execute a 1-out-of-2 oblivious transfer protocol in which Alice's inputs are $k_{\omega_{n+j}}^{0}, k_{\omega_{n+j}}^{1}$ while Bob's input is the bit y_j.
3. Bob, by using the decryption algorithm D and proceeding table after table, computes the circuit output value $C(x, y) = f(x, y)$. Then, he sends $f(x, y)$ to Alice and both of them output $f(x, y)$ and halt.

Theorem 7 in [19], under standard assumptions on the oblivious transfer protocol and on the symmetric encryption scheme, proves that the construction computes f in a computationally secure way, in presence of static semi-honest adversaries.

C Secret sharing and multi-secret sharing schemes

Let us briefly introduce secret sharing and multi-secret sharing schemes. We do not follow the traditional entropy-based characterization, e.g., [16, 6], since in our analysis we are not going to use the entropy function[12]. Roughly speaking, a secret sharing scheme is a method through which a dealer shares a secret s among a set of parties, in such a way that, later on, some subsets of parties can reconstruct the secret, while others do not get any information about it. Similarly, a multi-secret sharing scheme enables the dealer to share more than one secret among the set of parties, in such a way that different subsets of parties reconstruct different secrets.
Let $\mathcal{P} = \{1, \ldots, n\}$ be a set of n parties. A collection of subsets $\mathcal{A} \subset 2^{\mathcal{P}}$ is monotone if $A \in \mathcal{A}$ and $A \subseteq B$ imply that $B \in \mathcal{A}$.

Definition C.1 Access structure. *An access structure on the set of parties $\mathcal{P}$ is a pair $(\mathcal{A}, \mathcal{F})$ such that $\mathcal{A} \subset 2^{\mathcal{P}}$ is a monotone collection, $\mathcal{F} \subset 2^{\mathcal{P}}$, and $\mathcal{A} \cap \mathcal{F} = \emptyset$.*

$(\mathcal{A}, \mathcal{F})$ is a *specification* of the sets which reconstruct the secret and of the sets which do not get any information about it. Usually sets in $\mathcal{A}$ are called *authorized*, while sets in $\mathcal{F}$ are called *forbidden*. Sets in $2^{\mathcal{P}} \setminus (\mathcal{A} \cup \mathcal{F})$ are sets for which we *do not care*.

Let $S, R, SH_1, \ldots, SH_n$ be finite sets. The set S is usually referred to as the set of *secrets*, the set R as the set of *random strings*, and the sets $SH_1, \ldots, SH_n$ as the sets of *shares*. Moreover, denote with s, r and $sh_1, \ldots, sh_n$ elements belonging to S, R and $SH_1, \ldots, SH_n$, respectively, and for each $X = \{i_1, \ldots, i_m\} \subseteq \mathcal{P}$, with $SH_X = SH_{i_1} \times \ldots \times SH_{i_m}$ and with $sh_X = (sh_{i_1}, \ldots, sh_{i_m})$. Using the above notation, we state the following:

Definition C.2 Secret sharing scheme (SSS for short). *Let S be a set of secrets, where $|S| \geq 2$. A secret sharing scheme $\Sigma = (Shr, Rec)$ with secret domain S realizing the access structure $(\mathcal{A}, \mathcal{F})$ is a pair of algorithms Shr and Rec where*

[12] A comprehensive study of secret sharing schemes which does not use the language of information theory can be found in [4]. See also a recent survey [2].

- *Shr is a probabilistic algorithm which takes in input a secret $s \in S$ and outputs a set of shares $sh_1, \ldots, sh_n$.*
- *Rec is a deterministic algorithm which takes in input a set of shares sh_X for $X \subseteq \mathcal{P}$, and outputs either $s \in S$ or $\perp$*

satisfying the following properties:

1. **Correctness.** *For each $A \in \mathcal{A}$, and for every secret $s \in S$, it holds that*
$$Pr[Rec(Shr(s)_A) = s] = 1$$
2. **Privacy.** *For each $F \in \mathcal{F}$, and for every $s_1 \in S$ and $s_2 \in S$, it holds that*
$$Pr[Shr(s_1)_F = sh_F] = Pr[Shr(s_2)_F = sh_F]$$

Property 1 guarantees that each authorized subset reconstructs the secret, while property 2 that each forbidden subset does not get any information from its subset of shares, since the subset is *compatible* with each possible secret with the same probability. Moreover, the definition does not assume *any* probability distribution on the sets S and R, and can be weakened by not requiring perfect reconstruction or by requiring just statistical or computational privacy. Definition C.2 can also be easily extended to *multi-secret*, i.e., the case in which the dealer distributes more than one secret. Formally, it is necessary to consider, instead of a single set of secrets S and a single access structure $(\mathcal{A}, \mathcal{F})$, sets of secrets $S_1, \ldots, S_\ell$ and access structures $(\mathcal{A}_1, \mathcal{F}_1), \ldots, (\mathcal{A}_\ell, \mathcal{F}_\ell)$.

Definition C.3 Multi-secret sharing scheme (MSSS for short). *Let $S_1, \ldots, S_\ell$ be sets of secrets where, for $i = 1, \ldots, \ell$, it holds that $|S_i| \geq 2$. A multi-secret sharing scheme $\Sigma = (Shr, Rec)$ with secret domains $S_1, \ldots, S_\ell$ realizing the access structures $(\mathcal{A}_1, \mathcal{F}_1), \ldots, (\mathcal{A}_\ell, \mathcal{F}_\ell)$ is a pair of algorithms Shr and Rec where*

- *Shr is a probabilistic algorithm which takes in input a tuple of secrets $s_M \in S_M$ and outputs a sequence of shares $sh_1, \ldots, sh_n$.*
- *Rec is a deterministic algorithm which takes in input a sequence of shares sh_X for $X \in \mathcal{P}$, and outputs either $s \in S_m$, for $m \in M$, or $\perp$*

satisfying, for $i = 1, \ldots, \ell$, the following properties:

1. **Correctness.** *For each $A \in \mathcal{A}_i$, and for every secret $s_i \in S_i$, it holds that*
$$Pr[Rec(Shr(s_M)_A) = s_i] = 1$$
2. **Privacy.** *For each $F \in \mathcal{F}_i$, and for every $s_M, s'_M \in S_M$, it holds that*
$$Pr[Shr(s_M)_F = sh_F] = Pr[Shr(s'_M)_F = sh_F]$$

Remark. Notice that, in our construction we consider a simple case of MSSS: the set of parties is $\mathcal{P} = \{1, 2, 3\}$, the sets of secrets are two and are equal, i.e., $S_1 = S_2 = S$, and the access structures are defined by $\mathcal{A}_1 = \{\{1, 2\}\}, \mathcal{F}_1 = \{\{1\}, \{2\}, \{3\}\}$ and $\mathcal{A}_2 = \{\{1, 3\}\}, \mathcal{F}_2 = \{\{1\}, \{2\}, \{3\}\}$.

D Proofs for VCS

Theorem D.1 *The* Probabilistic $(2,2)$-VCS *construction realizes a probabilistic* $(2,2)$*-VCS with no pixel expansion.*

Proof: We show that both properties of Definition 3.1 are satisfied.

Correctness. It is immediate to check that, for each $i, j = 1, \dots, n$, due to the structure of the matrices in $\mathcal{C}_\bullet$, a secret pixel $I(i,j) = \bullet$ is perfectly reconstructed, i.e, $\mathtt{Sup}(sh_1, sh_2)(i,j) = \bullet$. On the other hand, for each $i, j = 1, \dots, n$, due to the structure of the matrices in $\mathcal{C}_\circ$, and since $r_{i,j} \in R$ is chosen *uniformly at random*, a secret pixel $I(i,j) = \circ$ is reconstructed as $\circ$ with probability 1/2, i.e., $Pr[\mathtt{Sup}(sh_1, sh_2)(i,j) = \circ] = Pr[r_{i,j} = 0] = 1/2 > 0$.

Privacy. Due to the structure of the matrices in the collections $\mathcal{C}_\circ, \mathcal{C}_\bullet$ for each pixel position (i,j), a $\circ$ pixel in $sh_1(i,j)$ appears when $I(i,j) = \circ$ and *Shr* chooses $r_{i,j} = 0$ or when $I(i,j) = \bullet$ and *Shr* chooses $r_{i,j} = 0$; while, a $\circ$ pixel in $sh_2(i,j)$ appears when $I(i,j) = \circ$ and *Shr* chooses $r_{i,j} = 0$ or when $I(i,j) = \bullet$ and *Shr* chooses $r_{i,j} = 1$. Hence, in both cases, twice with the same probability. Similarly, it is easy to check that a $\bullet$ pixel in $sh_1(i,j)$ appears when $I(i,j) = \circ$ and *Shr* chooses $r_{i,j} = 1$ or when $I(i,j) = \bullet$ and *Shr* chooses $r_{i,j} = 1$; while, a $\bullet$ pixel in $sh_2(i,j)$ appears when $I(i,j) = \circ$ and *Shr* chooses $r_{i,j} = 1$ or when $I(i,j) = \bullet$ and *Shr* chooses $r_{i,j} = 0$. Again, in both cases, twice with the same probability. Thus, the property is satisfied. □

Theorem D.2 *The* Probabilistic 2-MVCS *construction realizes a probabilistic* 2*-MVCS with no pixel expansion.*

Proof: We show that both properties of Definition 3.3 are satisfied.

Correctness. Notice that sh_1 and sh_2 are constructed as in a probabilistic $(2,2)$-VCS using I_0 as secret image. Hence, black pixels of I_0 are reconstructed perfectly, i.e, when $I_0(i,j) = \bullet$ we have $\mathtt{Sup}(sh_1, sh_2)(i,j) = \bullet$, while white pixels of I_0 are correctly reconstructed with probability 1/2, i.e., when $I_0(i,j) = \circ$ we have that $Pr[\mathtt{Sup}(sh_1, sh_2)(i,j) = \circ] = Pr[r_{i,j} = 0] = 1/2 > 0$. Moreover, by construction of $sh_3(i,j)$ we have that the same holds for sh_1 and sh_3 with respect to I_1, that is when $I_1(i,j) = \bullet$ we have $\mathtt{Sup}(sh_1, sh_3)(i,j) = \bullet$, and when $I_1(i,j) = \circ$ we have that $Pr[\mathtt{Sup}(sh_1, sh_3)(i,j) = \circ] = Pr[r_{i,j} = 0] = 1/2 > 0$.

Privacy. The privacy of sh_1 and sh_2 derives from the fact that they are constructed as in a probabilistic $(2,2)$-VCS. Since sh_1 satisfy privacy we have that for every i, j, $Pr[sh_1(i,j) = \circ] = Pr[sh_1(i,j) = \bullet] = Pr[r_{i,j} = 0] = 1/2 > 0$. Share sh_3 is constructed simply by flipping the pixels of sh_1 when I_1 has a black pixel and keeping the pixels of sh_1 when I_1 has a white pixel. Since $Pr[sh_1(i,j) = \circ] = Pr[sh_1(i,j) = \bullet] = Pr[r_{i,j} = 0] = 1/2$ we have that the same holds for sh_3, that is for every i, j, $Pr[sh_3(i,j) = \circ] = Pr[sh_3(i,j) = \bullet] = Pr[r_{i,j} = 0] = 1/2 > 0$. □

E Proof of the GESS

Theorem E.1 *The* GenGESS *construction for a gate G realizes a $GESS_G$.*

Proof: (Sketch). We show that both properties of Definition 3.5 are satisfied.

Correctness. Correctness follows from a simple inspection of all possible cases. For example, let us assume that the permutation bit $b = 0$, and let us analyze the four possible input-gate configurations, defined by properly setting the selector v:

- Let $v = (v_1, v_2) = (0, 0)$. In this case $Sel(Shr(s_0, s_1), (0, 0)) = (sh_{1,0}, sh_{2,0})$. Since $b = 0$, it holds that $Rec(sh_{1,0}, sh_{2,0}) = Rec_\Sigma(sh_1^A, sh_2^A) = s_{G(0,0)}$, due to the correctness property of the instance A of the 2-MSSS scheme Σ.
- Let $v = (v_1, v_2) = (0, 1)$. In this case $Sel(Shr(s_0, s_1), (0, 1)) = (sh_{1,0}, sh_{2,1})$. Since $b = 0$, it holds that $Rec(sh_{1,0}, sh_{2,1}) = Rec_\Sigma(sh_1^A, sh_3^A) = s_{G(0,1)}$, due to the correctness property of the instance A of the 2-MSSS scheme Σ.
- Let $v = (v_1, v_2) = (1, 0)$. In this case $Sel(Shr(s_0, s_1), (1, 0)) = (sh_{1,1}, sh_{2,0})$. Since $b = 0$, it holds that $Rec(sh_{1,1}, sh_{2,0}) = Rec_\Sigma(sh_1^B, sh_2^B) = s_{G(1,0)}$, due to the correctness property of the instance B of the 2-MSSS scheme Σ.
- Let $v = (v_1, v_2) = (1, 1)$. In this case $Sel(Shr(s_0, s_1), (1, 0)) = (sh_{1,1}, sh_{2,1})$. Since $b = 0$, it holds that $Rec(sh_{1,1}, sh_{2,1}) = Rec_\Sigma(sh_1^B, sh_3^B) = s_{G(1,1)}$, due to the correctness property of the instance B of the 2-MSSS scheme Σ.

If the permutation bit $b = 1$, a similar analysis holds, switching the first and the second parts of the share associated to the left input wire of G.

Privacy. We need to show that there exists a PPT simulator Sim which, on input $s_{G(v)}$, outputs a pair of shares $(sh_{1,x}, sh_{2,y})$, associated to the left and right input wires of G, distributed *exactly* as the shares provided in a real execution of the protocol by $Sel(Shr(s_0, s_1), v)$. Such an algorithm can be constructed as follows:

Sim$(s_{G(v)})$

1. Choose uniformly at random a secret $s' \in S$ and the permutation bit b.
2. Construct two instances, C and D, of the 2-MSSS scheme Σ
3. Run $Shr_C(s_{G(v)}, s') = (sh_1^C, sh_2^C, sh_3^C)$ and $Shr_D(s_{G(v)}, s') = (sh_1^D, sh_2^D, sh_3^D)$
4. If $b = 0$ then set $sh_{1,x} = b||sh_1^C$ and $sh_{2,y} = sh_2^C||sh_2^D$

   else set $sh_{1,x} = \bar{b}||sh_1^D$ and $sh_{2,y} = sh_2^D||sh_2^C$
5. Output $(sh_{1,x}, sh_{2,y})$.

Since **Sim** applies the same strategy of $Shr(\cdot,\cdot)$ and the same 2-MSSS scheme Σ, used in a real execution of the $GESS_G$ scheme, it follows that the pair of shares $(sh_{1,x}, sh_{2,y})$ is distributed *exactly* as the shares provided in a real execution of the protocol. Indeed, notice that the only differences between the output of **Sim** and the output of a real execution is that **Sim** invokes $Shr(\cdot,\cdot)$ by giving in input as first input the secret $s_{G(v)}$ and a random secret s' as second while, in a real execution, one of the secret is $s_{G(v)}$, and could be the first or the second in input, and another secret which might be different from the secret s'. We show that both differences do not induce a different distribution on the shares.

Case 1: (The pairs of secrets are different.) Assuming the $s_{G(v)} = s_0$, a difference in the distributions of $sh_3^C||sh_3^D$, provided by **Sim**, and $sh_3^C||sh_3^D$, provided by the real execution, would imply the existence of two sets of secrets, $\{s_{G(v)}, s'\}$ and $\{s_{G(v)}, s''\}$ which contradicts the privacy property of the instance C or D of the MSSS. Hence, the use of an s' different from the other real secret s'' used in a real execution, does not induce a different distribution on the shares.

Case 2: (The order of the secrets is different.) Similarly, a difference in the distributions of $sh_2^C||sh_2^D$ and $sh_3^C||sh_3^D$, provided either by **Sim** or by a real execution, would imply the existence of two sets of secrets, $\{s_{G(v)}, s'\}$ and $\{s', s_{G(v)}\}$ which contradicts the privacy property of the instance C or D of the MSSS. Hence, also the order does not matter.

$\square$

F Proofs for the visual TPC procotol

Theorem F.1 *Assuming that indistinguishable envelopes which perfectly hide the transparency inside exist, then Construction* 1 *realizes a physical perfectly secure* 1*-out-of-*2*-OT.*

Proof: (Sketch.) It is easy to check that the protocol is correct. To prove that it matches Definition 2.1 we show that an efficient physical simulator Sim_2 can be realized. Indeed, since Alice does not receive anything from Bob, Sim_1 is trivial, it does nothing. Instead, Sim_2

- chooses, from the same sets from which Alice chooses envelopes and post-its, two indistinguishable envelopes and two post-its
- writes 0 and 1 on the two post-its, associates them to the envelopes, and inserts the transparency z_σ he has received from the protocol in the envelope with number σ and a blank transparency in the other.

It is easy to see that this view is *identically distributed* to the real view he has running the protocol with Alice. $\square$

Theorem F.2 *Let* $f : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^m$ *be a boolean function, and let* $C(\cdot,\cdot)$ *be a boolean circuit that computes* $f(\cdot,\cdot)$*, i.e.,* $C(\cdot,\cdot)$ *is such that, for all inputs* $x, y \in \{0,1\}^n$*, it holds that* $C(x,y) = f(x,y)$*. Then, assuming* indistinguishable envelopes *can be used, the* V2PC Protocol *computes* f *in a perfectly secure way, in presence of a static semi-honest adversary.*

Proof: (Sketch). The proof, according to the paradigm formalized in [7], proceeds in two steps: first, in a *hybrid* model where we assume that the ideal functionality 1-out-of-2 OT is available, we show that the V2PC Protocol computes f in a perfectly secure way, in presence of a static semi-honest adversary. Actually, in this step, since the V2PC Protocol we have described coincides with Construction 1 given in [17], up to the use of $VGESSs$ instead of $GESSs$, the same proof given in [17] applies. Indeed, $VGESS$ schemes are a special class of $GESS$ schemes. Hence, any time the simulator Sim_{Bob} in [17] invokes the simulator Sim_{GESS_G} for a $GESS_G$, our simulator invokes the simulator Sim_{VGESS_G}. Then, in the second step, using Theorem F.1 and the composition theorem in [7], we conclude that V2PC Protocol computes f in a perfectly secure way, in presence of a static semi-honest adversary. $\square$

G The shares

The last pages of the paper contain enlarged versions of all the shares used in the examples. These pictures can be printed on transparencies and used to reproduce manually the visual computation of the examples.

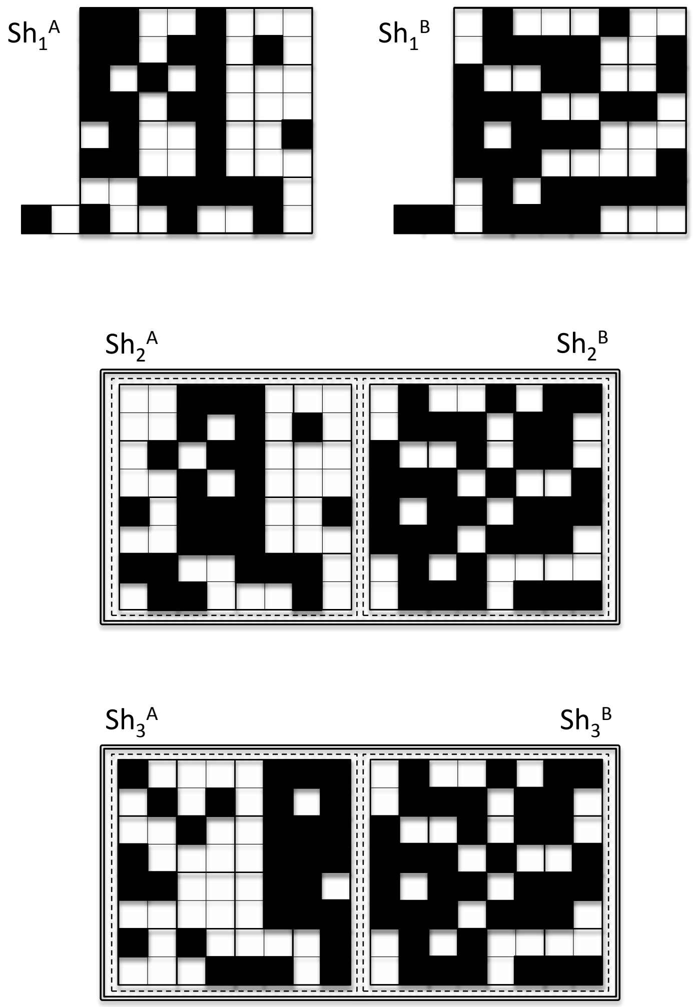


Figure 8: Shares for schemes A and B.

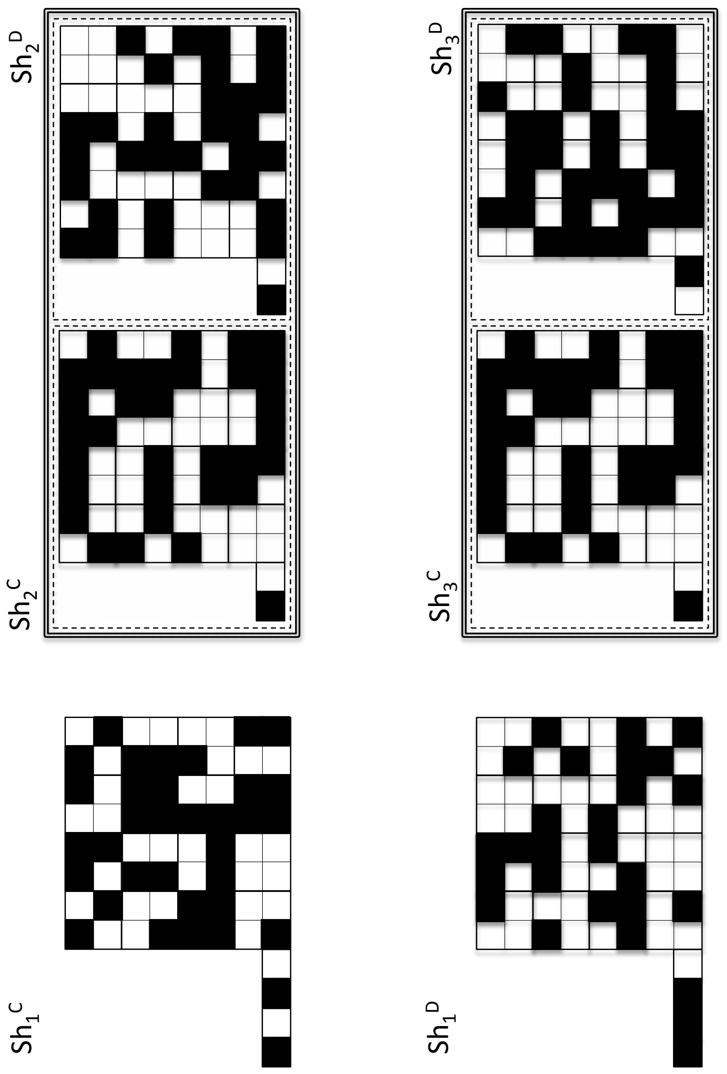
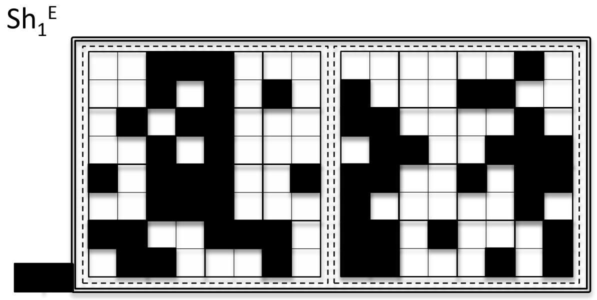


Figure 9: Shares for schemes C and D.



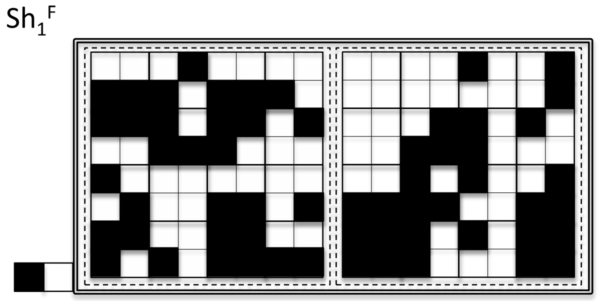


Figure 10: Shares 1 for schemes E and F.

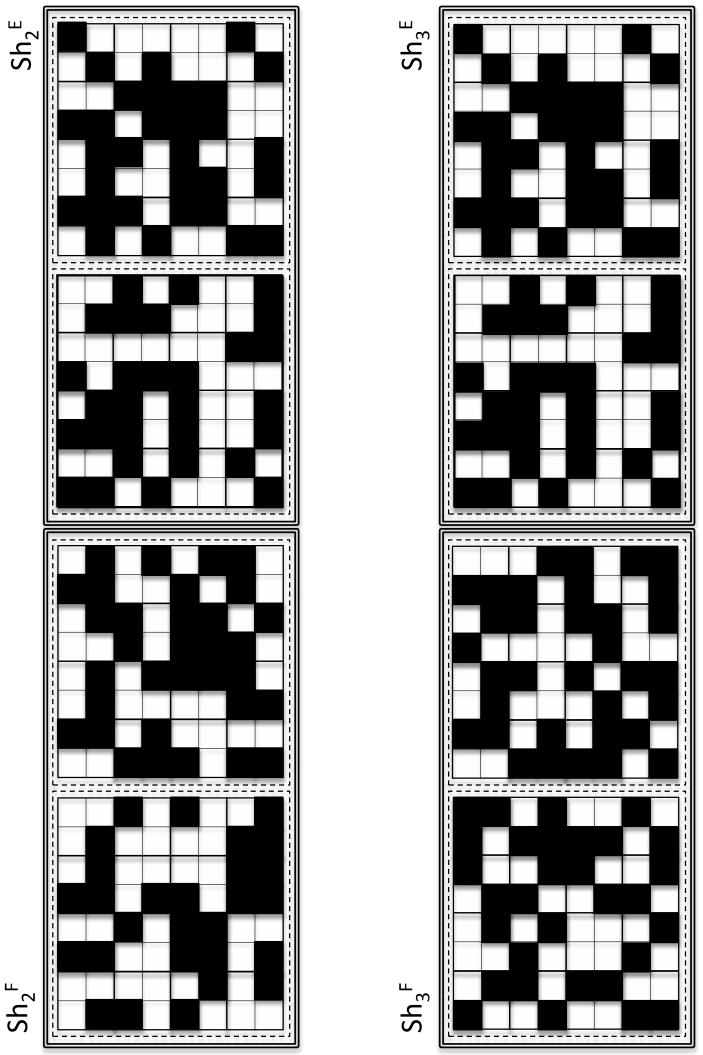


Figure 11: Shares 2 and 3 for schemes E and F.