# New Proof Techniques for Adaptive Security

## Thesis Proposal

Zahra Jafargholi

**Abstract**

Selective security refers to the case where the attacker decides on some parameters of the attack before the attack even begins. In contrast, adaptive security refers to the case where the attacker can make decisions throughout the course of the attack. Therefore these decisions can depend on the information the attacker receives during the attack.

One can use complexity-leveraging to reduce the problem of proving adaptive security to proving selective security. This technique which involves guessing the decisions the attacker makes during the attack, leads to an exponential loss in security reduction. Here we look at two settings, where we prove adaptive security *from scratch* – without reducing to selective security.

The first problem, Generalized Selective Decryption (GSD) Game, captures security requirements of Broadcast and Multicast Encryption protocols. We give an analysis of the system that proves GSDG is adaptively secure with only a quasi-polynomial loss in the security reduction. In study of the second problem, Adaptively Secure Garbling Schemes, we devise a new garbling scheme that has Yao's garbling scheme at heart. Our adaptively secure garbling scheme has a garbled input of length proportional to the circuit's width. This is a considerable improvement over the previous schemes where the garbled input grew with the size of the entire circuit. We propose to study the techniques used. Is it possible to generalize the techniques and use them in different setting? What separate the two seemingly related techniques? Finally, we propose using the new techniques to analyze adaptive security of Yao's garbling scheme.

# 1 Introduction

A common way to define security properties expected from a cryptographic primitive is to design a game that captures its interaction with the outside world. Often these games have two participants: one who runs the cryptographic primitive, called the *challenger* and one who is trying to break the primitive, called the *adversary* or *attacker*. Generally the attacker can make many choices during the course of the attack. The information he receives from the challenger can influence his future decisions – therefore these attacks are *adaptive*. There are cases where adaptive security is unnecessarily strong. Some attackers might be restricted to make some or all of their choices before the attack begins. These attacks are partially or completely *selective*. Clearly the restricted adversary it less powerful than the adaptive one. Therefore it is easier to prove that a primitive is *selectively* secure, than to prove it *adaptively* secure. Many useful and interesting primitives are proven to be selectively secure but there is no "good" analysis of their resistance against an adaptive attack. In this work we study techniques to prove adaptive security for two such primitives.

**Garbled Circuits.** *garbling scheme* (also referred to as a randomized encoding) can be used to garble a circuit $C$ and an input $x$ to derive a garbled circuit $\widetilde{C}$ and a garbled input $\widetilde{x}$. It's possible to evaluate $\widetilde{C}$ on $\widetilde{x}$ and get the correct output $C(x)$. However, the garbled values $\widetilde{C}, \widetilde{x}$ should not reveal anything else beyond this.

The notion of garbled circuits was introduced by Yao in (oral presentations of) [Yao82, Yao86], and can be instantiated based on one-way functions. Garbled circuits have since found countless applications in diverse areas of cryptography, most notably to secure function evaluation (SFE) starting with Yao's work, but also in parallel cryptography [AIK04, AIK05], verifiable computation [GGP10, AIK10], software protection [GKR08, GIS$^+$10], functional encryption [SS10, GVW12, GKP$^+$13], and many others. Once again there are various ways of defining security. What they all have in common (so far) is that the garbled circuit is only used to compute on one garbled input[1]. What separates them is, when this single input is chosen, garbled and published. If the garbled circuit and input are chosen and published at the same time, we have a selective scenario. If the input is chosen after the garbled circuit is published, we have an adaptive scenario. In practice, often the same user who creates the garbled circuit, chooses the input too. So one might hope that the attacker cannot make any adaptive decisions in either cases. However the attacker may be able to influence the choice of the input. Therefore seeing the garble circuit might give him an advantage in the attack. Yao's garbling scheme, uses a symmetric key encryption scheme to garble each gate of the circuit, is selectively secure. There is an easy reduction from an adaptive attack on Yao's garbling scheme to a selective one. We need to make sure that the adversary does not learn anything from the garbled circuit that can help him with the choice of the input. We can use One-Time Pad (OTP) to mask the entire circuit. This way, what adversary sees as the garbled circuit is truly random and can't give him any useful information. When the adversary receives the garbled input, he will also get the key to OTP. In fact we are really playing the selective security game, because we can decide on what the key of the One-Time Pad, (and consequently the decrypted garbled circuit) is going to be after the attacker has chosen his input. The problem is that the key of OTP is as long as its message, which here is the garbled circuit. We changed Yao's garbling scheme by adding the One-Time Pad and in this new scheme the garbled input (which includes the key of OTP) is as large as the garbled circuit.

In this work, [HJO$^+$15], we construct the first adaptively secure garbling scheme whose garbled input is smaller than the circuit size and which only relies on the existence of one-way functions.

[1] There are also "reusable" garbled circuits where this is not the case, but we do not consider them here.

In particular, our construction simply encrypts a Yao garbled circuit with a *somewhere equivocal* symmetric-key encryption scheme, which is a new primitive that we define and construct from one-way functions. This new primitive lets us decide on part of the garbled circuit after the adversary has chosen his input. This way, we have taken away part of the adaptivity power of the adversary, similar to using One-Time Pad, but not all of it. Furthermore, the key of this new encryption scheme is much smaller than the size of the circuit. We get various provably secure instantiations of the above approach depending on how we set the parameters of the encryption scheme.

**Generalized Selective Decryption Game.** Broadcast and Multicast encryptions protocols apply to online networks, where each user is assigned a secret key by the network's administrator. If a group of the online users wants to communicate with each other privately, the administrator creates a new key (assigned to that group) and broadcasts this new key, encrypted under the secret key of each user in the group. This way the group members can recover the new key and use it to communicate with each other. This is a very simple strategy of distributing keys. There are other complicated protocols, providing more efficient ways to distribute keys, but involve creating complex encryption chains of key encryptions under other keys. It becomes quite difficult to ensure that the public set of ciphertexts do not reveal anything about the secret keys. This is especially true when these networks are dynamic and users can login and logout at any time, which changes the chains of key ciphertexts. Panjwani [Pan07] introduces the *generalized selective decryption* (GSD) problem and uses it as an abstraction of security requirements of multicast encryption protocols [WGL00, MP06]. In GSD we have $n$ uniformly random keys, all hidden from the attacker at the beginning of the attack. The attacker can ask to see encryption of one key under another key. For example, the attacker asks: encrypt $k_1$ under $k_2$ and gets back $e \leftarrow \mathsf{Enc}_{k_2}(k_1)$. It helps to think of the keys as nodes in a graph, and create a directed edge for each encryption query. In our example, we add an edge from $k_2$ to $k_1$. The attacker can make many more queries like this[2]. At some point he asks to see one of the keys, in response he might get the key or he might get a new independent and uniformly random key. The attackers goal is to distinguish whether he got the real key or the random key.

This attack is surely adaptive. As the attack progresses, the attacker chooses *a*) the kind of graph created, *b*) where each nodes is placed; and finally *c*) which key is asked to be revealed. If we restrict the attacker to make any or all of these decisions before the attack starts, the attack becomes partially or completely selective. It is known that if the encryption scheme used is CPA-secure, then the selective GSD game is secure. This means there is a reduction from an attacker that breaks the security of selective GSD to an attacker that breaks the CPA security of the encryption scheme. There is also a reduction from an attacker that breaks adaptive security of GSD to one that breaks selective security of GSD. Consider all the possible strategies an adaptive attacker can devise; let's say there are $d$ such attack scenarios. Each such attack scenario can be used by a selective attacker. Therefore, by having the selective attacker choose a strategy randomly ahead of time, his probability of having a successful attack is at least $1/\mathrm{d}$ of the probability that the adaptive attacker succeeds. Unfortunately, $d$ is *exponentially large*[3] in $n$. This proves that an adaptive adversary is at most $d$ times (read: *exponentially*) more likely to succeed in an attack than the selective attacker.

We want a more careful analysis to avoid losing such a large factor in the reduction. We achieve this in [FJP15], by reducing the adaptive attack to a less-adaptive attack. Then reduce the less-adaptive attack to a partly-selective attack and so on. Finally we get to an attack which

[2]There are other rules that apply here

[3]The number of ways one can create a graph with $n$ nodes, alone, is exponential in $n$.

is "selective enough" so that we can reduce it to an attack on the CPA security of the encryption scheme. This means there is a reduction from an attacker that breaks the security of the "selective enough" game to an attacker that breaks the security of the CPA encryption. There are two reasons why this approach gives us a smaller reduction factor. First, even the "selective enough" version of the attack, requires us fixing very few parameters of the adaptive attack (instead of fixing all the attacker's choices). Second, we bypass a reduction to the selective attack. We reduce the "selective enough" attack straight to an attack on CPA security of the encryption scheme. Our reduction shows that an adaptive attack on GSD game[4] is at most $(n^{3(\log n)+5})$ times more likely to succeed than an attack on the CPA encryption.

**Proposal.** In these two works we made considerable improvements over the previous results. In both cases, we created variants of security games which are part adaptive and part selective, and we used new techniques to bridge the gap between adaptive and selective security more efficiently. Although a broad look at the techniques reveals clear similarities, the detailed works are very different and neither approaches is known to work for the other problem. We propose the study of these two approaches to better understand and formalize their nuances. We hope this study will lead us to other problems that can benefit from these techniques. One immediate candidate for such problems is adaptive security of Yao's original garbling scheme, which has not been proved with any non-trivial parameters yet.

## 2 Preliminaries

**General Notation.** For a positive integer $n$, we define the set $[n] := \{1, \ldots, n\}$We use the notation $x \leftarrow X$ for the process of sampling a value $x$ according to the distribution $X$. We say an adversary (or distinguisher) $\mathcal{D}$ is $t$-bounded if $\mathcal{D}$ runs in time $t$.

**Definition 1 (Indistinguishability)** Two distributions $X$ and $Y$ are $(\varepsilon, t)$-indistinguishable, denoted $Y \sim_{(\varepsilon,t)} X$ or $\Delta_t(Y, X) \leq \varepsilon$, if no $t$-bounded distinguisher $\mathcal{D}$ can distinguish them with advantage greater than $\varepsilon$, i.e.,

$$\Delta_t(Y, X) \leq \varepsilon \iff \forall \mathcal{D}_t : \left| \Pr\left[\mathcal{D}_t(X) = 1\right] - \Pr\left[\mathcal{D}_t(Y) = 1\right] \right| \leq \varepsilon \ . \qquad \diamondsuit$$

**Symmetric encryption.** A pair of algorithms $(\mathsf{Enc}, \mathsf{Dec})$ which take an input $k \in \{0,1\}^\lambda$, where $\lambda$ is the security parameter, and a message $m$ (or a ciphertext) from $\{0,1\}^*$ is a symmetric-key encryption scheme if for all $k, m$ we have $\mathsf{Dec}_k(\mathsf{Enc}_k(m)) = m$. Consider the game $\mathbf{Exp}^{\text{CPA-}b}_{\mathsf{Enc},\mathcal{D}}$ between a challenger $\mathcal{C}$ and a distinguisher $\mathcal{D}$. $\mathcal{C}$ chooses a uniformly random key $k \in \{0,1\}^\lambda$ and a bit $b \in \{0,1\}$. $\mathcal{D}$ can make encryption queries for messages $m$ and receives $\mathsf{Enc}_k(m)$. Finally, $\mathcal{D}$ outputs a pair $(m_0, m_1)$, is given $\mathsf{Enc}_k(m_b)$ and outputs a bit $b' \in \{0,1\}$, which is also the output of $\mathbf{Exp}^{\text{CPA-}b}_{\mathsf{Enc},\mathcal{D}}$. The encryption scheme is $(\varepsilon, t)$ CPA secure if $\mathbf{Exp}^{\text{CPA-0}}_{\mathsf{Enc},\mathcal{D}} \sim_{(\varepsilon,t)} \mathbf{Exp}^{\text{CPA-1}}_{\mathsf{Enc},\mathcal{D}}$.

In this work, we only consider polynomially bounded attackers, i.e. $t$ is a polynomial in security parameter.

[4] Restricted to trees.

# 3 Problem: Garbled Circuits

## 3.1 Background

**Garbled Circuits.** A *garbling scheme* (also referred to as a randomized encoding) can be used to garble a circuit $C$ and an input $x$ to derive a garbled circuit $\widetilde{C}$ and a garbled input $\widetilde{x}$. It's possible to evaluate $\widetilde{C}$ on $\widetilde{x}$ and get the correct output $C(x)$. However, the garbled values $\widetilde{C}, \widetilde{x}$ should not reveal anything else beyond this. In particular, there is a simulator that can simulate $\widetilde{C}, \widetilde{x}$ given only $C(x)$.

**On-line Complexity.** In many applications, the garbled circuit $\widetilde{C}$ can be computed in an *off-line* pre-processing phase before the input is known, and therefore the efficiency of this procedure may not be of paramount importance. On the other hand, once the input $x$ becomes available in the *on-line* phase, creating the garbled input $\widetilde{x}$ should be extremely efficient. Therefore, the main efficiency measure that we consider here is the *on-line complexity*, which is the time it takes to garble an input $x$, and hence also a bound on the size of $\widetilde{x}$. Ideally, the on-line complexity should only be linear in the input size $|x|$ and independent of the potentially much larger circuit size $|C|$. Note that, without any other restrictions on the structure of the garbling scheme, there is a trivial scheme where $\widetilde{C}$ is empty and $\widetilde{x} = C(x)$, whose on-line complexity is proportional to $|C|$. Therefore we think of non-trivial on-line complexity as being lower than $|C|$.

**Yao's Scheme.** Yao's garbling scheme already achieves essentially optimal on-line complexity, where the time to garble an input $x$ and the size of $\widetilde{x}$ are only linear in the input size $|x|$, independent of the circuit size.[5] However, it only realizes a selective security.

## 3.2 Selective vs. Adaptive Security.

Selective security is often unsatisfactory in precisely the scenarios envisioned in the off-line/on-line setting, where the garbled circuit $\widetilde{C}$ is given out first and the garbled input $\widetilde{x}$ is only given out later. Therefore, we need a stronger notion called *adaptive security*, defined via the following two stage game:

1. The adversary chooses a circuit $C$ and gets the garbled circuit $\widetilde{C}$.
2. After seeing $\widetilde{C}$ the adversary adaptively chooses an input $x$ and gets the garbled input $\widetilde{x}$.

In the real world $\widetilde{C}, \widetilde{x}$ are computed correctly using the garbling scheme, while in the simulated world they are created by a simulator who only gets the output $C(x)$ in step (2) of the game but does not get the input $x$. The adversary should not be able to distinguish these two worlds.

## 3.3 Our results

Our construction encrypts a Yao garbled circuit with a *somewhere equivocal* symmetric-key encryption scheme, which is a new primitive that we define and construct from one-way functions. The encrypted Yao garbled circuit is sent in the off-line phase, and the Yao garbled input along with the decryption key is sent in the on-line phase. We get various provably secure instantiations of the above approach depending on how we set the parameters of the encryption scheme.

[5] More precisely, in Yao's garbled circuits, the garbled input is of size $|x| \cdot \mathsf{poly}(\lambda)$ where $\lambda$ is the security parameter. The work of [AIKW13] shows how to reduce this to $|x| + \mathsf{poly}(\lambda)$ assuming stronger assumptions such as DDH, RSA or LWE.

As our main instantiation, we get a garbling scheme whose on-line complexity is $w \cdot \mathsf{poly}(\lambda)$ where $w$ is the *width* of the circuit and $\lambda$ is the security parameter, but is otherwise independent of the depth $d$ of the circuit.[6] Note that, if we think of the circuit as representing a Turing Machine or RAM computation, then the width $w$ of the circuit corresponds to the maximum of the input size $n$, output size $m$, and space complexity $s$ of the computation, meaning that our on-line complexity is $(n + m + s) \cdot \mathsf{poly}(\lambda)$, but otherwise independent of the run-time of the computation.

Alternately, we also get an instantiation where the on-line complexity is only $(n+m+d)\cdot\mathsf{poly}(\lambda)$, where $n$ is the input size, $m$ is the output size, and $d$ is the depth of the circuit, but is otherwise independent of the circuit's width $w$. In this case, we also incur a $2^{O(d)}$ security loss in our reduction, but this can be a significant improvement over the naive complexity-leveraging approach which incurs a $2^n$ security loss, where $n$ is the input size. In particular, in the case of $\mathsf{NC}^1$ circuits where $d = O(\log n)$, we get a polynomial reduction and achieve on-line complexity $(n+m)\cdot\mathsf{poly}(\lambda)$.

More broadly, we develop a connection between constructing adaptively secure schemes in our framework and a certain type of *pebble complexity* of the given circuit. The size of the garbled input is proportional to the maximal number of pebbles and the number of hybrids in our reduction is proportional to the number of moves needed to pebble the circuit.

## 3.4 Our Techniques

In order to explain our techniques, we must first explain the difficulties in proving the adaptive security of Yao's garbling schemes. Since these difficulties are subtle, we begin with a description of the scheme and the proof of selective security, following Lindell and Pinkas [LP09]. This allows us to fix a precise notation and terminology which will be needed to also explain our new construction and proof. We expect that the reader is already familiar with the basics of Yao circuits and refer to [LP09] for further details.

### 3.4.1 Yao's Scheme and The Challenge of Adaptive Security

**Yao's Scheme.** For each wire $w$ in the circuit, we pick two keys $k_w^0, k_w^1$ for a symmetric-key encryption scheme. For each gate in the circuit computing a function $g : \{0,1\}^2 \to \{0,1\}$ and having input wires $a, b$ and output wire $c$ we create a *garbled gate* consisting of 4 randomly ordered ciphertexts created as:

$$\begin{aligned} c_{0,0} &= \mathsf{Enc}_{k_a^0}(\mathsf{Enc}_{k_b^0}(k_c^{g(0,0)})) \qquad & c_{1,0} &= \mathsf{Enc}_{k_a^1}(\mathsf{Enc}_{k_b^0}(k_c^{g(1,0)})), \\ c_{0,1} &= \mathsf{Enc}_{k_a^0}(\mathsf{Enc}_{k_b^1}(k_c^{g(0,1)})) \qquad & c_{1,1} &= \mathsf{Enc}_{k_a^1}(\mathsf{Enc}_{k_b^1}(k_c^{g(1,1)})) \end{aligned} \tag{1}$$

where $(\mathsf{Enc}, \mathsf{Dec})$ is a CPA-secure encryption scheme. The garbled circuit $\widetilde{C}$ consists of all of the gabled gates, along with an *output mapping* $\{k_w^0 \to 0, k_w^1 \to 1\}$ which gives the correspondence between the keys and the bits they represent for each output wire $w$ . To garble an $n$-bit value $x = x_1 x_2 \cdots x_n$, the garbled input $\widetilde{x}$ consists of the keys $k_{w_i}^{x_i}$ for the $n$ input wires $w_i$.

To evaluate the garbled circuit on the garbled input, it's possible to decrypt (exactly) one ciphertext in each garbled gate and get the key $k_w^{v(w)}$ corresponding to the bit $v(w)$ going over the wire $w$ during the computation $C(x)$. Once the keys for the output wires are computed, it's possible to recover the actual output bits by looking them up in the output mapping.

[6] We consider circuits made up of fan-in 2 gates with arbitrary fan-out. The circuit is composed of levels and wires can only connect gates in level $i$ with those at the next level $i + 1$. The width of the circuit is the maximal number of gates in any level and the depth is the number of levels.

**Selective Security Simulator.** To prove the selective security of Yao's scheme, we need to define a simulator that gets the output $y = y_1 y_2 \cdots y_m = C(x)$ and must produce $\widetilde{C}$, $\widetilde{x}$. The simulator picks random keys $k_1^0, k_w^1$ for each wire $w$ just like the real scheme, but it creates the garbled gates as follows:

$$\begin{aligned} c_{0,0} &= \mathsf{Enc}_{k_a^0}(\mathsf{Enc}_{k_b^0}(k_c^0)) \qquad & c_{1,0} &= \mathsf{Enc}_{k_a^1}(\mathsf{Enc}_{k_b^0}(k_c^0)), \\ c_{0,1} &= \mathsf{Enc}_{k_a^0}(\mathsf{Enc}_{k_b^1}(k_c^0)) \qquad & c_{1,1} &= \mathsf{Enc}_{k_a^1}(\mathsf{Enc}_{k_b^1}(k_c^0)) \end{aligned} \tag{2}$$

where all four ciphertext encrypt the same key $k_c^0$. It creates the output mapping $\{k_w^0 \to y_w, k_w^1 \to 1 - y_w\}$ by "programming it" so that the key $k_w^0$ corresponds to the correct output bit $y_w$ for each output wire $w$. This defines the simulated garbled circuit $\widetilde{C}$. To create the simulated garbled input $\widetilde{x}$ the simulator simply gives out the keys $k_w^0$ for each input wire $w$. Note that, when evaluating the simulated garbled circuit on the simulated garbled input, the adversary only sees the keys $k_w^0$ for every wire $w$.

**Selective Security Hybrids.** To prove indistinguishability between the real world and the simulation, there is a series of carefully defined hybrid games that switch the distribution of one garbled gate at a time, starting with the input level and proceeding up the circuit level by level. In each step we switch the distribution of the ciphertexts in the targeted gate to:

$$\begin{aligned} c_{0,0} &= \mathsf{Enc}_{k_a^0}(\mathsf{Enc}_{k_b^0}(k_c^{v(c)})) \qquad & c_{1,0} &= \mathsf{Enc}_{k_a^1}(\mathsf{Enc}_{k_b^0}(k_c^{v(c)})), \\ c_{0,1} &= \mathsf{Enc}_{k_a^0}(\mathsf{Enc}_{k_b^1}(k_c^{v(c)})) \qquad & c_{1,1} &= \mathsf{Enc}_{k_a^1}(\mathsf{Enc}_{k_b^1}(k_c^{v(c)})) \end{aligned} \tag{3}$$

where $v(c)$ is the correct *value* of the bit going over the wire $c$ during the computation of $C(x)$.

Let us give names to the three modes for creating garbled gates that we defined above: (1) is called $\mathsf{RealGate}$ mode, (2) is called $\mathsf{SimGate}$ mode, and (3) is called $\mathsf{InputDepSimGate}$ mode, since the way that it is defined depends adaptively on the choice of the input $x$.

We can switch a gate from $\mathsf{RealGate}$ to $\mathsf{InputDepSimGate}$ mode if the gates in the previous level are in $\mathsf{InputDepSimGate}$ mode (or we are in the input level) by CPA security of ecryption. In particular, we are *not* changing the value contained in ciphertext $c_{v(a),v(b)}$ encrypted under the keys $k_a^{v(a)}, k_b^{v(b)}$ that the adversary obtains during evaluation, but we *can* change the values contained in all of the other ciphertexts since the keys $k^{1-v(a)}, k^{1-v(b)}$ do not appear anywhere inside the garbled gates in the previous level.

At the end of the above sequence of hybrid games, all gates are switched from $\mathsf{RealGate}$ to $\mathsf{InputDepSimGate}$ mode and the output mapping is computed as in the real world. The resulting distribution is *statistically identical* to the simulation where all the gates are in $\mathsf{SimGate}$ mode and the output mapping is programmed. This is because, at any level that's not the output, the keys $k_c^0, k_c^1$ are used completely identically in the subsequent level so there is no difference between always encrypting $k_c^{v(c)}$ ($\mathsf{InputDepSimGate}$) and $k_c^0$ ($\mathsf{SimGate}$). At the output level there is no difference between encrypting $k_c^{v(c)}$ and giving the real mapping $k_c^{v(c)} \to y_c$ or encrypting $k_c^0$ and giving the programmed mapping $k_c^0 \to y_c$ where $y_c$ is the output bit on wire $c$.

**Challenges in Achieving adaptive security.** There are two issues in using the above strategy in the adaptive setting: an immediate but easy to fix problem and a more subtle but difficult to overcome problem.

The first immediate issue is that the selective simulator needs to know the output $y = C(x)$ to create the garbled circuit $\widetilde{C}$ and in particular to program the output mapping $\{k_w^0 \to y_w, k_w^1 \to$

$1 - y_w\}$ for the output wires $w$. However, the adaptive simulator does not get the output $y$ until *after* it creates the garbled circuit $\widetilde{C}$. Therefore, we cannot (even syntactically) use the selective security simulator in the adaptive setting. This issue turns out to be easy to fix by modifying the construction to send the output-mapping as part of the garbled input $\widetilde{x}$ in the on-line phase, rather than as part of the garbled circuit $\widetilde{C}$ in the off-line phase. This modification raises on-line complexity to also being linear in the output size of the circuit, which we know to be necessary by the lower bound of [AIKW13]. With this modification, the adaptive simulator can program the output mapping after it learns the output $y = C(x)$ in the on-line phase and therefore we get a syntactically meaningful simulation strategy in the adaptive setting.

The second problem is where the true difficulty lies. Although we have a syntactically meaningful simulation strategy, the previous proof of indistinguishability of the real world and the simulation completely breaks down in the adaptive setting. Recall that the proof consisted of a sequence of hybrids where we changed one garbled gate at a time (starting from the input level) from RealGate mode to the InputDepSimGate mode. In the latter mode, the gate is created in a way that depends on the input $x$, but in the adaptive setting the input $x$ is chosen adaptively after the garbled circuit is created, leading to a circularity. In other words, the distribution of InputDepSimGate as specified in equation (3) doesn't even syntactically make sense in the adaptive setting. Therefore, *although we have a syntactically meaningful simulation strategy for the adaptive setting, we do not have any syntactically meaningful sequence of intermediate hybrids to prove indistinguishability between the real world and the simulated world.*

(One could hope to bypass InputDepSimGate mode altogether and define the hybrids by changing a gate directly from RealGate mode to SimGate mode. Unfortunately, this change is easily distinguishable already for the very first gate we would hope to change at the input level – the output value on the gate would no longer be $v(w)$ but 0 which may result in an overall incorrect output since we have not programmed the output map yet. On the other hand, we cannot immediately jump to a hybrid where we program the output map since all of the keys and their semantics are contained under encryption in prior levels of the circuit and we haven't argued about the security of the ciphertexts in these levels yet.)

### 3.4.2 Our Solution

**Outer Encryption Layer.** Our construction starts with the approach of [BHR12] which is to encrypt the entire garbled circuit with an additional outer encryption layer in the off-line phase (this is unrelated to the encryption used to construct the garbled gates). Then, in the on-line phase, we give out the secret key for this outer encryption scheme. The approach of [BHR12] required a symmetric-key, one-time encryption scheme which is *equivocal*, meaning that the ciphertext doesn't determine the message and it is possible to come up with a secret key that can open the ciphertext to any possible message. Unfortunately, any fully equivocal encryption scheme where a ciphertext can be opened to any message (e.g., the one-time pad) must necessarily have a secret key size which is as large as the message size. In our case, this is the entire garbled circuit and therefore this ruins the on-line efficiency of the scheme. Our main idea is to use a new type of *partially* equivocal encryption scheme, which we call *somewhere equivocal*.

**Somewhere Equivocal Encryption.** Intuitively, a somewhere equivocal encryption scheme allows us to create a simulated ciphertext which contains "holes" in some small subset of the message bit positions $I$ chosen by the simulator, but all other message bits are fixed. The simulator can later equivocate this ciphertext and "plug the holes" with any bits it wants by deriving a corresponding secret key. An adversary cannot distinguish between seeing a real encryption of

some message $m = m_1 m_2 \cdots m_n$ and the real secret key, from seeing a simulated encryption created using only $(m_i)_{i \notin I}$ with "holes" in positions $I$ and an equivocated secret key that later plugs the holes to the correct bits $(m_i)_{i \in I}$. We show how to construct somewhere equivocal encryption using one-way functions. The size of the secret key is only proportional to the maximum number of holes $t = |I|$ that we allow, which we call the "equivocation parameter", but can be much smaller than the message size.[7]

Our proof of security departs significantly from that of [BHR12]. In particular, our simulator does *not* take advantage of the equivocation property of the encryption scheme at all, and in fact, our simulation strategy is identical to the adaptive simulator we outlined above for the variant of Yao's garbling where the output map is sent in the on-line phase. However, we crucially rely on the equivocation property to carefully define a meaningful sequence of hybrids that allows us to prove the indistinguishability of the real and simulated worlds.

**Hybrids for adaptive security.** We define hybrid distributions where various garbled gates will be created in one of three modes discussed above: RealGate, SimGate and InputDepSimGate. However, to make the last option meaningful (even syntactically) in the adaptive setting, we rely on the somewhere equivocal encryption scheme. For these hybrids, when we create the encrypted garbled circuit in the off-line phase, we will simulate the outer encryption layer with a ciphertext that contains "holes" in place of all gates that are in InputDepSimGate mode. Only when we open the outer encryption in the on-line phase after the input $x$ is chosen, we will "plug the holes" by sampling these gates correctly in InputDepSimGate mode in a way that depends on the input $x$. Our equivocation parameter $t$ for the somewhere equivocal encryption scheme therefore needs to be large enough to support the maximum number of gates in InputDepSimGate mode that we will have in any hybrid.

**Sequence of hybrids.** For our main result, we use the following sequence of hybrids to prove indisitnguishability of real and simulated worlds. We start by switching the first two levels of gates (starting with the input level) to InputDepSimGate mode. We then switch the first level of gates to SimGate mode and switch the third level InputDepSimGate mode. We continue this process, where in each step $i$ we maintain level $i$ in InputDepSimGate mode but switch the previous level $i-1$ from InputDepSimGate to SimGate and then switch the next level $i+1$ from RealGate to InputDepSimGate. Eventually all gates will be in SimGate mode as we wanted. We can switch a level $i-1$ from InputDepSimGate to SimGate mode when the subsequent level $i$ is in InputDepSimGate mode since the keys $k_c^0, k_c^1$ for wires $c$ crossing from level $i-1$ to $i$ are used identically in level $i$ and therefore there is statistically no difference between encrypting the key $k_c^{v(c)}$ (InputDepSimGate) and $k_c^0$ (SimGate). We can also switch a level $i+1$ from RealGate to InputDepSimGate when the previous level $i$ is InputDepSimGate (or $i+1$ is the input level) by CPA security following the same argument as in the selective setting. With this strategy, at any point in time we have at most two levels in InputDepSimGate mode and therefore our equivocation parameter only needs to be proportional to the circuit width $w$.

**Connection to pebbling.** We can generalize the above idea and get other meaningful sequences of hybrids with different parameters and implications. We can think of the process of switching between RealGate, SimGate and InputDepSimGate modes as a new kind of *graph pebbling game*, where

[7] A different notion of partially equivocal encryption, called *somewhat non-committing* encryption, was introduced in [GWZ09]. The latter notion allows a ciphertext to be opened to some small, polynomial size, set of messages which can be chosen arbitrarily by the simulator at encryption time. The two notions are incomparable.

pebbles can be placed on the graph representing the circuit according to certain rules. Initially, all gates are in RealGate mode, which we associate with *not having any pebble* on them. We associate InputDepSimGate mode with having a *black pebble* and SimGate mode with having a *gray pebble*. The rules of the game go as follows:

- We can place (or remove) a black pebble on any input gate at any time.
- If both predecessors of a gate have black pebbles on them, we can place (or remove) a black pebble on that gate.
- If a gate has a black pebble on it and all successors of that gate have black or gray pebbles on them (or it is an output gate), we can change the pebble on that gate from black to a gray.

The goal of the game is to end up with a gray pebble on every gate. Any such pebbling strategy leads to a sequence of hybrids that shows the indistinguishability between the real world and the simulation. The number of steps needed to complete the pebbling corresponds to the number of hybrids in our proof, and therefore the security loss of our reduction. The maximum number of black pebbles that are in play at any given time corresponds to the equivocation parameter needed for our somewhere equivocal encryption scheme.

For example, the sequence of hybrids discussed above corresponds to a pebbling strategy where the number of black pebbles used is linear in the circuit width $w$ (but independent of the depth) and the number of steps is linear in the circuit size. We give an alternate recursive pebbling strategy where the number of black pebbles used is linear in the circuit depth $d$ (but independent of the width) and the number of steps is $2^{O(d)}$ times the circuit size.

# 4 Problem: Generalized Selective Decryption Game

## 4.1 Background

In the GSD game we consider a symmetric encryption scheme $\mathsf{Enc}$ and a parameter $n \in \mathbb{N}$. Initially, we sample $n$ random keys $k_1, \ldots, k_n$ and a bit $b \in \{0,1\}$. During the game the adversary $\mathcal{A}$ can make two types of queries. Encryption query: on input $(i,j)$ she gets $c = \mathsf{Enc}_{k_i}(k_j)$; corruption query: on input $i$, she gets $k_i$. At some point, $\mathcal{A}$ chooses some $i$ to be challenged on. If $b = 0$, she gets the key $k_i$; if $b = 1$, she gets a uniformly random $r_i$. Finally, $\mathcal{A}$ outputs a guess bit $b'$. The goal is prove that for any efficient $\mathcal{A}$, $|\Pr[b = b'] - 1/2|$ is negligible (or, equivalently, $k_i$ is pseudorandom) assuming only that $\mathsf{Enc}$ is a secure encryption scheme. We only allow one challenge query, but this notion is equivalent to allowing any number of challenge queries by a standard hybrid argument (losing a factor that is only the number of challenge queries).

Restrictions. It is convenient to think of the GSD game as dynamically building a graph, which we call key graph. Initially, we have a graph with $n$ vertices labeled $1, \ldots, n$, where we associate vertex $i$ with key $k_i$. On an encryption query $\mathsf{Enc}_{k_i}(k_j)$ we add a directed edge $i \to j$. On a corruption query $i$ we label the vertex $i$ as corrupted. Note that if $i$ is corrupted then $\mathcal{A}$ also learns all keys $k_j$ for $j$ where there's a path from $i$ to $j$ in the key graph by simply decrypting the keys along the path. To make the game non-trivial, we therefore only allow challenge queries for keys that are not reachable from any corrupted key. Another restriction we must make is to disallow encryption cycles, i.e., loops in the graph. The reason is simply that we cannot hope to prove security when allowing cycles assuming only a standard security notion (in our case CPA) of the underlying encryption scheme. This would require circular (or key-dependent-message) security [BRS03], which is stronger than CPA [ABBC10]. Finally, we require that the challenge query is a

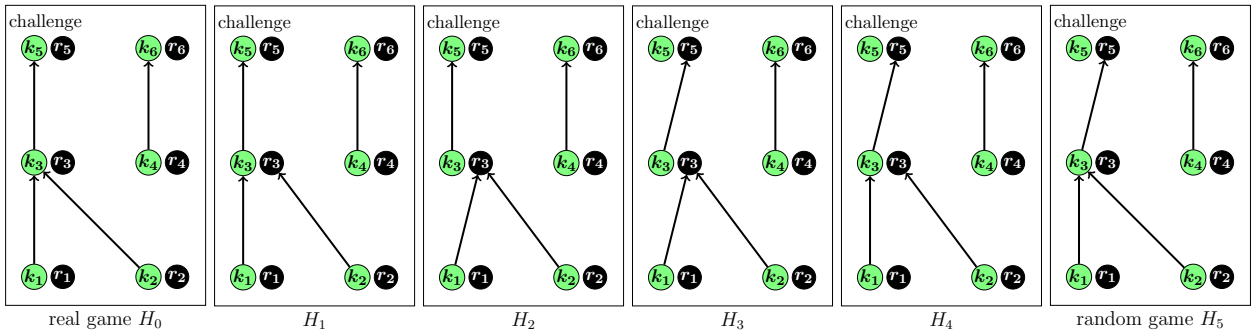


Figure 1: Illustration of the selective security game. The green nodes correspond to keys, the dark nodes are random values. The adversary commits to encryption queries $(1,3),(2,3),(3,5)$ and challenge 5 (there's also an encryption query $(4,6)$, but it's not in the connected component containing the challenge, and thus not relevant for defining the hybrids. The adversary can also corrupt keys 4 and 6 as they are outside of the component containing the challenge). The hybrid $H_0$ corresponds to the real game, the hybrid $H_5$ is the random game, where instead of an encryption of the challenge key $\mathsf{Enc}_{k_3}(k_5)$, the adversary gets an encryption of the random value $\mathsf{Enc}_{k_3}(r_5)$. If an adversary $\mathcal{A}$ can distinguish any two consecutive hybrids $H_i$ and $H_{i+1}$ with some advantage $\delta$, we can use this $\mathcal{A}$ to construct $\mathcal{B}$ which breaks the CPA security of $\mathsf{Enc}$ with the same advantage $\delta$: E.g., assume $\mathcal{B}$ is given an CPA challenge $c = \mathsf{Enc}_k(z)$ where $z$ is one of two messages (which we'll call $k_5$ and $r_5$). Now $\mathcal{B}$ can simulate the game $H_2$ for $\mathcal{A}$, but when $\mathcal{A}$ makes the encryption query $(3,5)$ we answer with $z$. If $z = k_5$, then we simulate the game $H_2$, but if $z = r_5$ we simulate the game $H_3$. (Note that $\mathcal{B}$ can simulate the games because $k_3$, which in the simulation is $B$'s challenger's key, is not used anywhere else.) Thus, $\mathcal{B}$ has the same advantage in the CPA game, as $\mathcal{A}$ has in distinguishing $H_3$ from $H_4$.

leaf in the graph; this restriction too is necessary unless we make additional assumptions on the underlying encryption scheme.

**Selective security of GSD.** In order to prove security of the GSD game, one must show how to turn an adversary $\mathcal{A}$ who breaks the GSD game with some advantage $\varepsilon = |\Pr[b = b'] - 1/2|$ into an adversary $\mathcal{B}$ who breaks the security of $\mathsf{Enc}$ with some advantage $\varepsilon' = \varepsilon'(\varepsilon)$. The security notion we consider is the standard notion of indistinguishability under chosen plaintext attacks (CPA). Recall that in the CPA security game an adversary $\mathcal{B}$ is given access to an encryption oracle $\mathsf{Enc}_k(.)$. At some point $\mathcal{B}$ chooses a pair of messages $(m_0, m_1)$, then gets a challenge ciphertext $c = \mathsf{Enc}_k(m_b)$ for a random bit $b$, and finally must output a guess $b'$. The advantage of $\mathcal{B}$ is $|\Pr[b = b'] - 1/2|$.

It's not at all clear how to construct an adversary $\mathcal{B}$ that breaks CPA from an $\mathcal{A}$ who breaks GSD. This problem becomes much easier if we assume that $\mathcal{A}$ breaks the *selective* security of GSD, where $\mathcal{A}$ must choose all its encryption, corruption and challenge queries before the experiment starts.

In fact, it is sufficient to know the topology of the connected component in the key graph that contains the challenge node. Let $\alpha$ denote the number of edges in this component. One can now define a sequence of $2\alpha$ hybrid games $H_0, \ldots, H_{2\alpha-1}$, where the first game is the real game (i.e., the GSD game with $b = 0$ where the adversary gets the key), the last hybrid is the random game ($b = 1$), and moreover, from any adversary who distinguishes $H_i$ from $H_{i+1}$ with some advantage $\varepsilon'$, we get an adversary against the CPA security of $\mathsf{Enc}$ with the same advantage. In particular, if we have an $\mathcal{A}$ with advantage $\varepsilon$ against GSD, we can break the CPA security with advantage

$\varepsilon' \geq \varepsilon/(2\alpha - 1) \geq \varepsilon/n^2$ (as an $n$ vertex graph has $\leq n^2$ edges).

**Adaptive security of GSD.** In the selective security proof for GSD we crucially relied on the fact that we knew the topology of the underlying key graph. A generic trick to prove adaptive security is "complexity leveraging", where one simply turns an adaptive adversary into a selective one by initially guessing the relevant choices to be made by the adaptive adversary and committing to those (as required by the selective security game). If during the security game the adaptive choices by the adversary disagree with our initial guess, we simply abort. The problem with this approach is that assuming the adaptive adversary has advantage $\varepsilon$, the selective adversary we get via complexity leveraging only has advantage $\varepsilon/d$ where $1/d$ is the probability that our guess is correct, and typically is exponentially small. Concretely, in the GSD game, we need to guess the nodes in the connected component containing the challenge, and the topology of the connected component. As the number of such choices is exponential in the number of keys $n$, this probability is $2^{-\Theta(n)}$.

### 4.2 Our Results

The main result of our work shows that GSD restricted to trees can be proven secure with only a quasi-polynomial loss

$$\varepsilon' = \varepsilon \cdot n^{3\log(n)+5} \ .$$

Our bound is actually even stronger as we don't need the entire key graph to be a tree, it is sufficient that the subgraph containing only the nodes from which the challenge node can be reached is a tree (when ignoring edge directions).

The bound above is derived from a more fine-grained bound: assuming that the longest path in the key graph is of length $\ell$, the in-degree of every node is at most $d$ and the challenge node can be reached from at most $s$ sources (i.e., nodes with in-degree 0) we get (note that $\ell, d$ and $s$ are at most $n$, and the above bound was derived setting $\ell = d = s = n$)

$$\varepsilon' = \varepsilon \cdot dn((2d+1)n)^{\lceil \log s \rceil} \, (3n)^{\lceil \log \ell \rceil} \ .$$

For more details, see [FJP15]. In the next section we give an overview of our technique in a simplified version of GSD.

### 4.3 Our Technique

Consider a version of the GSD game where the adversary is restricted to create a key graph with following properties[8]. First, it is a directed tree. Second it has only one source node. A source is a node with no ingoing edges. This implies that the indegree of every node in the tree is at most 1. Also assume the attacker decides which key is the challenge key at the beginning of the game. If we follow the naive approach, we can guess where each key (node) is on the path from the source to the challenge. Then assuming the adaptive attacker has advantage $\varepsilon_a$, we can show that the selective attacker has advantage $\varepsilon_s \geq \varepsilon_a/n!$. Furthermore we saw how we can break CPA security of the encryption scheme with advantage $\varepsilon_{cpa} \geq \varepsilon_s/(2n-3)$ – since the number of edges in the path is at most $n-1$. Combining the two we have, $\varepsilon_{cpa} \geq \varepsilon_a/n!(2n-3)$. In our approach, we abandon the reduction from an adaptive attacker to a selective attacker and by doing so we are free of the burden of guessing the adaptive attacker's every step. Although not entirely! We still need to break the CPA security using the adaptive attacker. We identify what we need from two hybrid games (say

[8]These properties hold for the connected component which includes the challenge node

**Game**, **Game**′) to be able to do just that. Imagine a series of variants of the GSD game. These games are similar to GSD in every sense, except that some responses to adversary's encryption queries are fake. This means the adversary receives $\mathsf{Enc}_{k_x}(r_y)$ instead of $\mathsf{Enc}_{k_x}(k_y)$, when he queries for $\mathsf{encrypt}(x, y)$, where $r_y$ is uniformly random and independent of other keys in the game. We are looking for two games, such that if an adversary $\mathcal{A}$ can distinguish between the two, we can use $\mathcal{A}$ to construct attacker $\mathcal{B}$ that will break the CPA security of the encryption scheme. For this, the two games should only differ in the response of one encryption query on the path to the challenge, say $\mathsf{encrypt}(y, z)$, which is responded to with a real ciphertext $\mathsf{Enc}_{k_y}(k_z)$ in **Game** and with a fake ciphertext $\mathsf{Enc}_{k_y}(r_z)$ in **Game**′. Moreover, the key $k_y$ must not be encrypted anywhere else in the game, as attacher $\mathcal{B}$ will set the unknown key of its CPA challenger $\mathcal{C}$ to be $k_y$. Thus, in **Game** and **Game**′ all queries $\mathsf{encrypt}(x, y)$, for any $x$, are responded to with a fake ciphertext $\mathsf{Enc}_{k_x}(r_y)$.

Summing up, for some $y$ we need the two games to have the following properties:

- Property 1. **Game** and **Game**′ are identical except for the response to one query $\mathsf{encrypt}(y, z)$, which is replied to with a real ciphertext in **Game** and a fake one in **Game**′.
- Property 2. Queries $\mathsf{encrypt}(x, y)$ are replied to with a fake response in both games.

We need to show that if there is an attacker $\mathcal{D}$, who can distinguish between the real and random GSD game, then there exists $\mathcal{A}$ who can distinguish two games which satisfy the properties of **Game** and **Game**′. We cannot define such hybrid games before we know what the graph looks like and where the relevant keys are placed on the graph. We need to know when to respond with fake encryptions. We do this with guessing the relevant nodes alone.

Start with assuming GSD on $n$ keys is $\varepsilon$ distinguishable. we use the notation $GSD(b, n)$ for GSD game on $n$ keys with b=0 for the real game, and $b = 1$ for the random game. We define two new hybrid games, $\mathbf{Game}(0, n), \mathbf{Game}(1, n)$, $\mathbf{Game}(b, n)$ is the same as $GSD(b, n)$ but we first guess the node which divides the path to the challenge in half, $w$ and respond to the query $\mathsf{encrypt}(*, w)$ with $\mathsf{Enc}_{k_*}(r_w)$. We show,

$$\Delta(GSD(0,n), GSD(1,n))/n \leq$$
$$\Delta\left(GSD(0,n), \mathbf{Game}(0,n)\right) + \Delta(\mathbf{Game}(0,n), \mathbf{Game}(1,n)) + \Delta(\mathbf{Game}(1,n), GSD(1,n))$$

Therefore at least one of the above three pair of games is $\varepsilon/3n$ distinguishable. Now consider the portion of the game which is different in each pair. For example, $GSD(0, n)$ and $\mathbf{Game}(0, n)$ are the same for the second half of the path to the challenge. But the first half is different. One ends with a fake response and one with a real response, just like the original games did with respect to the responses to the challenge query. In other words we have an instance of the GSD on $n/2$ keys. The same holds with the other two pairs. Next we repeat the same steps with the two $GSD(b, n/2)$ games. Finally after at most $\log n$ iterations, we have two games $GSD(0, 1)$ and $GSD(1, 1)$, i.e. the length of the path between the source and the challenge is only one. Let's call this edge $(y, z)$. $y$ is the source, this means $k_y$ is not used anywhere else in the game. Since the two games are only different in the response to query to $\mathsf{encrypt}(y, z)$ (one returns $\mathsf{Enc}_{k_y}(k_z)$ and the other $\mathsf{Enc}_{k_y}(r_z)$), these games satisfy the properties we were looking for in **Game** and **Game**′. The proof for general trees is more complicated but follows the same recursive approach.

# 5 Proposal

**Enhancing the two techniques.** We propose studying each technique to see whether it can support some features of the other. For example, is it possible to connect the recursive approach of

defining hybrid games in GSD solution, to pebbling games? if so, can this enable us to apply the nested hybrids to general graphs? Can we use Somewhere Equivocal Encryption in construction of Adaptively secure GSD protocols? Is it possible to use the GSD approach, to prove adaptive security of the Yao's garbling scheme?

**Adaptive security of Yao's Garbled Circuits.** Yao's garbled gate, is constructed using two layers of CPA encryption. The garbled circuit is a set of ciphertexts of keys encrypted under other keys, similar to the graph in the GSD game. The difference is, here each ciphertext is the result of two encryptions as opposed to one encryption in the case of GSDG. Furthermore in both cases, the adversary decides how the ciphertexts are created: in GSDG, by choosing every encrypt query, and in garbled circuit, by choosing the circuit. Also the adaptive attacker of the garbled circuit, can choose to see some of the keys in the graph by choosing his input. This is similar to corrupt queries that the attacker of the GSD game can make. And in both cases we need to show that the adversary does not learn anything about the keys he did not learn trivially. Given the similarities we propose using the GSD approach to prove adaptive security of Yao's garbled circuits. We hypothesis, the GSD approach can be generalized to support the double encryption.

**Proposed timetable.**

| | |
|---|---|
| Feb–March, 2016 | Proposal presentation |
| March–May 2016 | Work on open problems outlined in the Proposal statement |
| June–Aug 2016 | Consider any manageable open questions that come up while writing the final thesis. |
| Aug 2016 | Tentative defense date. |

# References


[ABBC10] Tolga Acar, Mira Belenkiy, Mihir Bellare, and David Cash. Cryptographic agility and its relation to circular encryption. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 403–422. Springer, Heidelberg, May 2010.

[AIK04] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in $NC^0$. In *45th FOCS*, pages 166–175. IEEE Computer Society Press, October 2004.

[AIK05] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Computationally private randomizing polynomials and their applications. In *20th Annual IEEE Conference on Computational Complexity (CCC 2005), 11-15 June 2005, San Jose, CA, USA*, pages 260–274. IEEE Computer Society, 2005.

[AIK10] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. From secrecy to soundness: Efficient verification via secure computation. In Samson Abramsky, Cyril Gavoille, Claude Kirchner, Friedhelm Meyer auf der Heide, and Paul G. Spirakis, editors, *ICALP 2010, Part I*, volume 6198 of *LNCS*, pages 152–163. Springer, Heidelberg, July 2010.

[AIKW13] Benny Applebaum, Yuval Ishai, Eyal Kushilevitz, and Brent Waters. Encoding functions with constant online rate or how to compress garbled circuits keys. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 166–184. Springer, Heidelberg, August 2013.

[BHR12] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Adaptively secure garbling with applications to one-time programs and secure outsourcing. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 134–153. Springer, Heidelberg, December 2012.

[BRS03] John Black, Phillip Rogaway, and Thomas Shrimpton. Encryption-scheme security in the presence of key-dependent messages. In Kaisa Nyberg and Howard M. Heys, editors, *SAC 2002*, volume 2595 of *LNCS*, pages 62–75. Springer, Heidelberg, August 2003.

[FJP15] Georg Fuchsbauer, Zahra Jafargholi, and Krzysztof Pietrzak. A quasipolynomial reduction for generalized selective decryption on trees. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, pages 601–620, 2015.

[GGP10] Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 465–482. Springer, Heidelberg, August 2010.

[GIS+10] Vipul Goyal, Yuval Ishai, Amit Sahai, Ramarathnam Venkatesan, and Akshay Wadia. Founding cryptography on tamper-proof hardware tokens. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 308–326. Springer, Heidelberg, February 2010.

[GKP+13] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 555–564. ACM Press, June 2013.

[GKR08] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. One-time programs. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 39–56. Springer, Heidelberg, August 2008.

[GVW12] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 162–179. Springer, Heidelberg, August 2012.

[GWZ09] Juan A. Garay, Daniel Wichs, and Hong-Sheng Zhou. Somewhat non-committing encryption and efficient adaptively secure oblivious transfer. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 505–523. Springer, Heidelberg, August 2009.

[HJO+15] Brett Hemenway, Zahra Jafargholi, Rafail Ostrovsky, Alessandra Scafuro, and Daniel Wichs. Adaptively secure garbled circuits from one-way functions. Cryptology ePrint Archive, Report 2015/1250, 2015. `https://eprint.iacr.org/2015/1250`.

[LP09] Yehuda Lindell and Benny Pinkas. A proof of security of Yao's protocol for two-party computation. *Journal of Cryptology*, 22(2):161–188, April 2009.

[MP06] Daniele Micciancio and Saurabh Panjwani. Corrupting one vs. corrupting many: The case of broadcast and multicast encryption. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *ICALP 2006, Part II*, volume 4052 of *LNCS*, pages 70–82. Springer, Heidelberg, July 2006.

[Pan07] Saurabh Panjwani. Tackling adaptive corruptions in multicast encryption protocols. In Salil P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 21–40. Springer, Heidelberg, February 2007.

[SS10] Amit Sahai and Hakan Seyalioglu. Worry-free encryption: functional encryption with public keys. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *ACM CCS 10*, pages 463–472. ACM Press, October 2010.

[WGL00] Chung Kei Wong, Mohamed Gouda, and Simon S. Lam. Secure group communications using key graphs. *IEEE/ACM Transactions on Networking*, 8(1):16–30, February 2000.

[Yao82] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *23rd FOCS*, pages 160–164. IEEE Computer Society Press, November 1982.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986.