Classes of Garbling Schemes

Tommi Meskanen, Valtteri Niemi, Noora Nieminen

***Abstract*—Bellare, Hoang and Rogaway elevated *garbled circuits from a cryptographic technique to a cryptographic goal* by defining several new security notions for garbled circuits [3]. This paper continues at the same path by extending some of their results and providing new results about the classes of garbling schemes defined in [3]. Furthermore, new classes of garbling schemes are defined and some results concerning them and their relation to earlier classes are proven.**

***Index Terms*—garbled circuits, garbling schemes, secure multiparty computations, privacy**

I. Introduction

The history of garbled circuits traces back to A. Yao, who introduced the technique in [7]. The term *garbled circuit* was introduced by Beaver, Micali and Rogaway [2] where they introduced a way of performing secure multiparty computation with Yao's circuit garbling technique. Since then Yao's garbled circuits have been used for various purposes even though there was no formal definition what is meant by garbling. No proof of security existed either - until Lindell and Pinkas introduced one for a particular garbled circuit using a protocol assuming semi-honest adversaries [5], [6]. After this result, also a proof of security against covert and malicious adversaries has been published [1], [6]. Again, these results are obtained for a specific protocol using garbling schemes rather than considering the security of garbling itself.

The first formal definition of a garbling scheme has recently been proposed by Bellare, Hoang and Rogaway in [3]. A garbling scheme is defined as a five-tuple of functions: the actual garbling procedure $\mathtt{Gb}$, the encryption function $\mathtt{En}$, the decryption function $\mathtt{De}$, the garbled evaluation function $\mathtt{Ev}$ and the original evaluation function $\mathtt{ev}$. The idea behind garbling is the following. Let f be a function which is to be evaluated for different inputs x but in such a way that neither f nor x can be learnt from the evaluation process. Therefore, a garbled version F is created and instead of computing $y = \mathtt{ev}(f, x)$ we compute $Y = \mathtt{Ev}(F, X)$ where X is obtained from x by encryption. After this y is obtained from Y by decryption. Figure 1 illustrates the garbling procedure.

Rogaway et al. define also three security notions for garbling schemes. These notions are expressed via code-based games which are defined in such a way that they capture the intuition behind the different notions: privacy, obliviousness and authenticity are all defined to be reached, if the adversary has only a negligible advantage for winning a particular game. Moreover, these notions have two different models, either based on indistinguishability or simulation. Roughly speaking, indistinguishability means that the adversary cannot distinguish between garblings of two functions. The simulation type means that an adversary is incapable of distinguishing garbling of the function of its own choice from another similar looking function devised by a simulator. Here we refer to the next section for the formal definitions.

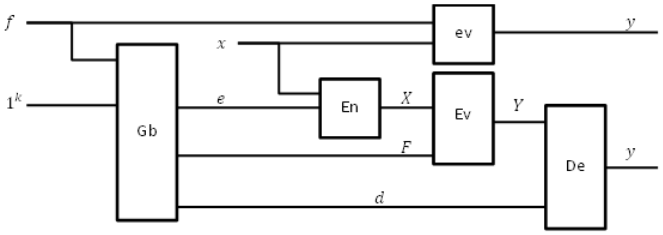


Figure 1: Description of the technique behind garbling. The diagram also illustrates that the result of evaluation with garbling must coincide with the result obtained without garbling.

Another seminal achievement in [3] is that relations between the different security notions have been proven. Rogaway et al. also provide two concrete garbling schemes, one of which achieves not only privacy but also obliviousness and authenticity. This example assures that the defined security classes are not empty.

This paper consists of three sections. In the first section we define all the necessary concepts, and give an informal description of them so that the idea behind the concept would be more comprehensive to the reader. In the second section we provide new results about the already known classes: some of the results are extensions to the results in [3], some inspired by the results in [3]. The third section provides modified definitions of the games used to define the different security notions. In this manner, we obtain new classes of garbling schemes by minor modifications in the games. Then, we prove some relations not only between the new and existing classes but also among the new classes. We also discuss intuition behind these new classes.

II. Definitions

In this section, we provide the basic definitions and notations. As usual, $\mathbb{N}$ will be the set of positive integers. A *string* is a finite sequence of bits. In addition to the basic strings, there is a special symbol $\perp$. The meaning of this symbol is explained later where the context of usage will be clearer.

Let A be a finite set. Notation $y \xleftarrow{} A$ means that an element is selected uniformly at random from the set A, and this element is assigned to y. If A denotes an algorithm, then notation $A(x_1, \ldots, x_n)$ means the output of the algorithm A on inputs $x_1, \ldots, x_n$.

T. Meskanen is a researcher at the Department of Mathematics and Statistics, University of Turku, Finland (email: tommes@utu.fi).

V. Niemi is a professor at the Department of Mathematics and Statistics, University of Turku, Finland (email: pevani@utu.fi).

N. Nieminen is a doctoral student at Turku Centre for Computer Science, University of Turku, Finland (email: nmniem@utu.fi).

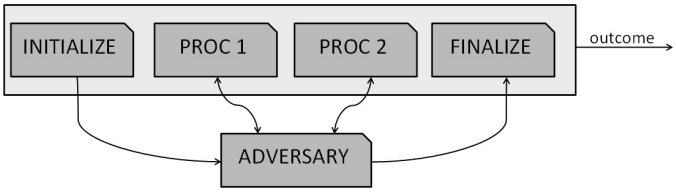


Figure 2: The idea of a code-based game is captured in the above image.

As usual, we say that a function $f : \mathbb{N} \to \mathbb{R}$ is negligible if for every $c > 0$ there is an integer N_c such that $|f(x)| < x^{-c}$ for all $x > N_c$

A. Code-based games

The proofs in this paper are heavily based on *code-based games*. Following the terminology presented in [4], a game is a collection of procedures called *oracles*. This collection may contain three types of procedures: `INITIALIZE`, `FINALIZE` and other named oracles. The word *may* is used, since all the procedures in a game are optional.

The entity playing a game is called *adversary*. When the game is run with an adversary, first the `INITIALIZE` procedure is called. It possibly provides an input to the adversary, who in turn may invoke other procedures before feeding its output to the `FINALIZE` procedure. The `FINALIZE` receives an output of the adversary, and creates a string that tells the outcome from the game typically consisting of one bit of information: whether the adversary has won or not. This description about code-based games is quite informal and gives only the intuition behind the concept. The Figure 2 serves as an illustration. For a more formal description, we refer to [4].

B. Garbling schemes

In this section we provide a formal definition of garbling schemes and their security, and here we follow the guidelines provided in [3].

Formally, a garbling scheme is a 5-tuple $\mathcal{G} = (\mathtt{Gb}, \mathtt{En}, \mathtt{De}, \mathtt{Ev}, \mathtt{ev})$ of algorithms, from which the first is probabilistic and the rest are deterministic. Let f denote the string that represents the original function. The last component in the 5-tuple is the evaluation function $\mathtt{ev}(f, \cdot) : \{0,1\}^n \to \{0,1\}^m$ which we want to garble. Here, the values $n = f.n$ and $m = f.m$ represent the lengths of the input x and the output $y = ev(f, x)$. They must also be efficiently computable from f. The first component `Gb` denotes the garbling algorithm. It takes f and 1^k as its inputs, where $k \in \mathbb{N}$ is a security parameter, and returns (F, e, d) on this input. String e describes the encryption algorithm $\mathtt{En}(e, \cdot)$ which maps an initial input x to a garbled input $X = \mathtt{En}(e, x)$. String F describes the garbled function $\mathtt{Ev}(F, \cdot)$. It returns the garbled output $Y = \mathtt{Ev}(F, X)$. Finally, string d describes the decryption algorithm $\mathtt{De}(d, \cdot)$ which on a garbled input returns the final output $y = \mathtt{De}(d, Y)$. Here we refer to Figure 1 to get an idea of how a garbling scheme works.

`NOTE:` Occasionally, we use a specific evaluation function $\mathtt{ev}_{circ}$ as `ev` in the 6-tuple. For it, we first define *a conventional circuit* by a 6-tuple $f = (n, m, q, A, B, G)$. The first component denotes the number of input wires ($n \geq 2$), the second is the number of output wires ($m \geq 1$), and the third component represents the number of gates ($q \geq 1$) in the circuit. The function A identifies the first incoming wire, whereas B identifies the second incoming wire of each gate. The remaining component G is a function identifying the functionality of each gate. For a more specific definition of a circuit, see [3]. Finally, the circuit evaluation function $\mathtt{ev}_{circ}$ is the usual canonical evaluation function:

proc $\mathtt{ev}_{circ}(f, x)$
$(n, m, q, A, B, G) \leftarrow f$
for $g \leftarrow n+1$ **to** $n+q$ **do** $a \leftarrow A(g), b \leftarrow B(g), x_g \leftarrow G_g(x_a, x_b)$
return $x_{n+q-m+1} \cdots x_{n+q}$

There are some additional requirements that garbling schemes must fulfill. These are *length, non-degeneracy* and *correctness* conditions. The length condition means that the lengths of F, e, d may only depend on the security parameter k, the values $f.n, f.m$ and the length of the string f. Non-degeneracy condition means the following: if $f.n = g.n$, $f.m = g.m$, $|f| = |g|$, $(F, e, d) = \mathtt{Gb}(1^k, f; r)$ and $(G, e', d') = \mathtt{Gb}(1^k, g; r)$ where r represents random coins of `Gb`, then $e = e'$ and $d = d'$. Correctness requires that $\mathtt{De}(d, \mathtt{Ev}(F, \mathtt{En}(e, x)))$ will always give the same result as $\mathtt{ev}(f, x)$.

By the concept of *a side-information function*, we capture the information revealed about f by the garbling process. In the case of circuits and $\mathtt{ev}_{circ}$, this might be the size of the circuit that was garbled, the topology of it or something else - even the whole initial circuit. Formally, a side-information function Φ deterministically maps string f to string $\Phi(f)$. Let $f = (n, m, q, A, B, G)$ be a circuit. Then, we define $\Phi_{size}(f) = (n, m, q)$, which is the side-information function revealing the size of the garbled circuit. Other side-information functions are $\Phi_{circ}(f) = f$ which thus reveals the entire circuit, and Φ_{topo} which reveals the topology of the initial circuit, i.e. $\Phi_{topo} = (n, m, q, A, B)$.

C. The security notions of garbling schemes

There are three types of security: *privacy, obliviousness* and *authenticity*. The first two types also have two distinct models: one based on *indistinguishability* and another based on *simulation*. In all cases, the security is defined through a code-based game consisting of a procedure named `GARBLE` and finalization procedure `FINALIZE`. The procedure `GARBLE` is not to be confused with the garbling function `Gb` : the garbling function `Gb` is a component of a garbling scheme $\mathcal{G}$, whose security the adversary tries to break via the procedure `GARBLE`.

Before the game starts, the garbling scheme $\mathcal{G}$ and the side-information function Φ are fixed in the games based on indistinguishability model. In simulation model, also the simulator $\mathcal{S}$ is fixed although details of it are not assumed to be known to the adversary. The `GARBLE` procedure gives the challenge of the game to the adversary and the `FINALIZE`

procedure determines whether the adversary wins the game or not. The adversary is assigned a certain advantage depending on the probability of winning the game. This advantage in turn determines whether the garbling scheme is secure or not.

Table 1 gives the different GARBLE procedures needed in the games to define different security notions. Note that in this first formal description we use the subscripts, but after that, we omit them if they are clear from the context. For example, we will write *PrvSim* game instead of $PrvSim_{\mathcal{G},\Phi,\mathcal{S}}$.

Let $\mathcal{G} = (\mathtt{Gb}, \mathtt{En}, \mathtt{De}, \mathtt{Ev}, \mathtt{ev})$ be a garbling scheme, $k \in \mathbb{N}$ a security parameter and Φ a side-information function. The following definitions are informal, and they are mentioned to capture the idea behind the security notions. For a more formal treatment, see [3].

PRIVACY: Privacy has two types of notions, and hence there are two different games with distinct GARBLE procedures, $PrvInd_{\mathcal{G},\Phi}$ and $PrvSim_{\mathcal{G},\Phi,\mathcal{S}}$. The biggest difference between these two is that the latter requires an auxiliary algorithm to be defined, namely the simulator $\mathcal{S}$.

The game $PrvInd$ consists of a GARBLE procedure, which is called by the adversary *exactly once* during one game, and a FINALIZE procedure. Informally the game goes as follows: the adversary calls the GARBLE procedure having two appropriate functions and their inputs as the feed. The procedure returns a garbled version of one of the functions and its input, and the adversary guesses which of the functions got garbled. The FINALIZE procedure takes two inputs, value of parameter b from GARBLE and adversary's guess b', and tells whether the answer given by the adversary was correct or not, and this will then be the outcome of the game.

The game $PrvSim$ has also two procedures, its own GARBLE and FINALIZE, from which the latter has the same functionality as in $PrvInd$ game. The difference in GARBLE procedure is, that now the other function, from which the function f is to be distinguished, is devised by the simulator. The adversary must tell the difference between an actual function and a "fake" function.

We define the advantage of an adversary A in game $PrvInd$ as follows:

$$\mathbf{Adv}^{prv.ind,\Phi}_{\mathcal{G}}(\mathcal{A},k) = 2 \cdot \Pr\left[PrvInd^{\mathcal{A}}_{\mathcal{G},\Phi}(k)\right] - 1.$$

If the advantage function $\mathbf{Adv}^{prv.ind,\Phi}_{\mathcal{G}}(\mathcal{A},\cdot)$ is negligible for all PT adversaries $\mathcal{A}$ then we say that the garbling scheme $\mathcal{G}$ is *prv.ind secure over* Φ. Similarly, we define the advantage of an adversary $\mathcal{B}$ in game $PrvSim$ as $\mathbf{Adv}^{prv.sim,\Phi,\mathcal{S}}_{\mathcal{G}}(\mathcal{B},k) = 2 \cdot \Pr\left[PrvSim^{\mathcal{B}}_{\mathcal{G},\Phi,\mathcal{S}}(k)\right] - 1$. Then, we define that a garbling scheme $\mathcal{G}$ is *prv.sim secure over* Φ if for every PT adversary there exists a PT simulator $\mathcal{S}$ such that $\mathbf{Adv}^{prv.sim,\Phi,\mathcal{S}}_{\mathcal{G}}(\mathcal{B},k)$ is negligible.

OBLIVIOUSNESS: At first sight, the games for obliviousness seem similar to the privacy games. The difference is that the decryption algorithm d is not given to the adversary, and hence the adversary cannot compute the final output $y = \mathtt{De}(d, \mathtt{Ev}(F,X))$. Informally, the adversary is asked to distinguish two functions and their inputs from each other without knowing the result of evaluation.

The adversary has an advantage which is calculated as in the privacy model. The obv.ind and the obv.sim security of a garbling scheme $\mathcal{G}$ are defined similarly as in the corresponding Prv-games.

AUTHENTICITY: Here the FINALIZE procedure is a little more complex than in the two cases above. The finalization procedure of a game checks whether the adversary is able to produce a valid garbled output Y different to $\mathtt{Ev}(F,X)$ or not. Also the advantage function is slightly different: $\mathbf{Adv}^{aut}_{\mathcal{G}}(\mathcal{A},k) = \Pr\left[Aut^{\mathcal{A}}_{\mathcal{G}}(k)\right]$. Again, a garbling scheme is aut-secure, if for all polynomial time adversaries $\mathcal{A}$ the advantage function $\mathbf{Adv}^{aut}_{\mathcal{G}}(\mathcal{A},\cdot)$ is negligible.

We denote $\mathtt{GS}(xxx,\Phi)$ to be the set of all garbling schemes that are $xxx-$secure over the side-information function Φ, where xxx denotes the type of security: $prv.ind$, $prv.sim$, $obv.ind$, $obv.sim$, $mod.ind$, $mod.sim$, $mod.ind2$ or $mod.sim2$. The notion $\mathtt{GS}(aut)$ means the set of all $aut-$secure garbling schemes. $\mathtt{GS}(ev)$ means the class of garbling schemes which use the evaluation function ev.

III. Results about established classes of garbling schemes

In this section we provide results concerning the security classes *prv.ind, prv.sim, obv.ind, obv.sim* defined in section 2. The first two theorems consider the effect of different side-information functions to the sets of garbling schemes. The following two theorems provide extensions to the existing results in [3] – the non-inclusions are obtained for any side-information function Φ instead of restricting it to Φ_{topo}. Then we continue with two results that provide parallel results to [3]. Finally, the last two theorems in this section provide new results about the established security classes of garbling schemes.

Theorem 1: Suppose that two different side-information functions Φ_a and Φ_b satisfy the condition

$$\Phi_a(f_0) = \Phi_a(f_1) \Rightarrow \Phi_b(f_0) = \Phi_b(f_1). \quad \textbf{(Condition } (*))$$

Then we have the inclusion $\mathtt{GS}(prv.ind,\Phi_b) \subseteq \mathtt{GS}(prv.ind,\Phi_a)$. If we additionally assume that there exists a polynomial time function g such that $g(\Phi_a(f)) = \Phi_b(f)$ then we also have $\mathtt{GS}(prv.sim,\Phi_b) \subseteq \mathtt{GS}(prv.sim,\Phi_a)$.

Proof: Let $\mathcal{G} = (\mathtt{Gb}, \mathtt{En}, \mathtt{De}, \mathtt{Ev}, \mathtt{ev}) \in \mathtt{GS}(prv.ind,\Phi_b)$. Suppose now that $\mathcal{A}$ is an arbitrary adversary playing the $PrvInd_{\Phi_a}$ game and let us construct $\mathcal{B}$ as an adversary playing the $PrvInd_{\Phi_b}$ game and using $\mathcal{A}$ as a subroutine. The latter adversary $\mathcal{B}$ tells the first adversary $\mathcal{A}$ to start the game. Adversary $\mathcal{A}$ chooses its input (f_0, f_1, x_0, x_1) which it wants to send to GARBLE procedure, which now in fact the adversary $\mathcal{B}$ pretends to be. Adversary $\mathcal{B}$ forwards the input from $\mathcal{A}$ to GARBLE procedure in $PrvInd_{\Phi_b}$ game. Adversary $\mathcal{B}$ receives an output (F,X,d) or $\perp$ from GARBLE. Now, if $\Phi_b(f_0) \neq \Phi_b(f_1)$, adversary $\mathcal{B}$ sends $\perp$ to $\mathcal{A}$. This is the normal answer: According to our assumption, $\Phi_b(f_0) \neq \Phi_b(f_1) \Rightarrow \Phi_a(f_0) \neq \Phi_a(f_1)$ and hence adversary $\mathcal{A}$ should receive $\perp$ also from its genuine GARBLE procedure. Otherwise, adversary $\mathcal{B}$ forwards the response from

proc $\texttt{GARBLE}(f_0, f_1, x_0, x_1)$ Game $\text{PrvInd}_{\mathcal{G},\Phi}$ $b \leftarrow \{0,1\}$ **if** $\Phi(f_0) \neq \Phi(f_1)$ **then return** $\perp$ **if** $\{x_0, x_1\} \not\subseteq \{0,1\}^{f_0.n}$ **then return** $\perp$ **if** $\texttt{ev}(f_0, x_0) \neq \texttt{ev}(f_1, x_1)$ **then return** $\perp$ $(F, e, d) \leftarrow \texttt{Gb}(1^k, f_b)$; $X \leftarrow \texttt{En}(e, x_b)$; **return** (F, X, d)	**proc** $\texttt{GARBLE}(f, x)$ Game $\text{PrvSim}_{\mathcal{G},\Phi,\mathcal{S}}$ $b \leftarrow \{0,1\}$ **if** $x \notin \{0,1\}^{f.n}$ **then return** $\perp$ **if** $b = 1$ **then** $(F, e, d) \leftarrow \texttt{Gb}(1^k, f)$; $X \leftarrow \texttt{En}(e, x)$ **else** $y \leftarrow \texttt{ev}(f, x)$; $(F, X, d) \leftarrow \mathcal{S}(1^k, y, \Phi(f))$ **return** (F, X, d)
proc $\texttt{GARBLE}(f_0, f_1, x_0, x_1)$ Game $\text{ObvInd}_{\mathcal{G},\Phi}$ $b \leftarrow \{0,1\}$; **if** $\Phi(f_0) \neq \Phi(f_1)$ **then return** $\perp$ **if** $\{x_0, x_1\} \not\subseteq \{0,1\}^{f_0.n}$ **then return** $\perp$ $(F, e, d) \leftarrow \texttt{Gb}(1^k, f_b)$; $X \leftarrow \texttt{En}(e, x_b)$; **return** (F, X)	**proc** $\texttt{GARBLE}(f, x)$ Game $\text{ObvSim}_{\mathcal{G},\Phi,\mathcal{S}}$ $b \leftarrow \{0,1\}$ **if** $x \notin \{0,1\}^{f.n}$ **then return** $\perp$ **if** $b = 1$ **then** $(F, e, d) \leftarrow \texttt{Gb}(1^k, f)$; $X \leftarrow \texttt{En}(e, x)$ **else** $(F, X) \leftarrow \mathcal{S}(1^k, \Phi(f))$ **return** (F, X)
proc $\texttt{FINALIZE}(b, b')$ Game $\text{PrvInd}_{\mathcal{G},\Phi}$, Game $\text{PrvSim}_{\mathcal{G},\Phi,\mathcal{S}}$, Game $\text{ObvInd}_{\mathcal{G},\Phi}$, Game $\text{ObvSim}_{\mathcal{G},\Phi,\mathcal{S}}$ **return** $b = b'$	
proc $\texttt{GARBLE}(f, x)$ Game $\text{Aut}_{\mathcal{G}}$ $(F, e, d) \leftarrow \texttt{Gb}(1^k, f)$; $x \leftarrow \texttt{En}(e, x)$ **return** (F, X)	**proc** $\texttt{FINALIZE}(Y)$ Game $\text{Aut}_{\mathcal{G}}$ **return** $\texttt{De}(d, Y) \neq \perp$ **and** $Y \neq \texttt{Ev}(F, X)$

Table I: The games defining the different security notions

its GARBLE to $\mathcal{A}$, who then sends its answer b' to $\mathcal{B}$. The adversary $\mathcal{B}$ answers the same b' in its $PrvInd_{\Phi_b}$ game.

Let us now consider the winning probabilities and advantages of both adversaries in their games. The behavior of adversaries $\mathcal{A}$ and $\mathcal{B}$ are the same at every step of the game: the inputs are the same, and the answers are the same. Therefore the probability of the answer b' being the correct one must be the same in both games. Hence the advantages of both adversaries are also equal. Because $\mathcal{G} \in \texttt{GS}(prv.ind, \Phi_b)$ the advantage of $\mathcal{B}$ in $PrvInd_{\Phi_b}$ game is negligible. Thus, the advantage of $\mathcal{A}$ is also negligible, and $\mathcal{G} \in \texttt{GS}(prv.ind, \Phi_a)$, which proves the claim.

For the second part, let us assume that there exists an efficient conversion g from the side-information function Φ_a into Φ_b. Our objective is to prove under these assumptions that $\texttt{GS}(prv.sim, \Phi_b) \subseteq \texttt{GS}(prv.sim, \Phi_a)$.

To do this, assume that $\mathcal{G} \in \texttt{GS}(prv.sim, \Phi_b)$. This means that for every polynomial time adversary $\mathcal{A}'$ there exists a simulator $\mathcal{S}$ such that the advantage of $\mathcal{A}'$ is negligible in $PrvSim_{\mathcal{G},\Phi_b,\mathcal{S}}$ game.

Let $\mathcal{A}$ be an arbitrary adversary playing $PrvSim_{\mathcal{G},\Phi_a,\mathcal{S}}$ games. Similarly to the first part of the proof, let $\mathcal{B}$ be an adversary who plays $PrvSim_{\mathcal{G},\Phi_b,\mathcal{S}}$ games by emulating $\mathcal{A}$, i.e. behaving just like $\mathcal{A}$ would behave in corresponding $PrvSim_{\mathcal{G},\Phi_a,\mathcal{S}}$ games. More precisely, by emulation of $\mathcal{A}$ we mean the following. First, adversary $\mathcal{B}$ tells $\mathcal{A}$ to start its game. Adversary $\mathcal{B}$ receives the GARBLE input (f, x) from $\mathcal{A}$, after which $\mathcal{B}$ forwards this input to its own GARBLE. This procedure returns (F, X, d) or $\perp$ to $\mathcal{B}$, who now consults adversary $\mathcal{A}$ by giving this output to him. Now, $\mathcal{A}$ returns b' to $\mathcal{B}$, who chooses the same b' as its own return value.

The assumption $\mathcal{G} \in \texttt{GS}(prv.sim, \Phi_b)$ implies that there exists a simulator $\mathcal{S}_{hard}$ such that the advantage of $\mathcal{B}$ is negligible in $PrvSim_{\mathcal{G},\Phi_b,\mathcal{S}_{hard}}$ game. Now, we define another simulator $\mathcal{S}'_{hard}$ by $\mathcal{S}'_{hard}(1^k, y, \Phi_a(f)) = \mathcal{S}_{hard}(1^k, y, g(\Phi_a(f)))$. First of all, $\mathcal{S}'_{hard}$ is polynomial time, because the conversion g is efficient and $\mathcal{S}_{hard}$ is a polynomial time simulator. Secondly, the win probability of $\mathcal{B}$ in its own $PrvSim_{\mathcal{G},\Phi_b,\mathcal{S}_{hard}}$ game is the same as the win probability that $\mathcal{A}$ has in the $PrvSim_{\mathcal{G},\Phi_a,\mathcal{S}'_{hard}}$ game, which implies equal advantages. By assumption, the advantage of $\mathcal{B}$ was negligible, and so is the advantage of $\mathcal{A}$ by the above argument. Now we have found a simulator against which $\mathcal{A}$ has a negligible advantage. □

NOTE: For example, $\Phi_a = \Phi_{topo}$ and $\Phi_b = \Phi_{size}$ satisfy the condition $(*)$.

Theorem 2: Let Φ_a and Φ_b be two different side-information functions satisfying the above condition $(*)$. Then the following inclusion holds: $\texttt{GS}(obv.ind, \Phi_b) \subseteq \texttt{GS}(obv.ind, \Phi_a)$. If we additionally assume that there exists a polynomial time function g such that $g(\Phi_a(f)) = \Phi_b(f)$ then we have also $\texttt{GS}(obv.sim, \Phi_b) \subseteq \texttt{GS}(obv.sim, \Phi_a)$.

Proof: The proof is similar to that of the previous theorem. □

The next four theorems consider non-inclusions of the form $A \not\subseteq B$ between sets of garbling schemes. In all cases we make an assumption that the set A is non-empty. The following two propositions provide a generalization to Propositions 5 and 7 in paper [3].

Theorem 3: For all Φ and for $\texttt{ev} = \texttt{ev}_{circ}$, we have $\texttt{GS}(obv.sim, \Phi) \bigcap \texttt{GS}(\texttt{ev}) \not\subseteq \texttt{GS}(prv.ind, \Phi)$.

Proof: Let $\mathcal{G} = (\texttt{Gb}, \texttt{En}, \texttt{De}, \texttt{Ev}, \texttt{ev}) \in \texttt{GS}(obv.sim, \Phi) \bigcap \texttt{GS}(\texttt{ev})$. Let us construct another garbling scheme $\mathcal{G}' = (\texttt{Gb}', \texttt{En}, \texttt{De}', \texttt{Ev}, \texttt{ev})$ such that $\mathcal{G}' \in \texttt{GS}(obv.sim, \Phi) \bigcap \texttt{GS}(\texttt{ev})$ but $\mathcal{G}' \notin \texttt{GS}(prv.ind, \Phi)$. The construction is as follows: The function $\texttt{Gb}'(1^k, f)$ picks $(F, e, d) \leftarrow \texttt{Gb}(1^k, f)$ and returns $(F, e, d||e)$. Let $\texttt{De}'(d||e, Y) = \texttt{De}(d, Y)$. Including e in the description of the decoding function does not harm $obv.sim$ security, because the adversary is given only (F, X) by the GARBLE procedure in the $obv.sim$ game. Thus $\mathcal{G}'$ inherits the $obv.sim$ security from $\mathcal{G}$.

On the other hand, $\mathcal{G}'$ is not $prv.ind$ secure. Adversary $\mathcal{A}$ makes a query (f_0, f_1, x_0, x_1), where $f_0 = f_1 = \texttt{AND}$ and $x_0 = 00, x_1 = 01$. This choice is fine for the PrvInd game, since $\texttt{ev}(f_0, x_0) = 0 = \texttt{ev}(f_1, x_1)$. Now, the adversary computes $X_0 = \texttt{En}(e, x_0)$ and $X_1 = \texttt{En}(e, x_1)$, which must be different because of the non-degeneracy condition (see Section 2). Then he/she compares these two with the garbled

input X received from GARBLE. This comparison now reveals which of the inputs, x_0 or x_1, was used. □

Theorem 4: For all Φ and for $\mathtt{ev} = \mathtt{ev}_{circ}$, we have $\mathtt{GS}(aut) \bigcap \mathtt{GS}(\mathtt{ev}) \nsubseteq \mathtt{GS}(prv.ind, \Phi) \bigcup \mathtt{GS}(obv.ind, \Phi)$.

Proof: Let $\mathcal{G} = (\mathtt{Gb}, \mathtt{En}, \mathtt{De}, \mathtt{Ev}, \mathtt{ev}) \in \mathtt{GS}(aut) \bigcap \mathtt{GS}(\mathtt{ev})$. Let us construct a garbling scheme $\mathcal{G}' = (\mathtt{Gb}, \mathtt{En}', \mathtt{De}, \mathtt{Ev}', \mathtt{ev})$ such that $\mathcal{G}' \in \mathtt{GS}(aut, \Phi) \bigcap \mathtt{GS}(\mathtt{ev})$ but $\mathcal{G}' \notin \mathtt{GS}(prv.ind, \Phi) \bigcup \mathtt{GS}(obv.ind, \Phi)$. The construction is as follows: We define that $\mathtt{Ev}'(F, X||x)) = \mathtt{Ev}(F, X)$, $\mathtt{En}'(e, x) = \mathtt{En}(e, x)||x = X||x$.

The new encoding function $\mathtt{En}'$ and evaluation function $\mathtt{Ev}'$ do not harm $aut-$security, since the adversary has chosen the function f and its input x. On the other hand, appending x to the encoding harms both obliviousness and privacy: In both games the adversary chooses the function f in such a way that $ev(f, \cdot)$ is not injective. This is possible because it is assumed that $\mathtt{ev} = \mathtt{ev}_{circ}$.

In both PrvInd and ObvInd game the adversary chooses inputs x_0, x_1 such that $x_0 \neq x_1$ and $\mathtt{ev}(f, x_0) = \mathtt{ev}(f, x_1)$. Now, the encoding $X||x_b$ reveals which of the inputs was used. □

The following two results provide parallel results compared to Propositions 8 and 9 in [3].

Theorem 5: Let P be a one-way permutation in the set of all functions f. Then, for $\Phi_P(f) = P(f)$ and for any $\mathtt{ev}$, $\mathtt{GS}(obv.ind, \Phi_P) \bigcap \mathtt{GS}(\mathtt{ev}) \nsubseteq \mathtt{GS}(obv.sim, \Phi_P)$.

Proof: Let $\mathcal{G} = (\mathtt{Gb}, \mathtt{En}, \mathtt{De}, \mathtt{Ev}, \mathtt{ev}) \in \mathtt{GS}(obv.ind, \Phi_P) \bigcap \mathtt{GS}(\mathtt{ev})$. We construct a new garbling scheme $\mathcal{G}' = (\mathtt{Gb}', \mathtt{En}, \mathtt{De}, \mathtt{Ev}', \mathtt{ev})$ such that $\mathcal{G}' \in \mathtt{GS}(obv.ind, \Phi_P) \bigcap \mathtt{GS}(\mathtt{ev})$ but $\mathcal{G}' \notin \mathtt{GS}(obv.sim, \Phi_P)$.

The construction is the following. The algorithm $\mathtt{Gb}'(1^k, f)$ picks $(F, e, d) \leftarrow \mathtt{Gb}(1^k, f)$ and returns $(F||f, e, d)$. Let $\mathtt{Ev}'(F||f, X)$ return $\mathtt{Ev}(F, X)$. First of all, we claim that the constructed garbling scheme is $obv.ind$ secure over Φ_P. The reasoning goes as follows. The adversary $\mathcal{A}$ sends (f_0, f_1, x_0, x_1) to its GARBLE. For the answer not being $\perp$ it must be that $\Phi_P(f_0) = \Phi_P(f_1)$, and hence $P(f_0) = P(f_1)$ by the definition of Φ_P. Since P is a one-way permutation, $f_0 = f_1$ must hold. Thus prepending f to the description of F does not harm $obv.ind$ security.

However, $\mathcal{G}'$ is not $obv.sim$ secure over Φ_P. We introduce an adversary $\mathcal{B}$ that breaks the $obv.sim$ security with respect to any PT simulator. The adversary chooses (f, x) to be sent to the GARBLE procedure in ObvSim game. Now, if the challenge bit b in the game is 0, the simulator $\mathcal{S}$ is called to produce $(F||f, X)$ from $(1^k, \Phi_P(f))$. However, the PT simulator manages to produce exactly the right function f with negligible probability, because $\Phi_P = P$ is a one-way permutation. In other words, this means that the adversary $\mathcal{B}$ will almost always detect from the parameter $F||f$ whether the simulator was used or not. □

Theorem 6: Let P be a one-way permutation in the set of all functions f and let $\Phi_P(f) = P(f)$ while $\mathtt{ev}$ is arbitrary. Assume that there exist x and y for which $\Phi_P(f) = P(f)$ is one-way even when restricted to functions f such that $y = \mathtt{ev}(f, x)$. Then $\mathtt{GS}(prv.ind, \Phi_P) \bigcap \mathtt{GS}(\mathtt{ev}) \nsubseteq \mathtt{GS}(prv.sim, \Phi_P)$.

Proof: Let $\mathcal{G} = (\mathtt{Gb}, \mathtt{En}, \mathtt{De}, \mathtt{Ev}, \mathtt{ev}) \in \mathtt{GS}(prv.ind, \Phi_P) \bigcap \mathtt{GS}(\mathtt{ev})$. We construct a new garbling scheme $\mathcal{G}' = (\mathtt{Gb}', \mathtt{En}, \mathtt{De}, \mathtt{Ev}', \mathtt{ev})$ such that $\mathcal{G}' \in \mathtt{GS}(prv.ind, \Phi_P) \bigcap \mathtt{GS}(\mathtt{ev})$ but $\mathcal{G}' \notin \mathtt{GS}(prv.sim, \Phi_P)$.

The construction is similar to that of the previous proof. The algorithm $\mathtt{Gb}'(1^k, f)$ picks $(F, e, d) \leftarrow \mathtt{Gb}(1^k, f)$ and returns $(F||f, e, d)$. Let $\mathtt{Ev}'(F||f, X)$ return $\mathtt{Ev}(F, X)$. First of all, the constructed garbling scheme is $prv.ind$ secure over Φ_P by exactly the same reasoning as in the previous proof.

However, $\mathcal{G}'$ is not $prv.sim$ secure over Φ_P. We prove this by introducing an adversary $\mathcal{B}$ having a non-negligible advantage in the $PrvSim_{\Phi_P}$ game. By the assumption, there exist x and y such that $\Phi_P(f)$ is still one-way, when restricted to f such that $y = \mathtt{ev}(f, x)$. Thus the adversary $\mathcal{B}$ can choose (f, x) satisfying $y = \mathtt{ev}(f, x)$ to be sent to the GARBLE procedure. Now, if the challenge bit b in the game is 0, the simulator $\mathcal{S}$ is called to produce $(F||f, X, d)$ from $(1^k, y, \Phi_P(f))$, where $y = \mathtt{ev}(f, x)$. However, the polynomial time simulator manages to produce exactly the right function f with negligible probability, because $\Phi_P = P$ is an injective one-way function. In other words, this means that the adversary $\mathcal{B}$ will almost always detect from $F||f$ whether the simulator was used or not. □

The following two propositions provide new results for garbling scheme classes in [3].

Theorem 7: If the function $h : (f, x) \mapsto (\Phi(f), \mathtt{ev}(f, x))$ is injective, then $\mathtt{GS}(\mathtt{ev}) \subseteq \mathtt{GS}(prv.ind, \Phi)$.

Proof: Let $\mathcal{G} = (\mathtt{Gb}, \mathtt{En}, \mathtt{De}, \mathtt{Ev}, \mathtt{ev})$ be an arbitrary garbling scheme over side-information function Φ. Let $\mathcal{B}$ be an adversary playing the $PrvInd_{\Phi}$ game. The adversary sends (f_0, f_1, x_0, x_1) to the GARBLE procedure of this game. For the output not being $\perp$ it must be that

$$\Phi(f_0) = \Phi(f_1), \mathtt{ev}(f_0, x_0) = \mathtt{ev}(f_1, x_1).$$

But by injectivity of h this implies

$$\begin{aligned} h(f_0, x_0) &= (\Phi(f_0), \mathtt{ev}(f_0, x_0)) \\ &= (\Phi(f_1), \mathtt{ev}(f_1, x_1)) = h(f_1, x_1) \\ &\Rightarrow (f_0, x_0) = (f_1, x_1). \end{aligned}$$

This in turn is equivalent to $f_0 = f_1$ and $x_0 = x_1$, meaning that the advantage of the adversary $\mathcal{B}$ in this game will be equal to 0. This completes the proof. □

Theorem 8: If the function $\mathtt{ev}$ is injective and efficiently invertible (i.e. given $y = \mathtt{ev}(f', x')$, f and x such that $\mathtt{ev}(f, x) = y$ can be found in polynomial time), then $\mathtt{GS}(\mathtt{ev}) \subseteq \mathtt{GS}(prv.sim, \Phi)$.

Proof: Let $\mathcal{G} = (\mathtt{Gb}, \mathtt{En}, \mathtt{De}, \mathtt{Ev}, \mathtt{ev})$ be an arbitrary garbling scheme over side-information function Φ. Let $\mathcal{B}$ be an adversary playing the $PrvSim_{\Phi}$ game. The adversary sends (f, x) to the GARBLE procedure of this game. But now, if the challenge bit $b = 0$, the simulator can always find the right f and x to be garbled because $y = \mathtt{ev}(f, x)$ can

be inverted efficiently and $\mathtt{ev}(f,x) = \mathtt{ev}(f',x')$ guarantees $f = f', x = x'$. This means that no matter what the challenge bit was, in both cases, $b = 0$ or $b = 1$, the pair (f, x) becomes garbled correctly because the simulator that knows f and x is able to use the normal garbling method. This means that the advantage of the adversary $\mathcal{B}$ in this game equals 0, proving the inclusion $\mathtt{GS}(\mathtt{ev}) \subseteq \mathtt{GS}(prv.sim, \Phi) \bigcap \mathtt{GS}(\mathtt{ev})$. □

IV. New classes of garbling schemes

In [3], the definitions and relations between different security types were, at least to some extent, based on intuition about what is meant by a garbling scheme that achieves privacy, obliviousness or authenticity, and the intuition was modeled as a game. In this section we consider the games defined in paper [3] from another point of view; we consider them purely as games, and try to achieve new results by modifying the existing game definitions in certain ways explained later.

The first modification we make is that in the indistinguishability model, the PrvInd game will be modified to the direction of ObvInd game by removing the decryption key d from the return value (F, X, d). The same end result can be obtained by tightening the ObvInd game by adding the evaluation test $\mathtt{ev}(f_0, x_0) \stackrel{?}{=} \mathtt{ev}(f_1, x_1)$ in it. In the absence of a better name we call the new class *ModInd*. The second modification concerns the PrvSim game, in which we again ease the requirements by removing the decryption key d from the return value (F, X, d). In ObvSim game, adding y to the input of the simulator $\mathcal{S}$ will lead to the same intermediate form as above. The new class shall be named *ModSim*.

Another modification is obtained by relaxing the PrvInd game by removing the evaluation test. This can also be achieved by adding d to the output (F, X) in ObvInd game. A similar modification in Sim side is to leave y out from the input of the simulator in PrvSim game, or add d to (F, X) in ObvSim game. The former modification is called *ModInd2* and the latter is called *ModSim2*.

The finalization procedure is not modified in any of these games.

A. Applications

Before proceeding to the descriptions of our modifications, it is convenient to discuss the possible applications that could utilize garbling schemes and more specifically, our modified security models. One typical example is outsourcing of a complex computation to a service in the cloud. In many cases the input data or the algorithm (or both of them) is privacy-sensitive data and should not be revealed to the party running the cloud service. With garbling schemes achieving different types of security, we can hide different amount of this information. In order to have an idea which type of security is most appropriate in different situations, let us take a closer look at which kind of information is revealed by a garbling scheme belonging to a specific security class.

Let the function f represent the algorithm, x represents the privacy-sensitive input data and $f(x) = y$ represents the output of the algorithm. These are all garbled with some garbling scheme, and the garbled function and garbled input are given to the server, which computes the garbled output. It depends on the garbling scheme how much the server is allowed to know about f, x and $f(x)$. It is worth noting that whatever the model of security is, the original function f is not known for the server, only the side-information function $\Phi(f)$ is. The following list provides the central differences between the models.

- **obv.sim**: Garbling does not reveal x, f or $f(x)$ to the server.
- **prv.sim**: The server is allowed to get $f(x)$ but not x or f.
- **mod.sim**: When computing $y_1 = f(x_1)$ and $y_2 = f(x_2)$, the server is allowed to find out whether $y_1 = y_2$ or not.

There are situations in which the output data is not sensitive and can be revealed to the party maintaining the cloud service. According to the list above, a prv.sim secure garbling scheme is then appropriate. Also garbling schemes of the two other types may fit the situation except if the server needs the output in further computations. The issue is that the output will remain garbled in the cloud. Of course, further computations could also be garbled but this arrangement would significantly and unnecessarily add the total complexity of computation.

If the output is sensitive data, an obv.sim secure scheme suits. Our modified model mod.sim is suitable as well except in some cases where the number and/or distribution of different output values may reveal too much information. On the other hand, mod.sim can actually be modified to apply to these cases as well. Instead of considering inputs x, the modified scheme would take inputs $x||i$ where i is for example an ever-increasing counter. The procedure then returns $\mathtt{ev}(f, x)||i$ as the output. The counter at the end of the evaluation result will now make sure that each output appears only once. According to the previous discussion, mod.sim secure garbling schemes can be used in the same applications as prv.sim or obv.sim secure schemes. In the following section we will prove that it is at least as easy to find a mod.sim secure scheme as it is to find an obv.sim or a prv.sim secure scheme. In conclusion, the modified security model mod.sim covers almost all applications except some esoteric cases.

B. Definitions and results

Next we give the formal definition of ModInd and ModSim games. Then we continue by proving some results concerning the new classes of garbling schemes that are secure with respect to these games.

The following proposition shows that mod.ind security is at least as easy to reach as prv.ind security or obv.ind security.

Theorem 9: $\mathtt{GS}(prv.ind, \Phi) \bigcup \mathtt{GS}(obv.ind, \Phi) \subseteq \mathtt{GS}(mod.ind, \Phi)$.

Proof: First suppose that $\mathcal{G}$ is a prv.ind secure garbling scheme. Dropping the decryption key d out of the output of GARBLE procedure does not increase the winning chances of any adversary.

proc $\texttt{GARBLE}(f_0, f_1, x_0, x_1)$ $\quad$ ModInd$_{\mathcal{G},\Phi}$	**proc** $\texttt{GARBLE}(f, x)$ $\quad$ ModSim$_{\mathcal{G},\Phi,\mathcal{S}}$
$b \leftarrow \{0,1\}$	$b \leftarrow \{0,1\}$
if $\Phi(f_0) \neq \Phi(f_1)$ **then return** $\perp$	**if** $x \notin \{0,1\}^{f.n}$ **then return** $\perp$
if $\{x_0, x_1\} \not\subset \{0,1\}^{f_0.n}$ **then return** $\perp$	**if** $b = 1$ **then** $(F, e, d) \leftarrow \texttt{Gb}(1^k, f); X \leftarrow \texttt{En}(e, x)$
if $\texttt{ev}(f_0, x_0) \neq \texttt{ev}(f_1, x_1)$ **then return** $\perp$	**else** $y \leftarrow \texttt{ev}(f, x)$; $(F, X) \leftarrow \mathcal{S}(1^k, y, \Phi(f))$
$(F, e, d) \leftarrow \texttt{Gb}(1^k, f_b)$; $X \leftarrow \texttt{En}(e, x_b)$	**return** (F, X)
return (F, X)	

Table II: The modified GARBLE procedures in Ind and Sim games

Secondly, suppose that $\mathcal{G}$ is an obv.ind secure scheme. Now, following the specification of ModInd GARBLE procedure, the adversary receives $\perp$ on all inputs whose evaluations $\texttt{ev}(f_0, x_0)$ and $\texttt{ev}(f_1, x_1)$ are not equal. However, this evaluation equality test is not a part of ObvInd game. Hence, even though GARBLE procedure in ObvInd game returns an output different from $\perp$, the corresponding procedure in ModInd game might return $\perp$. Otherwise the games are identical. Adversaries of both games are able to find out beforehand whether the GARBLE procedure returns $\perp$ and therefore the adversary in ModInd game does not receive . Therefore, the advantage of adversary playing the ModInd game cannot be better than the advantage of a corresponding adversary in ObvInd game. According to the assumption, the advantage in the ObvInd game is negligible, and thus the advantage in ModInd game must also be negligible. □

Theorem 10: $\texttt{GS}(prv.sim, \Phi) \bigcup \texttt{GS}(obv.sim, \Phi) \subseteq \texttt{GS}(mod.sim, \Phi)$.

Proof: First, suppose that garbling scheme $\mathcal{G}$ is prv.sim secure. As in the PrvInd case, omitting the decryption key d from (F, X, d) does not increase the winning probability of an adversary playing the modified game.

Secondly, suppose that the garbling scheme $\mathcal{G}$ belongs to the set of ObvSim secure schemes. In the ModSim game, the simulator's additional input y cannot make its work of producing a good output (F, X) more difficult. Let us explain in more details why this is the case.

Let $\mathcal{A}$ be an arbitrary adversary playing the ModSim game. Let $\mathcal{A}'$ be the corresponding adversary playing the ObvSim game: adversary $\mathcal{A}'$ behaves in ObvSim game exactly in the same way as $\mathcal{A}$ behaves in ModSim game. According to our assumption, there is a simulator $\mathcal{S}'$ such that the advantage of $\mathcal{A}'$ is negligible. Now, we construct a simulator $\mathcal{S}$ for the ModSim game. The simulator $\mathcal{S}$ will totally omit the additional input y and call simulator $\mathcal{S}'$ to produce an output to the adversary $\mathcal{A}$. Now, this simulator makes the advantage of adversary $\mathcal{A}$ negligible, because the adversary $\mathcal{A}$ behaves just like $\mathcal{A}'$ and also the simulators in both games behave identically. This completes the proof. □

As mentioned in the introductory part of this section, we have created four modifications to the prv.ind and prv.sim models in total, of which we have now covered two. In the rest of this section, we first give the descriptions of the two other modified games and provide some results concerning them. Finally, we give a diagram including the new models and their relations.

After these two definitions, we will now provide a result about mod.ind2 and mod.sim2.

Theorem 11: Assume that the following condition holds:

$$(\forall f_0, f_1)\ (\forall x_0, x_1): \qquad \textbf{(Condition } (**)\textbf{)}$$
$$\Phi(f_0) = \Phi(f_1) \Rightarrow \texttt{ev}(f_0, x_0) = \texttt{ev}(f_1, x_1).$$

Then $\texttt{GS}(mod.ind2, \Phi) = \texttt{GS}(prv.ind, \Phi)$. Otherwise $\texttt{GS}(mod.ind2, \Phi) = \emptyset$.

Proof: Suppose first that $(**)$ does not hold. Then the adversary can choose f_0, f_1, x_0, x_1 such that $\texttt{ev}(f_0, x_0) \neq \texttt{ev}(f_1, x_1)$ but still $\Phi(f_0) = \Phi(f_1)$ holds. In this case, the adversary will always win the game, because $b = 0$ if and only if $\texttt{ev}(f_0, x_0) = \texttt{De}(d, \texttt{Ev}(F, X))$, and thus the advantage would not satisfy the negligibility condition, and no garbling scheme is secure.

Now assume that $(**)$ holds. Then the the adversary in the ModInd2 game has no choice other than choosing f_0, f_1, x_0, x_1 such that $\Phi(f_0) = \Phi(f_1)$ for not receiving $\perp$ which now implies that $\texttt{ev}(f_0, x_0) = \texttt{ev}(f_1, x_1)$ must hold. It follows that the sets of PrvInd secure garbling schemes and ModInd2 secure garbling schemes must be equal. This completes the proof. □

Theorem 12: For any Φ, the inclusion $\texttt{GS}(mod.sim2, \Phi) \subseteq \texttt{GS}(prv.sim, \Phi)$ holds. If $(**)$ holds, and Φ is efficiently invertible (i.e. given $\phi = \Phi(f')$, a function f can be found in polynomial time such that $\Phi(f) = \phi$), then the equality $\texttt{GS}(mod.sim2, \Phi) = \texttt{GS}(prv.sim, \Phi)$ holds. Finally, if $(**)$ does not hold, then $\texttt{GS}(mod.sim2, \Phi) = \emptyset$.

Proof: The difference between the ModSim2 and PrvSim games is, that in PrvSim game the simulator gets $y = \texttt{ev}(f, x)$ as input, whereas the simulator in ModSim2 game does not. This means that simulator's task of creating a good output (F, X, d) in PrvSim game is not harder than the task of the simulator in the other game. Therefore, the advantage of an adversary in PrvSim game cannot be better than in ModSim2 game. This proves the first claim.

For the second part, suppose that $(**)$ holds and Φ is efficiently invertible. Even though y is not provided to the simulator, it still is able to produce (F', X', d') such that the adversary has no better chances than guessing to win the ModSim2 game. Namely, the simulator creates from $\Phi(f)$ such a function f' that $\Phi(f) = \Phi(f')$, and it then creates any suitable input x' to the function f'. Now, because of the condition $(**)$, the equality $\texttt{ev}(f, x) = \texttt{ev}(f', x')$ must hold and hence the simulator always learns the right y. This means that the setting in this new, modified game actually is exactly the same as in PrvSim game.

Finally suppose that $(**)$ does not hold. In the modified game, the adversary can choose f and x such that there

proc $\texttt{GARBLE}(f_0, f_1, x_0, x_1)$ $\quad$ ModInd2$_{\mathcal{G},\Phi}$	**proc** $\texttt{GARBLE}(f, x)$ $\quad$ ModSim2$_{\mathcal{G},\Phi,\mathcal{S}}$
$b \leftarrow \{0,1\}$	$b \leftarrow \{0,1\}$
if $\Phi(f_0) \neq \Phi(f_1)$ **then return** $\perp$	**if** $x \notin \{0,1\}^{f.n}$ **then return** $\perp$
if $\{x_0, x_1\} \not\subset \{0,1\}^{f_0.n}$ **then return** $\perp$	**if** $b = 1$ **then** $(F, e, d) \leftarrow \texttt{Gb}(1^k, f); X \leftarrow \texttt{En}(e, x)$
$(F, e, d) \leftarrow \texttt{Gb}(1^k, f_b); \quad X \leftarrow \texttt{En}(e, x_b)$	**else**$(F, X, d) \leftarrow \mathcal{S}(1^k, \Phi(f))$
return (F, X, d)	**return** (F, X, d)

Table III: Another modification of GARBLE procedure in Ind and Sim games

exists a function f' satisfying $f' \neq f$, $\Phi(f) = \Phi(f')$ and $\texttt{ev}(f, x) \neq \texttt{ev}(f', x')$ for some x'. Now the simulator has at most 50% chance to guess the correct f. If the guess was incorrect, distinguishing the simulated version from the actual garbled output is easy since the adversary is able to check if $\texttt{ev}(f, x) = \texttt{De}(d, \texttt{Ev}(F, X))$. Thus, no garbling scheme is ModSim2 secure. □

Corollary 1: The following inclusion holds: $\texttt{GS}(mod.sim2, \Phi) \subseteq \texttt{GS}(mod.ind2, \Phi)$.

Proof: The claim follows from Theorem 11 and Theorem 12 and Proposition 2 in [3]. □

NOTE: In practice, condition $(**)$ does not usually hold. Therefore, it is hard to imagine an application in which our second modification would have practical significance because of the above result.

The next theorem provides a relation between the modified simulation type and the modified indistinguishability type garbling schemes under our first modification.

Theorem 13: The following inclusion holds: $\texttt{GS}(mod.sim, \Phi) \subseteq \texttt{GS}(mod.ind, \Phi)$.

Proof: Let $\mathcal{G} = (\texttt{Gb}, \texttt{En}, \texttt{De}, \texttt{Ev}, \texttt{ev}) \in \texttt{GS}(mod.sim, \Phi)$. We need to prove that $\mathcal{G} \in \texttt{GS}(mod.ind, \Phi)$. Let $\mathcal{A}$ be the PT adversary playing the ModInd game. We construct a PT ModSim adversary $\mathcal{B}$ as follows. Let $\mathcal{B}$ run $\mathcal{A}$ as a subroutine. The latter makes its query f_0, f_1, x_0, x_1. Adversary $\mathcal{B}$ returns $\perp$ to $\mathcal{A}$ if $\Phi(f_0) \neq \Phi(f_1)$ or $\{x_0, x_1\} \not\subseteq \{0,1\}^{f_0.n}$ or $\texttt{ev}(f_0, x_0) \neq \texttt{ev}(f_1, x_1)$.

Regardless of whether $\mathcal{B}$ returned $\perp$ to $\mathcal{A}$ or not, adversary $\mathcal{B}$ picks $c \in \{0,1\}$ at random and makes its query to GARBLE with input f_c, x_c getting back (F, X) which is sent to adversary $\mathcal{A}$ in case $\perp$ was not sent earlier. In any case, adversary $\mathcal{A}$ returns a bit b' to adversary $\mathcal{B}$. The latter adversary now returns 1 if $\Phi(f_0) = \Phi(f_1)$, $\{x_0, x_1\} \subseteq \{0,1\}^{f_0.n}$, $\texttt{ev}(f_0, x_0) = \texttt{ev}(f_1, x_1)$ and $b' = c$ and 0 otherwise. Let $\mathcal{S}$ be any PT algorithm representing the simulator. Then there are two possible outcomes of the game:

1) If $\Phi(f_0) = \Phi(f_1)$, $\{x_0, x_1\} \subseteq \{0,1\}^{f_0.n}$ and $\texttt{ev}(f_0, x_0) = \texttt{ev}(f_1, x_1)$, then the input to the simulator $\mathcal{S}$ is the same regardless of c, or
2) $\Phi(f_0) \neq \Phi(f_1)$, $\{x_0, x_1\} \not\subset \{0,1\}^{f_0.n}$ or $\texttt{ev}(f_0, x_0) \neq \texttt{ev}(f_1, x_1)$ then adversary $\mathcal{B}$ always answers 0 regardless of b' received from adversary $\mathcal{A}$.

Let's analyze the win probabilities of both adversaries. First consider the case 2. Adversary $\mathcal{B}$ always answers 0, and there is 50% chance of it being the right answer, and hence the win probability of $\mathcal{B}$ is one half. The win probability of adversary $\mathcal{A}$ is the same: $\mathcal{A}$ does not get any information linked to the challenge bit, and thus its answer is as good as guessing but there is always 50% chance of answering right.

Next consider case 1. Now, there are two possibilities for challenge bit b. Suppose first that $b = 1$. In this case, adversary $\mathcal{B}$ wins if and only if $\mathcal{A}$ wins. On the other hand, if the challenge bit b equals 0, adversary $\mathcal{A}$ does not have any information because it is getting the same input regardless of c, so its answer is no better than a guess. Thus the win probability equals $\frac{1}{2}$. Furthermore, the adversary $\mathcal{A}$ wins if and only if adversary $\mathcal{B}$ loses, therefore $\Pr[\mathcal{B}\ wins] = \Pr[\mathcal{A}\ loses] = \frac{1}{2}$.

This case analysis above shows that in all cases $\Pr[\mathcal{B}\ wins] = \Pr[\mathcal{A}\ wins]$. Now continuing with $\Pr[\mathcal{A}\ wins]$ we obtain

$$\begin{aligned}\Pr[\mathcal{A}\ wins] &= \frac{1}{2}\Pr[\mathcal{A}\ wins|b=1] + \frac{1}{2}\Pr[\mathcal{A}\ wins|b=0] \\ &= \frac{1}{2}\cdot\left(\frac{1}{2} + \frac{1}{2}\cdot\mathbf{Adv}_{\mathcal{A}}\right) + \frac{1}{2}\cdot\frac{1}{2} \\ &= \frac{1}{2} + \frac{1}{4}\cdot\mathbf{Adv}_{\mathcal{A}}.\end{aligned}$$

By the definition of advantage of adversary $\mathcal{B}$ we have $\Pr[\mathcal{B}\ wins] = \frac{1}{2}\cdot\mathbf{Adv}_{\mathcal{B}} + \frac{1}{2}$ and therefore we obtain $\mathbf{Adv}_{\mathcal{A}} = 2\cdot\mathbf{Adv}_{\mathcal{B}}$. Now, since the $\mathbf{Adv}_{\mathcal{A}}$ is negligible according to the assumption, $\mathbf{Adv}_{\mathcal{B}}$ is also negligible. □

For our last theorem, we introduce a new condition:

The decryption key d can be efficiently computed from the tuple (F, X). **(Condition** $(***)$**)**

Theorem 14: The following inclusions hold: $\texttt{GS}(mod.ind2, \Phi) \subseteq \texttt{GS}(obv.ind, \Phi)$ and $\texttt{GS}(mod.sim2, \Phi) \subseteq \texttt{GS}(obv.sim, \Phi)$. If condition $(***)$ holds, then the classes are equal.

Proof: The difference between $ModInd2$ and $ObvInd$ (respectively $ModSim2$ and $ObvSim$) is that in ModInd2 game (in ModSim2 respectively) the adversary receives the decryption key d as an output from GARBLE together with F and X. This auxiliary output does not make the advantage smaller to the adversary in the modified games. The claim follows from this observation. □

These results complete the considerations about the possible relations between the new and old classes of garbling schemes. Results are collected into Figure 3.

If in addition to $(***)$ we require that $(\Phi, \texttt{ev})$ is efficiently invertible (i.e. given $y = \texttt{ev}(f', x')$ and $\phi = \Phi(f')$, f and x such that $y = \texttt{ev}(f, x)$ and $\Phi(f) = \phi$ can be found in polynomial time) and condition $(**)$ also holds, then all the

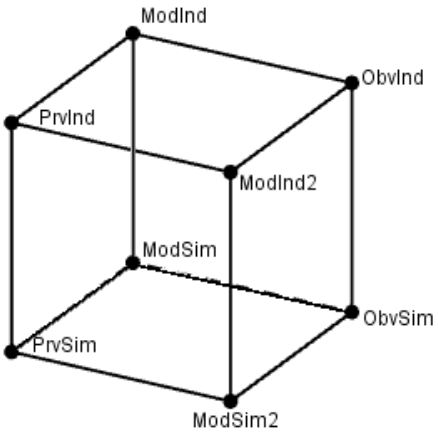


Figure 3: Inclusions between classes of garbling schemes

sets in the diagram collapse into one point: a garbling scheme that belongs to one security class will be secure also with respect to any other security model.

V. Conclusions

In this paper, we have considered different security classes of garbling schemes. Some of our results are obtained for the classes defined by Bellare, Hoang and Rogaway in [3]. We have also introduced new security classes and described their relation to the earlier classes. From these new classes, we see that the new classes $\mathtt{GS}(mod.ind, \Phi)$ and $\mathtt{GS}(mod.sim, \Phi)$ would be promising targets for future research - at least, it seems that these classes would have practical applications. Namely, our results show that all garbling schemes in the old obv-classes belong also to the new mod-classes, and therefore it is at least as easy to find a garbling scheme that is mod-secure. Moreover, it seems to be harder to find an application which would require obv-security but where mod-security would not suffice. The second new class sets too hard requirements for a secure garbling scheme and this class is practically always empty.

References

[1] Y. Aumann and Y. Lindell. Security against covert adversaries: Efficient protocols for realistic adversaries. *Proc. of TCC 2007*, 4392 of LNCS:137–156, 2007.

[2] D. Beaver, S. Micali, and P. Rogaway. The round complexity of secure protocols. *Proc. of the* 22^{nd} *STOC*, pages 503–513, 1990.

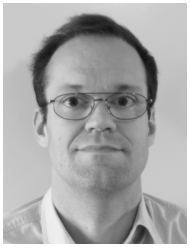
[3] M. Bellare, V. T. Hoang, and P. Rogaway. Foundations of garbled circuits. *Proc. of ACM Computer and Communications Security (CCS'12)*, 2012.

[4] M. Bellare and P. Rogaway. Code-based game-playing proofs and the security of triple encryption. *Advances in Cryptology, Proc. of Eurocrypt 2006*, 4004 of LNCS:409–426, 2006.

[5] Y. Lindell and B. Pinkas. A proof of security of Yao's protocol for secure two-party computation. *Electronic Colloquium on Computational Complexity*, TR04-063, 2004.

[6] Y. Lindell and B. Pinkas. A proof of security of Yao's protocol for secure two-party computation. *Journal of Cryptology*, 22(2):161–188, 2009.

[7] A. Yao. How to generate and exchange secrets. *Proc. of* 27^{th} *FOCS, 1986.*, pages 162–167, 1986.



Tommi Meskanen had his PhD in 2005. Since that he has been working as a researcher and lecturer at University of Turku. His main research interests are cryptography and public choice theory. His email address is tommes@utu.fi

Valtteri Niemi is a Professor of Mathematics at the University of Turku, Finland. Between 1997 and 2012 he was with Nokia Research Center in various positions, based in Finland and Switzerland. Niemi was also the chairman of the security standardization group of 3GPP during 2003-2009. His research interests include cryptography and mobile security. Valtteri can be contacted at valtteri.niemi@utu.fi.

Noora Nieminen is a doctoral student at Turku Centre for Computer Science, Department of Mathematics and Statistics at the University of Turku. Her research interests include cryptography and its applications. Contact her at nmniem@utu.fi.