Adaptively Secure Garbling with Applications to One-Time Programs and Secure Outsourcing

Mihir Bellare[1], Viet Tung Hoang[2], and Phillip Rogaway[2]

[1] Dept. of Computer Science and Eng., University of California, San Diego, USA
[2] Dept. of Computer Science, University of California, Davis, USA

Abstract. Standard constructions of garbled circuits provide only *static* security, meaning the input x is not allowed to depend on the garbled circuit F. But some applications—notably *one-time programs* (Goldwasser, Kalai, and Rothblum 2008) and *secure outsourcing* (Gennaro, Gentry, Parno 2010)—need *adaptive* security, where x may depend on F. We identify gaps in proofs from these papers with regard to adaptive security and suggest the need of a better abstraction boundary. To this end we investigate the adaptive security of *garbling schemes*, an abstraction of Yao's garbled-circuit technique that we recently introduced (Bellare, Hoang, Rogaway 2012). Building on that framework, we give definitions encompassing *privacy*, *authenticity*, and *obliviousness*, with either *coarse-grained* or *fine-grained* adaptivity. We show how adaptively secure garbling schemes support simple solutions for one-time programs and secure outsourcing, with privacy being the goal in the first case and obliviousness and authenticity the goal in the second.We give transforms that promote static-secure garbling schemes to adaptive-secure ones. Our work advances the thesis that conceptualizing garbling schemes as a first-class cryptographic primitive can simplify, unify, or improve treatments for higher-level protocols.

1 Introduction

Overview. Yao's garbled-circuit technique [10, 11, 18, 20, 21] has been extremely influential, engendering an enormous number of applications. Yet, at least in its conventional form, the technique provides only *static* security. Some applications, notably one-time programs [13] and secure outsourcing [9], require *adaptive* security.[1] In such cases Yao's technique can be enhanced in *ad hoc* ways, and the enhanced protocol incorporated into the higher-level application.

This paper provides a different approach. We create an abstraction for the goal of *adaptively secure garbling*. Via a single abstraction, we support a variety of applications in a simple and modular way. Let's look at two of the applications that motivate our work.

Two applications. *One-time programs* are due to Goldwasser, Kalai, and Rothblum (GKR) [13]. The authors aim to compile a program into one that

[1] In speaking of adversaries or security, *non-adaptive* and *dynamic* are common synonyms for what we are here calling *static* and *adaptive*.

X. Wang and K. Sako (Eds.): ASIACRYPT 2012, LNCS 7658, pp. 134–153, 2012.

can be executed just once, on an input of the user's choice. Unachievable in any "standard" model of computation, GKR assume what they call *one-time memory*. Their solution makes crucial use of Yao's garbled-circuit technique. Recognizing that this does not support adaptive queries, GKR embellish the method by a technique involving output-masking and n-out-of-n secret sharing.

In a different direction, *secure outsourcing* was formalized and investigated by Gennaro, Gentry, and Parno (GGP) [9]. Here a *client* transforms a function f into a function F that is handed to a *worker*. When, later, the client would like to evaluate f at x, he should be able to quickly map x to a garbled input X and give this to the worker, who will compute and return $Y = F(X)$. The client must be able to quickly reconstruct from this $y = f(x)$. He should be sure that the correct value was computed—the computation is *verifiable*—while the server shouldn't learn anything significant about x, including $f(x)$. GGP again make use of circuit garbling, and they again realize that they need something from it—its authenticity—that is a *novum* for this domain.

Issues. Assuming the existence of a one-way function, GKR [13] claim that their construction turns a (statically-secure) garbled circuit into a secure one-time program. We point to a gap in their proof, namely, the absence of a reduction showing that their simulator works based on the one-way function assumption. By presenting an example of a statically-secure garbled circuit that, under their transform, yields a program that is not one-time, we also show that the gap cannot be filled without changing either the construction or the assumption. The problem is that the GKR transform fails to ensure adaptive security of garbled circuits under the stated assumption.

Lindell and Pinkas (LP) [17] prove static security of a version of Yao's protocol assuming a semantically secure encryption scheme satisfying some extra properties (an elusive and efficiently verifiable range). GGP [9] build a one-time outsourcing scheme from the LP protocol, claiming to prove its security based on the same assumption as used in LP. We point to a gap in this proof arising from an implicit assumption of adaptive security of the LP construction.

We do not believe these are major problems for either work. In both cases, alternative ways to establish the the authors' main results already existed. Goyal, Ishai, Sahai, Venkatesan and Wadia [14] present an unconditional one-time compiler (no complexity-theoretic assumption is used at all), while Chung, Kalai and Vadhan [7] present secure outsourcing schemes based solely on FHE (garbled circuits are not employed). Our interpretation of the stated gaps is that they are symptoms of something else—a missing abstraction boundary. As recently argued by Bellare, Hoang and Rogaway (BHR) [4], it is useful and simplifying to see garbling not just as a technique, but as a first-class primitive. To do so, our earlier work defines syntax and security notions for *garbling schemes*, provides proven-correct solutions, then solves some example higher-level problems by employing a garbling scheme that satisfies the appropriate definition. But the security notions of BHR do not go far enough to handle what GKR or GGP need, since BHR deal only with static notions of security. The applications we

point to motivate the study of adaptive security for garbling schemes, while the gaps indicate that the issues may be more subtle than recognized.

Of course we communicated our findings to the GKR and GGP authors. GKR responded after a few weeks with an updated manuscript [12]. It modifies the claim from their original paper [13] to now claim that their transform works under the stronger assumption of a sub-exponentially hard one-way function. (This allows "complexity-leveraging," where a static adversary can guess the input that will be used by an adaptive adversary with a probability that, although exponentially-small, is enough under the stronger assumption.) GGP responded to acknowledge the gap and suggest that they would address it by assuming the LP construction, or some related realization of Yao's idea, already provides adaptive security.

Definitions. We now discuss our contributions in more depth. We start from the abstraction of a garbling scheme—the raw syntax—introduced by BHR [4]. That work gave multiple definitions sitting on top of this syntax, but all were for static adversaries, in the sense that the function f to garble and its input x are selected at the same time. We extend the definitions to adaptive ones, considering two flavors of adaptive security. With *coarse-grained* adaptive security the input x can depend on the garbled function F but x itself is atomic, provided all at once. With *fine-grained* adaptive security not only may x depend on the garbled function F, but individual bits of x can depend on the "tokens" the adversary has so-far learned.[2] We will see that coarse-grained adaptive security is what's needed for GGP's approach to secure outsourcing, while fine-grained adaptive security is what's needed for GKR's approach to one-time programs.

Orthogonal to adaptive security's granularity are the security aims themselves. Following BHR, we consider three different notions: privacy, obliviousness, and authenticity. This gives rise to nine different security notions: {prv, obv, aut} × {static, coarse, fine}. We compactly denote these prv, prv1, prv2, obv, obv1, obv2, aut, aut1, aut2. Informally, when a function f gets transformed into a garbled function F, an encoding function e, and a decoding function d, privacy ensures that F, d, and $X = e(x)$ don't reveal anything beyond $y = f(x)$ that shouldn't be revealed; obliviousness ensures that F and X don't reveal even y; and authenticity ensures that F and X don't enable the computation of a valid $Y \neq F(X)$. Privacy is the classical requirement, while obliviousness and authenticity are motivated by the application to secure outsourcing.

Our primary definitions for adaptive secrecy (prv1, prv2, obv1, obv2) are simulation-based. In the full version of this paper [3] we give indistinguishability-based counterparts as well. For static security this was already done by BHR, but it was not clear how to lift those definitions to the adaptive setting.

Relations. We explore the provable-security relationships among our definitions. As expected, the simulation-based definitions imply indistinguishability-based

[2] Fine-grained adaptive security requires the garbling scheme be *projective*: the garbled version of each $x = x_1 \cdots x_n \in \{0,1\}^n$ must be $(X_1^{x_1}, \ldots, X_n^{x_n})$ for some vector of $2n$ strings $(X_1^0, X_1^1, \ldots, X_n^0, X_n^1)$. Typical garbling schemes have this structure.

ones (namely, prv1 ⇒ prv1.ind, prv2 ⇒ prv2.ind, obv1 ⇒ obv1.ind, and obv2 ⇒ obv2.ind). But none of the converse statements hold. BHR had earlier shown that, for the static setting, the converse statements *do* hold as long as the associated side-information function[3] is efficiently invertible. In contrast, we show that, for adaptive privacy, this condition still won't guarantee equivalence of simulation-based and indistinguishability-based notions. (For obliviousness, it is true that obv1.ind ⇒ obv1 and obv2.ind ⇒ obv2 if Φ is efficiently invertible.) The results are our main reason to focus on simulation-based definitions for adaptive privacy. The full version [3] paints a complete picture of the relations among our basic definitions. Apart from the trivial relations (prv2 ⇒ prv1 ⇒ prv, obv2 ⇒ obv1 ⇒ obv, and aut2 ⇒ aut1 ⇒ aut) nothing implies anything else.

Achieving adaptive security. Basic garbling-scheme constructions [4, 10, 11, 18] either do not achieve adaptive security or present difficulties in proving adaptive security that we do not know how to overcome. One could give new constructions and directly prove them xxx1 or xxx2 secure, for xxx ∈ {prv, obv, aut}. An alternative is to provide generic ways to transform statically secure garbling schemes to adaptively secure ones. Combined with results in BHR [4], this would yield adaptively-secure garbling schemes.

The aim of the GKR construction was exactly to add adaptive security to statically-secure garbled circuit constructions. We reformulate it as a transform, OMSS (Output Masking and Secret Sharing), aiming to turn a prv-secure garbling scheme to a prv2-secure one. We show, by counterexample, that OMSS does not achieve this goal.

To give transforms that work we make two steps, first passing from static security to coarse-grained adaptive security, and thence to fine-grained adaptive security. We design these transformations first for privacy (prv-to-prv1, prv1-to-prv2) and then for simultaneously achieving all three goals (all-to-all1 and all1-to-all2). Our prv-to-prv1 transform uses a one-time-padding technique from [14], while our prv1-to-prv2 transform uses the secret-sharing component of OMSS.

Applications. We treat the two applications that motivated this work, one-time programs and secure outsourcing. We show that adaptive garbling schemes yield these applications easily and directly. Specifically, we show that a prv2 projective garbling scheme can be turned into a secure one-time program by simply putting the garbled inputs into the one-time memory. We also show how to easily turn an obv1+aut1-secure garbling scheme into a secure one-time outsourcing scheme. (GGP [9] show how to lift one-time outsourcing schemes to many-time ones using FHE.) The simplicity of these transformations underscores our tenet that abstracting garbling schemes and treating adaptive security for them enables modular and rigorous applications of the garbled-circuit technique. Basing the applications on garbling schemes also allows instantiations to inherit efficiency features of future schemes.

[3] The side-information function Φ captures that about f one allows to be revealed in its garbled counterpart F.

Transform	Model	Cost	See
prv-to-prv1	standard model	$\|F\| + \|d\| + \|X\|$	Theorem 2
prv1-to-prv2	standard model	$(n+1)\,\|X\|$	Theorem 3
all-to-all1	standard model	$\|F\| + \|d\| + \|X\| + k$	Theorem 5
all1-to-all2	standard model	$(n+1)\,\|X\|$	Theorem 6
rom-prv-to-prv1	random-oracle model	$\|X\| + k$	Full paper [3]
rom-prv1-to-prv2	random-oracle model	$\|X\| + nk$	Full paper [3]
rom-all-to-all1	random-oracle model	$\|X\| + 2k$	Full paper [3]
rom-all1-to-all2	random-oracle model	$\|X\| + nk$	Full paper [3]

Fig. 1. Achieving adaptive security. The name of each transform specifies its relevant property. The word all means that prv, obv, and aut are all upgraded. Column "Cost" specifies the length of the garbled input in the constructed scheme in terms of the lengths of the input scheme's garbled function F, decoding function d, garbled input X, number input bits n, and security parameter k.

Applying our prv-to-prv1 and then prv1-to-prv2 transforms to the prv-secure garbling scheme of BHR [4] yields a prv2-secure scheme based on any one-way function. Combining this with the above yields one-time programs based on one-way functions, recovering the claim of GKR [13]. Similarly, applying our all-to-all1 transform to the obv+aut-secure scheme of BHR yields an obv1+aut1-secure garbling scheme based on a one-way function, and combining this with the above yields a secure one-time outsourcing scheme based on one-way functions.

Efficiency. Let us say a garbling scheme has *short* garbled inputs if their length depends only on the security parameter k, the length n of f's input, and the length m of f's output. It does not depend on the length of f. The statically-secure schemes of BHR, as with all classical garbled-circuit constructions, have short garbled inputs. But our prv-to-prv1 and all-to-all1 transforms result in long garbled inputs. In the ROM (random-oracle model) we are able to provide schemes producing short garbled inputs, as illustrated in Fig. 1. Constructing an adaptively secure garbling scheme with short garbled inputs under standard assumptions remains open.

Short garbled inputs are particularly important for the application to secure outsourcing, for in their absence the outsourcing scheme may fail to be non-trivial. (Non-trivial means that the client effort is less than the effort needed to directly compute the function [9].) In particular, the one-time outsourcing scheme we noted above, derived by applying all-to-all1 to BHR, fails to be non-trivial. ROM schemes do not fill the gap because of the use of FHE in upgrading one-time schemes to many-time ones [9]. Thus, a secure and non-trivial instantiation of the GGP method is still lacking. (However, as we have noted before, non-trivial secure outsourcing may be achieved by entirely different means [7].)

Further related work. Applebaum, Ishai, and Kushilevitz [1] investigate ideas similar to obliviousness and authenticity. Their approach to obtaining these

ends from privacy can be lifted and formalized in our settings; one could specify transforms prv1-to-all1 and prv2-to-all2, effectively handling the constructive story "horizontally" instead of "vertically." The line of work on *randomized encodings* that the same authors have been at the center of provides an alternative to garbling schemes [15] but lacks the granularity to speak of adaptive security.

Concurrent work by Kamara and Wei (KW) investigates the garbling what they call *structured circuits* [16] and, in the process, give definitions somewhat resembling prv1, obv1, and aut1, although circuit-based, not function-hiding, and not allowing the adversary to specify the initial function. KW likewise draw motivation from GKR and GGP, indicating that, in these two setting, the adversary can choose the inputs to the computation as a function of the garbled circuit, motivating adaptive notions of privacy and unforgeability.

2 Framework

We now review the syntactic framework of garbling schemes from our earlier work [4]. See the full version for [3] basic notation, including conventions for randomized algorithms, code-based games, and circuits.

Garbling schemes. A *garbling scheme* [4] is a five-tuple of algorithms $\mathcal{G} = (\mathsf{Gb}, \mathsf{En}, \mathsf{De}, \mathsf{Ev}, \mathsf{ev})$.The first of these is probabilistic; the rest are deterministic. A string f, the *original function*, describes the function $\mathsf{ev}(f, \cdot)\colon\{0,1\}^n \to \{0,1\}^m$ that we want to garble. The values $n = f.n$ and $m = f.m$ are efficiently computable from f. On input f and a security parameter $k \in \mathbb{N}$, algorithm Gb returns a triple of strings $(F, e, d) \leftarrow \mathsf{Gb}(1^k, f)$. String e describes an *encoding function*, $\mathsf{En}(e, \cdot)$, that maps an *initial input* $x \in \{0,1\}^n$ to a *garbled input* $X = \mathsf{En}(e, x)$. String F describes a *garbled function*, $\mathsf{Ev}(F, \cdot)$, that maps a garbled input X to a *garbled output* $Y = \mathsf{Ev}(F, X)$. String d describes a *decoding function*, $\mathsf{De}(d, \cdot)$, that maps a garbled output Y to a *final output* $y = \mathsf{De}(d, Y)$. The correctness requirement is that if $f \in \{0,1\}^*$, $k \in \mathbb{N}$, $x \in \{0,1\}^{f.n}$, and $(F, e, d) \in [\mathsf{Gb}(1^k, f)]$, then $\mathsf{De}(d, \mathsf{Ev}(F, \mathsf{En}(e, x))) = \mathsf{ev}(f, x)$. We also require that e and d depend only on k, $f.n$, $f.m$, $|f|$ and the random coins r of Gb. This non-degeneracy requirement excludes trivial solutions.

A common design in existing garbling schemes is for e to encode a list of *tokens*, one pair for each bit in $x \in \{0,1\}^n$. Encoding function $\mathsf{En}(e, \cdot)$ then uses the bits of $x = x_1 \cdots x_n$ to select from $e = (X_1^0, X_1^1, \ldots, X_n^0, X_n^1)$ the subvector $X = (X_1^{x_1}, \ldots, X_n^{x_n})$. Formally, we say that garbling scheme $\mathcal{G} = (\mathsf{Gb}, \mathsf{En}, \mathsf{De}, \mathsf{Ev}, \mathsf{ev})$ is *projective* if for all f, $x, x' \in \{0,1\}^{f.n}$, $k \in \mathbb{N}$, and $i \in [1..n]$, when $(F, e, d) \in [\mathsf{Gb}(1^k, f)]$, $X = \mathsf{En}(e, x)$ and $X' = \mathsf{En}(e, x')$, then $X = (X_1, \ldots, X_n)$ and $X' = (X'_1, \ldots, X'_n)$ are n vectors, $|X_i| = |X'_i|$, and $X_i = X'_i$ if x and x' have the same ith bit. Let $\mathsf{GS}(\mathsf{proj})$ denote the set of all projective garbling schemes.

Boolean circuits arise often in this work. We say that $\mathcal{G} = (\mathsf{Gb}, \mathsf{En}, \mathsf{De}, \mathsf{Ev}, \mathsf{ev})$ is a circuit-garbling scheme if ev is the canonical circuit evaluation function.

Side-information functions. A garbled circuit might reveal the size of the circuit that is being garbled, its topology, the original circuit itself, or

something else. The information that we allow to be revealed is captured by a *side-information function*, Φ, which deterministically maps f to a string $\phi = \Phi(f)$. We parameterize our advantage notions by Φ. We require that $f.n$, $f.m$ and $|f|$ be easily determined from $\phi = \Phi(f)$. Side-information function Φ_{size} maps a circuit $f = (n, m, q, A, B, G)$ to (n, m, q), while Φ_{topo} maps f to $f^- = \text{Topo}(f) = (n, m, q, A, B)$ and Φ_{circ} is the identity, $\Phi_{\text{circ}}(f) = f$.

Sizes. We say that garbling scheme $\mathcal{G} = (\mathsf{Gb}, \mathsf{En}, \mathsf{De}, \mathsf{Ev}, \mathsf{ev})$ has *short garbled inputs* if there is a polynomial s such that $|\mathsf{En}(e, x)| \leq s(k, f.n, f.m)$ for all $k \in \mathbb{N}$, $f \in \{0,1\}^*$, $(F, e, d) \in [\mathsf{Gb}(1^k, f)]$, and $x \in \{0,1\}^{f.n}$. Let T be a transform that maps a garbling scheme $\mathcal{G}$ to a garbling scheme $\mathsf{T}[\mathcal{G}]$. We say that T *preserves* short garbled inputs if $\mathsf{T}[\mathcal{G}]$ has short garbled inputs when $\mathcal{G}$ does.

Typical Yao-style constructions, including Garble1 and Garble2 [4], have short garbled inputs. But they are only statically-secure. Keeping garbled inputs short seems challenging for adaptive security in the standard model.

3 Privacy and One-Time Programs

In this section we define coarse and fine-grained adaptive privacy for garbling schemes. We show that some natural approaches to achieve these aims fail. We provide alternatives that work. In [3], we provide more efficient ones in the ROM. We apply this to get secure one-time programs.

Definitions for adaptive privacy. On the top of Fig. 2 we review the defining game for the privacy notion from BHR [4]. The adversary is *static*, in the sense it must commit to its initial function f and its input x at the same time. Thus the latter is independent of the garbled function F (and the decoding function d) derived from f. It is natural to consider stronger privacy notions, ones where the adversary obtains F and *then* selects x. Two formulations for this are specified in Fig. 2. We call these *adaptive* security. The notion in the middle panel, denoted by prv1, this paper, is *coarse-grained* adaptive security. The notion in the bottom panel, denoted by prv2, is *fine-grained* adaptive security. This notion is only applicable for projective garbling schemes.

In detail, let $\mathcal{G} = (\mathsf{Gb}, \mathsf{En}, \mathsf{De}, \mathsf{Ev}, \mathsf{ev})$ be a garbling scheme and let Φ be a side-information function. We define three simulation-based notions of privacy via the games $\text{Prv}_{\mathcal{G},\Phi,\mathcal{S}}$, $\text{Prv1}_{\mathcal{G},\Phi,\mathcal{S}}$, and $\text{Prv2}_{\mathcal{G},\Phi,\mathcal{S}}$ of Fig. 2. Here $\mathcal{S}$, the *simulator*, is an always-terminating algorithm that maintains state across invocations. An adversary $\mathcal{A}$ interacting with any of these games must make exactly one Garble query. For game Prv1 it is followed by a single Input query. For game Prv2 it is followed by multiple Input queries. There, the garbling scheme must be projective. The advantage the adversary gets is defined by

$$\begin{aligned}
\mathbf{Adv}^{\text{prv},\Phi,\mathcal{S}}_{\mathcal{G}}(\mathcal{A}, k) &= 2\Pr[\text{Prv}^{\mathcal{A}}_{\mathcal{G},\Phi,\mathcal{S}}(k)] - 1 \\
\mathbf{Adv}^{\text{prv1},\Phi,\mathcal{S}}_{\mathcal{G}}(\mathcal{A}, k) &= 2\Pr[\text{Prv1}^{\mathcal{A}}_{\mathcal{G},\Phi,\mathcal{S}}(k)] - 1 \\
\mathbf{Adv}^{\text{prv2},\Phi,\mathcal{S}}_{\mathcal{G}}(\mathcal{A}, k) &= 2\Pr[\text{Prv2}^{\mathcal{A}}_{\mathcal{G},\Phi,\mathcal{S}}(k)] - 1 \,.
\end{aligned}$$

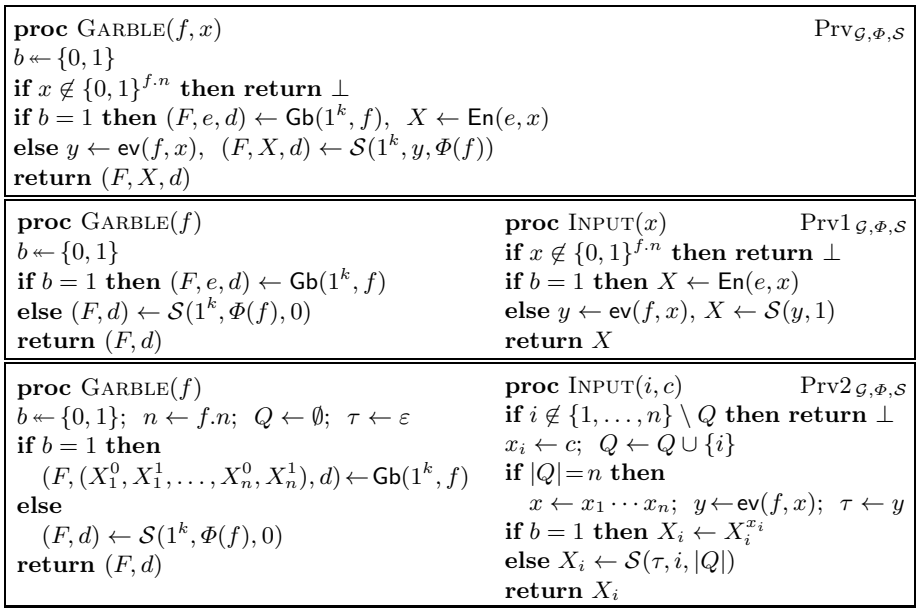

Fig. 2. Three kinds of privacy: prv, prv1, prv2. Games to define the static, coarse-grained, and fine-grained privacy of $\mathcal{G} = (\mathsf{Gb}, \mathsf{En}, \mathsf{De}, \mathsf{Ev}, \mathsf{ev})$. FINALIZE$(b')$ returns the predicate $(b = b')$. Notation $s \twoheadleftarrow S$ denotes uniform sampling from a finite set.

For xxx $\in \{\text{prv}, \text{prv1}, \text{prv2}\}$ we say that $\mathcal{G}$ is xxx-secure with respect to (or over) Φ if for every PT adversary $\mathcal{A}$ there exists a PT simulator $\mathcal{S}$ such that $\mathbf{Adv}^{\text{xxx},\Phi,\mathcal{S}}_{\mathcal{G}}(\mathcal{A}, \cdot)$ is negligible. We let $\mathsf{GS}(\text{xxx}, \Phi)$ be the set of all garbling schemes that are xxx-secure over Φ.

Let us now explain the three games, beginning with static privacy. Here we let the adversary select f and x and we do one of two things: garble f to make (F, e, d) and encode x to make X, giving the adversary (F, X, d); or, alternatively, we ask the simulator produce a "fake" (F, X, d) based only on the security parameter k, the partial information $\Phi(f)$ about f, and the output $y = \mathsf{ev}(f, x)$. The adversary will have to guess if the garbling was real or fake.

For coarse-grained adaptive privacy, we begin by letting the adversary pick f. Either we garble it to $(F, e, d) \leftarrow \mathsf{Gb}(1^k, f)$ and give the adversary (F, d); or else we ask the simulator to devise a fake (F, d) based solely on k and $\phi = \Phi(f)$. Only after the adversary has received (F, d) do we ask it to provide an input x. Corresponding to the two choices we either encode x to $X = \mathsf{En}(e, x)$ or ask the simulator to produce a fake X, assisting it only by providing $\mathsf{ev}(f, x)$.

Coarse-grained adaptive privacy is arguably not all *that* adaptive, as the adversary specifies its input x all in one shot. This is unavoidable as long as the encoding function e operates on x atomically. But if the encoding function e is projective, then we can dole out the garbled input component-by-component. Only after the adversary specifies all n bits, one by one, is the input

fully determined. At that point the simulator is handed y, which might be needed for constructing the final token $X_i^{x_i}$.

THE OMSS TRANSFORM. In the process of constructing one-time programs from garbled circuits, GKR [13] recognize the need for adaptive privacy of the garbled circuits. Their construction incorporates a technique to provide it. This technique is easily abstracted to provide, in our terminology, a transform that aims to convert a projective, prv garbling scheme into a projective, prv2 garbling scheme. Instead of garbling f we pick $r \twoheadleftarrow \{0,1\}^m$ and garble the circuit g defined by $g(x) = f(x) \oplus r$ for every $x \in \{0,1\}^n$ where $n = f.n$ and $m = f.m$. Then we secret share r as $r = r_1 \oplus \cdots \oplus r_n$ and include r_i in the i-th token, so that evaluation reconstructs r and it can be xored back at decoding time to recover $\mathsf{ev}(f,x)$ as $\mathsf{ev}(g,x) \oplus r$. Intuitively, this should work because the simulator can garble a dummy constant function with random output s and does not have to commit to r until it gets the target output value y of f and needs to provide the last token, at which point it can pick $r = s \oplus y$ so that y as desired [13]. Just the same, we show by counterexample that the OMSS does not in work, in general, to convert a prv-secure scheme to a prv2-secure one: we present a prv secure $\mathcal{G}$ such that $\mathsf{OMSS}[\mathcal{G}]$ is not prv2 secure. While this does not show that OMSS fails in the context in which GMR use it, our counterexample extends to that setting as well; see the full paper [3].

Now proceeding formally, we associate to circuit-garbling scheme $\mathcal{G} = (\mathsf{Gb}, \mathsf{En}, \mathsf{De}, \mathsf{Ev}, \mathsf{ev}) \in \mathsf{GS}(\mathsf{proj})$ the circuit-garbling scheme $\mathsf{OMSS}[\mathcal{G}] = (\mathsf{Gb}_2, \mathsf{En}_2, \mathsf{De}_2, \mathsf{Ev}_2, \mathsf{ev}) \in \mathsf{GS}(\mathsf{proj})$ defined at the top of Fig. 3. For simplicity we are assuming that the decoding rule d in $\mathcal{G}$ is always vacuous, meaning $d = \varepsilon$. (We do not need non-trivial d to achieve privacy [4], and this lets us stay closer to GKR [13], whose garbled circuits have no analogue of our decoding rule.) In the code, $g(\cdot) \leftarrow f(\cdot) \oplus r$ means that we construct from f, r a circuit g such that $\mathsf{ev}(g,x) = \mathsf{ev}(f,x) \oplus r$ for all $x \in \{0,1\}^{f.n}$. (Note we can do this in such a way that $\Phi_{\mathrm{topo}}(g) = \Phi_{\mathrm{topo}}(f)$.)

The claim under consideration is that if $\mathcal{G}$ is prv-secure relative to $\Phi = \Phi_{\mathrm{topo}}$ then $\mathcal{G}_2$ is prv2-secure relative to $\Phi = \Phi_{\mathrm{topo}}$. To prove this, we would need to let $\mathcal{A}_2$ be an arbitrary PT adversary and build a PT simulator $\mathcal{S}_2$ such that $\mathbf{Adv}^{\mathrm{prv2},\Phi,\mathcal{S}_2}_{\mathcal{G}_2}(\mathcal{A}_2,\cdot)$ is negligible. GKR suggest a plausible strategy for the simulator that, in particular, explains the intuition for the transform. We present here our understanding of this strategy adapted to our setting. In its first phase the simulator $\mathcal{S}_2$ has input $1^k, \phi, 0$ where $\phi = \Phi(f)$, with f being the query made by the adversary to GARBLE. Simulator $\mathcal{S}_2$ picks $s \twoheadleftarrow \{0,1\}^n$ and lets f_s be the circuit that has output s on all inputs and $\Phi_{\mathrm{topo}}(f_s) = \phi$. It also picks random m-bit strings $s_1, \ldots, s_n$ and a random input $w \twoheadleftarrow \{0,1\}^n$. It lets $(G, (X_1^0, X_1^1, \ldots, X_n^0, X_n^1), \varepsilon) \twoheadleftarrow \mathsf{Gb}(1^k, f_s)$ and returns G to the adversary, saving $\sigma = (s, s_1, \ldots, s_n)$ as state information. In the second phase, when given input τ, i, j, for $j \le n-1$, the simulator lets $T_i \leftarrow (X_i^{w_i}, s_i)$ and returns T_i to the adversary as the token for bit i of the input. In the case that $j = n$, the simulator obtains (from τ as per our game) the output $y = \mathsf{ev}(f,x)$ of the function on input x, the latter defined by the adversary's queries to INPUT. It now resets

$\mathbf{proc}\ \mathsf{Gb}_2(1^k, f)$
$n \leftarrow f.n,\ r_1, \ldots, r_n \twoheadleftarrow \{0,1\}^{f.m}$
$r \leftarrow r_1 \oplus \cdots \oplus r_n,\ g(\cdot) \leftarrow f(\cdot) \oplus r$
$(G, (X_1^0, X_1^1, \ldots, X_n^0, X_n^1), \varepsilon) \twoheadleftarrow \mathsf{Gb}(1^k, g)$
$\mathbf{for}\ i \in \{1, \ldots, n\}\ \mathbf{do}$
  $T_i^0 \leftarrow (X_i^0, r_i),\ T_i^1 \leftarrow (X_i^1, r_i)$
$\mathbf{return}\ (G, (T_1^0, T_1^1, \ldots, T_n^0, T_n^1), \varepsilon)$

$\mathbf{proc}\ \mathsf{Ev}_2(G, (T_1, \ldots, T_n))$
$\mathbf{for}\ i \in \{1, \ldots, n\}\ \mathbf{do}\ (X_i, r_i) \leftarrow T_i$
$Y \leftarrow \mathsf{Ev}(G, (X_1, \ldots, X_n))$
$r \leftarrow r_1 \oplus \cdots \oplus r_n$
$\mathbf{return}\ (Y, r)$

$\mathbf{proc}\ \mathsf{En}_2((T_1^0, T_1^1, \ldots, T_n^0, T_n^1), x)$
$x_1 \cdots x_n \leftarrow x$
$\mathbf{return}\ (T_1^{x_1}, \ldots, T_n^{x_n})$

$\mathbf{proc}\ \mathsf{De}_2(\varepsilon, (Y, r))$
$\mathbf{return}\ \mathsf{De}(\varepsilon, Y) \oplus r$

$\mathbf{proc}\ \mathsf{Gb}(1^k, g)$
$(n, m) \leftarrow (g.n, g.m)$
$(G', (Z_1^0, Z_1^1, \ldots, Z_n^0, Z_n^1), \varepsilon) \twoheadleftarrow \mathsf{Gb}'(1^k, g)$
$\mathbf{for}\ i \in \{1, \ldots, n\}\ \mathbf{do}\ V_i^0, V_i^1 \twoheadleftarrow \{0,1\}^m$
$v_1 \cdots v_n \leftarrow v \twoheadleftarrow \{0,1\}^n, \quad V \twoheadleftarrow \{0,1\}^m$
$\mathbf{if}\ n \geq k\ \mathbf{then}$
  $V \leftarrow \mathsf{ev}(g, \overline{v}) \oplus V_1^{v_1} \oplus \cdots \oplus V_n^{v_n}$
$\mathbf{for}\ i \in \{1, \ldots, n\}\ \mathbf{do}$
  $X_i^0 \leftarrow (Z_i^0, V_i^0), \quad X_i^1 \leftarrow (Z_i^1, V_i^1)$
$G \leftarrow (G', v, V)$
$\mathbf{return}\ (G, (X_1^0, X_1^1, \ldots, X_n^0, X_n^1), \varepsilon)$

$\mathbf{proc}\ \mathsf{Ev}(G, (X_1, \ldots, X_n))$
$\mathbf{for}\ i \in \{1, \ldots, n\}\ \mathbf{do}\ (Z_i, V_i) \leftarrow X_i$
$(G', v, V) \leftarrow G$
$\mathbf{return}\ \mathsf{Ev}'(G', (Z_1, \ldots, Z_n))$

$\mathbf{proc}\ \mathsf{En}((X_1^0, X_1^1, \ldots, X_n^0, X_n^1), x)$
$x_1 \cdots x_n \leftarrow x$
$\mathbf{return}\ (X_1^{x_1}, \ldots, X_n^{x_n})$

Fig. 3. OMSS definition (top). Scheme $\mathsf{OMSS}[\mathcal{G}] = (\mathsf{Gb}_2, \mathsf{En}_2, \mathsf{De}_2, \mathsf{Ev}_2, \mathsf{ev})$ where $\mathcal{G} = (\mathsf{Gb}, \mathsf{En}, \mathsf{De}, \mathsf{Ev}, \mathsf{ev})$. **OMSS counterexample (bottom).** The garbling scheme $\mathcal{G} = (\mathsf{Gb}, \mathsf{En}, \mathsf{De}, \mathsf{Ev}, \mathsf{ev})$ obtained from $\mathcal{G}' = (\mathsf{Gb}', \mathsf{En}', \mathsf{De}, \mathsf{Ev}', \mathsf{ev})$ is prv secure when $\mathcal{G}'$ is, but $\mathsf{OMSS}[\mathcal{G}]$ is not prv2 secure.

$s_i = y \oplus s \oplus s_i \oplus s_1 \oplus \cdots \oplus s_n$ and returns (X_i, s_i), so that evaluation of the garbled function indeed results in output y.

This simulation strategy is intuitive, but trying to prove it correct runs into problems. We have to show that $\mathbf{Adv}^{\mathrm{prv2}, \Phi, \mathcal{S}_2}_{\mathcal{G}_2}(\mathcal{A}_2, \cdot)$ is negligible. We must utilize the assumption of prv security to do this, which means we must perform a reduction. The only plausible path towards this is to construct from $\mathcal{A}_2$ an adversary $\mathcal{A}$ against the prv-security of $\mathcal{G}$ and then exploit the existence of a simulator $\mathcal{S}$ such that $\mathbf{Adv}^{\mathrm{prv}, \Phi, \mathcal{S}}_{\mathcal{G}}(\mathcal{A}, \cdot)$ is negligible. However, it is not clear how to construct $\mathcal{A}$, let alone how its simulator comes into play. (As we will see when proving our transforms, the proof template that works is different, not trying first to build $\mathcal{S}_2$, but instead building $\mathcal{A}$ from $\mathcal{A}_2$ and then $\mathcal{S}_2$ from $\mathcal{S}$.)

The problem turns out to be more than technical, for we will see that the transform itself does not work in general. By this we mean that we can exhibit a (projective) circuit-garbling scheme $\mathcal{G} = (\mathsf{Gb}, \mathsf{En}, \mathsf{De}, \mathsf{Ev}, \mathsf{ev})$ that is prv-secure relative to $\Phi = \Phi_{\mathrm{topo}}$ but the transformed scheme $\mathcal{G}_2 = \mathsf{OMSS}[\mathcal{G}]$ is subject to

$\mathbf{proc}\ \mathsf{Gb}_1(1^k, f)$
$(F, e, d) \leftarrow \mathsf{Gb}(1^k, f)$
$F' \twoheadleftarrow \{0,1\}^{|F|},\ \ d' \twoheadleftarrow \{0,1\}^{|d|}$
$F_1 \leftarrow F \oplus F',\ \ d_1 \leftarrow d \oplus d'$
$e_1 \leftarrow (e, d', F')$
$\mathbf{return}\ (F_1, e_1, d_1)$

$\mathbf{proc}\ \mathsf{En}_1(e_1, x)$
$(e, d', F') \leftarrow e_1, \quad X \leftarrow \mathsf{En}(e, x)$
$\mathbf{return}\ (X, d', F')$

$\mathbf{proc}\ \mathsf{Ev}_1(F_1, X_1)$
$(X, d', F') \leftarrow X_1, \quad F \leftarrow F_1 \oplus F'$
$Y \leftarrow \mathsf{Ev}(F, X)$
$\mathbf{return}\ (Y, d')$

$\mathbf{proc}\ \mathsf{De}_1(d_1, Y_1)$
$(Y, d') \leftarrow Y_1,\ \ d \leftarrow d_1 \oplus d'$
$\mathbf{return}\ \mathsf{De}(d, Y)$

$\mathbf{proc}\ \mathsf{Gb}_2(1^k, f)$
$(F, e, d) \leftarrow \mathsf{Gb}_1(1^k, f)$
$(X_1^0, X_1^1, \ldots, X_n^0, X_n^1) \leftarrow e$
$N \leftarrow |\mathsf{En}_1(e, 0^n)|$
$\mathbf{for}\ i \in \{1, \ldots, n\}\ \mathbf{do}$
  $Z_i \twoheadleftarrow \{0,1\}^{|X_i^0|}, \quad S_i \twoheadleftarrow \{0,1\}^N$
$Z \leftarrow (Z_1, \ldots, Z_n)$
$S_n \leftarrow Z \oplus S_1 \oplus \cdots \oplus S_{n-1}$
$\mathbf{for}\ i \in \{1, \ldots, n\}\ \mathbf{do}$
  $T_i^0 \leftarrow (X_i^0 \oplus Z_i, S_i),\ \ T_i^1 \leftarrow (X_i^1 \oplus Z_i, S_i)$
$\mathbf{return}\ (F, (T_1^0, T_1^1, \ldots, T_n^0, T_n^1), d)$

$\mathbf{proc}\ \mathsf{Ev}_2(F, X_2)$
$\big((U_1, S_1), \ldots, (U_n, S_n)\big) \leftarrow X_2$
$Z \leftarrow S_1 \oplus \cdots \oplus S_n$
$(Z_1, \ldots, Z_n) \leftarrow Z$
$X \leftarrow (U_1 \oplus Z_1, \ldots, U_n \oplus Z_n)$
$\mathbf{return}\ \mathsf{Ev}_1(F, X)$

$\mathbf{proc}\ \mathsf{En}_2(e_2, x)$
$(T_1^0, X_1^1, \ldots, T_n^0, X_n^1) \leftarrow e_2$
$x_1 \cdots x_n \leftarrow x$
$\mathbf{return}\ (T_1^{x_1}, \ldots, T_n^{x_n})$

Fig. 4. Transform prv-to-prv1 (top): Scheme $\mathcal{G}_1 = (\mathsf{Gb}_1, \mathsf{En}_1, \mathsf{De}_1, \mathsf{Ev}_1, \mathsf{ev}) \in \mathsf{GS}(\mathrm{prv1}, \Phi)$ obtained by applying the prv-to-prv1 transform to $\mathcal{G} = (\mathsf{Gb}, \mathsf{En}, \mathsf{De}, \mathsf{Ev}, \mathsf{ev}) \in \mathsf{GS}(\mathrm{prv}, \Phi)$. **Transform prv1-to-prv2 (bottom):** Projective garbling scheme $\mathcal{G}_2 = (\mathsf{Gb}_2, \mathsf{En}_2, \mathsf{De}, \mathsf{Ev}_2, \mathsf{ev}) \in \mathsf{GS}(\mathrm{prv2}, \Phi)$ obtained by applying the prv1-to-prv2 transform to projective garbling scheme $\mathcal{G}_1 = (\mathsf{Gb}_1, \mathsf{En}_1, \mathsf{De}_1, \mathsf{Ev}_1, \mathsf{ev}) \in \mathsf{GS}(\mathrm{prv1}, \Phi)$

an attack showing that it is not prv2 secure. This means, in particular, that the above simulation strategy does not in general work.

To carry this out, we start with an arbitrary projective circuit-garbling scheme $\mathcal{G}' = (\mathsf{Gb}', \mathsf{En}', \mathsf{De}, \mathsf{Ev}', \mathsf{ev})$ assumed to be prv-secure relative to $\Phi = \Phi_{\mathrm{topo}}$. We then transform it into the projective circuit-garbling scheme $\mathcal{G} = (\mathsf{Gb}, \mathsf{En}, \mathsf{De}, \mathsf{Ev}, \mathsf{ev})$ shown at the bottom of Fig. 3. (We assume the decoding rule of $\mathcal{G}'$ is vacuous, a feature inherited by $\mathcal{G}$. We are letting $\overline{v}$ denote the bitwise complement of a string v.) The following proposition, whose proof is in the full paper [3], says that $\mathcal{G}$ continues to be prv-secure but an attack shows that $\mathsf{OMSS}[\mathcal{G}]$ is not prv2-secure. (The proof shows it is in fact not even prv1 secure.)

Proposition 1. Let ev be the canonical circuit-evaluation function. Assume $\mathcal{G}' = (\mathsf{Gb}', \mathsf{En}', \mathsf{De}, \mathsf{Ev}', \mathsf{ev}) \in \mathsf{GS}(\mathrm{prv}, \Phi_{\mathrm{topo}}) \cap \mathsf{GS}(\mathsf{proj})$ and let $\mathcal{G} = (\mathsf{Gb}, \mathsf{En}, \mathsf{De}, \mathsf{Ev}, \mathsf{ev}) \in \mathsf{GS}(\mathsf{proj})$ be the garbling scheme shown at the bottom of Fig. 3. Then **(1)** $\mathcal{G} \in \mathsf{GS}(\mathrm{prv}, \Phi_{\mathrm{topo}}) \cap \mathsf{GS}(\mathsf{proj})$, but **(2)** $\mathsf{OMSS}[\mathcal{G}] \notin \mathsf{GS}(\mathrm{prv2}, \Phi_{\mathrm{topo}})$.

Achieving prv1 security. We now describe a transform prv-to-prv1 that successfully turns a prv secure circuit garbling scheme into a prv1 secure one.

Combined with established results [4], this yields prv1-secure schemes based on standard assumptions. The idea is to use one-time pads to mask F and d, and then append the pads to X. This will ensure that the adversary learns nothing about F and d until it fully specifies function f and x. Given a (not necessarily projective) garbling scheme $\mathcal{G} = (\mathsf{Gb}, \mathsf{En}, \mathsf{De}, \mathsf{Ev}, \mathsf{ev})$, the prv-to-prv1 transform returns the garbling scheme $\mathsf{prv\text{-}to\text{-}prv1}[\mathcal{G}] = (\mathsf{Gb}_1, \mathsf{En}_1, \mathsf{De}_1, \mathsf{Ev}_1, \mathsf{ev})$ at the top of Fig. 4. We claim:

Theorem 2. For any Φ, if $\mathcal{G} \in \mathsf{GS}(\mathrm{prv}, \Phi)$ then $\mathsf{prv\text{-}to\text{-}prv1}[\mathcal{G}] \in \mathsf{GS}(\mathrm{prv1}, \Phi)$.

The intuition behind the prv-to-prv1 transform (outlined above) is simple, but the proof template is instructive in indicating how to move from the intuition to a formal proof. Given any PT adversary $\mathcal{A}_1$ against the prv1-security of $\mathcal{G}_1$ we build a PT adversary $\mathcal{A}$ against the prv-security of $\mathcal{G}$. Now the assumption of prv-security yields a PT simulator $\mathcal{S}$ for $\mathcal{A}$ such that $\mathbf{Adv}^{\mathrm{prv},\Phi,\mathcal{S}}_{\mathcal{G}}(\mathcal{A},\cdot)$ is negligible. Now we build from $\mathcal{S}$ a PT simulator $\mathcal{S}_1$ such that for all $k \in \mathbb{N}$ we have $\mathbf{Adv}^{\mathrm{prv1},\Phi,\mathcal{S}_1}_{\mathcal{G}_1}(\mathcal{A}_1, k) \leq \mathbf{Adv}^{\mathrm{prv},\Phi,\mathcal{S}}_{\mathcal{G}}(\mathcal{A}, k)$. This yields the theorem. In the full paper [3] we provide a full proof that shows how to build $\mathcal{A}$ and $\mathcal{S}_1$.

Achieving prv2 security. Next we show how to transform a prv1 scheme into a prv2 one. Formally, given a projective garbling scheme $\mathcal{G} = (\mathsf{Gb}, \mathsf{En}, \mathsf{De}, \mathsf{Ev}, \mathsf{ev}) \in \mathsf{GS}(\mathrm{prv1}, \Phi)$, the prv1-to-prv2 transform returns the projective garbling scheme $\mathsf{prv1\text{-}to\text{-}prv2}[\mathcal{G}] = (\mathsf{Gb}_2, \mathsf{En}_2, \mathsf{De}, \mathsf{Ev}_2, \mathsf{ev})$ shown at the bottom of Fig. 4. The idea is to mask the garbled input and then use the second part of GKR's idea as represented by OMSS, namely secret-share the mask, putting a piece in each token, so that unless one has all tokens, one learns nothing about the garbled input. The formal proof of the following is in the full paper [3].

Theorem 3. For any Φ, if $\mathcal{G}_1 \in \mathsf{GS}(\mathrm{prv1}, \Phi) \cap \mathsf{GS}(\mathsf{proj})$ then $\mathsf{prv1\text{-}to\text{-}prv2}[\mathcal{G}_1] \in \mathsf{GS}(\mathrm{prv2}, \Phi) \cap \mathsf{GS}(\mathsf{proj})$.

One-time compilers. Starting from garbling schemes with prv2 security, we give simple designs, and proofs, for one-time programs. We begin with the definitions. Following GKR [13], the intent is that possession of a one-time program P for a function f should enable one to evaluate f at any single value x; but, beyond that, the one-time program should be useless. Unachievable in any standard model of computation (where possession of P would enable its repeated evaluation at multiple point), GKR suggest achieving one-time programs in a model of computation that provides *one-time memory*—tamper-resistant hardware whose read-once i-th location returns, on query $(i, b) \in \mathbb{N} \times \{0,1\}$, the string T^b_i, immediately thereafter expunging T^{1-b}_i. A *one-time compiler* probabilistically transforms the description of a function f into a one-time program P and its associated one-time memory T.

For a formal treatment, we begin by specifying two stateful oracles; see Fig. 5. The first, OTP_f, formalizes the desired behavior of a one-time program for f. Here f will now be regarded as a string, not a function, but this string represents a circuit computing a function $\mathsf{ev}(f) : \{0,1\}^{f.n} \to \{0,1\}^{f.m}$; we write ev for the canonical circuit-evaluation function [4]. The agent calling out to OTP_f

proc $\mathrm{OTP}_f(x)$	**proc** $\mathrm{OTM}_T(i,b)$
if $x \notin \{0,1\}^{f.n}$ **then ret** $\perp$	$(T_1^0, T_1^1, \ldots, T_\ell^0, T_\ell^1) \leftarrow T$
if *called* **then ret** $\perp$	**if** $i \notin [1..\ell]$ **or** $used_i$ **or** $b \notin \{0,1\}$ **then ret** $\perp$
$called \leftarrow \mathsf{true}$	$used_i \leftarrow \mathsf{true}$
ret $\mathsf{ev}(f,x)$	**ret** T_i^b

Fig. 5. Oracles model one-time programs and one-time memory. Oracle OTP depends on a string f representing a boolean circuit. Oracle OTM depends on a list of strings T.

provides x and, on the first query, it gets $\mathsf{ev}(f,x)$. Subsequent queries return nothing. On the right-hand side of Fig. 5 we similarly define an oracle OTM_T, this to model possession of a one-time-memory system. Given a list of ℓ pairs of strings (establish some convention so that every string T is regarded as denoting a list of ℓ pairs of strings, for some $\ell \in \mathbb{N}$) the oracle returns at most one string from each pair, otherwise satisfying each request.

Elaborating on GKR, we now define a *one-time compiler* as a pair of probabilistic algorithms $\Pi = (\mathsf{Co}, \mathsf{Ex})$ (for *compile* and *execute*). Algorithm Co, on input 1^k and a string f, produces a pair $(P,T) \leftarrow \mathsf{Co}(1^k, f)$ where P (the one-time program) is a string and T (the one-time-memory) encodes a list of 2ℓ strings, for some ℓ. Algorithm Ex, on input of strings P and x, and given access to an oracle $\mathcal{O}$, returns a string $y \leftarrow \mathsf{Ex}^{\mathcal{O}}(P,x)$. We require the following *correctness* condition of $\Pi = (\mathsf{Co}, \mathsf{Ex})$: if $(P,T) \leftarrow \mathsf{Co}(1^k, f)$ and $x \in \{0,1\}^{f.n}$ then $\mathsf{Ex}^{\mathrm{OTM}_T(\cdot,\cdot)}(P,x) = \mathsf{ev}(f,x)$.

The security of $\Pi = (\mathsf{Co}, \mathsf{Ex})$ will be relative to a side-information function Φ; the value $\phi = \Phi(f)$ captures the information about f that P is allowed to reveal. So fix a one-time compiler $\Pi = (\mathsf{Co}, \mathsf{Ex})$, an adversary $\mathcal{A}$, a security parameter k, and a string f. (1) Consider the distribution $\mathsf{Real}_{\Pi,\mathcal{A},f}(k)$ determined by the following experiment: first, sample $(P,T) \leftarrow \mathsf{Co}(1^k, f)$; then, run $\mathcal{A}^{\mathrm{OTM}_T(\cdot)}(1^k, P)$ and output whatever $\mathcal{A}$ outputs. (2) Alternatively, fix a one-time compiler $\Pi = (\mathsf{Co}, \mathsf{Ex})$, an information function Φ, a simulator $\mathcal{S}$, a security parameter k, and a string f. Consider the distribution $\mathsf{Fake}_{\Pi,\Phi,\mathcal{S},f}(k)$ determined by the following experiment: run $S^{\mathrm{OTP}_f(\cdot)}(1^k, \Phi(f))$ and output whatever S outputs. For $\mathcal{D}$ an algorithm and Π, Φ, $\mathcal{A}$, $\mathcal{S}$, and k as above, let

$$\mathbf{Adv}^{\mathrm{otc}}_{\Pi,\Phi,\mathcal{A},\mathcal{S},\mathcal{D}}(k) = \Pr[(f,\sigma) \leftarrow \mathcal{D}(1^k); v \leftarrow\!\!\!\!\!\!\$\ \mathsf{Real}_{\Pi,\mathcal{A},f}(k) \colon\ \mathcal{D}(\sigma, v) \Rightarrow 1] - \Pr[(f,\sigma) \leftarrow \mathcal{D}(1^k); v \leftarrow\!\!\!\!\!\!\$\ \mathsf{Fake}_{\Pi,\Phi,\mathcal{S},f}(k) \colon\ \mathcal{D}(\sigma, v) \Rightarrow 1]$$

One-time compiler Π is said to be (OTC-) *secure* with respect to side-information function Φ if for any PPT adversary $\mathcal{A}$ there is a PPT simulator $\mathcal{S}$ such that for all PPT distinguishers $\mathcal{D}$, function $\mathbf{Adv}^{\mathrm{otc}}_{\Pi,\Phi,\mathcal{A},\mathcal{S},\mathcal{D}}(k)$ is negligible.

Constructing an OTC from a garbling scheme. A circuit-garbling scheme $\mathcal{G} = (\mathsf{Gb}, \mathsf{En}, \mathsf{De}, \mathsf{Ev}, \mathsf{ev})$ can be turned into a one-time compiler $\Pi = (\mathsf{Co}, \mathsf{Ex})$ in a natural way: let $\mathrm{OTC}[\mathcal{G}] = (\mathsf{Co}, \mathsf{Ex})$ be defined as follows. (1) $\mathsf{Co}(1^k, f)$: let $(F, e, d) \leftarrow \mathsf{Gb}(f)$ and return (P,T) where $P = (F,d)$ and $T = e$.

(2) $\mathsf{Ex}^{\mathcal{O}}(P,x)$: Let $(F,d) \leftarrow P$, let $x_1 \cdots x_n \leftarrow x$, query oracle $\mathcal{O}$ on $(1,x_1)$, $\ldots,(n,x_n)$ to obtain $X_1,\ldots,X_n$, respectively, and return $\mathsf{De}(d,\mathsf{Ev}(F,X))$ with $X=(X_1,\ldots,X_n)$. The proof of the following is in the full paper [3].

Theorem 4. If $\mathcal{G}$ is a prv2-secure garbling scheme over side-information function Φ then $\mathrm{OTC}[\mathcal{G}]$ is OTC-secure with respect to side-information Φ.

The straightforwardness of the construction and its trivial proof are, we believe, points in our favor, evidence of our claim that the garbling scheme abstraction and appropriate security notions for it engender applications in direct, simple and less error-prone ways.

Separation. In the full paper [3], we elaborate on how Proposition 1 gives an example of a garbling scheme $\mathcal{G}$ such that $\mathrm{OTC}[\mathsf{OMSS}[\mathcal{G}]]$ is not otc-secure. We explain why this refutes GKR's claim [13] that their construction provides a secure one-time compiler assuming one-way functions.

4 Obliviousness, Authenticity and Secure Outsourcing

We define obliviousness and authenticity, both with either the coarse-grained or fine-grained adaptivity. We show how to achieve these goals, in combination with adaptive privacy, via generic transforms and in the standard model. In the full paper [3] we provide more efficient transforms in the ROM. Finally we apply this to obtain extremely simple and modular designs, and security proofs, for verifiable outsourcing schemes based on the paradigm of GGP [9].

Obliviousness. Intuitively, a garbling scheme is *oblivious* if garbled function F and garbled input X, these corresponding to f and x, reveal nothing of f or x beyond side-information $\Phi(f)$. In particular, possession F and X will not allow the calculation of $y=\mathsf{ev}(f,x)$.

The formal definition for *static* obliviousness is from BHR [4]. See the top of Fig. 6. We add to this two new definitions, to incorporate either coarse-grained or fine-grained adaptive security. See the rest of Fig. 6. Fine-grained adaptive security continues to require that $\mathcal{G}$ be projective. The games used for defining obliviousness closely mirror their privacy counterparts. The first important difference is that the adversary does not get the decoding function d. The second important difference is that the simulator must do without $y=\mathsf{ev}(f,x)$. For a garbling scheme $\mathcal{G}$, side-information Φ, simulator $\mathcal{S}$, adversary $\mathcal{A}$, and security parameter $k\in\mathbb{N}$, we let $\mathbf{Adv}^{\mathrm{obv},\Phi,\mathcal{S}}_{\mathcal{G}}(\mathcal{A},k)=2\Pr[\mathrm{Obv}^{\mathcal{A}}_{\mathcal{G},\Phi,\mathcal{S}}(k)]-1$, $\mathbf{Adv}^{\mathrm{obv1},\Phi,\mathcal{S}}_{\mathcal{G}}(\mathcal{A},k)=2\Pr[\mathrm{Obv1}^{\mathcal{A}}_{\mathcal{G},\Phi,\mathcal{S}}(k)]-1$, and finally $\mathbf{Adv}^{\mathrm{obv2},\Phi,\mathcal{S}}_{\mathcal{G}}(\mathcal{A},k)=2\Pr[\mathrm{Obv2}^{\mathcal{A}}_{\mathcal{G},\Phi,\mathcal{S}}(k)]-1$. Garbling scheme $\mathcal{G}$ is obv-secure with respect to Φ if for every PPT $\mathcal{A}$ there exists a simulator $\mathcal{S}$ such that $\mathbf{Adv}^{\mathrm{obv},\Phi,\mathcal{S}}_{\mathcal{G}}(\mathcal{A},k)$ is negligible. We similarly define obv1 and obv2 security. For $\mathrm{xxx}\in\{\mathrm{obv},\mathrm{obv1},\mathrm{obv2}\}$ we let $\mathsf{GS}(\mathrm{xxx},\Phi)$ denote the set of all garbling schemes that are xxx-secure over Φ.

Fig. 6 also formalizes the games underlying three definitions of authenticity, capturing an adversary's inability to create from F and X a garbled output

proc GARBLE(f, x) — $\mathrm{Obv}_{\mathcal{G},\Phi,\mathcal{S}}$

$b \twoheadleftarrow \{0,1\}$
if $x \notin \{0,1\}^{f.n}$ **then return** $\bot$
if $b = 1$ **then** $(F, e, d) \leftarrow \mathsf{Gb}(1^k, f)$, $X \leftarrow \mathsf{En}(e, x)$
else $(F, X) \leftarrow \mathcal{S}(1^k, \Phi(f))$
return (F, X)

proc GARBLE(f)	**proc** INPUT(x) — $\mathrm{Obv1}_{\mathcal{G},\Phi,\mathcal{S}}$
$b \twoheadleftarrow \{0,1\}$	**if** $x \notin \{0,1\}^{f.n}$ **then return** $\bot$
if $b = 1$ **then** $(F, e, d) \leftarrow \mathsf{Gb}(1^k, f)$	**if** $b = 1$ **then** $X \leftarrow \mathsf{En}(e, x)$
else $F \leftarrow \mathcal{S}(1^k, \Phi(f), 0)$	**else** $X \leftarrow \mathcal{S}(1)$
return F	**return** X

proc GARBLE(f)	**proc** INPUT(i, c) — $\mathrm{Obv2}_{\mathcal{G},\Phi,\mathcal{S}}$
$b \twoheadleftarrow \{0,1\}$; $n \leftarrow f.n$; $Q \leftarrow \emptyset$; $\sigma \leftarrow \varepsilon$	**if** $i \notin \{1, \ldots, n\} \setminus Q$ **then return** $\bot$
if $b = 1$ **then**	$x_i \leftarrow c$; $Q \leftarrow Q \cup \{i\}$
$(F, (X_1^0, X_1^1, \ldots, X_n^0, X_n^1), d) \leftarrow \mathsf{Gb}(1^k, f)$	**if** $b = 1$ **then** $X_i \leftarrow X_i^{x_i}$
else $F \leftarrow \mathcal{S}(1^k, \Phi(f), 0)$	**else** $X_i \leftarrow \mathcal{S}(i, \lvert Q \rvert)$
return F	**return** X_i

proc GARBLE(f, x) — $\mathrm{Aut}_{\mathcal{G}}$

if $x \notin \{0,1\}^{f.n}$ **then return** $\bot$
$(F, e, d) \leftarrow \mathsf{Gb}(1^k, f)$, $X \leftarrow \mathsf{En}(e, x)$
return (F, X)

proc GARBLE(f)	**proc** INPUT(x) — $\mathrm{Aut1}_{\mathcal{G}}$
$(F, e, d) \leftarrow \mathsf{Gb}(1^k, f)$	**if** $x \notin \{0,1\}^{f.n}$ **then return** $\bot$
return F	$X \leftarrow \mathsf{En}(e, x)$
	return X

proc GARBLE(f)	**proc** INPUT(i, c) — $\mathrm{Aut2}_{\mathcal{G}}$
$n \leftarrow f.n$; $Q \leftarrow \emptyset$; $\sigma \leftarrow \varepsilon$	**if** $i \notin \{1, \ldots, n\} \setminus Q$ **then return** $\bot$
$(F, (X_1^0, X_1^1, \ldots, X_n^0, X_n^1), d) \leftarrow \mathsf{Gb}(1^k, f)$	$x_i \leftarrow c$; $Q \leftarrow Q \cup \{i\}$, $X_i \leftarrow X_i^{x_i}$
return F	**if** $\lvert Q \rvert = n$ **then** $X \leftarrow (X_1, \ldots, X_n)$
	return X_i

Fig. 6. Obliviousness (top). Games for defining the obv, obv1, and obv2 security of $\mathcal{G} = (\mathsf{Gb}, \mathsf{En}, \mathsf{De}, \mathsf{Ev}, \mathsf{ev})$. For each game, FINALIZE(b') returns $(b = b')$. **Authenticity (bottom).** Games for defining the aut, aut1, and aut2 security of $\mathcal{G} = (\mathsf{Gb}, \mathsf{En}, \mathsf{De}, \mathsf{Ev}, \mathsf{ev})$. Procedure FINALIZE$(Y)$ of each game returns $(\mathsf{De}(d, Y) \neq \bot$ **and** $Y \neq \mathsf{Ev}(F, X))$.

$Y \neq F(X)$ that will be deemed authentic. The static definition of BHR [4] is strengthened either to allow the adversary to specify x subsequent to obtaining F, or, stronger, the bits of x are provided one-by-one, each corresponding token then issued. For the second case, game Aut2, the garbling scheme must once again be projective. For a garbling scheme $\mathcal{G}$, adversary $\mathcal{A}$, and security parameter $k \in \mathbb{N}$, we let $\mathbf{Adv}^{\mathrm{aut}}_{\mathcal{G}}(\mathcal{A}, k) = 2\Pr[\mathrm{Aut}^{\mathcal{A}}_{\mathcal{G}}(k)] - 1$, $\mathbf{Adv}^{\mathrm{aut1}}_{\mathcal{G}}(\mathcal{A}, k) =$

$\mathbf{proc}\ \mathsf{Gb}_1(1^k, f)$
$(F, e, d) \leftarrow \mathsf{Gb}(1^k, f)$
$F' \twoheadleftarrow \{0,1\}^{|F|}, \quad d' \twoheadleftarrow \{0,1\}^{|d|}$
$F_1 \leftarrow F \oplus F', \quad K \twoheadleftarrow \{0,1\}^k, \quad d_1 \leftarrow (d \oplus d', K)$
$\mathrm{tag} \leftarrow \mathsf{F}_K(d'), \quad e_1 \leftarrow (e, d', F', \mathrm{tag})$
$\mathbf{return}\ (F_1, e_1, d_1)$

$\mathbf{proc}\ \mathsf{En}_1(e_1, x)$
$(e, d', F', \mathrm{tag}) \leftarrow e_1$
$\mathbf{return}\ (\mathsf{En}(e, x), d', F', \mathrm{tag})$

$\mathbf{proc}\ \mathsf{Ev}_1(F_1, X_1)$
$(X, d', F', \mathrm{tag}) \leftarrow X_1, \quad F \leftarrow F_1 \oplus F'$
$Y \leftarrow \mathsf{Ev}(F, X)$
$\mathbf{return}\ (Y, d', \mathrm{tag})$

$\mathbf{proc}\ \mathsf{De}_1(d_1, Y_1)$
$(Y, d', \mathrm{tag}) \leftarrow Y_1$
$(D, K) \leftarrow d_1, \quad d \leftarrow D \oplus d'$
$\mathbf{if}\ \mathrm{tag} \neq \mathsf{F}_K(d')\ \mathbf{then\ return}\ \perp$
$\mathbf{return}\ \mathsf{De}(d, Y)$

Fig. 7. Scheme $\mathsf{all\text{-}to\text{-}all1}[\mathcal{G}] = (\mathsf{Gb}_1, \mathsf{En}_1, \mathsf{De}_1, \mathsf{Ev}_1, \mathsf{ev}) \in \mathsf{GS}(\mathrm{prv1}, \Phi) \cap \mathsf{GS}(\mathrm{obv1}, \Phi) \cap \mathsf{GS}(\mathrm{aut1})$ obtained from scheme $\mathcal{G} = (\mathsf{Gb}, \mathsf{En}, \mathsf{De}, \mathsf{Ev}, \mathsf{ev}) \in \mathsf{GS}(\mathrm{prv}, \Phi) \cap \mathsf{GS}(\mathrm{obv}, \Phi) \cap \mathsf{GS}(\mathrm{aut})$. The transform uses a PRF $\mathsf{F} : \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^k$.

$2\Pr[\mathrm{Aut1}^{\mathcal{A}}_{\mathcal{G}}(k)] - 1$, and $\mathbf{Adv}^{\mathrm{aut2}}_{\mathcal{G}}(\mathcal{A}, k) = 2\Pr[\mathrm{Aut2}^{\mathcal{A}}_{\mathcal{G}}(k)] - 1$. Garbling scheme $\mathcal{G}$ is aut-secure with respect to Φ if for every PPT $\mathcal{A}$ $\mathbf{Adv}^{\mathrm{aut}}_{\mathcal{G}}(\mathcal{A}, k)$ is negligible. We similarly define aut1 and aut2 security. For $\mathrm{xxx} \in \{\mathrm{aut}, \mathrm{aut1}, \mathrm{aut2}\}$ we let $\mathsf{GS}(\mathrm{xxx})$ denote the set of all garbling schemes that are xxx-secure.

Achieving obv1 and aut1 security. It is tempting to think that the prv-to-prv1 operator in Fig. 4 also promotes xxx-security, with $\mathrm{xxx} \in \{\mathrm{obv}, \mathrm{aut}\}$, to xxx1-security, but it does not. We now show how to change prv-to-prv1 to an operator all-to-all1 that promotes any $\mathrm{xxx} \in \{\mathrm{prv}, \mathrm{obv}, \mathrm{aut}\}$ to being xxx1 secure. See Fig. 7. The proof of the following is in the full paper [3].

Theorem 5. **(1)** For any Φ and any $\mathrm{xxx} \in \{\mathrm{prv}, \mathrm{obv}\}$, if $\mathcal{G} \in \mathsf{GS}(\mathrm{xxx}, \Phi)$ then $\mathsf{all\text{-}to\text{-}all1}[\mathcal{G}] \in \mathsf{GS}(\mathrm{xxx1}, \Phi)$ **(2)** If $\mathcal{G} \in \mathsf{GS}(\mathrm{aut})$ then $\mathsf{all\text{-}to\text{-}all1}[\mathcal{G}] \in \mathsf{GS}(\mathrm{aut1})$ **(3)** If $\mathcal{G} \in \mathsf{GS}(\mathsf{proj})$ then $\mathsf{all\text{-}to\text{-}all1}[\mathcal{G}] \in \mathsf{GS}(\mathsf{proj})$.

Achieving obv2 and aut2 security. The transform to promote coarse-grained to fine-grained security is unchanged. We let $\mathsf{all1\text{-}to\text{-}all2} = \mathsf{prv1\text{-}to\text{-}prv2}$ be the transform at the bottom of Fig. 4. We claim it has additional features captured by the following, whose proof is in the full paper [3].

Theorem 6. **(1)** For any Φ and any $\mathrm{xxx} \in \{\mathrm{prv}, \mathrm{obv}\}$ if $\mathcal{G}_1 \in \mathsf{GS}(\mathrm{xxx1}, \Phi) \cap \mathsf{GS}(\mathsf{proj})$ then $\mathsf{all1\text{-}to\text{-}all2}[\mathcal{G}_1] \in \mathsf{GS}(\mathrm{xxx2}, \Phi) \cap \mathsf{GS}(\mathsf{proj})$ **(2)** If $\mathcal{G}_1 \in \mathsf{GS}(\mathrm{aut1}) \cap \mathsf{GS}(\mathsf{proj})$ then $\mathsf{all1\text{-}to\text{-}all2}[\mathcal{G}_1] \in \mathsf{GS}(\mathrm{aut2}) \cap \mathsf{GS}(\mathsf{proj})$.

Outsourcing definitions. Towards the application to secure outsourcing, we begin with the definitions, following GGP [9]. An outsourcing scheme $\Pi = (\mathsf{Gen}, \mathsf{Inp}, \mathsf{Out}, \mathsf{Comp}, \mathsf{ev})$ is a tuple of PT algorithms that, intuitively, will be run partly on a *client* and partly on a *server*. Generation algorithm Gen is run by the client on input of the unary encoding 1^k and a string f describing the function $\mathsf{ev}(f, \cdot) : \{0,1\}^{f.n} \to \{0,1\}^{f.m}$ to be evaluated (so that ev, like in a garbling scheme, is a deterministic evaluation algorithm) to get back a public key pk that

is sent to the server and a secret key sk that is kept by the client. Algorithm Inp is run by the client on input pk, sk and $x \in \{0,1\}^{f.n}$ to return a garbled input X that is sent to the server. Associated state information St is preserved by the client. Algorithm Comp is run by the server on input pk, X to get a garbled output Y that is returned to the client. The latter runs deterministic algorithm Out on pk, sk, Y, St to get back $y \in \{0,1\}^{f.n} \cup \{\perp\}$. Correctness requires that for all $k \in \mathbb{N}$, all $f \in \{0,1\}^*$, and all $x \in \{0,1\}^{f.n}$, if $(pk, sk) \leftarrow \mathsf{Gen}(1^k, f)$, $(X, St) \leftarrow \mathsf{Inp}(pk, sk, x)$, $Y \leftarrow \mathsf{Comp}(pk, X)$, and $y \leftarrow \mathsf{Out}(pk, sk, Y, St)$, then $y = \mathsf{ev}(f, x)$. Our syntax is the same as that of GGP [9] except for distinguishing between functions and their descriptions, as represented the addition of ev to the list.

The games OSVF_{Π} and $\mathrm{OSPR}_{\Pi,\Phi,\mathcal{S}_{\mathrm{os}}}$ of Fig. 8 are used to define *verifiability* and *privacy* of an outsourcing scheme $\Pi = (\mathsf{Gen}, \mathsf{Inp}, \mathsf{Out}, \mathsf{Comp}, \mathsf{ev})$, where Φ is a side-information function and $\mathcal{S}_{\mathrm{os}}$ is a simulator. In both games, the adversary is allowed only one GetPK query, and this must be its first oracle query. For adversaries $\mathcal{A}_{\mathrm{os}}$ and $\mathcal{B}_{\mathrm{os}}$, we let $\mathbf{Adv}^{\mathrm{osvf}}_{\Pi}(\mathcal{A}_{\mathrm{os}}, k) = \Pr[\mathrm{OSVF}^{\mathcal{A}_{\mathrm{os}}}_{\Pi}(k)]$ and $\mathbf{Adv}^{\mathrm{ospr},\Phi,\mathcal{S}_{\mathrm{os}}}_{\Pi}(\mathcal{B}_{\mathrm{os}}, k) = 2\Pr[\mathrm{OSPR}^{\mathcal{B}_{\mathrm{os}}}_{\Pi,\Phi,\mathcal{S}_{\mathrm{os}}}(k)] - 1$. We say that Π is verifiable if $\mathbf{Adv}^{\mathrm{osvf}}_{\Pi}(\mathcal{A}_{\mathrm{os}}, \cdot)$ is negligible for all PT adversaries $\mathcal{A}_{\mathrm{os}}$. We say that Π is private over Φ if for all PT adversaries $\mathcal{B}_{\mathrm{os}}$ there is a PT simulator $\mathcal{S}_{\mathrm{os}}$ such that $\mathbf{Adv}^{\mathrm{ospr},\Phi,\mathcal{S}_{\mathrm{os}}}_{\Pi}(\mathcal{A}_{\mathrm{os}}, \cdot)$ is negligible. An adversary is said to be one-time if it makes only one Input query. We say that Π is one-time verifiable if $\mathbf{Adv}^{\mathrm{osvf}}_{\Pi}(\mathcal{A}_{\mathrm{os}}, \cdot)$ is negligible for all PT one-time adversaries $\mathcal{A}_{\mathrm{os}}$. We say that Π is one-time private over Φ if for all PT one-time adversaries $\mathcal{B}_{\mathrm{os}}$ there is a PT simulator $\mathcal{S}_{\mathrm{os}}$ such that $\mathbf{Adv}^{\mathrm{ospr},\Phi,\mathcal{S}_{\mathrm{os}}}_{\Pi}(\mathcal{A}_{\mathrm{os}}, \cdot)$ is negligible.

Our verifiability definition coincides with that of GGP [9] but our privacy definition is stronger: it requires not just "input privacy" (concealing each input x) but, also, privacy of the function f (relative to Φ). (As in our garbling definitions this is subject to $\Phi(f)$ being revealed). Also, while GGP use an indistinguishability-style formalization, we use a simulation-style one, as this is stronger for some side-information functions.

To be "interesting" the work of the client in an outsourcing scheme should be less than the work required to compute the function directly, for otherwise outsourcing is not buying anything. An outsourcing scheme is said to be non-trivial if this condition is met.

From garbling to outsourcing. GGP show how to use FHE to turn any one-time verifiable and private outsourcing scheme into a fully verifiable and private one. This allows us to focus on designing the former. We show how a garbling scheme that is both aut1 and obv1 secure immediately implies a one-time verifiable and private outsourcing scheme. The construction, given in Fig. 8, is very direct, and the proof of the following, given in the full paper [3], is trivial, points which reinforce our claim that the garbling scheme abstraction and adaptive security may be easily used in applications:

Theorem 7. If $\mathcal{G} \in \mathsf{GS}(\mathrm{obv1}, \Phi) \cap \mathsf{GS}(\mathrm{aut1})$ then outsourcing scheme $\Pi[\mathcal{G}]$ is one-time verifiable and also one-time private over Φ.

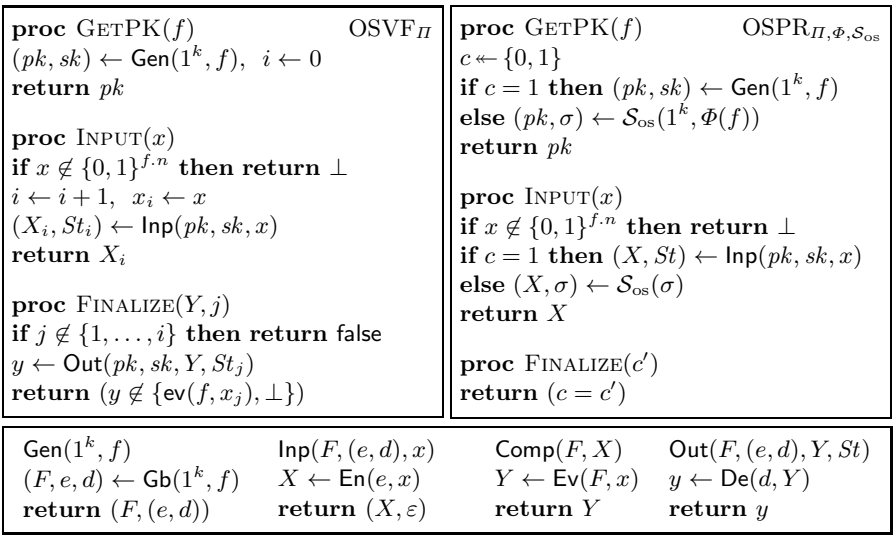

Fig. 8. Games to define the **verifiability** (OSVF) and **privacy** (OSPR) of outsourcing scheme $\Pi = (\mathsf{Gen}, \mathsf{Inp}, \mathsf{Out}, \mathsf{Comp}, \mathsf{ev})$. **Bottom:** constructing the outsourcing scheme $\Pi[\mathcal{G}] = (\mathsf{Gen}, \mathsf{Inp}, \mathsf{Out}, \mathsf{Comp}, \mathsf{ev})$ from garbling scheme $\mathcal{G} = (\mathsf{Gb}, \mathsf{En}, \mathsf{De}, \mathsf{Ev}, \mathsf{ev})$.

A benefit of our modular approach is that we may use any obv1 + aut1 garbling scheme as a starting point while GGP were tied to the scheme of [17]. However, the latter scheme is not adaptively secure, which brings us to our next point.

Discussion. GGP give a proof that their outsourcing scheme is one-time verifiable assuming the encryption scheme underlying the garbled-circuit construction of [17] meets the condition called Yao-secure in [17]. However, their proof has a gap. Quoting [9, p. 12 of Aug 2010 ePrint version]: "For any two values x, x' with $f(x) = f(x')$, the security of Yao's protocol implies that no efficient player P_2 can distinguish if x or x' was used." This claim is correct if both x and x' are chosen independently of the randomness in the garbled circuit. But in their setting, the string x is chosen *after* the adversary sees the garbled circuit, and the security proof given by [17] no longer applies.

One may try to give a new proof that the LP garbling scheme satisfies aut1 security. However, this seems to be difficult. Intuitively, an adaptive attack on the garbling scheme allows the adversary to mount a key-revealing selective-opening (SOA-K) attack on the underlying encryption scheme. But SOA-K secure encryption is notoriously hard to achieve [2]. The only known way to achieve it is via non-committing encryption [5, 6, 8], which is only possible with keys as long as the total number of bits of message ever encrypted [19], so the outsourcing scheme may fail to be non-trivial.

This brings us to a more full discussion of non-triviality. The obv1 + aut1 secure scheme obtained via our all-to-all1 transform has long garbled inputs, so the one-time verifiable outsourcing scheme yielded by Theorem 7, while secure,

is not non-trivial. Our ROM transforms coupled with Theorem 7 yield a non-trivial one-time outsourcing scheme in the ROM but the FHE-based method of GGP of lifting to a many-time scheme fails in the ROM. Finding a obv1 + aut1 garbling scheme with short garbled inputs in the standard model under standard assumptions is an open problem. We think Theorem 7 is still useful because it can be used at any point such a scheme emerges. All this again is an indication of the subtleties and hidden challenges underlying adaptive security of garbled circuits that seem to have been overlooked in the literature.

Acknowledgments. Thanks to the ASIACRPYT reviewers for their helpful comments, and thanks to the NSF for their continuing support: Bellare was supported in part by NSF grants CNS-1116800, CNS 0904380 and CCF-0915675, while Hoang and Rogaway were supported in part by NSF grant CNS 0904380.

References

1. Applebaum, B., Ishai, Y., Kushilevitz, E.: From Secrecy to Soundness: Efficient Verification via Secure Computation. In: Abramsky, S., Gavoille, C., Kirchner, C., Meyer auf der Heide, F., Spirakis, P.G. (eds.) ICALP 2010, Part I. LNCS, vol. 6198, pp. 152–163. Springer, Heidelberg (2010)
2. Bellare, M., Dowsley, R., Waters, B., Yilek, S.: Standard Security Does Not Imply Security against Selective-Opening. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 645–662. Springer, Heidelberg (2012)
3. Bellare, M., Hoang, V., Rogaway, P.: Adaptively secure garbling with applications to one-time programs and secure outsourcing. Cryptology ePrint Archive (2012)
4. Bellare, M., Hoang, V., Rogaway, P.: Foundations of garbled circuits. In: ACM Computer and Communications Security (CCS 2012). Association for Computing Machinery. ACM (2012); Full version as ePrint Archive, Report 2012/265 (May 2012)
5. Canetti, R., Feige, U., Goldreich, O., Naor, M.: Adaptively secure multi-party computation. In: 28th ACM STOC, pp. 639–648. ACM Press (May 1996)
6. Choi, S.G., Dachman-Soled, D., Malkin, T., Wee, H.: Improved Non-committing Encryption with Applications to Adaptively Secure Protocols. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 287–302. Springer, Heidelberg (2009)
7. Chung, K.-M., Kalai, Y., Vadhan, S.: Improved Delegation of Computation Using Fully Homomorphic Encryption. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 483–501. Springer, Heidelberg (2010)
8. Damgård, I., Nielsen, J.B.: Improved Non-committing Encryption Schemes Based on a General Complexity Assumption. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 432–450. Springer, Heidelberg (2000)
9. Gennaro, R., Gentry, C., Parno, B.: Non-interactive Verifiable Computing: Outsourcing Computation to Untrusted Workers. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 465–482. Springer, Heidelberg (2010)
10. Goldreich, O.: Foundations of Cryptography: Basic Applications, vol. 2. Cambridge University Press, Cambridge (2004)
11. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or a completeness theorem for protocols with honest majority. In: Aho, A. (ed.) STOC, pp. 218–229. ACM (1987)

12. Goldwasser, S., Kalai, Y.T., Rothblum, G.N.: One-time programs. Manuscript, full version of [13] (July 2012)
13. Goldwasser, S., Kalai, Y.T., Rothblum, G.N.: One-Time Programs. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 39–56. Springer, Heidelberg (2008)
14. Goyal, V., Ishai, Y., Sahai, A., Venkatesan, R., Wadia, A.: Founding Cryptography on Tamper-Proof Hardware Tokens. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 308–326. Springer, Heidelberg (2010)
15. Ishai, Y., Kushilevitz, E.: Randomizing polynomials: A new representation with applications to round-efficient secure computation. In: 41st FOCS, pp. 294–304. IEEE Computer Society Press (November 2000)
16. Kamara, S., Wei, L.: Special-purpose garbled circuits. (manuscript, 2012)
17. Lindell, Y., Pinkas, B.: A proof of security of Yao's protocol for two-party computation. Journal of Cryptology 22(2), 161–188 (2009)
18. Naor, M., Pinkas, B., Sumner, R.: Privacy preserving auctions and mechanism design. In: Proceedings of the 1st ACM Conference on Electronic Commerce, pp. 129–139. ACM (1999)
19. Nielsen, J.B.: Separating Random Oracle Proofs from Complexity Theoretic Proofs: The Non-committing Encryption Case. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 111–126. Springer, Heidelberg (2002)
20. Yao, A.: Protocols for secure computations (extended abstract). In: FOCS, pp. 160–164. IEEE Computer Society (1982)
21. Yao, A.: How to generate and exchange secrets. In: 27th Annual Symposium on Foundations of Computer Science, pp. 162–167. IEEE (1986)